**A logo with text on it

Description automatically generated**

**COIT20258: Software Engineering**

**Assignment 2**

**Disaster Response System Prototype**

**Submitted by:**

Name : Sagar Bhujel

Student ID : 12256739

Email : sagar.bhujel@cqumail.com

Date : 04th August 2024

**Tutor**

Name: Meiru Che

Email: m.che@cqu.edu.au

# Description of project and Test conducted.

The project demonstrates the early prototype of Disaster Response System. The system developed has the following functionalities:

1. Disaster Reporting
2. Registration of new Users based on the roles (additional feature)
3. User login based on the role (additional feature)
4. View the disaster reported along with the status update
5. Resource management as per the disaster reported
6. Response coordination related to the reported disaster
7. Communication hub is still underdevelopment
8. A live database for storing and fetching purpose.

This report reflects the class diagram utilized for the development of the system as well all the test conducted for the system.

# Program Compilation Description

## Prerequisites to compile the program

The system is built on NetBeans 20 using **java with ant** with the following environment

1. JavaFX SDK 22 library
2. MySQL-connector-j-9.0.0
3. MySQL workbench 8.0 CE

## Instruction

The main class for the project is located in the Model package named DisasterResponseSystem.java.

The database connection is maintained by the two java files namely

1. Interface IDatabaseManager.java
2. DisasaterDatabaseConn.java (implements the interface)

Both the files are present in the Persistence package.

The database created is called **disasterresponse.**

The project follows the MVC model such that the .fxml files are stored in the View packages, Model package holds the various models used in the project and the corresponding controller for each .fxml files are present in the Controller Package.

The Model package has the following java classes:

1. Agency.java
2. AgencyAssignment.java
3. Alert.java (under development)
4. DisasterReport.java
5. DisasterResponseSystem.java
6. Message.java (under development)
7. Notification.java (under development)
8. Resource.java
9. StrategicDecision.java (Under development)
10. User.java

The View package has the following .fxml files:

1. CommunicationHubView.fxml
2. DisasterReportingView.fxml
3. Login.fxml
4. MainMenu.fxml
5. Registration.fxml
6. ResourceManagementView.fxml
7. ResponseCoordinationView.fxml
8. ResponseMenu.fxml

The Controller package has the controller for respective .fxml files

1. CommunicationHubController.java
2. DisasterReportingController.java
3. LoginController.java
4. MainMenuController.java
5. RegistrationController.java
6. ResourceManagementController.java
7. ResponseCoordinationController.java
8. ResponseMenuController.java

Most the database content are input from the GUI however, new agency additions do not have a GUI rather hard programmed in the DisasterDatabaseConn which in turn creates an Agency database with default values such as Fire Department, Police, Medical Services, Red Cross.

# The class diagram

The image below gives the comprehensive overview of the class diagram

The class Diagram.

