Z PASSWORD MANAGER

PROJECT REPORT

Submitted to University of Kerala in partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF COMPUTER APPLICATIONS

By

Vishnu Shivalal P (70813181119)

Anjali B (03813181131)

SCHOOL OF DISTANCE EDUCATION

UNIVERSITY OF KERALA

THIRUVANANTHAPURAM

2021

DECLARATION

I hereby declare that the project on title “Z Password Manager” submitted to the University of Kerala for the award of Bachelor of Computer Applications is a record of original work done by me and this project work has not been submitted by any other candidate for the award of any other University.

Place:

Date:

ACKNOWLEDGEMENT

Before we get into the things, we would like to add a few heartfelt words for the several prominent personalities, who were part of this project in numerous ways, people who gave unending support and valuable assistance right from the project idea was conceived.

First and foremost we acknowledge the abiding presence and abounding grace of **Lord Almighty** for his shower of blessing which has enabled us to attain success in this endeavor. We are indebted to our **parents** for their moral support and their consistent encouragement with prayers to complete my project work.

We express our deep sense of gratitude to our honorable Director **Dr. R. Vasanthagopal,** School of Distance Education, University of Kerala for the kind inspiration and providing the required facilities and support to complete the project.

It is indeed a great pleasure and privilege to express a sense of gratitude to **Mrs. S.V. Arya**, Internal Guide and Course Coordinator of Bachelor of Computer Applications, School of Distance Education, University of Kerala for the valuable suggestions and support, which helped us a lot in the successful completion of this project.

Finally with a sense of gratitude we wish to acknowledge the untiring efforts of all the staff members of our department for their immense support and encouragement. We would be grateful to all of them for their continued support in the future.

Vishnu Shivalal P	**70813181119**
Anjali B	**03813181131**

SYNOPSIS

INTRODUCTION

The project entitled “Z Password Manager” has been developed as a main project to manage the passwords for naive users.

PROJECT

Z PASSWORD MANAGER is a JAVA application that can be used for managing and storing passwords under a master password in a secure way.

OBJECTIVE

In today’s world, mankind is focusing on increasing productivity, efficiency by introducing automated smart systems all over the world. Most of the time, people use different passwords for different purposes and they have to memorize those passwords all the time. This is a burden for users. Not most of them will not be able to memorize those passwords. Sometimes they can't even keep a track of those passwords and this is where ZPM comes into existence . In such a situation people are moving to automated systems. “**ZPM"** is a software solution for the Z Password Manager system which helps in better performance and increased efficiency of day to day activities and to run smoothly.People nowadays want to run every system in a better and smarter way by reducing the manual effort and by automating the existing systems. This led us to think of a cost effective automation system for users where so many passwords can be stored and it’ll reduce the burden of memorizing passwords.The user interface is carefully optimized for high speed input of a user passwords and the avoidance of forgetting passwords. Passwords and information about passwords can be stored in a secure way. For users, there is a rich set of password management that shows a complete picture of password storing and retrieving. By standardizing the entire password remembering process, the software radically improves serving speed. It’s easy to install and easy to use.

SCOPE

As this project is developed as a java application, it can only be accessed by users who are confined only in their system since their passwords are confidential.

PROBLEM DEFINITION

In the existing system, passwords aren't stored in a secure way and there's also a chance in security breaches for ages. This existing system must be improved since if a user forgets the password, it is hard for the user to recover it and can cause issues. Users might have to reset the password once again and it can be a time killing process.

CONTENTS

1. INTRODUCTION

A password is a word, phrase, or string of characters intended to differentiate an authorized user or process (for the purpose of permitting access) from an unauthorized user, or put another way a password is used to prove one's identity, or authorize access to a resource. It's strongly implied that a password is secret. A password is usually paired with a username or other mechanism to provide authentication.

A computer-based password storing is a mechanism that resides in a system specifically designed to support users by providing accessibility to store and retrieve passwords in a secure way.

That is why we chose to work on a high-end password management system for the design course. On the one hand, this project offers a variety of challenges due to its huge scope and variety of functionalities. On the other hand, it presented an opportunity to experience the variety of concepts and technologies.

Moreover, the complexity of such a project will allow both to automate a system that is completely based on physical record keeping and develop a multi-layered architecture for an application that would benefit. Details regarding the requirements, architecture, and design of this project will be elaborated and explained in detail throughout this report. The project has been developed as an application based software solution to be helpful for users.

2. SYSTEM STUDY ANALYSIS

EXISTING SYSTEM

A password manager tool is software that helps users encrypt, store, and manage passwords. Usual password manager provides medium security and less features at high cost. A normal password manager encrypts passwords and stores them in a less secure database. Those passwords can be accessed by using the master password. A casual password manager stores master passwords more secure than other passwords stored in the user’s database.

PROPOSED SYSTEM

Z Password Manager (ZPM) is an application which helps users to store their passwords in a secure way. ZPM works the same as a normal password manager. But ZPM provides more security for both master passwords and user passwords. This will prevent security threats and breaches. In this application, passwords are stored with some information like username, website address, remark etc. So in this manner, the passwords can be identified easily. This application reduces time taken for the encryption and decryption processes. This increases the productivity of app. ZPM prevents a main security threat called Shoulder Surfing.

Advantages

1. Increased productivity, efficiency by introducing smart systems.
2. Less errors.
3. Prevents security breaches and security threats.
4. Low cost.
5. Reduce the burden of remembering or noting the passwords.
6. Provides a search facility.
7. Passwords and info can be found out within no time.
8. AES level security is provided (Salting and Hashing).

Modules

1. Registration/Sign Up.
2. User Management.
3. Login/Sign In.
4. Homepage/Password List
5. Add Password
6. Edit/Update Password
7. Delete Password

Registration/Sign Up

This is the very beginning stage in the Z Password Manager. The users have to register their accounts by filling a user registration form. Once registration is completed, the users can sign in to their accounts.

User Management

This module deals with the intake of complete details of the users and allows multiple users to use this application. Each user will have a unique username and a password. User accounts can be managed in this module too like changing username, password etc. Resetting user account password is another feature provided by the user account management module.

Login/Sign In

Here, users can login or sign in using the credentials chosen by the user when signing up.

Homepage/Password List

In this module, passwords belonging to the user are shown after user login is verified.

Add Password

This module stores passwords and information about passwords inside the corresponding user's database.

Edit/Update Password

In this module, users can edit or update their passwords.

Delete Password

This module helps to delete passwords from the corresponding user database.

Feasibility Analysis

The main aim of the feasibility study is to determine whether developing the product is financially and technically feasible. Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that is spent on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus when a new application is proposed it normally goes through a feasibility study before it is approved for development. The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibility. The following are its types:

Economic Feasibility

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action. The project is economically feasible as it only requires a computer with an operating system.

Operational Feasibility

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. In the operational feasibility study it is found that the development of the proposed system is operationally feasible.

Security Feasibility

The security of the database from being tampered with by unauthorized personnel different password fields were provided to different users. The passwords are validated and given in the front-end keeping its flexibility and user friendliness in view. It is very important that users should maintain their own individuality and identity so as one cannot overlap or interfere or even tamper with the restricted fields. This application is made secure using proper authentication mechanisms.

3. REQUIREMENT ANALYSIS & SPECIFICATIONS

FUNCTIONAL AND NON - FUNCTIONAL REQUIREMENTS

Method of Requirement Elicitation

- **Observation:** An observation was made through working of an existing system to know how the data are processed and a caretaker was assigned.
- **Document Reference:** For preparation of information of caretakers, the details are hard coded in the file. Currently the updating of the information and the addition of new caretakers are done manually and the data is updated in the file.
- **Discussions:** To understand the system better, discussions were held with the caretakers, agencies and also with the end users. This led to the better design of the system.

Hardware Requirements

- **Processor:** Intel 3 Core
- **RAM:** 128mb RAM
- **HDD:** 20GB Hard Disk Space or Above
- **Keyboard:** Standard 102 Keys
- **Mouse:** 3 Buttons

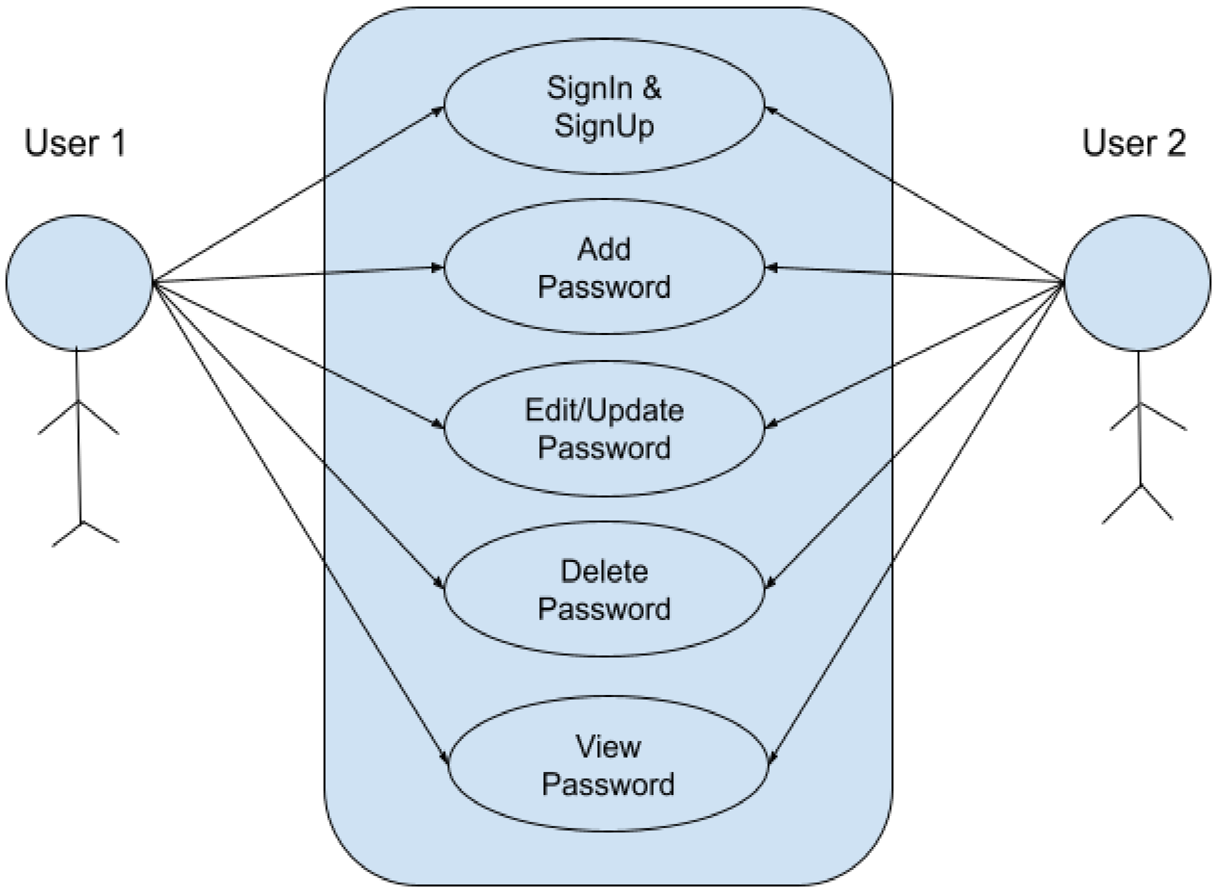
Hardware Requirements

- **Operating System:** Windows 10
- **Environment:** IntelliJ IDEA
- **Language:** Java, FXML
- **Backend:** MySQL Server

Front End : JavaFX

Back End : MySQL Server

USE CASE DIAGRAM



4. SOFTWARE DESIGN

Input Design

Input design is a part of overall system design, which requires very careful attention. If data going into the system is incorrect, then the processing and output will magnify these errors. Thus the designer has a number of clear objectives in the different stages of input design:

- To produce a cost effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that input is acceptable to and understood by the user.

Input design is processing of converting the user oriented description of the inputs of the system. The goal of designing input data to make data entry as easy, logical and free from errors as possible. In entering data, operators need to know the following.

- The allocated space for each field.
- Field sequence which must match that in the source document.
- The format in which data fields are entered.

When we approach input data design, we design source documents that capture the data and then select the media used to enter them into the computer. There are different ways in which data can be introduced into the system such as the data is converted into a machine identifiable form by some realistic source document and types in the relevant items using a keyboard connected to the system. The document can be read directly by a machine and this converts information held in the human sensible form into a machine readable form without need for human investigations. Data entered into a system through a keyboard. This is done interactively by the person using the system. Data is presented in a form suitable for computers as a result of some of the processing. The data entry in the system has been designed so as to make it user friendly and also to incorporate certain validation checks.

The field name must be documented. The field name must be known to data entry operator or users so that the data entry will not exceed the allocated space.

Input	Module
User Credentials	Registration
User details	User Management
Username and Password	Login
Password Lists	Password Management
Add password	Add
Delete password	Delete
Edit or Update Password	Update

Output Design

At the beginning of output design various types of outputs (external, internal, operational, and interactive and turnaround) are defined. Then the format, content, location, frequency, volume and sequence of the output are specified. The content of the output must be defined in detail. The system analyst has two specific objectives at this stage. To interpret and communicate the results of the computer part of a system users in a form that they can understand and which meets their requirements.

To communicate the output design specifications to programmers in a way, this is unambiguous, comprehensive and capable of being translated into a programming language. The primary consideration in the design of all output is the information requirement and other objectives of the users. It is the most important and direct source of information to the user. A major form of output is a hard copy. Print out should be designed around the output requirements of the user.

DATA FLOW DIAGRAMS

A data flow diagram is a graphic representation of a system that shows data flows to, from and within the system, processing functions that change the date in some manner, and the storage of this data. They are networks of related system functions that indicate form where information is revived and to where it is sent. An external entity is the originator or receiver of data or information.

A data store symbol portraits a file or database in which data resides. A process is depicted by a circle, sometimes it is called a bubble or transform. Process portraits the transformation of the content of status of data.

Data Flow Diagram (DFD) is an important tool used by system analysts. DFD provides an overview of what data a system would process, what transformation of data is done, what files are used and where the results flow. The graphical representation of the system makes it a good communication tool between the user and the analyst.

Analysis models help us to understand the relationship between different components in the design. Analysis model shows the user clearly how a system will function. This is the first technical representation of the system. The analysis modeling must achieve three primary objectives:

- To establish a basis for creation of software design.
- To describe what the user requires.
- To define the user requirements that can be validated once the software is built.

A data flow diagram is a graphical technique that depicts information flow and transforms that are applied as data move from input to output. The DFD is used to represent increasing information flow and functional details. A level 0 DFD, also called fundamental system model or a Context model, represents the entire software elements as single bible with input and output indicated by incoming and outgoing arrows respectively.

Additional process and information flow parts are represented in the next level i.e. Level 1 DFD. Each of the processes represented at Level 1 are sub-functions of the overall system depicted in the Context model. Any process which is complex in Level 1, will be further represented into sub-function in the next level, i.e. in Level 2.

