**CHAPTER 4**

# WEB-BASED INTERACTIVE MAP OF MSU-IIT

This chapter presents the overall specifications, functional requirements and nonfunctional requirements of the software being developed.

## **Systems Requirements Analysis and Specifications**

### **System’s Functions**

The system was developed in a modular approach. Each module has a set of functions. These functions are listed and summarized at the succeeding section.

#### ***Functional Requirements***

The main functional requirements of the system are the following:

* Displays a campus map of MSU-IIT using WebGL.
* Allows the users to navigate around the campus map.
* Presents information of a specific building in the map.
* Stores building and event information in the database.
* Allows management of data through CRUD functionalities.
* Allows the administrator to search, add, edit and delete building polygons in the map.

#### ***Non-functional Requirements***

The most important non-functional requirements of the system are the following:

* The system requires an internet browser.
* The system supports interaction with mouse, keyboard and touch screen.
* The system will run in both 64-bit and 32-bit.

#### *Map Module*

The map module is responsible for the visualization of the campus map. Visualization will be done using OSM Buildings’ integration of the coordinates of the buildings. Functions of this module are displaying of the campus map in 3D perspective, navigating around the map and displaying building information on the map.

#### *Content Module*

This module consists of two parts: the database and data management through CRUD (Create, Read, Update and Delete) functions. This module is mainly developed using MySQL for the database. This module is responsible for storing all building and event information and management of these data through the use of jQuery.

#### *Draw Module*

This module is responsible for the drawing of low polygon buildings. This lets the administrator draw polygons using coordinates. These coordinates will be saved in the database through the content module. This module is implemented through the use of Leaflet and Leaflet Draw library. This module’s functions include adding a new building polygon, editing the coordinates of a building polygon and deleting a building polygon.

#### *Search Module*

This module is responsible for the searching of all building and event information stored in the system. This module is developed using Datatables JavaScript library.

### **Physical Environment and Resources**

The interactive map is a web-based application that can be run in any internet browser. WebGL is used in visualizing lightweight 3D elements in any platform, thus enabling the system to be accessed almost anywhere as long as the user is connected to the internet.

#### Technical Requirements

System Requirements

Desktop

* CPU: 2.33Ghz or faster x86-compatible processor, or Intel® Atom™ 1.6GHz or faster processor for netbooks
* RAM: (WebGL adjusts to the provided memory)
* Video Card: 1GB of graphics memory
* Operating System**:** Windows XP (or higher), Linux or Mac
* Browser**:** Internet Explorer 8.0 or later, latest versions of Microsoft Edge, Mozilla Firefox, Google Chrome, Opera or Safari

Mobile

* RAM: WebGL adjusts to the provided memory
* Browser**:** Internet Explorer 8.0 or later, latest versions of Microsoft Edge, Mozilla Firefox, Google Chrome, Opera or Safari

#### User Specification

*End Users*

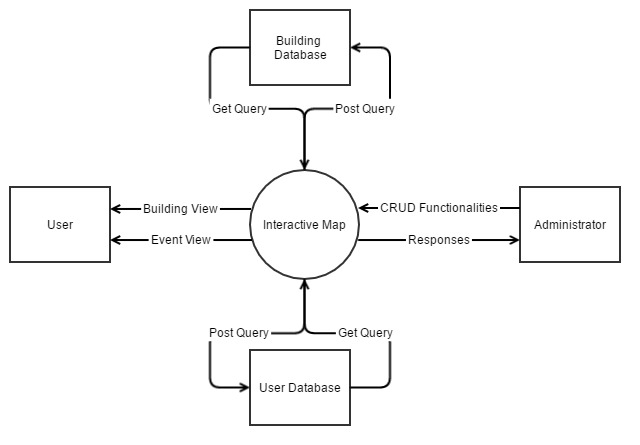
End users can be divided into different user groups: prospective students, students and visitors. These are anyone who are interested to explore the MSU-IIT campus. Educational level and experience is not a constraint but the user should be computer literate enough to know the elements and functionalities of the system.

*Administrator*

        The administrator is capable of adding, editing and deleting buildings and information in the system. S/he is also responsible for the overall maintenance of the system. S/he should be computer literate, has knowledge of how the system works, should identify the different elements and functionalities of the system. S/he should be a member of the Physical Plant Division.

