Scheduling Work Orders

Final Report

April 23, 2023

Slippery Rock University

Trever Bell [tlb1024@sru.edu](mailto:tlb1024@sru.edu)

Ellie Wurst [bmw1030@sru.edu](mailto:bmw1030@sru.edu)

John Thomas Friend Jr. [jtf1011@sru.edu](mailto:jtf1011@sru.edu)

Hunter Minteer [hgm1005@sru.edu](mailto:hgm1005@sru.edu)

Group Number: 1

Class: CpSc488 section 01

Contents

1. Problem Explanation………………………………………………………3

# System Overview

# Roles within the Program

# User

# Technician

# Manager

# Admin

# Problem Identification

1. Completion Status……………………………………………………………4

# System Evaluation

# Contributions

# Remining Open Issues

1. System Requirements………………………………………………………6

# Hardware Requirements

# Software Documentation

1. Future Work………………………………………………………………..7
2. UML Diagrams…………………………………………………………….8
3. Caveats and Minefields…………………………………………………….13
4. Documentation……………………………………………………………..14

# Method Documentation

# Class Documentation

# External Documentation

# Javadoc Documentation

1. Code Reusability…………………………………………………………..14
2. Testing…………………………………………………………………….15

# Human-Computer Testing

# Whitebox Testing

# Unit Testing

1. Post-Mortem Analysis………………………………………………………16
2. References…………………………………………………………………..16

## Problem Explanation

**1.1. System Overview**

This Work Order System program was created to allow users to create and view incidents. While also giving other roles such as Technicians and Managers the ability to edit, create, and assign technicians to incidents. Admin accounts exist to control the flow of user data and accounts.

Since the program is about creating incidents, the data in the system and the connections between said data are integral. Buildings are the main object and connect with departments and rooms. Some connections are made automatically by the program, and the data can be connected manually as well. However, if this can be done by uploading a single excel file, that would speed up the process of the program.

**1.2. Roles Within the Program**

**1.2.1 User**

Users are the most common role in the program, and they can create and view their created incidents. Once any new account gets registered into the program, they are given this role unless changed by an administrator account.

**1.2.2 Technician**

Technicians are the main worker role of the program. They will consist of all the people who are assigned or working on the incidents. They can self-assign themself to incidents, as well as change their status, update the incident, and close it.

**1.2.3 Manager**

Managers have control over the incident portion of the program, they can create new incidents, edit existing incidents, as well as assign any technician within the project to an incident. They can also upload excel files and connect departments and rooms to buildings.

**1.2.4 Admin**

Admin Roles has control over users and their information. They can create new users, as well being able to activate or deactivate any user account, as well as deleting them. Along with Managers, Admins can upload excel data, and connect existing departments and rooms to buildings.

**1.3. Problems Identification**

While the Work Orders System has basic functionality and usability with all roles, data connection, and incident creation. There still exist potential improvements that can be made to the program to improve the effectiveness and speed at which data can be loaded and Users can interact with the program.

* Automatic Data Connection – As it stands data can only be connected by admins and managers by hand, and only a single building at a time. If connections can be made via an uploaded excel file, that would improve the effectiveness of the program.
* Infinite Login Attempts – Any person can attempt any number of times to login to the program, leaving a security risk within the program and to user or incident data.
* Form Resubmissions – Throughout the program there are several back buttons, which are meant to take the user to the previous page they were on. However, if the current page has a form in which to submit (I.e., the incident reporting page) they will be taken to a page which appears to be an error. The main issue with this is that it is not consistent, so learning what makes the page appear would be the first step.
* GUI inconsistency – This is a non-functional issue as it stands, but a more consistent GUI or an improved GUI (such as making the options available to the roles a side bar, rather than a set of buttons on a landing page)
* Updates on Incidents – While updating an incident, only a single incident can be stored at a time, and there is no timestamp on the incident, so a user has no idea when or what number update the technician is on to properly track their incident.

**2. Completion Status**

Completion Status: Incomplete

The system evaluation section will go into more detail with this Completion Status

**2.1 System Evaluation**

While the program has the basics required for a Work Order System. The program resembles more of a beta build of a more complex and advanced project, so we have decided to mark the project as incomplete at the end of this semester.

This decision was reached because of the issues remaining in the program, while not critical to the running of the program, are important enough for our group to determine this project as incomplete.

The primary issue we feel that is holding this project from being considered complete is that of the data connections and Excel file uploading; while both do work on their own, they are not working in tandem to connect the data between Buildings, Rooms, and Departments without being manually done by an Admin or Manager. Or the objects which get connected by launching the program and going to our main login page.

The rest of this section will be dedicated to showing off and explaining the contributions that were made to the project in this iteration, as well as showing the issues which remain open. Future work to be done will be explained in a later section.