Definition

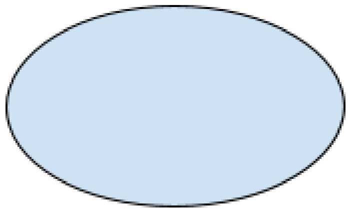
Data flow diagram is a means of representing a system at any level of detail with a graphic network of symbols showing data flows, data stores, data processes and data sources or destinations.

Data Flow Diagram Principles

A system can be decomposed into subsystems, and subsystems can be further decomposed into lower level subsystems. Each subsystem represents a process or activity in which data is processed. At the lowest level, processes can no longer be decomposed. Each ‘process’ has the characteristics of a system. A process must have input and output. The basic elements of DFD are

Process

An activity that changes or transforms data flows. Since they transform incoming data to outgoing data, all processes must have inputs and outputs on a DFD. This symbol is given a simple name based on its function, such as “Ship Order,” rather than being labeled “process” on a diagram. Processes are typically oriented from top to bottom and left to right on a data flow diagram. Circle or Bubble:

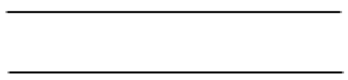


Data Flow

Movement of data between external entities, processes and data stores is represented with an arrow symbol, which indicates the direction of flow. This data could be electronic, written or verbal. Input and output data flows are labeled based on the type of data or its associated process or data store, and this name is written alongside the arrow. Arrows:

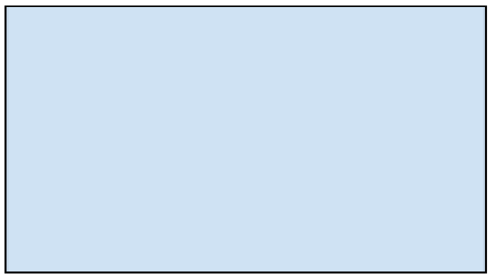
Data Store

A data store does not generate any operations but simply holds data for later access. Data stores could consist of files held long term or a batch of documents stored briefly while they wait to be processed. Input flows to a data store include information or operations that change the stored data. Output flows would be data retrieved from the store. A data store is a place for holding information within the system. Here data is stored or referenced by a process in the system. Two parallel lines:

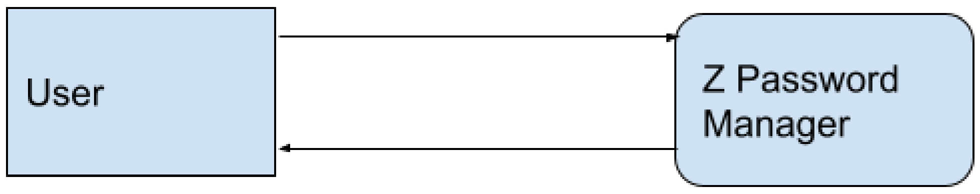


External Entity

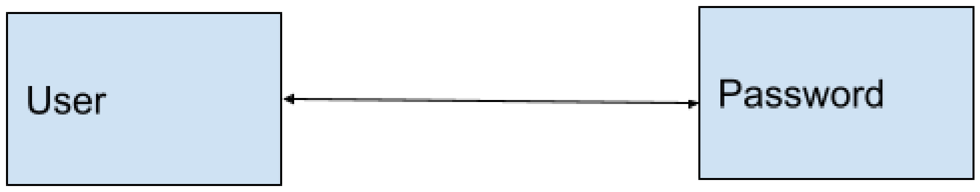
Also known as actors, sources or sinks, and terminators, external entities produce and consume data that flows between the entity and the system being diagrammed. These data flows are the inputs and outputs of the DFD. Since they are external to the system being analyzed, these entities are typically placed at the boundaries of the diagram. They can represent another system or indicate a subsystem. Rectangle:



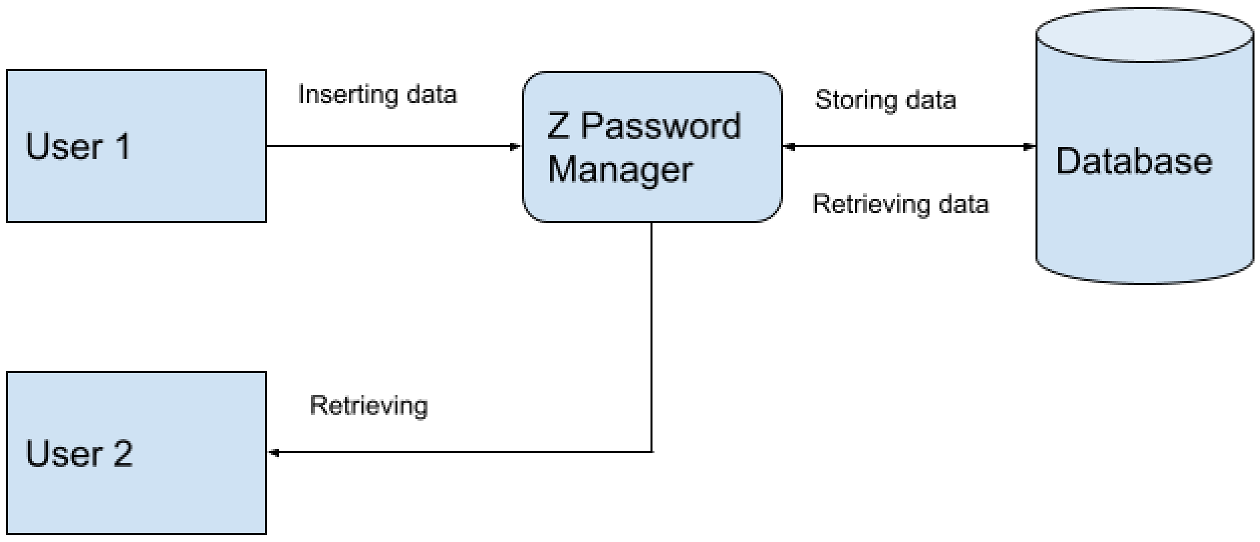
Level 0 (DFD Context diagram):



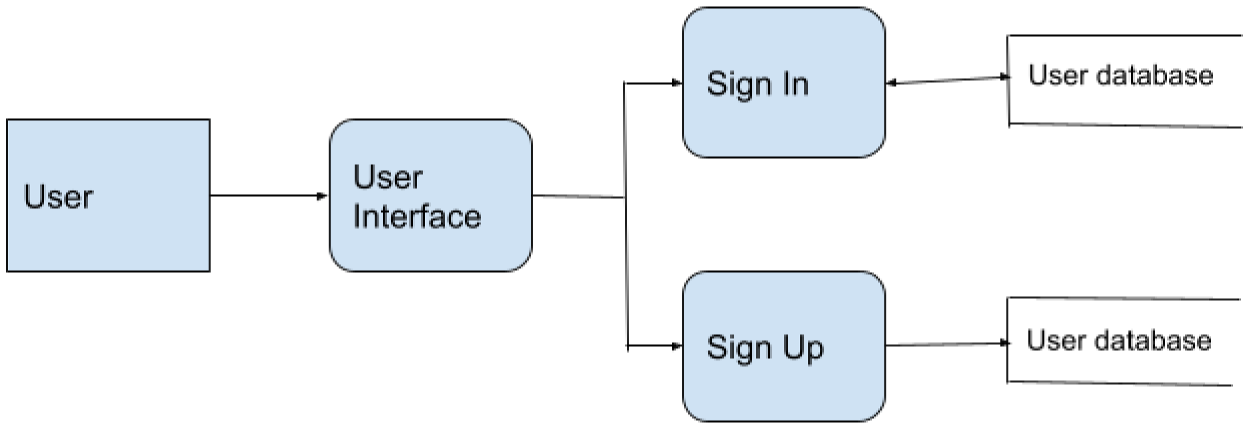
Level 1 (DFD Context diagram):



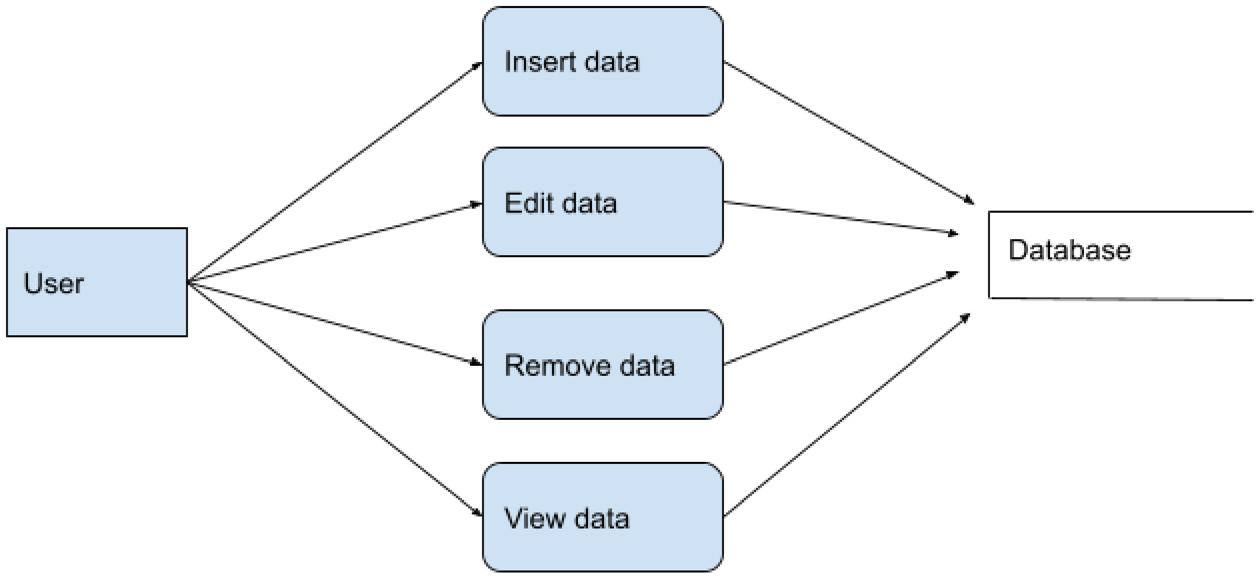
Level 1.1 (DFD Context diagram):



Level 1.2 (DFD Context diagram):



Level 1.3 (DFD Context diagram):



SOFTWARE DESIGN

Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation. For assessing user requirements, an SRS (Software Requirement Specification) document is created whereas for coding and implementation, there is a need for more specific and detailed requirements in software terms. The output of this process can directly be used in implementation in programming languages.

Software design is the first step in SDLC (Software Design Life Cycle), which moves the concentration from problem domain to solution domain. It tries to specify how to fulfill the requirements mentioned in SRS.

Software Design Levels

Software design yields three levels of results:

Architectural Design

The architectural design is the highest abstract version of the system. It identifies the software as a system with many components interacting with each other. At this level, the designers get the idea of the proposed solution domain.

High-level Design

The high-level design breaks the ‘single entity-multiple component’ concept of architectural design into a less-abstracted view of subsystems and modules and depicts their interaction with each other. High-level design focuses on how the system along with all of its components can be implemented in forms of modules. It recognizes the modular structure of each sub-system and their relation and interaction among each other.

Detailed Design

Detailed design deals with the implementation part of what is seen as a system and its sub-systems in the previous two designs. It is more detailed towards modules and their implementations. It defines the logical structure of each module and their interfaces to communicate with other modules.

Modularization

Modularization is a technique to divide a software system into multiple discrete and independent modules, which are expected to be capable of carrying out task(s) independently. These modules may work as basic constructs for the entire software. Designers tend to design modules such that they can be executed and/or compiled separately and independently.

Concurrency

Back in time, all software is meant to be executed sequentially. By sequential execution we mean that the coded instruction will be executed one after another implying only one portion of the program being activated at any given time. Say, a software has multiple modules, then only one of all the modules can be found active at any time of execution.

In software design, concurrency is implemented by splitting the software into multiple independent units of execution, like modules and executing them in parallel. In other words, concurrency provides capability to the software to execute more than one part of code in parallel to each other.

It is necessary for the programmers and designers to recognize those modules, which can be made parallel execution.

Coupling and Cohesion

When a software program is modularized, its tasks are divided into several modules based on some characteristics. As we know, modules are a set of instructions put together in order to achieve some tasks. They are, though, considered as a single entity but may refer to each other to work together. There are measures by which the quality of a design of modules and their interaction among them can be measured. These measures are called coupling and cohesion.

Cohesion

Cohesion is a measure that defines the degree of intra-dependability within elements of a module.The greater the cohesion, the better is the program design.

There are seven types of cohesion, namely :-

Coincidental cohesion: It is unplanned and random cohesion, which might be the result of breaking the program into smaller modules for the sake of modularization. Because it is unplanned, it may serve confusion to the programmers and is generally not-accepted.

Logical cohesion: When logically categorized elements are put together into a module, it is called logical cohesion.

Temporal Cohesion: When elements of a module are organized such that they are processed at a similar point in time, it is called temporal cohesion.

Procedural cohesion: When elements of a module are grouped together, which are executed sequentially in order to perform a task, it is called procedural cohesion.

Communicational cohesion: When elements of a module are grouped together, which are executed sequentially and work on the same data (information), it is called communicational cohesion.

Sequential cohesion: When elements of a module are grouped because the output of one element serves as input to another and so on, it is called sequential cohesion.
Functional cohesion - It is considered to be the highest degree of cohesion, and it is highly expected. Elements of modules in functional cohesion are grouped because they all contribute to a single well-defined function. It can also be reused.

Coupling

Coupling is a measure that defines the level of inter-dependability among modules of a program. It tells at what level the modules interfere and interact with each other. The lower the coupling, the better the program.
There are five levels of coupling, namely :-

Content coupling: When a module can directly access or modify or refer to the content of another module, it is called content level coupling.

Common coupling: When multiple modules have read and write access to some global data, it is called common or global coupling.

Control coupling: Two modules are called control-coupled if one of them decides the function of the other module or changes its flow of execution.

Stamp coupling: When multiple modules share a common data structure and work on different parts of it, it is called stamp coupling.

Data coupling: Data coupling is when two modules interact with each other by means of passing data (as parameter). If a module passes data structure as parameter, then the receiving module should use all its components. Ideally, no coupling is considered to be the best.

Design Verification

The output of the software design process is design documentation, pseudo codes, detailed logic diagrams, process diagrams, and detailed description of all functional or non-functional requirements. The next phase, which is the implementation of software, depends on all outputs mentioned above.

It then becomes necessary to verify the output before proceeding to the next phase. The earlier any mistake is detected, the better it is or it might not be detected until testing of the product. If the outputs of the design phase are in formal notation form, then their associated tools for verification should be used, otherwise a thorough design review can be used for verification and validation.

