

**COIT20258: Software Engineering**

Assignment 2

Individual Assignment

Student Name: Ahmed Shahriar Asif

Student Id: 12248535

Melbourne Campus

Submission Date: 13th September 2024

Table of Contents

[1. UML Diagram 2](#_Toc177164019)

[2. Test Plan, Test Data and Results: 2](#_Toc177164020)

[2.1 Application Username and Passwords 2](#_Toc177164021)

[3. Key Challenges and Learnings 4](#_Toc177164022)

[4. Project Results: 5](#_Toc177164023)

[5. Reflection: 6](#_Toc177164024)

[6. Future Works: 6](#_Toc177164025)

[7. Appendix 7](#_Toc177164026)

# UML Diagram

The green button in the UML diagram indicates that an attribute or method is public, which means that any other class can access it. The red button, on the other hand, stands for a private method or attribute, which is restricted and available only within the class in which it is declared.  
  
It is easier to comprehend the system's architecture and the interactions between its many parts because to this visual differentiation that clarifies the visibility of class members. Private elements are kept inside for encapsulation and restricted access, whereas public elements are intended for outward interaction.

A screenshot of a computer

Description automatically generated

Fig 01: UML Diagram

# Test Plan, Test Data and Results:

## Application Username and Passwords

In this application have set three default username and passwords.  
  
**For Visitor login**

**Username:** visitor

**Password:** visitor

**For Resource Manager Login**:

**Username:** manager

**Password:** manager

**For Emergency Responder:**

**Username:** responder

**Password**: responder

In this application if new users want to login, then they need to signup first or they can use default **visitor** user name and password.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Name | Test Plan | Test Data | Expected Outputs | Results |
| Login | Only click on login button | N/A | Error , can not be empty | passed |
| Login | Wrong inputs | Dsds, dsds | Invalid username or password | passed |
| Public user can login using Signup button | Input signup details | Asif, [asif@gmail.com](mailto:asif@gmail.com), asif, male, 10/10/1997 | Signup successful and directly take me in the dashboard | passed |
| Visitor login | Used visior username and password | visitor,  visitor | Login successful | passed |
| Resource manager login | Used manager username and password | manager,  manager | Login successful | passed |
| Emergency responder login | Used responder username and password | responder,responder | Login successful | passed |
| All tabs load data from database | Check all tabs | N/A | Loads all data from database | passed |
| Add disaster reports as a visitor | Login as visitor | Fire,Melbourne,13/09/2024, fire started | Should add in the disaster report table and also update in the live update tab | passed |
| Visitor or public user restrictions | Can not see add alert and add resource button | Fire,Melbourne,13/09/2024, fire started | They can only add disasters report, can’t see alert and resource buttons | passed |
| Resource manager can report disaster and update resources | Can add disaster report and update resources | Fire,Melbourne,13/09/2024, fire started.  pipe, 20, 50, 13/09/2024 | Shoul add in the disaster reports on the disaster table and also can update resources | passed |
| Emergency responder can add disaster report and can submit alerts | Can add disaster report and can add alerts | Fire,Melbourne,13/09/2024, fire started.  Flood, 2, high, 13/09/2024, flood happened | Should add in the disaster reports on the disaster table and also can add alerts | passed |
| Resource manager restrictions | Can not see “Add alert” button | N/A | Can not see alert button | passed |
| Emergency Responder Restrictions | Can not see “Add resource” button |  | Can not see “Add resource” button | passed |
| Database saved inputs and displayed in the live updates as well as their own tables | Check inputs | N/A | Should display all inputs | passed |
| Update profile | Change signup details | [asif@gmail.com](mailto:asif@gmail.com), password, male, 10/10/1997 | Should update details and display successful notification | passed |

# Key Challenges and Learnings

The process of creating the Disaster Response System (DRS) was rewarding because it called for the fusion of software engineering concepts with practical methods to problem-solving. For this project, a disaster response software prototype was constructed by capturing user stories from an extensive crisis management scenario and utilizing the Model-View-Controller (MVC) design to transform them into a working JavaFX application.

