PROJECT REPORT

On

SecurePass Vault

Submitted to Kurukshetra University in partial fulfillment of the requirement for the award of degree of

B. TECH

In

Computer Science & Engineering

Submitted to:
ER. NEHA
(HOD Computer Science)

Submitted By:
Vanshika Rani
Roll No.2021220307

DEPTT. OF COMPUTER SCIENCE AND ENGINEERING

YAMUNA INSTITUTE OF ENGINEERING AND TECHNOLOGY

GADHOLI, YAMUNA NAGAR-135003

May 2025

PREFACE

Training is an integral part of B. Tech and each and every student has to undergo the training for 6 weeks in a company.

This record is concerned about my practical research during the semester study after the Third year. I done my practical Training in Android App Development. During this semester, I got to learn many new things about the industry and current requirements of companies. This training proved to be a milestone in my knowledge of present industry. Every say and every moment was an experience in itself, an experience which theoretical study can’t provide.

ACKNOWLEDGEMENT

It is my pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behaviour and acts during the course of study.

I express my sincere Gratitude to Dr. Ramesh Nain worthy principal for providing me an opportunity to undergo the training at Yamuna Institute of Engineering and Technology.

I am Very Thankful to Mrs. NEHA for her support, cooperation and motivation provided to me during the study for constant inspiration, presence and blessings.

Lastly, I would like to thank the almighty and my parents for their moral support and my friends with whom I shared my day-to-day experience and received lots of suggestions that brings quality to my work.

Vanshika Rani

CANDIDATE'S DECLARATION

I, Vanshika Rani Roll No.2021220307, B. Tech (Semester-VIII) of Yamuna Institute of Engineering and Technology, Gadholi (Yamuna Nagar) hereby declare that the Project Report entitled "**SecurePass Vault**" is an original work and data provided in the study of authentic to the best of my knowledge. This report has not been submitted to any other institute for the award of any other degree.

Vanshika Rani

(Roll No.2021220307)

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER -1

INTRODUCTION: -

Welcome to the **SecurePass Vault** Android application, a robust solution designed to enhance online security and efficient password management. In an era marked by digital connectivity, the importance of strong, unique passwords cannot be overstated. With the rapid increase of online accounts across various platforms, managing passwords securely has become a difficult task for many individuals. SECUREPASS VAULT – a comprehensive solution designed to address these challenges, empowering users to create, store, and manage their passwords seamlessly and securely.

SecurePass Vault: Password Generator and Manager Android App

The "SecurePass Vault" is a **Java** and **XML**-based **Android application** designed to help users generate strong, unique passwords and securely manage them with ease. This mobile app provides a robust solution to address the growing concerns surrounding password security and management, making it convenient and accessible for users on the go.

This mobile app combines two key functionalities: **Password Generation** and **Password Management**, perfectly integrated with a clear and user-friendly interface.

This innovative tool offers two key functionalities: Password Generation and Password management perfectly integrated with a clear and user-friendly interface. The Password Generator feature enables users to create strong, random passwords that are virtually impossible for hackers to crack. By integrating a combination of lowercase alphabets, uppercase alphabets, numbers, and special characters, it ensures maximum security for every online account.

In addition to generating passwords, Password Management system, serves as a Secured Centralized Repository for storing, organizing, and accessing login credentials across diverse digital platforms. Using Firebase as the backend, the app ensures that all user data is stored securely in the cloud. Making Use of advanced encryption techniques, the application ensures the confidentiality and integrity of stored passwords, reducing the risks associated with unauthorized access and data leaks. With the Password Generator and Manager, users can safely store all their passwords in one centralized location, accessible only through a **Master Password.**

By combining robust password generation with seamless password management, it empowers users to safeguard their digital identities and protect sensitive information from unauthorized access in one centralized location, accessible only through a master password.

Key features of the "**SecurePass Vault**" include:

1. **Password Generation:**
   - Empowers users to create strong, random passwords tailored to specific requirements, encouraging a proactive approach toward enhancing internet security.
   - The application utilizes advanced algorithms to generate highly secure passwords consisting of combination of uppercase and lowercase letters, numbers and special characters.
   - Customizable options allow users to modify password length to meet the specific requirements of each online account.
2. **Password Storage Using Firebase:**
   - SecurePass Vault integrates Firebase Realtime Database for storing and syncing data in real time.
   - Each user's passwords are linked to their account and stored securely in the cloud with encryption, ensuring only the account holder can access them.
   - Changes made to stored passwords are synchronized instantly across devices.

3. **Password Management:**
    - Say goodbye to the hassle of remembering multiple passwords or resorting to insecure methods of storage.
    - The password manager provides a centralized repository where users can securely store and access all their login credentials.
    - All stored data is protected with encryption and can only be accessed through a Master Password, ensuring maximum security.
4. **Customization Options:** Offers flexibility in configuring password generation parameters, enabling users to tailor passwords to suit the unique requirements of different websites and applications.
5. **Intuitive User Interface:** The application provides a user-friendly interface that is clear and easy to navigate, designed to enhance usability and simplify the password creation and management process for users with varying levels of technical expertise.
6. **Category Organization:**
    - Users can organize passwords efficiently by categorizing them based on websites, applications, or any other criteria of their choosing.
    - This feature allows for easy navigation and quick retrieval of passwords whenever needed.
7. **Advanced Encryption Techniques:**
    - Ensures the confidentiality and integrity of all stored passwords through encryption, safeguarding sensitive data from unauthorized access or breaches.
    - Users can organize passwords efficiently by categorizing them based on websites, applications, or any other criteria of their choosing.
    - This feature allows for easy navigation and quick retrieval of passwords whenever needed.

Why Choose SecurePass Vault?

The SecurePass Vault Android application is ultimate tool for enhancing online defenses and managing passwords securely. By combining cutting-edge password generation, efficient password management, and the security of Firebase , this app provides a seamless and secure experience.

- **Enhanced Online Security:** Protects against weak passwords, hacking, and phishing attempts.
- **Convenience:** Centralizes password management, eliminating the need to remember multiple credentials.
- **Reliability:** Ensures data safety through encryption and real-time syncing with Firebase.
- **User-Focused Design:** Designed with a clean interface to make password management accessible for everyone.

Let’s embark on this journey together toward a safer and more secure online experience with the SecurePass Vault Android Application !

CHAPTER -2

REQUIREMENT SPECIFICATION: -

2.1 SOFTWARE DEVELOPMENT LIFE CYCLE: -

Figure 2.1: Software Development Life Cycle

The Software Development Life Cycle (SDLC) is a structured framework that outlines the phases and processes involved in developing high-quality software. It serves as a roadmap for software development projects, ensuring that software is built efficiently, with minimal errors, and in accordance with customer requirements. While there are various SDLC models, the core phases and their general objectives remain consistent. Here is an overview of the typical stages in an SDLC:

1. **Planning and Requirements Gathering:** In this initial phase, the project's objectives are defined, and the requirements are gathered. This involves discussions with stakeholders, clients, and end-users to understand their needs and expectations. It includes a project plan, feasibility study, and a detailed requirements document.
2. **Analysis:** Analysis is the initial phase where the focus is on complete understanding and documenting the requirements, objectives, and constraints of a software development project. This phase involves gathering, examining, and documenting information to define what the software needs to achieve, ensuring that it aligns with the needs of stakeholders and the project's goals.
3. **System Design:** The design phase involves creating an architectural blueprint for the software. Designers decide how the system will be structured, specifying components, modules, and interfaces. At this stage, technical decisions, such as database design, programming languages, and hardware requirements, are made.
4. **Software Development:** This phase follows the planning and design stages and is typically characterized by writing the actual code that will make up the software product. Development involves translating the software requirements and design specifications into executable code using programming languages and tools.
5. **Software Testing:** In the testing phase, the software is strictly tested to identify and fix defects. Various testing methods, such as unit testing, integration testing, system testing, and user acceptance testing, are employed. The goal is to ensure that the software meets its requirements and functions correctly.

6. **Implementation (Coding):** This is the phase where the actual code is written. Developers take the design specifications and write the program or code for the software. It is crucial to follow coding standards and best practices to ensure that the code is maintainable and error-free.
7. **Maintenance and Support:** After deployment (involve a phased rollout to end-users or a full release, depending on the project's scale), the software enters the maintenance phase. This involves ongoing support, bug fixes, and updates as needed. Changes and enhancements may be made based on user feedback or evolving requirements.

2.2 SOFTWARE AND HARDWARE REQUIREMENTS: -

2.2.1 SOFTWARE REQUIREMENTS -

- **Operating System:** Windows 11 is used as the operating system due to its stability, advanced features, and user-friendly interface.
- **Programming Language:** JAVA is used for coding the application.
- **Designing:** XML is used for designing the UI of the application.
- **IDE:** Android Studio is used for developing the mobile application, providing an integrated environment for coding, debugging, and UI design.
- **Database:** Firebase is used as the backend database for data storage due to its real-time capabilities and ease of integration with Android Studio.

2.2.2 HARDWARE REQUIREMENTS -

- **Processor:** Intel Core i3 (6th generation or higher) is used as it ensures reliable performance and allows the system to run efficiently for extended periods.
- **RAM:** 4 GB is used to support faster processing and smoother execution of the application.

2.3 PLATFORM USED: -

2.3.1 Front End: JAVA and XML.

2.3.2 Back End: Firebase Database.

CHAPTER -3

DESIGNING: -

3.1 JAVA-THE PROGRAMMING LANGUAGE: -

Java is the primary language for developing Android apps, making it perfect for creating mobile applications like Secure Pass Vault. It is also a high-level, versatile, and widely used programming language known for its portability, readability, and robustness.

JAVA had a significant impact on the world of software development since its introduction in the mid-1990s. Developed by James Gosling and his team at Sun Microsystems (now owned by Oracle Corporation), Java was designed to be platform-independent, meaning it could run on various operating systems without modification. This unique feature, along with its simplicity and robustness, makes Java an excellent choice for developing a variety of applications, from desktop and web applications to Mobile Apps and Enterprise-level software.

One of the key reasons for Java's popularity is its "Write Once, Run Anywhere" principle, which is achieved through the use of the Java Virtual Machine (JVM). This allows developers to write code once and then execute it on different platforms, making Java a powerful tool for cross-platform development and ensuring compatibility for Android devices of all types.

Java's rich ecosystem of libraries, frameworks, and tools enhances its suitability for Android app development. It integrates seamlessly with Android Studio, enabling developers to create efficient, secure, and scalable applications like Secure Pass Vault. Furthermore, its robust features simplify integrating Firebase for real-time database operations, ensuring reliable performance and data security.

Figure 3.1**:** JAVA logo

3.2 JAVA is used for: -

- **Mobile App Development:** The Secure Pass Vault app is developed using Java, leveraging Android Studio for UI design and Firebase for backend data storage.
- **Enterprise Software Development:** Java's scalability allows it to handle large and complex applications, making it suitable for future expansions of the Secure Pass Vault.
- **Web Development:** Java is used to build dynamic and secure web applications with frameworks like Spring and JSP, making it ideal for enterprise-level solutions.
- **Game Development:** Java supports game development by enabling the creation of cross-platform 2D and 3D games. Its ease of use and platform independence make it a popular choice for small-scale game developers.
- **Scientific and Research Computing:** Java is used in scientific and research computing for developing precise, reliable, and scalable applications. Its robustness makes it ideal for complex computational tasks.

3.3 WHY JAVA: -

Java offers several advantages that make it a compelling choice in many scenarios –

- Security: Java's built-in security features, such as encryption libraries, are crucial for securing sensitive data in the Secure Pass Vault app. These features help protect passwords and other personal information, ensuring the app's integrity and user trust.
- Robustness: Java enforces strong type checking and runtime checks, reducing the likelihood of crashes, making it ideal for the Secure Pass Vault app. These safeguards ensure that critical functions, like data storage and password management, run smoothly without errors.
- **Platform Independence**: Java's "write once, run anywhere" capability is a significant advantage for the Secure Pass Vault app, as it ensures compatibility across various Android devices. This allows for seamless deployment and consistent user experience across different platforms.
- Scalability: Java's built-in multithreading support and ability to manage large applications make it ideal for future expansions of the Secure Pass Vault app, particularly as it handles more user data and complex features.
- **Performance**: Java's performance optimizations, including Just-In-Time (JIT) compilation, ensure that the Secure Pass Vault app operates efficiently, providing fast and smooth user experiences even when processing real-time data from Firebase.
- Strong Ecosystem: Java has a rich ecosystem of libraries, tools, and frameworks (like Firebase for backend storage) significantly speeds up the development of the Secure Pass Vault app, helping create secure, scalable, and feature-rich mobile apps.
- **Object-Oriented**: Java's object-oriented nature promotes code reusability, maintainability, and modularity. This is essential for the long-term development of Secure Pass Vault, enabling easy updates, debugging, and future feature additions.
- **Portability**: Java's portability ensures that the Secure Pass Vault app can run on various Android devices, making it accessible to a broader audience without compatibility issues.
- **Large Community**: Java's vast and active developer community provides ample resources, tutorials, and support. This is especially helpful during the development of Secure Pass Vault, where solutions to issues can be quickly found, enabling smoother progress.