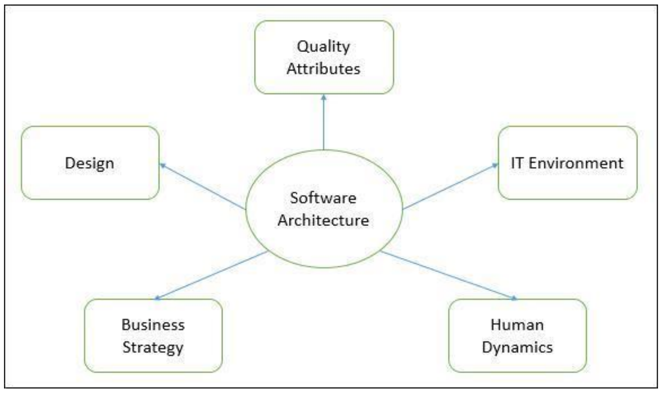
By structured verification approach, reviewers can detect defects that might be caused by overlooking some conditions. A good design review is important for good software design, accuracy and quality.

Architectural Design

Software Architecture typically refers to the bigger structures of a software system, and it deals with how multiple software processes cooperate to carry out their tasks.

Software Design refers to the smaller structures and it deals with the internal design of a single software process. By the end of this tutorial, the readers will develop a sound understanding of the concepts of software architecture and design concepts and will be in a position to choose and follow the right model for a given software project.

The architecture of a system describes its major components, their relationships (structures), and how they interact with each other. Software architecture and design includes several contributory factors such as Business strategy, quality attributes, human dynamics, design, and IT environment.



We can segregate Software Architecture and Design into two distinct phases: Software Architecture and Software Design. In architecture, nonfunctional decisions are cast and separated by the functional requirements. In design, functional requirements are accomplished.

Software Architecture

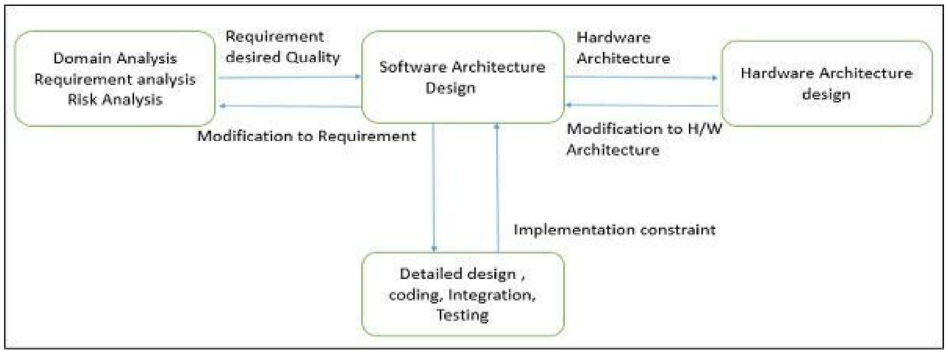
Architecture serves as a blueprint for a system. It provides an abstraction to manage the system complexity and establish a communication and coordination mechanism among components.

- It defines a structured solution to meet all the technical and operational requirements, while optimizing the common quality attributes like performance and security.
- Further, it involves a set of significant decisions about the organization related to software development and each of these decisions can have a considerable impact on quality, maintainability, performance, and the overall success of the final product. These decisions comprise of −
  - Selection of structural elements and their interfaces by which the system is composed.
  - Behavior as specified in collaborations among those elements.
  - Composition of these structural and behavioral elements into large subsystems.
  - Architectural decisions align with business objectives.
  - Architectural styles guide the organization.

Software Design: Software design provides a design plan that describes the elements of a system, how they fit, and work together to fulfill the requirement of the system. The objectives of having a design plan are as follows −

- To negotiate system requirements, and to set expectations with customers, marketing, and management personnel.
- Act as a blueprint during the development process.
- Guide the implementation tasks, including detailed design, coding, integration, and testing.

It comes before the detailed design, coding, integration, and testing and after the domain analysis, requirements analysis, and risk analysis.



***Goals of Architecture*:** The primary goal of the architecture is to identify requirements that affect the structure of the application. A well-laid architecture reduces the business risks associated with building a technical solution and builds a bridge between business and technical requirements.

Some of the other goals are as follows −

- Expose the structure of the system, but hide its implementation details.
- Realize all the use-cases and scenarios.
- Try to address the requirements of various stakeholders.
- Handle both functional and quality requirements.
- Reduce the goal of ownership and improve the organization’s market position.
- Improve quality and functionality offered by the system.
- Improve external confidence in either the organization or system.

Limitations: Software architecture is still an emerging discipline within software engineering. It has the following limitations :−

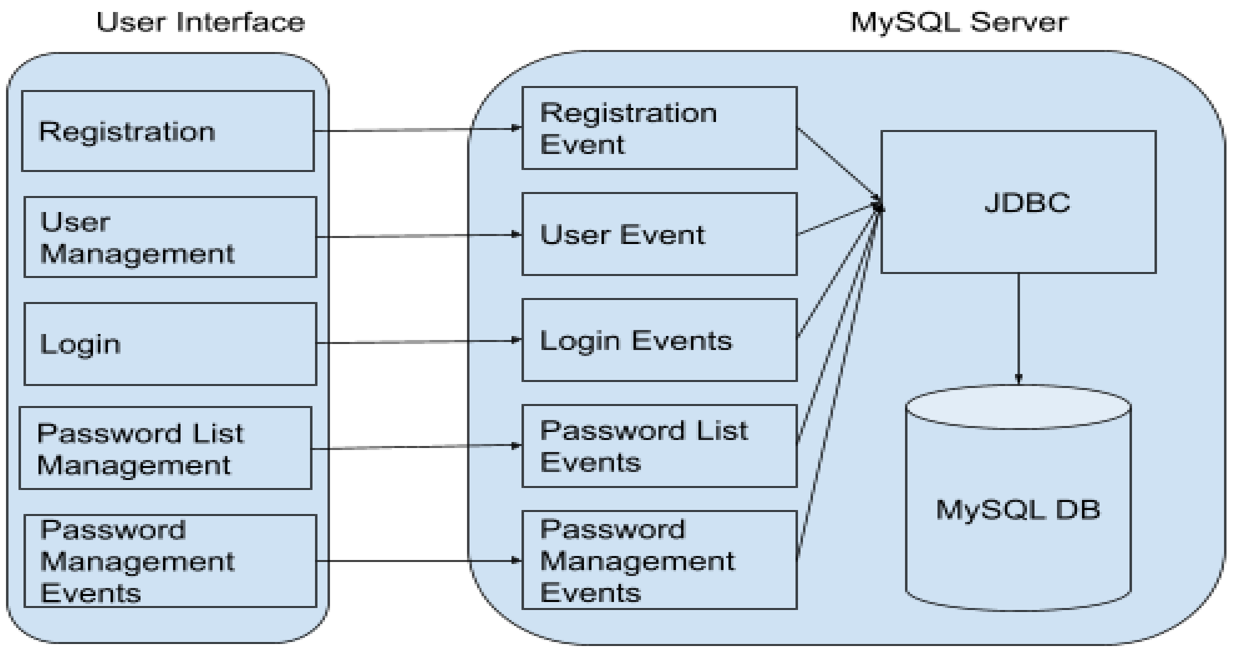
- Lack of tools and standardized ways to represent architecture.
- Lack of analysis methods to predict whether architecture will result in an implementation that meets requirements.

- Lack of awareness of the importance of architectural design to software development.
- Lack of understanding of the role of software architect and poor communication among stakeholders.
- Lack of understanding of the design process, design experience and evaluation design.

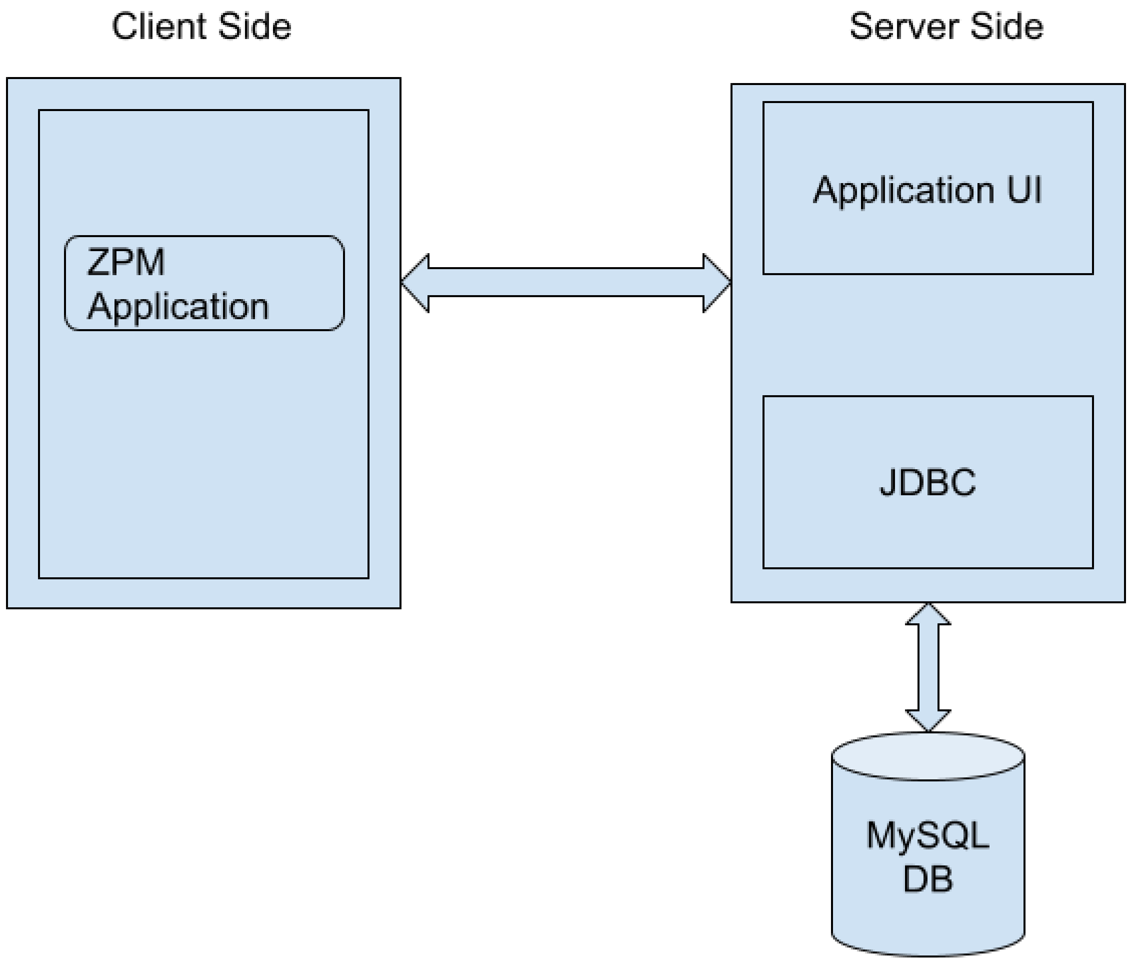
Role of Software Architect: A Software Architect provides a solution that the technical team can create and design for the entire application. A software architect should have expertise in the following areas −

- Design Expertise
- Domain Expertise
- Technology Expertise
- Methodological Expertise
- Hidden Role of Software Architect
- Deliverables of the Architect

Architecture of Z Password Manager (Module level architecture)

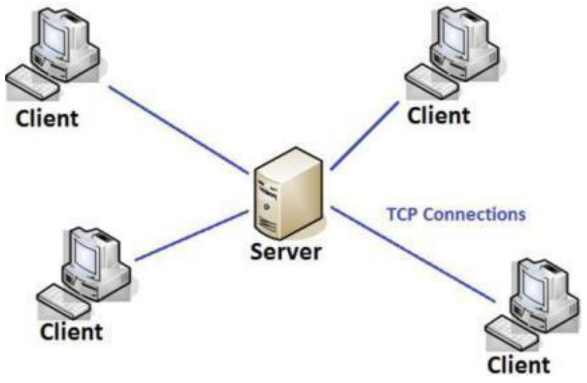


System level architecture



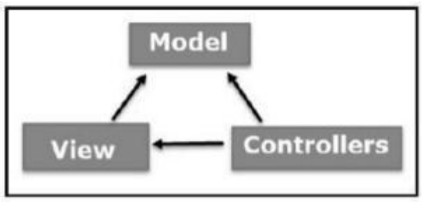
The above module diagram shows the overall architecture of the "Z Password Management". ***ZPM*** will be an application based software so it mainly consists of 2 major modules. The Application UI which will be accessible for the users. Second one will be the backend hosting MySQL Server Database. Backend application will be developed with help of JDBC.

Client Server Architecture is a computing model in which the server hosts, delivers and manages most of the resources and services to be consumed by the client. This type of architecture has one or more client computers connected to a central server over a network or internet connection. This system shares computing resources. **Client/server architecture** is also known as a networking computing model or client/server network because all the requests and services are delivered over a network.



Client-Server Architecture

Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the **model**, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects. Following are the components of MVC :−



MVC Architecture

Model: The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data.
For example, a Customer object will retrieve the customer information from the database, manipulate it and update its data back to the database or use it to render data.

View: The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controller: Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

User Interface Design

Form Design is an important feature of the input design phase, because using the forms as a means of entry enters almost all the information needed during the implementation of the project. This system provides many user-friendly forms that help the user to interact with the system easily. The input forms are designed in such a manner so as to avoid any sort of confusion and to guide the user in the correct track. A thorough study has been made on the type and how the input form is to be designed. Some inputs from the user may cause several errors and hence there are strict validation measures provided. Some of the features included are :-

Effectiveness: All the input screens serve a specific purpose. The forms are designed taking into consideration the requirements of the system.

Accuracy: Each form measures proper links and completion and each form is properly navigable to other areas of the project.

Ease of Use: The forms are authentic and require no extra time to understand. This is especially desired in these types of automated manual systems.

Simplicity: All the forms are simple to read and uncluttered and are thereby easy to use and also to move from one screen to another.

Attractiveness: The forms are designed in such an appealing manner that is sure to please the user. The software uses multiple forms for all types of inputs.

Error Avoidance: Every effort is made to ensure the accuracy of the input data.

Data Validation: The input data are validated to minimize the errors in the data entry. This enables the user to enter the required data or to correct them if they have been entered wrongly. Appropriate messages are provided to keep the user informed of wrong entry.

Database Design

Database Design is the process of converting user-originated inputs to a computer-based format. The database design phase is used to design the input within the predefined guidelines. Inaccurate input data is the most common cause of errors in the data processing. Errors entered by data entry operators can be controlled by input design. Input design consists of developing specifications and procedures for data preparations, data entry and validation.

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database model. Database management system manages the data accordingly. Database design involves classifying data and identifying interrelationships.