**2.1.1 Contributions**

This section will be a compressive list of critical issues which were created and solved for the program within the context of this semester. The numbers stated here correspond to the issue number on the GitHub issues page. They are also given a short description to explain more than the issue name alone. (Most issue names are shorted for brevity’s sake)

Issue #6: “Login Needs Connected to the DB [Database]” - User information is stored from registration and used when logging in to confirm the correct user is logging in.

Issue #8: “Users can Put in a Workorder and then check on the status of the workorder and be informed via e-mail when the workorder is done.” - Users can submit an incident, then view all incidents that they have submitted.

Issue #10: “Need login for user, administrator, managers, and workers” - The roles are an integral part of the program, and their main uses and features are explained in their section above.

Issue #11: “Spring Boot Project, POJO’s” - This is simply referring to the fact this program was created in Java, with use of the Java Spring Boot to easily develop the project for use with web applications.

Issue #55: “Options should be loaded from a database and shown through Thymeleaf” - This is the basis for most dropdown menus in the project, using HTML Thymeleaf in conjunction with JavaScript to populate dropdowns with database information.

Issue #74: “Give Technicians the ability to self-assign and complete incidents” (Abbreviated) - Since technicians are the main working class, they need to ability to be connected to and close incidents, so that any incident reported can be solved.

Issue #75: “Admin have a user list to update, edit, reassign roles and delete users” - Admin can control the information within users, or delete their account.

Issue #102: “Manager role who can control incidents” - Much like how admin control user information, the added Manager role is able to control incident information, as well as making new ones.

Issue #120: “Excel data uploading” - Excel data should be the main method in which data is being added and connected within the system, as the program relies on the information within the database to work properly and fully.

**2.1.2. Remaining Open Issues**

While these issues are still open, they are not critical, and are here for documenting the remaining aspects of the project, any future issues and potential fixes are covered in their own section or within the evaluation manual.

Issue #73: “More information on FAQ pages” - Since a basic issue system was the basis for this program, we had added a list of FAQ pages which then lead to a report form if they require more assistance. The information on these pages' ranges from little to nothing, and more would mean less trivial incidents for technicians to work on.

Issue #69: “Upload UML diagrams such as the use case diagram, and documentation updates” - while UML diagrams are included with the project, there is always more information to add to other documents, especially as more features get added.

**3. System Requirements**

Our project's purpose is to create an incident reporting application that allows users to report when and where they have various issues that they need assistance with. The variety of issues can be e-mailing issues, general technical issues, hardware problems, and other things of this nature. Our software should be able to allow a user to login, report their issue, check the issue status, and receive a response when it is done via e-mail. When a user is filling out a report it should request the building, department, and room number where it may have occurred, and any additional information on this specific incident. Users can do this primarily by using various ‘fill in the box’ and drop-down style menus to make it an easy and understandable process. Once an incident report is requested, the user should receive an e-mail confirming that their request went through specifically when it is submitted and receive an additional e-mail when it is marked as finished. They also can obtain additional e-mails as the process is being undergone in the event of being asked for more information.

The project will have an SQL database that stores the information of users, roles, incident reports, and location info within the database. It will have to separate incidents based on the type of incident and severity of the issue occurring. We will have our application have a general login page where admins, managers, technicians, and normal users will all login at the same central portal, and based on the verification they have been given when registered into the database they will have access to different information. A user should only be able to file incidents, check currently open ongoing incidents they filed or might have been cc’d in, and lastly, they will be able to see past incidents they requested. A technician should have the ability to look at all incidents that have been assigned to them as well as the ones that haven’t been assigned yet. Technicians can change the status of an incident to active and closed, update an incident, and assign currently unassigned incidents to themselves. A manager would be able to see a layer deeper with the ability to look at all the incidents with the ability to close them, assign technicians to work on these issues, and the ability to upload location data through excel and the ability to manually add, edit, or delete. An admin should only be able to manage users, whether it's deleting, freezing, or registering new users and assign them any role, be that normal user, admin, technician, or a manager. Every user, technician, and Manager should be able to file incidents.

The program will likely consist mostly of configuring various HTML files to work with our backend Java via the Controller java file since it seems like we will need a variety of HTML pages to cover the range of issues, and page needs for having a well-built and easy to use application for our users. It will have a lot of the work going on inside of the MySQL database and the various tables we will need to make and create as well as connecting the HTML to work with the Java code in a fluid manner that allows for our application to work in a secure, efficient manner. The Java side of this project will primarily make use of Maven and Spring Boot libraries to interact with both the HTML files, and the mySQl database that our information will be stored with.

**3.1. Hardware Requirements**

**Hardware –** Most if not all modern-day computers could run our program since it is not very intensive.

* Must needed hardware specs.
  + At least 2 GB ram (while running eclipse uses about 1 GB)
  + Windows 10 or higher.
  + 150Mb free on hard drive. (Eclipse alone)
  + A valid Gmail account.