## Design Models

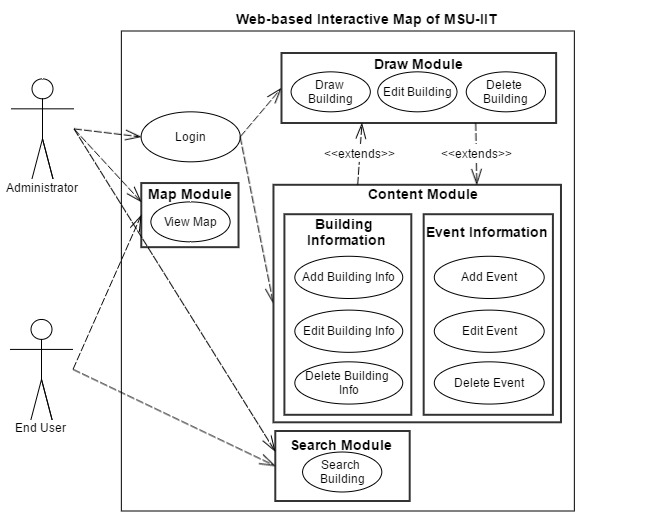
### Context Diagram



**Figure 4.1** Context Diagram

### UML Use Case Model

#### Use Case Diagram



**Figure 4.2** Use Case Diagram for Administrator and User

#### Use Case Specification

##### Login/Logout

**Table 4.1** Login Use Case Specification

|  |  |
| --- | --- |
| **Use Case Name** | **Login** |
| **Use Case ID** | **L1.1** |
| **Description** | This use case describes the actions the administrator will do to log into the system. |
| **Actor/s** | Administrator |
| **Trigger** | The administrator will click the “Login” button. |
| **Preconditions** | All elements of the page are loaded successfully. |
| **Normal Flow** | 1. The application requests for user’s login credentials. 2. The administrator enters his/her username and password. 3. The administrator will click the “Sign in” button. 4. The application validates the username and password provided by the user. 5. The administrator will be directed to the main page. 6. Use case ends. |
| **Alternative Flow** | 1A. The user forgot his/her credentials, s/he will click on the “Forgot Password?” link.   1. The administrator will be redirected to the “Retrieve Password” page. 2. The administrator will input his/her email address. 3. The administrator will click the “Send to Email” button. 4. The application will make the API call for password retrieval. 5. The application will send an email to the email address provided. 6. Use case resumes in step 1. |
| **Exceptional Flow** | If the administrator fails to enter the right username and password, the system will prompt an error message: “The username/password is incorrect.” |
| **Post condition** | The administrator is now logged into the system. |

**Table 4.2** Logout Use Case Specification

|  |  |
| --- | --- |
| **Use Case Name** | **Logout** |
| **Use Case ID** | **L1.2** |
| **Description** | This use case describes the actions the administrator will do to logout of the system. |
| **Actor/s** | Administrator |
| **Trigger** | The administrator will click the “Logout” button. |
| **Preconditions** | The administrator is in the main page. |
| **Normal Flow** | 1. The administrator will click the “Logout” button. 2. The application will prompt the user if s/he is sure. 3. The use case ends. |
| **Alternative Flow** | 2A. The administrator clicks the “Yes” button.   1. The administrator will be back to the login page. 2. Use case resumes in step 2.   2B. The administrator clicks the “No” button.   1. The administrator remains in the main page. 2. Use case resumes in step 2. |
| **Post condition** | The administrator is now logged out of the system. |

##### Map Module

**Table 4.3** View Map Use Case Specification

|  |  |
| --- | --- |
| **Use Case Name** | **View Map** |
| **Use Case ID** | **M2.1** |
| **Description** | This use case describes the actions the administrator or end user will do to view the map. |
| **Actor/s** | * Administrator * End user |
| **Trigger** | * The administrator will visit the website. * The end user will visit the website. |
| **Preconditions** | * The administrator is in the main page of the website. * The end user is in the main page of the website. |
| **Normal Flow** | 1. Administrator    1. The administrator will go to the website.    2. Use case ends. 2. End user    1. The end user will go to the website.    2. Use case ends. |
| **Alternative Flows** | * The administrator can cancel by clicking the “Cancel” button. * The end user can cancel by closing the page. |
| **Exceptional Flow** | If login credentials are incorrect, the website will prompt an error message saying “Incorrect username/password.” |
| **Post condition** | The administrator/end user can view the map. |

##### Draw Module

**Table 4.4** Draw Building Use Case Specification

|  |  |
| --- | --- |
| **Use Case Name** | **Draw Building** |
| **Use Case ID** | **D2.1** |
| **Description** | This use case describes the actions the administrator will do to draw buildings in the map.  This extends with the content module. |
| **Actor/s** | Administrator |
| **Trigger** | The administrator is in the Map Editor by clicking on the “Map Editor” menu item. |
| **Preconditions** | The administrator is logged into the application. |
| **Normal Flow** | 1. The application will display a map with Leaflet’s map drawing tools. 2. The application will also provide a form for the administrator to fill out with information about the building. 3. The administrator will draw a polygon using points. 4. The administrator will fill out the form provided. 5. The application will prompt if the process is successful. 6. Use case ends. |
| **Alternative Flow** | The administrator can cancel by clicking the “Cancel” button. |
| **Exceptional Flow** | If the administrator doesn’t fill out all the required fields, the application will prompt an error message saying “You need to fill out all fields.” |
| **Post condition** | A new building is drawn. |

