

# Integrated Framework of Web-based Urban Simulation Support System for Communities and Cities

Maher Albettar

A Thesis  
in  
The Department  
of  
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements  
for the Degree of Master of Applied Science (Electrical and Computer Engineering) at  
Concordia University  
Montreal, Quebec, Canada

November 2022

© Maher Albettar, 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Maher Albettar

Entitled: Integrated Framework of Web-based Urban Simulation Support System for Communities and Cities

and submitted in partial fulfillment of the requirements for the degree of

**Master of Applied Science (Electrical and Computer Engineering)**

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____	Chair
Dr. Chunyan Lai	
_____	Examiner
Dr. Andreas Athienitis	
_____	Examiner
Dr. Chunyan Lai	
_____	Thesis Supervisor(s)
Dr. Yong Zeng	
_____	Thesis Supervisor(s)
Dr. Liangzhu Wang	

Approved by:

\_\_\_\_\_  
Dr. Yousef R. Shayan, Chair of Department or Graduate Program Director

\_\_\_\_\_  
Dr. Mourad Debbabi      Dean the Gina Cody School of Engineering and  
Computer Science

## **ABSTRACT**

### **Integrated Framework of Web-based Urban Simulation Support System for Communities and Cities**

**Maher Albettar**

One of the most important agendas that urban planners and researchers face in the coming decades is to establish new designs that improve the sustainability and resilience of cities. Under the rapid development of Geographic Information System (GIS) technology and the Internet of Things (IoT), these technologies empower urban planners to enhance visibility into data and monitor fluctuations over time, evaluating the feasibility of proposed projects and predicting the effects on the environment, providing a better understanding a city as a multi-scale and multilayer complex system, scenario-testing, and strategic planning, collecting important aggregated data regarding building construction, energy consumption, and occupant wellbeings. However, many of these technologies generate vast amounts of data on some levels that are not detailed enough and are available at different scales, in various formats, and structured and unstructured forms. Usually, urban planners require a large amount of complex data to perform systematic dynamic simulations of many buildings. This adds difficulties to urban planners regarding data aggregation and real-time data management. This leads to an integrative solution for solving offline and online data processing and visualizing tasks and integrating data normalization and filtering techniques. Such solutions are needed to provide researchers with an integrative framework to reduce complexity and improve availability, accuracy, diversity, scalability, and integration efficiency. In this thesis, by analyzing the problems encountered and related requirements, the study leveraged the Niagara IoT framework and GIS integrations to build an integrated framework. The thesis work developed several modules for data preparation, creation, visualization, and integration. These modules simplify the data integration process and make it easier to prepare these data. The visualization and data integration requirements can be simplified with the help of GIS and an easy-to-use integrated framework to provide a real-time sensing system, geographic information system, and database integration system.

# Acknowledgments

All praise be to Allah, the Almighty, for having made everything possible by giving me the strength and courage to do this work.

I wish to express gratitude to my supervisor prof. Liangzhu Wang and Prof. Yong Zeng for their enthusiastic, support, and timely guidance during the long research process, and their assistance, comments, and encouragement will always be greatly appreciated.

I am thankful to the members of the Department of Electrical and Computer Engineering at Concordia University for their valuable assistance and support.

I would like to thank Dr. Khaldoun Albitar for his valuable comments through his professional counsel and guidance; I have acquired a wealth of knowledge previously unknown to me. Especially, I would like to thank him for his patience with me during my study period and his tireless efforts to correct me and guide me through this journey. My profound gratitude goes to my lab partners for their assistance in translating, research, and even moral support. They provided a very comfortable and welcoming learning environment for me. I was fortunate to meet and work with Dr. Danlin Hou, Shujie Yan, Dongxue Zhan, Jiwei Zou Senwen Yang, and Eslam Ali. It was a pleasure to learn from them.

Last but not least a heartfelt thank you to my family back home for their prayers and immense support, it has been a guiding light for me to this day. I am also grateful for all the support, love, and encouragement from my mother and father, my brother, sisters, and friends have all provided overwhelming support and encouragement to finish this thesis and I am deeply thankful for them as well.

To my mom, I will be forever grateful for the many sacrifices she has made so that this goal was attained.

It is not possible in this limited piece of paper to list all the names of those people who helped or supported me to get my Master's degree. To those whom I did not mention their names, deep in my heart, I am grateful to all.



# Table of Contents

<b>List of Figures</b> .....	<b>vii</b>
<b>List of Tables</b> .....	<b>xi</b>
<b>1. Introduction</b> .....	<b>1</b>
<b>1.1 Motivations</b> .....	<b>1</b>
<b>1.2 Research Problem and Objective</b> .....	<b>2</b>
<b>1.3 Thesis Structure</b> .....	<b>3</b>
<b>2. Literature Review</b> .....	<b>5</b>
<b>2.1 Review on Major Components</b> .....	<b>5</b>
2.1.1 Internet of Things (IoT).....	5
2.1.2 Geospatial Information System (GIS).....	8
2.1.3 Urban Simulations.....	11
<b>2.2 Review on Major Frameworks</b> .....	<b>14</b>
2.2.1 IoT Niagara Framework .....	14
2.2.2 Vuejs Framework .....	16
2.2.3 CMC Weather Forecasting Framework. ....	17
<b>2.3 Summary</b> .....	<b>19</b>
<b>3. System Software Design</b> .....	<b>20</b>
<b>3.1 System Architecture</b> .....	<b>20</b>
<b>3.2 Integrated Framework</b> .....	<b>21</b>
3.2.1 Dashboard Design .....	23
3.2.2 Data Collection Processing .....	26
3.2.3 Urban Modeling Data Input Generation Support Module .....	40
3.2.4 Urban Modeling Visualization Environment Support Modules .....	43
3.2.5 Application Module - City Reduced Probability of Infection (CityRPI) .....	50
3.2.6 Application Module - Fatima-CFD Workflow and Design Architecture .....	57
<b>3.3 Real-time Weather Station Data Integration</b> .....	<b>64</b>
<b>3.4 Canadian Metrological Centre Weather Forecasting Framework Integration</b> .....	<b>67</b>
<b>3.5 Automated Forecasting System Integration</b> .....	<b>69</b>

3.5.1	Preparing Weather Input Data Process.....	73
3.5.2	Synchronizing Input Data with Supercomputer Process.....	75
3.5.3	Synchronizing the Output with the Online Server Process:.....	77
3.5.4	Sending Notifying Emails Process.....	79
<b>3.6</b>	<b>Application Programming Interface (API) Integration Module .....</b>	<b>80</b>
<b>4.</b>	<b>Performance Evaluation .....</b>	<b>84</b>
<b>4.1</b>	<b>Data Input Generation Speed Performance .....</b>	<b>84</b>
<b>4.2</b>	<b>City-Scale Visualization Performance.....</b>	<b>86</b>
<b>4.3</b>	<b>Scalability Evaluation Summary .....</b>	<b>87</b>
4.3.1	System Functions Scalability, from Development Viewpoint.....	87
4.3.2	Geographic Usages and Applications Scalability the Use Viewpoint .....	89
<b>5.</b>	<b>Conclusion and Future Work.....</b>	<b>90</b>
<b>5.1</b>	<b>Conclusion.....</b>	<b>90</b>
<b>5.2</b>	<b>Contributions.....</b>	<b>91</b>
<b>5.1</b>	<b>Future Work and limitations .....</b>	<b>92</b>
<b>REFERENCES</b>	<b>.....</b>	<b>93</b>

# List of Figures

Figure 2-1 Architecture of Internet of Things .....	7
Figure 2-2 GIS Components .....	8
Figure 2-3 GIS Application in Urban planning [12] .....	10
Figure 2-4 GIS and Urban Planning [12] .....	11
Figure 2-5 Preprocessing and Data Collection Generation in CityFFD Simulation.[22] .....	14
Figure 2-6 Integration Architecture for Niagara Framework [25]. .....	16
Figure 2-7 Vue Framework Components [28] .....	17
Figure 2-8 CMC Weather Forecasting Framework Supported Continental Domains [29]. .....	18
Figure 3-1 System Architecture Design .....	21
Figure 3-2 Niagara Software Subsystems .....	22
Figure 3-3 Dashboard Modules Layers .....	23
Figure 3-4 Dashboard- Supported Map Style Systems.....	24
Figure 3-5 Map Styles .....	24
Figure 3-6 Dashboard- Files Conversion Modules .....	25
Figure 3-7 Supported Layers System .....	25
Figure 3-8 Data Integration- Default Data Sources .....	26
Figure 3-9 Sample of GeoJSON Structure [32] .....	27
Figure 3-10 Example of GeoJSON Supported Geometries [32].....	28
Figure 3-11 Converting Shp to GeoJSON UI. ....	29
Figure 3-12 Default SHP Data Source Building Attributes .....	30
Figure 3-13 Overall SHP Data Source Buildings Layout. ....	30
Figure 3-14 Example of 12 Nodes for a Building Representation OSM File .....	31
Figure 3-15 Example of Valid Polygons [36] .....	32
Figure 3-16 Example of Invalid Polygons [36].....	32
Figure 3-17 Example for 12 Points Building Layout.....	32
Figure 3-18 Example for 12 Points after Conversion to GeoJSON Data Structure. ....	32
Figure 3-19 Converting OSM to GeoJSON UI.....	33
Figure 3-20 Example of Turfjs Polygons Intersection .....	33

Figure 3-21 Mind Map of Supported Conversion Extensions .....	34
Figure 3-22 Converting GeoJSON to STL Workflow .....	36
Figure 3-23 Merging Phases of the Terrain Spatial Data.....	37
Figure 3-24 Integrating the Terrain Spatial Data with 3D Model STL Workflow .....	38
Figure 3-25 3D Model Coordinate Plane .....	39
Figure 3-26 Customize the 3D Model Coordination Origin .....	39
Figure 3-27 Pre-defined Backup Sectional Data Collection. ....	39
Figure 3-28 Dashboard, BES, and CFD Data Collection Generation Module UI .....	41
Figure 3-29 Dashboard, Customize the Bounding Box Selection UI. ....	41
Figure 3-30 Example of Two Custom Selection Points (Top Left, Bottom Right) .....	41
Figure 3-31 Dashboard, Customize the Building Information UI. ....	42
Figure 3-32 Example of Building Information Data Collection Result .....	42
Figure 3-33 Dashboard, Supported Legends Colors. ....	43
Figure 3-34 Example of BES Result for 12500 Buildings.....	44
Figure 3-35 BES Data Output Collection File Structure.....	45
Figure 3-36 BES Visualization Control Module UI.....	45
Figure 3-37 BES Visualization Module, Variable, and Timestamp. ....	46
Figure 3-38 BES Legend.....	46
Figure 3-39 BES- Trend Variables Analysis.....	46
Figure 3-40 City-scale Interactive 3D Map BES Data Collection Output Visualization.....	47
Figure 3-41 City-scale Interactive 2D Map BES Data Collection Output Visualization.....	47
Figure 3-42 VTK Standard Data Structure, Mesh Size (202, 73, 206).....	48
Figure 3-43 Example of CFD Simulation Result for Mesh Size (202, 73, 206).....	48
Figure 3-44 Dashboard CFD Visualization Module UI .....	49
Figure 3-45 CFD Visualization XZ side .....	49
Figure 3-46 CFD Visualization XY side.....	49
Figure 3-47 Application Module - CityRPI Dashboard .....	51
Figure 3-48 Third-Party API and Map-based Data Integration Workflow .....	52
Figure 3-49 Quebec Province GeoJSON Layout .....	53
Figure 3-50 Example of a Dictionary Creation. ....	53
Figure 3-51 The Geometric Layout of North America .....	54

Figure 3-52 Map Provider Support the Value-Color Transformation.....	55
Figure 3-53 Application Module- CityRPI, Real-time Model Calculation Reporting.....	55
Figure 3-54 Application Module- CityRPI North America Real-time Monitoring .....	56
Figure 3-55 Application Module- CityRPI General User Input for North America.....	56
Figure 3-56 Application Module- CityRPI, Highest Daily Cases and Prevalence Rate. ....	57
Figure 3-57 CityRPI Filtering Control UI.....	57
Figure 3-58 Application Module- Fatima-CFD Dashboard.....	59
Figure 3-59 Boundary Condition Input Section. ....	60
Figure 3-60 Fatima-CFD Mesh and Import Model Visualization.....	60
Figure 3-61 Plane Parallel to XY Plane at Z = 1.5.....	61
Figure 3-62 FaTIMA-CFD Applications in the Same Place with the Input Files.....	62
Figure 3-63 Result Generator & Final Results Files. ....	63
Figure 3-64 Hiding a Component in the Geometry.....	63
Figure 3-65 Model Representation Option.....	64
Figure 3-66 HOBO RX3000 Station.....	65
Figure 3-67 Example of Sensors Data History UI .....	65
Figure 3-68 Weather Stations Driver Hierarchy .....	65
Figure 3-69 New Driver Device Configurations.....	65
Figure 3-70 New Driver Weather Sensors Data Points.....	66
Figure 3-71 Real-time Sensors Monitoring.....	66
Figure 3-72 Sample of Track Sensor's History, Rain Sensor .....	66
Figure 3-73 Real-time Weather Stations Monitoring.....	66
Figure 3-74 Example of 2.5 Km Points Grid .....	67
Figure 3-75 CMC Driver Hierarchy.....	68
Figure 3-76 CMC New Driver Device Configurations.....	68
Figure 3-77 CMC Device Driver Configurations. ....	68
Figure 3-78 Sample of Model HRDPS URL Path, Girb2 Files .....	69
Figure 3-79 Forecasted Regional Map -Montreal.....	69
Figure 3-80 Automatic Forecasting Module Design.....	70
Figure 3-81 Schedule New Case Automatic Forecasting Workflow. ....	71
Figure 3-82 Middleware CMC Server Settings UI .....	72

Figure 3-83 Middleware SSH Settings UI ..... 72

Figure 3-84 Middleware Upload Settings UI..... 73

Figure 3-85 CMC Bounding Box Points..... 73

Figure 3-86 Preparing Weather Data Input Collection Workflow..... 74

Figure 3-87 Weather Data Input Collected from CMC GRIB2 ..... 75

Figure 3-88 Automatic Forecasting System Integration, Building\_info.txt..... 76

Figure 3-89 Workflow of Synchronizing Data Input Collection with Backend Server ..... 76

Figure 3-90 Automatic Forecasting System Integration Result. .... 77

Figure 3-91 Synchronizing the Output Files with Online Server..... 78

Figure 3-92 Data Result with Colors Information Representation for Result Values ..... 78

Figure 3-93 Online Monitoring the Forecasting of New Cases Simulation..... 79

Figure 3-94 Middleware SMTP Settings UI ..... 80

Figure 3-95 Sending Notification Email Workflow..... 80

Figure 3-96 Example of API's Response Message, Covid New Cases in Canada..... 81

Figure 3-97 API Integration Module UI. .... 81

Figure 3-98 API Integration Module, Link Configuration 1 ..... 82

Figure 3-99 API Integration Module, Link Configuration 2..... 82

Figure 3-100 API Integration, Points Settings ..... 82

Figure 3-101 API Integration Module Result ..... 83

Figure 4-1 The Generated Data Files List..... 85

Figure 4-2 Extending North America Data Source Development. .... 88

Figure 4-3 3D Model Generation and Visualization Functions of Annual Energy Performance Application, Marina District, Qatar..... 89

# List of Tables

Table 1 Attribute Variables Description [30].....30

Table 2 Geometric Data Layout and Third-party API Integration Dictionary Definition. ....53

Table 3 PC and Connection Characteristics. ....84

Table 4 Speed Test Result for Three Cases of Data Input Generation. ....85

Table 5 Generated Data Files Description. ....85

Table 6 Visualization and Data Processing Test for Three Use Cases. ....86

# Chapter 1

## 1.Introduction

### 1.1 Motivations

Urban modeling simulations are vital in optimizing buildings during the pre-design, commission, and operation phases [35]. In building engineering, Building Energy Simulation (BES) and Computational Fluid Dynamics simulation (CFD) are examples of urban modeling simulations that provide more detailed and high-resolution information at a lower cost than laboratory experiments and field tests. This information is essential as it allows designers to investigate internal and external conditions before a building is built, allowing them to test options and select the most effective solutions.

Urban modeling simulations require a high level of user expertise and expert knowledge of using different software and tools for many inputs parameters preparation. Different tools also require different input detail levels; hence the parametrization process highly influences the quality of the simulation results [35]. Moreover, input parameter simulation preparation is usually error-prone and expensive [37].

For instance, the BES applications allow the detailed calculation of the energy consumption to maintain specified building performance criteria. The BES inputs include outdoor climate, occupancy, lighting, and equipment loads [36]. Some main BES tools are DOE-2 [37] Energy Plus, TRNSYS, and EPS-r. IDA ICE.

At the same time, the CFD simulation is typically used to model the movement and temperature of the air within spaces. The complexity in CFD simulation data preparation, presented in modeling an urban-scale CFD problem and the simulation, can be divided into three main sections: data input model preparation, model simulation, and result visualization; preparing the 3D model of the city for the simulation is the most crucial step in model preparation; the CFD data inputs include



geometrical data, building footprint, heights, number of floors, orientation, and other necessary data. Furthermore, the data collection preparation process of the 3D model of the city requires using software such as ArcGIS, OpenStreetMap, and Rhino, all together.

The prior knowledge and skill to learn how to use a variety of software and tools spend much time doing it. So the preparation process can be an expensive solution. Therefore, developing an easy-to-use and efficient method to prepare geometry would be constructive for many users.

Driven by the importance of urban simulations in supporting healthy communities, sustainable buildings, and green planning, there is no denying that the IoT significantly impacts urban modeling applications. That provides a way of sensing and collecting environmental and societal data, both automatically, remotely, and with increasing levels of spatial and temporal detail. However, it is not as simple as connecting all devices or integrating data sets. It is about making sure that data is being collected efficiently and that it can be accessed by relevant stakeholders when it needs to be accessed.

Urban simulations will likely be much more data-focused in the future, and data integration will become more challenging because of the diversity in data types and sources from geographical data, weather forecasting, and other APIs data sources. There are clear advantages to embracing the importance of integrating the Internet of Things (IoT) and Geographic Information Systems (GIS). Moreover, the integration can open up new data preparation and visualization potentials. Additionally, merging IoT and GIS data offers the opportunity to take a creative step toward saving lives and money. It gives decision-makers more authority by verifying the veracity of the information they receive on the highest level of applicability for multiple purposes.

## **1.2 Research Problem and Objective**

This study aims to introduce and implement an integrated framework that integrates different frameworks to support urban modeling simulations and solve data interoperability. In summary, this study will achieve the following:

- Support urban modeling simulations by leveraging the IoT Niagara framework and GIS tools to support environmental sensing.
- Provide a clear system architecture for measuring and controlling environmental systems.
- Provide a solution for preprocessing heterogeneous data for multi-scale urban modeling simulations.

- Provide a user guide illustrating the functionalities of the proposed integrated framework.
- Provide an integrated, scalable, extendable framework as a development approach for future development.

Depending on the discussion of the objectives, this study aims to answer the coming questions:

- How can GIS enable efficient IoT usages to support multi-scale urban modeling simulations, and How it simplify the data collection preparation (pre-process stage) and visualization (post-process stage)?
- How do we build a reusable, extendable, integrated framework that supports urban applications and enables future functionalities?

### **1.3 Thesis Structure**

In the following chapters, this thesis introduces the following contents:

- In Chapter 2: we have reviewed the relevant research on GIS and IoT concepts and IoT-GIS integration-related solutions; the relevant content is divided into two parts for discussion. The First part has provided a better understanding and addressed the significant components. In contrast, the second part surveyed the major relevant frameworks and detailed related research of the platform design process, which used and improved our solution.
- In Chapter 3: we have introduced the implementation approach for data processing and data visualization in the system, which is illustrated by the system architecture design; the system architecture takes the Integrated framework as the core and expands around the system and data sources, modules, and components. The proposed system illustrates that implementation development plays two roles, the first role is a platform, and the second is the implementation development approach framework. Then we presented the framework components and their importance in supporting urban modeling applications. we illustrated the additional modules, such as a 3D modeling module, integration with the Canadian meteorological weather forecasting framework, automated short-term forecasting system, and real-time weather monitoring. In summary, in this chapter, the overall system offers monitoring tools and information systems to urban planners, politicians, and city leaders to improve areas like scenario testing and strategic planning. The system is an integrated framework with modules for data preparation, creation, and integration, a real-time sensing system, a geographic information system, and a database. The 3D modeling module allows the import and merging

of various data sources such as files ( \*.OSM, \*.SHP, \*.OBJ, \*.STL, and \*.GeoJSON ), which presents a solution to data interoperability.

- In Chapter 4: In this chapter, we evaluated the system and its operating status, evaluating the data input generation speed for a few urban-scale cases, visualization processing and rendering performance, and scalability of the system. The evaluation test was performed on a PC with specific characteristics and described the platform functions.
- In Chapter 5: We summarized the overall advantages of the proposed integrated framework in supporting the urban simulations, illustrated the contributions and the limitations that it faces, and we discussed the future direction of the work

# Chapter 2

## 2.Literature Review

This chapter reviews the relevant research on GIS and IoT concepts and IoT-GIS integration-related solutions; the relevant content is divided into two parts for discussion. The First part provides a better understanding and addresses the significant components. In contrast, the second part surveys the major relevant frameworks and detail related research of the platform design process, which used and improved our solution.

### 2.1 Review on Major Components.

#### 2.1.1 Internet of Things (IoT)

IoT is a system of entities (including cyber-physical devices, information resources, and people) that exchange information and interact with the physical world by sensing and processing information[1]. The Internet of Things (IoT) focuses on facilitating communication between Things. It is a wide-area network that uses standard communication protocols. Things are any computing equipment and physical items. They are linked to the Internet and can transfer data through a network without needing human-to-human or human-to-computer interaction. The things in this network would act as consumers or suppliers of the data sent over the network; each sensor generates data and transmits it to them to inform consumers of the current situation. It is anticipated that there will be over 21 billion IoT devices by 2025[2]. IoT environmental monitoring sensors and connectivity provide an effective, efficient way to monitor and support a healthy environment, providing the tools for analysis, preventative detection of contaminants, and energy conservation to reduce our carbon footprint. The IoT can potentially be used to positively impact the environment and lead to more sustainable business practices. IoT sensors can be used to cut down on energy use and track carbon emissions and waste.

##### 2.1.1.1 Fundamental IoT Characteristics

The following are IoT fundamental characteristics [3]:

- **Interconnectivity:** anything can be interconnected with the global information and communication infrastructure.
- **Heterogeneity:** The devices in the IoT are heterogeneous based on different hardware platforms and networks. They can interact with other devices on different networks.
- **Dynamic changes:** The state of devices changes dynamically, e.g., sleeping and waking up, connected and disconnected as well as the context of devices including location and speed. Moreover, the number of devices can change dynamically.
- **Enormous scale:** The number of devices that need to be managed and communicate with each other will be at least an order of magnitude larger than the devices connected to the current Internet.
- **Safety:** As both the creators and recipients of the IoT, well-being. Securing the must design for safety. The benefit includes the safety of our data and physical good endpoints, the networks, and the data moving across. All of it means creating a security paradigm that will scale.
- **Connectivity:** Connectivity enables network accessibility and compatibility. Accessibility is getting on a network, while compatibility provides the standard ability to consume and produce data.

### 2.1.1.2 IoT Architecture

The Internet of Things has advanced significantly over the past few years, and various architectural designs have been put out by researchers [4]–[6]. Early in those studies, a three-layer design was devised, consisting of the Perception layer, Network layer, and Application layer. According to [7], [8], a five-layer architecture is also used. This design suggests the perception, transport, processing,

application, and business layers. Our study takes into account the five-layer IoT architecture shown in Figure 2-1

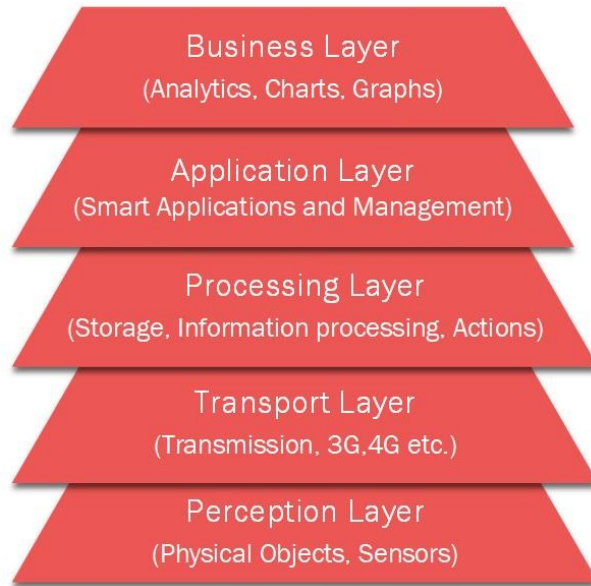


Figure 2-1 Architecture of Internet of Things

- **Perception Layer:**

The foundation of IoT architecture is this layer. Numerous sensors and actuators are utilized at the perception layer to collect essential data such as temperature, moisture content, intruder detection, and sounds. The primary purpose of the perception layer is to gather data from the environment and transmit it to another layer so that actions can be taken depending on that data.

- **Transport Layer**

As its name implies, it serves as the connecting layer between the Perception and Processing layers. Using networking technologies like 3G, 4G, UTMS, WiFi, and infrared., it receives data from the Perception layer and sends it to the Processing layer. Because it facilitates communication between the Processing and Perception layers, this layer is known as the communication layer. Data is always transferred securely, maintaining the collected information's privacy.

- **Processing Layer**

Advanced functions, including storage, computing, processing, and the ability to take action, are available in the Processing layer. It maintains all data sets and sends the necessary data to each device depending on its address and name. Additionally, it can make decisions based on calculations made from sensor data sets.

- **Application Layer**

Based on information collected from the Processing layer, the Application layer controls every aspect of the application process. Sending emails, turning on alarms, and security systems, turning devices on or off, using smartwatches, smart agriculture, and other functions are all included in this application.

- **Business Layer**

Any device's success depends not just on the technologies it uses but also on how it is distributed to its users. For the device, the business layer does these responsibilities. It entails creating graphs, flowcharts, outcomes analysis, and how the device can be improved, among other things.

## 2.1.2 Geospatial Information System (GIS)

### 2.1.2.1 Definition

A Geographic Information System (GIS) is an organized activity by which people measure and represent geographic phenomena and then transform these representations into other forms while interacting with social structures[9]. Digital mapping is just one application of GIS's power; layers can also be used for data integration, analysis, management of natural resources, and decision-making are all aided by it [10].

A GIS may result from integrating data, software, hardware, processes, and people, as shown in Figure 2-2.

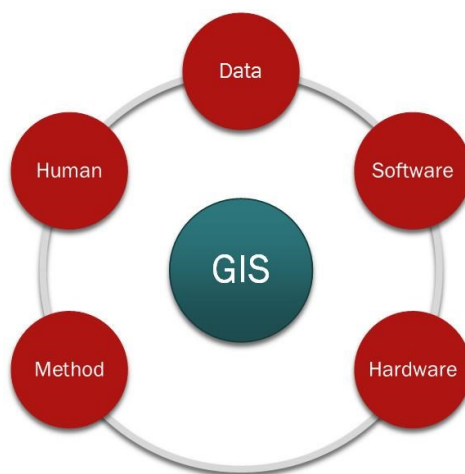


Figure 2-2 GIS Components

GIS can offer data storage and management facilities and provide priceless geospatial data. The Data can be viewed as the essential component of a functioning GIS. If geospatial data is present, any data can be used in GIS. The Software is a collection of tools to store, process, and present data. Key software elements include a graphical user interface, a database management system, and tools like ArcMap's Toolbox. The Hardware refers to the device on which GIS can be used. Nowadays, diverse groups of devices on a network with various architectures or topologies might serve as hardware[2].

### **2.1.2.2 GIS in Urban Planning**

Urban planning focuses on how spaces function effectively to improve people's lives in an urban area. Moreover, the plan for an urban area must achieve the objectives and the needs of the people under study. These achievements are strongly linked with economic, environmental, and social factors as key elements for sustainable development. The urban area is a region surrounding a city. Most inhabitants of urban areas. Urban areas are very developed, meaning human structures such as houses, commercial buildings, roads, bridges, and railways are dense. "Urban area" can refer to towns, cities, and suburbs. An urban area includes the city itself, as well as the surrounding areas. Urban areas are systems of tremendous, ever-evolving complexity; responsibly guiding an area's development requires spatial information that's robust, nuanced, and constantly updated. This challenge has made geographic information science and technology (GIST) invaluable to urban simulations and has put geographic data analytics and technical innovation at the center of urban modeling simulations. GIS empowers urban planners to enhance visibility into data and monitor fluctuations over time, evaluating the feasibility of proposed projects and predicting the effects on the environment, providing a better understanding of a city as a multi-scale and multilayer complex system, scenario-testing, and strategic planning, collecting important aggregated data regarding building construction, energy consumption, and occupant wellbeings.

Figure 2-3 gives us knowledge about GIS applications in urban area planning [11] GIS in urban planning enables spatial analysis and modeling, contributing to many crucial urban planning tasks. These tasks include site selection, land suitability analysis, land use and transport modeling, the identification of planning action areas, and impact assessments[12]. GIS applications cover significant areas of urban arranging and advancement: Infrastructure management (transport, public utilities, and stormwater/waste), regional planning, resource management, environmental assessment, socio-economic development, emergency management, and education. The benefits of



using GIS in urban planning include increasing efficiency, saving time/money, and supporting decision-making. Improve accuracy, manage resources, automate tasks, increase government access, increase public involvement, and promote more extensive public agency collaboration.