**3.1. Hardware Requirements**

**Software**

* Must needed software.
  + Eclipse IDE for Enterprise Java and Web Developers – 2022-06
  + Java edition – JDK 17
  + SQL Workbench

**4. Future Work**

This section will talk about future additions or changes that can be made within the project to improve it.

1. In-System Notifications – All roles would benefit from a system in which they can see notifications and updates in the system, rather than through emails or not at all as it stands currently

USER would benefit from this as they could see their incidents getting updated or closed quickly, as opposed to having to wait for an email, which always has the chance to fail on send. As well as being able to see changes to their account, such as being made aware of a role or status change.

ADMIN would benefit from this as they could get updates about users faster, coupled with the password attempt checking to know if someone is attempting to access the project who shouldn’t, or simply needs a password change or other issues which may occur.

TECHNICIAN would benefit as they can be assigned to incidents by managers, and being made aware of any new incidents they are assigned to would be important.

MANAGER would benefit from this as they can better keep track of incidents and how the technicians are getting along in their currently assigned incidents. They could also be in contact with admin in relation to new information being added to the excel files, to then be uploaded into the system to connect with buildings.

2. Stronger FAQ pages – Since our system is all about bigger issues that a user may encounter while on campus, having several FAQ pages can help reduce the overall workload of managers and technicians, allowing users to find their own solution before creating an incident.

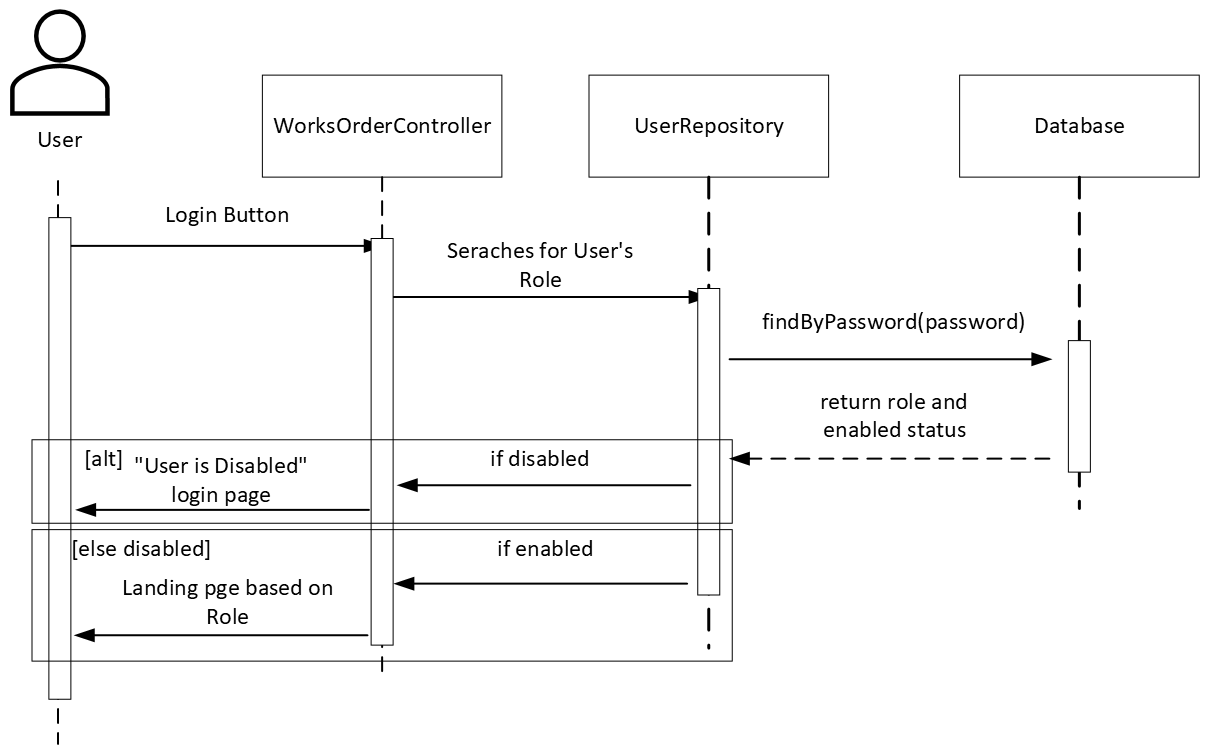
3. Unique Role Profile Pages – As it stands all roles rely on the same profile page to show basic user information, but more information and options would be helpful to giving each role more actions to do within the system, such as contacting admin/managers, or the option to change more than their password.

