SOFTWARE DESIGN DOCUMENT

for

Risk Tracking System

Release 1.0

Version 1.0

Prepared by SRT View/Model

Contents

List of Figures

List of Tables

Revision History

Date	Description	Revised by
04/12/22	Baseline	View/Model

1 INTRODUCTION

This document presents a collective effort to define the architectural blueprint and design nuances of the Software Risk Tracking System (SRTS), crafted to serve as an instrumental resource in identifying, monitoring, and mitigating software development risks. Designed with input from a diverse team of project managers, developers, and stakeholders, this document is structured to offer a clear, comprehensive guide to the system's architectural integrity, design decisions, and the strategic management of potential software risks.

1.1 Purpose

The core aim of this Software Design Document (SDD) is to furnish a detailed architectural and design framework for the Software Risk Tracking System. As a product of collaborative insights, this document is intended for the collective use of the development team, project stakeholders, and future system maintainers, providing a unified blueprint for the system's construction, implementation, and sustained management, ensuring a cohesive understanding and application of the system's functionalities across all team members.

1.2 Scope

The Software Risk Tracking System is devised as a collaborative tool to assist teams in effectively identifying, and cataloging risks throughout the software development life cycle. By facilitating a unified platform for comprehensive risk documentation, evaluation, and prioritization, the system is designed to preemptively address potential disruptions to project schedules, budget adherence, and quality standards, thereby bolstering overall project management and software delivery success.

1.3 Overview

Organized systematically, this document begins with an introduction to the system's objectives, scope, and collaborative design methodology. It proceeds to detail the system architecture, encompassing architectural and data design, and interface specifications. The document concludes with insights on implementation strategies, quality considerations, and appendices for supplementary resources and terminology clarifications.

1.4 References

IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.

ISO/IEC 12207:2008, Systems and software engineering – Software life cycle processes.

1.5 Definitions and Acronyms

Definitions:

Risk Tracking: A collaborative process of identifying, recording, evaluating, and monitoring potential project risks, along with the formulation of strategic actions to diminish their impact.

Stakeholder: Any team member, group, or entity that may influence, be impacted by, or perceive themselves to be impacted by a decision, activity, or outcome within a project.

Table 1.1: Acronyms

Acronym	Meaning
SRT	Software Risk Team
SDD	Software Design Document
SRTS	Software Risk Tracking System
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
UI	User Interface
API	Application Programming Interface
DBMS	Database Management System
SQL	Structured Query Language
HTTP	Hypertext Transfer Protocol
SaaS	Software as a Service
REST	Representational State Transfer

Mitigation Strategy: Collaboratively developed actions aimed at decreasing the likelihood, severity, or potential impact of identified risks on a project.

2 SYSTEM OVERVIEW

2.1 Functionality

Refer to section 5.

2.2 Context

The system serves software development organizations of all sizes and industries, including small businesses. It's designed for project managers, security managers, and stakeholders. Whether it's a small startup or a large enterprise, the system adapts to varying needs. The system makes sure your project info stays safe by using strong security methods.

2.3 Design

Refer to section 3.1.

3 SYSTEM ARCHITECTURE

3.1 Architectural Design

All the views have been broken down into their own class inheriting the DashboardView or the RiskTracker classes depending on where they are initially seen. The Risk and User classes contain data used throughout, and the DatabaseConnection is used to communicate to the database which is used for persistent storage and will also be used to get specific Risk and User data throughout the software. A report class is used to help organize the risk report and risk matrix.

Refer to figures 3.1-3.4 for the class diagram.