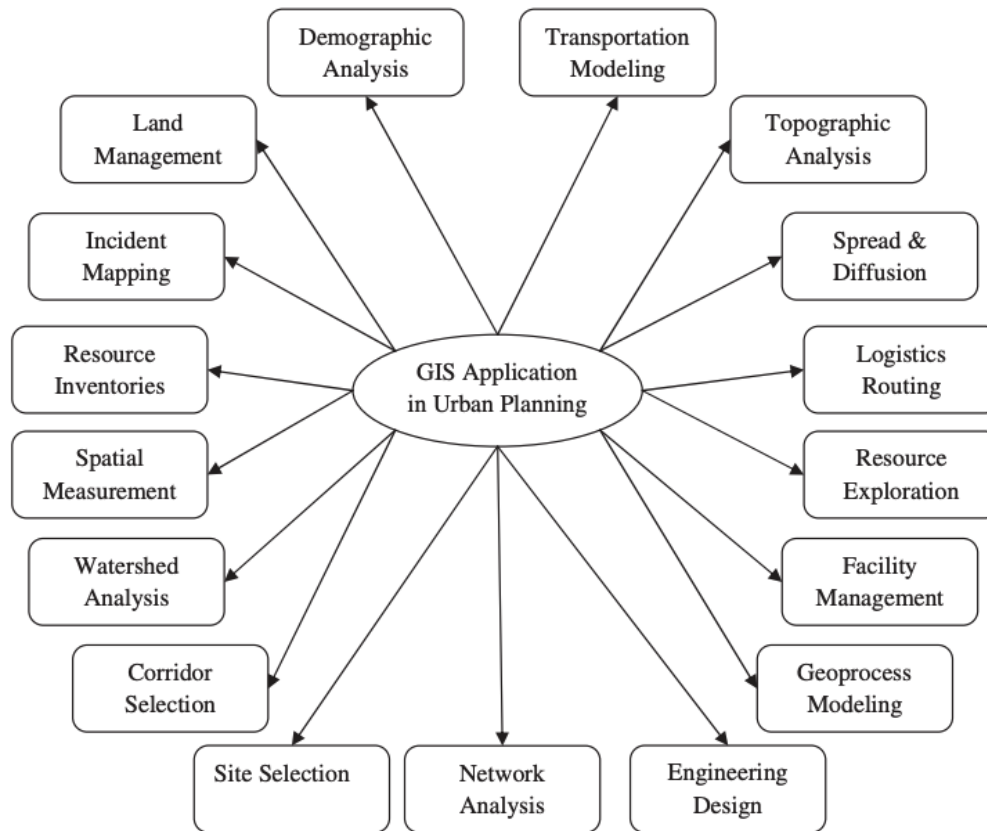


Figure 2-3 GIS Application in Urban planning [12]

### 2.1.2.3 GIS Application in City Urban Planning

GIS technologies have increased over the past few decades, and now GIS has provided a wide range of data execution and dissection tools; the typical GIS applications include:

- Review and analysis of development plans.
- Regional planning beyond the borders of a city or town.
- Review of environmental impact.
- Disaster risk management and mitigation.
- Mapping the delivery of utilities and planning for service interruptions.
- Preservation of historical sites.
- Checks on regulatory compliance.

Another significant move was the relevance of GIS, remote sensing methods such as satellite images, light detection and ranging, and data generation, which authoritatively remotely sensed data[11]. Figure 2-4 shows how GIS can support urban planning. Using GIS in city development can detect environmental problems or natural disasters, estimate and compare urban planning scenarios, predict, prevent transportation congestion, and even analyze employment and social services. On the one hand, cities are showing considerable opportunities for development in the coming long period, while on the other hand, they are also offering different challenges [13].

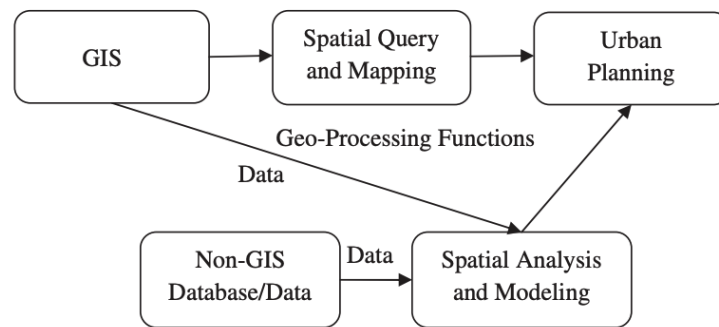


Figure 2-4 GIS and Urban Planning [12]

GIS can store information in a database and represent it visually in the form of mapped data. GIS technology assesses urban development and its extension path and finds suitable urban planning locations. There are also many GIS applications; Nowadays. GIS tools assist urban planners in inspecting problems more quickly and thoroughly and formulating solutions. Many arranging divisions that had gotten mapping frameworks in the past have moved to GIS[11].

Implementing innovative technologies in urban areas increases efficiency and sustainability and improves people's quality of life. More cities are striving to become cities by implementing innovative technologies, and GIS is an essential technology for collecting, analyzing, and presenting intelligent data.

### 2.1.3 Urban Simulations

With the increased energy prices and global warming crisis in the last century, the requirement for more sustainable cities and buildings has increased significantly. According to the international energy agency, Buildings are responsible for 27% of total energy sector emissions and consume 30% of the final global energy. To address these needs, Urban simulation was used to solve and predict multiple problems before they could happen. Urban simulation provides comprehensive

thermal environment modeling that affects pedestrians' health and thermal comfort. Moreover, it provides urban pollutant and contaminants dispersion which has gained more interest in the last three years due to the Covid -19 pandemic. The urban simulations applications include urban building energy simulation, urban energy system sizing, urban wind/temperature field analysis, outdoor/indoor thermal comfort, airflow, and pollutant dissemination.

As stated earlier, urban simulation has many applications due to its importance. Researchers have summarized these applications into two main categories (Single-objective simulation & multi-objective simulation). A summary of these simulations will be provided below:

- **Single Objective Simulation:** As the name implies, a Single objective means that only one parameter will be studied (airflow, pollutant, or thermal simulation). It analyzes the interaction between a building or a group of buildings and the surrounding environment, including air.
- **Multi-Objective Simulation:** Multi-simulation is focused on more than one parameter. It can include all types of CFD simulation and more as daylight and solar simulation. Of course, these simulations are challenging due to lots of uncertainty and complexity. It also requires a lot of computing power, time, and capability, which is not easily found to this date despite the significant advancement in computing technology in the last century.

### **2.1.3.1 Building Energy Simulation (BES)**

Urban simulation studies are necessary for urban building energy simulation/urban energy system sizing. The building interacts with other buildings and the surrounding natural environment, significantly influencing building energy consumption. Urban simulation is crucial when the traditional building simulation is scaled up to urban scenarios, as many properties well-known on individual buildings become unreliable when considering large-scale. Building Energy Simulation (BES) simulations are vital in optimizing the buildings during the pre-design, commission, and operation phases[35]. The urban building energy model's reliable input parameters include geometrical data, envelope properties, HVAC system, and occupancy behavior. Besides, the correct meteorological data is crucial to urban building energy consumption assessment as the outdoor boundary condition. Whole BES tools allow the detailed calculation of the energy consumption to maintain specified building performance criteria. The inputs include outdoor climate, occupancy, lighting, and equipment loads, and These calculations are generally performed over an entire year [36]. Some main BES tools are DOE-2 [37] Energy Plus, TRNSYS, and EPS-r. IDA ICE. Their application requires a high level of user expertise and expert knowledge of many

input parameters. Different tools require different levels of input detail, which often do not match with available data; hence the parametrization process highly influences the quality of the simulation results [35]. However, simulation preparation of input parameters (e.g., building geometry and material properties) is usually error-prone and expensive [37]. The potential overall error in performance predictions is a function of the degree of estimation of uncertain input parameters; however, preparing digital design models for BES typically requires tedious manual alteration. Previous studies show how this input preparation is complicated; determining the effect input data may have on BES is a complex task, as it deals with a variety of physical and stochastic phenomena: weather conditions, thermal heat transfer through building structures, radiative heat exchange between windows, walls, solar radiation, occupancy behavior, HVAC system and electric equipment [38].

### **2.1.3.2 Computational Fluid Dynamics (CFD)**

CFD is a subfield of fluid mechanics that analyses and resolves fluid flow problems using numerical analysis and data structures. CFD is a widely used approach to solving wind and temperature studies by providing more detailed high-resolution information at a lower cost than laboratory experiments and field tests. The idea of CFD simulation is that it relies on numerical computation and approximation. There is no exact mathematical solution to these governing equations. The domain is then discretized to get a solution, and the equations are approximated numerically over this domain. This requires a lot of computing power, especially for larger problems such as urban simulation. Another problem also is that numerical simulation requires a set of boundary conditions to be known before starting the problem. This can be easy to acquire for smaller cases, but for larger cases, there are a lot of uncertainty and difficulties in getting these data. City fast fluid dynamics (CityFFD) is a fast and stable numerical model suitable for modeling large-scale airflow problems. CFD simulation, commonly idealized building geometry, will be used to simulate the wind and thermal field of urban areas under various climatic conditions, such as the single building [14]–[16], building block [17], [18], or mixed high-rise and low-rise buildings [19], [20]. However, these buildings or urban geometries are all artificial and cannot be used to simulate actual conditions. Thus, for better predicting the microclimate of the urban area, it is of great importance to have a system helping export the geometry of real-world buildings for both CFD and urban energy simulation [21].

For instance, the complexity in CFD simulation data preparation, shown in modeling an urban-scale CFD problem, can be divided into two main sections: model preparation and model simulation. Preparing the 3D model of the city for the simulation is the most crucial step in model preparation. The geometrical data include building footprint, heights, number of floors, orientation, and other necessary data. It is often better to create geometry specifically for each CFD simulation. Currently, the preparation process and creation of the 3D model of the city requires using much software, such as ArcGIS, OpenStreetMap, and Rhino. Using this software requires prior knowledge and skill to learn how to use it and spend much time doing it. So the preparation process can be an expensive solution. Therefore, developing an easy-to-use and efficient method to prepare geometry would be constructive for many users. The flowchart below shows the CityFFD implementation workflow in the [32], which shows preprocessing and input data files.

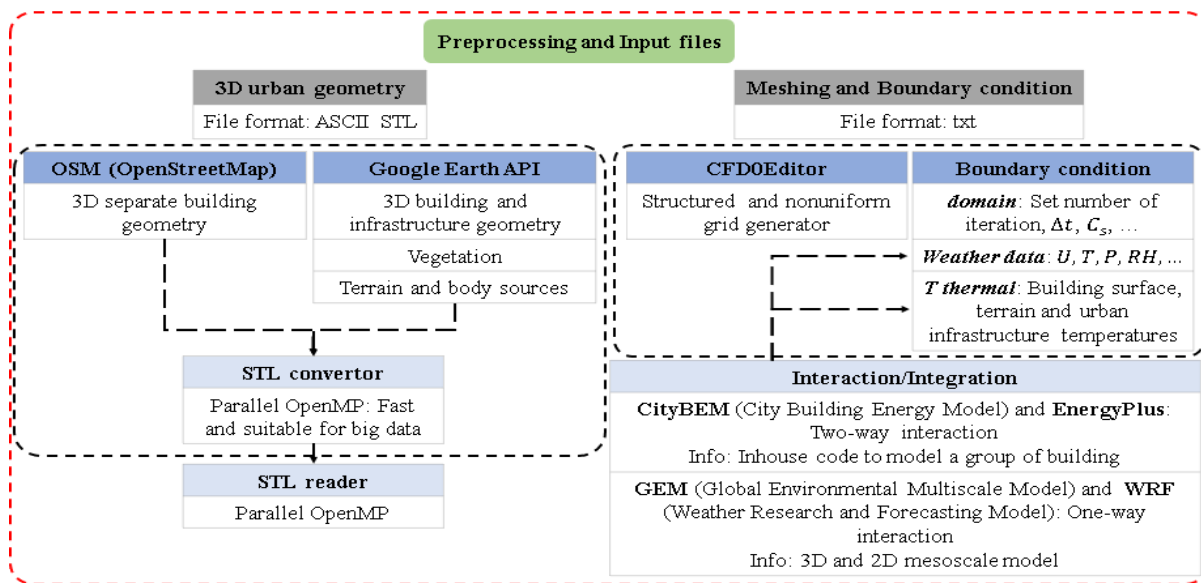


Figure 2-5 Preprocessing and Data Collection Generation in CityFFD Simulation.[22]

## 2.2 Review on Major Frameworks.

### 2.2.1 IoT Niagara Framework

The Niagara Framework is a Java software framework for integrating disparate building automation systems into a single, manageable interface that can run on multiple hardware platforms. The framework uses the Java Virtual Machine (JVM) as a common runtime environment

across various operating systems and hardware platforms. The core framework scales from small, embedded controllers to high-end servers[23].

Heterogeneous system integration Niagara is designed from the ground up to assume that there will never be anyone standard network protocol, distributed architecture, or Fieldbus. Instead, the framework integrates cleanly with all networks and protocols, standardizing the box's contents. The framework also scales to highly distributed systems composed of thousands of nodes running the framework software. Systems of this size span many network topologies and usually communicate over unreliable Internet connections. The framework is designed to provide an infrastructure for managing systems of this scale. Component software Framework architecture is centered on component-oriented development[24], [25]. Components are pieces of self-describing software that can be assembled like building blocks to create new applications. This component-centric architecture solves many problems:

- (1) Components normalize the data and features of heterogeneous protocols and networks to integrate them seamlessly.
- (2) Components and graphical tools provided by the framework allow applications to be assembled without requiring a Java developer.
- (3) Components provide unsurpassed visibility into applications. Since they are self-describing, tools can quickly interrogate how an application is assembled, configured, and occurring at any time—these aid debugging and application maintenance.
- (4) Components enable software reuse.

Building automation systems automate many services required to successfully operate and manage facilities, including environmental control, fire and life safety, lighting, energy management, and security access control. These services have traditionally been provided as independent, standalone systems by multiple, proprietary vendors as independent, standalone systems by multiple, proprietary vendors. It is widespread for a single hospital facility to have 2-5 independent automation systems, while a national chain might have ten or more systems throughout the country. Figure 2-6. While improvements in systems integration have occurred over the past ten years, only recently has it become cost-effective and, in some cases, technically possible for a user to deploy an easily portable and fully integrated facility management solution. The Niagara software suite implements a highly efficient adaptation of the JavaBean component software model and internet

technologies to provide customers with true interoperability across a wide range of automation products. As a subset of the complete framework, the Niagara object model can integrate a wide range of physical devices, controllers, and primitive control applications, including BACnet®, Lonworks®, Modbus®, and oBIX objects, and legacy control points. The architecture supports future enhancements by allowing legacy systems to be brought forward[25].

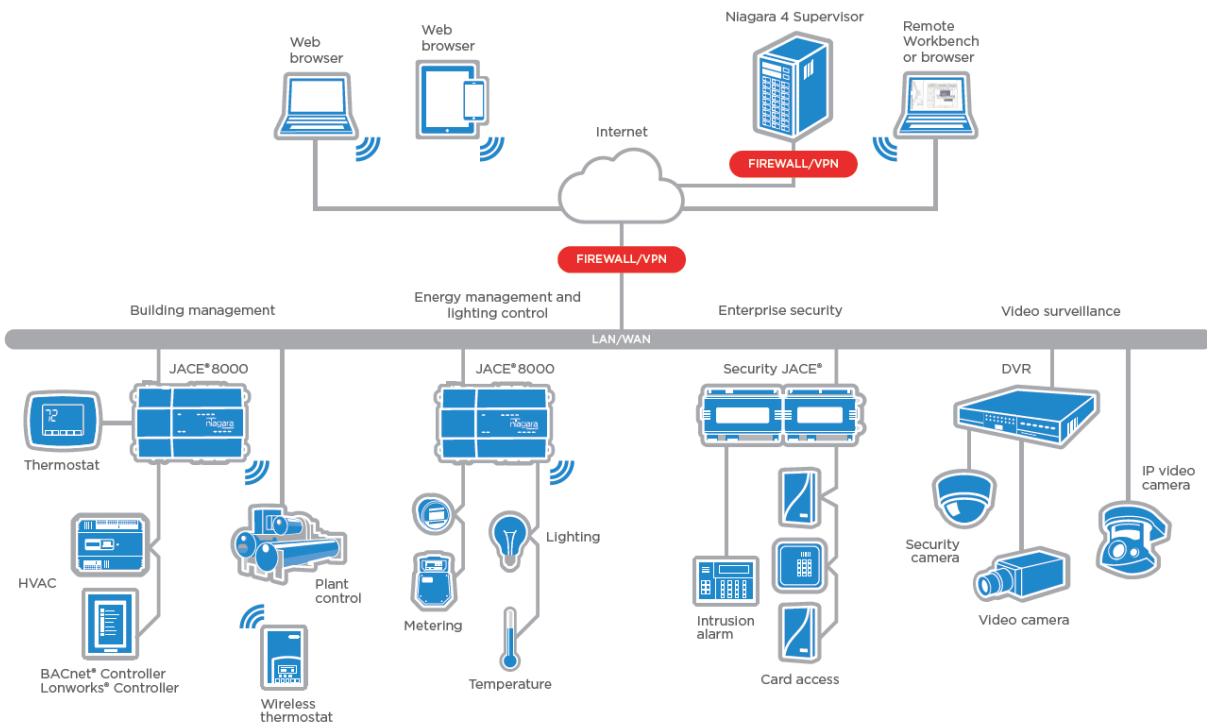


Figure 2-6 Integration Architecture for Niagara Framework [25].

### 2.2.2 Vuejs Framework

Vue is an open-source, progressive JavaScript framework used to create user interfaces. Vue's progressive adoption strategy simplifies integrating into projects that use other JavaScript libraries. Additionally, Vue can serve as a web application framework that powers sophisticated single-page applications[26]. It consists of an approachable core library that focuses on the view layer only and an ecosystem of supporting libraries that helps to tackle complexity in large Single-Page Applications [27].

Web Components is an umbrella term for web-native APIs that allow developers to create reusable custom elements. In the Vue framework centered on component-oriented development, the components allow us to split the UI into independent and reusable pieces and think about each piece in isolation [28]. It is common for an app in the Vue framework to be organized into a tree of nested components and encapsulate custom content and logic in each component, as in Figure 2-7.

The component-based design produces inherent efficiencies for designers and developers. It solves the challenges associated with duplication and inconsistency in applying themes while driving the development of better quality, reusable components. Components allow for far more possibilities in terms of product design and can be configured to provide optimal performance for the target application. - Cost control: Starting from the sensor gives manufacturers control over final camera costs. Usually, a component provides a particular function or group of related parts. In programming design, a system is divided into components that are made up of modules. Component testing means testing all related modules that form a component as a group to ensure they work together.

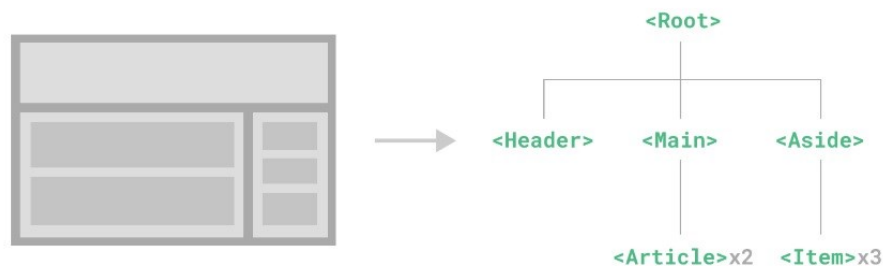


Figure 2-7 Vue Framework Components [28]

Vue has excellent support for both consuming and creating custom elements. Whether for integrating custom elements into an existing Vue application or using Vue to build and distribute custom elements.

### 2.2.3 CMC Weather Forecasting Framework.

In November 2014, the experimental 2.5-km pan-Canadian High-Resolution Deterministic Prediction System (HRDPS) was introduced, with 48-h integrations run four times per day [29].

The center services provide an online High-Resolution Deterministic Prediction System or HRDPS. The HRDPS is operational except for the northern domain, which remains experimental.



The fields in the HRDPS high-resolution GRIB2 dataset are made available four times a day for the Pan-Canadian domain for a 48-hour forecast period (except for the northern domain).

The High-Resolution Deterministic Prediction System (HRDPS) carries out physics calculations to arrive at deterministic predictions of atmospheric elements from the current day to 48 hours into the future. Atmospheric elements include temperature, precipitation, cloud cover, wind speed and direction, and humidity; this product contains raw numerical results of these calculations. Geographical coverage of the system is most of Canada. Data is available at a horizontal resolution of 2.5 km. Data is available for 28 vertical levels. Predictions are performed up to four times a day. CMC supports six domains continental, north, experimental, east, prairies, west, and maritimes[30].

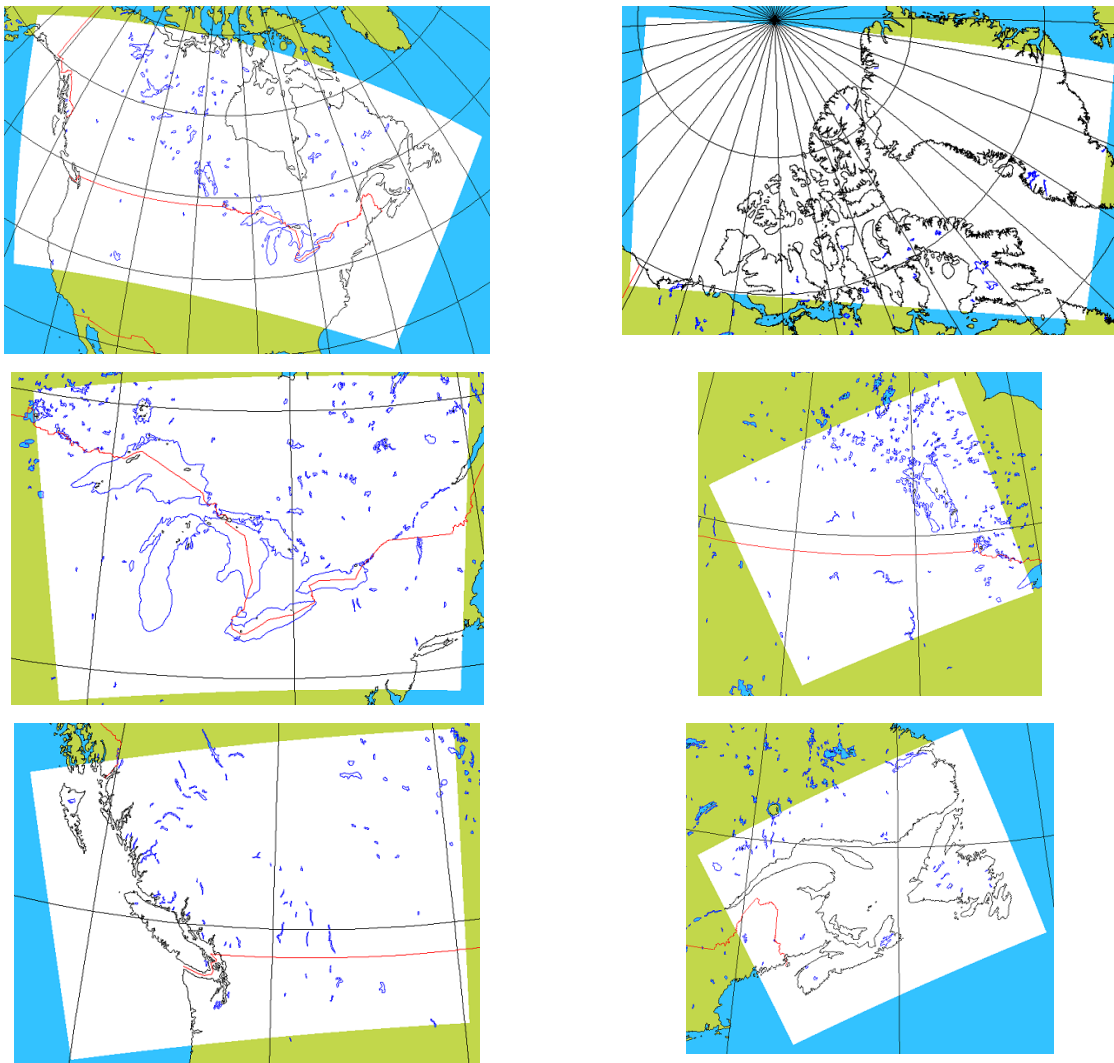


Figure 2-8 CMC Weather Forecasting Framework Supported Continental Domains [29].

Users benefit most from using these new data with a detailed forecast of 39 data variables for different environmental applications. Significantly during the change of seasons and in wintertime, when rapid changes in temperature and winds cause phase transitions of precipitation (freezing rain to snow to rain, for example), 2.5 km forecasts could add much value. Also, in the case of short-term forecasts in the presence of complex terrain or along shores, the influence of changes in altitude, topography, and nature of the terrain will be better described for phenomena at this scale (lake or sea breezes, local valley flows, and phase changes.). Even over less rugged terrain or water away from shore, these more precise forecasts could be helpful repeatedly over a long period. As well, for hydrological forecasts on smaller basins, the HRDPS should be considered [31].

Summarizing all weather data variables supported by CMC HRDPS as follows:

Absolute vorticity, Air Density, Albedo, Average surface relative humidity, Convective Available Potential Energy, Deep soil temperature, Dew point depression, Dew Point temperature, Downward incident solar flux (Accumulated), Geopotential Height, Helicity, Ice fraction, Incoming I.R. accumulated flux, Incoming visible accumulated flux, Land cover, Lifted Index, Mean Sea Level Pressure, Outgoing infrared energy exiting the atmosphere, Precipitable water, Showalter Index, Snow Depth, Soil moisture, Soil temperature near the surface, Specific humidity, Surface accumulated net I.R. flux, Surface Pressure, Temperature, Thickness between two isobaric levels, Total Cloud, U Wind Component, Upward surface latent heat flux, Upward surface sensible heat flux, V Wind Component, Vertical Velocity, Visible flux absorbed at the surface – Accumulated Visible solar flux leaving the atmosphere, Water temperature, Wind direction, and Wind speed – Module.

### **2.3 Summary**

To support urban simulations, this chapter did a comprehensive review of current urban simulation tools, highlighting their importance and application requirements. Moreover, the relevant research on GIS and IoT concepts and IoT-GIS integration-related solutions; the relevant content is divided into two parts for discussion. The first part provides a better understanding and addresses the significant components. In contrast, the second part investigates surveys of the major main relevant frameworks and detail related research of the platform design process, which was used to improve our solution.

# Chapter 3

## 3. System Software Design

In this chapter, we have illustrated the system architecture design; The system is an integrated framework with modules for data preparation, creation, and integration, a real-time sensing system, a geographic information system integration, and a database. The Integrated framework plays as the core and expands around the system and data sources, and modules. we have illustrated the implementation development approach for data processing and data visualization modules, and then we presented the framework components and their importance in supporting urban modeling applications. We illustrated the integration with the Canadian meteorological weather forecasting framework, automated short-term forecasting system, and real-time weather monitoring. In summary, in this chapter, the overall system offers monitoring tools and information systems to support urban planners, researchers, politicians, and city leaders to improve areas like scenario testing and strategic planning. The system plays two roles, the first role is as a platform, and the second is as a development approach framework.

### 3.1 System Architecture

The system is an integrated framework with modules for data preparation, creation, and integration, a real-time sensing system, a geographic information system, and a database; The Integrated framework plays as the core and expands around the system and data sources, modules, and components. The system architecture design is shown in Figure 3-1. The core unit is composed of five main parts. The Integrated framework's core unit builds on the IoT Niagara and Vuejs framework, providing high scalability. The core unit is integrated with several systems.

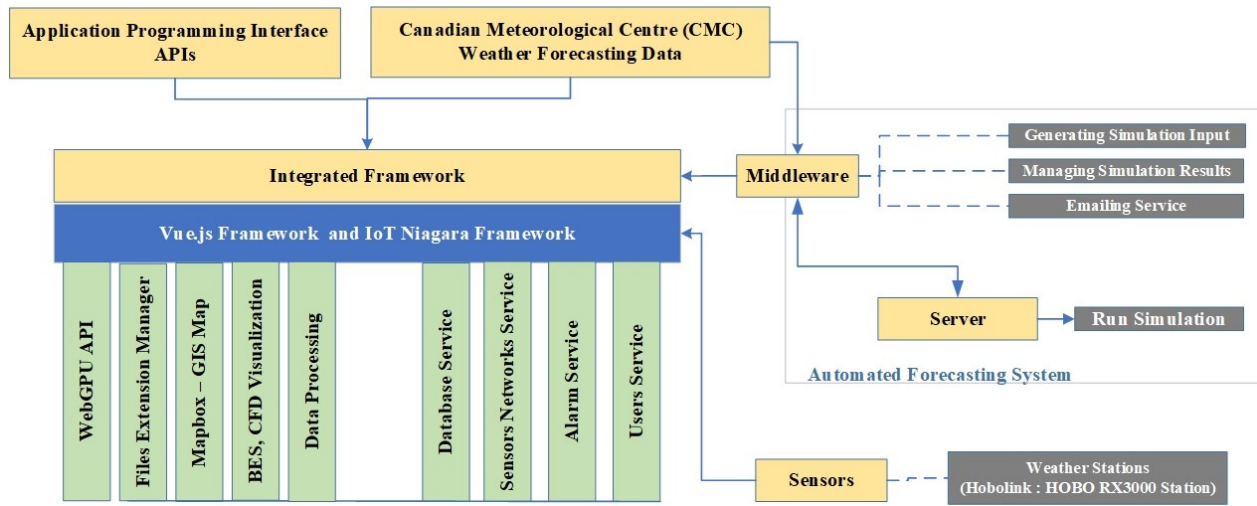


Figure 3-1 System Architecture Design

The system has been designed to provide additional services such as a 3D modeling, integration module particularly for integrate with the Canadian Meteorological Center (CMC) weather forecasting framework and providing a short-term forecasting service, and real-time weather monitoring, additionally provide a 3D modeling module, that allows importing and merging of various data sources such as files ( \*.OSM, \*.SHP, \*.OBJ, \*.STL, and \*.GeoJSON ).

### 3.2 Integrated Framework

This framework is built on the Niagara and Vuejs framework; in this study, we extended the functionality of the Niagara framework and integrated it with Canadian Metrological Centre (CMC) weather forecasting framework, a real-time weather station, application programming interface APIs integration module, an automated forecasting system solution, the provided integrated framework playing a container for other system components.

Firstly, summarizing the advantage of using the Niagara framework in our system integration:

- Niagara framework allows connecting devices on traditional communications media.
- Allow modeling of those devices in software.
- Programming applications to use the information in those devices.

Before a device, such as a temperature sensor, can be used, the information from those devices must be pulled into the system software. then models those devices and their data types in software through the standard object model. This usually entails simplifying the device's data types to make

them easier to manipulate and control through the software. The Niagara common object model is then used to build applications, with the goal being to provide nonprogrammers with a means to program the system easily without developing raw code. The Niagara common object model is similar to a programming language in that a few key concepts are used. However, the real power is in the reusable libraries of applications and collections of objects that are available once the users understand the key concepts and put them to work, and use the objects to build control system solutions quickly and efficiently.

The Niagara common object model allows the framework to:

- Provide secure two-way communication between devices and the Internet.
- Send real-time device information across the Internet
- Control devices in real-time across the Internet.

The following figure illustrates the Niagara software subsystems and the software processes and protocols, respectively.