4. Side Menu Navigation – Only a handful of buttons are what roles are allowed to use to make their way across the system, having options in a side menu would not only make navigation easier, therefore increasing the speed at which work can be done. But it would also clean up the number of stray buttons around the project, improving the user experience with the system.

5. Mobile application – For use with technicians to be made aware of the problem there are going to be solved if they must go to the room in person to solve the problem while on campus.

6. Improved Data Connection – Allowing connections to be made via uploading an excel file would reduce the workload on the admin and managers to connect it manually.

**5. UML Diagrams**

Figure 1. Sequence Diagram of any user logging in.

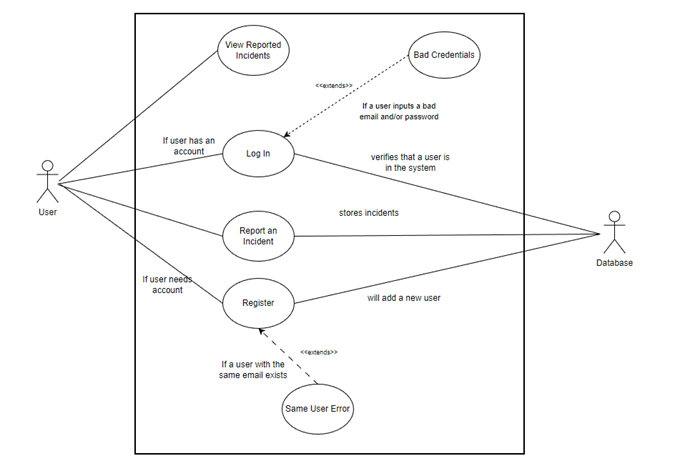


Figure 2. Use case diagram of a user logging in and creating an incident.

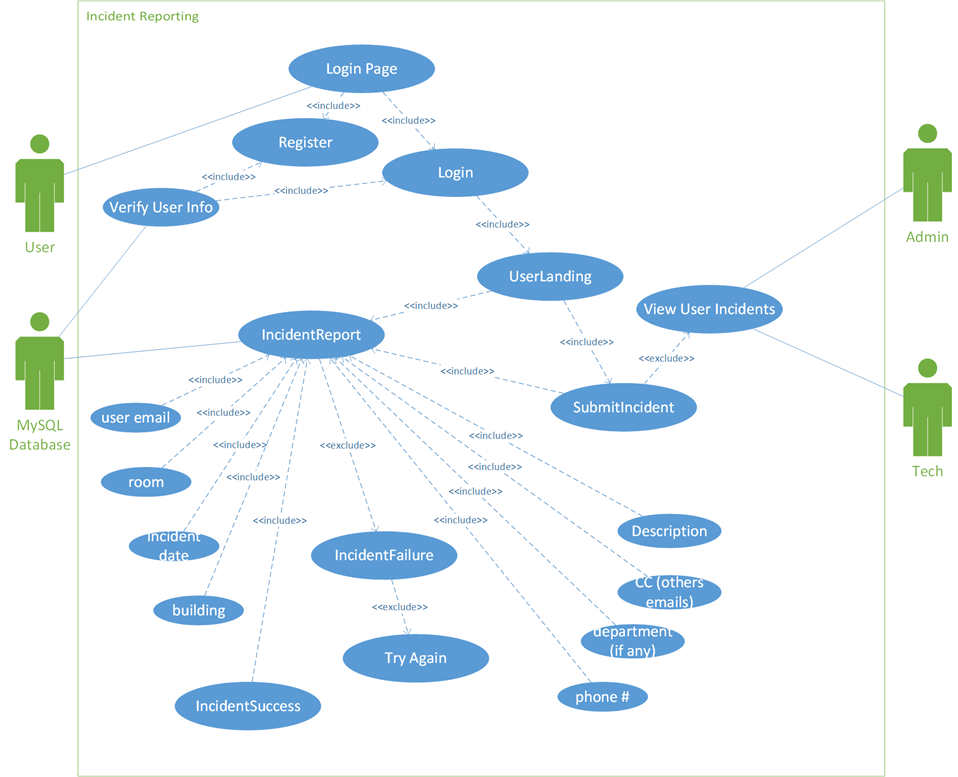


Figure 3. Use case diagram for a user reporting an incident.