3.4 TOP UNKNOWN FACTS ABOUT JAVA: -

- **Popular for Android Development**: Java remains one of the primary programming languages for Android app development, making it a go-to choice for building mobile applications like Secure Pass Vault.
- **Named After Coffee**: Java was originally called "Oak" but was later renamed "Java" after a type of coffee. The name was inspired by the coffee consumed by its creators.
- **Java's Ties to C++**: Java was influenced by C++ in many ways, and its syntax is similar in some aspects. However, Java was designed to be simpler and avoid some of the more error-prone features of C++.
- **Java Doesn't Stand for Anything**: Unlike many programming languages with acronyms, "Java" doesn't stand for anything. It was simply chosen as the name of the language.
- **Java's Checked Exceptions**: Unlike many programming languages, Java has a system of checked exceptions. This means that developers are required to handle or declare exceptions in their code explicitly.
- **WORA (Write Once, Run Anywhere)**: This is one of Java's core principles, emphasizing its platform independence. The ability to write code on one platform and run it on another is a significant part of Java's appeal.
- **Java's "Hello World" Tradition**: The tradition of starting programming tutorials with a "Hello, World!" program is also observed in Java. It's often the first program beginners write to get familiar with the language.

3.5 PROJECT DEVELOPMENT LIFE CYCLE: -

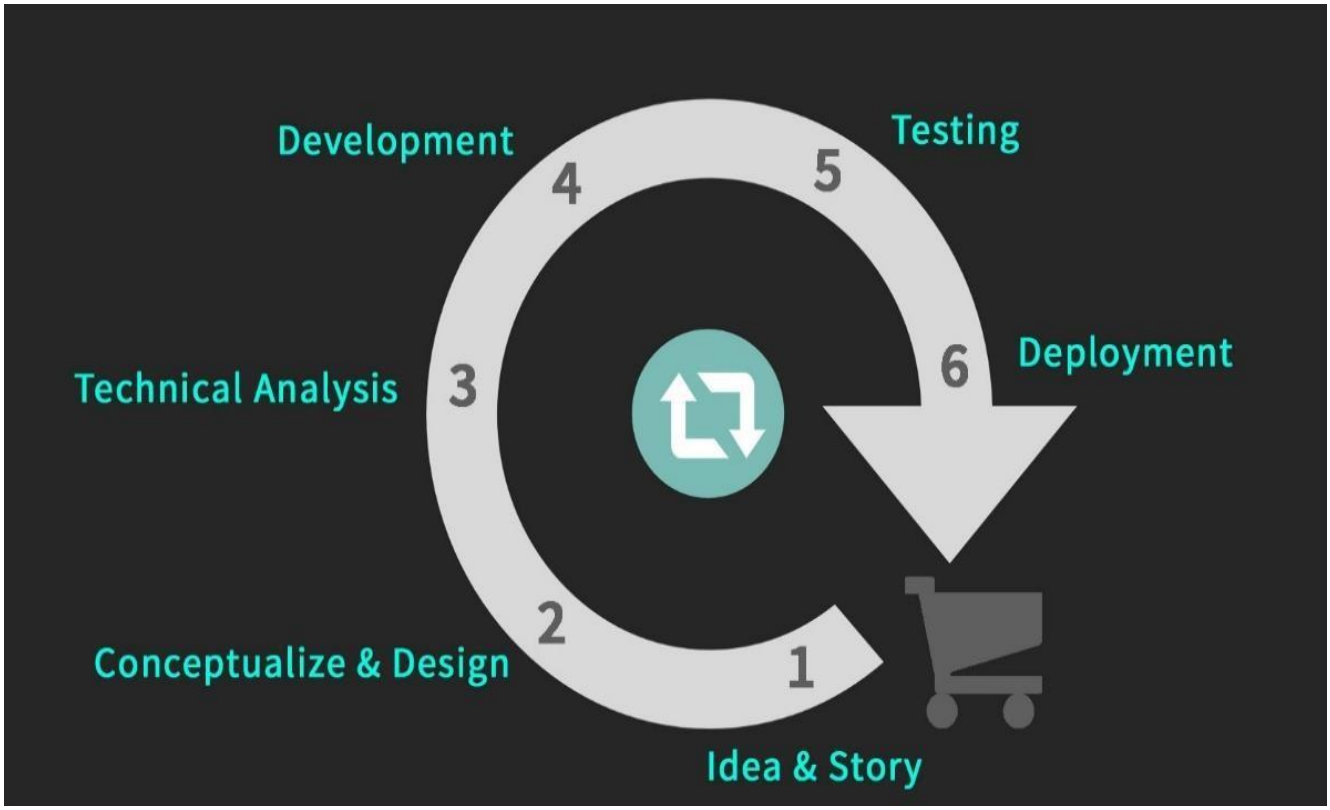


Figure 3.2**:** Project Development life cycle

1. **Idea & Story:** The idea for the Secure Pass Vault app emerged from the need for a secure and user-friendly way to manage passwords and sensitive information. The app aims to simplify password management while ensuring data security, addressing common user concerns about privacy and convenience.
2. **Conceptualization and Design:** In this phase, the concept of Secure Pass Vault was detailed, focusing on features like encrypted password storage, user-friendly interface, and integration with Firebase. The design document outlined the app's layout, flow, and functionalities, creating a clear roadmap for development.
3. **Technical Analysis:** Technical analysis involved evaluating the app's requirements, such as using Firebase for secure data storage, Android Studio for development, and XML for designing the UI. Potential challenges like implementing encryption and ensuring compatibility across devices were also addressed.
4. **Development:** During development, the Secure Pass Vault app was built using Java for coding and XML for UI design. Firebase was integrated to handle backend data storage securely, and features like encrypted password management were implemented.
5. **Testing:** The testing phase involved rigorous checks for bugs, UI issues, and functionality errors. The app was tested to ensure seamless data synchronization with Firebase, robust encryption, and smooth performance across different Android devices.
6. **Deployment:** The Secure Pass Vault app was deployed through app distribution platforms, making it accessible to users. The deployment ensured compatibility across various Android versions and devices, providing a secure and reliable experience.

These stages in the project development life cycle are interconnected, and they collectively guide the process from the initial idea to a fully realized and working project for the client.

3.6 User Interface Design in Secure Pass Vault :-

The **Secure Pass Vault** application uses **XML** for designing its user interface, ensuring a structured, responsive, and visually appealing experience. Below are the key highlights of how XML contributed to the UI design:

1. Declarative Layouts

- **Structured Design**: XML enabled clear definitions of UI components like buttons, text fields, and input forms.
- **Separation of Concerns**: Simplified debugging by keeping UI design separate from Java logic.

2. Layout Choices

- **ConstraintLayout**: Used for responsive designs across various screen sizes.
- **Nested Layouts**: Combined LinearLayout and RelativeLayout for structured alignment in specific sections.

3. Styling and Customization

- **Theming**: Defined app-wide styles, colors, and fonts using styles.xml and colors.xml.
- **Enhanced Aesthetics:** Customized components like EditText and Button with tailored layouts and color schemes to ensure a clean and user-friendly design.

4. Dynamic Interaction

- **ID Assignment**: Components were linked to Java code via unique IDs.
- **Event Handling**: User interactions like button clicks and text changes were processed in Java.

5. Reusability and Modularity

- **Fragments**: Reusable XML layouts were created for forms and navigation bars.
- **Custom Views**: Designed unique components like password strength indicators.

6. Accessibility and Responsiveness

- **Screen Adaptability**: Optimized layouts using scalable units like dp and sp.
- **Dark Mode Support**: Integrated light and dark themes for better user experience.

7. Design Efficiency

- **UI Previews**: Android Studio previews enabled quick design iterations.
- **Localization**: Externalized strings to strings.xml for future multilingual support.

Figure 3.3**:** XML Logo

CHAPTER -4

IMPLEMENTATION AND TESTING: -

OBJECTIVE: - The primary objective of this project, **Secure Pass Vault**, is to develop a comprehensive Android-based application for password management and generation, addressing the increasing concerns surrounding online security and the need for efficient password management solutions. The app is designed to provide users with a secure and user-friendly platform to generate strong, unique passwords and effectively manage them across various accounts and platforms.

By leveraging the capabilities of **Java**, integrating with **Firebase** for seamless data storage and retrieval, and employing robust encryption techniques, **Secure Pass Vault** aims to offer a reliable and accessible solution for enhancing online security. The project focuses on simplifying password management processes, safeguarding sensitive user information, and mitigating the risks of unauthorized access and cyber threats. Through its intuitive design and advanced features, the app empowers users to take control of their digital security and build a secure online presence.

4.1 FRONTEND: -

4.1.1 System Design: - We designed the GUI screens of Secure Pass Vault using XML layouts, which provide a visual interface for users to interact with the app. These screens present information, options, and controls in a graphical format, making it intuitive and user-friendly compared to traditional command-line interfaces. The layouts were created either by dragging and dropping UI components in the Android Studio Layout Editor or by writing XML code to define the structure and appearance of the required UI elements, such as buttons, text fields, and menus. This approach allowed for a seamless and visually appealing design tailored to the app's functionality.