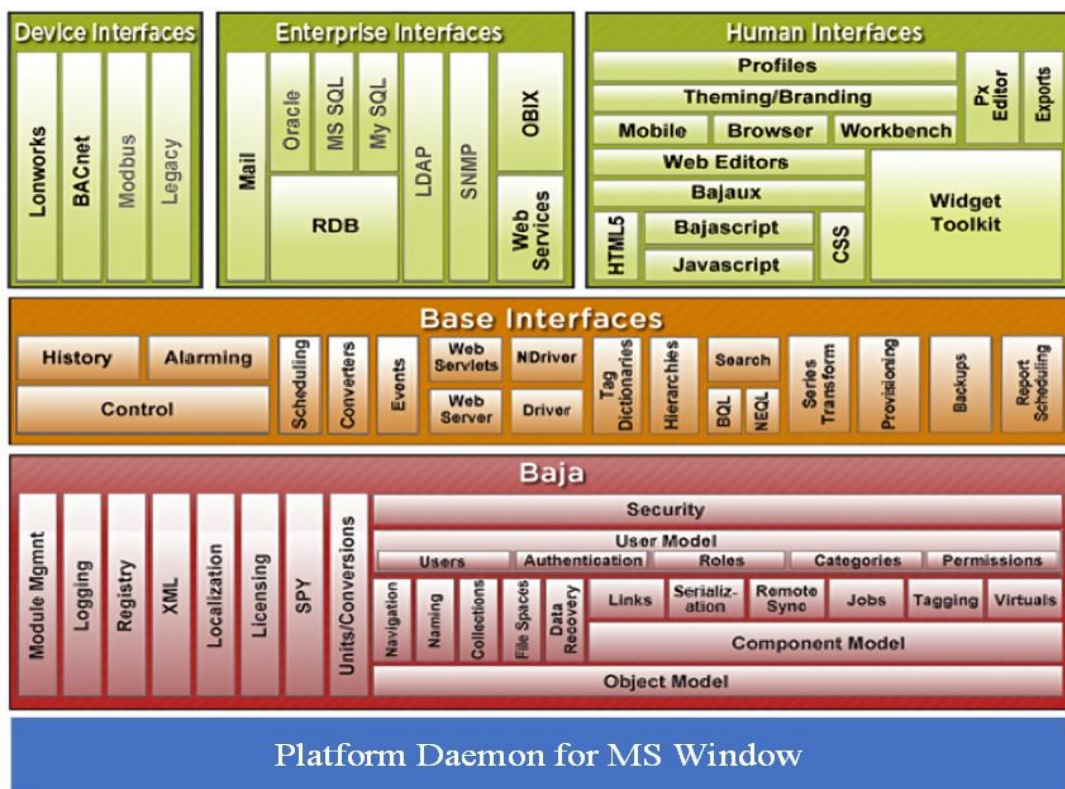


Figure 3-2 Niagara Software Subsystems

In addition to IoT Niagara framework built-in features, we have developed and improved several modules and integration drivers, that are used on the Niagara framework and support urban

simulation applications. The frontend component is built on the Vuejs framework, providing several modules such as visualization using an interactive map, data integration tool, files extension converter, and real-time monitoring.

### 3.2.1 Dashboard Design

This section explains the dashboard design and the applications module used. The dashboard web application is divided into four application modules the Main dashboard, CityRPI, Fatima-CFD, and Fatima-CFD (WebGPU). The core application includes several functionalities such as real-time monitoring, mapping functions, a visualization module, and a data integration support module. CityRPI is an application module for monitoring and mitigating indoor airborne COVID-19 transmission risks at the city scale, and the Fatima-CFD application module is for supporting input generation and indoor CFD simulation visualization, while the Fatima-CFD (WebGPU) is an upgraded version of Fatima-CFD that is leveraging the WebGPU API to run CFD offline simulation on the individual browser. The implementation of these applications uses the same design approach, data sources, map provider, and implementation frameworks.

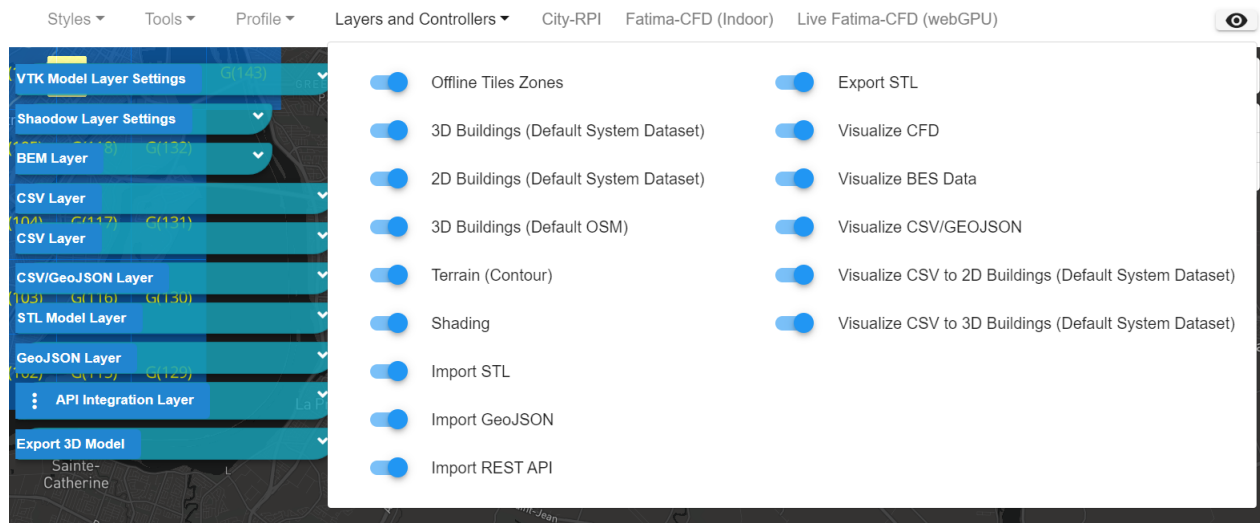


Figure 3-3 Dashboard Modules Layers

Figure 3-3 presents the main dashboard user interfaces and main menus. Each item in the layers and controllers menu represents a functionality or a data integration control method, that can be imported as a layer to the map, then can be exported in other data forms, the map component plays an essential role in our system, by mapping the real-time monitoring data, visualization data. and data integration support modules, by providing a valuable ground for data integration in the system,



data processing is an essential part of this integrated framework, which is also implemented for each of the previous web applications. as shown in Figure 3-7.

The main dashboard is built using the Vue JavaScript framework, as we have explained in the first section of chapter 2. Vue is an open-source, progressive JavaScript framework used to create user interfaces. Vue's progressive adoption strategy makes it simple to integrate into projects and implements web components as an umbrella term for a set of web-native APIs. Vue allows us to create reusable custom components. The components allow us to split the UI into independent and reusable pieces and think about each piece in isolation, this adds a high level of extensibility, and the layers can interact together.

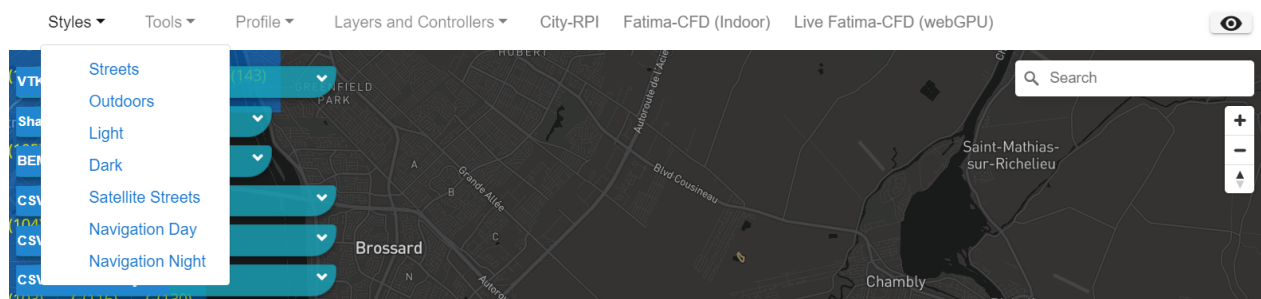


Figure 3-4 Dashboard- Supported Map Style Systems

The Styles menu provides an easy switching of map styles, map style is a set of customizations associated with a map, referenced in app code to display the customized map. The Map component supports six map styles and the ability to change the fonts, colors, and icons on a map style through the visual interface of the map provider studio.

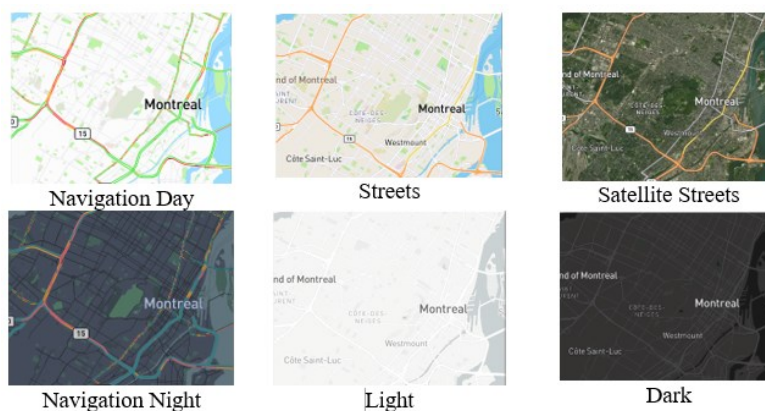


Figure 3-5 Map Styles

Data conversion and data preprocessing modules are essential for any academic and research project, particularly in urban modeling application data preparation; it is an integral part of data

analysis and GIS mapping. The Tools menu focuses on the 3D modeling modules, that allow converting and manipulating various data sources such as files ( \*.OSM, \*.SHP, \*.OBJ, \*.STL, and \*.GeoJSON ).

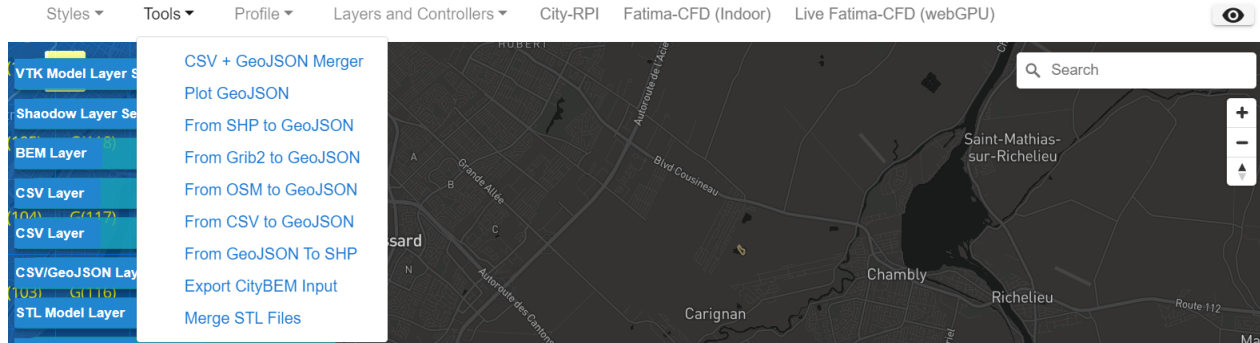


Figure 3-6 Dashboard- Files Conversion Modules

They allow users to transform data formats and merge data sets more easily and quickly. The data collection processing section discusses these tools' purposes and advantages in detail.

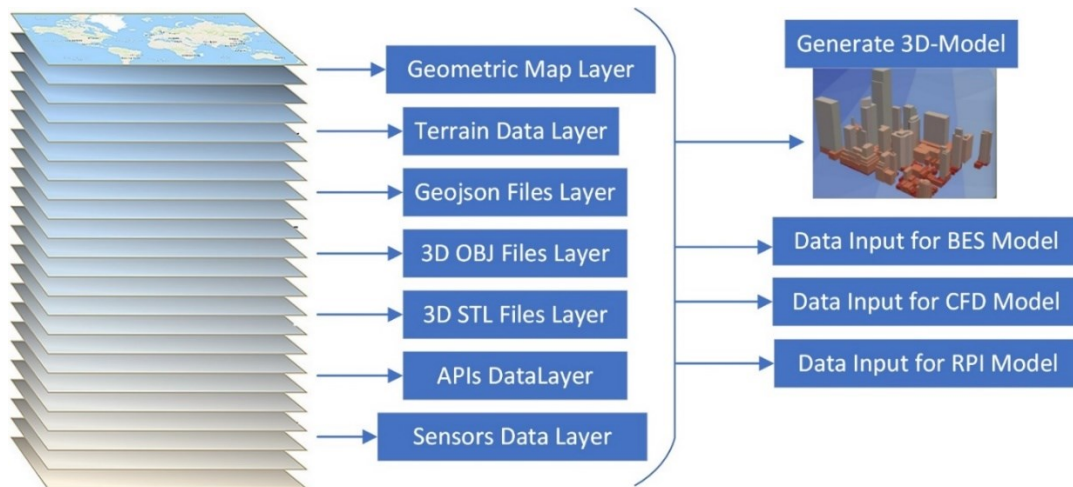


Figure 3-7 Supported Layers System

As mentioned previously the map plays playing an essential role in the dashboard, mainly is that providing high flexibility in combing multiple data sources together, such as geometric data, terrain data, external three-dimensional models, sensors data, and external APIs data, and provides the ability to export integrated data as data input for other urban simulation models. The system arranges all the data sources as a set of layers. Each layer comprehensively introduces a data source for various educational services and applications, mapping, visualization, and research. Various



data sources can be beneficial to improve visualization, planning processes, evaluation of plans and decision-making, and extract useful information for other purposes, such as generating insights into sustainable scenario-testing and strategic planning. The following section will explain the data processing phase in detail.

### 3.2.2 Data Collection Processing

#### 3.2.2.1 Default System Dataset Preparation and Deployment

In this section, we explain how the system dataset is prepared and deployed to be the default dataset for Montreal city, which became one of the integrated framework data sources after the deployment. The default system dataset combines two primary data sources—building layout and buildings characteristics dataset. The first dataset is extracted from OpenStreetMap, and its primary purpose is to provide building outlines, heights, and other essential information about buildings in the city. The second dataset is a collection of attributes about buildings from the City of Montreal open data portal. It includes information such as the construction year of the building type of construction material used for buildings. These datasets are combined to create a better and more accurate representation of Montreal's urban environment concerning the number of buildings in it and their general shape and characteristics. Figure 3-8.

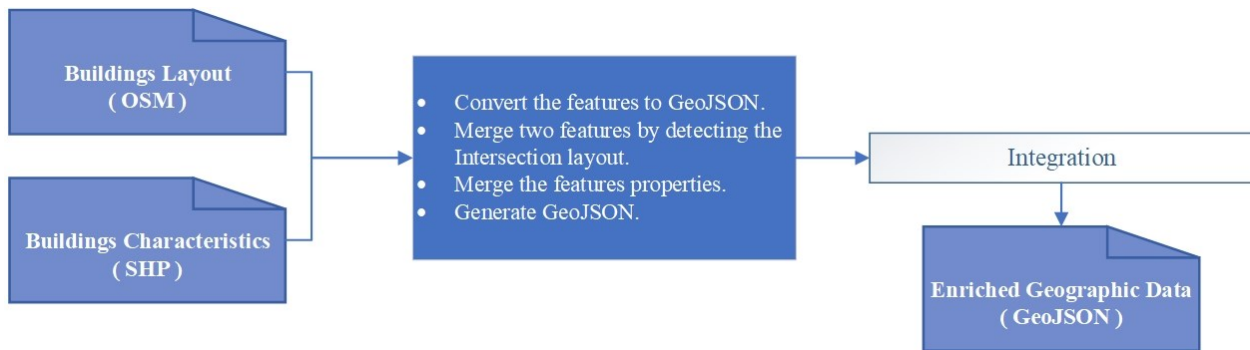


Figure 3-8 Data Integration- Default Data Sources

Before we explain the data sources, we summarize each source's file data structure below. Many standards have been invented that guide how to handle and transmit a massive volume of data. One of these data-interchange formats, easily readable for humans and parable for machines, is the very well-known JSON (JavaScript Object Notation). The JSON format, briefly summed up, consists of key-value pairs that can contain any kind of information if typed with a keyboard, from text to numbers and data sets. These values can also be more key-value pairs and so on. At the same time,

geospatial data is a kind of unique data related to a physical part of a map. In the geospatial data, many formats can be used (Shapefile, OSM, GeoJSON, among many others).

- **GeoJSON file format**

GeoJSON, as its name implies, is JSON data with a specific structure specifically designed to treat geospatial information. The following explains the basics of this standard and how to use it. a GeoJSON is simply a JSON consisting of a list of key-value pairs. However, it must follow a specific structure.

```

{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}

```

Figure 3-9 Sample of GeoJSON Structure [32]

Attributes	Description
Type	This property indicates whether the data consists of a single element (“Feature”) or a set of elements (“FeatureCollection”).
Geometry	<p>The geometry attribute indicates what type of geometry is represented by the GeoJSON. It comprises two properties: “type” which indicates the type of geometry being defined, and “coordinates”, which consist of the numerical description of the points that define the geometry. Since the example above consists of a single point, there are just two elements: latitude and longitude. GeoJSON supports the following geometries:</p> <ul style="list-style-type: none"> <li>• Point: Consists of a single point in space.</li> <li>• LineString: Represents a line in a map.</li> <li>• Polygon: Displays a polygonal shape.</li> <li>• MultiPoint: Set of points.</li> <li>• MultiLineString: Set of line strings.</li> <li>• MultiPolygon: Set of polygons.</li> <li>• GeometryCollection: Set of mixed geometries (points, line strings &amp; polygons).</li> </ul>

Properties	The properties attribute is made up of metadata about the geospatial data being represented. It can be anything if the data is representable with textual characters (from a simple name to the population being contained in a polygon or even the longitude of a line string in the real world).
------------	--

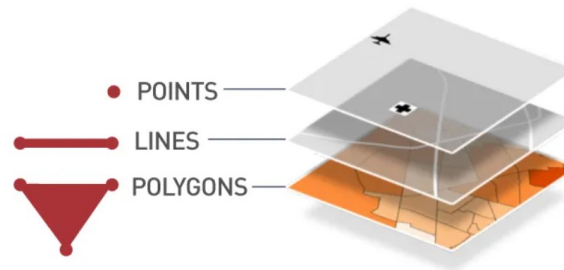


Figure 3-10 Example of GeoJSON Supported Geometries [32]

- **OpenStreetMap OSM file format**

OSM is being used in industry and by government agencies worldwide and has a wide range of applications, including Web-based mapping, Web GIS, data analysis, routing and navigation, and data extraction. Leading companies in this domain include Mapbox, MapQuest, Stamen, Mapzen, CampToCamp, and Geofabrik. Most of these companies also provide OSM services to the user community, including OSM data extracts, web-map layers for online mapping, and specialist visualization. The OpenStreetMap (OSM) project was founded in 2004 and has now positioned itself as the most famous example of Volunteered Geographic Information (VGI) on the Internet. [33]. The OSM data model is very straightforward to understand. There are three primitive data types or objects: nodes, ways (polygons and polylines), and relations (logical collections of ways and nodes). A way comprises at least two nodes (for polylines) or three nodes (for closed polygons). A node represents a geographic point feature, and its coordinate is usually expressed as latitude and longitude. Within OSM, every object must have at least one attribute or tag (a key/value pair) assigned to it to describe its characteristics.

- **SHP file format**

Shapefile is a vector data format for storing geographical data and associated attribute information. It is developed and regulated by Esri as an open specification for data interoperability among Esri and other GIS software products. Shapefile can be a point, line, or polygon feature such as:

Point Features: Well, Post Office, Temple, Hospital, Mosque, School.

Linear Features: Road, River, Highways, Rail track, Street, coastlines.

Areal Features: Pond, Soil Type, Lake, Reserved Forest, political boundaries, state or county boundaries, climate zones. Shapefile consists of several supporting files. There are three essential files, i.e., the main file that contains the feature geometry (.shp), an index file that stores the index of the feature geometry (.shx), and a dBASE table (.dbf) that stores the attribute information of features.

The aim of this transformation and merging is for deployment to obtain one data source that any GIS map provider can support; both data sources OSM and SHP are converted to GeoJSON.

### 3.2.2.1.1 Converting the SHP dataset to GeoJSON.

Converting the SHP to GeoJSON is necessary to generate the default system dataset. Utilizing the open-source third-party JavaScript-based library, we can convert from Shapefile to GeoJSON in just a few simple steps. The library we used is called ShpJS and can be found in the source code. The library collects the building's layout features from the \*.shp file in Figure 3-13, the attribute information of features from the \*.dbf file in Figure 3-12, and the data map projection settings from the \*.prj file to create the GeoJSON. The GeoJSON file can then be saved as a workspace to be used to merge in the final stage of data pre-processing.

The dashboard provides a user-friendly tool interface for converting Shp to GeoJSON. Using the prepared data can also be done manually if preferred for later data upgrades; when more building geometries are provided, the user can follow the same data pre-processing approach. Once we have our Shapefile in GeoJSON format, the next stage is converting the OSM to GeoJSON.



Figure 3-11 Converting Shp to GeoJSON UI.

The data source of the SHP file represented the characteristics of the building, which is provided by Property Assessment from Montreal.ca [34]

Key	Value
ANNEE_CONS	1982
CATEGORIE_	Regulier
CIVIQUE_DE	987
CIVIQUE_FI	987
CODE_UTILI	4890
ETAGE_HORS	1
LIBELLE_UT	Autres services publics (infrast...
NOM_RUE	rue de la Commune Ouest (MTL)

Figure 3-12 Default SHP Data Source

Building Attributes

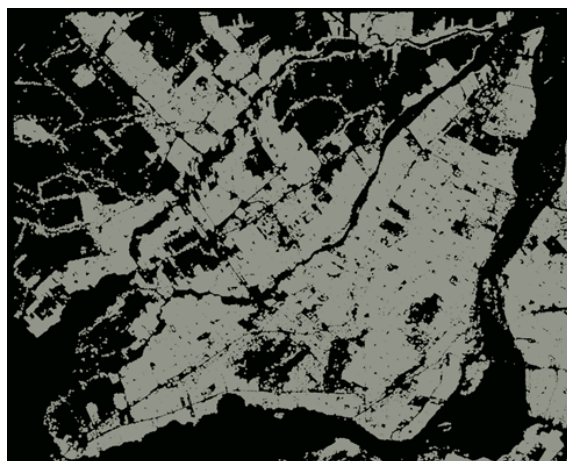


Figure 3-13 Overall SHP Data Source

Buildings Layout.

The meaning of each of the building characteristics is as follows:

Table 1 Attribute Variables Description [30]

Attribute Name	Description
NOM_RUE	Street name
ANNEE_CONSTRUCTION	Year of construction
CATEGORIE_UEF	Evaluation unit category (Regular or Condominium)
ETAGE_HORS_SOL	Maximum number of floors

### 3.2.2.1.2 Converting the OSM dataset to GeoJSON

The second data source is the OSM dataset, generated using google by the previous study [35]. There are many reasons to convert an OpenStreetMap (OSM) dataset into GeoJSON. OSM is an excellent dataset for mapping, but its lack of structure makes it challenging to work with it. GeoJSON is a JSON-based format that is more suited for storing spatial data. First, we need to clean up the data, and we can remove duplicates and unnecessary objects by using the clean function.

Next, we convert the data into a spatial format. The easiest way to do this is to extract the layout and attributes of the building, and based on the OSM specification, the building element is implemented as a "way" XML element. As shown in the following figure, each "way" element

includes the "nd" element and the "tag" element. The "nd" represents the point (latitude, longitude). In contrast, the "tag" elements represent the attribute name and value associated with the feature. In the example below Figure 3-14 of building representation in OSM file, it includes 12 "nd" elements, as mentioned before, it indicates the point geometry (latitude, longitude), and includes 3 "tags" representing the key and value attribute for the building feature. The first and the last "nd" is identical, and the criteria for valid polygons are defined in the OGC's Simple Feature standards document [36] and adhered to by the majority of GIS software and spatial databases. The reason for requiring the start points and end points to match is likely to relate to the topological concept of a closed set.

```

▼ <way id="21347837" visible="true" version="4" changeset="37423384" timestamp="2016-02-24T22:21:34Z" user="Ozvelpoon" uid="3633084">
  <nd ref="229598005"/>
  <nd ref="4024427991"/>
  <nd ref="1231398317"/>
  <nd ref="1231398279"/>
  <nd ref="1231379329"/>
  <nd ref="213716043"/>
  <nd ref="1231379347"/>
  <nd ref="4024427777"/>
  <nd ref="1231379328"/>
  <nd ref="1231379338"/>
  <nd ref="229598000"/>
  <nd ref="229598005"/>
  <tag k="building" v="yes"/>
  <tag k="building:levels" v="7"/>
  <tag k="name" v="building0"/>
  v="yes"/>
</way>

▼ <node id="229598005" visible="true" timestamp="2008-01-15T17:57:56Z" lat="45.4979288" lon="-73.5789404">
  <tag k="created_by" v="JOSM"/>
</node>

```

Figure 3-14 Example of 12 Nodes for a Building Representation OSM File

The rules for a valid polygon are [36]:

- Polygons are topologically closed
- The boundary of a Polygon consists of a set of LinearRings that make up its exterior and interior boundaries.
- No two Rings in the boundary cross and the Rings in the boundary of a Polygon may intersect at a Point but only as a tangent.
- A Polygon may not have cut lines, spikes, or punctures.
- The interior of every Polygon is a connected point set.
- The exterior of a Polygon with one or more holes is not connected. Each hole defines a connected component of the exterior.

- The exterior of a Polygon with one or more holes is not connected. Each hole defines a connected component of the exterior.



Figure 3-15 Example of Valid Polygons [36]

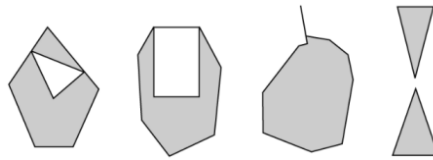


Figure 3-16 Example of Invalid Polygons [36]

The following figures, Figure 3-17 and Figure 3-18 show the result of converting the OSM building to the GeoJSON polygon feature.



Figure 3-17 Example for 12 Points Building Layout.

```

{
  "type": "FeatureCollection",
  "features": [
    {
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              -73.5789404,
              45.4979288
            ],
            [2],
            [2],
            [2],
            [2],
            [2],
            [2],
            [2],
            [2],
            [2],
            [2],
            [2]
          ]
        ]
      },
      "type": "Feature",
      "properties": {
        "building": "yes",
        "building:levels": 7,
        "name": "building0"
      }
    }
  ]
}

```

Figure 3-18 Example for 12 Points after Conversion to GeoJSON Data Structure.

Similar to converting Shp to GeoJSON, The dashboard provides a user-friendly interface to convert the OSM to GeoJSON. Moreover, if it is preferred for later data upgrades; when more building

geometries are provided, data preparation can also be done manually; the user can follow the same data pre-processing approach. Once we have our OSM in GeoJSON format, the next stage is merging the GeoJSON files.



Figure 3-19 Converting OSM to GeoJSON UI.

### 3.2.2.1.3 Merging the GeoJSON files

Building layouts and building characteristics datasets have been merged to create the default system dataset for Montreal. This dataset is based on the OpenStreetMap data and includes building outlines, heights, and other essential information about buildings in Montreal. The Buildings Characteristics dataset was collected by the City of Montreal from property owners and developers to merge the GeoJSON files of the OpenStreetMap and Buildings Characteristics dataset. Using Turfjs, Turfjs is a JavaScript library for spatial analysis. It includes traditional spatial operations, helper functions for creating GeoJSON data, and data classification and statistics tools. One of the most powerful features of Turfjs is its ability to merge GeoJSON features. The user can easily combine different data types into a single feature. For example, the user can merge a point feature with a line feature to create a polygon. After merging the GeoJSON file data with scanned data, Using the map provider CLI we have deployed it to the map provider database, The CLI to manipulate, create and publish the tile set and use the map database.



Figure 3-20 Example of Turfjs Polygons Intersection



### 3.2.2.2 Data Conversion and Preparation modules

Data conversion and data preparation modules are essential for any academic and research project; It is an integral part of data analysis and GIS mapping. They allow users to transform data formats and merge data sets more easily and quickly. There are many different types of data conversion and GIS data preparation modules available, and the most common are QGIS, ArcGIS, and GeoTools. Our system provides a versatile and powerful data conversion module that can be used to transform different types of files and convert them to other formats and export data to a geodatabase. The module is easy to use and provides many features, including support for different data types, map projections, and coordinate systems. That allows users to easily prepare data as input datasets for some analysis simulation model applications and create meaningful maps that can be easily interpreted and analyzed by all stakeholders. The module can also convert multiple datasets into a single data set to simplify the workflow and enable the user to save time and effort. In our study, the GeoJSON is the main file extension for geographic data and the most commonly used file format for geographic data. Moreover, the GeoJSON file format is widely accepted as the most standardized and effective way to store geographic data. The system provides converting from SHP, OSM, and CSV to GeoJSON file format and from GeoJSON to SHP and STL file format.

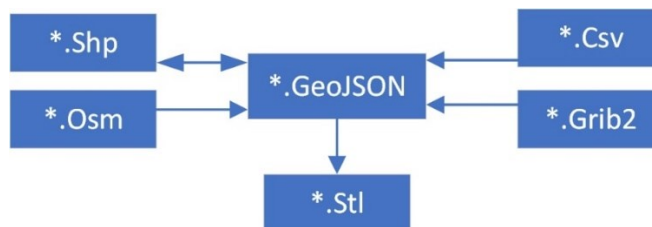


Figure 3-21 Mind Map of Supported Conversion Extensions

The modules' purpose is to decrease the complexity of data conversion, improve geographic data management efficiency, and further data manipulation and analysis. One of the most critical issues is the converting process between file formats. Most scholars and urban researchers spend much more time studying and configuring the software tools for each specific data format; with the tools implemented in our framework, they can convert their data in one step without any effort.

For instance, the CityFFD simulation and CityBEM simulation workflow shows that the data preparation is focused on the spatial data, so in our framework, we try to automate most of the data preparation tasks, including converting between different file formats and adding or merging

geographic features. As input for simulation and modeling applications. The following section explains generating the 3D model approach, which is achieved by converting the spatial data to STL.

- **Converting the spatial data to STL**

The STL file format is a 3D model representation and is essential for CFD and BES urban simulations. However, generating a city-scale 3D model is not easy. Merge a group of files representing vegetation, people, bridges, buildings, and terrain and export a unified 3D urban geometry in STL. In our study, we proposed and implemented an efficient way for 3D model generation. By merging different layers representing vegetation, people, and other geographical features into a single 3D model and enriching the building information data collection, we could create a more accurate representation of the urban environment.

In the previous section, we have shown the importance of the GeoJSON file format. The figure below shows our study's general applied workflow for converting the GeoJSON data set to the STL 3D model format. The GeoJSON data set is first read, and the necessary geometries information is extracted. These geometries are then used to create a vertex in the 3D model format by converting the data geometric to metric and applying the triangulation process. The triangulation process is implemented using a third-party JavaScript library called Ear clipping [37]fter triangulating the data metric for all GeoJSON features, we can generate STL facets to build the STL solid.