In a majority of cases, a person who is doing the design of a database is a person with expertise in the area of database design, rather than expertise in the domain from which the data to be stored is drawn e.g. financial information, biological information etc. Therefore, the data to be stored in the database must be determined in cooperation with a person who does have expertise in that domain, and who is aware of what data must be stored within the system. This process is one which is generally considered part of requirements analysis, and requires skill on the part of the database designer to elicit the needed information from those with the domain knowledge. This is because those with the necessary domain knowledge frequently cannot express clearly what their system requirements for the database are as they are unaccustomed to thinking in terms of the discrete data elements which must be stored. Data to be stored can be determined by Requirement Specification.

Database design forms an important part of every project. The management of data involves both the definition of structure for the storage of information and provision of mechanisms for manipulation of information. The database system must provide safety for the information stored; despite system crashes or attempts of unauthorized access the database used in this project is MySQL Server DB

Following is the database design for the Z Password Manager.

User Table

NAME	**DATATYPE**	**DESCRIPTION**
ID	**INT**	**ID NUMBER**
LNAME	**VARCHAR**	**NAME OF USER**
EMAIL	**VARCHAR**	**EMAIL OF USER**
USERNAME	**VARCHAR**	**USERNAME OF USER**
PASSWORD	**VARCHAR**	**PASSWORD OF USER**

Password Table (Home Table)

NAME	DATATYPE	DESCRIPTION
ID	INT	ID NUMBER
EMAIL	VARCHAR	EMAIL OF USER
USERNAME	VARCHAR	USERNAME OF USER
PASSWORD	VARCHAR	PASSWORD OF USER
URL	VARCHAR	URL OF WEBSITE
REMARK	VARCHAR	INFO ABOUT PASSWORD

5. SYSTEM IMPLEMENTATION

The system developed is very user-friendly and the detailed documentation is also given to the user as online help wherever necessary. The implementation phase normally ends with the formal test involving all the components. The entire system is developed using Java, FXML, and MySQL Database.

MODULE DESCRIPTION

Following are the modules involved in the system.

1. Registration/Sign Up
2. User Management
3. Login/Sign In
4. Password List Management
5. Add Password
6. Edit/Update Password
7. Delete Password

1. Registration/Sign Up

This is the very beginning stage in the Z Password Manager. The users have to register their accounts by filling a user registration form. Once registration is completed, the users can sign in to their accounts.

2. User Management

This module deals with the intake of complete details of the users and allows multiple users to use this application. Each user will have a unique username and a password. User accounts can be managed in this module too like changing username, password etc. Resetting user account password is another feature provided by the user account management module.

3. Login/Sign In

Here, users can login or sign in using the credentials chosen by the user when signing up.

4. Homepage/Password List

In this module, passwords belonging to the user are shown after user login is verified.

5. Add Password

This module stores passwords and information about passwords inside the corresponding user's database.

6. Edit/Update Password

In this module, users can edit or update their passwords.

7. Delete Password

This module helps to delete passwords from the corresponding user database.

IMPLEMENTATION DETAILS

The entire system is developed using the following latest technologies and frameworks.

- Java & JavaFX
- FXML & Scene Builder
- JDBC
- IntelliJ IDEA
- MySQL Server

Java

Java is a general-purpose, class-based, object-oriented programming language designed for having lesser implementation dependencies. It is a computing platform for application development. Java is fast, secure, and reliable, therefore. It is widely used for developing Java applications in laptops, data centers, game consoles, scientific supercomputers, cell phones, etc. Java Platform is a collection of programs that help programmers to develop and run Java programming applications efficiently. It includes an execution engine, a compiler, and a set of libraries in it. It is a set of computer software and specifications. James Gosling developed the Java platform at Sun Microsystems, and the Oracle Corporation later acquired it. Java is a multi-platform, object-oriented, and network-centric language. It is among the most used programming languages. Java is also used as a computing platform. It is considered as one of the fast, secure, and reliable programming languages preferred by most organizations to build their projects.

JavaFX

JavaFX is an open source, next generation client application platform for desktop, mobile and embedded systems built on Java. It is a collaborative effort by many individuals and companies with the goal of producing a modern, efficient, and fully featured toolkit for developing rich client applications.

FXML

FXML is an XML format that enables you to compose JavaFX GUIs in a fashion similar to how you compose web GUIs in HTML. *FXML* thus enables you to separate your JavaFX layout code from the rest of your application code. This cleans up both the layout code and the rest of the application code. FXML can be used both to compose the layout of a whole application GUI, or just part of an application GUI, e.g. the layout of one part of a form, tab, dialog etc.

Scene builder

Scene Builder is a visual layout tool that lets users quickly design JavaFX application user interfaces, without coding. Users can drag and drop UI components to a work area, modify their properties, apply style sheets, and the FXML code for the layout that they are creating is automatically generated in the background. The result is an FXML file that can then be combined with a Java project by binding the UI to the application's logic

JDBC (Java Database Connectivity)

Java Database Connectivity is an application programming interface for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database.

IntelliJ IDEA

IntelliJ is one of the most powerful and popular Integrated Development Environments (IDE) for Java. It is developed and maintained by **JetBrains** and available as community and ultimate edition. This feature rich IDE enables rapid development and helps in improving code quality. **IntelliJ IDEA** is an integrated development environment (IDE) written in Java for developing computer software. It is developed by JetBrains (formerly known as IntelliJ), and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition. Both can be used for commercial development.

MySQL Server

MySQL is an open-source relational database management system. Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data

Technologies

- User Interface - JavaFX
- Server Side - JDBC
- Database - MySQL Database

Development Tools

- IntelliJ IDEA (Coding)
- Scene Builder (Form Designing)
- MySQL Workbench (Database)

Hardware Requirements

Machine with Windows 7(32-bit or 64-bit) or above version Operating system installed with following specs

- Processor with 1GHZ or higher.
- Ram size with 1GB or higher.
- Hard disk with enough space.

Software required to be pre-installed in the system

- Java Runtime Environment.

6. SYSTEM TESTING

The software, which has been developed, has to be tested to prove its validity. Testing is considered to be the least creative phase of the whole cycle of system design. In the real sense it is the phase, which helps to bring out the creativity of the other phases, and makes it shine. That is testing represents the ultimate review of specification, design and code. Any software has to be tested with pre-planned strategies. As Roger Pressman states, the preparation for testing should start as soon as the design of the system starts. To carry out the testing in an efficient manner a certain amount of strategic planning has to be done. Any testing strategy must incorporate test planning, test case design, test execution and the resultant data collection and evaluation. Testing is a process of executing a program with the intent of finding an error.

System Testing Process:

System Testing is performed in the following steps:

Test Environment Setup: Create testing environment for better quality testing.

Create Test Case: Generate test case for the testing process.

Create Test Data: Generate the data that is to be tested.

Execute Test Case: After the generation of the test case and the test data, test cases are executed.

Defect Reporting: Defects in the system are detected.

Regression Testing: It is carried out to test the side effects of the testing process.

Log Defects: Defects are fixed in this step.

Retest: If the test is not successful then again the test is performed.

Test Case

In software engineering, a test case is a specification of the inputs, execution conditions, testing procedure, and expected results that define a single test to be executed to achieve a particular software testing objective, such as to exercise a particular program path or to verify compliance with a specific requirement. Test cases underlie testing that is methodical rather than haphazard. A battery of test cases can be built to produce the desired coverage of the software being tested. Formally defined test cases allow the same tests to be run repeatedly against successive versions of the software, allowing for effective and consistent regression testing. A formal written test-case is characterized by a known input and by an expected output, which is worked out before the test is executed. The known input should test a precondition and the expected output should test a post condition. A good test case is one that has a high probability of finding yet undiscovered errors.

REQUEST ID	TEST CASE ID	TEST CASE SCENARIO	STATUS (PASS/FAIL)
SS-01	TC-01	REGISTRATION (SIGN UP)	PASS
SS-02	TC-02	AUTHORIZATION (SIGN IN)	PASS
SS-03	TC-03	PASSWORD MANAGEMENT (ADD, DELETE, UPDATE)	PASS
SS-04	TC-04	VALIDATION (PASSWORD)	PASS

Testing Methodologies

This is the phase where the bug in the programs was to be found and corrected. One of the goals during dynamic testing is to produce a test suite, where the salary calculated with the desired outputs such as reports in this case. This is applied to ensure that the modification of the program does not have any side effects. This type of testing is called regression testing. Testing generally removes all the residual bugs and improves the bugs and improves the reliability of the program.

The basic types of testing are

- Black Box Testing.
- White Box Testing.
- Unit Testing.
- Integration Testing.
- Validation Testing.
- Output Testing.

Black Box Testing: Black box testing methods focus on the functional requirements of the software i.e. black box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. It maintains the integrity of external information. Specific knowledge of the application's code, internal structure and programming knowledge in general is not required.[2] The tester is aware of what the software is supposed to do but is not aware of how it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of how the software produces the output in the first place. Black box testing attempts to find errors in the following categories.

- Incorrect or missing functions.
- Interface errors.
- Errors in data structures or external database access.
- Performance errors or initialization and termination error.

White box Testing: White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of software testing that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the expected outputs. White box testing of software is predicted on a close examination of procedural detail. Logical paths through the software are tested by providing test cases that exercise specific sets of conditions and/or loops. The "status of the program" may be examined at various points to determine if the expected or asserted status corresponds to the actual status. Using white box testing methods, the software engineer can derive test cases that:

- Guarantee that all independent paths within a module have been exercised at least once.
- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to ensure their validity.

Unit Testing: Unit testing is a way of testing the smallest piece of code referred to as a unit that can be logically isolated in a system. It is mainly focused on the functional correctness of standalone modules. A unit can be almost anything you want it to be – a specific piece of functionality, a program, or a particular method within the application. Smaller the unit, the better it is. Smaller tests usually give you a much more granular view of your production, the code is performing. Also, your tests can run faster if they are small. So, it is also a micro-level of software testing**.** This is the first level of testing. In this different modules are tested against the specifications produced during the design of the modules. Unit testing is done for the verification of the code produced during the coding of a single program module in an isolated environment.

Unit testing first focuses on the modules independently of one another to locate errors. After coding each dialogue is tested and run individually. All unnecessary coding was removed and it was ensured that all the modules worked, as the programmer would expect. Logical errors found were corrected. So, by working all the modules independently and verifying the outputs of each module in the presence of staff was concluded that the program was functioning as expected.

Benefits of Unit Testing

- Makes Coding Agile
- Helps Find Software Bugs Early
- Provides Documentation
- Debugging is Made Easier

Integration Testing: Data can be lost accessing an interface, one module can have an adverse effect on another sub function when combined, may not produce the desired major functions. Integration testing is a systematic testing for constructing the program structure, while at the same time conducting tests to uncover errors associated within the interface. The objectives are to take unit tests as a whole. Here correction is difficult because the vast expenses of the entire program complicate the isolation of causes. Thus in the integration testing step, all the errors uncovered are corrected for the next testing steps.

7. CONCLUSION & FUTURE ENHANCEMENT

CONCLUSION

The project on achievement provides the doctors with an application that offers a more flexible environment and performs functions with lesser time compared to existing systems. The proposed system is capable of running for a long period without major modification. Since the system has been generated by using and there are chances for reusability of codes and also its present features can be enhanced by some simple modifications in the codes so as to reuse it in the changing scenario. It will help to automate the day to day activities of a user. By reducing manual effort, time required to process the details, reduced manual errors proposed system will help to run faster, efficiently and cost effectively automate the systems.

FUTURE ENHANCEMENT

Futuristic requirements are also listed along with the functional spec. In the future this application can be used everywhere. Syncing up the passwords will help users to access it from anywhere in the world. New updates and features can be easily implemented in this application.

8.BIBLIOGRAPHY & REFERENCES

- IntelliJ IDEA Community - https://www.jetbrains.com/idea/
- MySQL Server Community - https://www.mysql.com/
- JavaFX - https://openjfx.io/
- Scene Builder - https://gluonhq.com/products/scene-builder/
- MySQL Workbench - https://www.mysql.com/products/workbench/

9. SYSTEM CODE

Main.java

```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.stage.Stage;
import javafx.stage.StageStyle;

public class Main extends Application {

        @Override
        public void start(Stage primaryStage) throws Exception{
        primaryStage.initStyle(StageStyle.UNIFIED);
        Parent root = FXMLLoader.load(getClass().getResource("login.fxml"));
        primaryStage.getIcons().add(new
Image(getClass().getResourceAsStream("titlelogo.png")));
        primaryStage.setTitle("Z - Password Manager");
        primaryStage.setScene(new Scene(root, 600, 400));
        primaryStage.setResizable(false);
        root.requestFocus();
        primaryStage.show();
        }

        public static void main(String[] args) {
        launch(args);
        }
}
```

homeDBConnection.java

```
package sample;

import java.sql.Connection;
import java.sql.DriverManager;

public class homeDBConnection {
        public Connection databaselink;

        public Connection getConnection() {
        String databaseName = "zpasswordmanager";
        String databaseUsername = "root";
        String databasePassword = "root";
        String databaseUrl = "jdbc:mysql://localhost:3306/" + databaseName;

        try {
        databaselink = DriverManager.getConnection(databaseUrl,
databaseUsername, databasePassword);
        }catch (Exception e) {
        e.printStackTrace();
        }
        return databaselink;
        }
}
```

