**Work Order System**

Technical Manual

CLASS: CPSC488

04/16/2023

Trevor Bell trevorbell030@gmail.com

Ellie Wurst bmw1030@sru.edu

John T. Friend Jr. jtf1011@sru.edu

Hunter Minteer

Overview of this Document

This document’s purpose is to give a more technical understanding of the various files and interactions happening within the project, as well as including UML diagrams to grasp the construction of the project.

**Important Classes and files**

WorkOrdersApplication.java - This is the main file, when you wish to run the program, this is the file you should look for.

~~~~~~~~~~~~~~~~~~~~~~~~ CONTROLLER FILES ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

WorkOrdersController.java - This is the controller part of Maven, denotes the links in the project and can take data from the model and give it to the HTML to view, input, store, or anything else that will need to be done or acted on by the data.

AdminController.java – This controller is used for handling all admin files and actions for admin users. It also has functionality for uploading and creating new buildings, rooms, and departments.

ForgotPasswordController.java – this controller is used for handling the forgotten password functionality for users.

TechController.java – handles information for the technician role within our project.

ManagerController.java – handles information and URL data for the Manager role within our project. Managers may also upload excel data like Admins.

ExcelController.java – This is what holds the URL data and code for uploading different excel files.   
~~~~~~~~~~~~~~~~~~~~~~~~~~~ DOMAIN FILES ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Archive.java – This is to setup a MySQL table for storing past incidents and keeping a storage of previous incidents.

Buildings.java – This is used to set up the buildings table for buildings, the buildings\_departments sub table with is a joint table of building and department id’s as well as buildings\_rooms with building and room ids.

CustomUserDetails.java – this table stores a user’s role, authentication status, password, username, and name. It’s a side table used alongside the users table to allow for authentication status of a user with unique details per user.

Departments.java – This domain is used for setting up the departments table. The departments table stores a department and its id.

Incidents.java - holds information for the incidents that a user/admin/technician can create or send in, if anything needs to be added or removed, it would be here.

Role.java - Creates a role id that is given and attached to users, also generates a users\_roles table that will show the relationship between the two of them.

Rooms.java – table for the rooms connected to a building; this file has the config for the rooms table which contains a room and an id for each entry.

Users.java - This is what sets the idea for a general user, if anything needs changed within the user (such has giving them a new property) then it will go here.

~~~~~~~~~~~~~~~~~~~~~~~~~~ EXCEPTION FILES ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

FileNotFoundException.java – this is what occurs if a file cannot be found. The function will run if so.

FileStorageException.java – this is what occurs if a file is too large / unable to be stored in the database.

~~~~~~~~~~~~~~~~~~~~~~~~~ REPOSITORY FILES ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

BuildingRepository.java – This is the repository for the building object and buildings table.

DepartmentsRepository.java – This is the repository for the departments object and departments table.

IncidentRepository.java – This is the repository for the incident entity and its table.

RoleRepository.java – This is the repository for the role entity and its table it is connected to.

RoomsRepository.java – This is the repository holding information on the rooms entity and its table.

UserRepository.java – This is the repository holding information on the user entity and its table it is connected to.

~~~~~~~~~~~~~~~~~~~~~~~~~~~ SECURITY FILES ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

PasswordConstraintValidator.java – this file is responsible for defining the requirements of a password that it must fulfill in order to be accepted by our application.

ValidPassword.java – ensures that a password being put into the system is valid.

WebSecurityConfig.java - is the file that makes sure that people accessing different URLs and aspects of the project have the proper authorization and ability to access it. Will work with the data.sql file to determine which roles a user can have and what pages they are allowed to access.

~~~~~~~~~~~~~~~~~~~~~~~~~~~ SERVICES FILES ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

AdminServices.java – This class has important functions used for admin functionality. It currently has enabling, disabling, and deleting user accounts within it.

BuildingsService.java – This class has functionality for linking departments and rooms to a building as well as the functionality for saving, listing, getting, and deleting buildings.

CustomUserDetailsService.java – This class has functionality for CustomUserDetails.

DepartmentsService.java – This class has core functionality for departments and its needed saving. Listing, getting, and deletion of departments.

ExcelService.java – This class contains information for excel file reading and cell information differentiation.

IncidentServices.java - will save incidents so that the data can then be taken and shown to the user who has reported them.

RoomServices.java – this class contains information on necessary functions for rooms. It primarily is functionality for being able to save, list, get, and delete rooms.