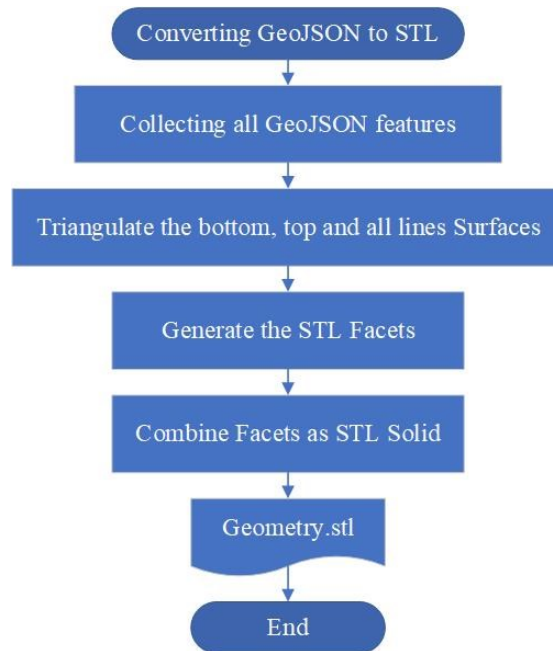


Figure 3-22 Converting GeoJSON to STL Workflow

Integrating the terrain data in the 3D model STL process starts with receiving four points from the user input and then using Turfjs to calculate the bounding box from these four points [38]. These four points are used to query and collect the buildings' features as GeoJSON data structure by query and collect the features from the Map component. In our case, we are using Mapbox as a map provider [39] The map provider supports the buildings querying feature that receives the bounding box as input and returns all the features in the bounding box. Then the system must know the elevations of all grid points, so we created a grid point for this purpose, and the default cell width for the grid points is 250 meters. We can query and collect the elevation data using the same map provider querying feature API. Next, taking the coordinates of the grid points and the elevation of each point, we can generate a triangles mesh; the triangles mesh is generated using Turfjs, then updating the building elevation is based on the intersection with the triangles.

Figure 3-23 shows the 3D model with terrain generated from a spatial dataset collected from the GIS map provider.

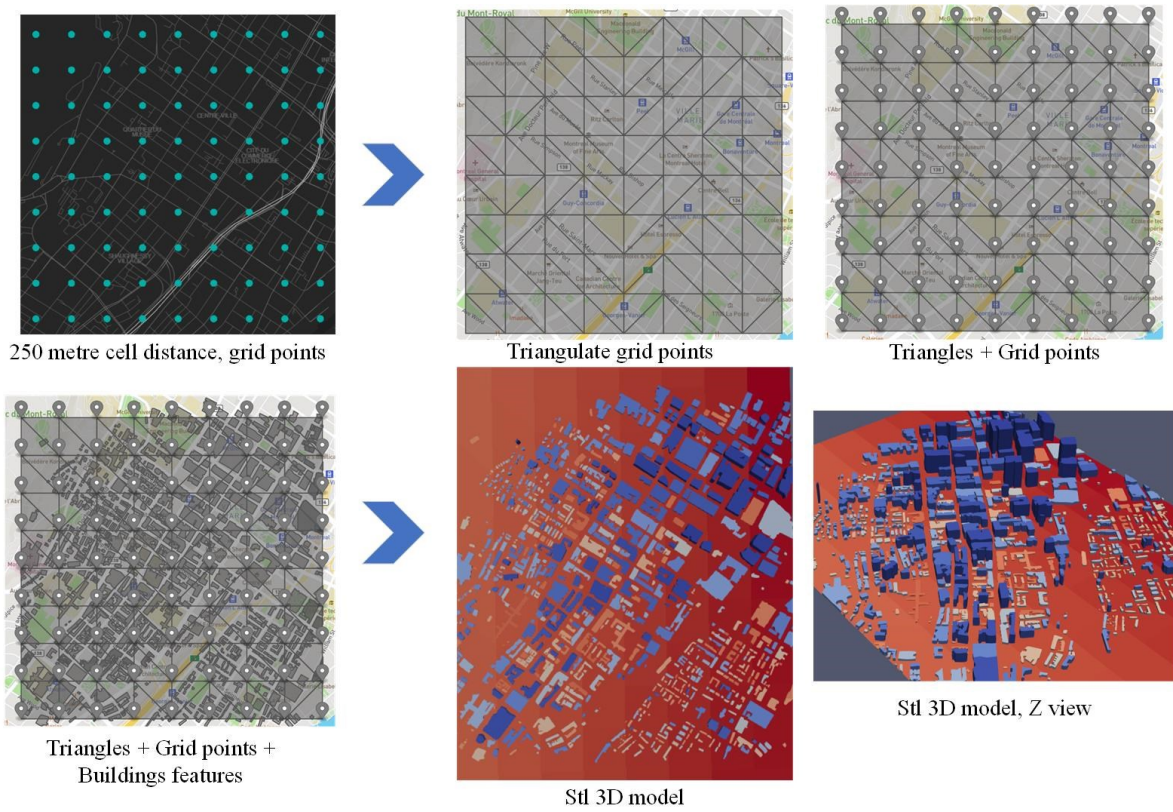


Figure 3-23 Merging Phases of the Terrain Spatial Data.

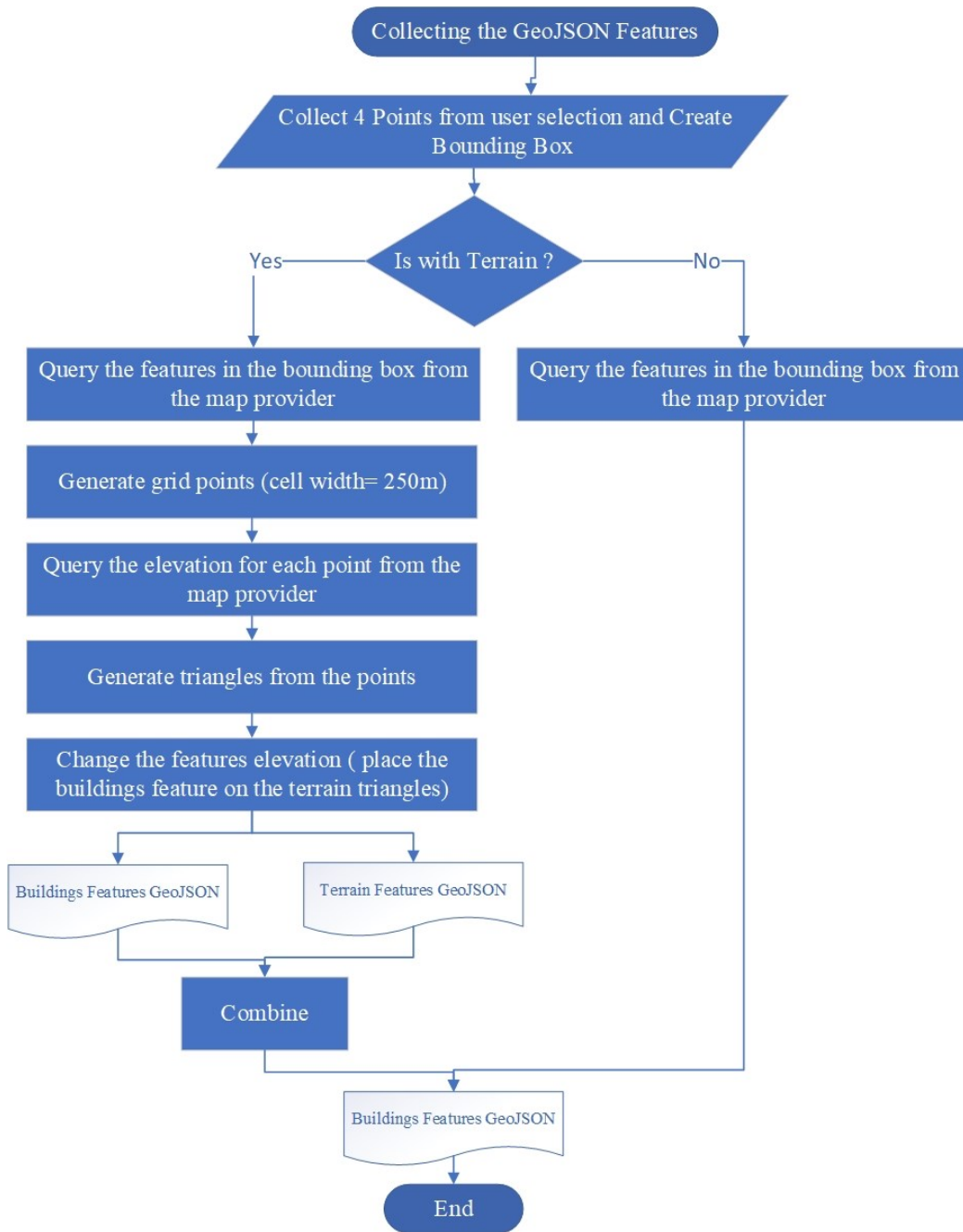


Figure 3-24 Integrating the Terrain Spatial Data with 3D Model STL Workflow

The default final STL file center coordinate is (0,0,0),

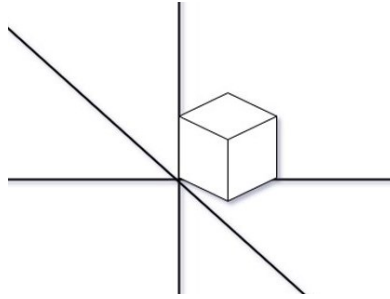


Figure 3-25 3D Model Coordinate Plane

The dashboard provides functionality for the user to be able to customize the center of coordinates as shown below.



Figure 3-26 Customize the 3D Model Coordination Origin

As part of increasing the availability of the dataset and not depending on the map provider for all extraction time, the dashboard provides a sectional pre-defined data for download, with the terrain layer included.

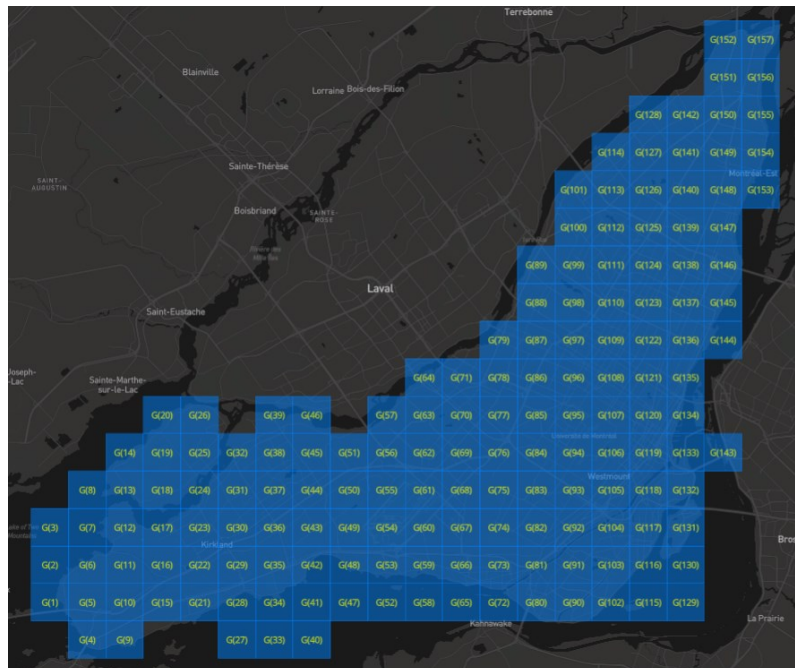


Figure 3-27 Pre-defined Backup Sectional Data Collection.

### **3.2.3 Urban Modeling Data Input Generation Support Module**

The system has implemented an additional module that also makes it possible to extract and generate the data input collection for feeding the BES and CFD simulation; two groups of input data are considered for BES simulation: buildings geometry [5], i.e., external wall area, net volume, building height. The geometries can be extracted from GIS (Geographic Information Systems) [40],[41] CityGML[42], BIM (Building Information Modeling)[43], CAD (Computer Aided Design) [44] [45], or digital images[46][47]. Moreover, the second is the building information construction year and building type, envelope material thermal properties, and schedule. At the same time, the CFD input data generation module is where complex CFD models are converted into ready-to-use input datasets. The process allows fast and easy simulation, helping export the geometry of real-world buildings, and extracting the building geometry of urban-scale simulation. The CFD simulation is not only for urban scale but also for building performance simulation.

The system can help users to automate the process of data collection, which helps to save time and increase accuracy. The system provides pre-defined data collection as an internal resource (the default system data collection) and an external resource as user-defined input layers. The generation process of simulation input starts by selecting the bounding box of buildings ' boundaries (box width and length). The boundary calculation is based on the input elevation and building footprint, which defines the desired area for each output layer; it is typically a rectangle surrounding the study area (e.g., a city block).

The bounding box was used for the extraction of external and internal building geometry in the selected study area. The internal resource is the default system data, and the building information is collected from it. In contrast, the external resource is user-predefined layers (for example, the external GeoJSON, STL, OSM, and Terrain) Figure 3-28.





Figure 3-28 Dashboard, BES, and CFD Data Collection Generation Module UI

The dashboard allows users to customize the bounding selection settings instead of the polygon selection tool. Figure 3-29

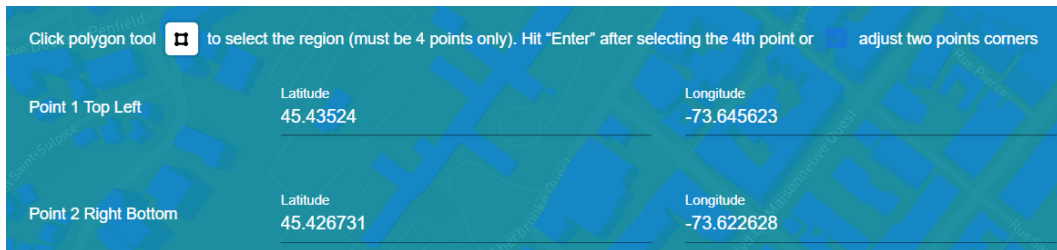


Figure 3-29 Dashboard, Customize the Bounding Box Selection UI.

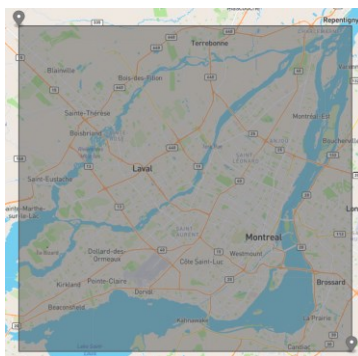


Figure 3-30 Example of Two Custom Selection Points (Top Left, Bottom Right)

For the BES input group, the dashboard provides functionality for users to select from the available building information to be extracted from the system layers.



For example, the collected building information from external resources Figure 3-32. As shown in the example, the building information file includes additional columns: “Building\_stl” and “Building\_osm”. The “Building\_stl” refers to the solid name in the generated 3D model STL.



Figure 3-31 Dashboard, Customize the Building Information UI.

Building_stl	Building_osm	year	type
b0	0	1966	Condominium
b1	1	1962	Régulier
b2	2	1962	Régulier
b3	3	1956	Régulier
b4	4	2013	Condominium
b5	5	1990	Régulier
b6	6	1830	Condominium
b7	7	1958	Condominium
b8	8	2010	Régulier
b9	9	1986	Régulier
b10	10	1986	Régulier
b11	11	1988	Condominium
b12	12	1968	Régulier
b13	13	1929	Régulier
b14	14	1900	Régulier
b15	15	2003	Régulier
b16	16	1955	Régulier
b17	17	1962	Régulier
b18	18	1990	Régulier
b19	19	1909	Régulier
b20	20	1962	Régulier
b21	21	1990	Régulier
b22	22	2001	Régulier

Figure 3-32 Example of Building Information Data Collection Result

While the “Building\_osm” refers to the geometric building data feature index, the system, by default, will export a GeoJSON file representing the geometric buildings data features, including an id matching the Building\_osm id used for visualization purposes once the simulation result is ready.

Similarly, CFD data collection generation is part of the tool functions with selecting an area that defines the target area's bounding box. Once the target area is defined, the user can gather the necessary 3D model data for CFD simulation.

The system provides two types of CFD data input preparation modules; the first is for indoor (building or room-scale) using the Fatima-CFD web application, and the second is for outdoor (urban and landscape scale) using the core web application; the user can choose the appropriate tool for their data set.

The original CFD model includes input files for boundary conditions, mesh, particle information, and geometrical objects. The first data collection tool using a core web application (outdoor) can export the geometry, domain settings, and other parameters as CFD simulation Figure 3-28. The second data collection generation tool using the Fatima-CFD web application (indoor), the Fatima-CFD web application is explained in detail below.

### 3.2.4 Urban Modeling Visualization Environment Support Modules

This section will discuss the dashboard visualization modules for BES and CFD result data output; it is a powerful module for analyzing. City-scale visualization. It is well-known that city-scale monitoring and visualization are essential to managing urban simulations, mainly the dashboard uses the Mapbox GL for building web maps, visualizing and animating geographic data, querying and filtering features on a map, dynamically displaying and styling custom client-side data on a map, 3D data visualizations and animations, adding markers and popups to maps programmatically.

#### 3.2.4.1 BES Data Visualization Module.

Building energy simulation (BES) is a computer modeling technique used to evaluate the effects of building design and operation on energy use. BES data can be used to improve building efficiency and occupant comfort. Our framework provides a powerful and intuitive way to analyze BES data. By mapping the simulation at the city-scale level, we can compare the impacts of different design strategies or building characteristics at the community level using graphical depictions of building footprints and annual energy consumption. Our visualization support module allows users to explore and analyze BES data in various ways and to explore data and make inferences. With about 36 color legend schemes to choose from, the user can easily find the best color scheme for data applications Figure 3-33.



Figure 3-33 Dashboard, Supported Legends Colors.

Figure 3-36 shows the BES visualization control panel and an example of an imported user, BES simulation result for about 12500 buildings Figure 3-34. Suppose the simulation is for the external area (not in the default system data region “Montreal” ). In this case, the user can import the GeoJSON file pre-generated during the data input generation stage.

	A	B	C	D	E	F
1	building	building_osm_id	Thermal_load(W)	Thermal_load_A(W/m2)	Ts_ext(C)	h_conv_out(W/m2K)
2	timeloop0					
3	b0	379146	-222872	-28.4823	11.1713	34.6126
4	b1	410787	-110131	-35.2681	11.2032	21.9995
5	b2	119228	-449011	-5.21827	11.1993	17.0874
6	b3	119228	-406187	-4.72058	11.2014	13.734
7	b4	16121	-627414	-5.91933	11.2104	13.4737
600743	b12488	284369	-10937.8	-9.84759	17.81	10.0369
600744	b12489	283466	-9456.11	-11.0533	16.2032	11.0958
600745	b12490	283473	-2137.52	-7.90392	15.7283	8.02803
600746	b12491	285284	-6310.25	-11.3384	18.3863	9.46014
600747	b12492	285282	-9313.15	-10.2564	15.9368	12.2776
600748	b12493	10057	23949.5	8.68055	17.8595	7.18029
600749	b12494	10051	-5572.16	-15.2862	14.672	5.73402
600750	b12495	10962	-7641.13	-10.2468	16.4832	7.22059
600751	b12496	10960	-8187.23	-10.4688	16.1655	8.12994
600752	b12497	10975	-5307.71	-11.802	17.8801	9.46866
600753	b12498	10973	-7983.22	-3.65054	14.9427	11.2454
600754	b12499	286954	-1198.75	-1.69729	16.9365	8.14774
600755	b12500	286156	-17151.1	-10.4463	17.7593	9.04015
600756	b12501	12707	-5216.04	-12.8913	18.3243	16.458
600757	b12502	12706	-4616.95	-7.38698	17.3101	12.1821
600758	b12503	287852	-4391.11	-7.78285	17.2845	11.2178
600759	b12504	287851	-5333.54	-4.39685	14.729	13.4866
600760	b12505	390812	-5344.72	-9.4815	18.1727	5.70453
600761	b12506	11871	-17021	-9.39659	18.0178	9.49196
600762	b12507	11867	-6676.51	-10.4725	17.6713	12.8956

Figure 3-34 Example of BES Result for 12500 Buildings

In the analyzing data output, the tool follows the CityBEM result custom file structure, composed of three major parts (header, time loops, and data) Figure 3-35.

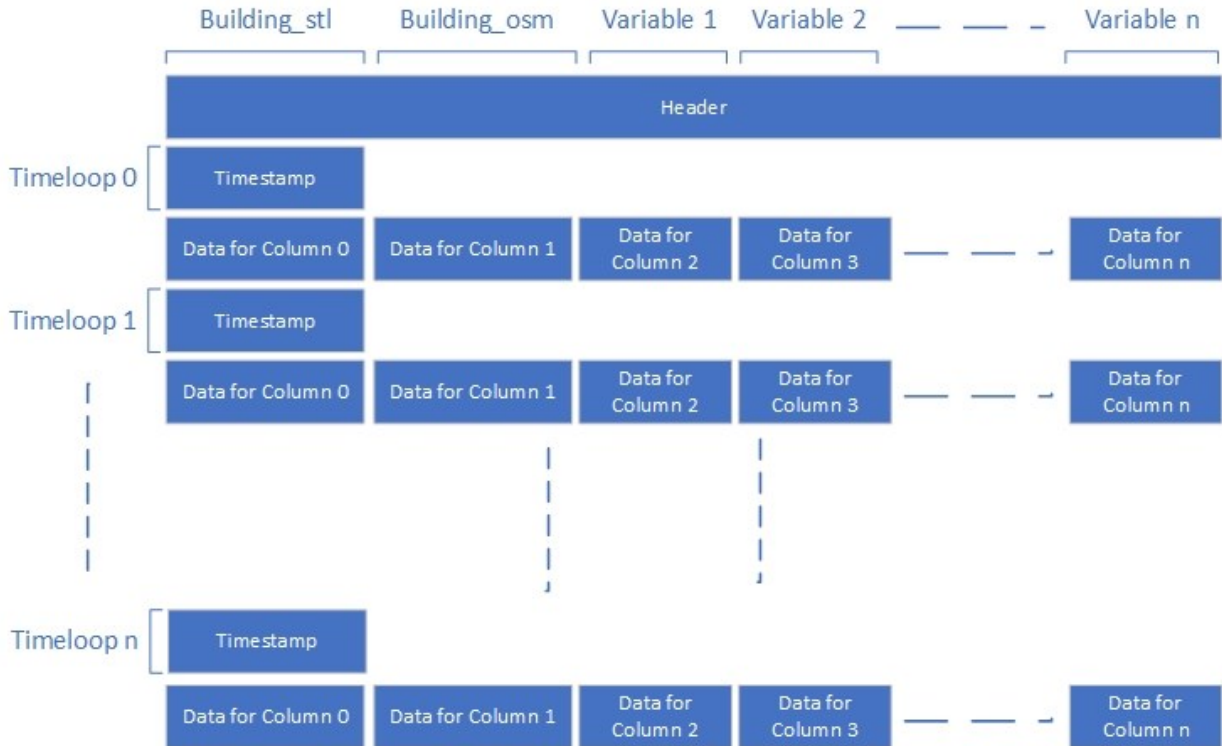


Figure 3-35 BES Data Output Collection File Structure

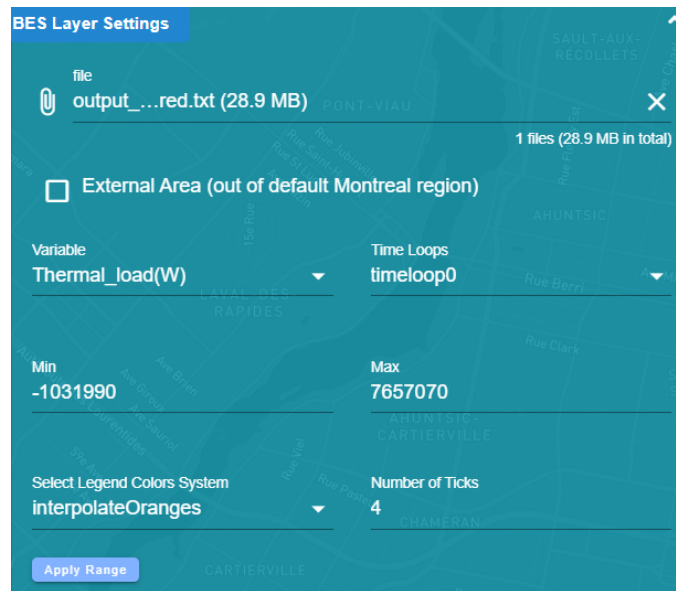


Figure 3-36 BES Visualization Control Module UI

Once the user uploads the resulting output file, the system analysis the result file and dynamically loads the data variables list and timestamp list Figure 3-37. Moreover, the data's maximum and minimum values are in the block of the selected variable at the specific timestamp. The tools

allow the user to change the minimum and maximum for discarding the outliers on the custom range.

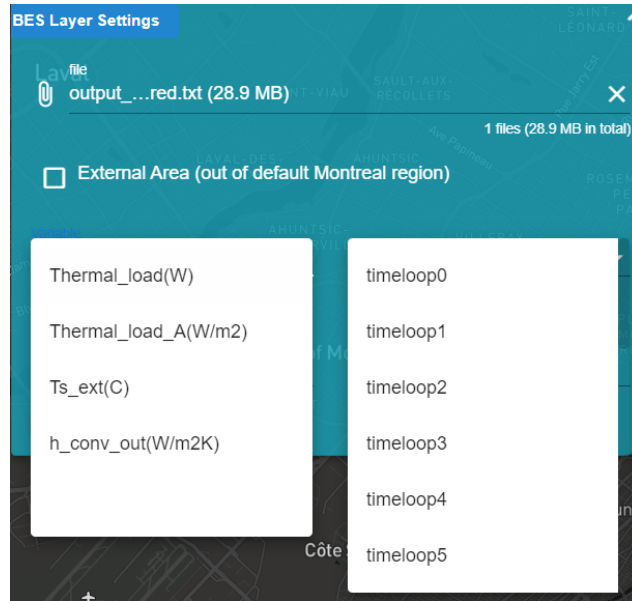


Figure 3-37 BES Visualization Module, Variable, and Timestamp.

The Color legend is generated after loading and parsing the result file, depending on the minimum and maximum values in the result file.



Figure 3-38 BES Legend

The module also provides trend analysis for all the variables defined in the loaded BES data output file.

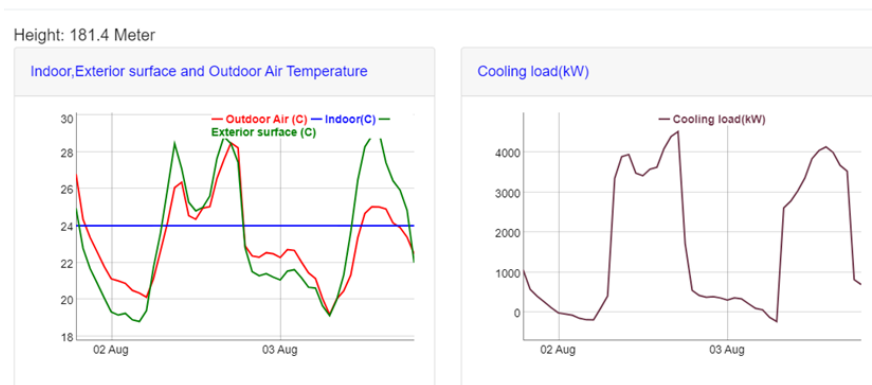


Figure 3-39 BES- Trend Variables Analysis

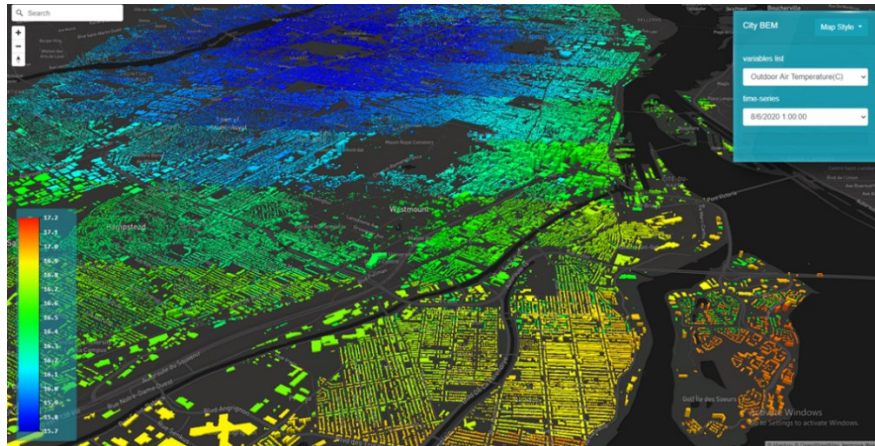


Figure 3-40 City-scale Interactive 3D Map BES Data Collection Output Visualization



Figure 3-41 City-scale Interactive 2D Map BES Data Collection Output Visualization

### 3.2.4.2 CFD Data Visualization Module

As mentioned previously, CFD is a powerful technique used to study the flow of air, water, or other fluid substances. It can be used to simulate the movement of these substances in a controlled and realistic environment. To visualize CFD data, we use different types of charts and graphs. The



streamlined graph is a standard chart used to visualize CFD data. The streamlined graph comprises a set of curves (which represent the fluid flow) connecting two fixed points.

The second type uses a 3D fluid particle rendering system and streamlines data with different colors. Moreover, the last step is to use volume rendering to show the result of the CFD simulation.

The tools support the standard ASCII \*.vtk file structure Figure 3-42.