homeDBController.java

```
package sample;

import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Stage;

import java.io.IOException;
import java.net.URL;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Optional;
import java.util.ResourceBundle;

public class homeDBController implements Initializable {

        int var, checkId;
        Controller obj = new Controller();
        homeDBConnection connectnow = new homeDBConnection();

        @FXML
        Button addData, signOutButton, exitButton, editDataButton,
removeDataButton, accButton;

        @FXML
        private TableView<homeDatas> homeTable;
        @FXML
        private TableColumn<homeDatas, Integer> id;

        @FXML
        private TableColumn<homeDatas, String> tusername;

        @FXML
        private TableColumn<homeDatas, String> email;

        @FXML
        private TableColumn<homeDatas, String> tpassword;

        @FXML
        private TableColumn<homeDatas, String> url;

        @FXML
        private TableColumn<homeDatas, String> remark;

        ObservableList<homeDatas> list = FXCollections.observableArrayList();

        @Override
        public void initialize(URL location, ResourceBundle resources) {
        String dbName = obj.getDBName();
        String query = "SELECT * FROM " + dbName;
        try {
        Connection connectdb = connectnow.getConnection();
        ResultSet resultSet = connectdb.createStatement().executeQuery(query);
        while (resultSet.next()) {
                list.add(new homeDatas(resultSet.getInt("id"),
resultSet.getString("username"),
                resultSet.getString("email"), resultSet.getString("password"),
                resultSet.getString("url"),
                resultSet.getString("remark")));
        }

        } catch (SQLException throwables) {
        throwables.printStackTrace();
        }
```

```
id.setCellValueFactory(new PropertyValueFactory<homeDatas, Integer>("id"));
        tusername.setCellValueFactory(new PropertyValueFactory<homeDatas,
String>("username"));
        email.setCellValueFactory(new PropertyValueFactory<homeDatas,
String>("email"));
        tpassword.setCellValueFactory(new PropertyValueFactory<homeDatas,
String>("password"));
        url.setCellValueFactory(new PropertyValueFactory<homeDatas,
String>("url"));
        remark.setCellValueFactory(new PropertyValueFactory<homeDatas,
String>("remark"));
        homeTable.setItems(list);
        }

        public void goToAddData() throws IOException {
        Parent root = FXMLLoader.load(getClass().getResource("addData.fxml"));
        Stage signUp = (Stage) addData.getScene().getWindow();
        signUp.setScene(new Scene(root, 600, 400));
        root.requestFocus();
        }

        public void handleExit() {
        Alert alertmsg = new Alert(Alert.AlertType.CONFIRMATION);
        alertmsg.setTitle("Exit");
        alertmsg.setHeaderText("Are you sure?");
        alertmsg.setContentText("Do you want to exit?");
        Optional<ButtonType> result = alertmsg.showAndWait();
        ButtonType button = result.orElse(ButtonType.CANCEL);
        if(button == ButtonType.OK) {
        Platform.exit();
        }
        }

        public void goToLoginPage() throws IOException {
        Alert alertmsg = new Alert(Alert.AlertType.CONFIRMATION);
        alertmsg.setTitle("Exit");
        alertmsg.setHeaderText("Are you sure?");
        alertmsg.setContentText("Do you want to signout?");
```

```
Optional<ButtonType> result = alertmsg.showAndWait();
        ButtonType button = result.orElse(ButtonType.CANCEL);
if(button == ButtonType.OK) {
        Parent root = FXMLLoader.load(getClass().getResource("login.fxml"));
        Stage signUp = (Stage) exitButton.getScene().getWindow();
        signUp.setScene(new Scene(root, 600, 400));
        root.requestFocus();
}
        }

        public void goToEditData() throws IOException {
        Boolean isNumeric = true;
        TextInputDialog dialog = new TextInputDialog();
        dialog.setTitle("Edit Data");
        dialog.setHeaderText("ID Confirmation");
        dialog.setContentText("Please enter the ID:");
        Optional<String> result = dialog.showAndWait();
        try {
        var = Integer.parseInt(result.get());
        obj.setVariable(var);
        }catch (NumberFormatException e) {
        isNumeric = false;
        }

        if (!isNumeric) {
        Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
        alertmsg.setTitle("Error");
        alertmsg.setHeaderText("ID Is a Numeric Value");
        alertmsg.setContentText("Please Enter a Valid ID");
        alertmsg.showAndWait();
        }
        else {
        Boolean test = checkData();
        if (test) {
                displayEditPage();
        }
else {
            Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
            alertmsg.setTitle("Error");
        alertmsg.setHeaderText("Wrong ID");
            alertmsg.setContentText("Please enter a valid ID");
            alertmsg.showAndWait();
        }
        }
        }

        public boolean checkData() {
        Controller ctrl = new Controller();
        Connection databaselink = connectnow.getConnection();
        String idCheckQuery = "SELECT ID FROM " + ctrl.currentUserDB + "
WHERE ID = " + var;
        try {
        Statement statement = databaselink.createStatement();
        ResultSet resultSet = statement.executeQuery(idCheckQuery);
        while (resultSet.next()) {
            checkId = resultSet.getInt(1);
            if (checkId == var) {
            return true;
            }
            else {
            return false;
            }
        }
        }catch (Exception e) {
        e.printStackTrace();
        }
        return false;
        }

        public void displayEditPage() throws IOException {
        Parent root = FXMLLoader.load(getClass().getResource("editData.fxml"));
        Stage signUp = (Stage) editDataButton.getScene().getWindow();
        signUp.setScene(new Scene(root, 600, 400));
        root.requestFocus();
        }
```

```
public void goToRemoveData() throws IOException{
        Boolean isNumeric = true;
        TextInputDialog dialog = new TextInputDialog();
        dialog.setTitle("Remove Data");
        dialog.setHeaderText("ID Confirmation");
        dialog.setContentText("Please enter the ID:");
        Optional<String> result = dialog.showAndWait();
        try {
        var = Integer.parseInt(result.get());
        obj.setVariable(var);
        }catch (NumberFormatException e) {
        isNumeric = false;
        }

        if (!isNumeric) {
        Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
        alertmsg.setTitle("Error");
        alertmsg.setHeaderText("ID Is a Numeric Value");
        alertmsg.setContentText("Please Enter a Valid ID");
        alertmsg.showAndWait();
        }
        else {
        Boolean test = checkData();
        if (test) {
                displayRemovePage();
        }
        else {
                Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
                alertmsg.setTitle("Error");
                alertmsg.setHeaderText("Wrong ID");
                alertmsg.setContentText("Please enter a valid ID");
                alertmsg.showAndWait();
        }
        }
        }

        public void displayRemovePage() throws IOException {
        Parent root =
FXMLLoader.load(getClass().getResource("removeData.fxml"));
```

```
        Stage signUp = (Stage) removeDataButton.getScene().getWindow();
        signUp.setScene(new Scene(root, 600, 400));
        root.requestFocus();
        }

        public void goToAccount() throws IOException {
        Parent root = FXMLLoader.load(getClass().getResource("account.fxml"));
        Stage signUp = (Stage) accButton.getScene().getWindow();
        signUp.setScene(new Scene(root, 600, 400));
        root.requestFocus();
        }
}
```

Controller.java

```
package sample;

import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Optional;

public class Controller{

        private static int var, check, userID;
        public static String currentUserDB;

        String loginUsername, loginPassword;
        String newNameUser, newEmailId, newUserName, newPassword,
newRetypePass, newUrl, newRemark;

        Encrypt_Decrypt ed = new Encrypt_Decrypt();
        homeDBConnection connectNow = new homeDBConnection();
        Connection databaselink = connectNow.getConnection();

        @FXML
        Button signinButton, signupButton, resetButton, signUpButton,
        cancelButton, retrieveButton, editButton, editUserButton, secQuesButton;
```

```
@FXML
        TextField uname, newName, newEmail, newUser, newPass,
reTypeNewPass, addName, addEmail, addUser, addPass, addRetypePass,
        addUrl, addRemark, oldEmail, oldUsername, oldUrl, oldRemark,
        newURL, newREMARK, newUsername, remEmail, remUser, remUrl,
remRemark,
        currName, currEmail, currUser, currPass, remPass, oldPass;

        @FXML
        PasswordField pword, oldRetypePass, newPassWord, newReTypePass,
remRetypePass;

        public void handleSignIn() {
        loginUsername = uname.getText();
        loginPassword = pword.getText();
        if(loginUsername.isEmpty() || loginPassword.isEmpty()) {
        Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
        alertmsg.setTitle("Missing Information");
        alertmsg.setHeaderText("Please enter valid details");
        alertmsg.setContentText("Fields are mandatory");
        alertmsg.showAndWait();
        }
        else {
        validateLogin(loginUsername, loginPassword);
        }
        }

        public void goToSignUp() throws IOException {
        Parent root = FXMLLoader.load(getClass().getResource("signup.fxml"));
        Stage signUp = (Stage) signupButton.getScene().getWindow();
        signUp.setScene(new Scene(root, 600, 400));
        root.requestFocus();
        }

        public void goToReset() throws IOException {
        Parent root = FXMLLoader.load(getClass().getResource("reset.fxml"));
        Stage reset = (Stage) resetButton.getScene().getWindow();
        reset.setScene(new Scene(root, 600, 400));
        root.requestFocus();
        }
        public void handleSignUp() throws Exception {
        newNameUser = newName.getText();
        newEmailId = newEmail.getText();
        newUserName = newUser.getText();
        newPassword = newPass.getText();
        newRetypePass = reTypeNewPass.getText();
        if (newNameUser.isEmpty() || newEmailId.isEmpty() ||
                newUserName.isEmpty() || newPassword.isEmpty() ||
newRetypePass.isEmpty()) {
        Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
        alertmsg.setTitle("Missing Information");
        alertmsg.setHeaderText("Please fill out the form");
        alertmsg.setContentText("Fields are mandatory");
        alertmsg.showAndWait();
        }
        else if (!newPassword.equals(newRetypePass)) {
        Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
        alertmsg.setTitle("Missing Information");
        alertmsg.setHeaderText("Passwords doesn't match");
        alertmsg.setContentText("Fields are mandatory");
        alertmsg.showAndWait();
        } else {
        String encrypted_pass = ed.encrypt(newPassword);
        validateSignUp(newNameUser, newEmailId, newUserName,
encrypted_pass);
        }
        }

        public void createNewDatabase(String newUserDatabase) {
        String newDBQuery = "CREATE TABLE " + newUserDatabase + "(ID int
not null auto_increment primary key, " +
                " EMAIL varchar(50), " + " USERNAME varchar(50), " + "
PASSWORD varchar(150), " +
                " URL varchar(50)," + " REMARK varchar(50))";
        try {
        Statement statement = databaselink.createStatement();
        statement.executeUpdate(newDBQuery);
        }catch (Exception e) {
        e.printStackTrace();
```

```
}
        }

        public String getDBName() {
        return currentUserDB;
        }

        public void validateLogin(String loginUsername, String loginPassword) {
        String loginQuery = "SELECT * FROM USERDATABASE WHERE
USERNAME = '" + loginUsername + "'";
        String User, Pass;
        Boolean checkValue = false;
        try {
        Statement statement = databaselink.createStatement();
        ResultSet resultSet = statement.executeQuery(loginQuery);
        while (resultSet.next()) {
                userID = resultSet.getInt(1);
                User = resultSet.getString(4);
                Pass = resultSet.getString(5);
                String checkPass = ed.decrypt(Pass);
                if ((User.equals(loginUsername)) &&
checkPass.equals(loginPassword)) {
                checkValue = true;
                currentUserDB = loginUsername;
                }
        }
        if (checkValue == true) {
                Parent root =
FXMLLoader.load(getClass().getResource("home.fxml"));
                Stage logIn = (Stage) signupButton.getScene().getWindow();
                logIn.setScene(new Scene(root, 600, 400));
                root.requestFocus();
        }
        else {
                Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
                alertmsg.setTitle("Error");
                alertmsg.setHeaderText("Sign up required");
                alertmsg.setContentText("User not found");
                alertmsg.showAndWait();
        }
```

```
}catch (Exception e) {
        e.printStackTrace();
        }
        }

        public void validateSignUp(String newNameUser, String newEmailId,
String newUserName, String newPassword) {
        String signUpQuery = "INSERT INTO USERDATABASE(LNAME,
EMAIL, USERNAME, PASSWORD) " +
                "VALUES('" + newNameUser + "','" +newEmailId + "','" +
newUserName + "','" + newPassword + "')";
        try {
        Statement statement = databaselink.createStatement();
        int result = statement.executeUpdate(signUpQuery);
        if (result != 0) {
                createNewDatabase(newUserName);
                Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
                alertmsg.setTitle("Success");
                alertmsg.setHeaderText("Successfully created account");
                alertmsg.setContentText("Sign in to proceed");
                alertmsg.showAndWait();
                check = 1;
        }
        else {
                Alert alertmsg = new Alert(Alert.AlertType.ERROR);
                alertmsg.setTitle("Error");
                alertmsg.setHeaderText("Error occurred");
                alertmsg.setContentText("Try again");
                alertmsg.showAndWait();
        }
        if (check == 1) {
                newName.setText("");
                newEmail.setText("");
                newUser.setText("");
                newPass.setText("");
                reTypeNewPass.setText("");
        }
        }catch (Exception e) {
        e.printStackTrace();
```

```
}
        }

        public void handleAddData() throws Exception {
        newEmailId = addEmail.getText();
        newUserName = addUser.getText();
        newPassword = addPass.getText();
        newRetypePass = addRetypePass.getText();
        newUrl = addUrl.getText();
        newRemark = addRemark.getText();
        if (newEmailId.isEmpty() || newUserName.isEmpty()
                || newPassword.isEmpty() || newRetypePass.isEmpty() ||
newUrl.isEmpty() || newRemark.isEmpty()) {
        Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
        alertmsg.setTitle("Missing Information");
        alertmsg.setHeaderText("Please fill out the form");
        alertmsg.setContentText("Fields are mandatory");
        alertmsg.showAndWait();
        }
        else if (!newPassword.equals(newRetypePass)) {
        Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
        alertmsg.setTitle("Missing Information");
        alertmsg.setHeaderText("Passwords doesn't match");
        alertmsg.setContentText("Fields are mandatory");
        alertmsg.showAndWait();
        } else {
        String encrypt_pass = ed.encrypt(newPassword);
        validateAddData(newEmailId, newUserName, encrypt_pass, newUrl,
newRemark);
        }
        }

        private void validateAddData(String newEmailId, String newUserName,
String newPassword, String newUrl, String newRemark) throws IOException{
        String signUpQuery = "INSERT INTO "+  currentUserDB + "(EMAIL,
USERNAME, PASSWORD, URL, REMARK) " +
                "VALUES('" + newEmailId + "','" + newUserName + "','" +
newPassword + "','" +
                newUrl + "','" + newRemark + "')";
```

```
try {
        Statement statement = databaselink.createStatement();
        int result = statement.executeUpdate(signUpQuery);
        idIncrement();
        if (result != 0) {
                Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
                alertmsg.setTitle("Success");
                alertmsg.setHeaderText("Successfully updated Data");
                alertmsg.setContentText("Proceed to homepage");
                alertmsg.showAndWait();
                check = 1;
        }
        else {
                Alert alertmsg = new Alert(Alert.AlertType.ERROR);
                alertmsg.setTitle("Error");
                alertmsg.setHeaderText("Error occurred");
                alertmsg.setContentText("Try again");
                alertmsg.showAndWait();
        }
        if (check == 1) {
                addEmail.setText("");
                addUser.setText("");
                addPass.setText("");
                addRetypePass.setText("");
                addUrl.setText("");
                addRemark.setText("");
        }
        }catch (Exception e) {
        e.printStackTrace();
        }
        }

        public void handleExit() {
        Alert alertmsg = new Alert(Alert.AlertType.CONFIRMATION);
        alertmsg.setTitle("Exit");
        alertmsg.setHeaderText("Are you sure?");
        alertmsg.setContentText("Do you want to exit?");
        Optional<ButtonType> result = alertmsg.showAndWait();
        ButtonType button = result.orElse(ButtonType.CANCEL);
```

```
if(button == ButtonType.OK) {
        Platform.exit();
        }
        }

        public void goToSignIn() throws IOException {
        Alert alertmsg = new Alert(Alert.AlertType.CONFIRMATION);
        alertmsg.setTitle("Exit");
        alertmsg.setHeaderText("Are you sure?");
        alertmsg.setContentText("Do you wanna leave?");
        Optional<ButtonType> result = alertmsg.showAndWait();
        ButtonType button = result.orElse(ButtonType.CANCEL);
        if(button == ButtonType.OK) {
        Parent root = FXMLLoader.load(getClass().getResource("login.fxml"));
        Stage signUp = (Stage) signinButton.getScene().getWindow();
        signUp.setScene(new Scene(root, 600, 400));
        root.requestFocus();
        }
        }

        public void goToHome() throws IOException {
        Parent root = FXMLLoader.load(getClass().getResource("home.fxml"));
        Stage signUp = (Stage) cancelButton.getScene().getWindow();
        signUp.setScene(new Scene(root, 600, 400));
        root.requestFocus();
        }

        public void goToEditPage() throws Exception {
        String query = "SELECT * FROM " + currentUserDB + " WHERE ID = '" +
var + "'";
        try {
        ResultSet resultSet = databaselink.createStatement().executeQuery(query);
        while (resultSet.next()) {
                String pass = resultSet.getString(4);
                String decypt_pass = ed.decrypt(pass);
                oldEmail.setText(resultSet.getString(2));
                oldUsername.setText(resultSet.getString(3));
                oldPass.setText(decypt_pass);
                oldRetypePass.setText(decypt_pass);
oldUrl.setText(resultSet.getString(5));
                oldRemark.setText(resultSet.getString(6));
        }
        } catch (SQLException throwables) {
        throwables.printStackTrace();
        }
        }

        public static void setVariable(int x) {
        var = x;
        }

        public void handleEditData() throws Exception {
        newEmailId = newEmail.getText();
        newUserName = newUsername.getText();
        newPassword = newPassWord.getText();
        newRetypePass = newReTypePass.getText();
        newUrl = newURL.getText();
        newRemark = newREMARK.getText();
        if (newEmailId.isEmpty() || newUserName.isEmpty()
                || newPassword.isEmpty() || newRetypePass.isEmpty() ||
newUrl.isEmpty() || newRemark.isEmpty()) {
        Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
        alertmsg.setTitle("Missing Information");
        alertmsg.setHeaderText("Please fill out the form");
        alertmsg.setContentText("Fields are mandatory");
        alertmsg.showAndWait();
        }
        else if (!newPassword.equals(newRetypePass)) {
        Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
        alertmsg.setTitle("Missing Information");
        alertmsg.setHeaderText("Passwords doesn't match");
        alertmsg.setContentText("Fields are mandatory");
        alertmsg.showAndWait();
        } else {
        String encrypt_pass = ed.encrypt(newPassword);
        validateEditData(newEmailId, newUserName, encrypt_pass, newUrl,
newRemark);
```

```
}
        }

        public void validateEditData(String newEmailId, String newUserName,
String newPassword, String newUrl, String newRemark) {
        String editDataQuery = "UPDATE " + currentUserDB + " SET EMAIL = '"
+ newEmailId + "', " +
                "USERNAME = '" + newUserName + "', PASSWORD = '" +
newPassword + "', " +
                "URL = '" + newUrl + "', REMARK = '" + newRemark + "'  where ID
= '" + var + "'";
        try {
        Statement statement = databaselink.createStatement();
        int result = statement.executeUpdate(editDataQuery);
        if (result != 0) {
                Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
                alertmsg.setTitle("Success");
                alertmsg.setHeaderText("Successfully updated data");
                alertmsg.setContentText("Proceed to homepage");
                alertmsg.showAndWait();
                check = 1;
                goToHome();
        }
        else {
                Alert alertmsg = new Alert(Alert.AlertType.ERROR);
                alertmsg.setTitle("Error");
                alertmsg.setHeaderText("Error occurred.\nCrosscheck ID");
                alertmsg.setContentText("Try again");
                alertmsg.showAndWait();
        }
        if (check == 1) {
                newEmail.setText("");
                newUsername.setText("");
                newPassWord.setText("");
                newReTypePass.setText("");
                newURL.setText("");
                newREMARK.setText("");
        }
        }catch (Exception e) {
```

```
e.printStackTrace();
        }
        }

        public void validateRemoveData() {
        Boolean set = false;
        String removeDataQuery = "DELETE FROM " + currentUserDB + "
WHERE ID = '" + var + "'";
        try {
        Statement statement = databaselink.createStatement();
        int result = statement.executeUpdate(removeDataQuery);
        idIncrement();
        if (result != 0) {
                set = true;
                Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
                alertmsg.setTitle("Success");
                alertmsg.setHeaderText("Successfully removed Data");
                alertmsg.setContentText("Proceed to homepage");
                alertmsg.showAndWait();
                goToHome();
        }
        else {
                Alert alertmsg = new Alert(Alert.AlertType.ERROR);
                alertmsg.setTitle("Error");
                alertmsg.setHeaderText("Error occurred.\nCrosscheck ID");
                alertmsg.setContentText("Try again");
                alertmsg.showAndWait();
        }
        if (set == true) {
                homeDBController hDBC = new homeDBController();
        }
        }catch (Exception e) {
        e.printStackTrace();
        }
        }

        public void idIncrement() {
        String idIncrementQuery = "ALTER TABLE " + currentUserDB + "
AUTO_INCREMENT = 1";
```

```
try {
        Statement statement = databaselink.createStatement();
        statement.executeUpdate(idIncrementQuery);
        }catch (Exception e) {
        e.printStackTrace();
        }
        }

        public void RetrieveRemoveData() throws Exception {
        String query = "SELECT * FROM " + currentUserDB + " WHERE ID = '" +
var + "'";
        try {
        Connection connectdb = connectNow.getConnection();
        ResultSet resultSet = connectdb.createStatement().executeQuery(query);
        while (resultSet.next()) {
                String pass = resultSet.getString(4);
                String decrypt_pass = ed.decrypt(pass);
                remEmail.setText(resultSet.getString(2));
                remUser.setText(resultSet.getString(3));
                remPass.setText(decrypt_pass);
                remRetypePass.setText(decrypt_pass);
                remUrl.setText(resultSet.getString(5));
                remRemark.setText(resultSet.getString(6));
        }
        } catch (SQLException throwables) {
        throwables.printStackTrace();
        }
        }

        public void goToEditUser() throws IOException {
        Parent root =
FXMLLoader.load(getClass().getResource("editUserData.fxml"));
        Stage signUp = (Stage) editUserButton.getScene().getWindow();
        signUp.setScene(new Scene(root, 600, 400));
        root.requestFocus();
        }

        public void validateEditUserData(String newnameuser, String newemailid,
String newusername, String newpassword)
```

```
{
        String editUserDataQuery = "UPDATE USERDATABASE SET LNAME =
'" + newnameuser +
                "', EMAIL = '" + newemailid + "', USERNAME = '" + newusername
+ "', PASSWORD = '" + newpassword +
                "' WHERE ID = " + userID;
        String dbRenameQuery = "RENAME TABLE " + currentUserDB + " TO "
+ newusername;

        try {
        Statement statement = databaselink.createStatement();
        int result = statement.executeUpdate(editUserDataQuery);
        if (result != 0) {
                Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
                alertmsg.setTitle("Success");
                alertmsg.setHeaderText("Successfully updated data");
                alertmsg.setContentText("Proceed to homepage");
                alertmsg.showAndWait();
                goToHome();
        }
        else {
                Alert alertmsg = new Alert(Alert.AlertType.ERROR);
                alertmsg.setTitle("Error");
                alertmsg.setHeaderText("Error occurred.");
                alertmsg.setContentText("Try again");
                alertmsg.showAndWait();
        }
        statement.executeUpdate(dbRenameQuery);
        }catch (Exception e) {
        e.printStackTrace();
        }
        }

        public void editUserData() throws Exception {
        newNameUser = addName.getText();
        newEmailId = addEmail.getText();
        newUserName = addUser.getText();
        newPassword = addPass.getText();
        newRetypePass = addRetypePass.getText();
```

```
if (newNameUser.isEmpty() || newEmailId.isEmpty() || newUserName.isEmpty()
                || newPassword.isEmpty() || newRetypePass.isEmpty()) {
        Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
        alertmsg.setTitle("Missing Information");
        alertmsg.setHeaderText("Please fill out the form");
        alertmsg.setContentText("Fields are mandatory");
        alertmsg.showAndWait();
        }
        else if (!newPassword.equals(newRetypePass)) {
        Alert alertmsg = new Alert(Alert.AlertType.INFORMATION);
        alertmsg.setTitle("Missing Information");
        alertmsg.setHeaderText("Passwords doesn't match");
        alertmsg.setContentText("Fields are mandatory");
        alertmsg.showAndWait();
        } else {
        String encrypt_pass = ed.encrypt(newPassword);
        validateEditUserData(newNameUser, newEmailId, newUserName,
encrypt_pass);
        }
        }

        public void retrieveUserData() throws Exception {
        String retrieveUserQuery = "SELECT * FROM USERDATABASE WHERE
USERNAME = '" + currentUserDB + "'";
        try {
        Connection connectdb = connectNow.getConnection();
        ResultSet resultSet =
connectdb.createStatement().executeQuery(retrieveUserQuery);
        while (resultSet.next()) {
                String pass = resultSet.getString(5);
                String decrypt_pass = ed.decrypt(pass);
                currName.setText(resultSet.getString(2));
                currEmail.setText(resultSet.getString(3));
                currUser.setText(resultSet.getString(4));
                currPass.setText(decrypt_pass);
        }
        } catch (SQLException throwables) {
        throwables.printStackTrace();
        }
```

```
}


        public void setSecQues() throws Exception {
        Parent root =
FXMLLoader.load(getClass().getResource("securityquestion.fxml"));
        Stage signUp = (Stage) secQuesButton.getScene().getWindow();
        signUp.setScene(new Scene(root, 600, 400));
        root.requestFocus();
        }
}
```