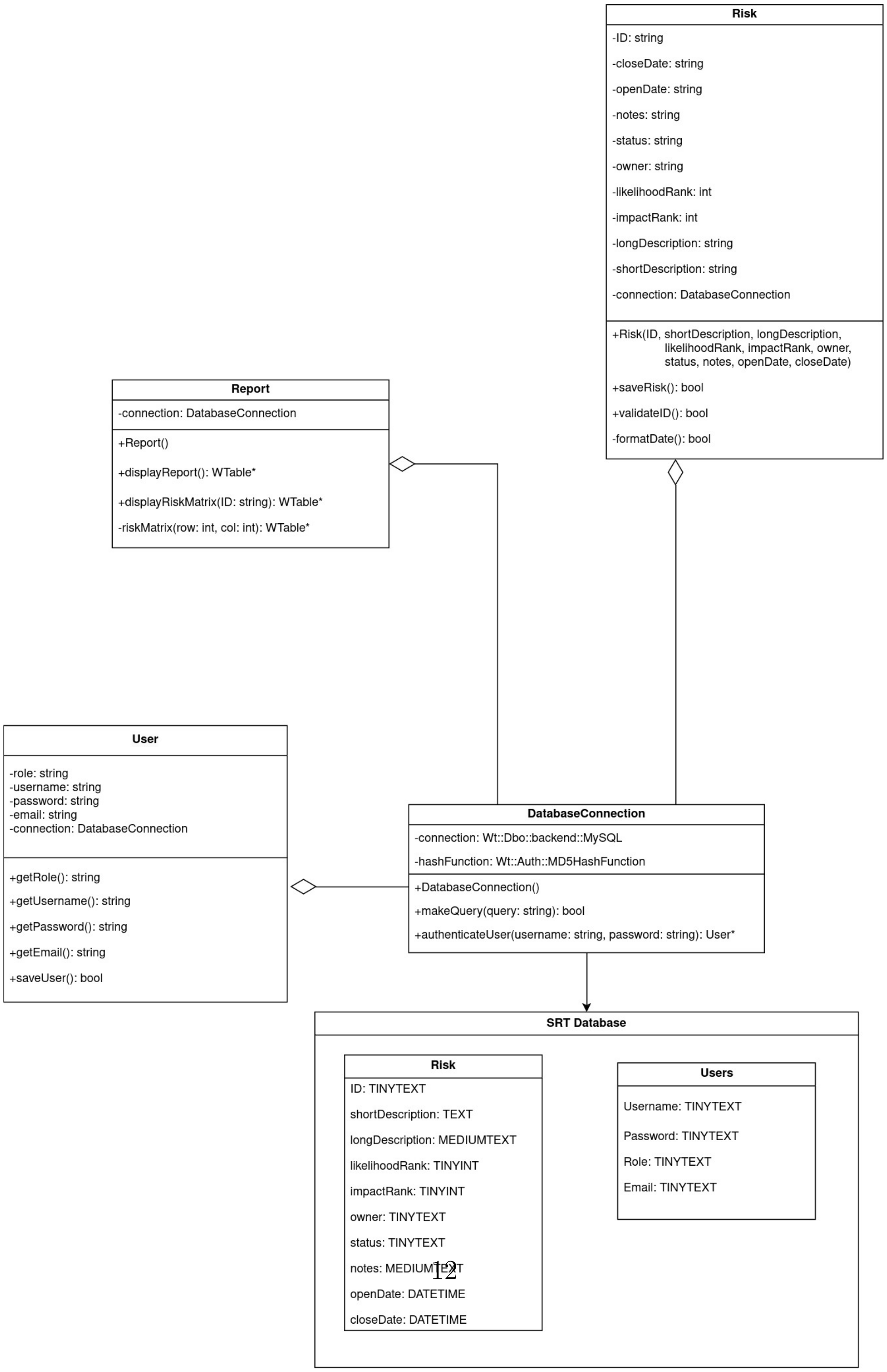


Figure 3.1: Database Connection UML

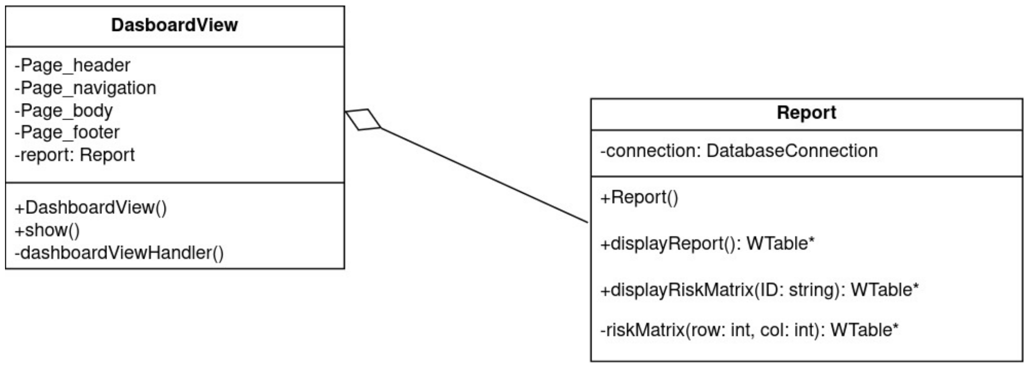


Figure 3.2: Report UML

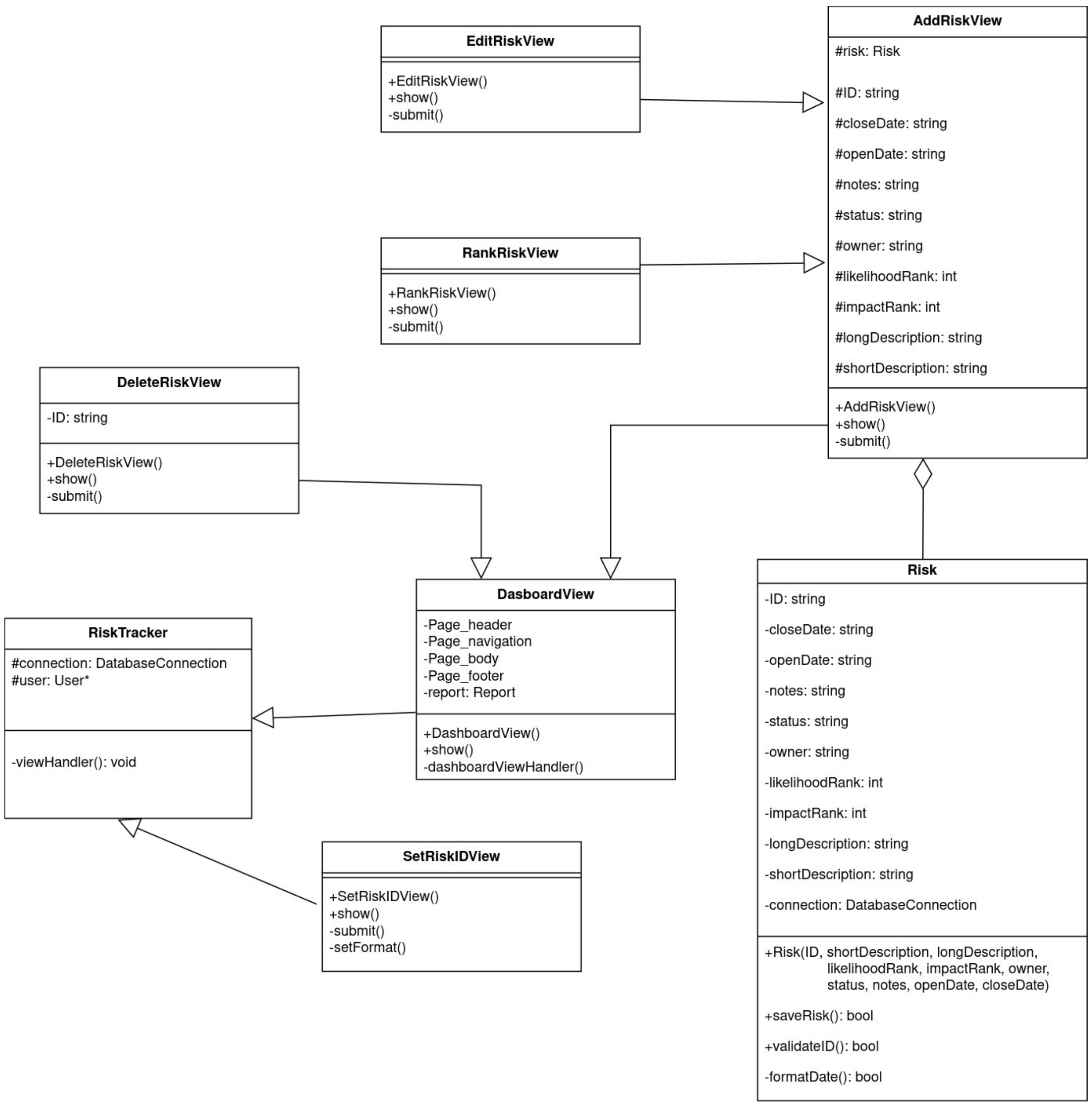


Figure 3.3: Risk UML

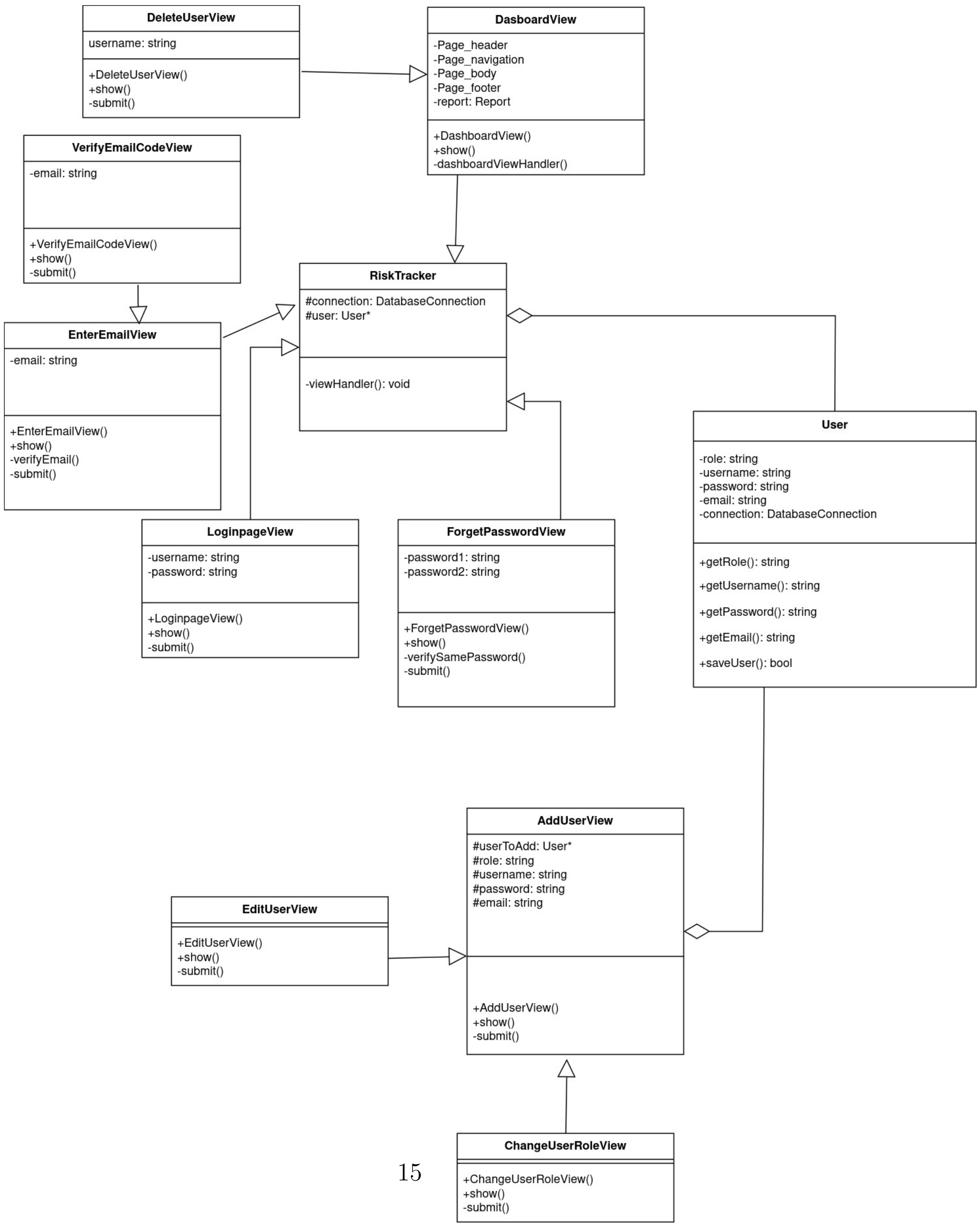


Figure 3.4: User UML

3.2 Decomposition Description

Refer to sections 3.1 and 5

3.3 Design Rationale

The software will be OO so a class diagram representing every class relationship was chosen. A database that stores Risk and User data was chosen as the persistent storage mechanism to easily query for the data throughout the program using the DatabaseConnection class. Some critical issues/trade-offs with these choices include increased system load, and unneeded overhead in the DatabaseConnection class used in some other classes. If a smaller system fails, it could cause the whole system to fail. MD5 was chosen as the hashing function due to its ease of use in a MySQL query, however, an issue with MD5 is a higher amount of collisions than other hashing methods such as bcrypt.

4 DATA DESIGN

4.1 Data Description

Refer to section 3.1

4.2 Data Dictionary

Refer to the tables below.

Table 4.1: Data Dictionary - Risk

Data	Attributes/Methods	Description
Risk Class	ID - string closeDate - string openDate - string notes - string status - string owner - string likelihoodRank - int impactRank - int longDescription - string shortDescription - string	Object that contains risk data
Risk Table (Database)	ID - TINYTEXT shortDescription - TEXT longDescription - MEDIUMTEXT likelihoodRank - TINYINT impactRank - TINYINT owner - TINYTEXT status - TINYTEXT notes - MEDIUM-TEXT openDate - DATE-TIME closeDate - DATE-TIME	Table in the SRT database that contains risk data

Table 4.2: Data Dictionary - User

User Class	data related attributes: role - string username - string password - string email - string getRole() - string getUsername() - string getPassword() - string getEmail() - string	Object that contains user data
User Table (Database)	Username - TINYTEXT Password - TINYTEXT Role - TINYTEXT Email - TINYTEXT	Table in the SRT database that contains risk data

5 COMPONENT DESIGN

All submit methods are attached to a button that is declared in the show methods for the corresponding class. For all show methods, refer to section 6.2 for the corresponding layout. All constructors initialize default values unless otherwise stated.

RiskTracker Class: Class that represents the website.

- viewHandler method: Method to handle all the transitions between sub-views for the main webpage.

DasboardView Class: Class for the layout and functionality of the dashboard view.

- dashboardViewHandler method: Method to handle all the transitions between sub-views for the dashboard.

SetRiskIDView Class: Class for the layout and functionality of the set risk ID view.

- submit method: The class will call the setFormat method and transition to the login screen.
- setFormat method: Create a starting risk with the risk ID format that will be used for all risk IDs to follow.

AddUserView Class: Class for the layout and functionality of the add user view.