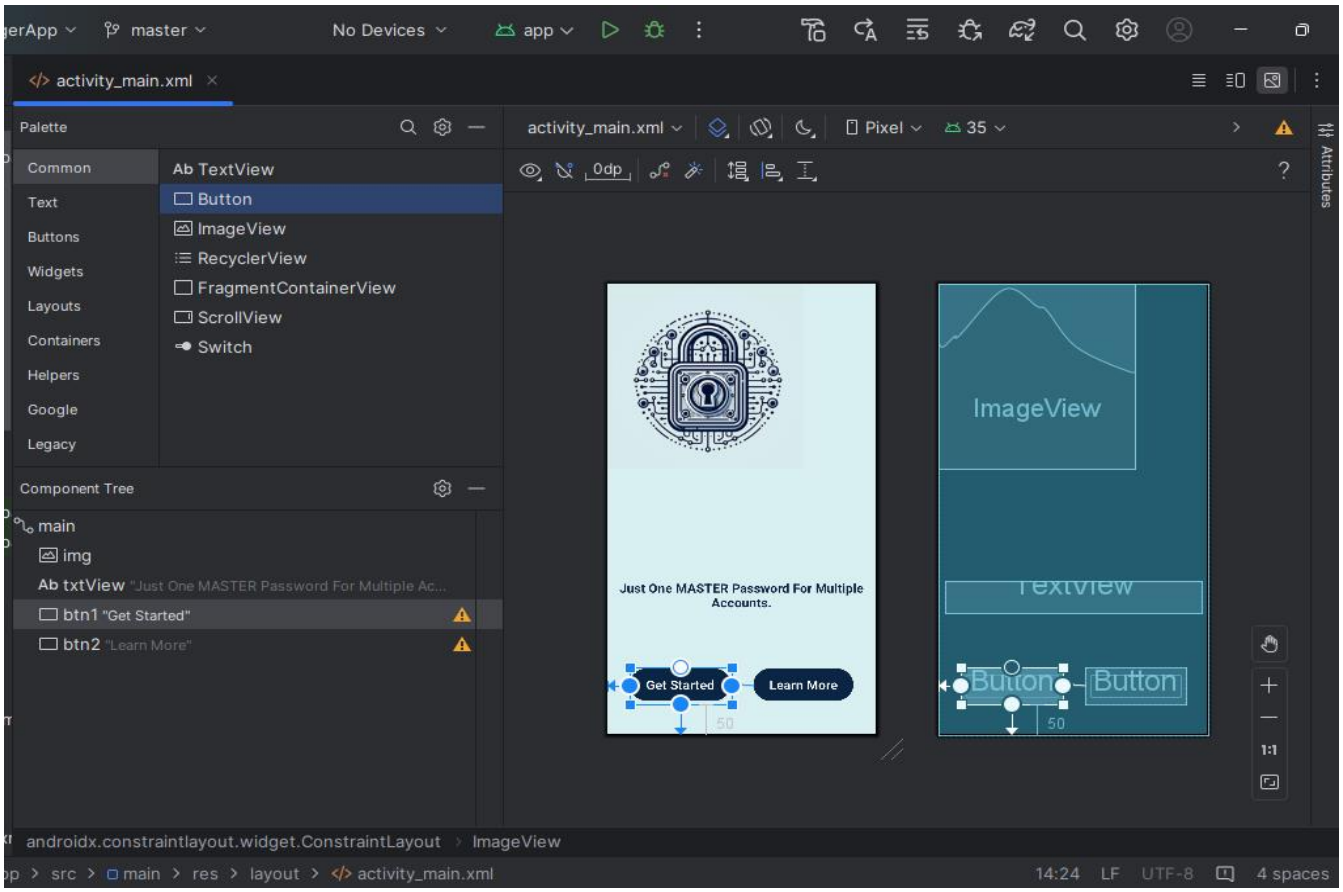


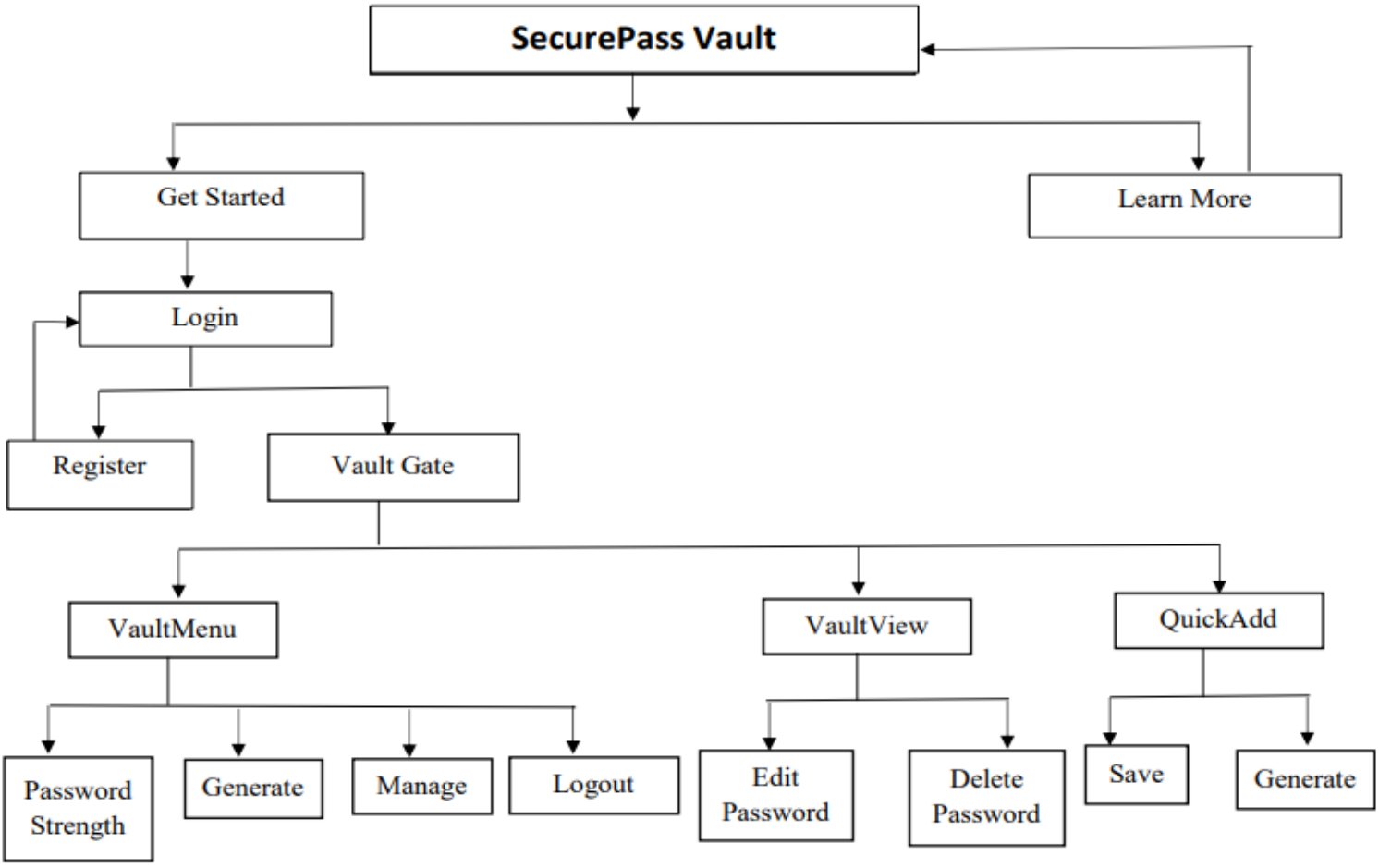
Figure 4.1: Palette in NetBeans

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:background="#D9F0F2">

    <ImageView
        android:id="@+id/img"
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:background="@drawable/c"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        />

    <TextView
        android:id="@+id/txtView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="200dp"
        android:layout_marginRight="10dp"
        android:layout_marginLeft="10dp"
```

androidx.constraintlayout.widget.ConstraintLayout › ImageView

Figure 4.2**:** Code for User Interface Design

4.1.2 PROJECT OVERVIEW: -



4.1.3 FILES in project: -

1. **Utility -** In the dynamic realm of app development, **Firebase** plays a crucial role as a backend service, offering seamless database management, user authentication, and real-time data synchronization. The **utility** file in the Secure Pass Vault project bridges the Android app with Firebase services, enabling secure and efficient data interaction.

   Firebase Firestore serves as the cloud-based database for this application, providing real-time data storage and retrieval capabilities. Unlike traditional relational databases like MySQL, Firestore allows for flexible, zschema-less storage. This enables the app to scale effortlessly while offering robust features such as offline data persistence and automatic synchronization.

   The **Utility** class in the project encapsulates reusable methods to simplify interactions with Firebase. For example:

   - **getCollectionReference()** connects the app to the user's specific collection in Firestore, allowing the storage and retrieval of passwords and notes efficiently.
   - **timestampToString()** converts Firebase timestamps into a readable date and time format for displaying to the user.
   - **showToast()** enhances user experience by providing instant feedback through customizable messages.

   Using Firebase as a backend eliminates the complexity of maintaining server infrastructure and ensures platform independence, enabling the Secure Pass Vault to function seamlessly across devices and environments.

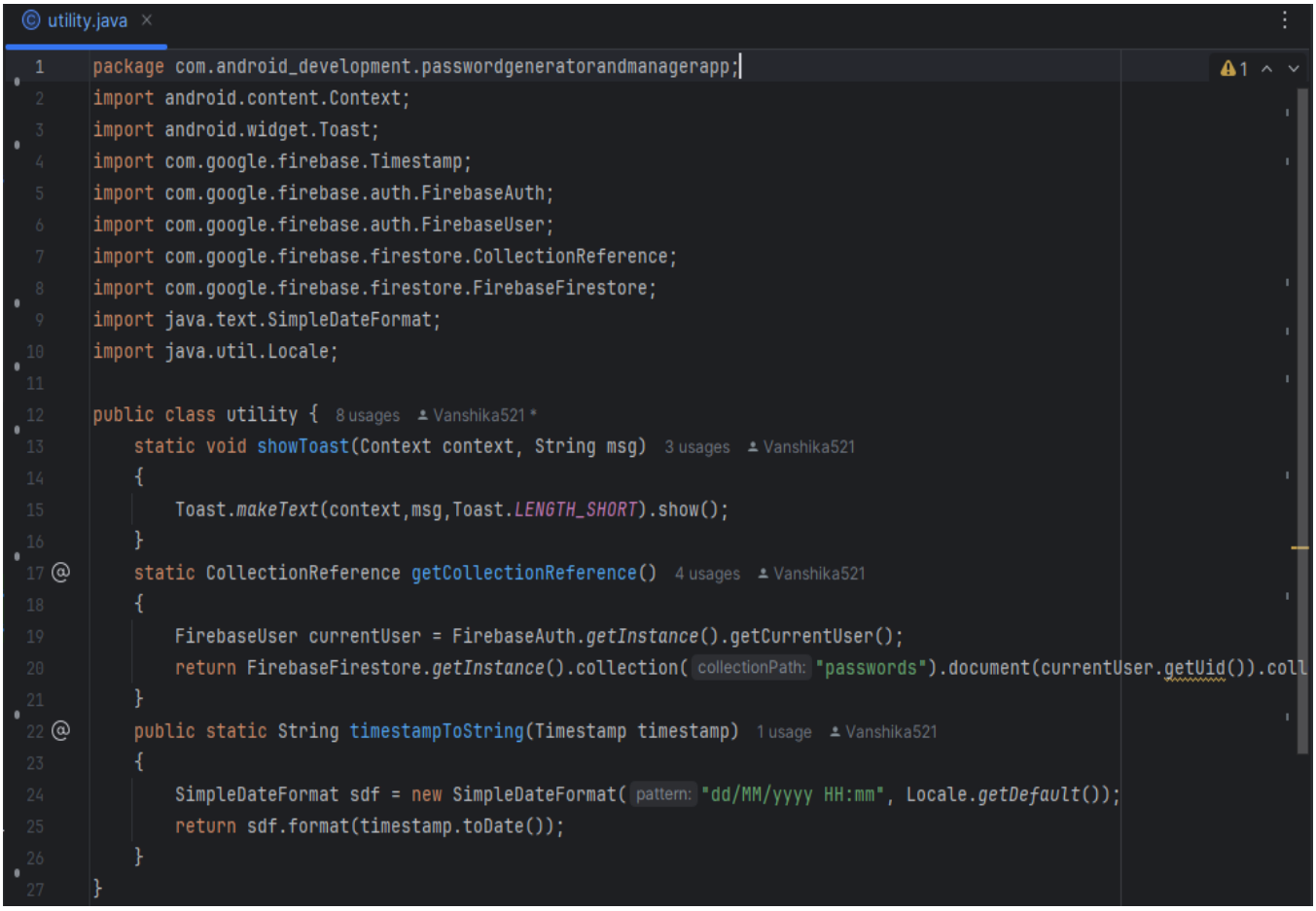


Figure 4.3**:** Code for Utility file

2. **MainActivity-** The **MainActivity** file serves as the starting point for the Secure Pass Vault application, providing users with a clean and intuitive Graphical User Interface (GUI). It offers two primary navigation options:

   1. **Get Started** – Allows users to securely sign in or sign up to manage their saved passwords. This button navigates to the **login** screen.
   2. **Learn More** – Directs users to the **learn_more** screen, where they can access detailed information about the app's features and functionality.

   The layout integrates interactive elements such as buttons, animations, and text views, designed to enhance user engagement and provide seamless user experience. The interface follows **Edge-to-Edge UI principles**, ensuring a visually appealing design compatible with various devices.

Components and Their Roles :-

- **Buttons (btn1 and btn2):-** The two buttons serve as the primary means of navigation:
  - The **Get Started** button (btn1) transitions to the login screen.
  - The **Learn More** button (btn2) opens a screen with app information.

The behavior of these buttons is defined using OnClickListener methods, which invoke **Intents** to navigate to the specified activities.

Code Snippet for Button Actions:

```
48
49      btn1.setOnClickListener(new View.OnClickListener() {  Vanshika521
50          @Override  Vanshika521
51          public void onClick(View view) {
52              Intent intent = new Intent( packageContext: MainActivity.this,login.class);
53              startActivity(intent);
54              finish();
55          }
56      });
57
58      btn2.setOnClickListener(new View.OnClickListener() {  Vanshika521
59          @Override  Vanshika521
60          public void onClick(View view) {
61              Intent intent = new Intent( packageContext: MainActivity.this,learn_more.class);
62              startActivity(intent);
63          }
64      });
65
```

Figure 4.4: Code for Navigation Buttons

- **Animations:-** Animations are applied to various elements to create a dynamic and engaging experience:
  - **Blur Animation**: Enhances the visibility of the tagline text (txtView).
  - **Top Animation**: Adds a smooth sliding effect to the buttons and image (img).

These animations are defined in separate resource files and are applied programmatically.

Code Snippet for Animation Application:

```
40
41          blur = AnimationUtils.loadAnimation( context: this,R.anim.blur);
42          top = AnimationUtils.loadAnimation( context: this,R.anim.top);
43
44          txtView.setAnimation(blur);
45          img.startAnimation(top);
46          btn1.setAnimation(top);
47          btn2.setAnimation(top);
48
```

Figure 4.5**:** Code for Animations

- **ImageView (**img**):** A central image is included to improve the visual appeal and provide branding or thematic elements for the app.
- **TextView (txtView):** A tagline reading *"Just One MASTER Password For Multiple Accounts"* is displayed using a TextView. This serves as the app's message to encourage users to consolidate their password management.

Action Handling:- The app uses **Intents** to handle navigation between screens, which is equivalent to the **ActionPerformed** method in Java Swing applications. For example, clicking the "Get Started" button executes the following code to transition to the login activity:

```
btn1.setOnClickListener(new View.OnClickListener() {  Vanshika521
    @Override  Vanshika521
    public void onClick(View view) {
        Intent intent = new Intent( packageContext: MainActivity.this,login.class);
        startActivity(intent);
        finish();
    }
});
```

Figure 4.6**:** Code for Intent Navigation

Integrated Features and Design

- The use of **Edge-to-Edge UI** ensures the design is adaptive and visually appealing, covering the entire screen seamlessly.
- **Reusable Code**: The structure of OnClickListener and animations demonstrates concise and maintainable code practices, making it easy to extend functionality in the future.

By combining robust functionality with a clean and engaging interface, **MainActivity** forms backbone of Secure Pass Vault, ensuring users can navigate effortlessly while enjoying a visually appealing experience. This integration of design and function makes the app both user-friendly and efficient.

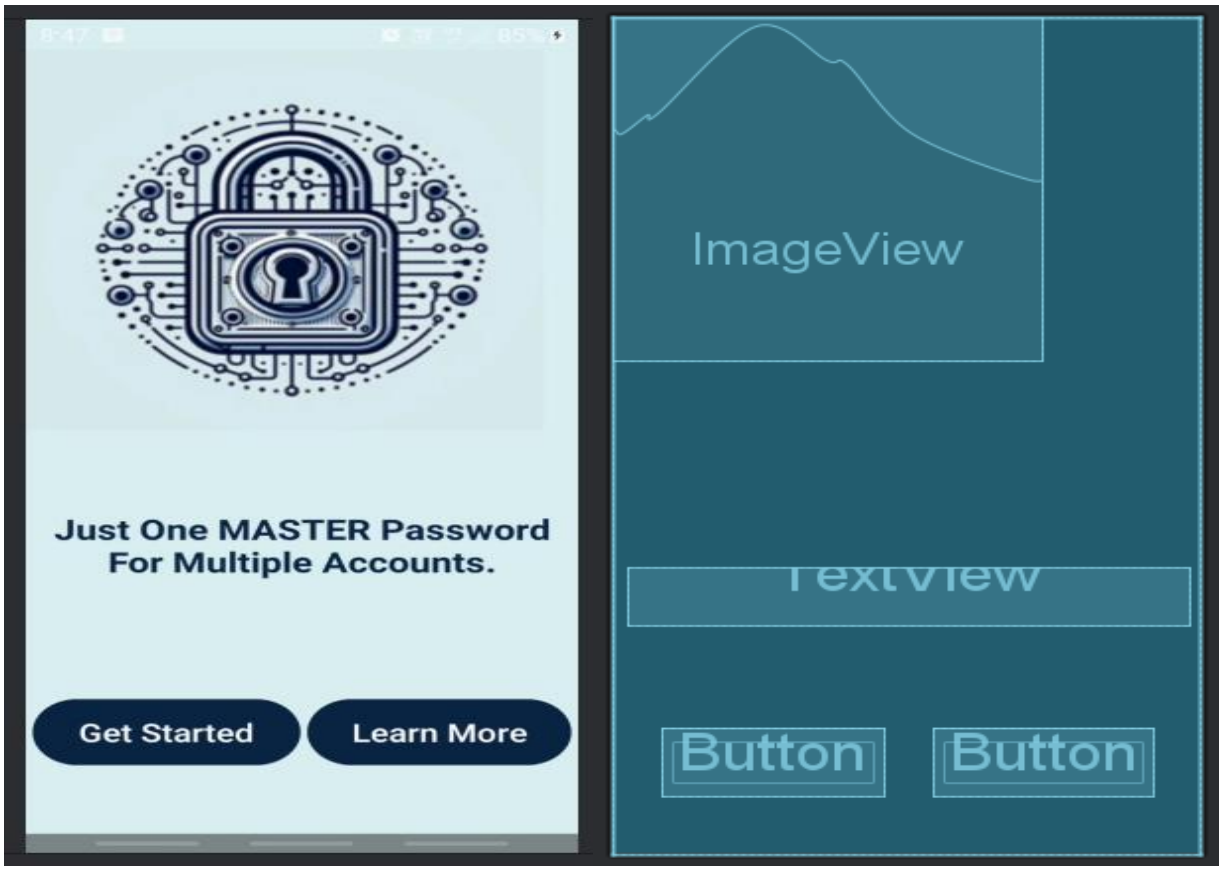


Figure 4.7**:** Welcome Screen

3. **Features And Details Of App (learn_more) –** The learn_more file is designed to provide an engaging and interactive onboarding experience for users of the Secure Pass Vault app. Its purpose is to educate users on the app's key features, security benefits, and functionality. The layout displayed in the image includes:

- **Two TextViews**: These display dynamic content—titles and descriptions explaining the app's security features and guiding the user through the onboarding process.**Titles** highlight the core aspects of secure password management, while **Descriptions** offer concise and actionable advice on topics such as detail or short description of password.
- **Two Buttons**:
  1. **"Next" Button**: This button cycles through different pieces of information, enabling users to learn about each feature one at a time. It uses an animation for smooth transitions, enhancing user experience.
  2. **"Back to Home Screen" Button**: This button redirects the user to the home screen (MainActivity), allowing them to proceed with using the app after reviewing the information.

Functionality of learn_more:-

1. **Dynamic Text Management:-** Titles and descriptions are stored in arrays (title[] and description[]), which are iteratively displayed in the TextView components using the "Next" button. This approach allows for a clean and organized way to showcase multiple pieces of information.
2. **Animation:-** The buttons incorporate a "top" animation, which visually enhances their appearance during transitions. This creates a smooth and polished user experience.
3. **Navigation:-** The "Back to Home Screen" button facilitates navigation, ensuring that users can exit the onboarding process conveniently.

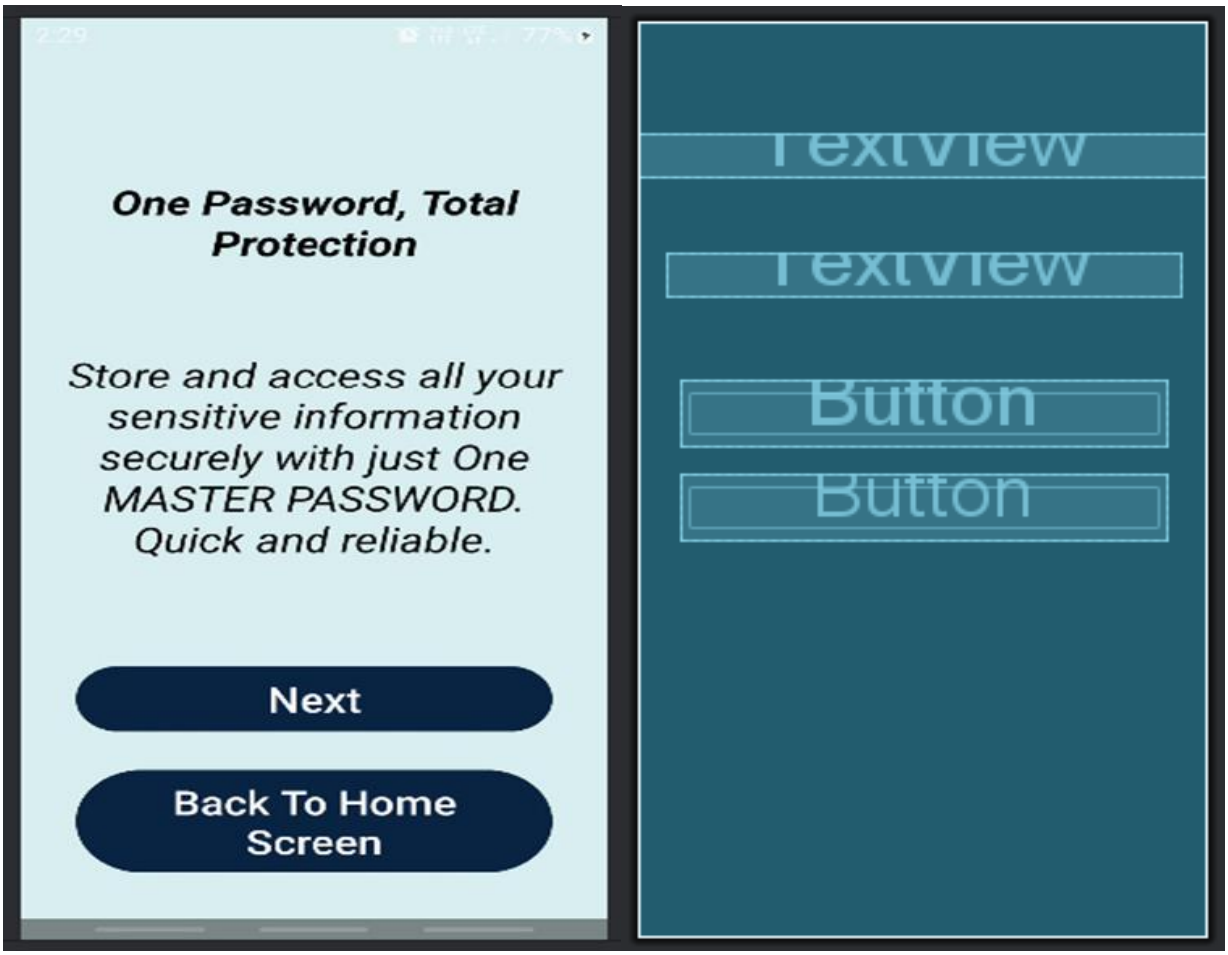


Figure 4.8**:** Features And Details Of App

4. **Login -** The **Login Page** of the Secure Pass Vault app provides a secure and user-friendly gateway for users to access their accounts. It uses Firebase as the backend for authentication, ensuring both functionality and robust security. The design focuses on simplicity, accessibility, and security, as reflected in the following features:

Design Components

1. **ImageView:-** Positioned at the top, the lock icon symbolizes trust and security, aligning with the app's core functionality.
2. **TextView:- Title ("Login")**: Bold and prominent, clearly indicating purpose of page.
3. **EditText Fields:**
   - **Email:** Accepts the registered email ID of the user, validated through Firebase.
   - **Password:** A masked input field for secure password entry.
4. **Button:**
   - **Login Button:** Initiates the authentication process and transitions users to the app's main dashboard upon successful login.
5. **Navigation TextView**:
   - **"Don't Have An Account? Register Here…"**: Redirects new users to the registration page for account creation.

Core Functional Features

1. **Firebase Authentication:**
   - Validates the email and password entered by the user against records stored in Firebase Realtime Database.
   - Ensures secure data transmission using Firebase's encryption protocols.
2. **Input Validation:**
   - **Email Validation:**
     - Ensures the email follows the proper format.
     - Displays an error message if the email field is empty or unregistered.
   - **Password Validation:**
     - Checks for non-empty input and matches the entered password against the Firebase database.
     - Displays an error message for incorrect passwords.
3. **Error Handling and Feedback:**

   Provides informative feedback for login issues, such as:
   - **Invalid Email:** If no account matches the provided email.
   - **Incorrect Password:** If the password does not match the stored record.
   - **Empty Fields:** Prompts users to fill in missing information.

User Experience

- **Responsive Design**: Optimized for all screen sizes to ensure a seamless experience.
- **Accessibility**: High-contrast design, clearly labeled fields make interface user friendly.
- **Streamlined Navigation**: Simplifies transitions to the registration page for new users.

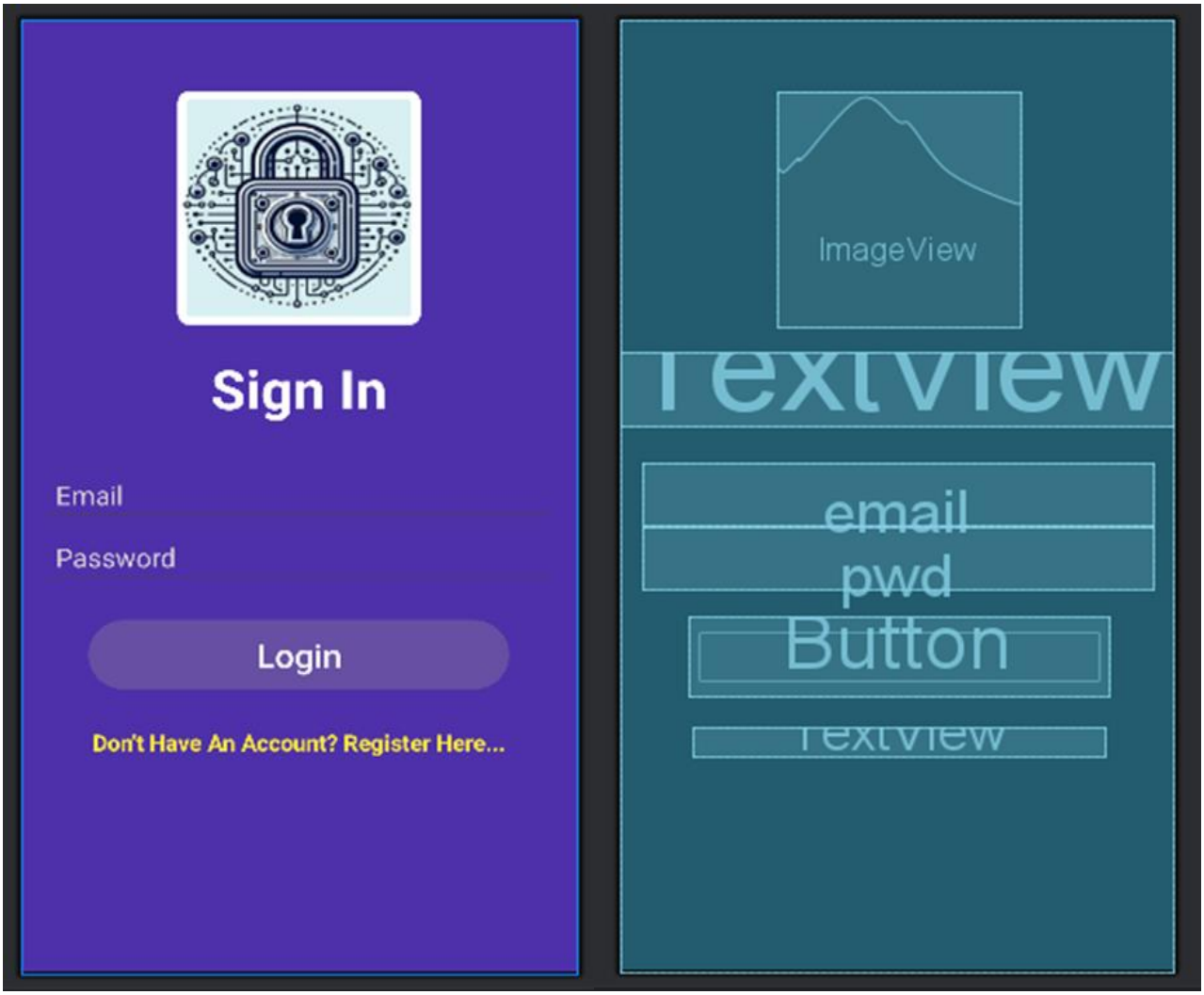


Figure 4.9 **:** Login Screen

5. **Register -** The **Regitser** screen in the Secure Pass Vault app facilitates a smooth and secure onboarding process for new users by allowing them to create an account. The design is clean, user-friendly, as demonstrated in the provided layout. Key components include:

Design Components

1. **ImageView**:
   - Positioned at the top, the logo (a secure lock icon) emphasizes security, aligning with the app's purpose.
   - Adds a visual element to make the screen more engaging and professional.
2. **TextView**:
   - **Title Text (Sign Up)**: Clearly communicates purpose of the screen with bold styling.
3. **EditText Fields**:
   - **Full Name**: Allows users to enter full name, which may be used for personalization within app.
   - **Email**: Collects a valid email address, essential for user identification processes.
   - **Password**: A secure password entry field to ensure the user's account is protected. Input is masked for privacy.