homeDatas.java

```
package sample;

public class homeDatas {
	private int id;
	private String username, email, password, url, remark;

	Encrypt_Decrypt ed = new Encrypt_Decrypt();

	public homeDatas(int id, String username, String email, String password, String
url, String remark) {
	this.id = id;
	this.username = username;
	this.email = email;
	this.password = password;
	this.url = url;
	this.remark = remark;
	}

	public int getId() {
	return id;
	}

	public String getUsername() {
	return username;
	}

	public String getEmail() {
	return email;
	}

	public String getPassword() {
	String decrypt_pass = null;
	try {
	decrypt_pass = ed.decrypt(password);
	} catch (Exception e) {
	e.printStackTrace();
	}
	return decrypt_pass;
	}
public String getUrl() {
        return url;
        }

        public String getRemark() {
        return remark;
        }
}
```

Encrypt_Decrypt.java

```
package sample;

import javax.crypto.*;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;
import java.nio.ByteBuffer;
import java.security.AlgorithmParameters;
import java.security.SecureRandom;
import java.util.Base64;

public class Encrypt_Decrypt {
        public String encrypt(String Password) throws Exception {
        byte[] ivBytes;
        String password="ZP4ssw0rdM4n4g3r";

        SecureRandom random = new SecureRandom();
        byte bytes[] = new byte[20];
        random.nextBytes(bytes);
        byte[] saltBytes = bytes;

        /*Deriving the key*/
        SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
        PBEKeySpec spec = new
PBEKeySpec(password.toCharArray(),saltBytes,65556,256);
        SecretKey secretKey = factory.generateSecret(spec);
        SecretKeySpec secret = new SecretKeySpec(secretKey.getEncoded(),
"AES");

        /*Encrypting the password*/
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, secret);
        AlgorithmParameters params = cipher.getParameters();
        ivBytes =   params.getParameterSpec(IvParameterSpec.class).getIV();
        byte[] encryptedTextBytes = cipher.doFinal(Password.getBytes("UTF-8"));
```

```
/*Prepend Salt and IV*/
        byte[] buffer = new byte[saltBytes.length + ivBytes.length +
encryptedTextBytes.length];
        System.arraycopy(saltBytes, 0, buffer, 0, saltBytes.length);
        System.arraycopy(ivBytes, 0, buffer, saltBytes.length, ivBytes.length);
        System.arraycopy(encryptedTextBytes, 0, buffer, saltBytes.length +
ivBytes.length, encryptedTextBytes.length);
        String encodeBytes = Base64.getEncoder().encodeToString(buffer);
        return encodeBytes;
        }

        @SuppressWarnings("static-access")
        public String decrypt(String Password) throws Exception {
        String password="ZP4ssw0rdM4n4g3r";
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");

        /*Stripping of Salt and IV*/
        byte[] decodeBytes = Base64.getDecoder().decode(Password);
        ByteBuffer buffer = ByteBuffer.wrap(decodeBytes);
        byte[] saltBytes = new byte[20];
        buffer.get(saltBytes, 0, saltBytes.length);
        byte[] ivBytes1 = new byte[cipher.getBlockSize()];
        buffer.get(ivBytes1, 0, ivBytes1.length);
        byte[] encryptedTextBytes = new byte[buffer.capacity() - saltBytes.length -
ivBytes1.length];
        buffer.get(encryptedTextBytes);

        /*Deriving the key*/
        SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
        PBEKeySpec spec = new PBEKeySpec(password.toCharArray(), saltBytes,
65556, 256);
        SecretKey secretKey = factory.generateSecret(spec);
        SecretKeySpec secret = new SecretKeySpec(secretKey.getEncoded(),
"AES");
        cipher.init(Cipher.DECRYPT_MODE, secret, new
IvParameterSpec(ivBytes1));
        byte[] decryptedTextBytes = null;
        try {
```

```
decryptedTextBytes = cipher.doFinal(encryptedTextBytes);
} catch (IllegalBlockSizeException e) {
        e.printStackTrace();
        } catch (BadPaddingException e) {
        e.printStackTrace();
        }
        return new String(decryptedTextBytes);
        }
}
```