- submit method: The method will query the database for the username to see if it exists, if it does display a "user already exists message", otherwise, set the input field text to the associated variable to create a User object and call the saveUser method in the User object to store the user in the database.

EditUserView Class: Class for the layout and functionality of the edit user view.

- submit method: The method will set the input field text to the associated variable to create a User object, the User selected will be deleted from the user table in the SRT database, the User object will call the saveUser method to save a new user in the database.

  The method adds the edited user to the database and removes the old user from the database.

ChangeUserRoleView Class: Class for the layout and functionality of the change user role view.

- submit method: Same as editUserView, but it only has the role input field.

DeleteUserView Class: Class for the layout and functionality of the delete user view.

- submit method: A username will be retrieved from an input field, the method will show a pop up window asking if the admin is sure it wants to delete the user, if no the method will return, if yes, the record for the user will be removed from the SRT database.

AddRiskView Class: Class for the layout and functionality of the add risk view.

- submit method: The method will query the database for the risk ID to see if it exists, if it does display a "risk already exists message", otherwise, set the input field text to the associated variable to create a Risk object and call the saveRisk method in the Risk object to store the risk in the database.

EditRiskView Class: Class for the layout and functionality of the edit risk view.

- submit method: The method will set the input field text to the associated variable to create a Risk object, the Risk selected will be deleted from the user table in the SRT database, the Risk object will call the saveRisk method to save a new Risk in the database.

  The method adds the edited Risk to the database and removes the old Risk from the database.

RankRiskView Class: Class for the layout and functionality of the rank risk view.

- submit method: Same as editRiskView, but it only has the likelihood and impact rank input fields.

DeleteRiskView Class: Class for the layout and functionality of the delete risk view.

- submit method: A Risk ID will be retrieved from an input field, the method will show a pop up window asking if the user is sure it wants to delete the risk, if no the method will return, if yes, the record for the risk will be removed from the SRT database.

LoginpageView Class: Class for the layout and functionality of the login page view.

- submit method: This method retrieve a username and password from an input field and pass that into a the DatabaseConnection (connection variable) authenticate_user method. If the method returns nullptr, display a "user cannot be found" message, otherwise, the User will be stored in user and the screen will be cleared.

EnterEmailView Class: Class for the layout and functionality of the enter email view.

- submit method: The method will get the email from an input field and call the verifyEmail method to ensure the email is in the database. If the method returns false, display a "incorrect email" message, otherwise clear the screen.