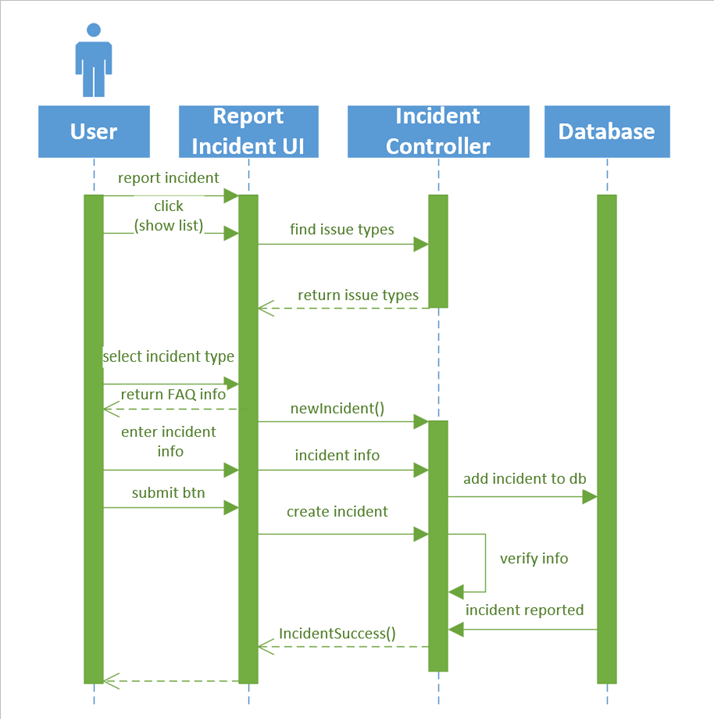


Figure 4. Sequence Diagram for a user reporting an incident.

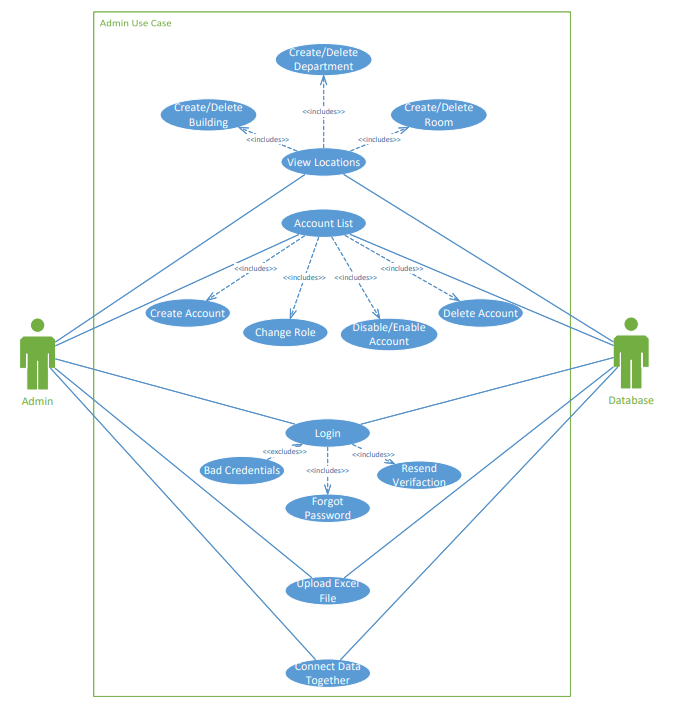


Figure 5. Use case diagram for admin functions.

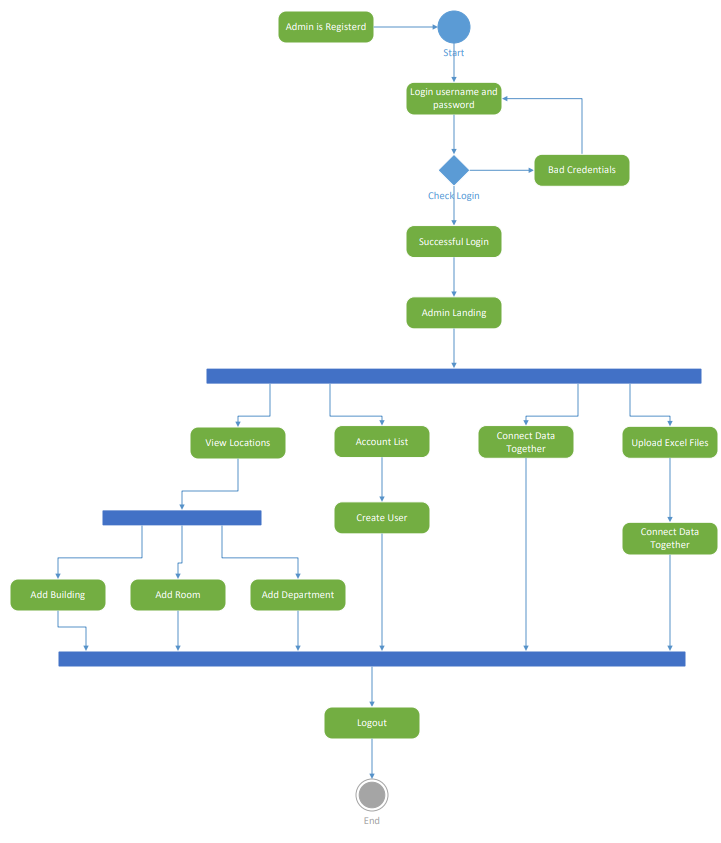


Figure 6. Activity diagram for administrator.