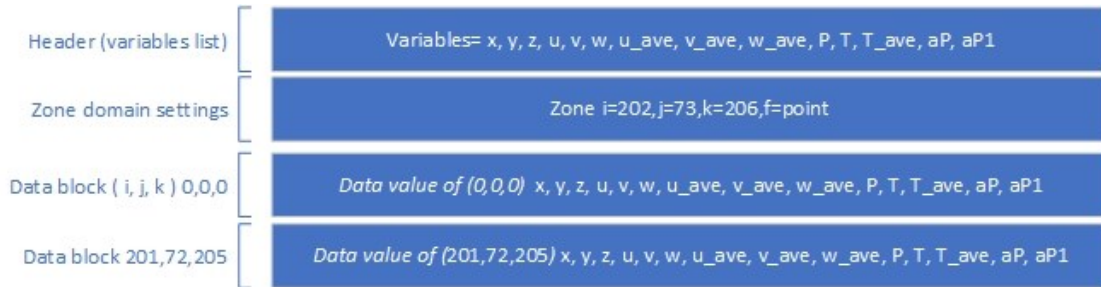


Figure 3-42 VTK Standard Data Structure, Mesh Size (202, 73, 206)

	A	B	C	D	E	F	G	H	I	J	K	L
1	variables=x, y, z, u, v, w, u_ave, v_ave, w_ave, P, T, T_ave, aP, aP1											
2	Zone i=202,j=73,k=206,f=point											
3	-2748.45336914 -13.63379955 -2798.59814453 20.00000000 0.00000000 0.00000000 20.65999985 0.00000000 0.00000000 0.00000000 35.00000000 34.99663544 0.00000000 1.00000000											
4	-2721.10595703 -13.63379955 -2798.59814453 -19.61542892 -0.26232544 0.63840300 -18.97047615 -0.18844171 0.73861521 0.00000000 35.00000000 34.99663544 0.00000000 0.98077148											
5	-2693.75781250 -13.63379955 -2798.59814453 -18.98960495 -0.43707249 1.09693968 -18.49575043 -0.31406483 1.23177600 0.00000000 35.00000000 34.99663544 0.00000000 0.94948024											
6	-2666.41064453 -13.63379955 -2798.59814453 -18.18736458 -0.52056545 1.35097885 -17.86414909 -0.37362394 1.55367410 0.00000000 35.00000000 34.99663544 0.00000000 0.90936822											
7	-2639.06250000 -13.63379955 -2798.59814453 -17.33471298 -0.53319341 1.41953349 -17.21642494 -0.38116369 1.73163390 0.00000000 35.00000000 34.99663544 0.00000000 0.86673570											
8	-2611.71508789 -13.63379955 -2798.59814453 -16.50520134 -0.49989229 1.35239708 -16.58424377 -0.35401779 1.82207096 0.00000000 35.00000000 34.99663544 0.00000000 0.82526004											
9	-2584.36718750 -13.63379955 -2798.59814453 -15.74455357 -0.44237542 1.19767714 -16.01634026 -0.30785409 1.85874832 0.00000000 35.00000000 34.99663544 0.00000000 0.78722769											
10	-2557.01928711 -13.63379955 -2798.59814453 -15.07174015 -0.37619674 1.00258565 -15.52211094 -0.25453925 1.87350166 0.00000000 35.00000000 34.99663544 0.00000000 0.75358701											
11	-2529.67163086 -13.63379955 -2798.59814453 -14.49234200 -0.31125966 0.80350465 -15.10936451 -0.20226303 1.88557816 0.00000000 35.00000000 34.99663544 0.00000000 0.72461712											
12	-2502.32373047 -13.63379955 -2798.59814453 -14.00109100 -0.25291193 0.62420917 -14.77003670 -0.15583922 1.90671146 0.00000000 35.00000000 34.99663544 0.00000000 0.70005453											
13	-2474.97583008 -13.63379955 -2798.59814453 -13.59071541 -0.20330814 0.47743750 -14.49653339 -0.11728880 1.94373012 0.00000000 35.00000000 34.99663544 0.00000000 0.67953575											
14	-2447.62792969 -13.63379955 -2798.59814453 -13.25092316 -0.16321294 0.36352038 -14.27736855 -0.08708368 1.99745357 0.00000000 35.00000000 34.99663544 0.00000000 0.66254616											
15	-2420.28027344 -13.63379955 -2798.59814453 -12.97182655 -0.13183135 0.27877083 -14.10232830 -0.06429200 2.06734848 0.00000000 35.00000000 34.99663544 0.00000000 0.64859134											
16	-2392.93212891 -13.63379955 -2798.59814453 -12.74289131 -0.10817553 0.21476699 -13.96097660 -0.04777673 2.14924812 0.00000000 35.00000000 34.99663544 0.00000000 0.63714457											
17	-2365.58447266 -13.63379955 -2798.59814453 -12.55524254 -0.09038562 0.16637817 -13.84475136 -0.03609697 2.24082232 0.00000000 35.00000000 34.99663544 0.00000000 0.62776214											
18	-2338.23632813 -13.63379955 -2798.59814453 -12.40044594 -0.07701912 0.12806071 -13.74546719 -0.02825136 2.33803630 0.00000000 35.00000000 34.99663544 0.00000000 0.62002230											
19	-2310.88867188 -13.63379955 -2798.59814453 -12.27208042 -0.06644505 0.09912611 -13.65752125 -0.02304014 2.43977237 0.00000000 35.00000000 34.99663544 0.00000000 0.61360401											
20	-2283.54052734 -13.63379955 -2798.59814453 -12.16489315 -0.05800109 0.07597924 -13.57638264 -0.01986487 2.54327869 0.00000000 35.00000000 34.99663544 0.00000000 0.60824466											
21	-2256.19287109 -13.63379955 -2798.59814453 -12.07595730 -0.05079316 0.06411052 -13.49910259 -0.01803713 2.64846754 0.00000000 35.00000000 34.99663544 0.00000000 0.60379785											
22	-2228.84497070 -13.63379955 -2798.59814453 -12.00279045 -0.04469940 0.05631624 -13.42314339 -0.01729218 2.75339150 0.00000000 35.00000000 34.99663544 0.00000000 0.60013950											
23	-2201.49707031 -13.63379955 -2798.59814453 -11.94422817 -0.03918380 0.05410433 -13.34719276 -0.01716799 2.85846424 0.00000000 35.00000000 34.99663544 0.00000000 0.59721142											
24	-2174.14916992 -13.63379955 -2798.59814453 -11.89878273 -0.03434695 0.05479737 -13.26981735 -0.01757570 2.96210980 0.00000000 35.00000000 34.99663544 0.00000000 0.59493911											
25	-2146.80151367 -13.63379955 -2798.59814453 -11.86574173 -0.02980130 0.05869290 -13.19034386 -0.01819451 3.06508589 0.00000000 35.00000000 34.99663544 0.00000000 0.59328711											
26	-2119.45336914 -13.63379955 -2798.59814453 -11.84376717 -0.02575938 0.06402648 -13.10806370 -0.01903376 3.16609550 0.00000000 35.00000000 34.99663544 0.00000000 0.59218836											
27	-2092.10571289 -13.63379955 -2798.59814453 -11.83215714 -0.02192260 0.07208835 -13.02299213 -0.01981322 3.26609969 0.00000000 35.00000000 34.99663544 0.00000000 0.59160787											
28	-2064.75781250 -13.63379955 -2798.59814453 -11.82955933 -0.0187548 0.08181382 -12.93470669 -0.02060436 3.36398363 0.00000000 35.00000000 34.99663544 0.00000000 0.59147799											
29	-2037.40991211 -13.63379955 -2798.59814453 -11.83525562 -0.01546921 0.09477352 -12.84356594 -0.02121103 3.46092248 0.00000000 35.00000000 34.99663544 0.00000000 0.59176278											

Figure 3-43 Example of CFD Simulation Result for Mesh Size (202, 73, 206)

The user begins visualization by importing the vtk data result and the position setting file. The position setting file is one of the files generated during the data input preparation and refers to the center of the selected simulation domain.

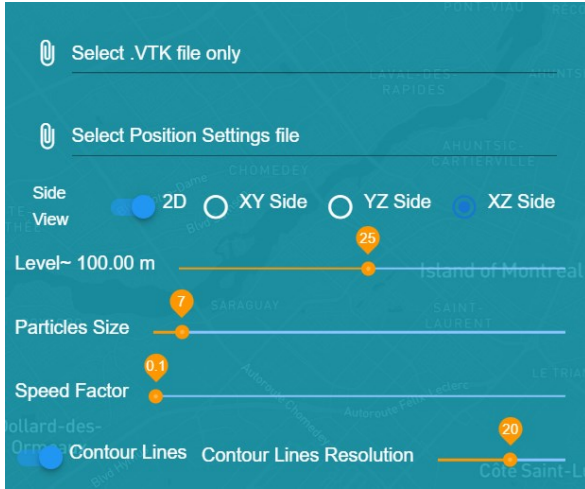


Figure 3-44 Dashboard CFD Visualization Module UI

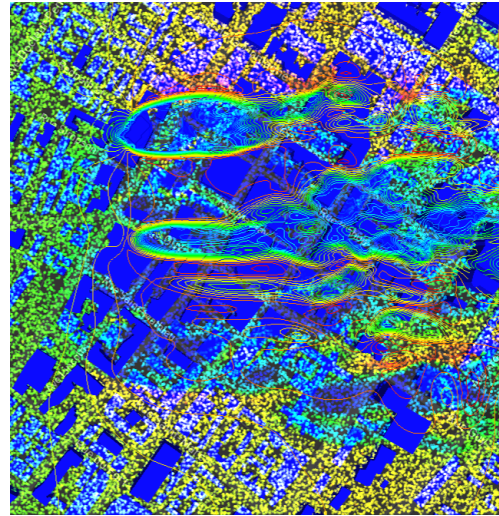


Figure 3-45 CFD Visualization XZ side

The tool provides different viewpoints in 2D and 3D, an interactive tool to visualize all supported levels based on the vtk source file, adjust the particles' size and speed factor, and provide the functionality to visualize the contour lines for that selected data level.

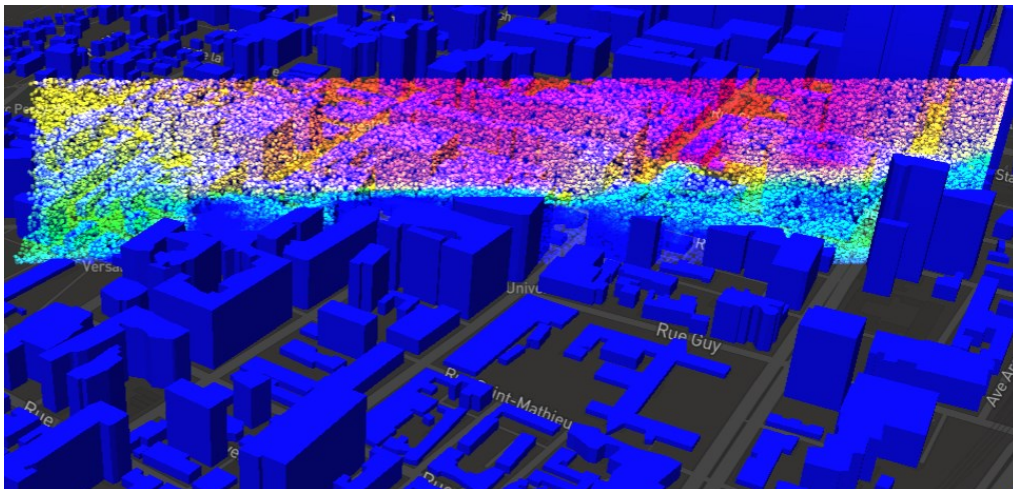


Figure 3-46 CFD Visualization XY side



### **3.2.5 Application Module - City Reduced Probability of Infection (CityRPI)**

This section introduces the City Reduced Probability of Infection (CityRPI) application module implementation and structure design. The CityRPI is a real-time interactive application module for the public, it is to compare various strategies to curb indoor airborne transmission of COVID-19 in different archetype buildings at a city scale. The RPI Model is developed by other of our laboratory members and motivated to address the building-specific problem, such as the impact of mitigation strategies should be studied separately for each building to find the best strategy depending on the building condition. Also, to update the current ventilation standards, the required minimum ventilation rate to control the airborne infection risk should be investigated for each building type; however, many of the required input data for this model is provided and processed using this web application, and leveraging of implementing this model on this dashboard the building owner, manager, and engineer with providing the required information about their building, they can use the CityRPI to assess the infection risk and evaluate the mitigation strategies specific to his/her building [48]. The CityRPI application module can help identify high-risk areas, assess the impact of potential mitigation measures, and optimize the implementation of targeted interventions. The CityRPI application module can also be used to monitor the impact of interventions over time so that cumulative risks and benefits. The data input preparation steps are intended to be carried out by model developers and experts with intimate knowledge of their local context. The model input parameters are often related to cities, buildings, and occupancies, e.g., lockdown dates and rates, occupancy levels, age, sex, and exposure time. Building system-specific parameters are also important, including floor area or room size, outdoor air ventilation, recirculation rates, duct filter types, with/without air cleaners and their capacities, and mask types.

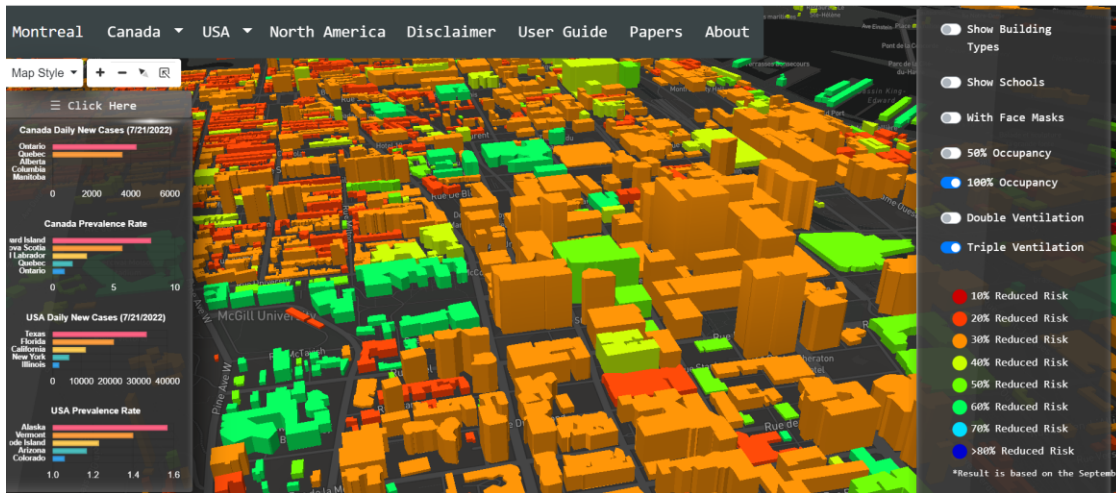


Figure 3-47 Application Module - CityRPI Dashboard

The following flowchart shows the third-party API and Map-based data integration steps; it begins by finding or generating the geospatial data layout such as GeoJSON, then finding or generating a relationship between the layout and the API query response data. The GeoJSON can have an attribute that is a good match with one of the API response attributes, for example, city name. In case there is no identical attribute. Hence, a dictionary must be defined as a link between the geospatial data and the API response. This way, we can have a relation and integrated output between the geospatial data with third-party API data input; this integration can feed any calculation model internally on the web application side or externally by exporting them as CSV data collection. The layout supports the visualization process through the map supporting result value-color translation. In general, most web applications rely on input from users. In the CityRPI, we integrated the pre-defined geospatial and API data that provide the daily Covid-19 new cases and historical data set. The inputs were also collected from the user. All these features were under the client's control and can be accessed from a front-end user interface.

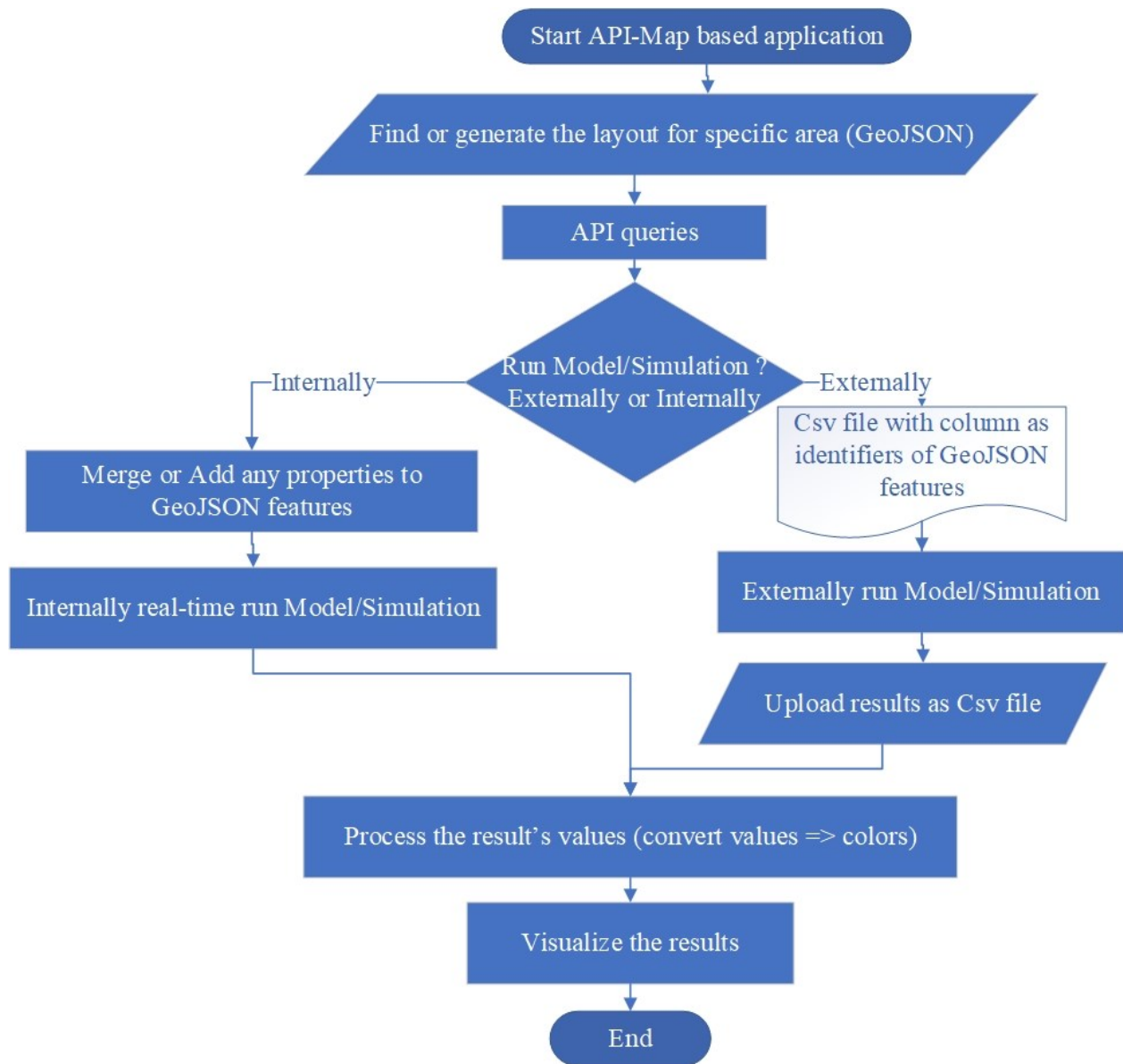


Figure 3-48 Third-Party API and Map-based Data Integration Workflow

The primary purpose of this integration is to create an API for querying covid-19 new cases and historical data integrated with geospatial data as input to feed the CityRPI model.

The following table shows the relationship or the link between the geometric layout and the API response, this example for the Quebec geometric layout Figure 3-49, the “CA-QC” in the GeoJSON properties used as an index in the dictionary to get the attribute that matches the API State attribute.

Table 2 Geometric Data Layout and Third-party API Integration Dictionary Definition.


Layout Properties (GeoJSON)	Dictionary declaration (JavaScript)	API data response (JSON Response)
<pre data-bbox="228 333 479 443">"properties": {   "ios": "CA-QC" }</pre>  <p data-bbox="245 884 505 989">Figure 3-49 Quebec Province GeoJSON Layout</p>	<pre data-bbox="594 354 1016 390">dic['CA-QC'] = 'Quebec';</pre>	<pre data-bbox="1062 338 1406 632">"Key": "canada_quebec", "Api": "https://kustom.radio", "Name": "Quebec, Canada", "NameFr": "Québec, Canada", "Country": "Canada", "CountryFr": "Canada", "State": "Quebec", "StateFr": "Québec", "Lat": "52.9399", "Long": "-73.5491",</pre>

Figure 3-50 shows that In the same way, we can apply that to all of North America's provinces  
Figure 3-51.

```
ISO_3166_2['CA-AB'] = 'Alberta';
ISO_3166_2['CA-BC'] = 'British Columbia';
ISO_3166_2['CA-MB'] = 'Manitoba';
ISO_3166_2['CA-NB'] = 'New Brunswick';
ISO_3166_2['CA-NL'] = 'Newfoundland and Labrador';
ISO_3166_2['CA-NS'] = 'Nova Scotia';
ISO_3166_2['CA-ON'] = 'Ontario';
ISO_3166_2['CA-PE'] = 'Prince Edward Island';
ISO_3166_2['CA-QC'] = 'Quebec';
ISO_3166_2['CA-SK'] = 'Saskatchewan';
ISO_3166_2['CA-NT'] = 'Northwest Territories';
ISO_3166_2['CA-NU'] = 'Nunavut';
ISO_3166_2['CA-YT'] = 'Yukon';
```

Figure 3-50 Example of a Dictionary Creation.



Figure 3-51 The Geometric Layout of North America

We have added a dynamic layer restful API for other client model calculation applications that require the same integration with third-party API in the core web application. The section “Application Programming Interface (API) Integration” explains the implementation detail. While for server model calculation applications, the framework also provided backend thi<sup>rd</sup>-party API integration to feed calculation models.

In the urban-scale model calculation, the client-side integration is more efficient in feeding real-time monitoring than the server-side integration.

**CityRPI web user interfaces:**



Figure 3-52 Map Provider Support the Value-Color Transformation

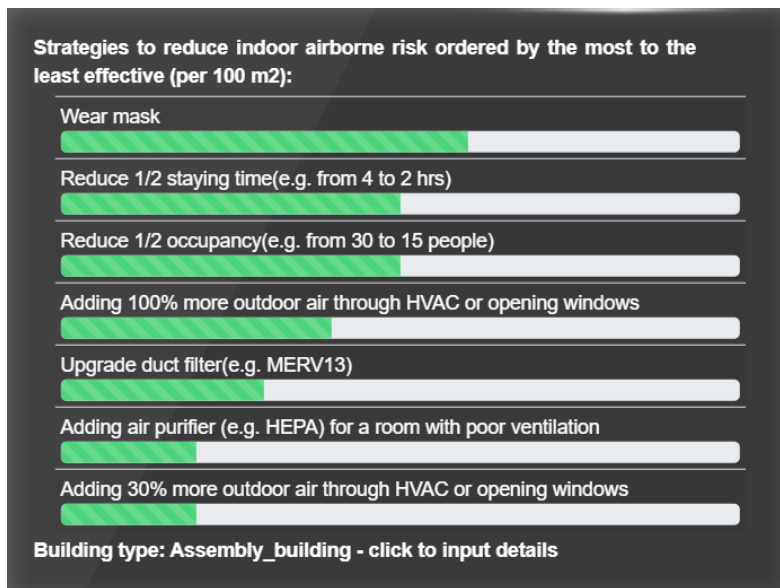


Figure 3-53 Application Module- CityRPI, Real-time Model Calculation Reporting.



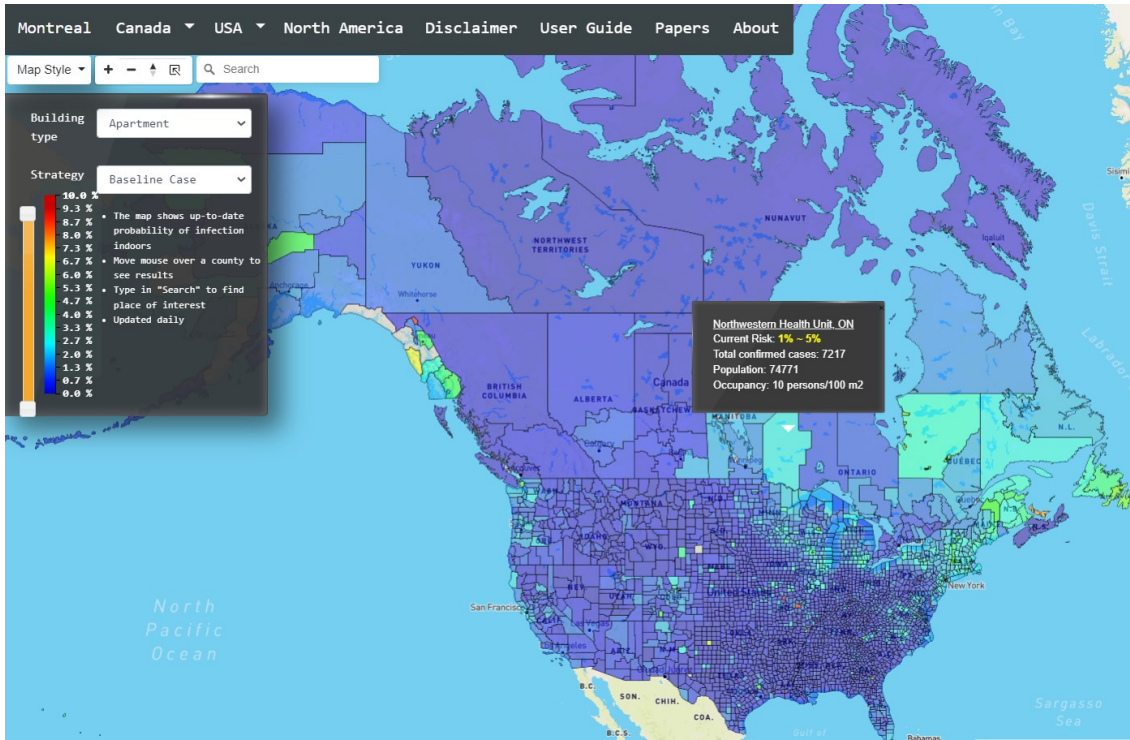


Figure 3-54 Application Module- CityRPI North America Real-time Monitoring

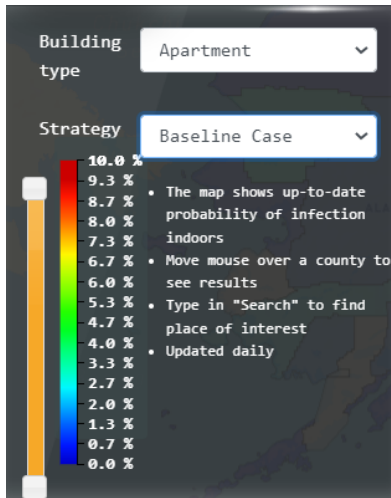


Figure 3-55 Application Module- CityRPI General User Input for North America

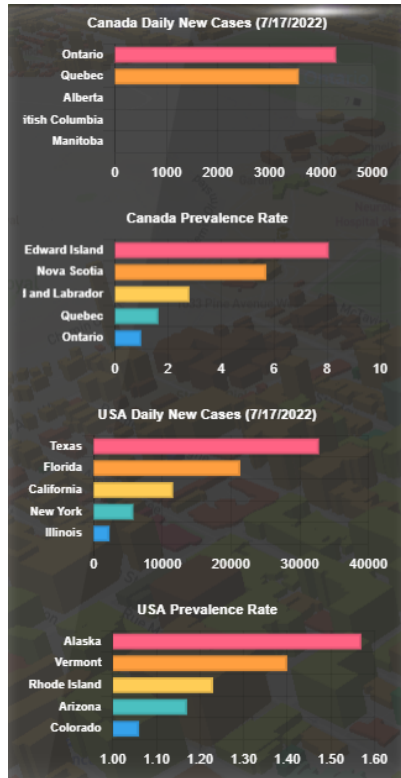


Figure 3-56 Application Module- CityRPI, Highest Daily Cases and Prevalence Rate.

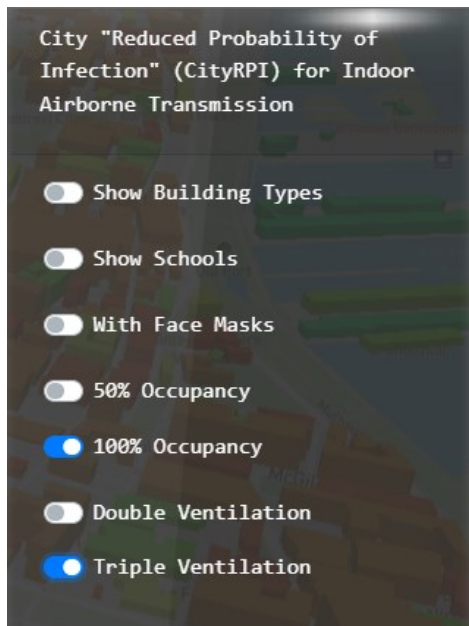


Figure 3-57 CityRPI Filtering Control UI

### 3.2.6 Application Module - Fatima-CFD Workflow and Design Architecture

The Fatima-CFD application module is for supporting the model of fate and transport of indoor microbiological aerosols, like the CityRPI, other laboratory members developed the computational



model for it. The application module supports data preparation and visualization; the CFD model includes inputs for particle information, general settings, CFD0 settings, boundary conditions, and slice settings. It solves room airflow, heat transfer, and trace contaminant concentrations with the drift-flux model for modeling particles.

Like the Core and other web applications, this application has been developed using a front-end Vue JavaScript framework and built on Paraview Glance. This general-purpose standalone web application can be used to visualize many data types. It is also a portal for building custom viewers on the web, which can involve remote services. It is an open-source JavaScript visualization application created by Kitware, based on Visualization Toolkit (VTK), and intended to serve as a lightweight companion to Paraview. It is a part of the Paraview Web suite of tools[49].

It provides user-friendly interfaces supporting input generation and simulation visualization with plenty of utilities. For example, a new interface to generate meshes similar to the CFD0 Editor is added. The CFD-0 Setting is a dynamic tool to generate a uniform and non-uniform domain and subdomain as CFD0 Editor does. Boundary Condition – selecting a 3D model geometry \*.STL file, the application automatically analyzes the contents as multi-solids. Each solid has a set of boundary conditions. By default, an STL model should be ready without holes/cracks for a CFD analysis.

The module can be used to provide CFD simulations of a contaminant source with known properties under different air condition systems. The module mainly consists of 9 different sections Figure 3-58 is presented as follows:

- Particle Information: To define the particle properties, including (Diameter, Density, Air- Mean Length, and Deposition Factor).
- General Settings: To define the general settings for the calculations like any CFD Model, including (Model scale, The number of steps required for the calculations, and the time step value)
- CFD00 Settings: This is used to define the mesh settings in every direction by defining the mesh planes and every plan setting. Multiple planes could be created to control the mesh.
- Boundary Conditions: This is where the input room geometry is inserted as a .stl file having the geometry of the inlets and outlets separated from the model as different solids, and the boundary and initial conditions are defined as shown in Figure 3-59.
- Slices Settings: This is the part where the user defines the desired output data, which slices extract by defining different planes to view results.

- Exporting the Inputs: after finishing the setup, the inputs are downloaded to the device for the simulation application to read.
- Simulation Application: an application should be run to simulate the previously downloaded input files.
- Result Generator: an application should be run to view the .vtk results file to show the results of the Fatima-CFD simulation.
- Visualization: The section where the model can be hidden, and the representation and color settings can be edited Figure 3-60.