- verifyEmail method: The method will query the database to see if the email field exists, based on the email variable in the class. If it does not exist, return false, otherwise return true.

VerifyEmailCodeView Class: The view for a class to verify the code sent to a user's email.

- submit method: This method will send the query to the database to check whether the entered random token matches with token send to the user's email.

ForgetPasswordView Class: Class for the layout and functionality of the forget password view.

- submit method: Store input from the password and repeated password field into password1 and password2, if the verifySamePassword method returns true, clear the screen, otherwise display a ”passwords do not match” message.
- verifySamePassword method: If password1 and password2 variables are the same, return true, otherwise return false.

Report Class: Class for organizing reports that are shown to the user.

- displayRiskMatrix method: The method will query for a risk based on its ID, if it does not exists it will return nullptr. If it does exists, the method will query for the ranks of the risk based on the ID of the risk, and it will return the riskMatrix method to get the base risk matrix, it will pass in the ranks as the row and col to show the highlighted risk level.
- displayReport method: The method querys the database for all risk table records, if none exist the method will return nullptr. If it does exists, the text will be formatted as a 2D array, and will be put into a WTable, the address of the table will be returned.
- riskMatrix method: View for the risk matrix, refer to section 6.2.

Risk Class: Class for creating and saving risks.

- saveRisk method: The method first uses the validateID method to ensure the risk ID matches the predefined ID format. If the format is appropriate then the risk is added as an entry into the risk table. If the risk is successfully saved then the saveRisk method returns true, else it returns false.
- validateID method: For each organization an administrator sets the ID format that all risks must follow. The validateID method verifies that every risk matches the set ID format. If the test is passed then the method returns true, else it returns false.
- formatDate method: Puts the closeDate and openDate attributes into a format that will go into the database.

User Class: serves as the representation of a user in our program.

- getRole method: This function will return the role assigned to the user by the admin.
- getUsername method: This method will retrieve the username associated with the user account which is trying to access the system.
- getPassword method: This getter method in this class retrieves the password associated with the specific user account.
- getEmail method: Similar to the other getter methods in the class, this function will retrieve the email address associated with the user account.
- saveUser: This method will save the user data into the database using the DatabaseConnection class.

DatabaseConnection Class: class used to connect to the database and perform operations on and gather information from the existing risk and user tables.

- makeQuery method: The method uses a string parameter to make a query using prepared statements.
- authenticateUser method: The method will use the hash function to

generate a hash for the username and password. Then the method will query the database for those values in the MD5 hash of the table. If successful the method returns true. If unsuccessful the method returns false.

6 HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

Refer to section 6.2 for the layout of every screen. Refer to section 5 for the functionality of every screen.

Screen 1 - Login Screen

User Interaction: Users enter their email address and password then press the 'Login' button. If a user has forgotten their password, they can click on the 'Forgot Password' link.

Security Features: The system utilizes data transmission encryption to prevent the interception of communication by a third party while the data is being transferred. During entry, passwords are hashed and not visible. If several failed attempts have been made to log in to the system, the account in question will temporarily lock to prevent any brute-force attacks on the account/system.

Screen 2 - Set Risk ID Screen

User Interaction: Users will enter or edit a Risk ID, and it will prompt them to either save the changes or to abort and revert all the edits. ID validation info is available for the user to fill out the form without mistakes.

Security Features: The system examines the uniqueness and the format of the Risk ID against that database to avoid duplication and to maintain consistency as well. Input validation and sanitization are put in place to prevent SQL injection or other class of attacks. Changes are recorded for auditing purposes, and each major update requires re-authentication to fasten the identity of the person carrying the action.

Screen 3 - Forgot Password Screen

User Interaction: Users enter a new password of their choosing in the 'Create new password' text input field, re-enter their new password in the 'Confirm password' text input field to confirm it, and then click the 'Reset' button to make the new password active for their account.

Security Features: The system limits the number of times a user can reset their password within a given period to avoid spamming of the password reset function. The system also ensures that the passwords entered in both text input fields match before allowing the user to reset their password.

Screen 4 - Enter Email Screen

User Interaction: Users enter their email addresses and press the submit button. If they need to go back to a previous step or cancel the operation, they can use the 'Back' button.

Security Features: The system implements input validation procedures to verify the format of the provided email address, making sure it fulfills email format rules. The system protects privacy by not letting people know if an email is registered or not, thus preventing Data leakage.

Screen 5 - Verify Email Code Screen

User Interaction: Users input the code they have received and click on 'Verify' to submit. If the code was not received or has been missed, they can choose to send it again.

Security Features: The system makes the code unable to be used anymore after a certain period to improve the security level. Rate limiting on the 'Resend code' button blocks spamming. Several wrong attempts result in a temporary lockdown of the system, leading the user to either restart or seek technical support.

Screen 6 - Audit View Screen

User Interaction: Auditors can engage with the data Table by clicking on particular entries to view additional information. The risks can be sorted using the 'Sort' drop-down menu, filtering by diverse fields like Risk ID which can be entered in the 'Risk ID' field. Aside from that, they can make use of this field to look for some risk entries that they are looking for.

Security Features: The screen can be viewed only by audit-level privileged users to ensure integrity. The data shown is read-only to exclude unauthorized editing. This system records all actions performed on this screen for security auditing, and a timed session expiry prevents unauthorized access due to idle sessions.

Screen 7 - Admin View Screen

User Interaction: Admins can interact with the 'Sort' drop down menu button to sort the 'Risk Report' by either 'Rank', Id', or 'Date'. If an admin wants to see information for a specific risk, they can click the 'Risk Id' button to enter the corresponding Id for the risk they are searching for. The 'Add' button allows an admin to add a risk into the 'Risk Report'. Within the 'Risk Report', an admin can use the edit button to bring up the 'Edit Risk' screen and edit a risk of their choosing. Also within the 'Risk Report', an admin can use the delete button to bring up the 'Delete Risk' screen and delete a risk of their choosing. In the 'User' section the admin can interact with three different buttons including 'Add', 'Edit', and 'Delete'. The 'Add' button allows the admin to add a new user to the system using the 'Add User' screen. The 'Edit' button allows the admin to edit a specific user using the 'Edit User' screen. Lastly, The 'Delete' button allows an admin to delete a specific user from the system.

Security Features: This screen is protected from unauthorized users and can only be viewed by admin-level privileged users. Also, a timed session expiry prevents unauthorized access due to idle sessions.

Screen 8 - Track View Screen

User Interaction: The user can interact with the program by clicking on the "Add a Risk" button, which will lead to an opening window of the sort form or modal. Sorting function can also be applied to put the risk table in a preferred prospective. At times, some risks require updating, therefore, a user can click the "Edit" button to change them or use the "Delete" button to completely delete the associated risk from the system after the confirmation to prevent any accidental data losing.

Security Features: The security measures implemented on the "Track View" screen include authentication of users, role-based access, validation of data input, and safe transmission of the data. Steps like editing and giving the permission of deletions will mainly be logged and confirmation by the user is needed in order to avoid unwanted changes. An automatic logged out feature will help to prevent cases of unauthorized access, thus maintaining the system's security and data integrity.

