TABLE OF CONTENT

Chapter 1 : Introduction

1.1 Introduction

Note-taking apps have become an essential tool in today's fast-paced and ever-changing world. They allow individuals to capture, organize and manage information efficiently, making it easier to stay on top of tasks and projects. A note-keeping app, also known as a note-taking app, is a digital tool that enables users to create, edit, and organize notes on a variety of devices, including smartphones, tablets, and computers.

The concept of note-taking has been around for centuries, with ancient civilizations using clay tablets and parchment scrolls to record information. However, the development of the personal computer in the 1970s and the subsequent rise of the internet and mobile devices have led to the proliferation of note-taking apps. Early examples include Evernote, which was launched in 2008, and OneNote, which was released in 2003.

The need for a reliable note-taking app has never been greater, as individuals and teams are increasingly reliant on technology to manage their daily lives. With the ability to capture and organize information in a variety of formats, including text, images, audio, and video, note-taking apps have become an essential tool for staying organized and productive.

NoteWell is a simple and user-friendly note-taking app that allows users to easily create, read, update and delete notes. It has a clean and intuitive interface that makes it easy for users to navigate and find the information they need. With NoteWell, users can create new notes, edit existing ones, and use them for different purposes, such as work, personal, or school.

NoteWell app is designed to be easy to use for all users, it has all the basic functionality of a note-taking app, such as creating, editing, and deleting notes. NoteWell is a great choice for individuals and teams who are looking for a simple and efficient way to capture, organize and manage information.

1.2 Problem Statement

The problem with traditional methods of note-taking, such as pen and paper or a basic word processor, is that they lack the organization, collaboration, and accessibility features that modern users require. Students, professionals, and individuals in various fields need a note-taking app that can easily capture, store, and retrieve notes, as well as share them with others.

Current note-taking apps also have their own limitations, such as a lack of integration with other tools, poor synchronization across devices, limited customization options, and high subscription fees. Therefore, there is a need for a note-taking app that offers a user-friendly interface, robust features, seamless synchronization, and affordable pricing, while addressing the privacy and security concerns of users. Such an app would enable users to take notes efficiently, collaborate with others, and stay organized, thereby enhancing productivity and success.

1.3 Objective

- To store all notes and important information digitally, usually in a cloud-based storage system.

- To provide regular data backup and restore features to ensure that users' notes are safe and can be retrieved easily in case of any loss or damage.

1.4 Problem Solution

The app should have a simple and intuitive interface that allows users to take notes quickly and easily.It should offer a wide range of features that cater to different users' needs and preferences. The app should enable users to sync their notes seamlessly across different devices and platforms, without losing any data. It should also provide automatic backups and restore options to ensure the safety and availability of users' notes.The app should ensure the privacy and security of users' data through strong encryption, two-factor authentication, and other security measures. It should also provide clear privacy policies and terms of use, and transparently disclose any data sharing practices.

Chapter 2 : System Analysis

2.1 Background study

Note-taking apps have become increasingly popular in recent years as more people seek to manage their digital lives more efficiently. These apps offer users a way to capture, organize, and retrieve notes on a variety of devices.

The history of note-taking can be traced back to the invention of writing itself. However, it wasn't until the invention of paper and the printing press that note-taking became more widespread. Note-taking was originally done by hand, and later, typewriters were used to create typed notes.

With the advent of personal computers and mobile devices, note-taking has become even more ubiquitous. The first digital note-taking software was introduced in the 1980s, but it wasn't until the 2000s that note-taking apps became more widely used.The rise of smartphones and tablets has also contributed to the popularity of note-taking apps. These devices make it easy to capture notes on the go and sync them across multiple devices.

Today, there are many different note-taking apps available, each with their own features and benefits. Some note-taking apps offer simple text-based note-taking, while others offer more advanced features like voice memos, image and video capture, and handwriting recognition. Note-taking apps can be useful for a variety of purposes, including personal organization, work productivity, and academic studies. Many note-taking apps also offer collaboration features, allowing users to share notes and work together on projects.

Overall, note-taking apps have evolved significantly over the years and continue to be an important tool for many people in managing their digital lives.

2.2 Feasibility Analysis

A feasibility analysis of a note-taking app would involve evaluating the technical, financial, and market feasibility of the app. Here are some factors to consider:

1) **Technical feasibility:** This involves evaluating whether the app can be developed and implemented using existing technology and resources. It may involve considering factors such as the app's required features, the development timeline, and the availability of skilled developers.