**Table 4.5** Edit Building Use Case Specification

|  |  |
| --- | --- |
| **Use Case Name** | **Draw Building** |
| **Use Case ID** | **D2.2** |
| **Description** | This use case describes the actions the administrator will do to edit buildings in the map. |
| **Actor/s** | Administrator |
| **Trigger** | The administrator is in the Map Editor by clicking on the “Map Editor” menu item. |
| **Preconditions** | The administrator is logged into the application. |
| **Normal Flow** | 1. The application will display a map with Leaflet’s map drawing tools. 2. The administrator will click on a building in the map. 3. The application will give an option to edit the building.   C1. If the administrator chooses to edit the building:   1. The application will display the points of the building; these points can then be manipulated in the map. 2. The administrator will also replace the information about the building. 3. The administrator clicks the “Update” button. 4. Use case ends.   C2. If the administrator cancels:   1. The administrator clicks the “Cancel” button. 2. Use case ends. |
| **Post condition** | The chosen building is updated. |

**Table 4.6** Remove Building Use Case Specification

|  |  |
| --- | --- |
| **Use Case Name** | **Remove Building** |
| **Use Case ID** | **D2.3** |
| **Description** | This use case describes the actions the administrator will do to remove buildings in the map. |
| **Actor/s** | Administrator |
| **Trigger** | The administrator is in the Map Editor by clicking on the “Map Editor” menu item. |
| **Preconditions** | The administrator is logged into the application. |
| **Normal Flow** | 1. The application will display a map with Leaflet’s map drawing tools. 2. The administrator will click on a building in the map. 3. The application will give an option to remove the building. |
| **Alternative Flow** | C1. If the administrator chooses to remove the building:   1. The administrator clicks the “Remove” button. 2. Use case ends.   C2. If the administrator cancels:   1. The administrator clicks the “Cancel” button. 2. Use case ends. |
| **Post condition** | The chosen building is removed. |

##### Content Module

**Table 4.7** Add Building Information Use Case Specification

|  |  |
| --- | --- |
| **Use Case Name** | **Add Building Information** |
| **Use Case ID** | **C5.1** |
| **Description** | This use case describes the actions the administrator will do to add information about a building. |
| **Actor/s** | Administrator |
| **Trigger** | The administrator is in the Map Editor by clicking on the “Map Editor” menu item. |
| **Preconditions** | The administrator is logged into the application. |
| **Normal Flow** | 1. The application will display a map of the campus. 2. The administrator will click on a building in the map. 3. The application will provide a form for the administrator to fill out with information about the building. 4. The administrator will fill out the form provided. 5. The application will prompt if the process is successful. 6. Use case ends. |
| **Alternative Flow** | The user can cancel by clicking the “Cancel” button. |
| **Post condition** | Information about a building is added. |

**Table 4.8** Edit Building Information Use Case Specification

|  |  |
| --- | --- |
| **Use Case Name** | **Edit Building Information** |
| **Use Case ID** | **C5.2** |
| **Description** | This use case describes the actions the administrator will do to edit information about a building. |
| **Actor/s** | Administrator |
| **Trigger** | The administrator is in the Map Editor by clicking on the “Map Editor” menu item. |
| **Preconditions** | The administrator is logged into the application. |
| **Normal Flow** | 1. The application will display a map of the campus. 2. The administrator will click on a building in the map. 3. The application will provide a form for the administrator to fill out with information about the building. 4. The administrator will fill out the form provided. 5. The application will prompt if the process is successful. 6. Use case ends. |
| **Alternative Flow** | The user can cancel by clicking the “Cancel” button. |
| **Post condition** | Information about chosen building is updated. |