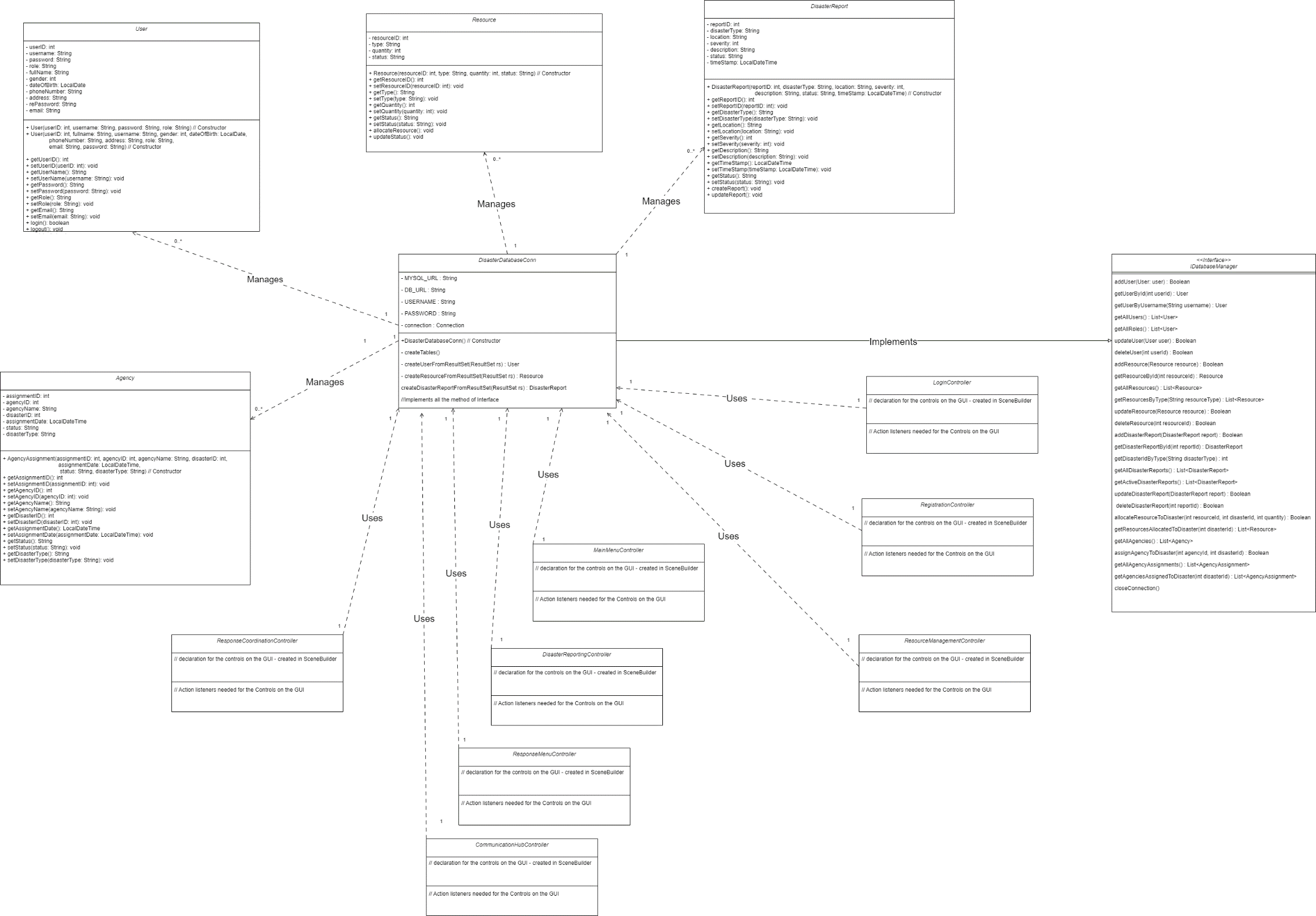
The class DisasterDatabaseConn ---implements--- > interface IDatabaseManage

DisasterDatabaseConn 1----------Manages----------0..\*> all the model class

The model classes have an indirect relationship with the Controller classes however, the controller classes have relation with the DisasterDatabaseConn as all the objects of controller classes are manages by it.

Therefore,

All controller classes 1---------------Uses----------------1 > DisasterDatabaseConn



# Testing

## Execution of the system

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input Validate** | | **Test Cases** | **Test Input** | **Expected Output** | **Pass/Fail** |
| Not applicable | | Starting project | Execute button on neatbeans | MainMenu.fxml deployed | Pass |
| Actual Output | |  | | | |
| Not applicable | | Login button action | Click the login button | Login.fxml deployed | Pass |
| Actual output | |  | | | |
| Not applicable | | Registration button action | Click the Registration button | Registration.fxml deployed | Pass |
| Actual output | |  | | | |
| Not applicable | | Report Disaster button action | Click the Report Disaster button | DisasterReporingView.fxml deployed | Pass |
| Actual output | |  | | | |
| Not applicable | View Disaster button action | Click the View Disaster button | DisasterReportingView.fxml deployed | Pass |
| Actual output |  | | | |

## Registration of users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Validate** | **Test Cases** | **Test Input** | **Expected Output** | **Pass/Fail** |
| Date of birth validation | Error pop up for wrong date format (yyyy-mm-dd) | dd-mm-yyyy wrong date format | Error pop up | Pass |
| Actual Output |  | | | |
| Password validation | Match password | Password and retype password should match | Error pop up for password mismatch | Pass |
| Actual output |  | | | |
| Blank input | | Blank input fields error | Address and username are blank | Error pop up for cannot have blank input fields | Pass |
| Actual output | |  | | | |
| Successful Registration | | Successful creation of user data in the database | Correct data added | Successful message pop up | Pass |
| Actual output | |  | | | |