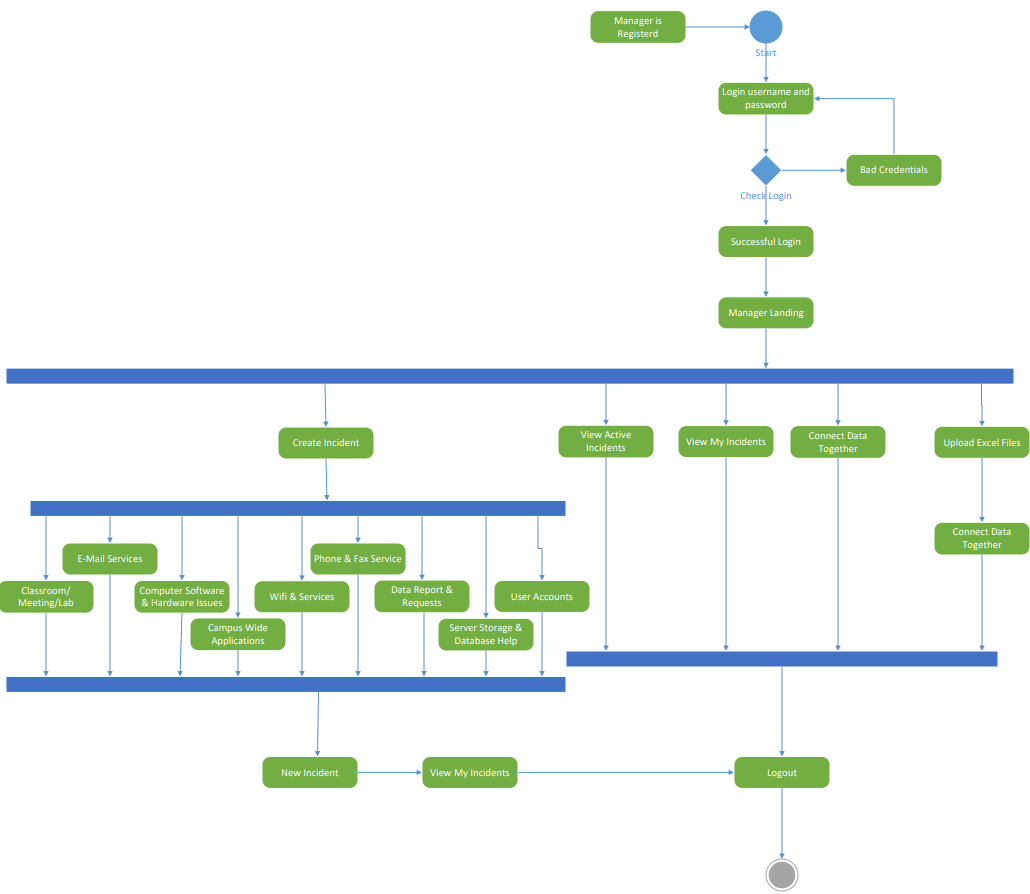


Figure 7. Activity diagram for manager.

**6. Caveats and Minefields**

* **Build Path** – Currently in our project when you first download the program and go to run it without changing the build path it will error. This error is not always present but most of the time it must be fixed before you can launch are program.
  + How to fix-
    - Open the program and in the project, explorer right click on the second line down that says WorkOrders.
    - After right clicking go to build path and then select configure build path
    - Once the new menu has opened you want to look for this:
    - If it shows Excluded(none) then the issues is not present, but if it has Excluded(\*\*) then you will need to left click on it and click remove, apply and close.
* **Emails –** The Emails that are currently sent out through are system tend to go straight to the junk email box. We believe it’s an Outlook issue because the email that is sending it out is not an official account and multiple uses of our program will result in emails almost always going there. So, if you expect an email always check the junk folder as well.
* **SQL –** Users must ensure that they have SQL Workbench set up properly with a database that can be used in conjunction with our program. A simple database will work since all that is needed is to run the program so the schema can be loaded into the database and begin to populate the fields.
* **Application.Properties –** When a fresh installation of our program is done one issue that may cause an error is the application.properties file is not configured correctly. When a fresh installation is done you must ensure in the application.properties file that you confirm that the password for your database is set up correctly because if not data will not be loading into and saved.

**7. Documentation**

This section will overview the documentation for the Work Orders System Application

**7.1 Method Documentation**

All important function methods in the Work Orders System have been completely documented and put together to where anybody should be able to look at our code and understand it.