**Table 4.9** Remove Building Information Use Case Specification

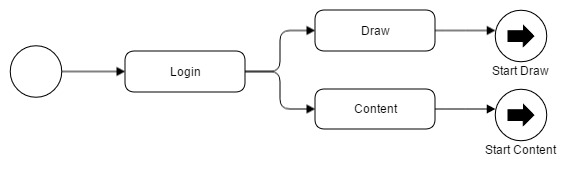
|  |  |
| --- | --- |
| **Use Case Name** | **Remove Building Information** |
| **Use Case ID** | **C5.3** |
| **Description** | This use case describes the actions the administrator will do to add information about a building. |
| **Actor/s** | Administrator |
| **Trigger** | The administrator is in the Map Editor by clicking on the “Map Editor” menu item. |
| **Preconditions** | The administrator is logged into the application. |
| **Normal Flow** | 1. The application will display a map of the campus. 2. The administrator will click on a building in the map. 3. The application will provide an option for the administrator to remove all information about the building. 4. The administrator will click the “Remove Information” button. 5. The application will prompt the administrator if he/she is sure. 6. The application will prompt if the process is successful. 7. Use case ends. |
| **Alternative Flow** | The user can cancel by clicking the “Cancel” button. |
| **Post condition** | Information about chosen building is removed. |

##### Search Module

**Table 4.10** Search Building Use Case Specification

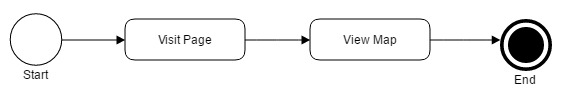
|  |  |
| --- | --- |
| **Use Case Name** | **Search Building** |
| **Use Case ID** | **S6.1** |
| **Description** | This use case describes the actions the administrator will do to search for buildings. |
| **Actor/s** | * Administrator |
| **Trigger** | * The administrator will click on the search bar in the Buildings page. |
| **Preconditions** | * The administrator has logged in and is in the main page of the website. |
| **Normal Flow** | 1. The administrator will input the name of the building inside the campus. 2. The application will present results of the search. 3. The administrator will choose on the list of results. 4. The administrator will be directed to that building. 5. The application will show details of that building. 6. Use case ends. |
| **Alternative Flow** | 1. The administrator inputs an erroneous search. 2. The application prompts with an error message. 3. Use case ends. |
| **Post condition** | The search will be successful. |

### Activity Diagrams



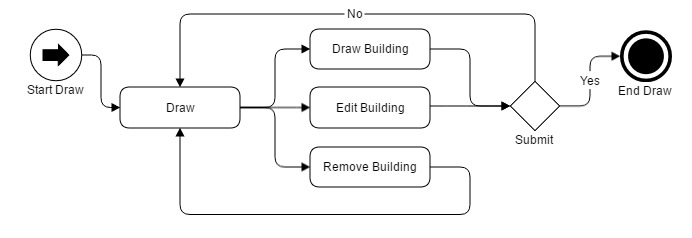
**Figure 4.3** Login Activity Diagram

To control the content within the system, administrators must perform login activity through a special route. Then, s/he can perform draw map activity or add content activity, etc.



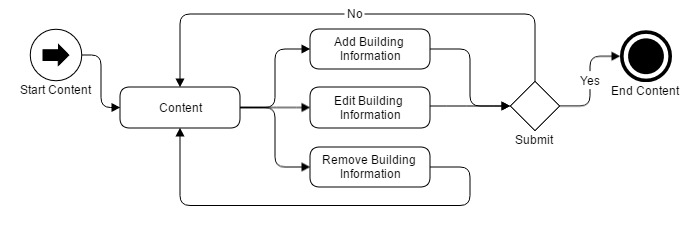
**Figure 4.4** Map Module Activity Diagram

Based on Figure 4.4, web visitors can navigate the main service by just going to its front page. The system then performs a set of task such as requesting data information from the database to render the map module.



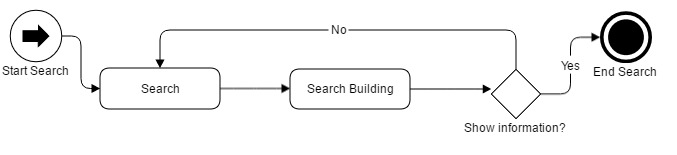
**Figure 4.5** Draw Module Activity Diagram

With authenticated user privileges, draw module enables the user to have draw, edit and remove activity in modifying the data within the system. In Figure 4.5, draw and edit building prompts the user to verify any changes or added information. However, remove building activity would redirect the authenticated user to the main section of the module. Otherwise the activity sufficed then terminated.