UserManagementService.java – this class contains information for manager type users and their core functionality.

UserNotFoundService.java – this class contains functions for when an entered username does not exist in the SQL database.

UserServices.java - This class will run when a user needs to be registered so they get added to the database, as well as attaching a default role and sending emails to users when they need an email.

UtilityServices.java – This contains utilities that are used in various locations within the program.

Excel datasheets are kept in the data folder of our project where they are stored. The program is currently setup where you need to follow their naming protocol for the different types of data being entered into it.

Data.SQL - This will seed the database with some pre-determined information, data can be added and removed to suit the needs of people using the program, and works with the WebSecurityConfig file to determine which roles exist in which order (note that all users are given the default role of “User” located \*placeholder\*)

Application.properties - Holds critical information for running the program and there are two variations located within the project. Although the only information that is shared between them is the SMTP information, which must also be configured with a viable Gmail and special application password.

* H2Application.properties.txt - replacing the information in this with what is in the normal file, will let it run with H2 compatibility, a memory-based database service that has no persistence, and will drop all information between each run.
* MySQLApplication.Properties.txt - using the information located in this file will give the project mySQL support and must be installed on the server to be used properly. Also note that there is a username and password which must be set to the username and password that is used to access mySQL on that server the project is being run.

1. Interactions Between Classes

Since this project uses an “MVC” format, which stands for “Model – View – Controller” there is a general rule with how classes will be interacting with each other. The Controllers will be doing most of this interaction, as the Controller pulls information from the models (or objects) in the Java project to display in the View (the HTML pages).

But the Repository files are really what make this project work, they are what allow a user to pull information from a connected database, a plethora of different methods are available without any extra work done by the user, but others can be added to achieve different purposes. They also include the option to save new data into them, making them highly dynamic and versatile, and learning to use these properly is critical to the project, since it fully relies on stored and connected information within the database to show, load, and save data.

1. UML diagrams

This section will detail different UML diagrams for reference to this project, such as use-case and sequence diagrams.