**7.2 Class Documentation**

All classes incorporated into the Enterprise Routing System are documented on the class level as well as the functional level. This ensures that any programmer can pick up any part of the system with an insight into what is happening.

**7.3 External Documentation**

Our Work Orders System Application comes with a lot of documentation in the form of manuals. Notably in this documentation you will be able to find a ***User Manual*** which explains the basic functionality of each user type and what they can do within the program, the ***Technical Manual,*** a vital resource for anyone seeking to program with this application. It contains information relevant to programmers for our application on the various classes and methods. The other documents cover more niche topics like the ***Testing Manual*** and others.

**7.4 Javadoc Documentation**

The last relevant form of documentation currently implemented that is important for our project is Javadoc’s. These are stored within the ***Javadoc*** folder by opening the *index.html* page.

**8. Code Reusability**

In our code, we utilized a similar reusable domain structure for our buildings, departments, and rooms that was adjusted and tweaked according to each different location type. We also utilized a lot of common UI structures across our program with our group’s own implementations of common features such as logging out, viewing your profile, the user email of the currently logged-in user, and back buttons to properly traverse the program. Since we didn’t have an initial basis to start from, we did our best to make almost all our code replicable and easily usable in different potential areas of our code. We tried to make our code implementations as flexible as possible.

**9. Testing**

# **9.1 Human-Computer Testing**

Human-Computer Testing was conducted throughout the continued development of the Work Order System Program. This was done by the contributors to the project during all of production. These tests were to ensure things were working as they should. Most of our early testing occurred with direct user input, and trying to do different things in our systems before test-cases were introduced into this project.

# **9.2 Whitebox Testing**

Whitebox Testing was performed by those working on this project. Whitebox testing was performed by testers who had an inside knowledge of the system. Whitebox testing was undergone specifically by attempting to gain access to places where access should have been denied without being able to login. This was done by understanding the file structure of the System as well as our old html. Whitebox testing was implemented to ensure that users could not access pages of the application without being logged in. Whitebox Testing the application caused errors to be discovered causing changes to be made to the WebSecurityConfig Class to ensure that users could only access their appropriate web-content and weren’t able to see content outside of their scope.

# **9.3 Unit Testing**

Since we were creating a fresh project, we had no previous code to build off with our code. Because of this, we had to create new unique tests for all our implementations that we could, and to complete a huge chunk of this, we utilized Unit Testing. Most of our unit tests were to ensure that our functions are running as intended and that our various domains, repositories, and web-based tests.

*Tests Written:*

* *BuildingsTest*
* *DepartmentsTest*
* *IncidentTest*
* *RoleTest*
* *RoomsTest*
* *UsersTest*
* *BuildingRepoTest*
* *DepartmentRepoTest*
* *IncidentRepoTest*
* *RoleRepoTest*
* *RoomRepoTest*
* *UsersRepoTest*
* *BuildingsServiceTest*
* *TestingWebAppTest*
* *WebTests*

**10. Post-Mortem Analysis**

The project as we worked on and created it from scratch slowly began to unravel and show patterns and the structure that would make up the core of our program. We needed a strong understanding of Thymeleaf, JavaScript, the Spring Boot Application Framework, and its sub-framework Spring Security. Our program had a lot of small hurdles we slowly had to find our way through that could have potentially been made easier had we had known about patterns earlier on. Notably the Singleton and Decorator patterns. In terms of the systems UI, it easily could have been improved through the combined usage of various CSS frameworks like Bootstrap, jQuery, and general CSS if we had more time that we felt we could have put forward towards this part of the project. An issue we often had with this project was getting hung-on on issues like database troubles, excel file issues, and getting our login to be properly working. Stronger communication and organization between the group members would also have increased productivity toward the goal of the project. A stronger understanding of MySQL and databases, in general, would have made the creation and connections of tables easier.

**11. References**

“Conditionals in Thymeleaf.” *Baeldung*, 5 Aug. 2022, <https://www.baeldung.com/spring-thymeleaf-conditionals>.

“CSS Dropdowns.” *CSS Dropdowns*, <https://www.w3schools.com/css/css_dropdowns.asp>.

“Getting Started: Testing the Web Layer.” *Getting Started | Testing the Web Layer*, <https://spring.io/guides/gs/testing-web/>.

“How to Send Email in Java Using GMAIL SMTP?: Pepipost.” *How To Send Email In Java Using Gmail SMTP? | Pepipost*, <https://netcorecloud.com/tutorials/send-email-in-java-using-gmail-smtp/>.

“How to Work with Dates in Thymeleaf.” *Baeldung*, 29 May 2022, <https://www.baeldung.com/dates-in-thymeleaf>.

“The Import Org.springframework.security Cannot Be Resolved.” *Stack Overflow*, 1 Sept. 2019, <https://stackoverflow.com/questions/59337122/the-import-org-springframework-security-cannot-be-resolved>.