## Reporting of disaster

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Validate** | **Test Cases** | **Test Input** | **Expected Output** | **Pass/Fail** |
| Not applicable | Report disaster | Add all the data in the GUI | Display Successfully created pop up | Pass |
| Actual Output |  | | | |
| Not applicable | The reported disaster Viewing | Click the View Disaster Tab | The reported disaster will be in the tab | Pass |
| Actual output |  | | | |
| Not applicable | View Details on the disaster | Click the View Detail Button | Pop up with disaster detail | Pass |
| Actual output |  | | | |

## Login functionality

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Validate** | **Test Cases** | **Test Input** | **Expected Output** | **Pass/Fail** |
| Select role blank | Login without select role | Username and password without role | Error pop up | pass |
| Actual Output |  | | | |
| Wrong password | Case if wrong password entered | Wrong password entered | Error pop up | pass |
| Actual Output |  | | | |
| NA | Successful login | Correct data entered | Display the ResponseMenu.fxml | pass |
| Actual Output |  | | | |
| NA | Successful logout button | Click logout button | Logout successful pop up with login.fxml display | pass |
| Actual Output |  | | | |

## Resource Menu button

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Validate** | **Test Cases** | **Test Input** | **Expected Output** | **Pass/Fail** |
| NA | Adding resource | Select resource type = Clothing  Quantity = 50  Click add resource | Resource added visible in table view | Pass |
| Actual Output |  | | | |
| NA | Locate resource | Highlight the resource  Select the disaster from dropdown  Click allocate. | Message on testArea resource allocated | Pass |
| Actual Output |  | | | |
| NA | Update the resource | Highlight the resource  Click the update resource status button  Choose available, in use or depleted | The table showing status of resource gets updates. | Pass |
| Actual Output |  | | | |

## Response Menu Button

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Validate** | **Test Cases** | **Test Input** | **Expected Output** | **Pass/Fail** |
| NA | Adding agency to disaster | Go to Agency Coordination Tab  Select Agency from drop down  Select Disaster from drop down  Click Assign to disaster button | The data enters are displayed on the table View | pass |
| Actual Output |  | | | |

# Functionality Under Development

The Update Resource Effort and strategic Decision in response Coordination are still under development.

The whole of communication Hub module is under development.