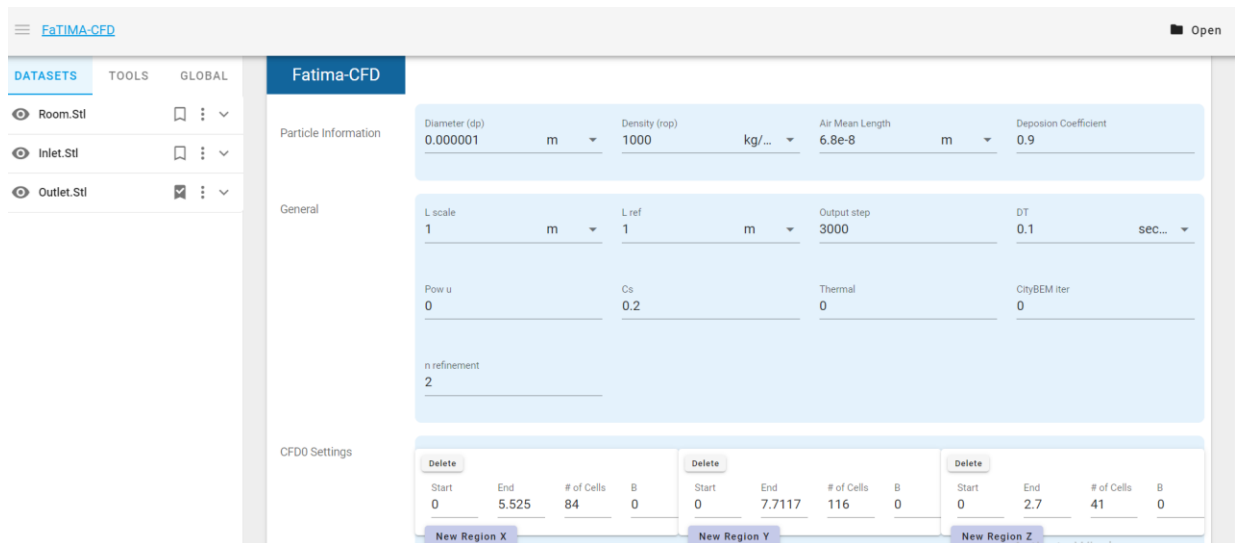


Figure 3-58 Application Module- Fatima-CFD Dashboard

STL file (ASCII only)  
geome...y.stl

<input checked="" type="checkbox"/>	Room	BC Type Wall constant T	Thickness Type Without thickness	BC value U 0	m/s	BC value V 0	m/s
		BC value W 0	m/s	BC value T 0	C	BC value C 0	
<input checked="" type="checkbox"/>	Qsinlet	BC Type Inlet	Thickness Type Without thickness	BC value U 0	m/s	BC value V 0	m/s
		BC value W -0.625	m/s	BC value T 0	C	BC value C 1	
<input checked="" type="checkbox"/>	QrOutlet	BC Type outlet	Thickness Type Without thickness	BC value U 0	m/s	BC value V 0	m/s
		BC value W 0.625	m/s	BC value T 0	C	BC value C 0	
<input checked="" type="checkbox"/>	QacInlet	BC Type Inlet	Thickness Type Without thickness	BC value U 0	m/s	BC value V 0.3933	m/s
		BC value W 0	m/s	BC value T 0	C	BC value C 1	
<input checked="" type="checkbox"/>	QacOutlet	BC Type outlet	Thickness Type Without thickness	BC value U 0	m/s	BC value V 0	m/s
		BC value W 0.3146	m/s	BC value T 0	C	BC value C 0	

Figure 3-59 Boundary Condition Input Section.

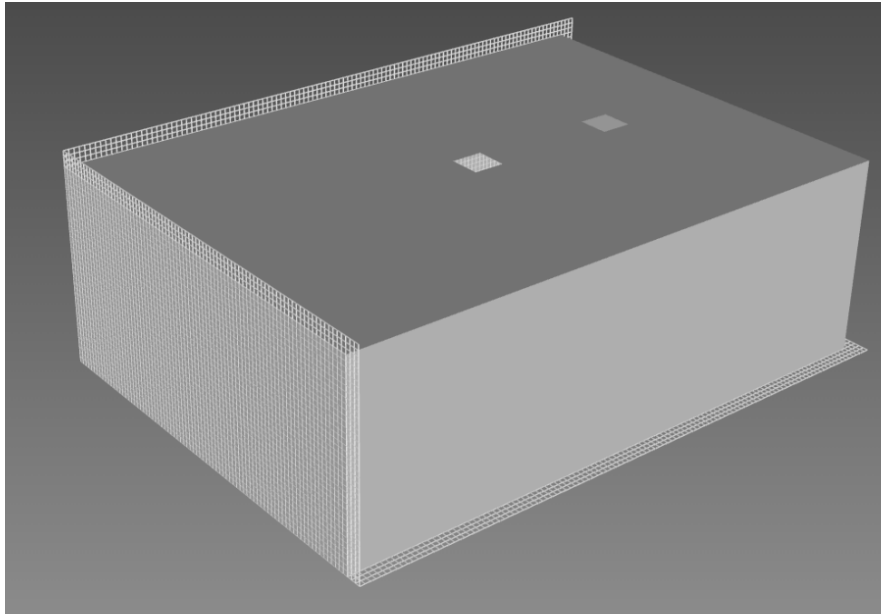


Figure 3-60 Fatima-CFD Mesh and Import Model Visualization

- Slices Settings Slices are orthogonal planes that the user can define to view the results. The user can add more than one slice to view the results. Slices are usually defined by the origin and the normal planes where the user inputs values for x, y, and z. For instance, if the user wants to create a certain plane at  $Z = 1.5$  and parallel to the XY plane, then the user will need to define the origin plane to have the values of  $(0,0,1.5)$  and the normal plane to be  $(0,0,1)$ . For more details.

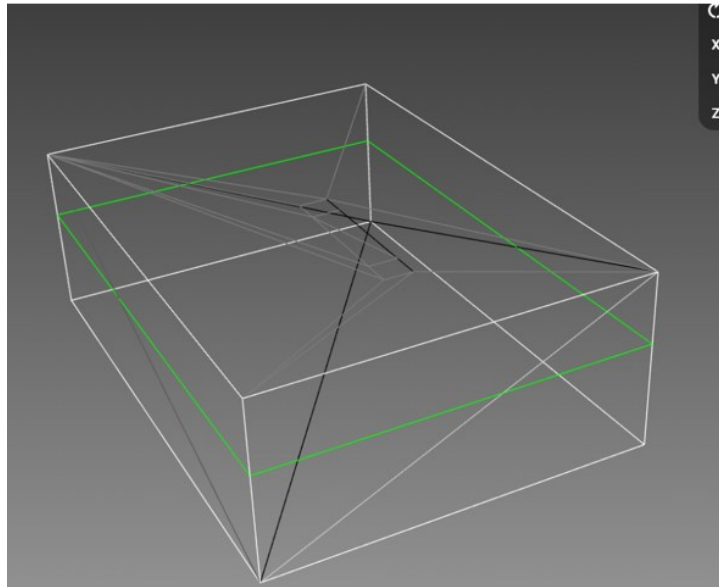
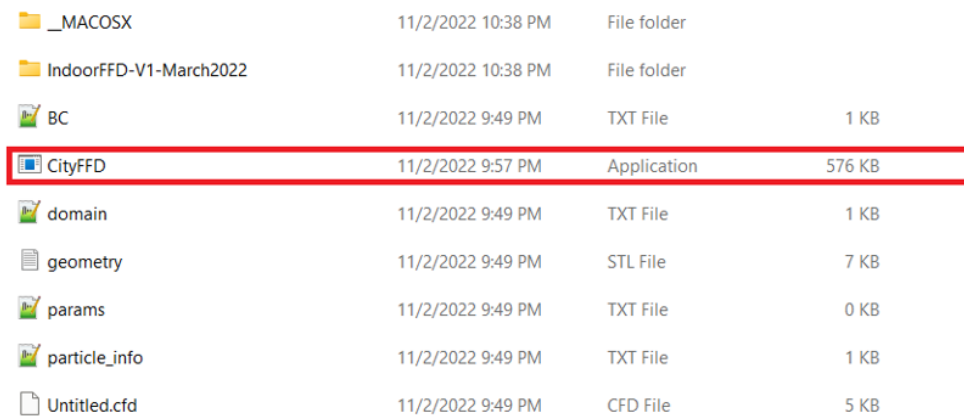


Figure 3-61 Plane Parallel to XY Plane at  $Z = 1.5$

- Exporting Inputs: after setting up the case and defining all the required information to start the calculation, inputs shall be downloaded to the user's computer as the simulation is conducted using CPU or GPU devices. Inputs are downloaded as a compressed .zip file. It is highly recommended to create a separate empty folder for the project's files. The user should then extract the compressed file which is extracted into 6 files (BC, Domain, Geometry, Params, "Particle\_info", Untitled. CFD). These 6 items contain the project information, as the name suggests. For instance, the BC file contains information about the velocities and temperatures defined in the Boundary Conditions section.
- Simulation Application: same as exporting the inputs, a compressed file should be defined, which then is extracted to have two folders (Indoor FFD & MACOSX). then simply need to copy the Fatima-CFD file to be with the extracted input files in the same place as it reads the files available in the same folder as shown in Figure 3-62. Then need to run the file, which opens a black window until the calculations are finished, and newer files are generated. The user

should stop the application after the time steps pass the pre-defined time steps and the result file is generated. Once the simulation is done, the .vtk file is generated and named after the number of output steps the user defined in the general settings. This file must be renamed to Result.vtk as the result generator application will search for a file with this name.



Folder	_MACOSX	11/2/2022 10:38 PM	File folder	
Folder	IndoorFFD-V1-March2022	11/2/2022 10:38 PM	File folder	
Text File	BC	11/2/2022 9:49 PM	TXT File	1 KB
Application	CityFFD	11/2/2022 9:57 PM	Application	576 KB
Text File	domain	11/2/2022 9:49 PM	TXT File	1 KB
STL File	geometry	11/2/2022 9:49 PM	STL File	7 KB
Text File	params	11/2/2022 9:49 PM	TXT File	0 KB
Text File	particle_info	11/2/2022 9:49 PM	TXT File	1 KB
CFD File	Untitled.cfd	11/2/2022 9:49 PM	CFD File	5 KB

Figure 3-62 FaTIMA-CFD Applications in the Same Place with the Input Files

- Result Generator Once the simulation is done, the result file is generated. The user needs to download the result generator compressed file. The file needs to be extracted in the same place where the result file is like the simulation application, as shown in Figure 3-63. The user should run the result generator application and wait for it to finish. Once finished, the “Final\_Results.vtkjs” file will be created having all the simulation results. This file can be inserted into the ParaView application or the online web app to visualize the results. The user can always refer to the ParaView tutorials (Paraview, n.d.) for more help with the ParaView settings.

6000.vtk	11/2/2022 10:54 PM	VTK File	40,168 KB
BC	11/2/2022 9:49 PM	TXT File	1 KB
CityFFD	11/2/2022 9:57 PM	Application	576 KB
domain	11/2/2022 9:49 PM	TXT File	1 KB
Final_Result.vtkjs	11/3/2022 1:02 AM	VTKJS File	8 KB
geometry	11/2/2022 9:49 PM	STL File	7 KB
params	11/2/2022 9:49 PM	TXT File	0 KB
particle_info	11/2/2022 9:49 PM	TXT File	1 KB
Result Generator	11/3/2022 12:45 AM	Application	301,638 KB
Result.vtk	11/2/2022 10:48 PM	VTK File	40,168 KB
Untitled.cfd	11/2/2022 9:49 PM	CFD File	5 KB

Figure 3-63 Result Generator & Final Results Files.

- Visualization This part discusses how to use the visualization settings inside the tool. Multiple settings are available and will be briefly discussed below: Hiding a Component, the user can hide the room or the mesh, or any of the components in the model by clicking on the eye can, which makes the component disappear in the drawing layout as shown in Figure 3-64.



Figure 3-64 Hiding a Component in the Geometry.

- Representation: in the representation part, the user can change the way the component is by selecting the display type from four types (Surface, Surface with edges, Wireframe, and Points). The user can also choose to control the opacity of the model and the point size if the user chooses the point type display, as shown in Figure 3-65.

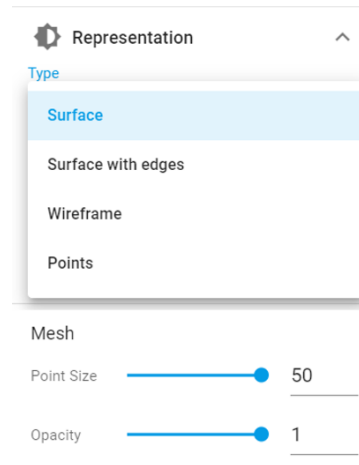


Figure 3-65 Model Representation Option.

The user has two choices to change the color of the model. The first one is to choose the solid color option and select a color from a set of pre-defined colors. The second option is to choose the normals option and select the filter for the colors.

The current application module contributes to collecting and manipulating the data input for the Fatima-CFD model in one phase and another, providing a web-based CFD result visualization tool.

### 3.3 Real-time Weather Station Data Integration

Real-time weather station integration monitors selected buildings' indoor and outdoor thermal and exterior climatic conditions to assess risks of summertime overheating. Many typical facilities housing vulnerable populations, including hospitals, schools, senior homes, and multi-unit residential social housing, are monitored during the summertime, especially during “heat waves.” the users can access the weather data and provide high-resolution microclimate information.

The system allows us to view the indoor and outdoor weather data (rain, solar radiation, wind speed, gust speed, wind direction, temperature, RH, dew point) in real-time tracking of the environment and observe the cycles and forecast variations; we have developed a Hobolink weather station connection driver, this driver provides a solution to easily integrate with weather stations that providing public access[50], the public access allows us to share the device page with others, the driver will grant this access to collect the points data, the driver not only for the station provided by HOBO RX3000 Station Figure 3-66 but also it can be used for any HTTP public access pages based.



Figure 3-66 HOBO RX3000 Station

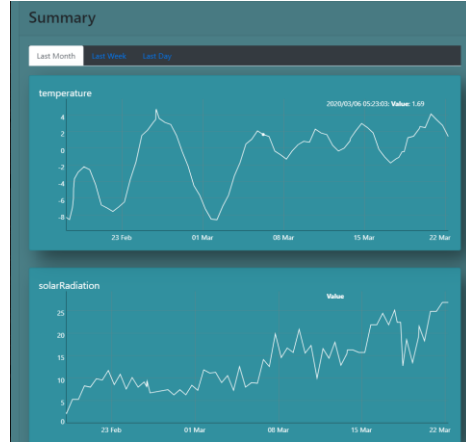


Figure 3-67 Example of Sensors Data History UI

The following figures show the driver hierarchy; the driver consists of a network and devices; the devices represent the weather station, and each device has several properties, such as the path property, which refers to the HTTP URL for the public page of the device. Moreover, based on the “Poll Scheduler property”, the time poll frequency in Figure 3-69 shows 10 seconds, which means that every 10 seconds, the system will query the latest update from the link.

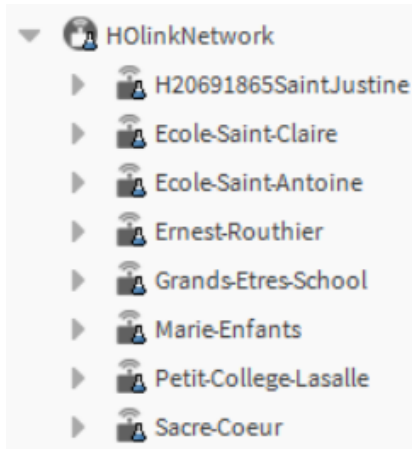


Figure 3-68 Weather Stations Driver Hierarchy

Ecole-Saint-Claire		HOLinkDevice
Status	[down]	
Enabled	<input checked="" type="checkbox"/> true	
Fault Cause		
Health	Fail [15-Jul-22 8:52 PM UTC] HTTP error fu...	
Alarm Source Info	Alarm Source Info	
pollFrequency	Normal	
points	HOLinkPointDeviceExt	
path	<a href="https://www.hobolink.com/p/b4a2b53b27def">https://www.hobolink.com/p/b4a2b53b27def</a>	
pollScheduler	+00000h 00m 10s	
readOnce	<input type="checkbox"/> false	
lastSuccessTime	19-May-2022 07:53:35 PM UTC	

Figure 3-69 New Driver Device Configurations

Each device consists of a point container, referring to the collected sensor points Figure 3-70. Figure 3-71 and Figure 3-73 show the user interface on the front-end monitoring dashboard.



points (HOLinkPointDeviceExt)		
Discovery Preferences	HOLinkPointDiscoveryPreferences	
SolarRadiation	213	{ok}
Rain	0.0	{ok}
WindDirection	WNW 289	{ok}
Temperature	5.57	{ok}
WindSpeed		{ok}
RH	52.40	{ok}
DewPoint	-3.40	{ok}
GustSpeed		{ok}

Figure 3-70 New Driver Weather Sensors Data Points

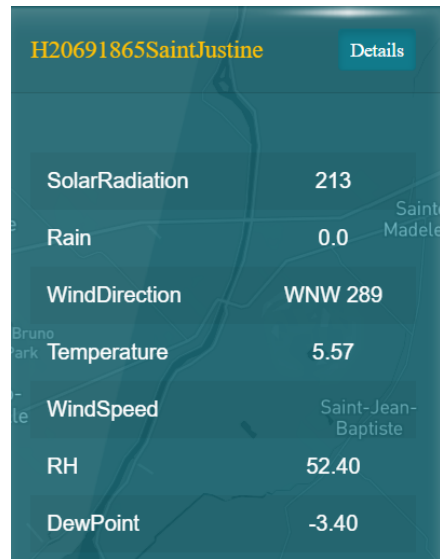


Figure 3-71 Real-time Sensors Monitoring

The system also provides history tracking functionality; monitoring sensor points Figure 3-72

OverheatingStation/Grands-Etres-School_Rain			
Timestamp	Trend Flags	Status	Value
16-Apr-21 1:04:42 AM UTC	{start}	{null}	
16-Apr-21 1:04:43 AM UTC	{start}	{null}	
16-Apr-21 1:08:54 AM UTC	{start}	{ok}	0.0
16-Apr-21 1:40:08 AM UTC	{start}	{ok}	0.0
16-Apr-21 1:43:28 AM UTC	{start}	{ok}	0.0
16-Apr-21 10:19:10 AM UTC	{}	{ok}	0.2
16-Apr-21 8:36:08 PM UTC	{start}	{ok}	0.2
17-Apr-21 10:19:23 AM UTC	{}	{ok}	0.0
17-Apr-21 5:10:28 PM UTC	{start}	{ok}	0.0
17-Apr-21 10:29:09 PM UTC	{start}	{ok}	0.0
18-Apr-21 7:19:23 PM UTC	{start}	{ok}	0.0
19-Apr-21 4:04:07 PM UTC	{start}	{ok}	0.0
30-Apr-21 2:48:49 PM UTC	{}	{ok}	0.2
01-May-21 2:48:59 PM UTC	{}	{ok}	0.0
25-May-21 5:38:21 PM UTC	{start}	{ok}	0.0
25-May-21 7:41:38 PM UTC	{start}	{ok}	0.0
26-May-21 4:40:33 PM UTC	{start}	{ok}	0.0

Figure 3-72 Sample of Track Sensor's History, Rain Sensor



Figure 3-73 Real-time Weather Stations Monitoring

### 3.4 Canadian Metrological Centre Weather Forecasting Framework Integration

The Canadian Metrological Centre (CMC) provides an online High-Resolution Deterministic Prediction System or HRDPS. The HRDPS carries out physics calculations to arrive at deterministic predictions of atmospheric elements from the current day to 48 hours into the future. The atmospheric elements include temperature, precipitation, cloud cover, wind speed and direction, humidity, and others. The data is available at a horizontal resolution of 2.5 km. Predictions are performed up to four times a day. Figure 3-74 below shows a points grid for the 2.5 km cell width distance.

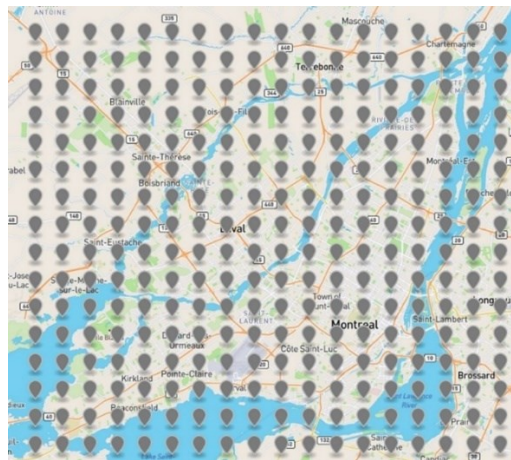


Figure 3-74 Example of 2.5 Km Points Grid

While weather stations provide discrete real-time weather data for measuring weather effects in the city, regional forecasted 48-hour weather data are retrieved and visualized on the system by integrating with the Canadian Meteorological Centre (CMC). These forecasted weathers are updated automatically and available in real-time on the front-end user interface to show more specific weather conditions of the city than those available from regular weather services. The forecasted CMC data are also used as the input boundary conditions for the urban microclimate analysis through CityFFD and CityBEM. We have created a CMC connection driver; this driver provides a solution to integrate with the CMC server and provide a data flow control input for CityFFD and CityBEM simulations.

The features of this driver are the following:

- It can be extracted from a region point using its bounding box, Figure 3-76. (lon1, lat1, lon2, lat2)
- It can be extracted from one location using its latitude and longitude Figure 3-77.

- Can be visualized the extracted point or region of points on an interactive map Figure 3-79
- It can be used in any backend logic model since it became a data point.

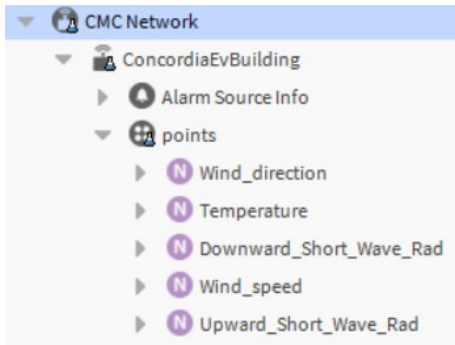


Figure 3-75 CMC Driver Hierarchy

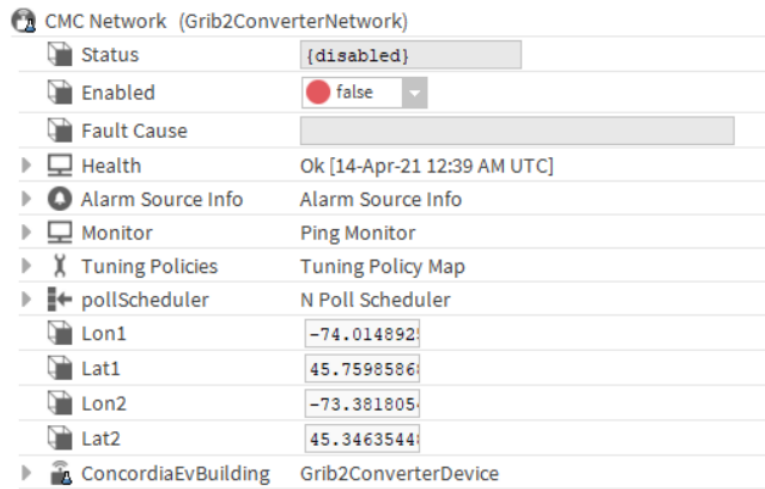


Figure 3-76 CMC New Driver Device Configurations

Figure 3-77 below shows the device configurations, which collect the sensor data points from specific CMC HTTP URLs and for a specific location (latitude, longitude).

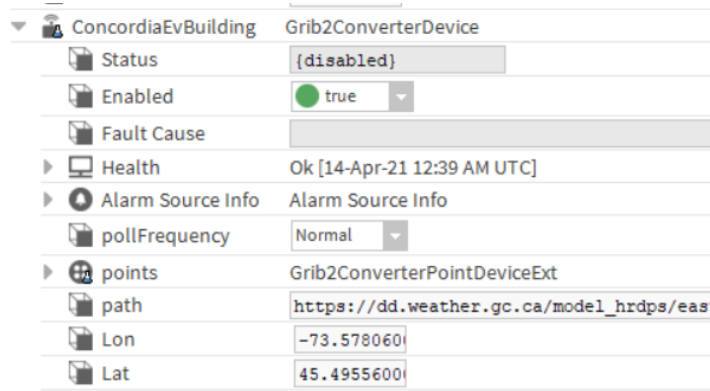


Figure 3-77 CMC Device Driver Configurations.

The path link, for instance, “[https://dd.weather.gc.ca/model\\_hrdps/east/grib2/12/](https://dd.weather.gc.ca/model_hrdps/east/grib2/12/)“. Contains about 39 girb2 files, each file holding the 2.5 km grid points for one data variable such as temperature and wind speed.

Name	Last modified	Size
<a href="#">Parent Directory</a>	-	-
<a href="#">CMC_hrdps_east_ABSV_ISBL_0250_ps2.5km_2022071812_P000-00.grib2</a>	2022-07-18 15:01	40K
<a href="#">CMC_hrdps_east_ABSV_ISBL_0500_ps2.5km_2022071812_P000-00.grib2</a>	2022-07-18 15:02	38K
<a href="#">CMC_hrdps_east_ABSV_ISBL_0700_ps2.5km_2022071812_P000-00.grib2</a>	2022-07-18 15:01	26K
<a href="#">CMC_hrdps_east_ABSV_ISBL_0850_ps2.5km_2022071812_P000-00.grib2</a>	2022-07-18 15:01	47K
<a href="#">CMC_hrdps_east_ABSV_ISBL_1000_ps2.5km_2022071812_P000-00.grib2</a>	2022-07-18 15:01	86K
<a href="#">CMC_hrdps_east_ALBD0_SFC_0_ps2.5km_2022071812_P000-00.grib2</a>	2022-07-18 15:01	105K
<a href="#">CMC_hrdps_east_CAPE_ETAL_10000_ps2.5km_2022071812_P000-00.grib2</a>	2022-07-18 15:01	419K

Figure 3-78 Sample of Model HRDPS URL Path, Girb2 Files

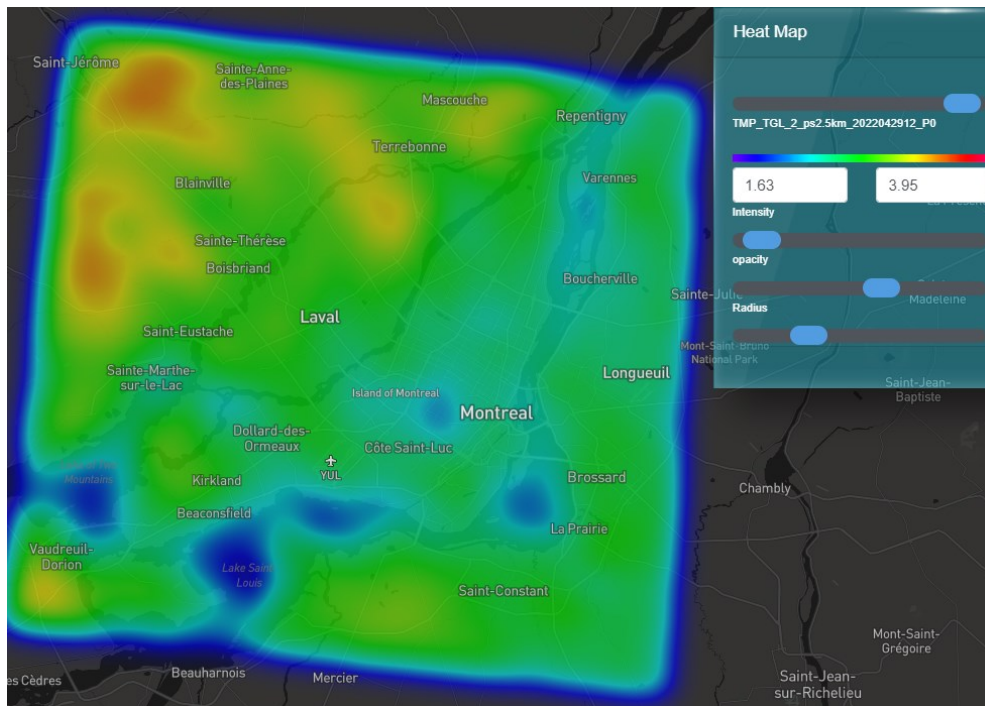


Figure 3-79 Forecasted Regional Map -Montreal

### 3.5 Automated Forecasting System Integration

A short-term forecasting service is an automatic large-scale short-term forecasting integrated with the Canadian Meteorological Centre. The simulation runs on a back-end server, and the website is updated every 48h providing many forecasting items for weather conditions, energy consumption, and thermal load. This integration consists of four components, a CMC server to collect the weather

forecasting, the integrated framework (Niagara framework based), a middleware, and the supercomputer server to run simulations.

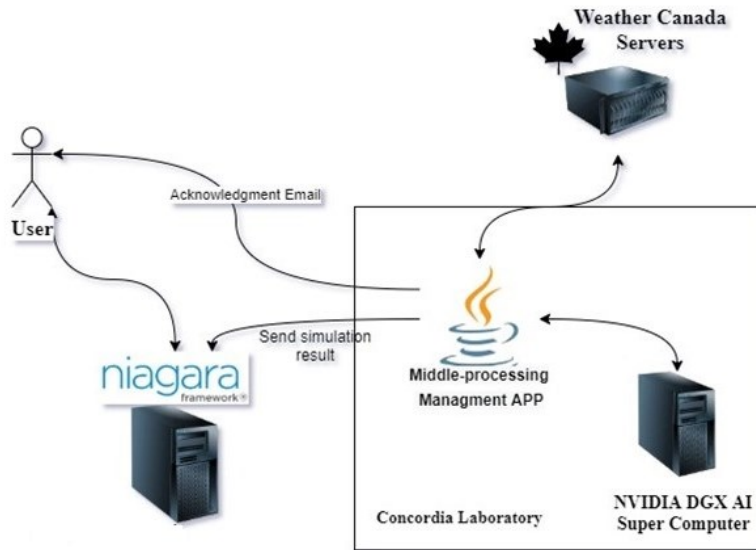


Figure 3-80 Automatic Forecasting Module Design

This section focuses on middleware implementation and its working mechanism; Middleware is the software layer between the operating system and the applications on each side of a distributed computer network. Typically, it supports complex, distributed software applications.

In our system, the middleware service starts working once it is run and with “run region” start status, periodically scheduling a new forecasting task case.