“Input Type=‘Date’ Thymeleaf.” *Stack Overflow*, 1 May 2016, <https://stackoverflow.com/questions/38693971/input-type-date-thymeleaf>.

“Java - Sending Email.” *Tutorials Point*, <https://www.tutorialspoint.com/java/java_sending_email.htm>.

“MySQL Workbench - Connect to a Localhost.” *Stack Overflow*, 1 Sept. 2013, <https://stackoverflow.com/questions/20431491/mysql-workbench-connect-to-a-localhost>.

Paraschiv, Eugen. “Retrieve User Information in Spring Security.” *Baeldung*, 8 Oct. 2021, <https://www.baeldung.com/get-user-in-spring-security>.

“Resolving Failed to Configure a DataSource Error.” *Baeldung*, 19 Apr. 2023, <https://www.baeldung.com/spring-boot-failed-to-configure-data-source>.

Samper, José Miguel. *Thymeleaf*, <https://www.thymeleaf.org/doc/articles/springsecurity.html>.

“Spring Blog.” *Spring Security without the WebSecurityConfigurerAdapter*, <https://spring.io/blog/2022/02/21/spring-security-without-the-websecurityconfigureradapter>.

*Spring Boot Crud Example with Spring MVC – Spring Data JPA – Thymeleaf - Hibernate - Mysql*, <https://www.codejava.net/frameworks/spring-boot/spring-boot-crud-example-with-spring-mvc-spring-data-jpa-thymeleaf-hibernate-mysql>.

“Spring Boot Starter Web - Javatpoint.” *Www.javatpoint.com*, <https://www.javatpoint.com/spring-boot-starter-web>.

“Spring Security Gets the Currently Logged in User.” *Spring Cloud*, 10 Feb. 2022, <https://www.springcloud.io/post/2022-02/spring-security-get-current-user/#gsc.tab=0>.

“Spring Security with Thymeleaf.” *Baeldung*, 29 Oct. 2022, <https://www.baeldung.com/spring-security-thymeleaf>.

“Thymeleaf Tutorial #1 - Thymeleaf Template Basics.” *YouTube*, YouTube, 21 June 2020, <https://www.youtube.com/watch?v=Gbam0vw6l4k>.

“Working with Microsoft Excel in Java.” *Baeldung*, 2 Dec. 2022, <https://www.baeldung.com/java-microsoft-excel>.

“Check If the Value Exists in Array in Javascript - Javatpoint.” *Www.javatpoint.com*, <https://www.javatpoint.com/check-if-the-value-exists-in-array-in-javascript#:~:text=The%20indexof()%20method%20in,found%20and%20returns%20%2D1%20otherwise>.

“How Can I Remove a Specific Item from an Array in JavaScript?” *Stack Overflow*, 1 Jan. 1958, <https://stackoverflow.com/questions/5767325/how-can-i-remove-a-specific-item-from-an-array-in-javascript>.

“How to - Cascading Dropdown List.” *How To Create a Cascading Dropdown*, <https://www.w3schools.com/howto/howto_js_cascading_dropdown.asp>.

*JavaScript Arrays*, <https://www.w3schools.com/js/js_arrays.asp>.

“Joining Two Table Entities in Spring Data JPA.” *Stack Overflow*, 1 Aug. 1960, <https://stackoverflow.com/questions/19977130/joining-two-table-entities-in-spring-data-jpa>.

Paraschiv, Written by: Eugen. “Redirect to Different Pages after Login with Spring Security.” *Baeldung*, 14 Oct. 2022, <https://www.baeldung.com/spring-redirect-after-login>.

Prasad, Ananta. “Uncaught Typeerror: Data.push Is Not a Function.” *Stack Overflow*, 1 Jan. 1962, <https://stackoverflow.com/questions/29696572/uncaught-typeerror-data-push-is-not-a-function>.

“Send Javascript Variables to Spring Controller.” *Stack Overflow*, 1 May 1964, <https://stackoverflow.com/questions/45791117/send-javascript-variables-to-spring-controller>.

“Spring Security, How to Restrict User Access Certain Resources Based on Dynamic Roles?” *Stack Overflow*, 1 Oct. 1960, <https://stackoverflow.com/questions/21128806/spring-security-how-to-restrict-user-access-certain-resources-based-on-dynamic>.

Trandafir, Written by: Emanuel. “Joining Tables with Spring Data JPA Specifications.” *Baeldung*, 18 Mar. 2023, <https://www.baeldung.com/spring-jpa-joining-tables>.

Freilino, Zachary. *Web Based Evaluations*. 5 Dec. 2022.

Gearhart, Joshua, and Nicolas Bushée. *Web Routing*. Dec. 2022.