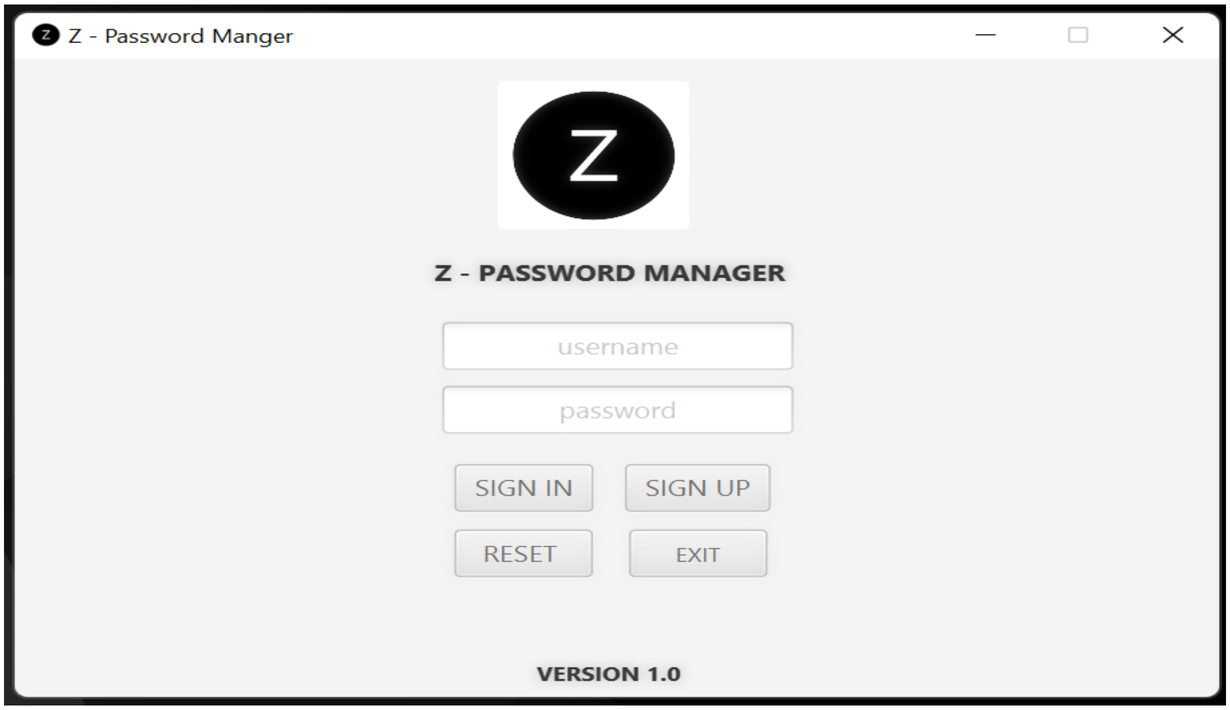
resetController.java

```
package sample;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.stage.Stage;

import java.net.URL;
import java.util.Optional;
import java.util.ResourceBundle;

public class resetController implements Initializable {

        @FXML
        private ChoiceBox ques1, ques2, ques3;

        @FXML
        private Button submitButton, cancelButton;

        @FXML
        private TextField ans1, ans2, ans3;

        ObservableList<String> ques = FXCollections.observableArrayList(
        "Where did you born?",
        "What is your first pet's name?",
        "What is your favourite food?",
        "What is your friend's nickname?",
        "What is your favourite book?"
        );

        @Override
        public void initialize(URL location, ResourceBundle resources) {
```

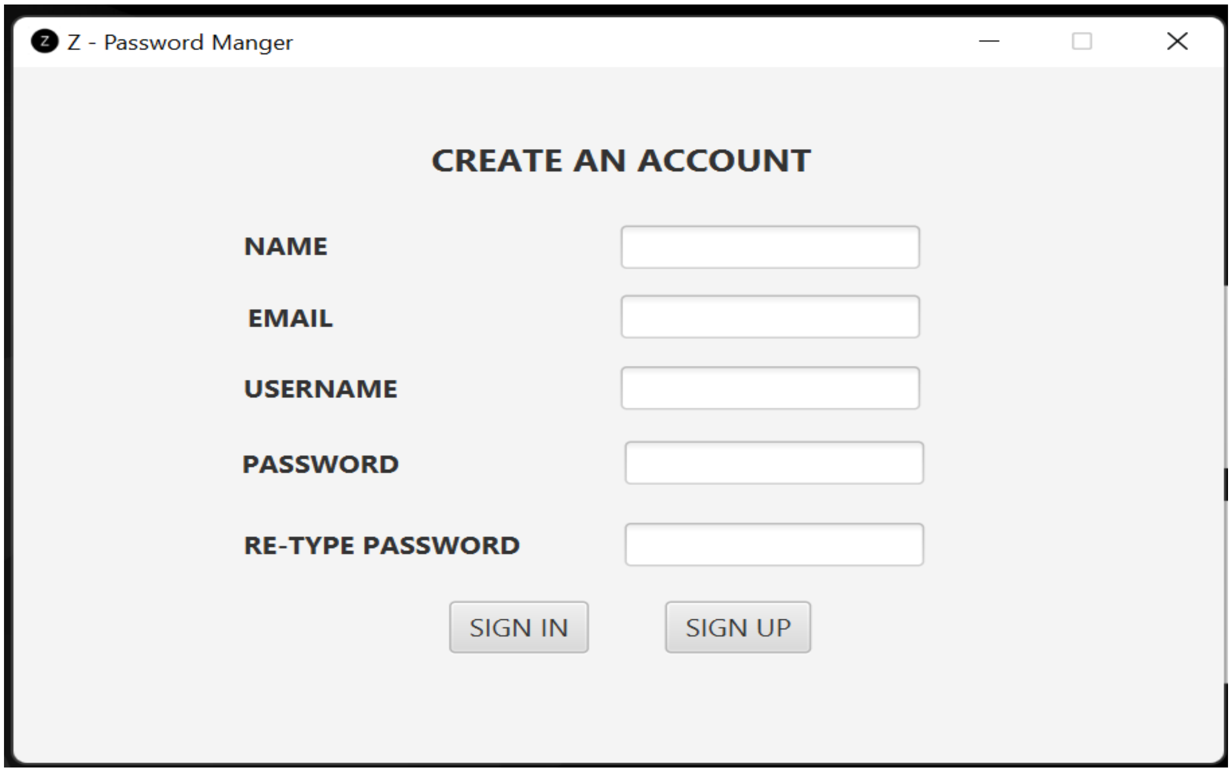
```
        ques1.getItems().addAll(ques);
        ques2.getItems().addAll(ques);
        ques3.getItems().addAll(ques);
        }

        public void goToHome() throws Exception {
        Alert alertmsg = new Alert(Alert.AlertType.CONFIRMATION);
        alertmsg.setTitle("Exit");
        alertmsg.setHeaderText("Are you sure?");
        alertmsg.setContentText("Do you want to exit?");
        Optional<ButtonType> result = alertmsg.showAndWait();
        ButtonType button = result.orElse(ButtonType.CANCEL);
        if(button == ButtonType.OK) {
        Parent root = FXMLLoader.load(getClass().getResource("login.fxml"));
        Stage signUp = (Stage) cancelButton.getScene().getWindow();
        signUp.setScene(new Scene(root, 600, 400));
        root.requestFocus();
        }
        }

        public void submitData() throws Exception {
        String a1 = ans1.getText();
        String a2 = ans2.getText();
        String a3 = ans3.getText();

        Boolean check = checkData(a1, a2, a3);
        if (check) {
        System.out.println("Gotcha !!!");
        }
        else {
        System.out.println("Nothing's gonna happen !!!");
        }
        }

        public Boolean checkData(String a1, String a2, String a3) {
        System.out.println(a1 + " " + a2 + " " + a3);
        return false;
        }
}
```

10. USER MANUAL(Screenshots)

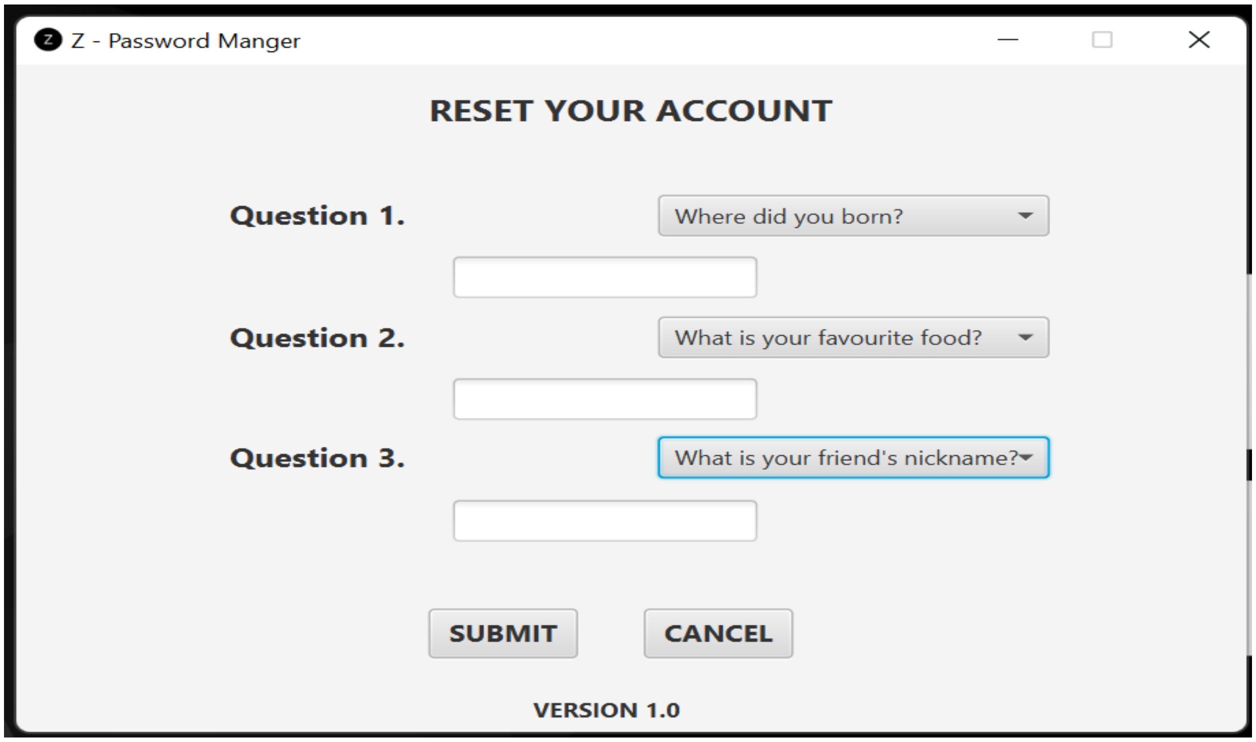
Sign in Page



Sign up Page



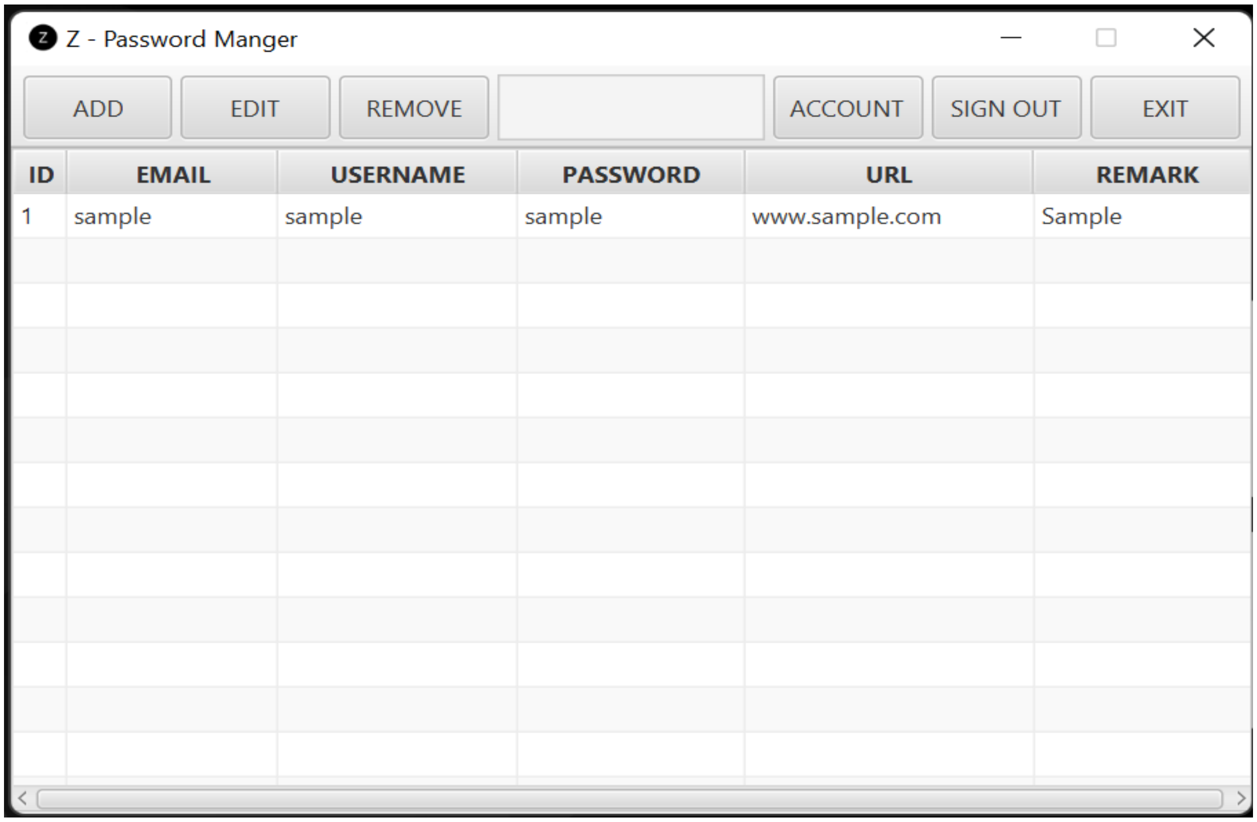
Account Reset Page



RESET YOUR ACCOUNT

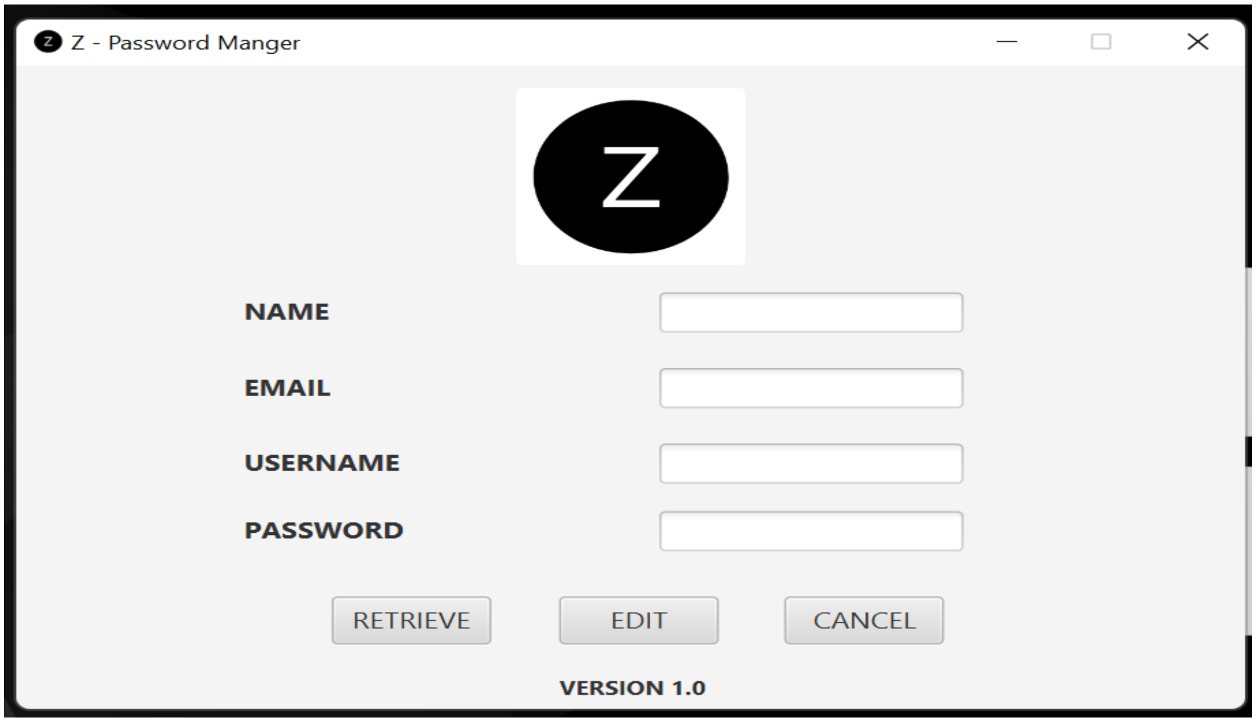
Question 1.	Where did you born?
Question 2.	What is your favourite food?
Question 3.	What is your friend's nickname?