2) **Financial feasibility:** This involves evaluating whether the app can generate enough revenue to cover its development and ongoing operating costs. It may involve considering factors such as the app's pricing strategy, target market, and expected revenue streams.

3) **Market feasibility:** This involves evaluating whether there is sufficient demand for the app in the market. It may involve considering factors such as the size of the target market, competing apps, and trends in note-taking behavior.

4) **User experience feasibility:** This involves evaluating whether the app provides a user-friendly and intuitive experience for its target audience. It may involve considering factors such as the app's user interface, ease of use, and accessibility.

5) **Security and privacy feasibility:** This involves evaluating whether the app provides sufficient security and privacy protections for its users. It may involve considering factors such as data encryption, secure user authentication, and adherence to privacy regulations

In summary, a feasibility analysis of a note-taking app would involve evaluating whether the app can be developed and implemented using existing technology and resources, whether it can generate enough revenue to cover its costs, whether there is sufficient demand for the app in the market, whether it provides a user-friendly experience, and whether it provides sufficient security and privacy protections for its users.

2.3 Requirement Analysis

Requirement analysis is an important process in developing a note-taking app. It involves identifying and documenting the specific features and functionalities that the app must have in order to meet the needs of its users.

2.3.1 Functional Requirements

Functional requirements describe what the note-taking app should do in terms of its features and functionalities. Here are some key functional requirements to consider for a note-taking app:

1. **Note creation and editing:** Users should be able to create new notes, edit existing notes and delete notes.

2. **Cloud synchronization:** The app should synchronize notes across multiple devices and platforms, such as desktop, web, and mobile.

3. **Backup and restore:** The app should provide automatic backup options and restore functionality to prevent data loss.

4. **Security and privacy:** The app should provide adequate security measures to protect users' data, such as data encryption, secure user authentication, and adherence to privacy regulations.

5. **Responsive UI:**The app should work seamlessly across multiple devices without breakage.

In summary, functional requirements for a note-taking app should cover key areas such as note creation and editing, cloud synchronization,backup and restore, security and privacy, and customization.

2.4 Non-Functional Requirements

Non-functional requirements describe how the note-taking app should perform in terms of its performance, usability, and other characteristics that are not directly related to its features and functionalities. Here are some key non-functional requirements to consider for a note-taking app:

1. **Usability:** The app should have a user-friendly interface that is easy to navigate, with intuitive controls and clear visual cues.

2. **Performance:** The app should be fast and responsive, with minimal lag or delay when creating, editing, or syncing notes.

3. **Scalability:** The app should be able to handle a large number of notes and users, without compromising on its performance or functionality.

4. **Reliability:** The app should be reliable, with minimal downtime or crashes.

5. **Availability:** The app should be available to users 24/7, with minimal maintenance or downtime.

6. **Compatibility:** The app should be compatible with a wide range of devices and platforms, including desktop, web, and mobile.

7. **Accessibility:** The app should be accessible to users with disabilities, with support for assistive technologies such as screen readers.

8. **Security:** The app should have robust security measures in place, such as data encryption and secure user authentication, to protect users' data.

9. **Privacy:** The app should adhere to privacy regulations and protect users' personal information from unauthorized access or disclosure.

10. **Performance under stress:** The app should be able to handle high traffic or usage without compromising on its performance or functionality.

In summary, non-functional requirements for a note-taking app should cover key areas such as usability, performance, scalability, reliability, availability, compatibility, accessibility, security, privacy, and performance under stress. These requirements are critical to ensuring that the app is not only functional but also performs well and meets the needs of its users.

2.5 Structuring System Requirements

2.5.1 Use Case Diagram

Fig 1: Use Case Diagram

The use case diagram above depicts the interplay between the system's actors, namely the user and the backend server. As the primary drivers of system activity, these actors play a crucial role in the functionality of the system. In order to provide a comprehensive overview of the system's operation, a brief description of each actor is presented below:

1. User:

   The system provides a straightforward user registration process via a Google sign-up. Once registered, users can log in to the system and access a list of notes associated with their account, provided they have been authenticated by the backend server. To ensure secure access, each user is given a unique access token by the backend server, which serves to verify their identity. It's worth noting that users are only able to access and edit notes that are associated with their own account, thereby ensuring both the security and privacy of the system.