   - **Confirm Password**: A field to verify accuracy of entered password, preventing typos or errors.
4. **Button**:
   - **Register Button**: Submits the user's data for account creation. Styled with rounded corners and a gradient background for a modern appearance.
5. **Navigation TextView**:
   - **"Already Have An Account? Login Here..."**: A secondary option for existing users to switch to the login screen seamlessly.

Functional Features:-

1. **Validation**:
   - **Full Name Validation**: Ensures the user has entered a name. If empty, an error message is displayed.
   - **Email Validation**: Checks for a valid email format to prevent invalid entries.
   - **Password Validation**:
   - Enforces strong passwords, including minimum length and use of special characters.
   - Prevents the use of overly simple passwords (e.g., matching the full name).
   - Displays an error if the password is empty or does not meet strength criteria.
   - **Confirm Password Validation**: Ensures the confirmation password matches the initial password, reducing errors during entry.
2. **Error Messaging**:
   - Provides clear, user-friendly error messages for any invalid or missing fields, guiding users to correct the issue.
3. **Navigation**:
   - Clicking the **Register Button** submits the data to create an account.
   - The **Login TextView** serves as a shortcut for users who already have an account, enhancing navigation flexibility.

User Experience Enhancements

- **Responsive Layout**: The screen components are aligned for optimal readability and ease of use.
- **Visual Clarity**: Colors and fonts are selected for high contrast, ensuring accessibility for all users.
- **Smooth Navigation**: The "Login Here" option provides quick transitions between screens, catering to different user needs.

This design ensures a user-friendly, secure, and visually appealing experience while onboarding users into the Secure Pass Vault app.

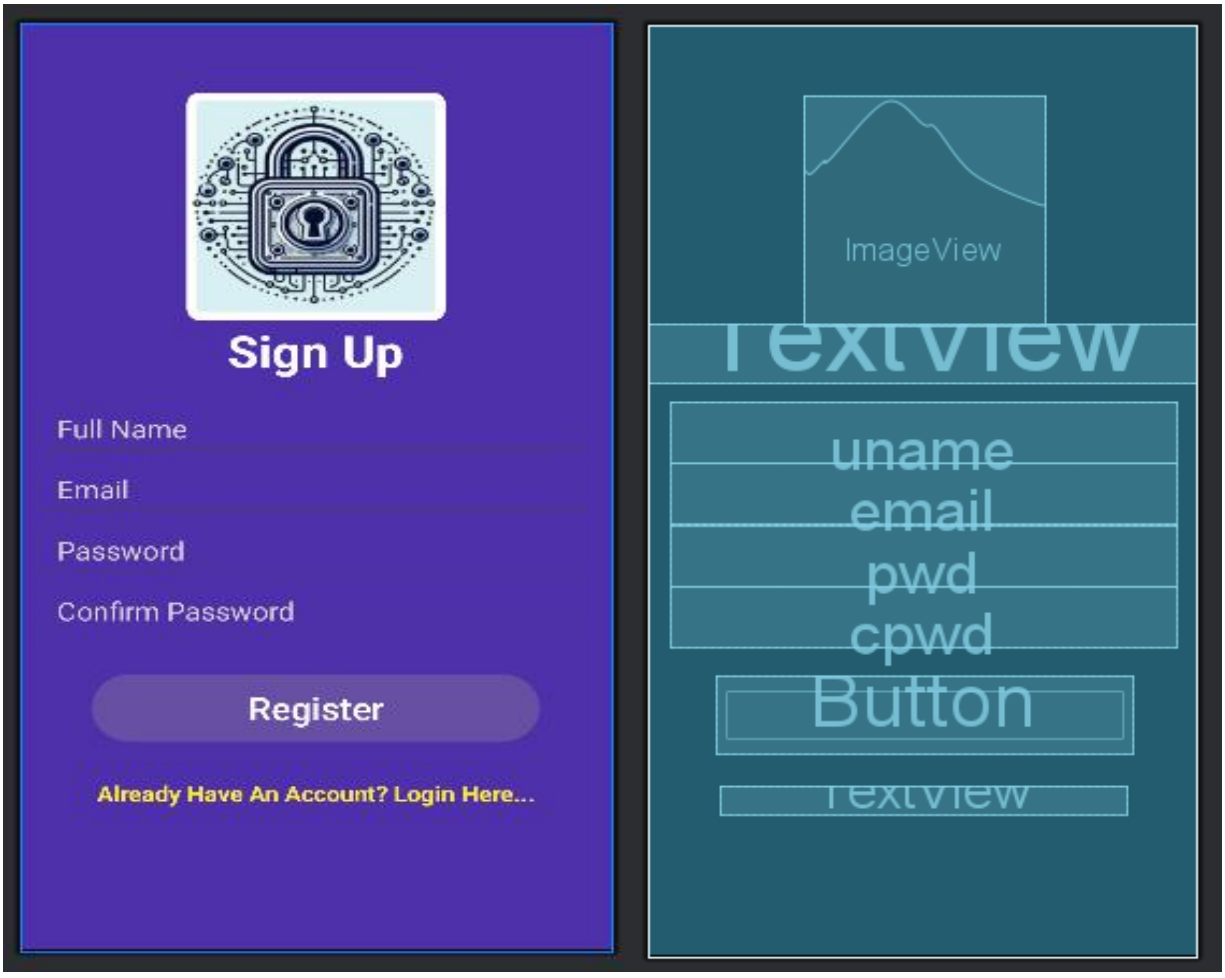


Figure 4.10 **:** Registration Screen

6. **Home Screen (SplashScreen) -** The **Splash Screen** of the Secure Pass Vault app introduces users to its core functionality while seamlessly guiding them through various features and actions. This page integrates modern UI design elements and **Firebase Firestore** to manage user data efficiently.

Design Components

1. **Toolbar :** The header section labeled **"Password Generator And Manager"** reinforces the app's purpose and branding.
2. **ImageView:** Displays lock icon at the center, symbolizing security and trust, aligning with app's theme.
3. **RecyclerView :** Dynamically displays a list of saved passwords and descriptions fetched from Firebase Firestore, offering users quick access to their stored data.

   Additional Functionality:

   - Users can **click on any saved password entry** to edit its information.
   - Includes an option to **delete password**, ensuring complete control over stored data.
4. **Floating Action Button (FAB) :** Positioned at the bottom-right corner, the **Add Button** enables users to create and save new passwords, promoting ease of navigation and action.
5. **Menu Button :** Located in the toolbar, it offers access to various options like:
   - **Password Power Check**: Redirects to a password strength analyzer.
   - **Craft New Password**: Opens a password generator page.
   - **Secure Your Secrets**: Manages passwords(add,edit,delete).
   - **Log Out Securely**: Signs out the user and returns to the login page.

Core Functional Features

1. **RecyclerView with Firebase Firestore Integration**:
   - Uses a query to fetch and display password entries in descending order based on timestamps.
   - The adapter class dynamically binds data (e.g., password, description, timestamp) to individual list items.
   - **On item click:** Allows users to edit the password and description directly.
   - **Delete option:** Enables users to remove any saved password.
2. **Floating Action Button (FAB)**:
   - On click, redirects users to the **Manage Password** page (manage_pwd activity), allowing them to add new password entries.
3. **Popup Menu**:
   - Provides shortcut for accessing additional functionalities, managing app navigation.
4. **Edge-to-Edge Experience**:
   - Adjusts padding and margins dynamically to support different screen sizes and configurations for a seamless UI experience.
5. **Authentication with Firebase**:
   - Implements secure logout feature to enhance user data protection.

RecyclerView Layout :- The RecyclerView layout dynamically displays user data, focusing on simplicity and readability. Each item contains:

1. **Password :** Displays the stored password, highlighted with larger font size for visibility.
2. **Description :** Provides a brief description or title for the password entry.
3. **Timestamp :** Shows the time the password was created or modified, positioned on the right side for clear differentiation.
4. **Interactive Features:**
   - **Edit Password:** Users can tap any item to modify the password or description.
   - **Delete Password:** Offers a delete option to remove unwanted entries.

User Experience

- **Minimalistic and Intuitive Design :** Keeps focus on essential features while maintaining clarity and usability.
- **Responsive Layout :** Adapts to various screen sizes and orientations.
- **Quick Navigation :** Simplifies navigation between password management, strength checking, and generation through menu options.

The Splash Screen, with its integration of **Firebase** Firestore and modern **UI components**, provides users with a secure and efficient way to **Manage passwords** and explore additional functionalities of the app.

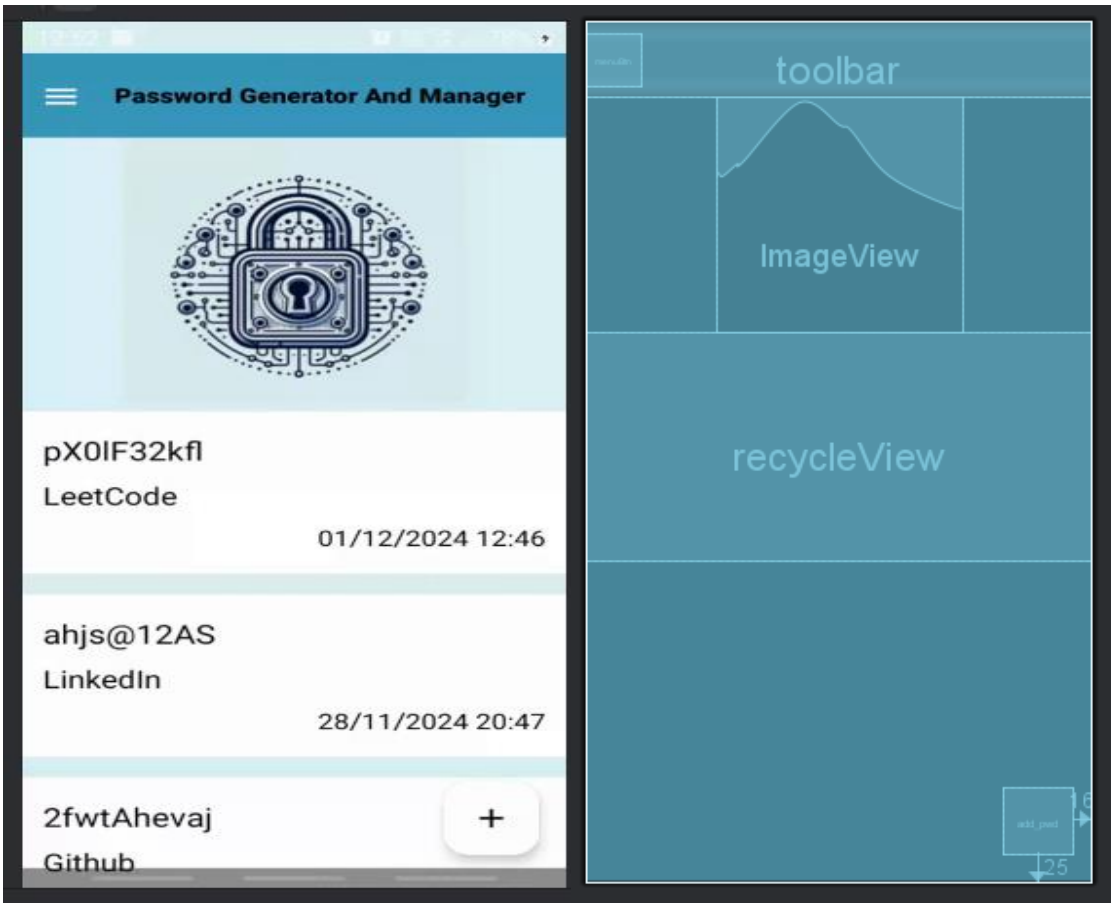


Figure 4.11**:** Dashboard Screen

7. **Password Power Check(pwd_strength) -** The **Password Strength Analyzer Screen** of the Secure Pass Vault app allows users to assess the strength of their passwords and provides dynamic feedback based on security level. This feature enhances user awareness and promotes stronger password practices, aligning with the app's goal of ensuring robust data protection.

Design Components