1. Comprehending User Requirements: During the first stage, I spent my time examining and improving the case study's user stories. The disaster response system (DRS) has to coordinate responses from several departments, including fire and emergency, hospitals, and law enforcement, while managing a variety of disaster types, including hurricanes, fires, and earthquakes. It became clear how complicated the system was when interfaces for many departments and stakeholders had to be designed. To guarantee the system's resilience and expandability, a meticulous evaluation of both functional and non-functional prerequisites was necessary.
2. Design and Necessities Engineering: I created a revised requirements specification at this step by utilizing the use cases and functional requirements. Key classes like User, Resource, Disaster, and DisasterAlert, as well as controllers like DashboardController, ResourceController, and DisasterAlertController, were identified during this process.  
     
   I was able to comprehend the interactions between classes and the data structures involved in handling catastrophe reports, resources, and warnings on a deeper level by translating the use cases into UML class diagrams. To illustrate the connections between things such as users and the reports they submit, cardinality was defined. The system's data flow was guaranteed to be coherent throughout this design phase, which also set the groundwork for the implementation phase.
3. MVC Architecture and GUI Design: Because the JavaFX framework is structured, the shift from design to the Model-View-Controller (MVC) pattern was seamless. I created the graphical user interface (GUI) using JavaFX Scene Builder. The graphical user interface (GUI) comprised many displays, including the dashboard, login page, and forms for reporting disasters. These screens offered a simple and easy-to-use interface for people to engage with the system.
4. **Creative Features:** In addition to the needs listed, I added **two creative features** to the project:  
     
   **Live Updates** **for catastrophe Alerts:** Using a LiveUpdate class, a new feature was implemented that offers real-time updates for catastrophe alerts. Users can view real-time updates in a dedicated dashboard section if new disasters or resources are reported. This function improves the responsiveness of the system and notifies users as soon as important situations arise.

**Functionality for User** **Profile Updates:** I put in place a function that allows users to edit their password, email address, and personal information. The UpdateprofileController now has this feature, enabling users to effectively and safely manage their profiles.

1. **Database Integration:** To imitate the database interactions, even though this was a prototype, I used an in-memory H2 database. This gave me the opportunity to put procedures in place for managing user data, resources, and disaster reports in real-time. As an illustration, DatabaseConnector oversaw the management of the database connection and performed tasks like adding new resources and retrieving catastrophe reports from the disaster reports and resources tables.

# Project Results:

The DRS prototype effectively illustrated how JavaFX and MVC architecture can be used to prototype a complicated disaster management system. The following results were attained:  
  
**Resource Management and Disaster Reporting:** The system's well-organized and user-friendly interface enables users to manage resources and report disasters.

**Prioritized Response:** By analyzing the kind and intensity of catastrophe alerts, the system is able to determine and oversee disaster priorities, guaranteeing prompt replies.

**Departmental Coordination:** Despite being a prototype, the architecture is in place to enable departmental coordination, and it includes for Emergency responses and Resource managements.

# Reflection:

When I think back on the project, I learned a lot in a few different areas:  
  
**Design Thinking:** Constructing a catastrophe response system necessitated carefully considering user requirements, scalability, and real-world circumstances. My comprehension of the requirements engineering process and the significance of turning user stories into specific functional and non-functional requirements has improved as a result of this experience.

**MVC and Modularity:** The MVC design pattern offered definite advantages in terms of modularity and system maintainability. It was simpler to extend and test the system because system logic, user interface, and event processing were separated. I can see how using this design approach will pay out in the long run, particularly for larger systems.

**Creative Thinking:** By adding functions like user profile management and live disaster updates, the system became more functional and user-focused. This encounter made clear how crucial it is to consider alternatives in order to design a more responsive and interesting user experience.