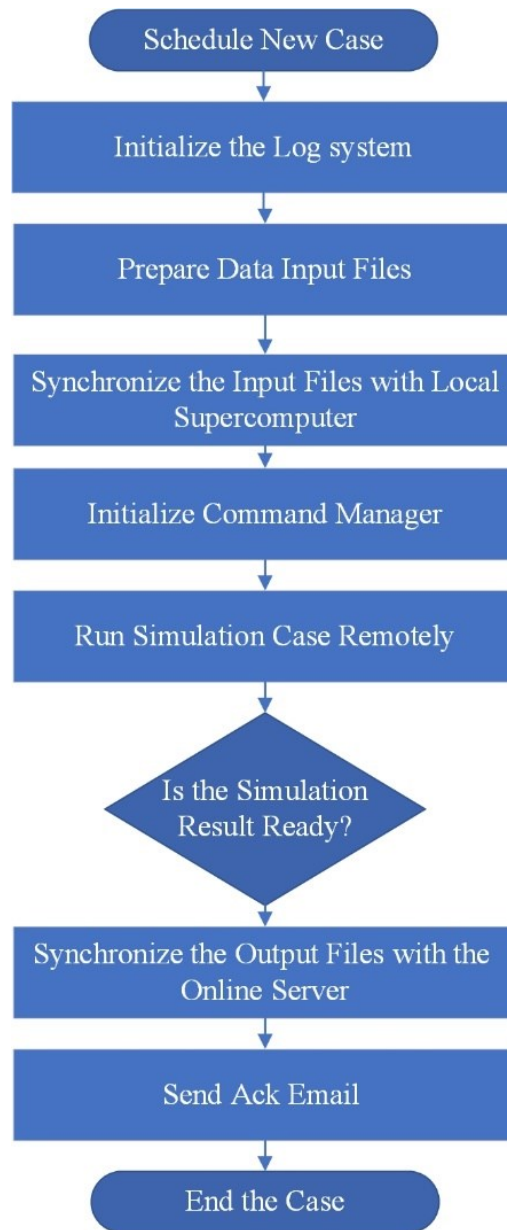


Figure 3-81 Schedule New Case Automatic Forecasting Workflow.

As shown in the chart Figure 3-81, scheduling a new case process starts by initializing a log system; the purpose of this log system is to manage, maintain, and troubleshoot the middleware system by tracking and logging all of the procedures in the local and remote systems after that will start the preparing data input process, which is mainly for collecting all the input files used in the forecasting simulations such as the weather data.

CMC Server Setting

CMC URL:

Latitude:

Longitude:

Start:

---

Latitude 1:       Latitude 2:

Longitude 1:       Longitude 2:

---

Figure 3-82 Middleware CMC Server Settings UI

By synchronizing collected input files and simulation application files with the local supercomputer using FileUtils, the FileUtils provides a method for manipulates like moving, opening, checking existence, and reading of files [51]hen **initializing the command manager**, which is based on JSch implementation of SSH. The SSH provides support for secure remote login, secure file transfer, and secure TCP/IP and X11 forwarding. It can automatically encrypt, authenticate, and compress transmitted data [52], so we can integrate its functionality into middleware programs.

SSH Settings

Host Name:

Port:

User Name:

Password:

Remote URL:

Schedule:

Script:

Figure 3-83 Middleware SSH Settings UI



After running **the simulations remotely**, the system will wait until the simulation is ready by waiting for the “done” signal from the remote server. It was then **synchronized back to the output** and deployed, making the result available online and sending notification emails to subscribers.

Upload Process	
Host Name	208.167.255.91
Port	19370
User Name	username
Password	*****
Device ID	1
Files List	output_CityBEM_filtered.txt positionSetting.txt

Save

Figure 3-84 Middleware Upload Settings UI

### 3.5.1 Preparing Weather Input Data Process

Figure 3-82 is shown the initial configuration for the CMC server, including the bounding box, which is the bounding of the targeted domain simulation and is defined by two points, the top left and bottom right point, as shown in Figure 3-85.

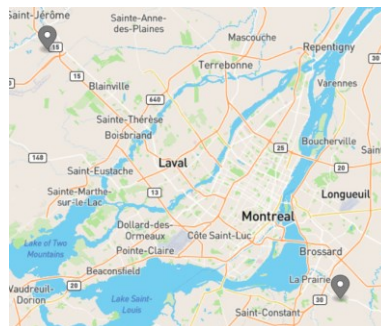


Figure 3-85 CMC Bounding Box Points

The weather data inputs are collected from several URL paths, mainly derived from one main path such as Figure 3-78. After loading the domain set and generating and downloading all sub-paths, we have a set of girb2 files representing the forecasting weather input data. Afterward, we used the NetCDF Java library to extract and use GIRB2 files. The NetCDF implements the Common Data Model (CDM) to interface netCDF files to a variety of data formats (e.g., netCDF, HDF, GRIB)



and provides a higher-level interface to geoscience-specific features of datasets, in particular, providing geolocation and data subsetting in coordinate space [53]. Based on the selected domain, extract the data points from the grib2 files and generate WeatherData.txt as shown in Figure 3-87.

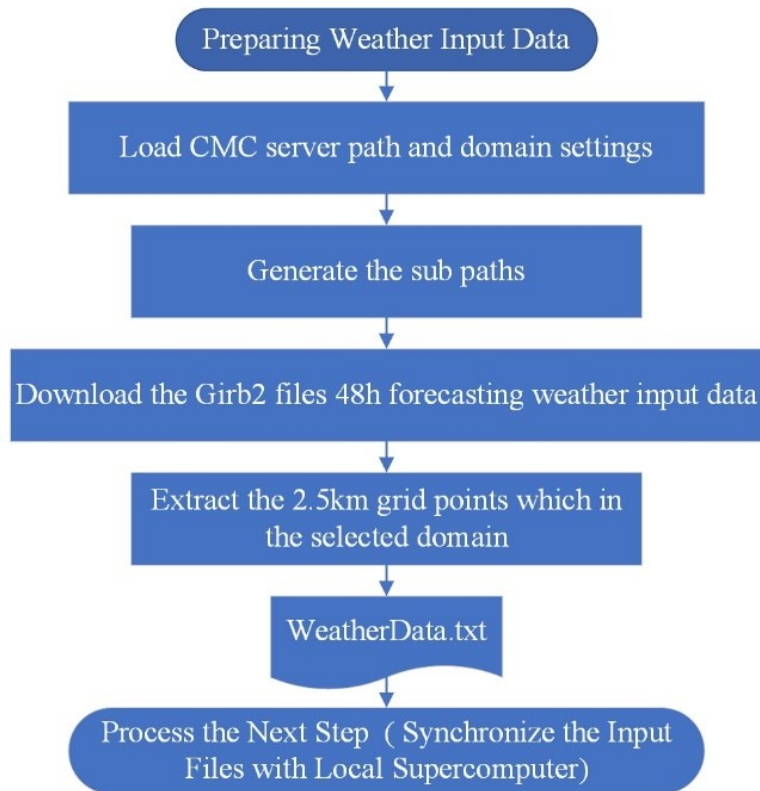


Figure 3-86 Preparing Weather Data Input Collection Workflow.

	A	B	C	D	E	F	G	H	I	J	K
1	Ni	29									
2	Nj	39									
3	i	j	lat	lng							
4	0	0	45.18134919	-74.17946692							
5	Date	Time	GHI	Ta	TD	TG	WS	WD	u	w	RH
6	2020-10-16	7:00:00	23.79133464	9.621636963	8.372399902	8.321801758	2.894356012	201.1645966	2.781928405	-0.798856106	91.52820587
7	2020-10-16	8:00:00	22.69100043	9.735131836	8.722619629	8.967431641	3.643899918	206.7909088	3.387681034	1.342245813	93.31869507
8	2020-10-16	9:00:00	57.72040799	10.03106079	8.739038086	9.1519104	4.348498821	204.3986969	-4.050845816	1.581167345	91.1215744
9	2020-10-16	10:00:00	104.387066	10.50930176	8.372460938	9.889764404	4.430779457	202.0074005	1.920680444	-3.992842751	86.93767548
10	2020-10-16	11:00:00	205.4424132	11.14223633	8.299310303	10.62764893	4.867046356	208.0812531	2.975179184	-3.85181114	82.48815155
11	2020-10-16	12:00:00	158.2820833	11.90603027	8.531335449	12.47228394	4.811409473	212.4176331	2.461728922	4.133951117	79.96716309
12	2020-10-16	13:00:00	115.2021528	11.8467041	8.209283447	11.18105469	5.226836205	234.3765106	-2.527986886	-4.574833222	77.82119751
13	2020-10-16	14:00:00	115.0127778	12.18947754	8.066522217	11.36550293	5.512438774	234.92453	-4.789402901	-2.729212541	75.60690308
14	2020-10-16	15:00:00	37.23590278	12.20583496	6.367456055	10.81209717	6.530733109	242.1786041	-5.872208501	2.857908722	67.36766815
15	2020-10-16	16:00:00	9.4975	11.02462158	6.197015381	9.1519104	5.501936913	225.1439056	3.531801075	4.218730965	72.40080261
16	2020-10-16	17:00:00	0.163263889	10.60567627	6.325219727	8.598504639	3.53407979	253.7355652	-3.000872317	-1.86667761	75.20458221
17	2020-10-16	18:00:00	-0.382222222	9.814019775	4.417443848	6.938287354	3.565186262	277.5313416	1.130419027	-3.381228461	68.20827484
18	2020-10-16	19:00:00	0.325069444	9.021386719	3.840478516	5.278070068	3.539191723	281.7752686	2.488271341	2.516820173	69.59172821
19	2020-10-16	20:00:00	0	8.055413818	3.421624756	3.34118042	2.977109432	278.361908	-1.447966027	-2.601264107	72.79801941
20	2020-10-16	21:00:00	0	7.411889648	2.364373779	1.680963135	2.758340359	281.0362549	0.11217998	2.75605827	70.71482849
21	2020-10-16	22:00:00	0	6.655511475	1.056115723	0.574151611	2.65017271	294.1334839	1.433550191	2.228979418	67.1312027
22	2020-10-16	23:00:00	0	5.740167236	0.385858154	-0.164770508	1.505167365	302.6189575	0.541154477	-1.404521495	67.94805908
23	2020-10-17	0:00:00	0	5.400018311	0.746209717	0.298822021	1.225194812	306.0315857	-0.11832815	1.219467415	71.82463074
24	2020-10-17	1:00:00	0	5.586663818	1.103845215	1.035760498	1.404388309	268.4010315	-0.039559446	1.403831034	72.08259583
25	2020-10-17	2:00:00	0	5.683068848	1.288201904	1.40090332	1.27165103	279.0992432	-1.204863696	-0.406693762	73.23556519
26	2020-10-17	3:00:00	0	5.687432861	1.964349365	1.67989502	1.605032682	292.1044006	-1.594750581	0.181384935	77.04406738
27	2020-10-17	4:00:00	0	5.614556885	1.849572754	1.67989502	1.241246104	327.2460938	0.951412504	-0.797186388	76.89745331

Figure 3-87 Weather Data Input Collected from CMC GRIB2

### 3.5.2 Synchronizing Input Data with Supercomputer Process.

In this process, as Figure 3-89 shows, the first step is loading the connection configuration, which is saved locally in the middleware. After that, using the connection configuration to initialize the SSH session, then copying the input files to the server after, the input files sample shown in Figure 3-88 is worth mentioning that the building\_info.txt file maps the buildings by using the “building\_osm” id, which is a unique id for each building, after that initialize the case individual log file on both remote servers and the local middleware device, as we mentioned previously that the library that middleware using is JSch implementation of SSH. This SSH provides support for secure remote login, secure file transfer, and secure TCP/IP and X11 forwarding; the X11 forwarding is a mechanism that allows a middleware to start up remote applications and then forward the application display to the local middleware machine. So in this way, we can also forward the display of the ongoing simulation case to the log file. The middleware every minute will check if the simulation is completed or not; this checking is achieved by listening to the log and waiting for a display from the remote server referring to the end of the simulation.

Building\_info.txt - Notepad

Building_stl	Building_osm	year	type	lon	lat
b0	165747	1990	9100	-73.56214335	45.50029417
b1	329792	1966	7114	-73.56625402	45.49817818
b2	88909	1962	6000	-73.56881868	45.50175049
b3	239443	1962	6000	-73.57135706	45.49874239

Figure 3-88 Automatic Forecasting System Integration, Building\_info.txt

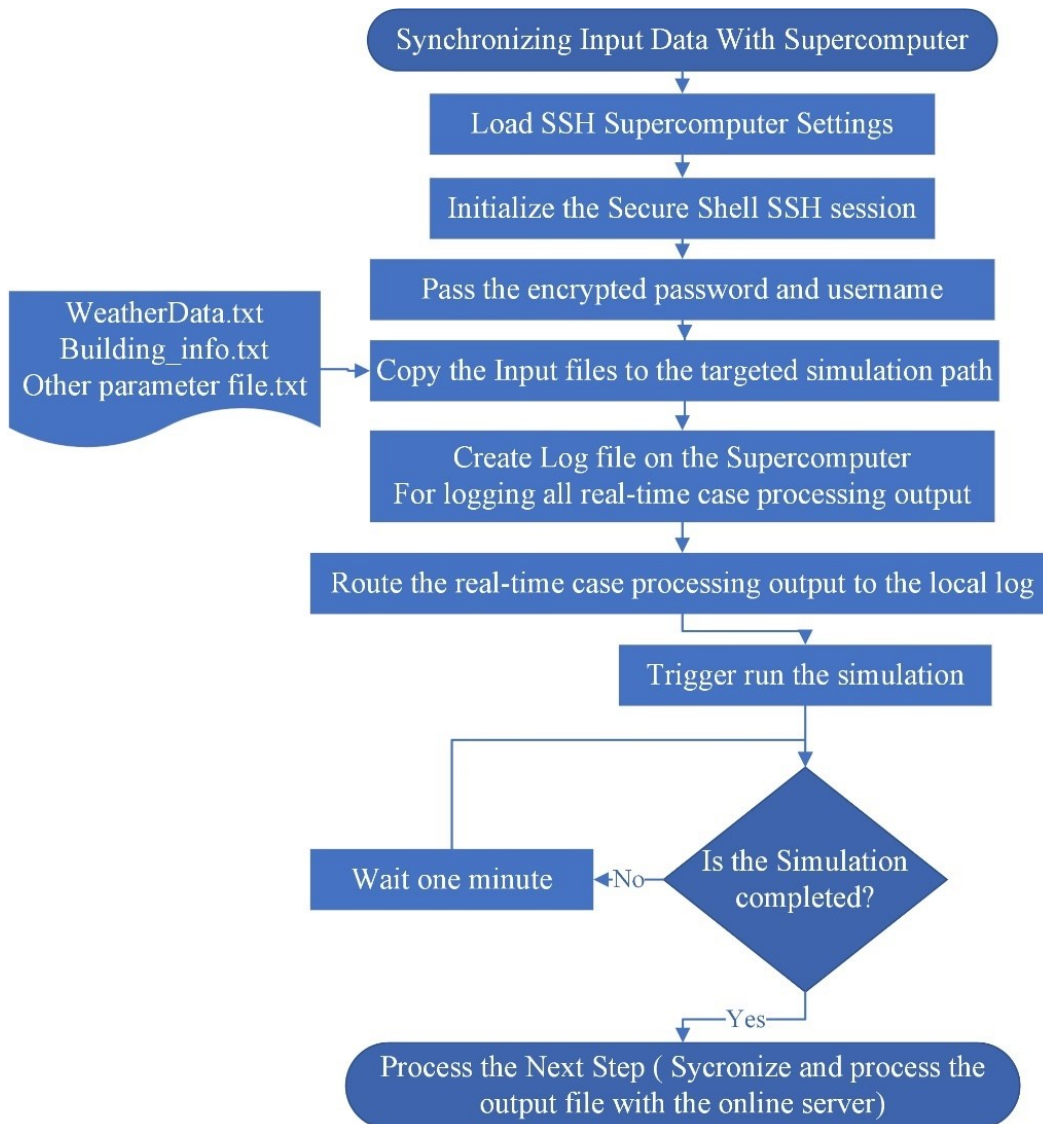


Figure 3-89 Workflow of Synchronizing Data Input Collection with Backend Server

### 3.5.3 Synchronizing the Output with the Online Server Process:

After the simulation is completed, synchronizing the output and deploying process start by copying the data result from the server to the local middleware device, then processing this result to generate the corresponding GeoJSON file. We have experimented with two solutions for the data result processing to get the most efficient result.

The first solution was to analyze the data result and map between each row and the corresponding geometric building feature by using the “Building\_osm\_id” as a matching factor, to generate the GeoJSON file by converting result data values into color information, then save it in the database as result history as shown in Figure 3-92. In this solution, we have used Mapbox as a map provider; the final step is converting the GeoJSON file to a Mapbox tileset and using Mapbox CLI to deploy the tileset to the map provider.

The second solution to process the result is generating the GeoJSON file without calculating the color information and processing the color information to the front-end implementation. The implementation of converting data result values to color information is discussed in Chapter 6. the second solution shows lower processing time and effective use of resources.

building	building_osm_id	T_a(C)	T_in(C)	Ts_ext(C)	Cooling_load(kW)
time	10/18/2020 13:00:00				
b0	165747	14.4116	20	16.0758	547.662
b1	329792	14.4116	20	16.1407	2043.9
b2	88909	14.4116	20	16.2301	-1452.64
b3	239443	14.4114	20	16.1685	-747.64
b4	378436	15.1301	18	15.8941	13.7513
b5	30489	15.1409	18	15.7914	19.507
b6	378442	15.0276	18	15.9069	15.4895
b7	378441	15.0187	18	15.8799	8.61334
b8	418964	14.6755	18	15.5818	4.801
b9	241184	14.4331	20	16.1169	2125.47
b10	378412	14.9643	18	15.6161	8.53806
b11	378446	14.9765	18	15.8801	10.2641
b12	161032	14.8806	18	15.529	10.3436
b13	378443	14.9568	18	15.6245	44.4271
b14	378443	14.9568	18	15.6145	43.1036
b15	43176	14.8822	18	15.5353	9.2631
b16	378447	14.9614	18	15.727	7.37553
b17	378410	14.8785	18	15.6484	8.24177
b18	144238	14.7067	20	15.3643	-13.7351

Figure 3-90 Automatic Forecasting System Integration Result.

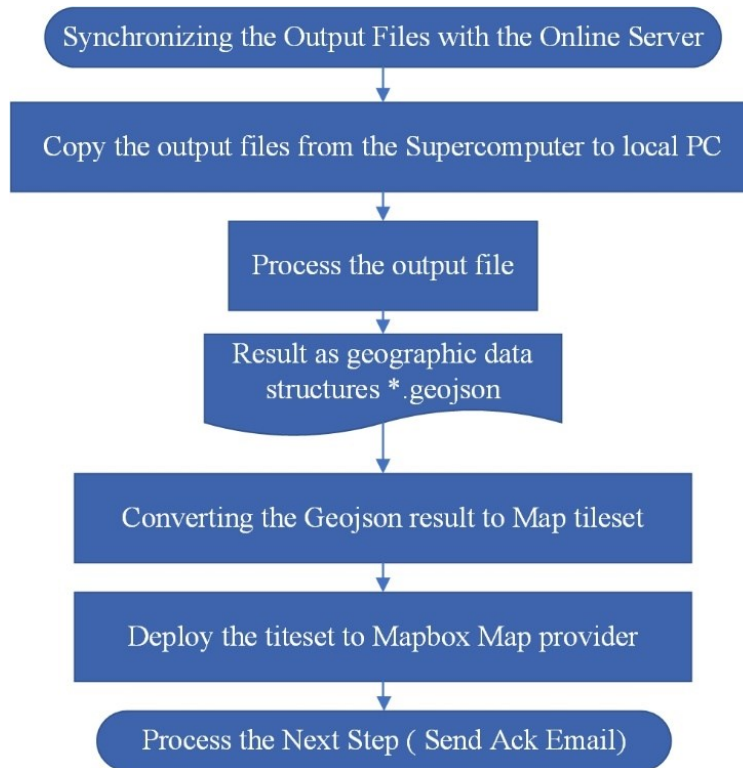


Figure 3-91 Synchronizing the Output Files with Online Server

id	colors	values	q1	q3	building_node_id
74834	#cb0000,#34ff00,#0000cb,#ceff00,	4559.91,25.8397,24.0,24.3587,	0.0529172	14.69225	165747
74835	#cb0000,#00ff40,#0000cb,#ebff00,	83.4733,25.448,24.0,24.3716,	0.0529172	14.69225	329792
74836	#cb0000,#00ff17,#0000cb,#fdff00,	6121.75,25.5855,24.0,24.3797,	0.0529172	14.69225	88909
74837	#cb0000,#00ff43,#0000cb,#ffec00,	3160.51,25.4357,24.0,24.3889,	0.0529172	14.69225	239443
74838	#cb0000,#00ff2f,#0000cb,#ff7600,	2769.36,25.5024,24.0,24.4414,	0.0529172	14.69225	241184
74839	#cb0000,#3aff00,#0000cb,#ebff00,	1801.59,25.8594,24.0,24.3718,	0.0529172	14.69225	226042
74840	#cb0000,#61ff00,#cb0000,#ffb300,	27.4755,25.9889,26.0,24.414,	0.0529172	14.69225	376002
74841	#cb0000,#45ff00,#0000cb,#ddff00,	3147.42,25.8952,24.0,24.3653,	0.0529172	14.69225	238580
74842	#cb0000,#45ff00,#0000cb,#ddff00,	3147.42,25.8952,24.0,24.3653,	0.0529172	14.69225	238580
74843	#cb0000,#ff5b00,#0000cb,#f7ff00,	2791.58,27.0667,24.0,24.377,	0.0529172	14.69225	297046
74844	#cb0000,#00ff0f,#0000cb,#ffeb00,	2072.72,25.6126,24.0,24.389,	0.0529172	14.69225	340266
74845	#b6ff00,#00ffa0,#cb0000,#ff7300,	9.75135,25.6269,26.0,24.443,	0.0529172	14.69225	95201
74846	#cb0000,#b7ff00,#0000cb,#ffdd00,	3083.99,26.2797,24.0,24.3953,	0.0529172	14.69225	175519
74847	#cb0000,#00ff20,#0000cb,#ffe500,	97.0773,25.555,24.0,24.3918,	0.0529172	14.69225	39768
74848	#cb0000,#0bff00,#0000cb,#b9ff00,	754.577,25.7027,24.0,24.3492,	0.0529172	14.69225	434031
74849	#cb0000,#00ffb3,#0000cb,#f5ff00,	3085.19,25.0593,24.0,24.376,	0.0529172	14.69225	113904
74850	#cb0000,#00ffb8,#cb0000,#ff7d00,	15.0433,25.0429,26.0,24.4383,	0.0529172	14.69225	387638
74851	#cb0000,#9bff00,#0000cb,#ffe300,	350.606,26.186,24.0,24.3928,	0.0529172	14.69225	308582
74852	#cb0000,#9eff00,#cb0000,#ff7000,	266.576,26.1936,26.0,24.4441,	0.0529172	14.69225	321923

Figure 3-92 Data Result with Colors Information Representation for Result Values

The real-time forecasting simulation result visualization can be uploaded directly to the dashboard. The visualization environment provides users with various tools to visualize real-time forecasting simulation results. The environment includes a real-time map. The map allows users to see how



events affect different regions of the world and provides users with information about the status of events. Figure 3-93 shows monitoring the result of the forecasting system on an interactive map; on the right side shows a “New Cases” tree; this tree represents the latest cases results; by clicking on any of the cases, the map navigates the view to the simulation target area and load the result variable and the time loop list.

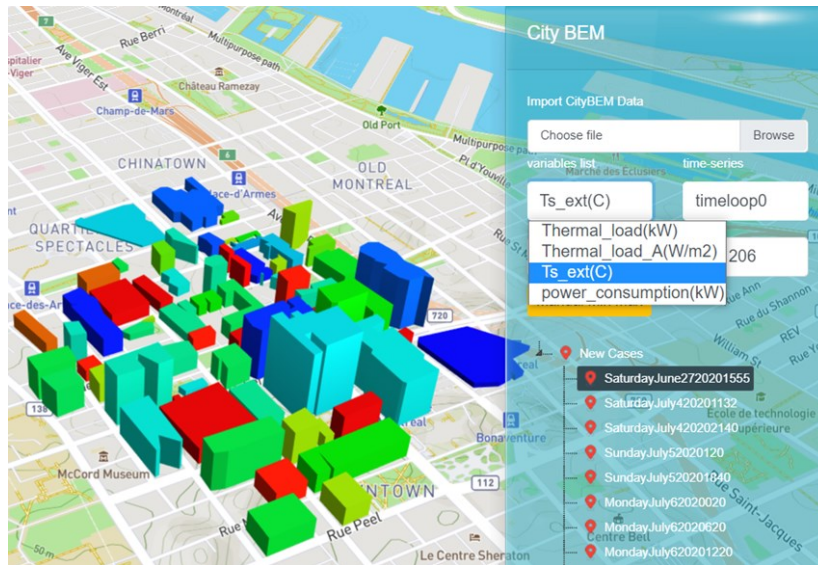


Figure 3-93 Online Monitoring the Forecasting of New Cases Simulation

### 3.5.4 Sending Notifying Emails Process

After processing and deploying the result and making the result available online, the automated forecasting system will send an acknowledgment of completion simulation for all the preconfigured emails, the Figure 3-94 below shows the Simple Mail Transfer Protocol configuration, with an email template and receiver emails list.

SMTP Settings

Email Host	smtp.gmail.com
Email Port	465
Email User Host	ciytfdd@gmail.com
Email Password Host	*****
From	ciytfdd@gmail.com
Subject	Short-term forecasting finished
Email Content	new case ready on the website maher.cn@hotmail.com New forecasting simulation is finished. You can check the results on the website.

Save

maher1989@hotmail.com,alikatall@gmail.com

Figure 3-94 Middleware SMTP Settings UI

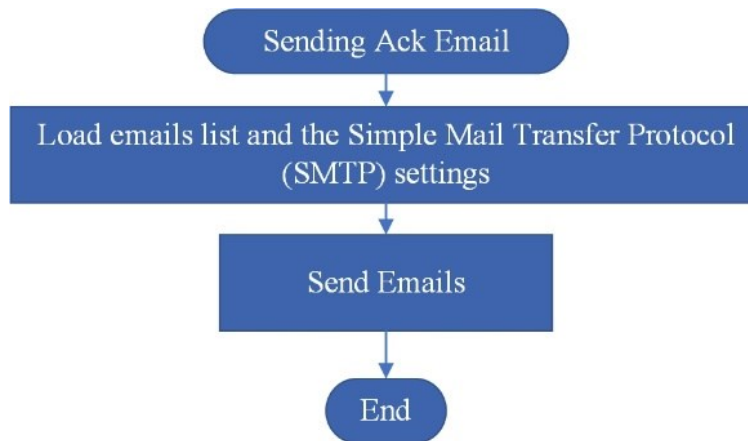


Figure 3-95 Sending Notification Email Workflow

### 3.6 Application Programming Interface (API) Integration Module

API integration is the connection between two or more applications. To perform a designed function built around sharing data and executing pre-defined processes via their APIs, that lets those systems exchange data. Through open standards, users can freely and quickly access thousands of real-time and archived weather, climate, and water datasets and integrate them into their domain-specific applications and decision support systems. The API integration module is providing an easy-to-use module in which the users can integrate and engage third-party API data sources and create interactive web maps and display and animate them. Many third-party APIs such as GeoMet-OGC-API provide public access to the Meteorological Service of Canada (MSC) and Environment and Climate Change Canada (ECCC) data and enable on-demand raw data

clipping and reprojection, on-demand format conversion, and custom visualization. The system provides an APIs Integration tool to import APIs data to the Map to enrich the data input and consider more resources, for example, to query the latest covid new cases and link it with a specific location in Figure 3-96 below the system querying from <https://kustom.radio-canada.ca/covid-19/canada>.

```
[
  {
    "Key": "canada",
    "Api": "https://kustom.radio-canada.ca/covid-19/canada",
    "Name": "Canada",
    "NameFr": "Canada",
    "Country": "Canada",
    "CountryFr": "Canada",
    "State": "",
    "StateFr": "",
    "Lat": "53.448",
    "Long": "-97.229",
    "Confirmed": "3554314",
    "Deaths": "37903",
    "Recovered": "3326019",
    "Population": "38048738",
    "History": [ 776 ],
    "Regions": [ 14 ],
    "DateUpdate": "2022-07-14 10:20:39"
  }
]
```

Figure 3-96 Example of API's Response Message, Covid New Cases in Canada

The system supports integration with multi-API sources; each source represents by one link

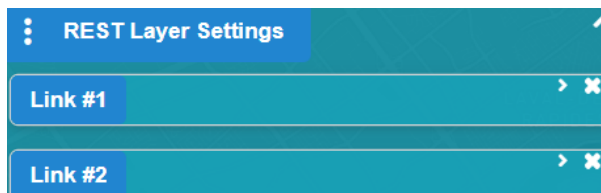


Figure 3-97 API Integration Module UI.

The following figures show the configuration of linking with the radio Canada API request and summarize the configuration steps:



- Set the URL Link Figure 3-98
- Set the Point latitude, longitude, and value attribute in the result tree. Figure 3-99

In this way, added one point from one link to the system, and this point can be used as part of data preparation applications.

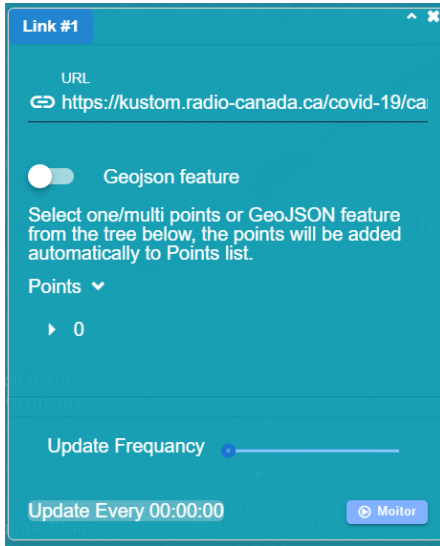


Figure 3-98 API Integration Module, Link Configuration 1

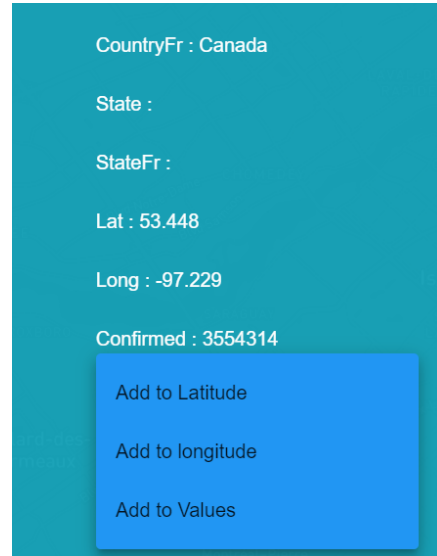


Figure 3-99 API Integration Module, Link Configuration 2

The tool supports collecting multi points from one link and collecting multi-links from multi-API sources.

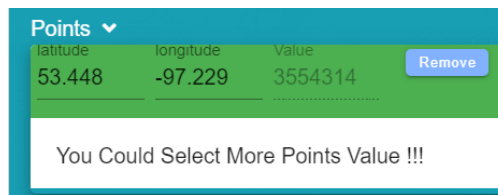


Figure 3-100 API Integration, Points Settings

The figure shows that after importing one data point from an external API to the map it becomes part of the system data source.