1. **Header TextView: "Strong password is your first line of defense!"** serves as an informative and motivational message to encourage users to prioritize password security. Bold font and centered alignment enhance readability and emphasize the importance of password strength.
2. **Password Input Field**(EditText)**:** A user-friendly input field with hint **"Add Your Password Here..."**, prompting users to enter their password for analysis.
3. **Analyze Button :** A prominently displayed button labeled **"Analyze Password"**, Triggers the password strength analysis upon a click, offering immediate feedback.
4. **Progress Bar :** A horizontal progress bar dynamically updates its color and visibility to indicate password strength levels:
   - **Dark Green** for master-level security.
   - **Green** for strong passwords.
   - **Yellow** for moderate passwords.
   - **Red** for weak passwords.

5. **Feedback TextView :** Displays feedback about the password strength:
   - "🔐 **Master-Level Security!** 💪" for highly secure passwords.
   - "🔒 **You're well-protected!** 👍 " for strong passwords.
   - "⚠**Decent, but could be stronger!** 🛠" for moderate passwords.
   - " ❗ **Exposed to Danger!** 🔘 " for weak passwords.
6. **Edge-to-Edge Layout** : Uses **Insets** for padding to provide a seamless, full-screen experience while adapting to various devices and orientations.

Core Functional Features

1. **Password Validation :**
   - Implements regular expressions to evaluate the password based on:
     - Inclusion of uppercase and lowercase letters.
     - Presence of digits.
     - Use of special characters.
   - Multiple conditions are assessed to categorize the password into one of four security levels.
2. **Dynamic Feedback :** Provides real-time feedback via :-
   - Progress bar updates.
   - Descriptive and engaging text messages.
3. **Animation :** The feedback TextView uses a **top-down animation** to appear smoothly, adding a dynamic touch to the UI.
4. **User Interaction**:
   - Displays a **Toast message** if the password input field is empty, ensuring no action is performed without user input.
5. **Visibility Control**:
   - The progress bar remains **invisible** by default and becomes visible only when a password is analyzed, maintaining a clutter-free interface.

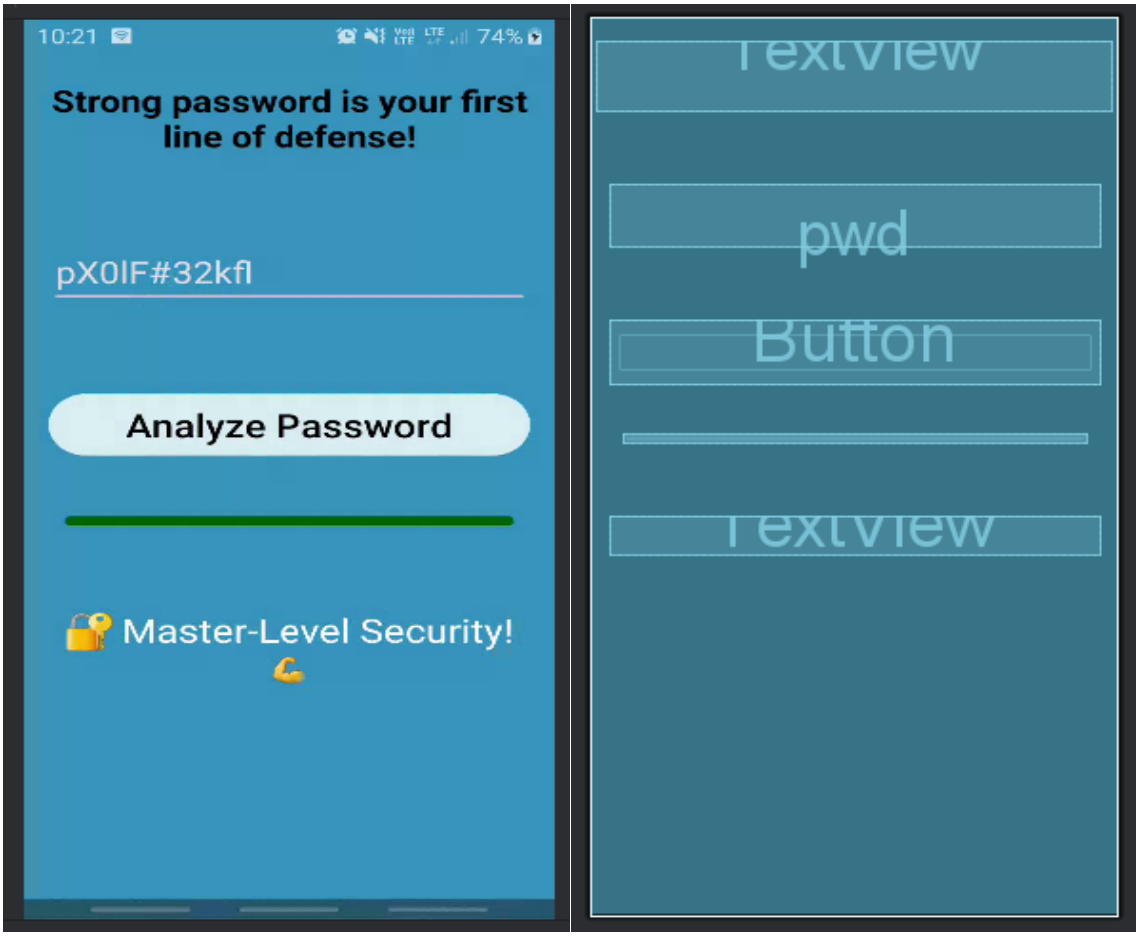


Figure 4.12**:** Password Power Check

8. **Password Generator (generate_pwd)-** In today's rapidly evolving digital era, ensuring robust online security is paramount. With the increase in cyber threats and data loss or misuse, the need for strong, unique passwords has never been more critical. The **Password Generator** feature in Secure Pass Vault addresses this need by providing an intuitive and secure solution for creating highly complex passwords tailored to user preferences.

Functionality and Features:- The **Password Generator** in SecurePass Vault allows users**:-**

1. **Customize Password**:- Users can select specific criteria for password generation:
   - **Lowercase letters**
   - **Uppercase letters**
   - **Numbers**
   - **Special characters**

   These options can be enabled individually or in combination to create a password that meets diverse security requirements.
2. **Adjustable Password Length**:-
   - The slider allows users to choose password length ranging from **8 to 20 characters**.
   - The chosen length is dynamically displayed on the screen for clarity.
3. **Generate Passwords**:
   - By clicking **"Generate Password"** button, users can create unique, random password each time.
   - The password is displayed on the screen for immediate use.
4. **Copy to Clipboard**:- A single-click copy icon allows users to conveniently copy the generated password for use in their accounts or systems.

How It Works:-

1. **User Input**:
   - Users interact with the checkboxes to define password characteristics.
   - Adjust the slider to set the desired password length.
2. **Generate Password**:
   - When the **Generate Password** button is clicked, the app uses a secure algorithm to produce a password based on the selected parameters.
   - A new password is generated each time the button is pressed.
3. **Convenience**:- Once the password is displayed, users can copy it directly with the copy icon for easy integration into any platform.

User Interface:- The Password Generator screen is designed with user convenience and simplicity in mind:

- **Checkboxes**: Allow users to toggle password criteria (lowercase, uppercase, numbers, special characters).
- **Slider**: Provides an interactive way to select password length, with the length dynamically displayed below the slider.
- **Generate Button**: Triggers password generation with a single tap.
- **Copy Icon**: Simplifies copying the password for immediate use.

Importance:- Incorporating this Password Generator into Secure Pass Vault ensures users can:

- Avoid weak and predictable passwords.
- Generate unique, random passwords to strengthen online security.
- Adapt password generation to fit specific use cases or platform requirements.

The **Password Generator** feature in Secure Pass Vault is a powerful tool that empowers users to take control of their online security. By combining flexibility, ease of use, and robust functionality, it offers a seamless way to create strong, unique passwords that protect personal and professional accounts from potential threats. With Secure Pass Vault, users can embark on a safer digital journey with confidence.

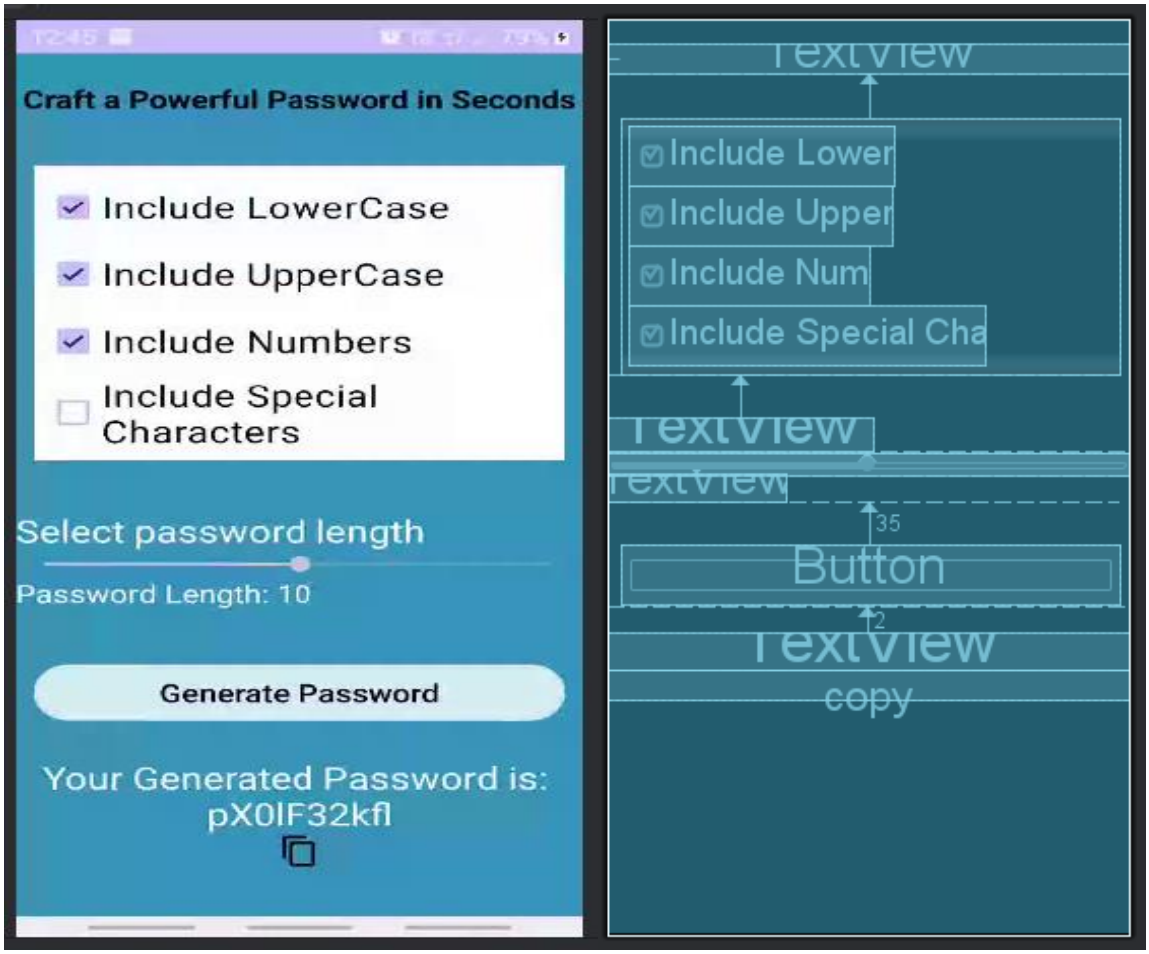


Figure 4.13**:** Craft New Password

9. Password Manager (manage_pwd) - The manage_pwd file is a key component of Secure Pass Vault app, designed to provide users with a seamless and secure way to manage their passwords. In today's digital world, managing multiple accounts requires a robust system for adding, editing, deleting, and organizing passwords. This functionality ensures that sensitive information is easily accessible yet securely protected. By incorporating dynamic modes for adding new passwords or modifying existing ones, the manage_pwd file streamlines password management. Its user-friendly interface, combined with integration with Firestore for real-time storage and retrieval, offers an efficient way to safeguard digital credentials. Users can not only store and edit passwords but also generate strong, random passwords to enhance security posture.