# Future Works:

In the future, more improvements might consist of:  
  
**Integration with External Systems:** The DRS prototype would be more akin to a deployment situation in the real world if it were connected to genuine fire departments, hospitals, and other disaster management organizations.  
**Mobile and Web Versions:** The DRS would be more accessible if the desktop-based JavaFX program were expanded to mobile and web platforms, particularly in emergency situations where users might not have access to a PC.

# Appendix

Actual Outputs

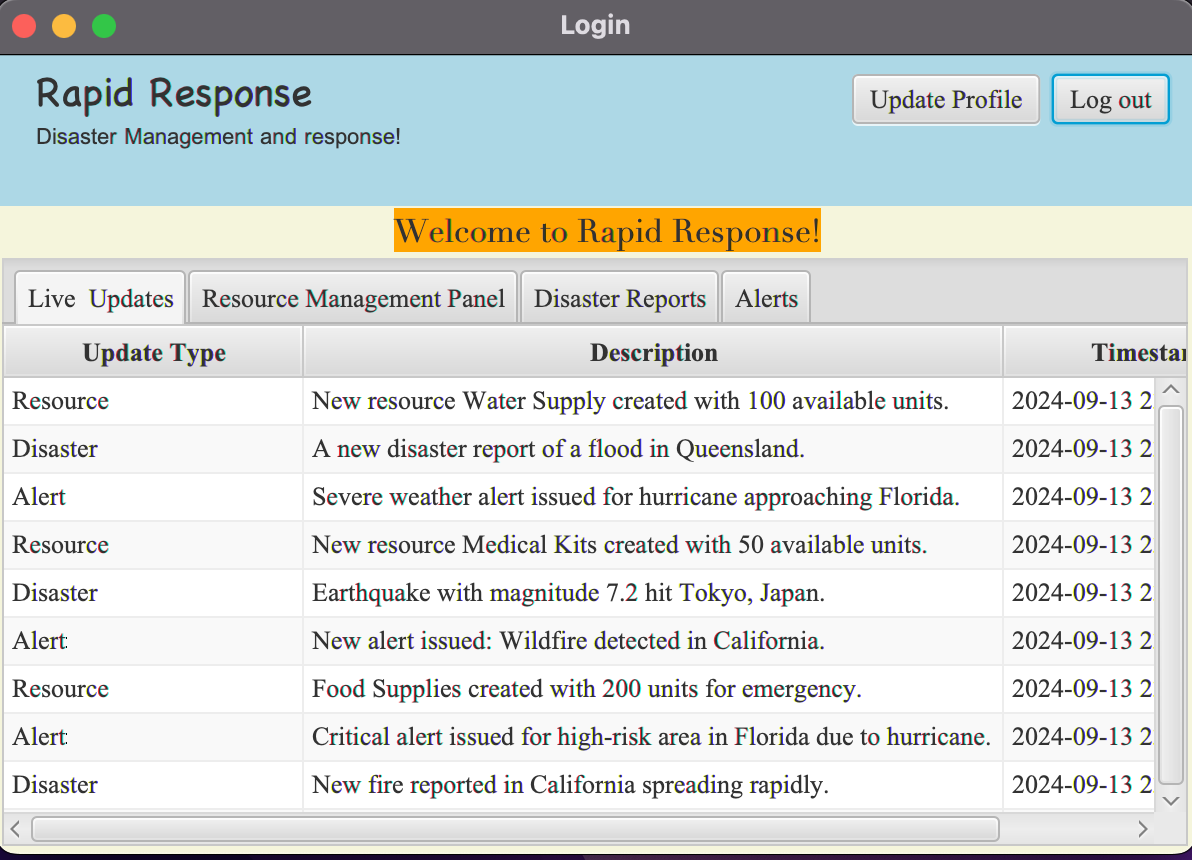


Fig 02: Live Update tab

A screenshot of a computer

Description automatically generated

Fig 03: Disaster Report Tabs

A screenshot of a login form

Description automatically generated

Fig 04: update profile page

A screenshot of a login form

Description automatically generated

Fig 05: log in and signup page.