Screen 9 - Add and Edit Risk/User Screens

User Interaction: The Admin/Track will be able to enter text fields representing different fields of a Risk. Admin will be able to enter information on Users, and pressing submit will allow the corresponding fields to update the database.

Security Features: The view does not allow unauthorized users to add or edit anything they do not have permission to ensure integrity.

Screen 10 - Delete Risk/User

User Interaction: The user will be prompted with a message on if they are sure they want to delete the Risk or User.

Security Features: The view does not allow unauthorized users to delete anything they do not have permission to ensure integrity.

6.2 Screen Designs

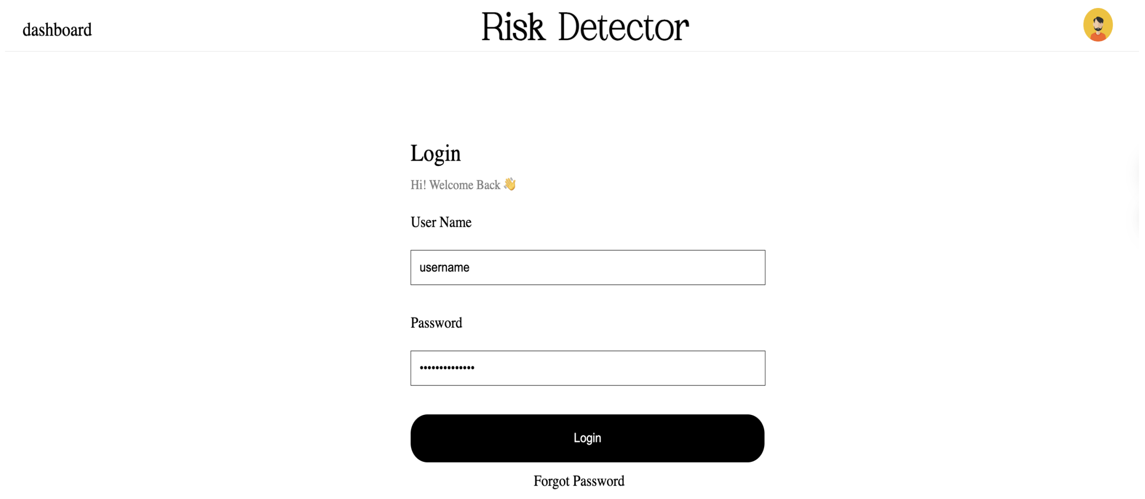


Figure 6.1: Login Page View

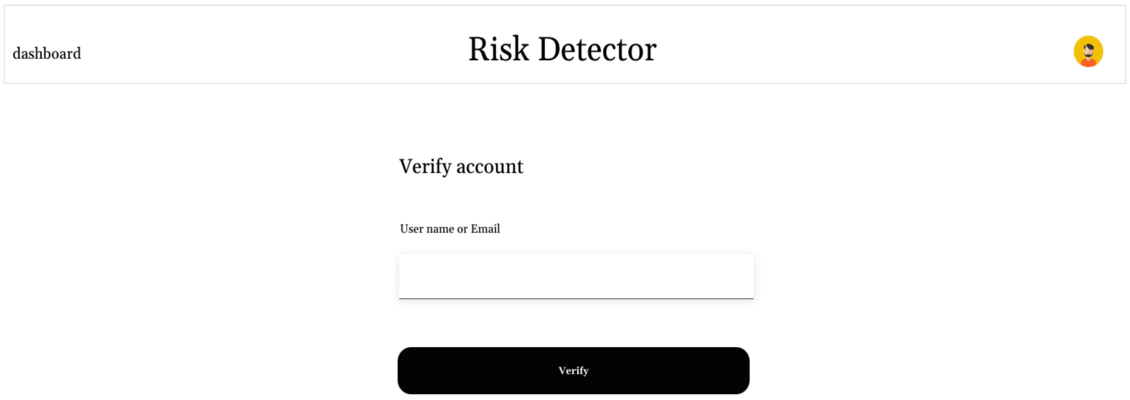
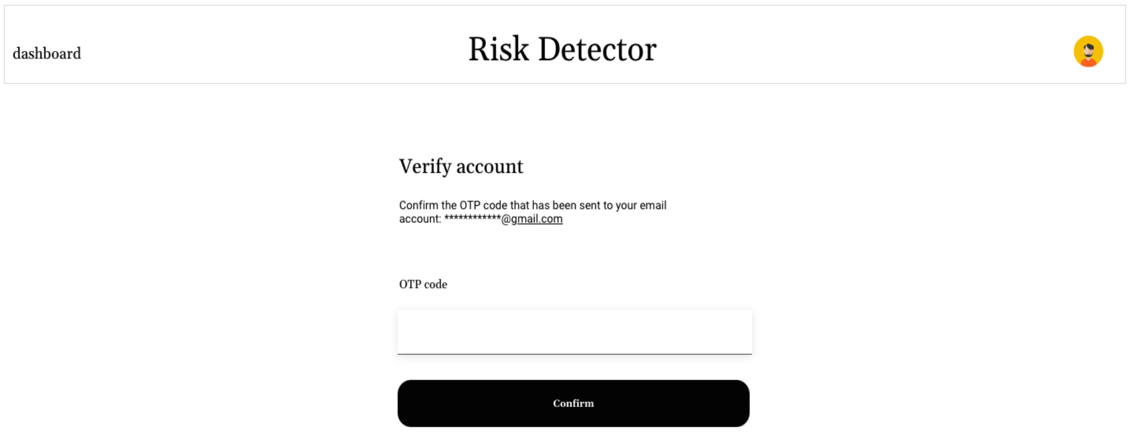


Figure 6.2: Verify Account for Forgot Password View



Confirm the OTP code that has been sent to your email account: ************@gmail.com