The manage_pwd file is a cornerstone of the Secure Pass Vault app, offering users a secure and intuitive solution to manage their passwords. In the digital era, managing multiple accounts demands a reliable system for adding, editing, deleting, and organizing credentials. This feature ensures sensitive information remains protected while accessible through a user-friendly interface and real-time integration with Firestore. Users can efficiently store, edit, and generate strong, random passwords to bolster security.

Core Functionalities and Features —

1. **Dynamic Access Options**:-
   - The **Manage Passwords** functionality is central to this feature. When activated, it dynamically enables the Sign-In and Sign-Up options, ensuring they are accessible only when required.
   - This design minimizes distractions and streamlines user interaction, emphasizing security-focused workflows.

2. **Encryption for Maximum Security**:-
    - All stored passwords are encrypted and can only be accessed using a **Master Password**.
    - This ensures that only authorized users can retrieve sensitive data, fortifying the app against unauthorized access.

3. **User-Friendly Password Storage**:-
    - Users can create new entries by inputting their **password** and an optional **description** for better organization.
    - A dedicated Save button simplifies securely storing passwords in the vault.
4. **Seamless Account Management**:-
    - The **Sign-Up** feature enables users to create new accounts directly within the app.
    - The **Sign-In** functionality ensures users can securely log in to manage or retrieve their stored passwords.
5. **Comprehensive Security Measures**:-Features such as multi-factor authentication (planned or existing) and robust encryption algorithms ensure top-notch protection against cyber threats.

User Interface and Workflow:-

- **Add New Password Panel**:-
    - Users can input a password and optional details, such as a description, for better organization.
    - The interface provides clear labels and fields, ensuring ease of use.
- **Encryption Notification**:- The fact that all stored passwords are encrypted is reinforced via a minor alert. for added user confidence.

Enhanced Security Practices:- By integrating the sign-in and sign-up options with the act of managing passwords, Secure Pass Vault encourages users to:

- Adopt proactive password management practices.
- Create and store strong passwords for personal and professional use.
- Regularly update and organize passwords to maintain optimal security.

Future Scope

1. **Biometric Authentication**
    - Integrating fingerprint or facial recognition for quicker and more secure access.
2. **Password Sharing**
    - Introducing secure sharing options to send encrypted passwords to trusted contacts.
3. **Dark Web Monitoring**
    - Alerts users if their stored credentials appear in known data breaches.

This approach helps users maintain digital hygiene and protect credentials from threats. The Password Management feature in Secure Pass Vault combines intuitive functionality and strong security, offering a reliable tool to manage digital identities. Its encrypted storage, user-friendly interface, and dynamic features ensure a safer online experience.

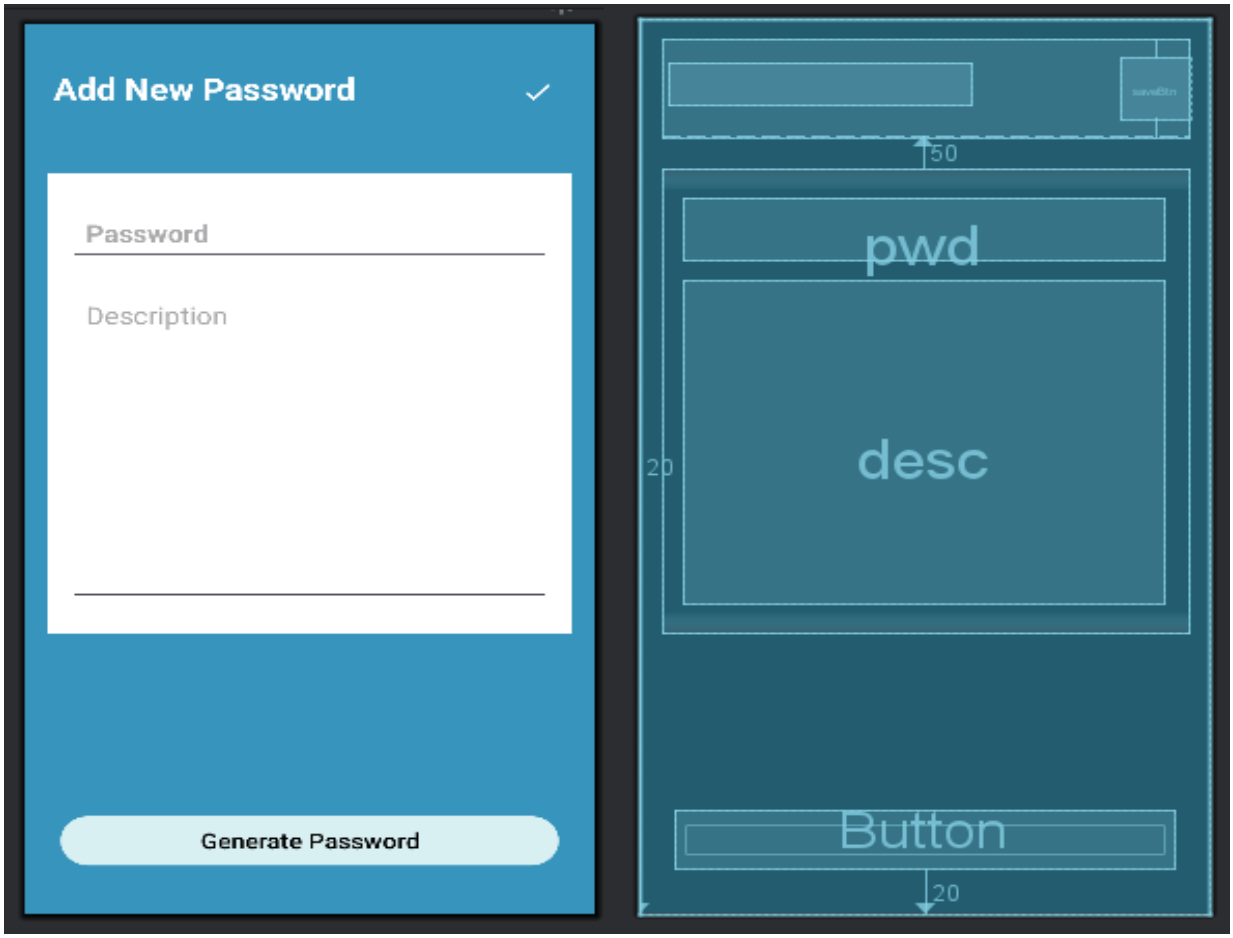


Figure 4.14**:** Vault Key Creator

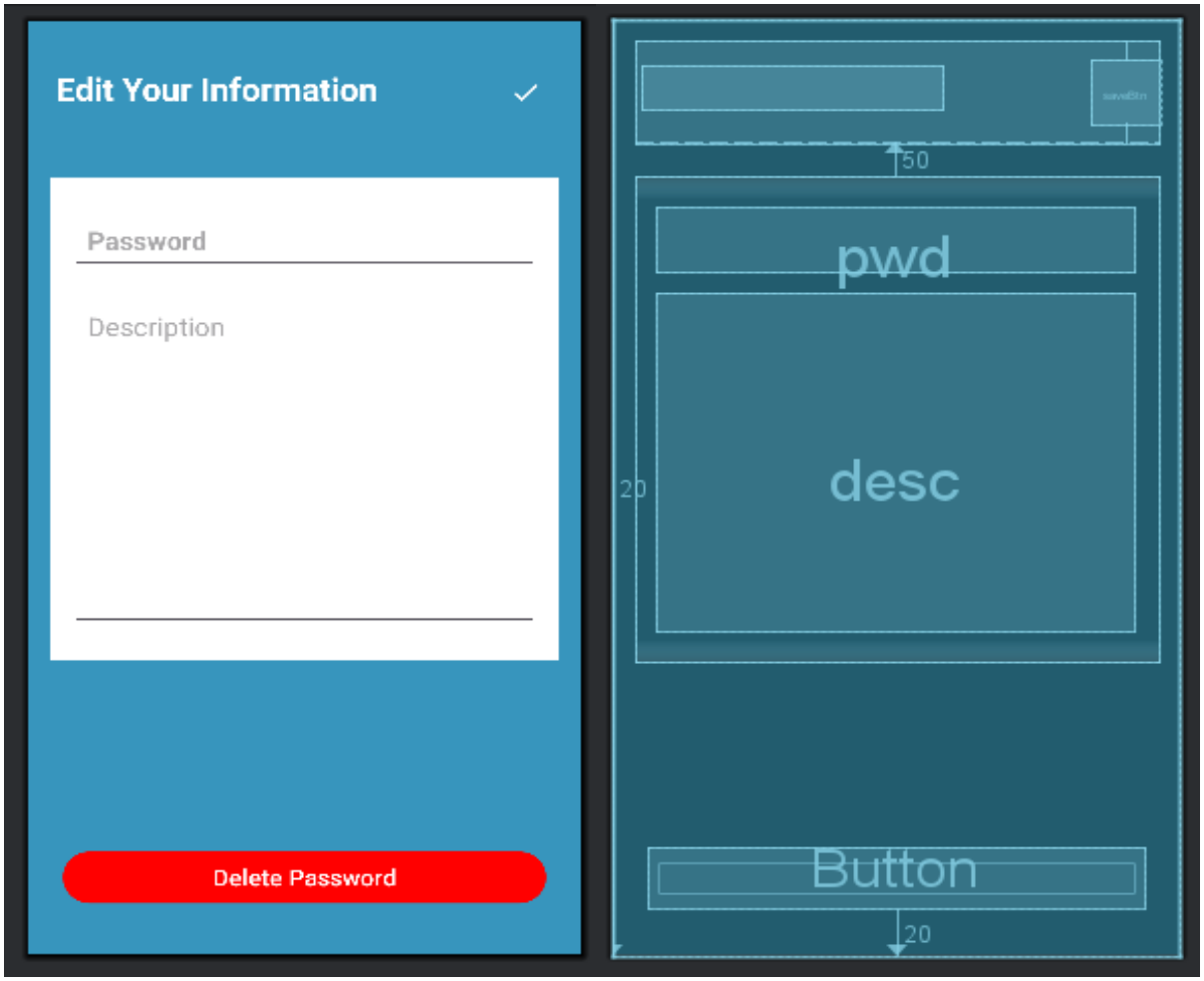


Figure 4.15**:** Vault Key Editor

4.2 BACKEND:-

In the field of software development, effective database design is crucial for managing and securing sensitive information such as user credentials. The **Secure Pass Vault** project utilizes **Firebase Firestore database** for storing user-related information and **Firebase Authentication** for user login and identity management. The database is organized into collections and subcollections to structure and manage sensitive data effectively. The integration of these tools ensures secure and efficient handling of sensitive data.

4.2.1 Authentication:- Firebase Authentication is a core component used to manage user sign-ups and logins. It securely stores user emails and generates unique user IDs (UIDs) for each registered user. These UIDs act as primary keys for identifying user-specific data in the Firestore database.

Key features of the Authentication system:-

- **UID:** A unique identifier for each user generated by Firebase Authentication.
- **Email:** The registered email address used for authentication and user identification.

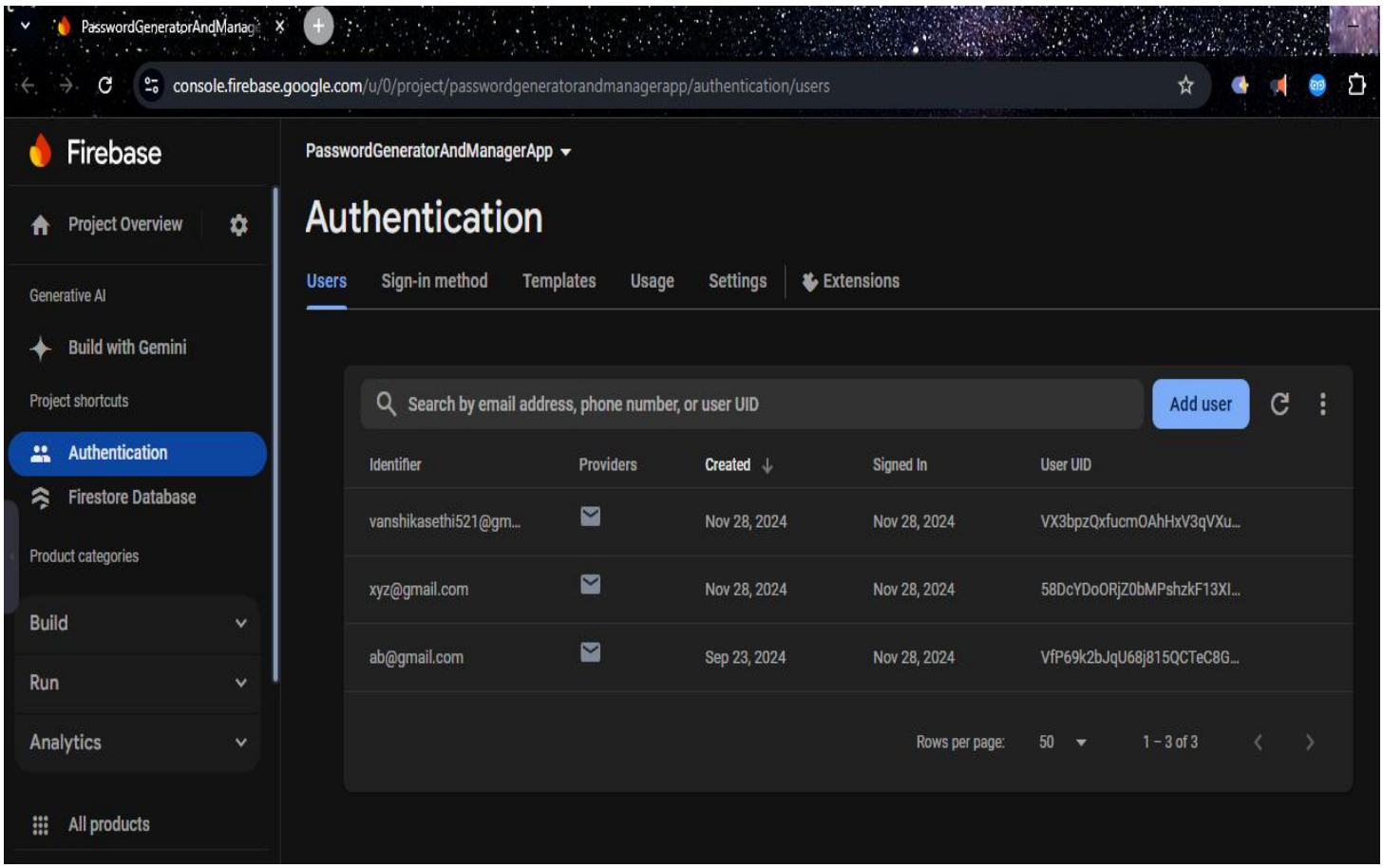


Figure 4.16**:** Authentication Table

4.2.2 Database Overview:- In Firebase Firestore, user data is structured into collections and subcollections for secure and efficient storage. The hierarchical database design ensures logical grouping of related data, making it easier to manage user credentials.

Figure 4.17: Database Overview shows the high-level structure of the Firestore database.

- Database contains a **passwords** collection, where each user has a document named after their UID.
- Each user document contains a subcollection named **pwd**, where individual password entries are stored.

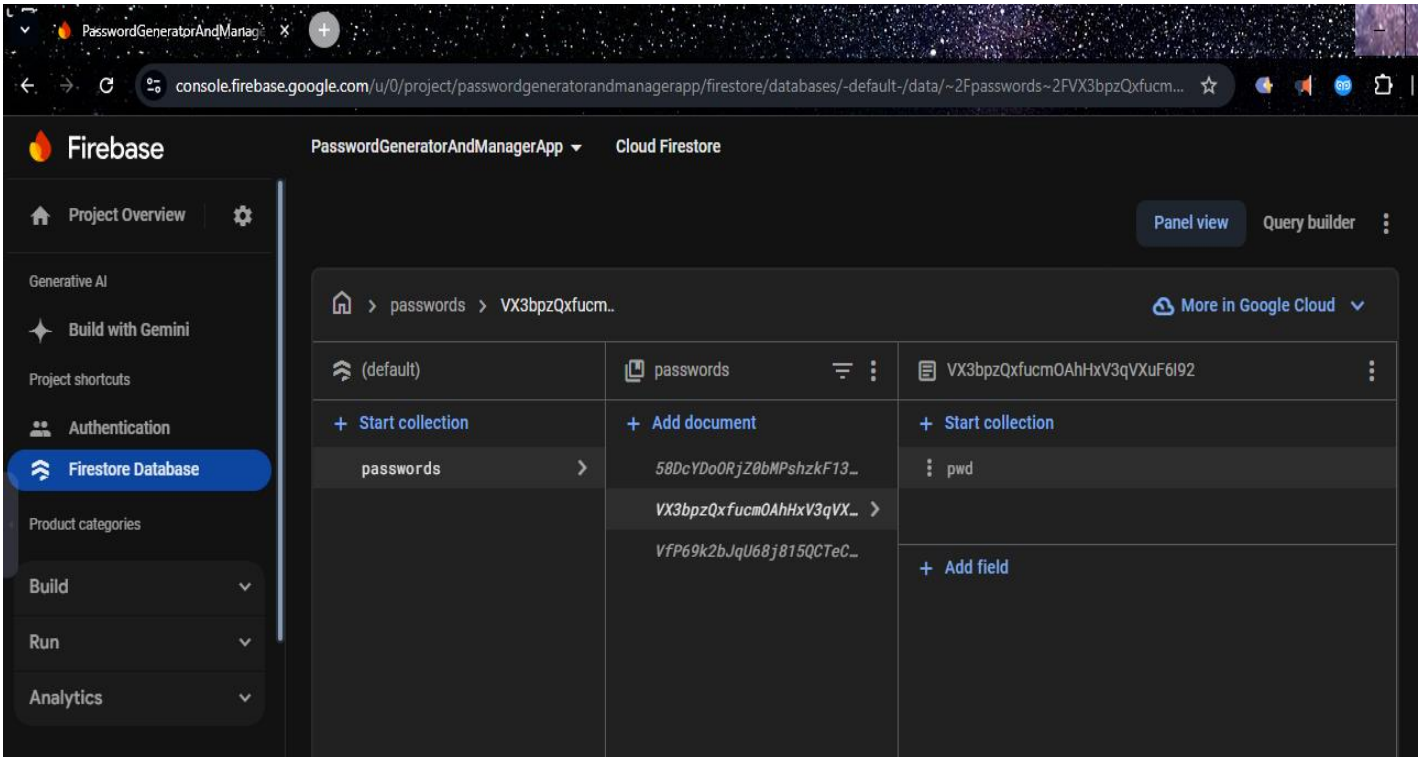


Figure 4.17**:** Database Overview

4.2.3 Passwords Collection:- The Firestore database employs a hierarchical structure to store encrypted passwords for each user. This includes:

- **UID:** A unique identifier linking the authentication details to the stored password data.
- **Password Subcollection (pwd):** Each user has a pwd subcollection that stores individual password entries as documents.

Each document in the pwd subcollection includes the following fields:

1. **Title:** Represents the name of the account.
2. **Password:** Stores the encrypted password value for the account.
3. **Timestamp:** Indicates when the password was created or last updated.
4. **Content :** Additional information or description for the stored password entry.

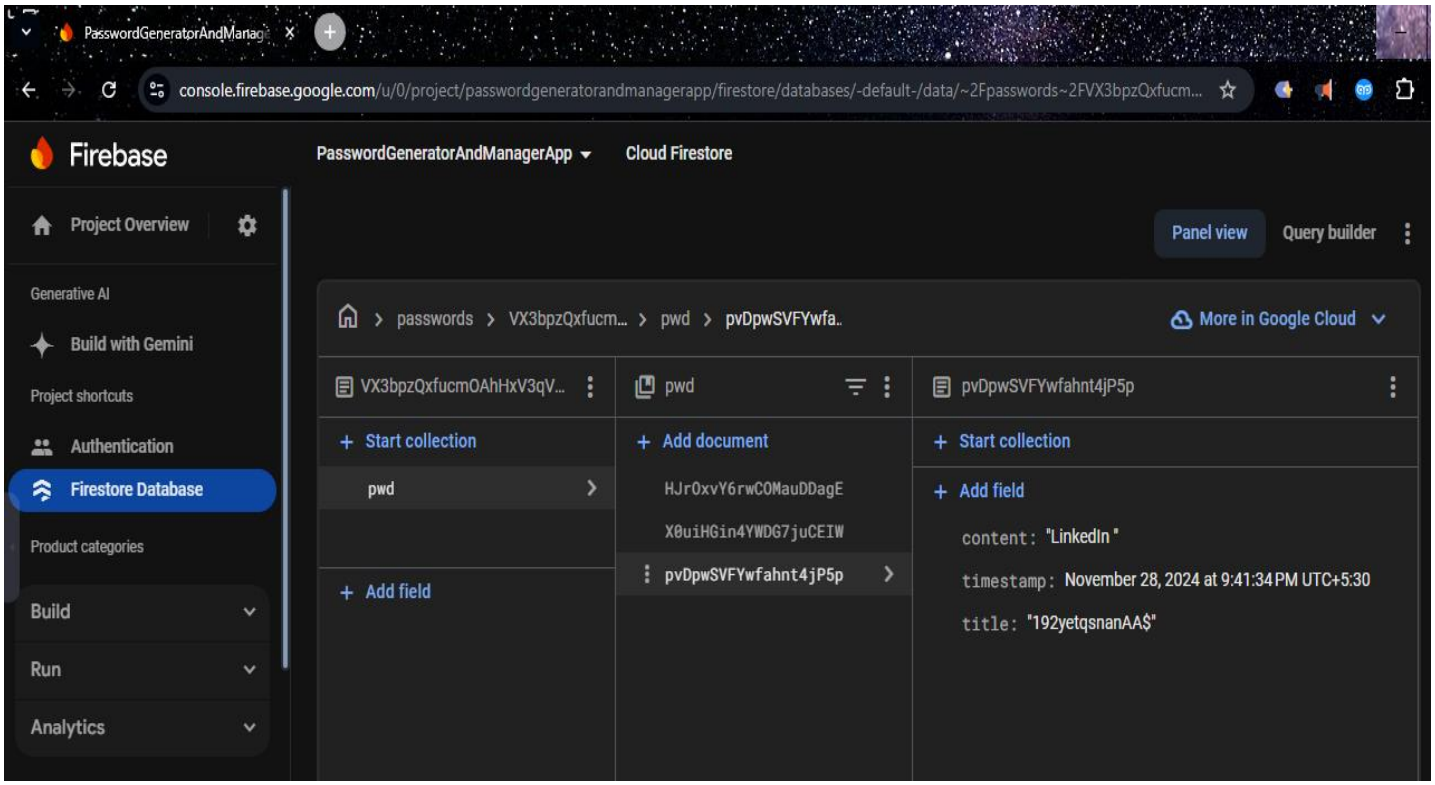


Figure 4.18**:** Firestore Password Document Structure