**Figure 4.6** Content Module Activity Diagram

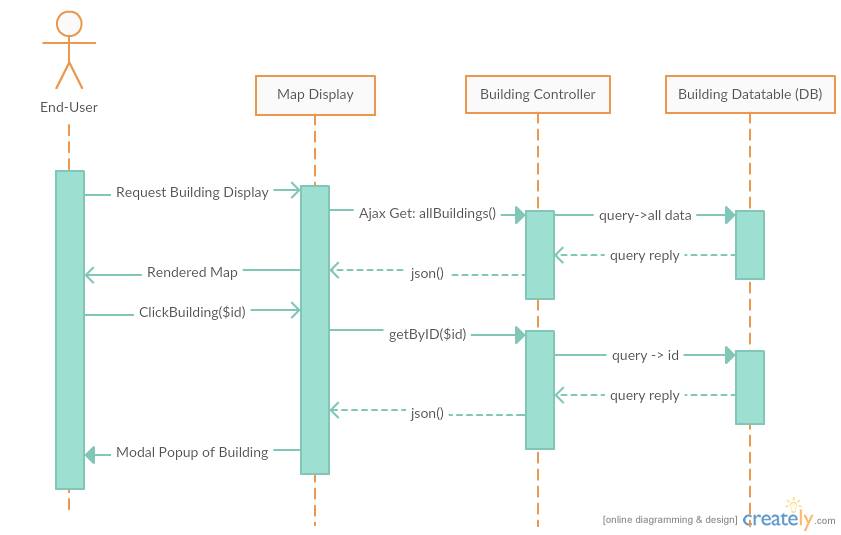
The same with Figure 4.5, the authenticated user can add, edit or delete building information. Add and edit activity also prompts verification message to complete the activity.



**Figure 4.7** Search Module Activity Diagram

Lastly, search functionality was only integrated within the content module thus limited to authenticated user. Users with administrative rights can search and modify the data by clicking it to show its information. Until then the search activity terminates.

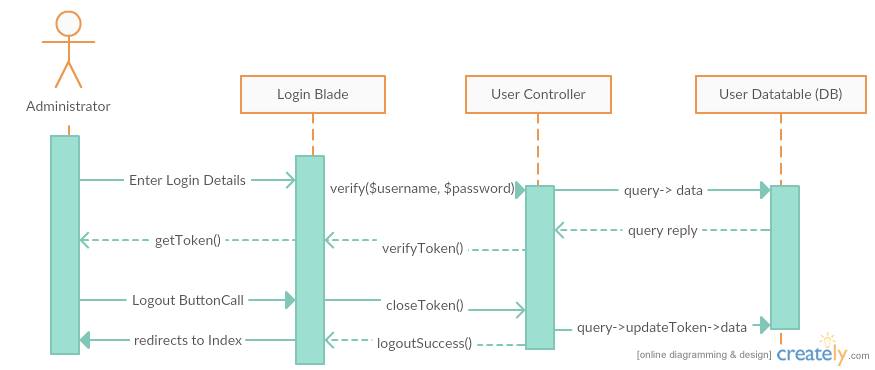
### Sequence Diagrams



**Figure 4.8** Sequence Diagram for End-User

Through Laravel framework, requesting data is easily achievable. As shown in Figure 4.8, the End-user, when visiting the front page, issues a request in his/her device to the browser, the browser which handles the Map Display as view component then sends Ajax Get request on the building controller. The building controller then sends query to the database of structural data and so on. The database service then queries the reply. The controller converts the data in JSON format which then the view component arranges and converts into relevant and readable information for the user.

Same goes when the end user clicked on a selected building, the module interprets a get request, the controller interprets the request and the database issue a query.



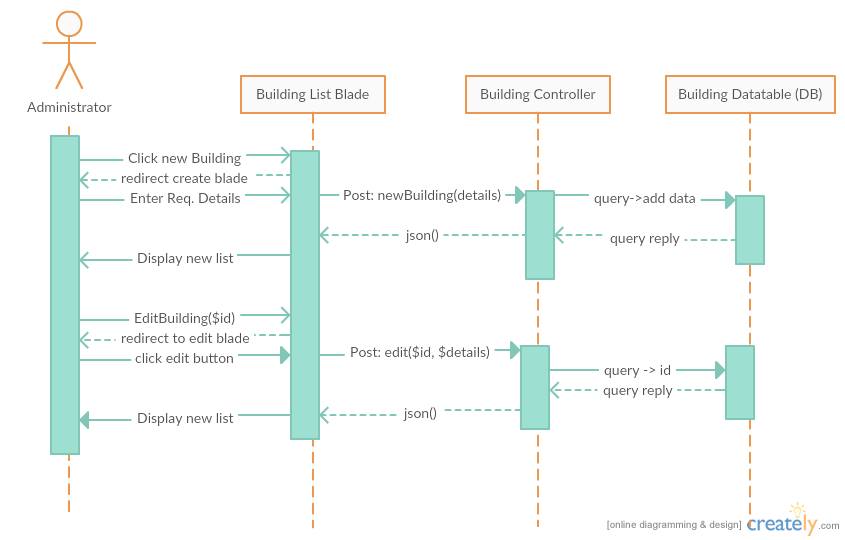
**Figure 4.9** Sequence Diagram for Administrator Authentication

To authenticate a user, s/he must enter a designated login details. The view component which is the login blade will issue a verification to the user controller, then the controller queries the request to the database. The database will issue a reply the controller repacks the reply through verifyToken function. This will send a set of string as token for the user activity.