Figure 6.3: Verify Your Account by getting OTP Code View

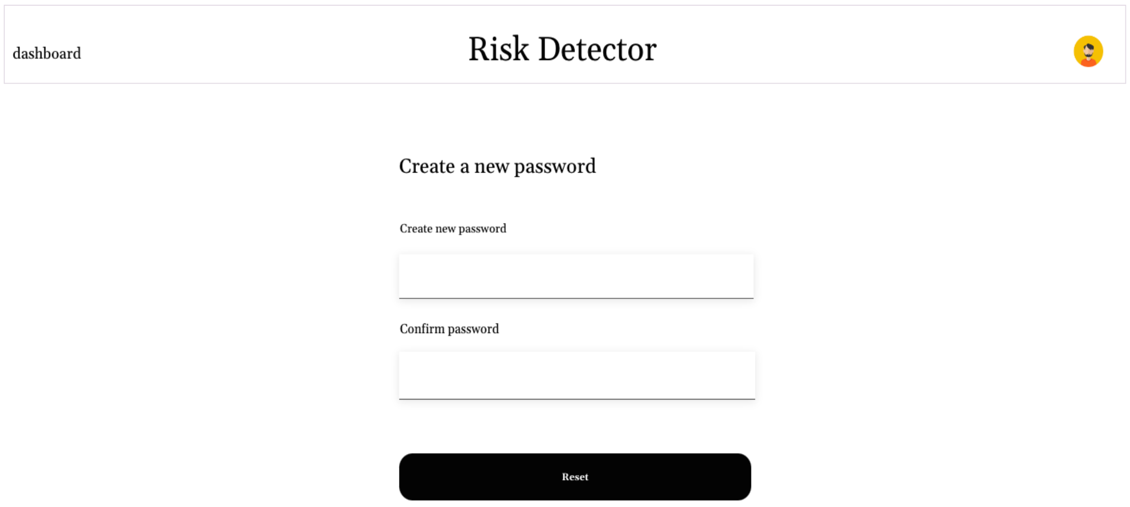


Figure 6.4: Enter New Password and Confirm Password View

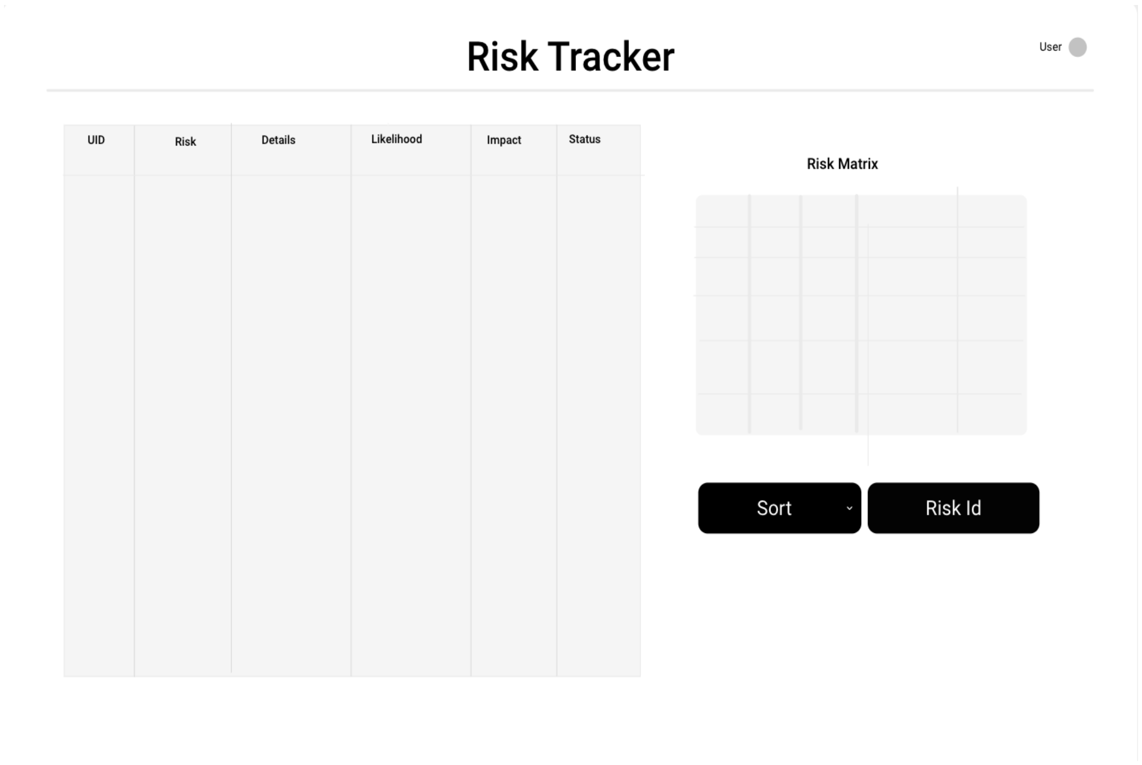
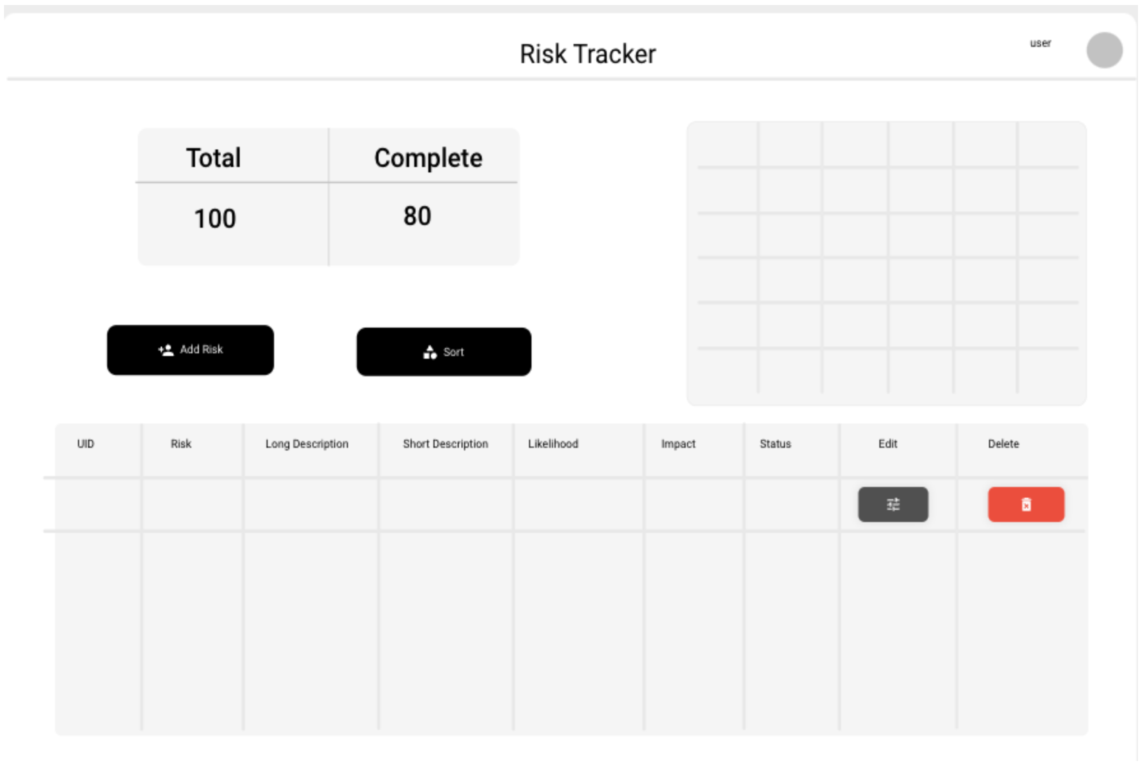