CHAPTER - 5

CONCLUSION AND FUTURE SCOPE: -

5.1 Conclusion: -

In an era where digital security is vital to modern life, **Secure Pass Vault** emerges as more than just an app—it is a dependable protector, a vigilant guardian, and a symbol of empowerment in safeguarding our digital identities. **Secure Pass Vault** combines robust security features with an intuitive design, simplifying the complexities of password management and enabling users to confidently navigate the ever-evolving digital landscape.

Secure Pass Vault does more than manage passwords; it offers a comprehensive solution for strengthening users' online defenses. With features such as **secure storage**, **automated password generation**, and **streamlined access**, it ensures that sensitive information is safeguarded with advanced encryption and authentication measures. Every stored password and every secured login builds a virtual stronghold for its users, developing trust and peace of mind.

This is just the beginning. As cyber threats grow more sophisticated, **Secure Pass Vault** is committed to evolving alongside them. By incorporating cutting-edge technologies like **multi-factor authentication**, **biometric login options**, and **encrypted cloud synchronization**, the app will remain at the forefront of digital security solutions. Its commitement to privacy standards and integration of innovative features ensures its ongoing relevance in the dynamic world of password management.

However, the true impact of **Secure Pass Vault** extends beyond its technical strengths. It promotes a culture of digital awareness, empowering users to take control of their online safety and establish a secure digital presence. The app emphasizes not only protecting personal and organizational data but also building confidence in a world where security and privacy are necessary.

With **Secure Pass Vault**, users can navigate the digital landscape knowing that their sensitive data is shielded by robust security measures. The app stands as a powerful defense against cyber threats, transforming security into empowerment and serving as a trusted partner in managing digital identities.

As we look to the future, **Secure Pass Vault** will continue to innovate and adapt, ensuring that users are equipped with the best tools to face emerging challenges. In this fast-paced digital era, it is not merely an app but a testament to our collective commitment to security, privacy, and peace of mind. With **Secure Pass Vault** by our side, we confidently step into the expansive digital world, armed with a secure and trusted solution tailored to the needs of modern users.

5.2 Future Scope: -

The future scope of the Secure Pass Vault project is essential for ensuring its relevance, adaptability, and utility in the constantly evolving landscape of cybersecurity. Key considerations include :-

1. Adaptation to Evolving Security Threats
2. Enhanced User Experience.
3. Compliance with Industry Standards.
4. Competitive Advantage (staying innovative for passwords).
5. Long-Term Sustainability.

Future Enhancements and Trends to Consider

1. **Enhanced Password Analysis:** Integrate advanced tools to analyze password strength and complexity. Provide users with real-time feedback and actionable suggestions to strengthen passwords based on evolving security guidelines.

2. **Biometric Authentication:** Introduce biometric authentication methods such as fingerprint or facial recognition for user login. This feature would add an extra layer of security and convenience.
3. **Multi-Factor Authentication (MFA):** Support multi-factor authentication, enabling users to combine traditional passwords with OTPs or biometrics. MFA enhances resilience against unauthorized access and ensures better protection of sensitive data.
4. **Cloud Synchronization:** Implement secure cloud storage options for syncing password vaults across devices. By using end-to-end encryption, users can access their vaults securely from any device while ensuring data privacy.
5. **Password Sharing and Collaboration:** Add features for securely sharing selected passwords with trusted individuals or team members. Introduce permissions and audit trails to maintain control and accountability during collaborations.
6. **Cross-Platform Compatibility:** Expand the app to support platforms like iOS and desktop systems, ensuring users can seamlessly access their vault from different devices. This feature would improve the app's usability and accessibility.
7. **Machine Learning and AI Integration**: Leverage machine learning algorithms to analyze user behaviors, predict password vulnerabilities, and detect suspicious activities. AI can also offer personalized suggestions to users, improving the overall experience.
8. **Customization Options**: Provide users with more options to personalize the app's appearance, such as themes, font sizes, and layouts. Adding dark mode and accessibility features will further enhance the user experience.
9. **Data Backup and Recovery:** Offer secure backup solutions that allow users to restore their data in case of device loss. Automated backups can be implemented for peace of mind.
10. **Security Monitoring and Alerts:** Add monitoring features to detect and notify users of suspicious login attempts or unauthorized access to their account

Bibliography

- https://console.firebase.google.com/u/0/
- https://console.firebase.google.com/u/0/project/passwordgeneratorandmanagerapp/firestore/databases/-default-/data
- https://chatgpt.com/securepassvault
- https://www.geeksforgeeks.org/introduction-to-android-development/
- https://www.javatpoint.com/android-tutorial
- https://www.coursera.org/specializations/android-app-development
- https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.linkedin.com%2Fpulse%2Fsoftware-development-life-cycle-sdlc-tutorial-richard-harris&psig=AOvVaw1GwylTUbRRmd74qwTvruxF&ust=1733419622799000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCOifr7rRjooDFQAAAAAdAAAAABAE
- https://en.wikipedia.org/wiki/Android_software_development
- https://www.javatpoint.com/java-logo
- https://www.pngwing.com/en/search?q=xml+Icon