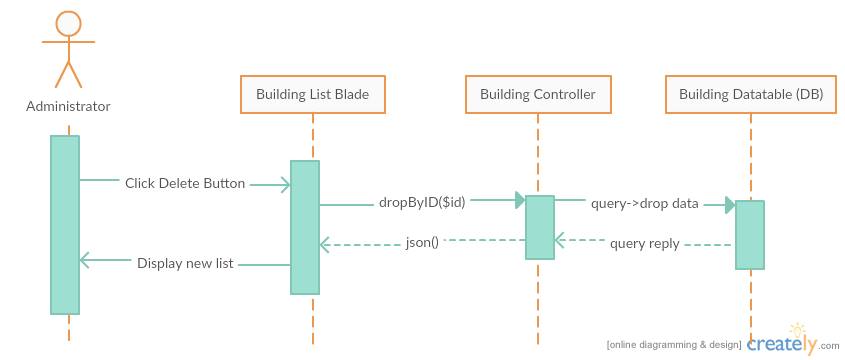
The logout button will be present in the content blades however its functionality runs with the controller. The controller sends a query to the database at the same time closes the verified token of the user and redirects the user to the front page.

In creating building data requires the user to navigate to the create blade view component. Upon redirection to the right blade, the admin may now enter the required details. The data made will be posted to the controller then will be added to the database. The database will then send a reply which will be packed into JSON format for the view component to display the newly made data.

Additionally, in Figure 4.10, edit buildings follow the same steps of creating buildings but requires to choose on what building to edit.

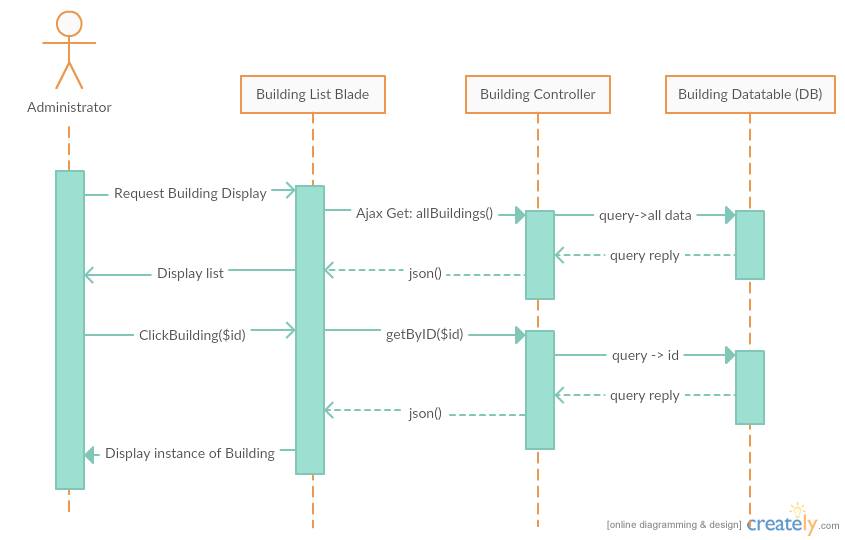


**Figure 4.10** Sequence Diagram for Administrator Create and Edit



**Figure 4.11** Sequence Diagram for Administrator Delete

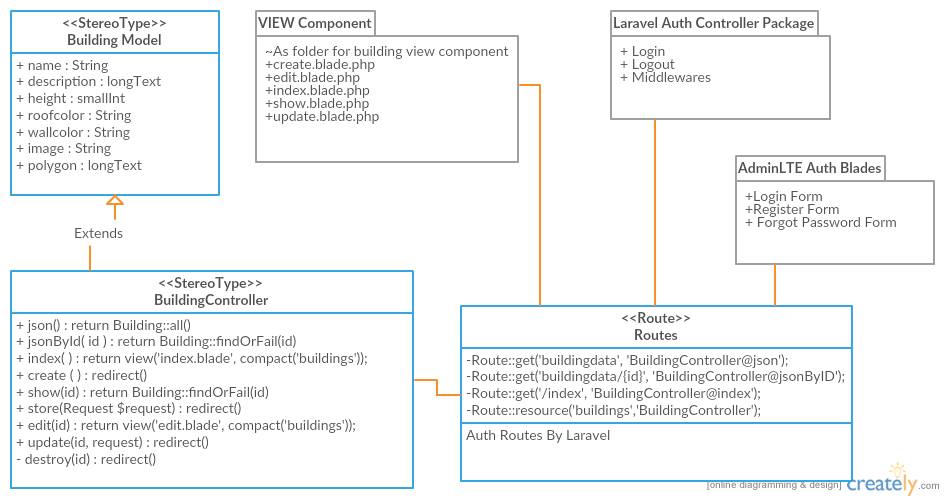
Finally, the administrator can delete selected data by a push of a button. The blade will instantly interpret the data to be dropped and will be sent to the building controller. The controller then queries to drop the data from the table and after which the database queries a reply. The controller issues another JSON format data to refresh the list.