# The Database created screenshot

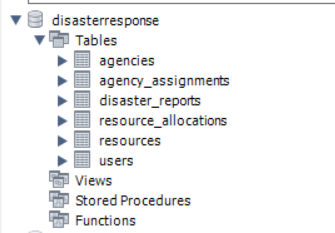


Figure 1 disasterresponse database schema

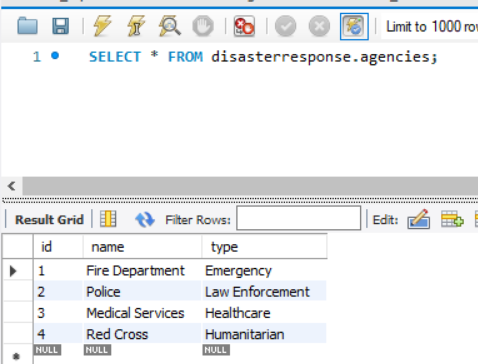


Figure 2 Agency table

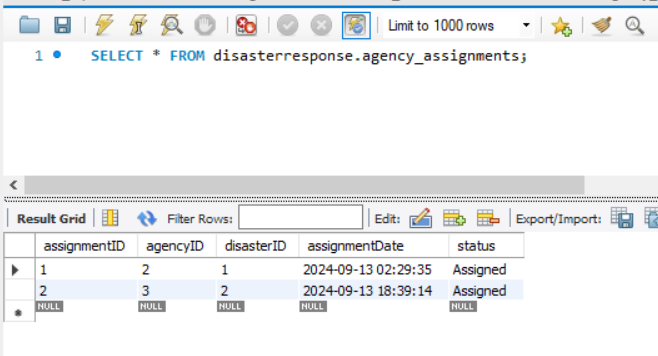


Figure 3 AgencyAssignment table

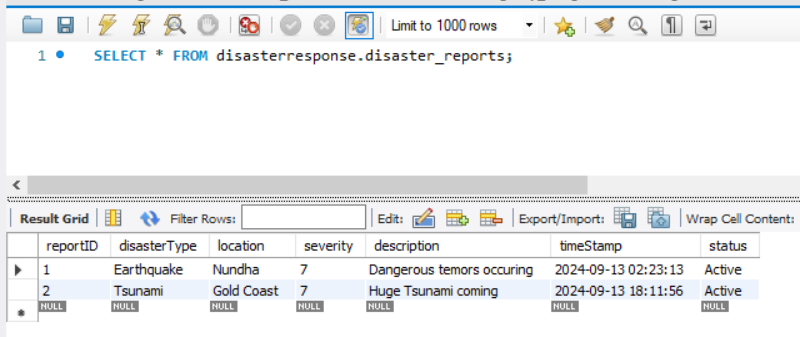


Figure 4 Report Disaster table

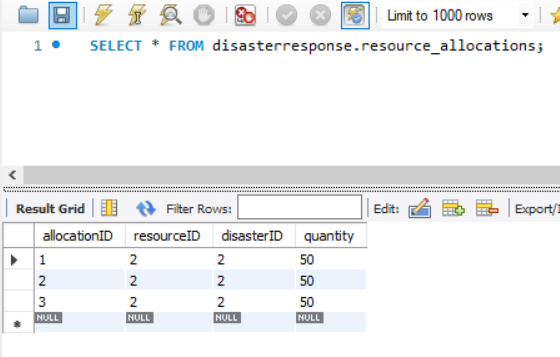


Figure 5 Resource allocation table

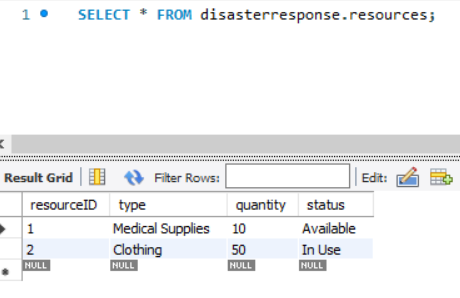


Figure 6 Resource table

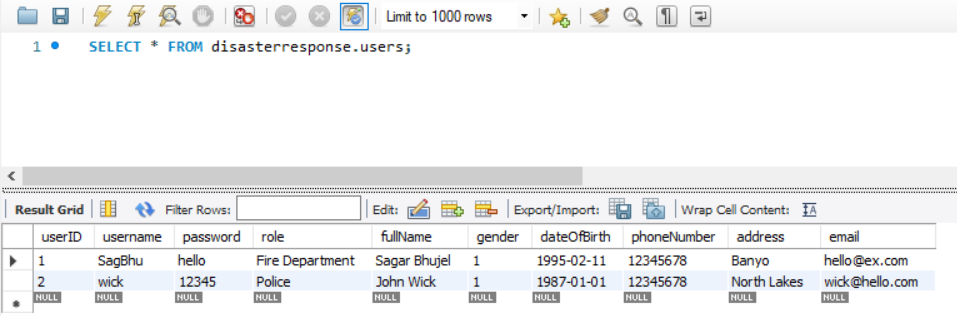


Figure 7 User Table

# Reflection on the overall

# Lesson learned

To have the smooth progression of the development it is important to fully study the user expectation and draft a concrete specification. That is because when the actual development occurs, we need to check with the specification as it gives the clear picture of the product.

# User Interface design

In the GUI development we should always be mindful that the GUI does not complicate things. The phrase, “ the system is there to ease user experience”, should be always at the back of our mind while designing the interface.

# Challenges.

The most challenging part of the project is the designing of the overall project especially the class diagram. That is because the initial requirement specification needed to be altered to accommodate the changes while development progressed.

Additionally, implementation of database made it difficult as we need to related all the DBMS relationship with each other to effectively produce the product.

# Conclusion

Overall, the project provides me with good insight into the process of software development. Starting from requirement gathering/specification to final product. The implementation of each and every phase of software development were captured during the process and it build up my confidence in approaching more software development in future.