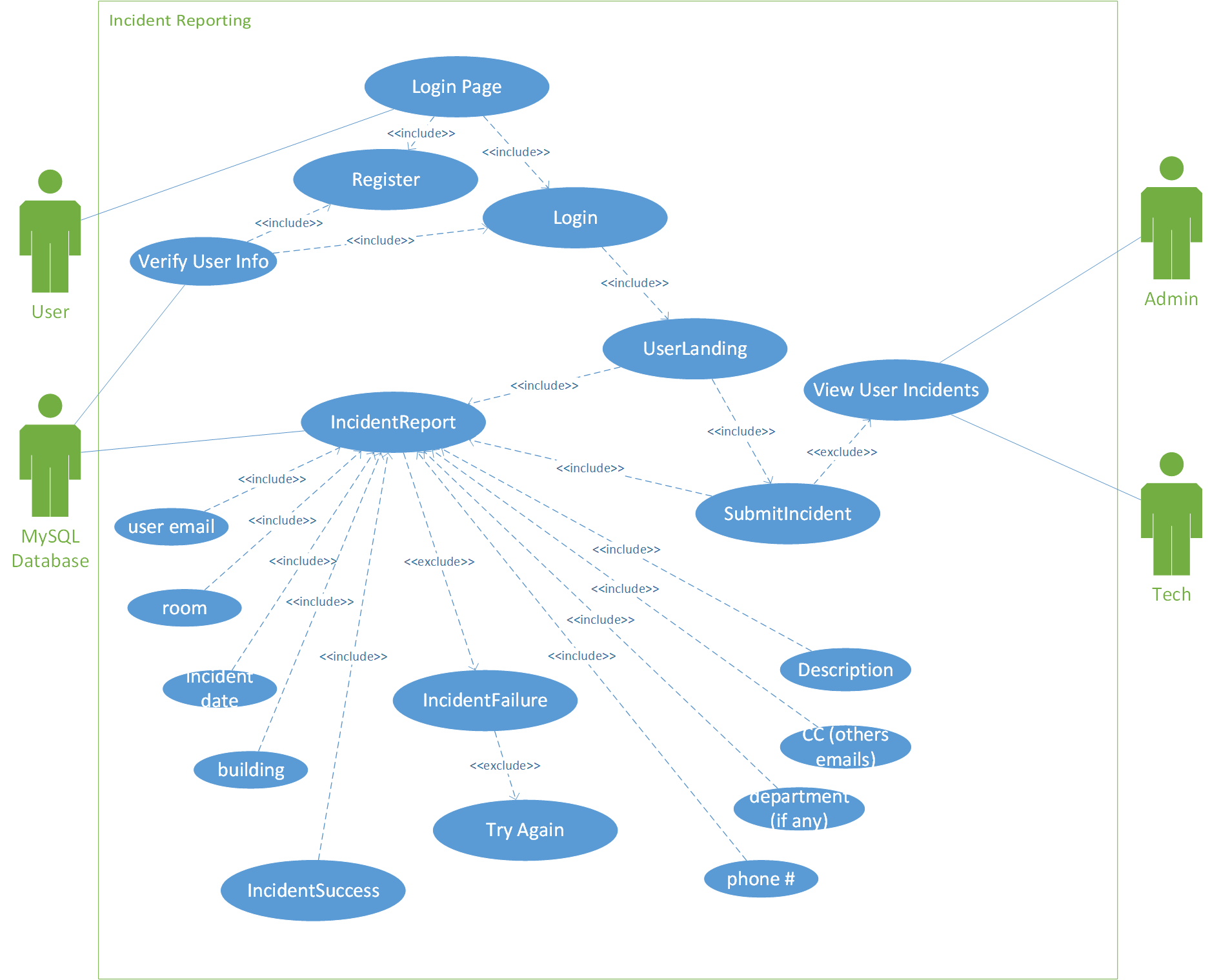


Figure 1. Use Case of Incident Reports

Diagram

Description automatically generated

Figure 2. Use Case of Login

Diagram

Description automatically generated

Figure 3. Use case diagram for admin functions.

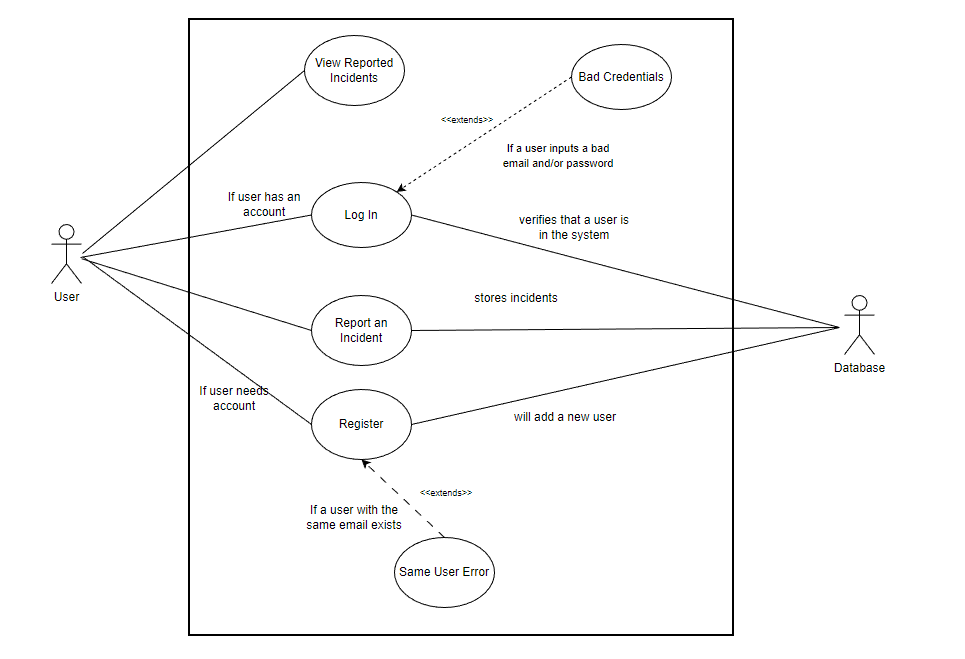


Figure 4. Use Case of a User Registering and creating an incident

Timeline

Description automatically generated

Figure 5. Sequence Diagram of Incident Reporting

Diagram

Description automatically generated

Figure 6. Sequence diagram of a user logging in

Diagram

Description automatically generated

Figure 7. Sequence diagram of a user requesting password change

Timeline

Description automatically generated

Figure 8. Sequence diagram of a admin freezing an account

Diagram

Description automatically generated

Figure 9. Activity diagram for administrator.

Timeline

Description automatically generated

Figure 10. Activity diagram for manager.