**Figure 4.12** Sequence Diagram for Administrator View

The administrator follows the same sequence with End-user in viewing data but the difference is the blade component used. It can be explained with the descriptions in figure 4.8.

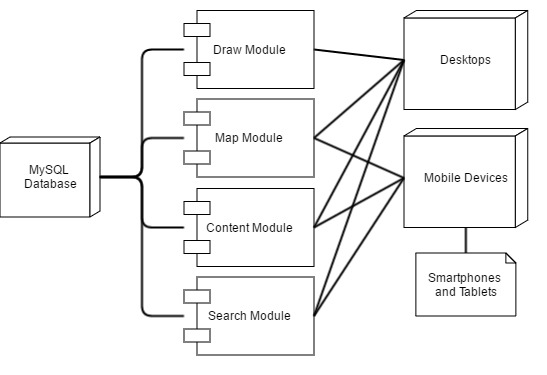
### Class Diagram



**Figure 4.13** Class Diagram of the Developed System within Laravel Framework

Laravel framework follows an eloquent class hierarchy. Models are constructed through Laravel own model class interface. Additionally, it is already connected through a database management system but this project uses MySQL services. Controllers are also extendedly created from an interface of the Laravel services. And routes are premade classes that act as an intermediary to address route, view components and controllers. With the same practices, the project will make use of pre made packages within AdminLTE framework and Laravel for easier build.

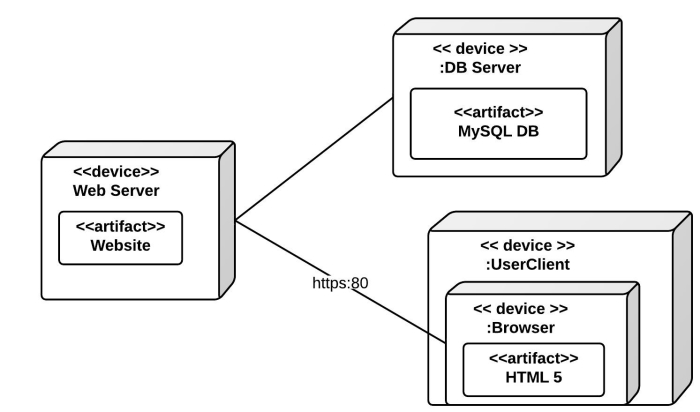
### Component Diagram

****

**Figure 4.14** Component Diagram

Through Figure 4.14, selective modules only work on specific devices, whereas the map, content and search modules are the only modules available for mobile devices. However, all of the modules can be used for desktop and larger screen device users.

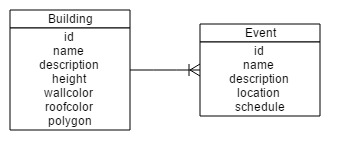
### Deployment Diagram

****

**Figure 4.15** Deployment Diagram

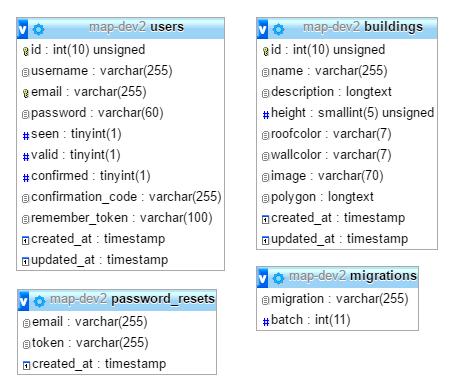
### Database Design Model

#### Entity Relationship Diagram



**Figure 4.16** Entity Relationship Diagram

#### Schema

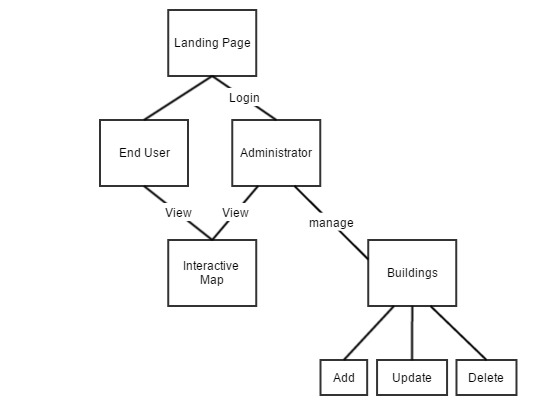


**Figure 4.17** Database Schema

## **Structure and Interface**

This section discusses the system functions through screen shots. Walkthrough to the system’s interface will be reflected here.