Home Page

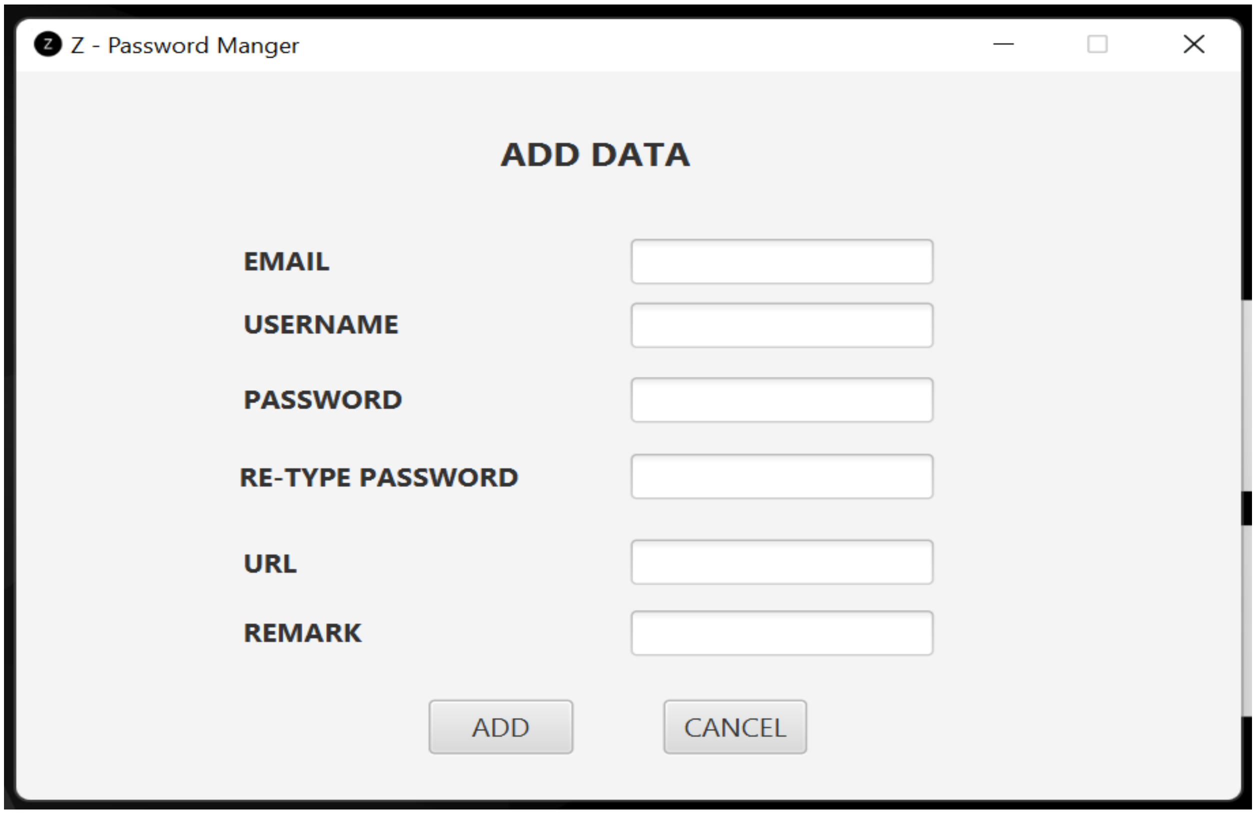


ID	EMAIL	USERNAME	PASSWORD	URL	REMARK
1	sample	sample	sample	www.sample.com	Sample

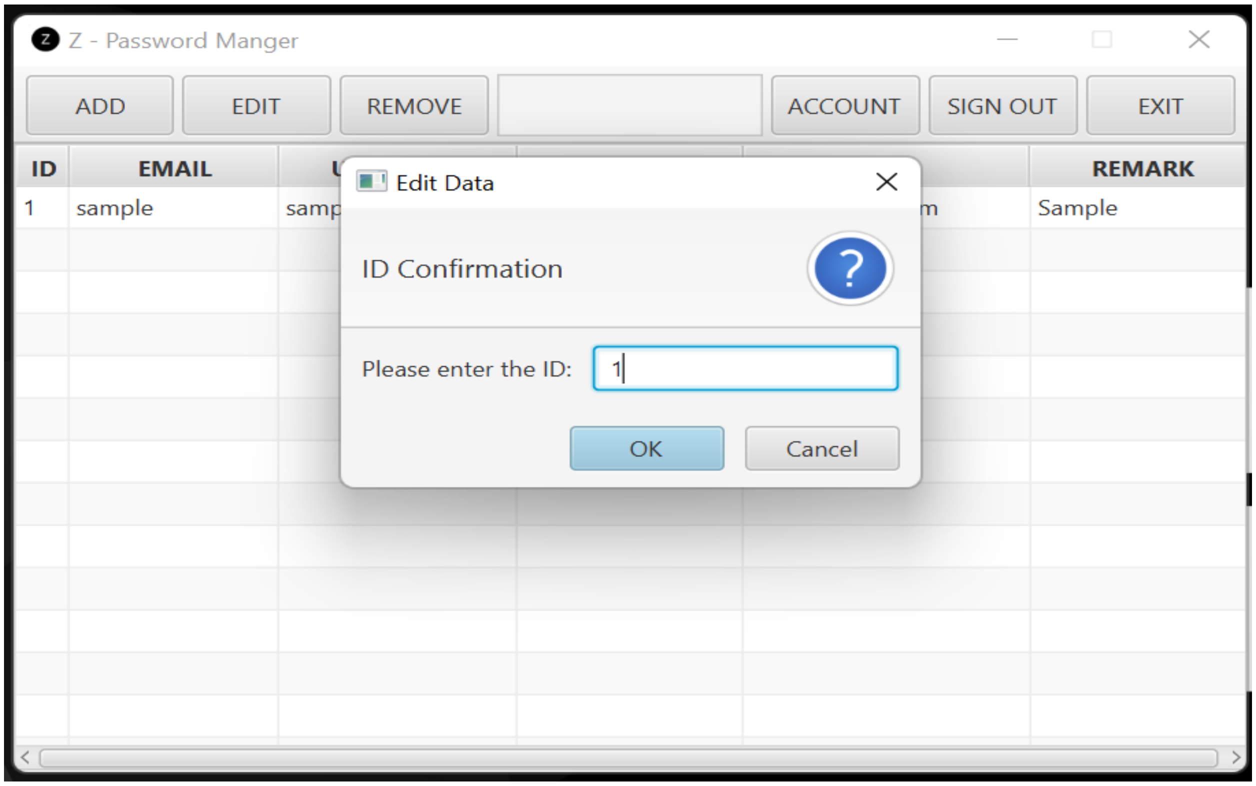
Account Page



Add Data Page



Edit Data Page



Z - Password Manger

ADD | EDIT | REMOVE | ACCOUNT | SIGN OUT | EXIT

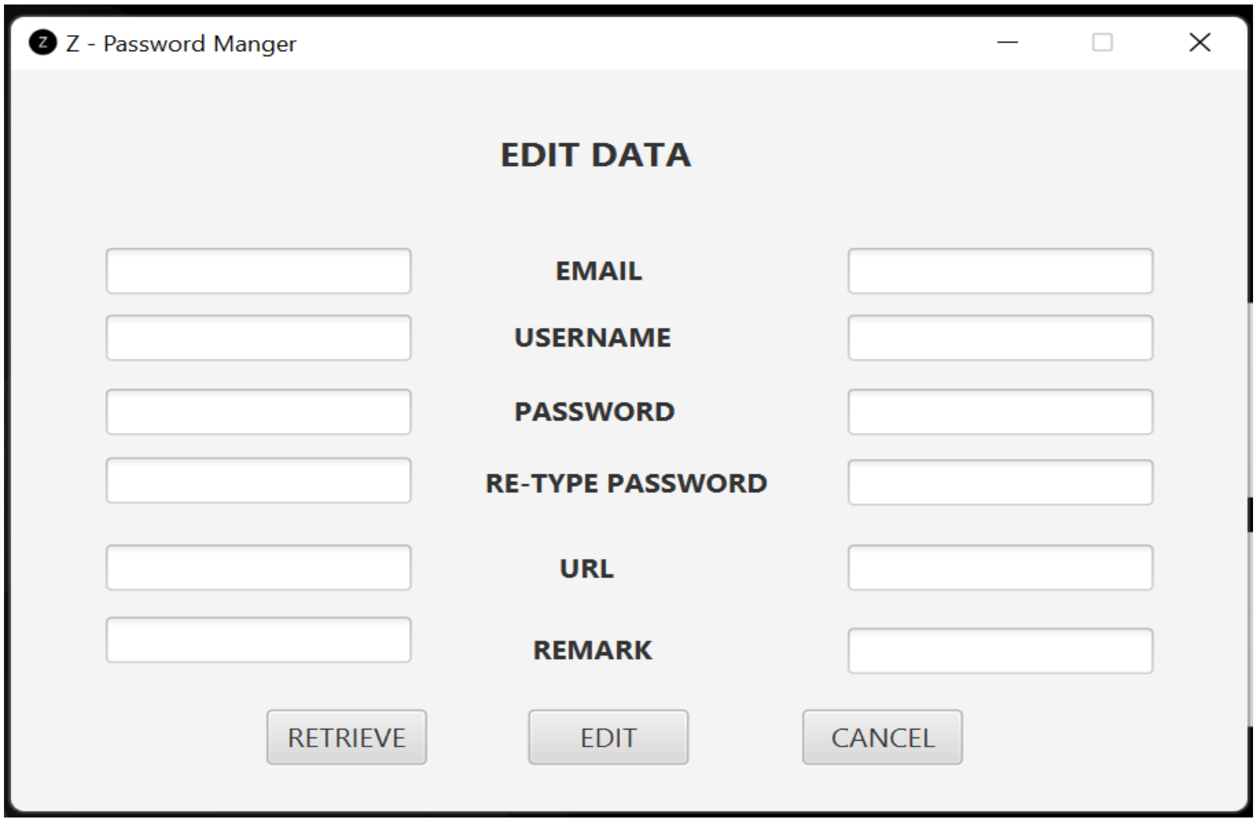
ID	EMAIL	U		REMARK
1	sample	samp	m	Sample

Edit Data

ID Confirmation

Please enter the ID: 1

OK Cancel



Z - Password Manger

EDIT DATA

EMAIL

USERNAME

PASSWORD

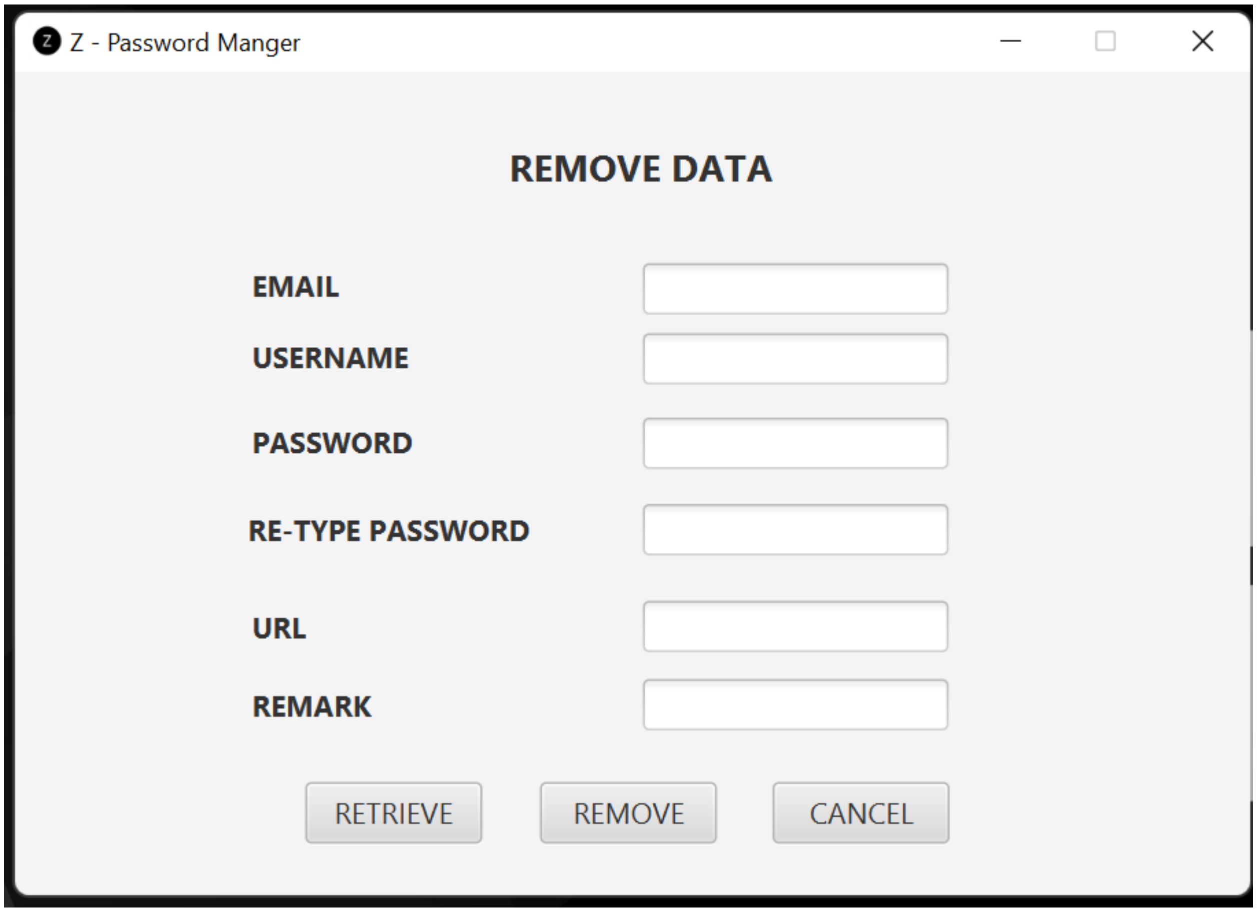
RE-TYPE PASSWORD

URL

REMARK

RETRIEVE EDIT CANCEL

Remove Data Page


Z - Password Manger
REMOVE DATA
EMAIL
USERNAME
PASSWORD
RE-TYPE PASSWORD
URL
REMARK
RETRIEVE
REMOVE
CANCEL