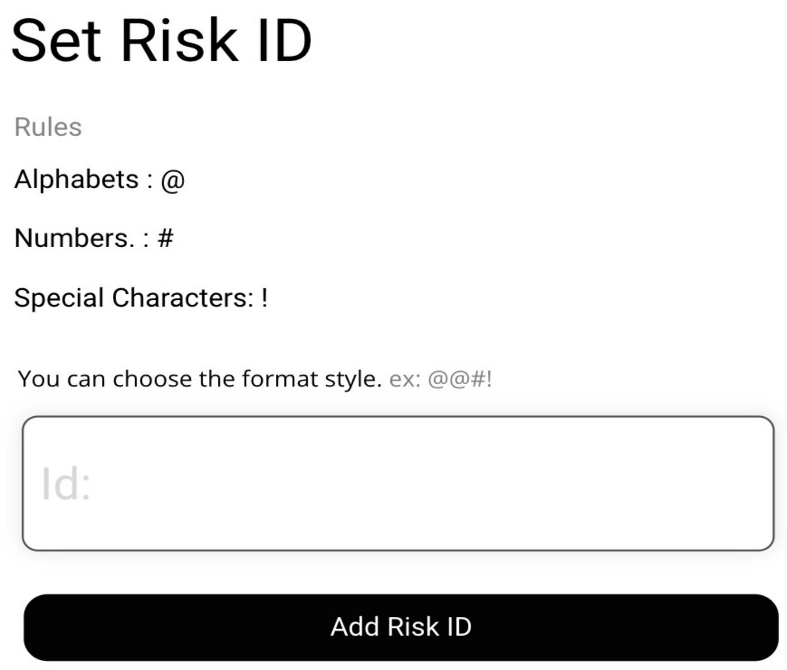
Figure 6.5: Audit View



Total	Complete
100	80

UID	Risk	Long Description	Short Description	Likelihood	Impact	Status	Edit	Delete

Figure 6.6: Track View



Set Risk ID

Rules

Alphabets : @

Numbers. : #

Special Characters: !

You can choose the format style. ex: @@#!

Id:

Figure 6.7: Set Risk ID

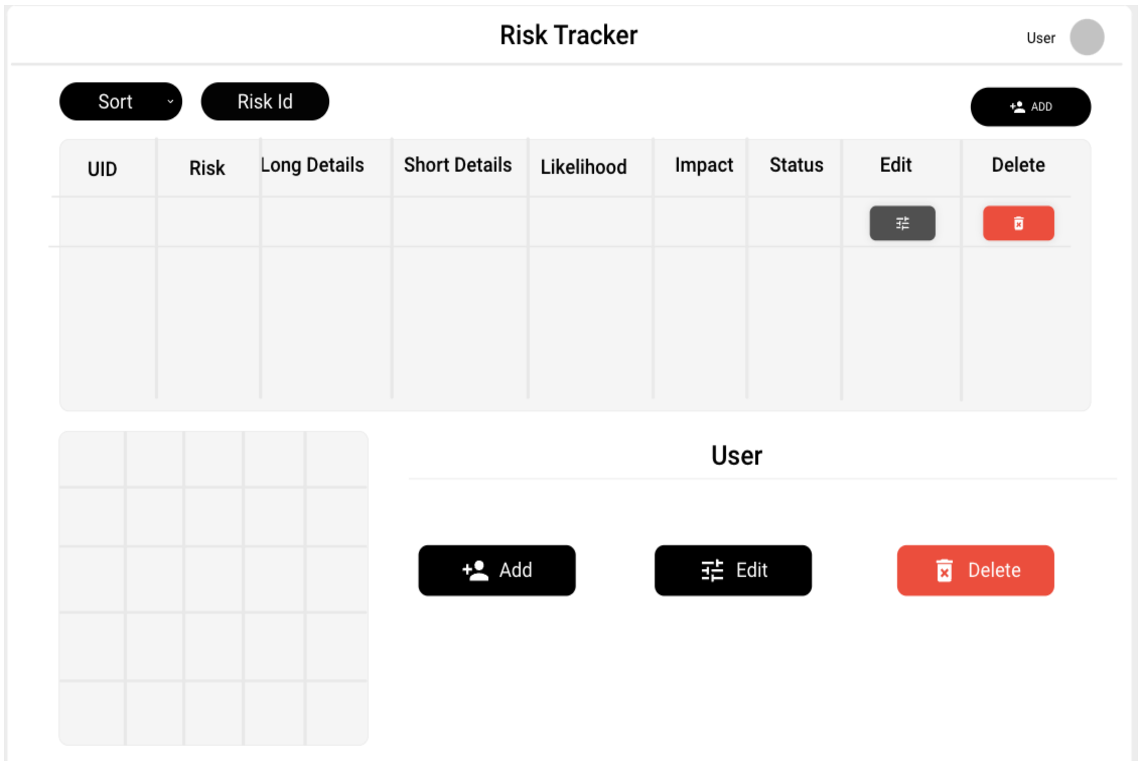


Figure 6.8: Admin View

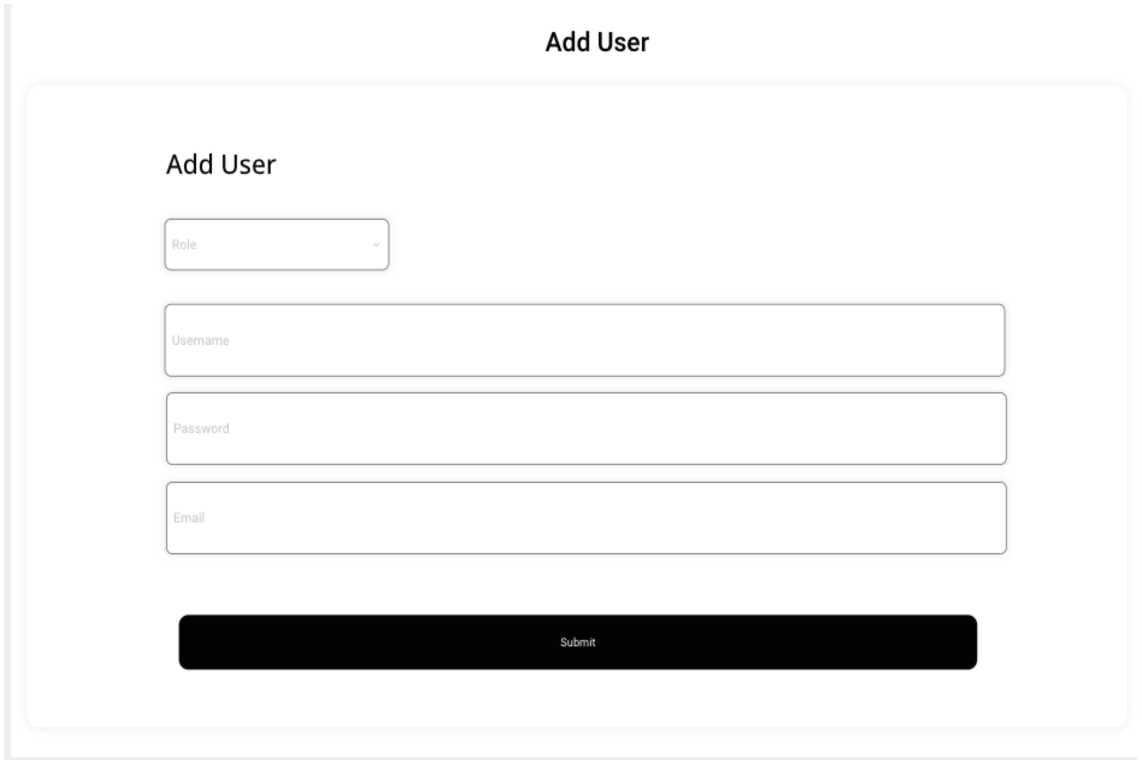


Figure 6.9: Add User

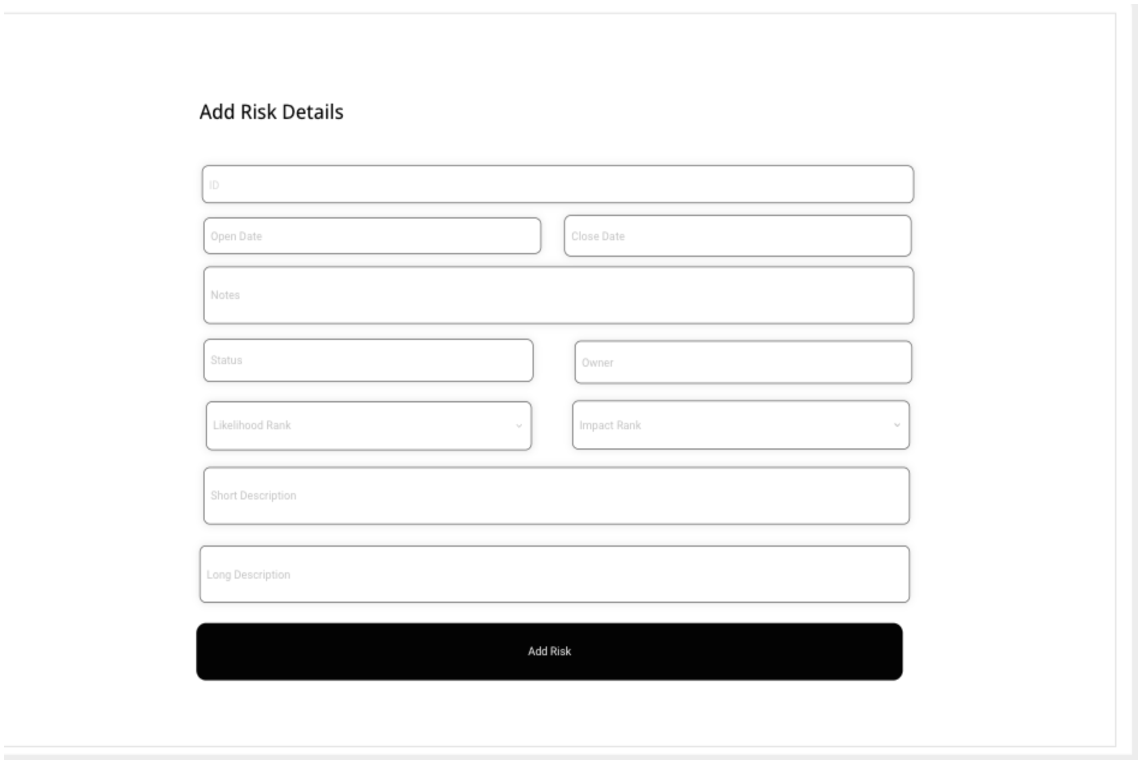


Figure 6.10: Add Risk

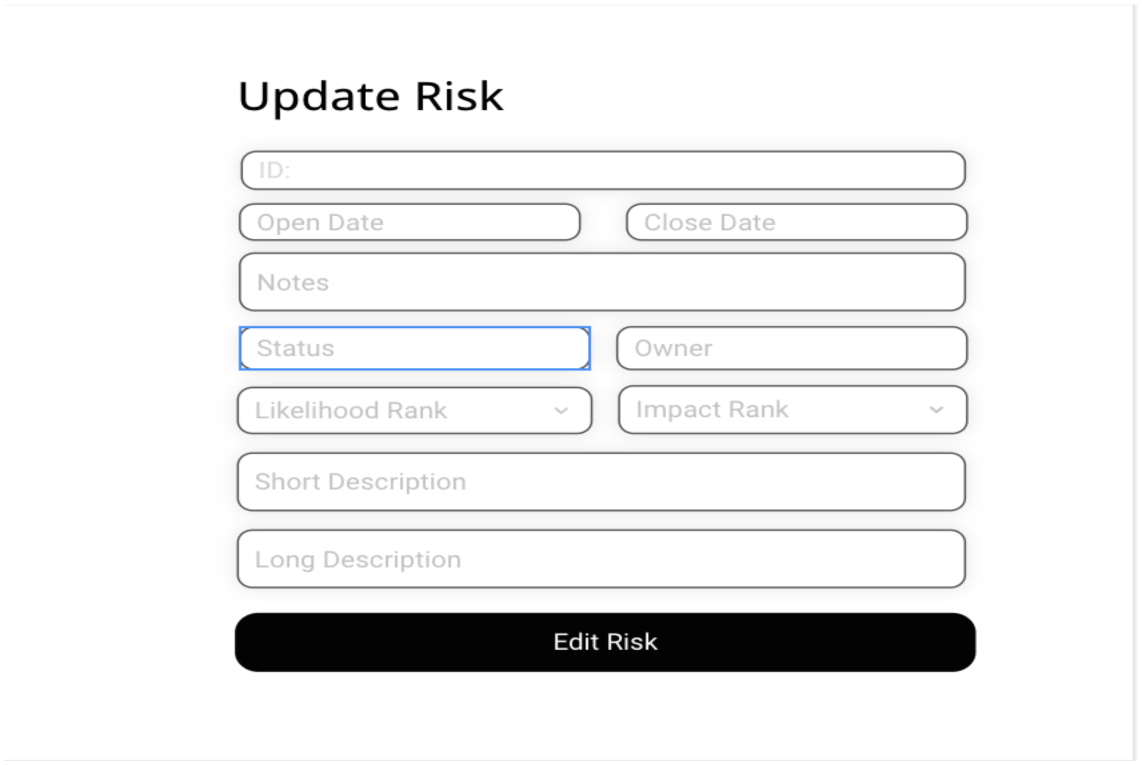


Figure 6.11: Update Risk

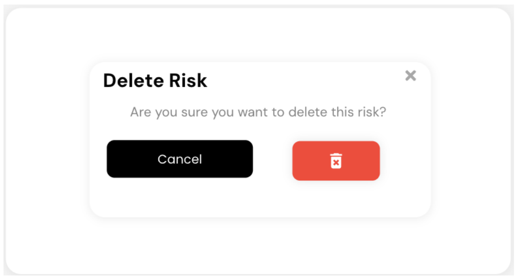


Figure 6.12: Delete Risk

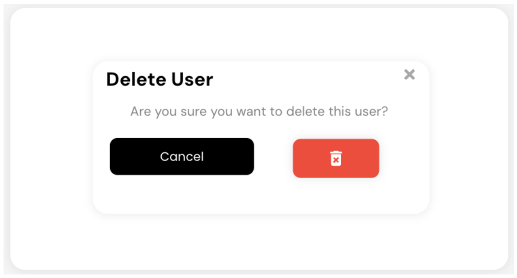


Figure 6.13: Delete User

7 REQUIREMENTS TRACEABILTY MATRIX

Software Requirements Specification	Software Design Document
2.2	5
2.3	3.2, 5
3.2	5
3.3	3.3
3.4	6.1, 6.2
4.1	5