### Program Structure



**Figure 4.18** Program Structure of the System

## **Development Phases**

### Pre-development Phase

On a heuristic approach, the project developers established the system that introduces different visualization of buildings and structures of MSU-IIT.



**Figure 4.19** Interactive Map Prototype using Google Maps

Initially, the system used polygon information integrated on Google Maps whereas it highlighted different sections in representation of 2-dimensional shapes within Google Maps. In line with this, the developers aimed to represent buildings with another layer of representation thus the problem for isometric visualization from GIS MSU-IIT. The project then introduced isometric visualization by elevating a duplicate copy of the shape of an area added with additional polygon shape as side or visual perimeter for the represented buildings. However, isometric visualization requires the underlying Google map to be in a 45-degree angle view in order to best represent the buildings and structures.

Isometric representation was limited on Google Maps because some areas were not covered with 45-degree map visuals especially in the Philippines. Google Map requires additional financial cost to developers for its different services particularly in 45-degree visual data. This lags the development process thus different programming and geographical map library was searched.

### Prototype Development



**Figure 4.20** Interactive Map Prototype Using OpenStreetMap and OSM Buildings

In contrast to Google Maps and its API library, OpenStreetMap was chosen to be the geospatial data provider for this project. OpenStreetMap is a free open data system which has limited data regarding the Philippines.

Using Leaflet.js library, the OSM canvas was efficiently controlled and few features of the library was integrated to the system for the functionalities needed in the system. OpenStreetMap API can read GeoJSON format hence illustrating 3D low polygons of buildings and structures of the campus is easier to integrate. It was verified by seeding data of the buildings in the prototype. With this, a database can be expected in the system to contain these 3D data and other content.

### Summary of the Development Process

The prototype was integrated with Laravel 5 website development framework. These include GUI designs by using the AdminLTE template to the environment and migrating sample data to the database was done through Laravel database seeding services. Additionally, models in a form of classes to represent each data entity and controllers for routing and information control were also made through the same service.

With this done, the requirements of making the modules was first supplying the view component through GUI designs. Additionally, models of buildings and user data were created for the view component. These models act as an intermediary component to the database whereas the entity variables were declared in the model classes. These variables then were used by the controller component to be routed to the different parts of the view components.

In the map module, integration of multiple libraries was done mostly within the view components since these were JavaScript libraries and require less in the model component management. First, OpenSteetMap was added to render the map tiles and other services. OSM Buildings was then added to render 3D models of the building structures, however in the initial prototype the system used an external file with GeoJSON data of the structures. Lastly, Leaflet library was added for functionalities present in the map. To make this component dynamic since it was using a static information in displaying 3D maps, an additional code using AJAX to get building data through a selected route then converting the retrieved data into GeoJSON format. With this, a selected address route for building structures was required for this module to work.

The content module was created by following the MVC framework. First, models and its data variables were made using the Laravel Eloquent programming technique which followed SOLID principles. Initially, building information was migrated to MySQL database with a model class within the system directory folder. Inside the model class, data variables were declared. With this, building controller was made to use this model for data management and other end-user functionalities as services. The building controller contains different functions or methods that channels different data based on the requirements. In instance, a function that returns building structural data to complete the map module services was added within the building controller. Additionally, an external class called route.php routes these data by providing a registered address to where they are addressed. Lastly, multiple view components or Laravel blade pages were made to show the controlled data. In an instance, a blade for creating building data was accomplished by registering a routed data that is connected to a controller method within the building controller.

To complete the component module, CRUD functionalities were followed where as a create page was made for creating building data, a list page was made for displaying the list of building data, update page was made for updating the selected data and a delete function to delete a selected data. These pages have their own methods within the building controller and can be replicated for other entities as content.

The draw module is a specialized module of content and map modules since it follows the same mechanics however offers more complex services since this module relies on the creating and updating of a map and its contents. Integration of the same services was done. Additionally, Leaflet draw functions was used to retrieve the data created and updated on the page to be posted through the route class then to the content controller.

Lastly, a search module was integrated within the content page for the buildings. Yajra Datatables services was integrated for the search functionality for the content and can be integrated to other services or content.