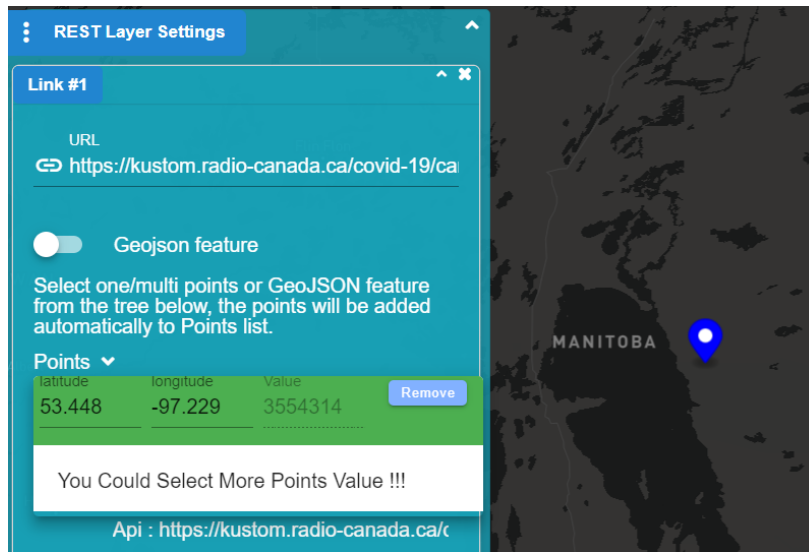


Figure 3-101 API Integration Module Result

In Figure 3-98, we added a frequency update time; this feature is for future improvement to support real-time APIs Integration.

# Chapter 4

## 4. Performance Evaluation

This chapter aims at evaluating the system platform and its operating status, and it focuses on evaluating the data input generation speed for a few urban-scale cases, visualization processing and rendering performance, and scalability of the system. The evaluation test performs on a PC with the following characteristics.

Table 3 PC and Connection Characteristics.

<b>Processor</b>	Intel Core i7 8500U
<b>RAM</b>	8 G
<b>GPU</b>	NVIDIA GeForce MX 150, 2Gbyte
<b>Internet Bandwidth</b>	Download Speed 30 Mbps, Upload 10 Mbps

### 4.1 Data Input Generation Speed Performance



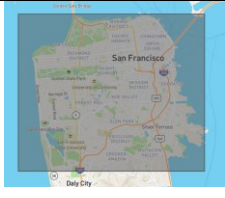
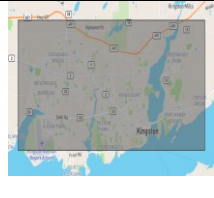
Using the modules explained in chapter 3, we can visualize two data sources: Imported GeoJSON-based data source and map provider tile set-based data source. We evaluated generating the data input and 3D model on four huge city-scale area cases. The first case is in San Francisco city, which used an external GeoJSON data source, the second two cases are for Montreal city for both external GeoJSON and internal Tile set data sources, and the third is for Kingston city.

The evaluation summary includes the following:

- **The total number of buildings** is the total number of features extracted from the geometric data source and represents the layout of the building.
- **The extraction process** is the map time response of querying the building features, and extracting the data from the map is the first step in the data input generation.

- **The data generation speed** is the time to process the dataset and generate the final set of files ( building information file, domain settings file, geo features data file, 3D model file, and position setting file) Figure 4-1.

Table 4 Speed Test Result for Three Cases of Data Input Generation.

Case	Montreal	Montreal	San Francisco	Kingston
<b>Total buildings</b>	314,237	314,237	208,663	32,503
<b>Extraction process</b>	1 sec	3.15 sec	1 sec	1 sec
<b>Generation speed</b>	36.13 sec	35.91 sec	36 sec	4 sec
<b>Data size</b>	1.31Gb	1.31Gb	875 Mb	127 Mb
<b>Area (m<sup>2</sup>)</b>	2,710,701,087	2,710,701,087	270,531,291	384,204,956
<b>Data source</b>	Imported as GeoJSON	Map provider tiles set	Imported as GeoJSON	Imported as GeoJSON
				

As shown in the evaluation result, importing the GeoJSON data set locally is much faster than using the map provider tile set in extracting, processing, and generating the data files.

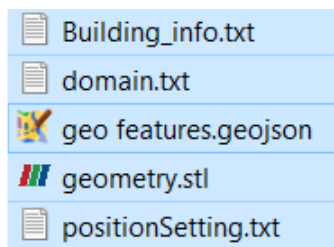







Figure 4-1 The Generated Data Files List

Table 5 Generated Data Files Description.

 <b>Building_info.txt</b>	We customized building information data input for the BES simulation.
--	---

 domain.txt	Domain settings. Data input for the CFD simulation.
 geo features.geojson	Geospatial data for the extracted buildings were used in the BES visualization phase.
 geometry.stl	3D model, Data input for CFD and BES simulation.
 positionSetting.txt	The bounding box settings for the extracted area and used in CFD in the visualization phase.

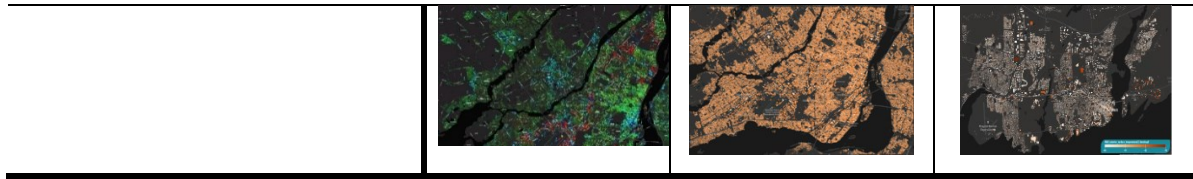
## 4.2 City-Scale Visualization Performance

Using our system visualization support module to test the visualization performance for two cases, one for Montreal city and another one for Kingston City. In the visualization evaluation process, we used two processing color solutions. The first one uses Mapbox paint properties with a custom linear interpolation color range and is applied to the external GeoJSON data. At the same time, the second solution is Mapbox paint properties with prior calculating colors for each of the buildings in the layer data and getting the actual calculated color value without any interpolation for the tile set. The paint properties define how data for that data layer is styled, and the Mapbox Graphics Library (GL) applies these properties later in the rendering process for both painting solutions.

The table below summarizes the performance testing result

Table 6 Visualization and Data Processing Test for Three Use Cases.

City	Montreal	Montreal	Kingston
<b>Rendering system</b>	Mapbox GL, pre-calculated colors.	Mapbox GL, linear interpolation colors range based	Mapbox GL, linear interpolation colors range based
<b>Total buildings</b>	314,237	314,237	32,503
<b>Parsing and colors processing</b>	~ 4 h	48 sec	5 sec
<b>Rendering process</b>	1 sec	6 sec	1.5 sec



The table shows a significant time cost for parsing and processing colors and deploying the result to the map provider to be used as a tile set. The processing and deployment cost around four hours. This solution is suitable for static result visualization. In the visualization performance evaluation, we found that using Map provider paint properties with the range-based color linear interpolation and GeoJSON dataset is much faster than using the map provider tile with pre-calculated color in parsing and processing the colors-values sets. While in the rendering process, using the pre-defined colors illustrates faster rendering.

### 4.3 Scalability Evaluation Summary

This section summarizes the two scalability viewpoints that our system supports. The first is for the system developer who will extend system functionalities, and the second is for the system user.

#### 4.3.1 System Functions Scalability, from Development Viewpoint.

The development scalability measures a system's ability to increase or decrease performance and cost in response to changes in application and system processing demands. Functional scalability is the ability to enhance the system by adding new functionality without disrupting existing activities. Our system is scalable on two levels, the back and front end.

- **Backend data source scalability:** Two levels of data sources, the first one is provided by the Map provider, and the other one is from the IoT Niagara framework service that integrates with an external SQL database if required in a future upgrade.

The example case studies below show the scalable data source in one of our system applications extended from continental countries, provinces and territories, cities, districts, and building scales. Alternatively, extending from a fewer details application to more details one. Case 1- Application for extending the USA to North America

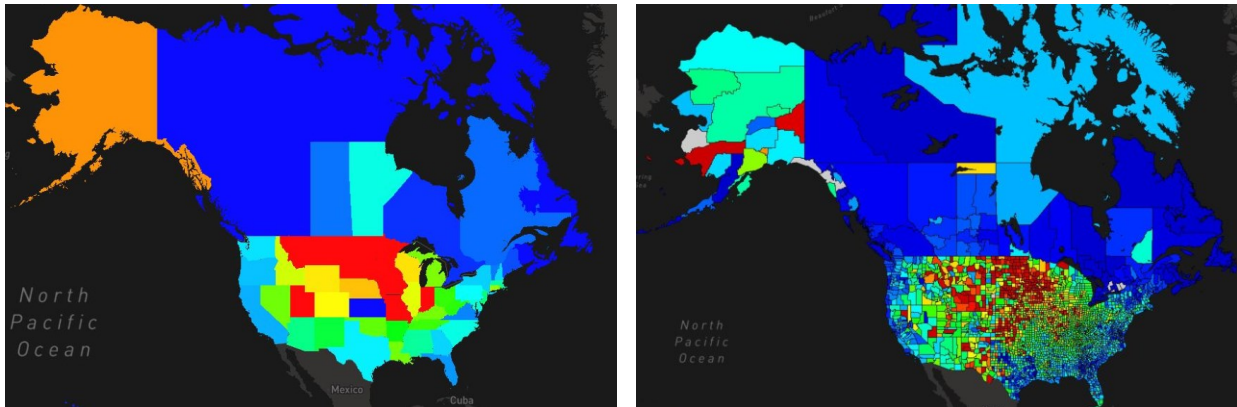


Figure 4-2 Extending North America Data Source Development.

- Backend services and devices integrations scalability:** IoT Niagara framework as a development platform includes hardware and software. Niagara allows developers to extend many hardware modules by developing drivers through the Java language. It can also be paired with other industrial hardware and has control capabilities with the hardware. The strong compatibility of the Niagara platform is to provide the ability to integrate with different devices, sensors, and subsystems. Niagara's design aim is to integrate with all networks and protocols, so Niagara implements many protocols and communication methods for integration with other systems like oBix, MQTT, LDAP, SNMP, NTP, Modbus, BacNet, Cloud, and LonWorks.
- Front-end data source layers scalability:** The front. is represented by the Vuejs framework component-based; moreover, it is flexible and scalable, and it can be used for vast, modular SPA (Single Page Apps), interactive parts to be integrated using a different technology. The system comprises one component and a sub-control panel with interfaces linked to the map system. A single-page application is an app that does not need to reload the page during its use and works within a browser such as Facebook, Google Maps, Gmail, Twitter, Google Drive, or even GitHub. All these are examples of a SPA. Vue is Web-components based, and the components are framework agnostic, which helps reuse and utilize components in projects based on different applications. This system has the power to standardize the development and consolidate a consistent and happy visual and functional experience for the users of applications. Components and widgets built on the Web Component standards will work across modern browsers and can be used with any JavaScript library or framework that works with HTML. This means more reuse, better stability, abstraction and standardization, less work, and everything else that comes with better modularity.

### 4.3.2 Geographic Usages and Applications Scalability the Use Viewpoint

Geographic scalability is the ability to maintain effectiveness during expansion from a local area to a larger region. and reflects the ability the grow with the user's demands. The system can be used in a local area (city, town, suburb) or a more extensive region, such as a state, province, or country (statewide, regional/county-wide, national). Moreover, it provides indoor and outdoor urban-scale support solutions, allowing users to extend the studying cases, the locations, and the applications. The system shows data integration feasibility, visualization efficiency, and diversity in processing several data types for more significant support in environmental applications, different cases, and locations, such as Qatar, Kingston, Montreal, and San Francisco. For example, possible user scenarios. The user can import/export in various ways for different applications. For example case study it was an application for district-scale wind and temperature field and district-level annual building energy simulation for Marina district, the southern district of Lusail City in Qatar, for encountering significant urbanization.

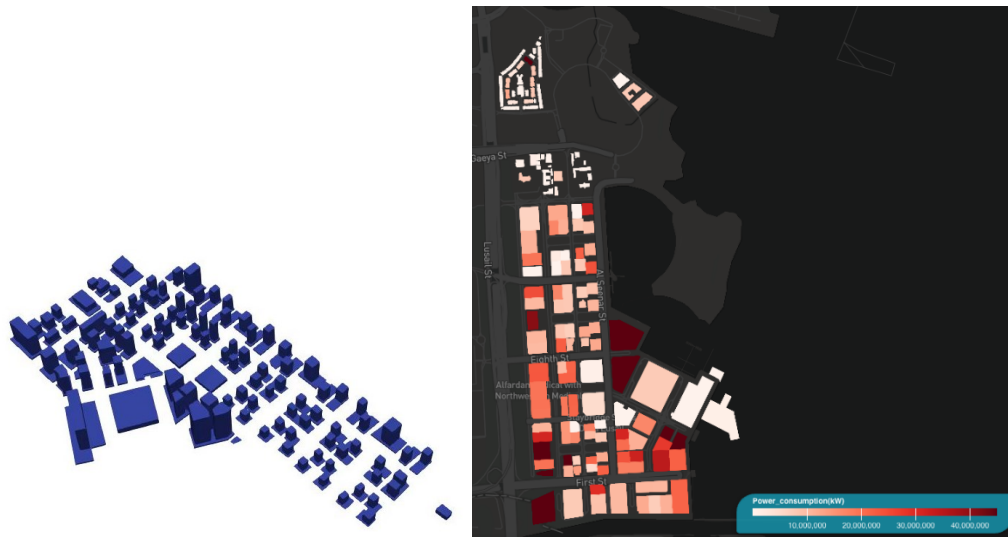


Figure 4-3 3D Model Generation and Visualization Functions of Annual Energy Performance Application, Marina District, Qatar.



# Chapter 5

## 5. Conclusion and Future Work

### 5.1 Conclusion

Here the problems that remained to be solved, problems encountered, and related requirements are summarized as follows:

- (1) The future of urban planning is inextricably linked to the future of the Internet of Things. At the same time, cities are complex systems with many moving parts - from geography to infrastructure; this makes data integration more challenging than ever before.
- (2) Urban modeling applications are complex fields requiring many data sources. The visualization and data integration requirements play an essential role in the success of these simulations.
- (3) Urban modeling applications require a high level of user knowledge to prepare data as input using different expert software.
- (4) At the city-scale level, Urban modeling applications are complex engineering calculations requiring much data to be processed. This data can be stored in different formats, which makes it difficult for the user to prepare it as input for the simulations.

The research questions addressed in this thesis are as follows:

- (1) How can GIS enable efficient IoT usages to support multi-scale urban modeling applications and simplify the data collection preparation (pre-process stage) and visualization (post-process stage)?
- (2) How do we build a reusable, extendable, integrated framework that supports urban applications and enables future functionalities?

In this thesis, by analyzing the problems encountered and related requirements, we leveraged the Niagara IoT framework and GIS integrations to build an integrated framework and developed many modules. These modules are for data preparation, creation, visualization, and integration, moreover

the real-time sensing systems, geographic information systems, and databases. These modules simplify the data integration process and make it easier to prepare. The visualization and data integration requirements can be simplified with the help of GIS and an easy-to-use integrated framework.

## 5.2 Contributions

We leveraged the scalable development frameworks IoT Niagara and VueJS to build this integrated framework; the system supports a high level of scalability, and to clarify the theoretical and practical importance, we categorize the contributions as follows:

**As Practical contributions,** we provided Integration and implementation solutions for the following:

- Implemented an automatic large-scale short-term forecasting BES service and integrated it with the Canadian Meteorological Centre (CMC) weather forecasting framework.
- Weather station integration, which provided discrete real-time weather data for monitoring.
- Provided an APIs Integration module to import APIs data to enrich the data input and consider more resources.
- Implemented an urban simulations data generation support module.
- Implemented additional support modules that enable extraction and generation of the input data for feeding the urban model applications.

**As theoretical contributions,** we provided new solutions for old issues that are solved differently:

- Solutions provided a clear scalable system architecture for measuring and controlling environmental systems.
- Solutions helped to improve areas like scenario testing and strategic planning for urban planners and policymakers.
- Solutions helped reduce the complexity of data input generation by exploring and solving data heterogeneity.
- Solutions allowed the researcher to view the indoor and outdoor weather data (rain, solar radiation, wind speed, gust speed, wind direction, temperature, RH, dew point) in real-time tracking of the environment.
- Solutions supported the environmental urban simulations applications, for example, supporting the response to known and unknowable risks, assessing the environmental consequences and

conditions of buildings, and enabling more effective management of urban-scale applications and simulation of environmental events.

## 5.1 Future Work and limitations

In this thesis, we primarily discussed supporting the applications of urban simulations at a multi-scale, from indoor to the outdoor environment, from building-scale to city-scale as a whole, and from energy to health applications, concentrating on the pre-processing and post-processing stages of urban modeling. Including but not limited to the framework.

1. This study does not delve into the different types of urban simulations, therefore more research and development are required to involve more stakeholders in supporting urban applications.
2. Automated forecasting system even if a system works reliably today, that does not mean it will work reliably in the future. One common reason for degradation is increased load: perhaps the system has grown from 10,000 concurrent users to 100,000 concurrent users or from 1 million to 10 million. Perhaps it is processing much larger volumes of data than it did before.
3. The system depends on the third-party map provider hosting the spatial data collection using the Mapbox processing tile CLI, the uploading and processing cost more money in larger cases.
4. The system can still be improved in related software functions for supporting urban simulation, such as adding more data analysis methods and introducing more mathematical calculations by utilizing features of reusable and component-oriented development of this integrated framework.
5. In the backend service and during processing the default system data collection, we got failures of processing tiles at the map provider service, which is the cause because the size of the tileset ("Job exceeded max allocated time"), so for larger cases we suggested decreasing the amount of the layer in each dataset file, that can make processing time shorter.
6. The integrated framework can be used for real-time feeding input and the base for building WebGPU computing modules; WebGPU exposes an API for performing operations, such as rendering and computation, on a Graphics Processing Unit. Specifically, the GPU Compute has contributed significantly to the recent machine learning boom, as convolution neural networks and other models can take advantage of the architecture to run more efficiently on GPUs, so It's highly suggested to engage the WebGPU API in future development.

## REFERENCES

- [1] “2413-2019 - IEEE Standard for an Architectural Framework for the Internet of Things (IoT),” 2020.
- [2] J. S. Bazargani, A. Sadeghi-Niaraki, and S. M. Choi, “A Survey of GIS and IoT Integration: Applications and Architecture,” *Applied Sciences* 2021, Vol. 11, Page 10365, vol. 11, no. 21, p. 10365, Nov. 2021, doi: 10.3390/APP112110365.
- [3] K. K. Patel, S. M. Patel, and P. G. Scholar, “Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges,” *International Journal of Engineering Science and Computing*, vol. 6, no. 5, 2016, doi: 10.4010/2016.1482.
- [4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, Oct. 2015, doi: 10.1109/COMST.2015.2444095.
- [5] P. Sethi and S. R. Sarangi, “Internet of Things: Architectures, Protocols, and Applications,” *Journal of Electrical and Computer Engineering*, vol. 2017, 2017, doi: 10.1155/2017/9324035.
- [6] M. U.Farooq, M. Waseem, S. Mazhar, A. Khairi, and T. Kamal, “A Review on Internet of Things (IoT),” *Int J Comput Appl*, vol. 113, no. 1, pp. 1–7, Mar. 2015, doi: 10.5120/19787-1571.
- [7] R. Zaheer and S. Khan, “Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges,” in *10th International Conference on Frontiers of Information Technology (FIT): Proceedings*, 2012, pp. 257–260. doi: 10.1109/FIT.2012.53.
- [8] D. Bandyopadhyay and J. Sen, “Internet of Things: Applications and Challenges in Technology and Standardization,” *Wireless Personal Communications* 2011 58:1, vol. 58, no. 1, pp. 49–69, Apr. 2011, doi: 10.1007/S11277-011-0288-5.
- [9] N. R. Chrisman, “What Does ‘GIS’ Mean?,” *Transactions in GIS*, vol. 3, no. 2, pp. 175–186, Mar. 1999, doi: 10.1111/1467-9671.00014.
- [10] L. Miloudi and K. Rezeg, “LEVERAGING the POWER of INTEGRATED SOLUTIONS of IoT and GIS,” *Proceedings - PAIS 2018: International Conference on Pattern Analysis and Intelligent Systems*, Dec. 2018, doi: 10.1109/PAIS.2018.8598500.
- [11] S. Banerjee, C. Chakraborty, and D. Das, “An Approach towards GIS Application in Smart City Urban Planning,” *Internet of Things and Secure Smart Environments*, pp. 71–110, Nov. 2020, doi: 10.1201/9780367276706-2.
- [12] “Importance of GIS in planning,” Mar. 03, 2018. <https://grindgis.com/gis/importance-of-gis-in-planning> (accessed Jul. 03, 2022).
- [13] S. Khafa and A. Kosovrasti, “Geographic Information Systems (GIS) in Urban Planning,” *European Journal of Interdisciplinary Studies*, vol. 1, no. 1, pp. 85–92, Apr. 2015, doi: 10.26417/EJIS.V1I1.P85-92.
- [14] J. Liu, J. Niu, Y. Du, C. M. Mak, and Y. Zhang, “LES for pedestrian level wind around an idealized building array—Assessment of sensitivity to influencing parameters,” *Sustain Cities Soc*, vol. 44, pp. 406–415, Jan. 2019, doi: 10.1016/J.SCS.2018.10.034.

- [15] J. Liu and J. Niu, “CFD simulation of the wind environment around an isolated high-rise building: An evaluation of SRANS, LES and DES models,” *Build Environ*, vol. 96, pp. 91–106, Feb. 2016, doi: 10.1016/J.BUILDENV.2015.11.007.
- [16] S. Gilham, D. M. Deaves, R. P. Hoxey, C. R. Boon, and A. Mercer, “Gas build-up within a single building volume — comparison of measurements with both CFD and simple zone modelling,” *J Hazard Mater*, vol. 53, no. 1–3, pp. 93–114, May 1997, doi: 10.1016/S0304-3894(96)01836-5.
- [17] D. J. Kim, D. il Lee, J. J. Kim, M. S. Park, and S. H. Lee, “Development of a Building-Scale Meteorological Prediction System Including a Realistic Surface Heating,” *Atmosphere 2020, Vol. 11, Page 67*, vol. 11, no. 1, p. 67, Jan. 2020, doi: 10.3390/ATMOS11010067.
- [18] K. Javanroodi, M. Mahdavinejad, and V. M. Nik, “Impacts of urban morphology on reducing cooling load and increasing ventilation potential in hot-arid climate,” *Appl Energy*, vol. 231, pp. 714–746, Dec. 2018, doi: 10.1016/J.APENERGY.2018.09.116.
- [19] I. Lun, A. Mochida, and R. Ooka, “Progress in Numerical Modelling for Urban Thermal Environment Studies,” <http://dx.doi.org/10.3763/aber.2009.0306>, vol. 3, no. 1, pp. 147–188, 2011, doi: 10.3763/ABER.2009.0306.
- [20] R. Yoshie *et al.*, “Cooperative project for CFD prediction of pedestrian wind environment in the Architectural Institute of Japan,” *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 95, no. 9–11, pp. 1551–1578, Oct. 2007, doi: 10.1016/J.JWEIA.2007.02.023.
- [21] H. H. Hu, “Computational Fluid Dynamics,” *Fluid Mechanics*, pp. 421–472, 2012, doi: 10.1016/B978-0-12-382100-3.10010-1.
- [22] M. Mortezaadeh, L. L. Wang, M. Albettar, and S. Yang, “CityFFD – City fast fluid dynamics for urban microclimate simulations on graphics processing units,” *Urban Clim*, vol. 41, p. 101063, Jan. 2022, doi: 10.1016/J.UCLIM.2021.101063.
- [23] Tridium Inc, *Niagara 4 Platform Guide*, 1st ed. 2018.
- [24] “Tridium Inc - Company Profile and News - Bloomberg Markets.” <https://www.bloomberg.com/profile/company/63395Z:US> (accessed Jul. 03, 2022).
- [25] M. Albettar, “Evaluation and assessment of cyber security based on Niagara framework: a review,” <https://doi.org/10.1080/23742917.2019.1627699>, vol. 3, no. 3, pp. 125–136, Jul. 2019, doi: 10.1080/23742917.2019.1627699.
- [26] K. Kyoreva, “State of the Art JavaScript Application Development with Vue.js,” in *7 TH INTERNATIONAL CONFERENCE ON APPLICATION OF INFORMATION AND COMMUNICATION TECHNOLOGY AND STATISTICS IN ECONOMY*, Nov. 2017.
- [27] “vue - npm.” <https://www.npmjs.com/package/vue> (accessed Jul. 01, 2022).
- [28] “Components Basics | Vue.js.” <https://vuejs.org/guide/essentials/component-basics.html> (accessed Jul. 01, 2022).
- [29] J. A. Milbrandt, S. Bélair, M. Faucher, M. Vallée, M. L. Carrera, and A. Glazer, “The Pan-Canadian High Resolution (2.5 km) Deterministic Prediction System,” *Weather Forecast*, vol. 31, no. 6, pp. 1791–1816, Dec. 2016, doi: 10.1175/WAF-D-16-0035.1.
- [30] “HRDPS data in GRIB2 format.” [https://weather.gc.ca/grib/grib2\\_HRDPS\\_HR\\_e.html](https://weather.gc.ca/grib/grib2_HRDPS_HR_e.html) (accessed Jul. 11, 2022).
- [31] MSC Open Data / Données ouvertes du SMC, “High Resolution Deterministic Prediction System (HRDPS) data in GRIB2 format,” *MSC Open Data / Données ouvertes du SMC*. [https://eccc-msc.github.io/open-data/msc-data/nwp\\_hrdps/readme\\_hrdps-datamart\\_en/](https://eccc-msc.github.io/open-data/msc-data/nwp_hrdps/readme_hrdps-datamart_en/) (accessed Jul. 11, 2022).

- [32] “GeoJSON.” <https://geojson.org/> (accessed Jul. 24, 2022).
- [33] J. Jokar Arsanjani, A. Zipf, P. Mooney, and M. Helbich, “An Introduction to OpenStreetMap in Geographic Information Science: Experiences, Research, and Applications,” *Lecture Notes in Geoinformation and Cartography*, vol. 0, no. 9783319142791, pp. 1–15, 2015, doi: 10.1007/978-3-319-14280-7\_1/FIGURES/6.
- [34] “Property Assessment Units - Dataset, Montreal Data.” <https://donnees.montreal.ca/ville-de-montreal/unites-evaluation-fonciere> (accessed Jul. 14, 2022).
- [35] A. Katal, “Development of Multi-Scale City Building Energy Model for Urban Climate Resilience,” Concordia university, Montreal, 2020. Accessed: Jul. 05, 2022. [Online]. Available: <https://spectrum.library.concordia.ca/id/eprint/987739/>
- [36] John R. Herring, “OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture ,” 2011, pp. 26–28. Accessed: Jul. 29, 2022. [Online]. Available: <https://www.ogc.org/standards/sfa>
- [37] D. Eberly, *Triangulation by Ear Clipping*. Mountain View, CA, 2002. Accessed: Jul. 14, 2022. [Online]. Available: <https://www.geometrictools.com/>
- [38] “Turf.js | Advanced geospatial analysis.” <https://turfjs.org/> (accessed Jul. 29, 2022).
- [39] “Find elevations with the Tilequery API | Help | Mapbox.” <https://docs.mapbox.com/help/tutorials/find-elevations-with-tilequery-api/> (accessed Jul. 29, 2022).
- [40] A. Mastrucci, O. Baume, F. Stazi, S. Salvucci, and U. Leopold, “A GIS-BASED APPROACH TO ESTIMATE ENERGY SAVINGS AND INDOOR THERMAL COMFORT FOR URBAN HOUSING STOCK RETROFITTING,” in *Fifth German-Austrian IBPSA Conference* , 2014.
- [41] A. Alhamwi, W. Medjroubi, T. Vogt, and C. Agert, “GIS-based urban energy systems models and tools: Introducing a model for the optimisation of flexibilisation technologies in urban areas,” *Appl Energy*, vol. 191, pp. 1–9, Apr. 2017, doi: 10.1016/J.APENERGY.2017.01.048.
- [42] R. Kaden, T. H. Kolbe, R. Kaden, and T. H. Kolbe, “City-Wide Total Energy Demand Estimation of Buildings Using Semantic 3d City Models and Statistical Data,” *ISPA*, vol. II2, no. 2W1, pp. 163–171, Sep. 2013, doi: 10.5194/ISPRSANNALS-II-2-W1-163-2013.
- [43] C. Cerezo, T. Dogan, and C. Reinhart, “TOWARDS STANDARIZED BUILDING PROPERTIES TEMPLATE FILES FOR EARLY DESIGN ENERGY MODEL GENERATION,” in *Building Simulation Conference* , 2014.
- [44] C. M. Rose and V. Bazjanac, “An algorithm to generate space boundaries for building energy simulation,” *Eng Comput*, vol. 31, no. 2, pp. 271–280, Apr. 2015, doi: 10.1007/S00366-013-0347-5/TABLES/1.
- [45] T. Dogan and C. Reinhart, “AUTOMATED CONVERSION OF ARCHITECTURAL MASSING MODELS INTO THERMAL ‘SHOEBOX’ MODELS,” in *13th Conference of International Building Performance Simulation Association*.
- [46] “Streamlining Energy Analysis of Existing Buildings with Rapid Energy Modeling,” *Building Performance Analysis*, 2013.
- [47] A. I. Nutkiewicz, “INTEGRATING PHYSICAL AND DATA-DRIVEN PERSPECTIVES ON BUILDING ENERGY PERFORMANCE: A TALE OF TWO CITIES A DISSERTATION SUBMITTED TO THE DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING AND THE COMMITTEE ON GRADUATE STUDIES OF STANFORD UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY,” Stanford

- University, 2021. Accessed: Jul. 29, 2022. [Online]. Available: <https://purl.stanford.edu/nm585pw3295>
- [48] M. Albettar, L. (Leon) Wang, and A. Katal, “A real-time web tool for monitoring and mitigating indoor airborne COVID-19 transmission risks at city scale,” *Sustain Cities Soc*, vol. 80, May 2022, doi: 10.1016/J.SCS.2022.103810.
- [49] A. Razoumov, “Web-based 3D scientific visualization,” 2020. Accessed: Jul. 31, 2022. [Online]. Available: <https://bit.ly/vispages>
- [50] O. Computer Corporation, *HOBOLink® User’s Guide*. 2014. Accessed: Jul. 18, 2022. [Online]. Available: [www.onsetcomp.com](http://www.onsetcomp.com)
- [51] “FileUtils (Apache Commons IO 2.5 API).” <https://commons.apache.org/proper/commons-io/javadocs/api-2.5/org/apache/commons/io/FileUtils.html> (accessed Jul. 17, 2022).
- [52] “JSch - Java Secure Channel.” <http://www.jcraft.com/jsch/> (accessed Jul. 17, 2022).
- [53] “Unidata | NetCDF Java.” <https://www.unidata.ucar.edu/software/netcdf-java/> (accessed Jul. 17, 2022).