2. Backend Server:

   Backend server registers a new user and provides the user with a unique token for verification that is used to communicate with the server. It also handles the login and authentication . It has access to all notes in the database and also responds to requests from the user and returns an adequate response.

2.5.2 ER Diagram

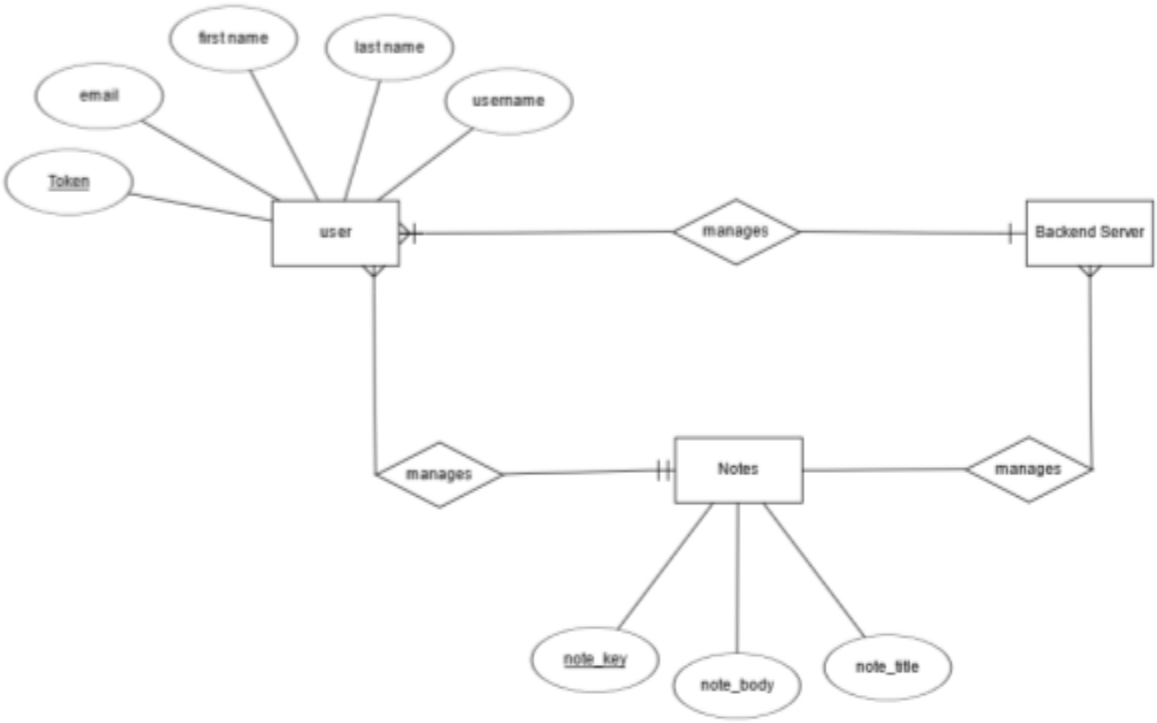

Fig 2: ER Diagram

The proposed ER diagram shown above depicts three main entities: User, Backend Server, and Notes. It is important to note that one backend server is capable of managing multiple users and notes. Similarly, each user has the ability to manage multiple notes simultaneously. This design allows for efficient management and organization of data, promoting optimal system performance and user experience.

2.5.3 Context Diagram

Fig 3: Context Diagram

Upon analyzing the above context diagram, it is observed that two distinct entities exist, namely the user and the backend server. The user entity is responsible for registering to the system and gaining access to its functionalities. On the other hand, the backend server entity is responsible for managing user and notes details, promoting efficient data management and organization.

The notes functionality is another key aspect of this system, and it is accessed and managed by the user entity. Specifically, the user is able to add and edit notes according to their preferences, thus allowing them to customize their experience within the system. Moreover, both entities have visibility over the notes created, ensuring seamless collaboration and information sharing.

It is also important to note that any changes made to the notes are registered to the backend server entity. This feature promotes accountability and traceability, and ensures that all modifications are tracked and recorded. Overall, this design enhances the overall user experience and system functionality, while promoting optimal data management and security.

2.5.4 Data Flow Diagram (DFD) Level 1 for User

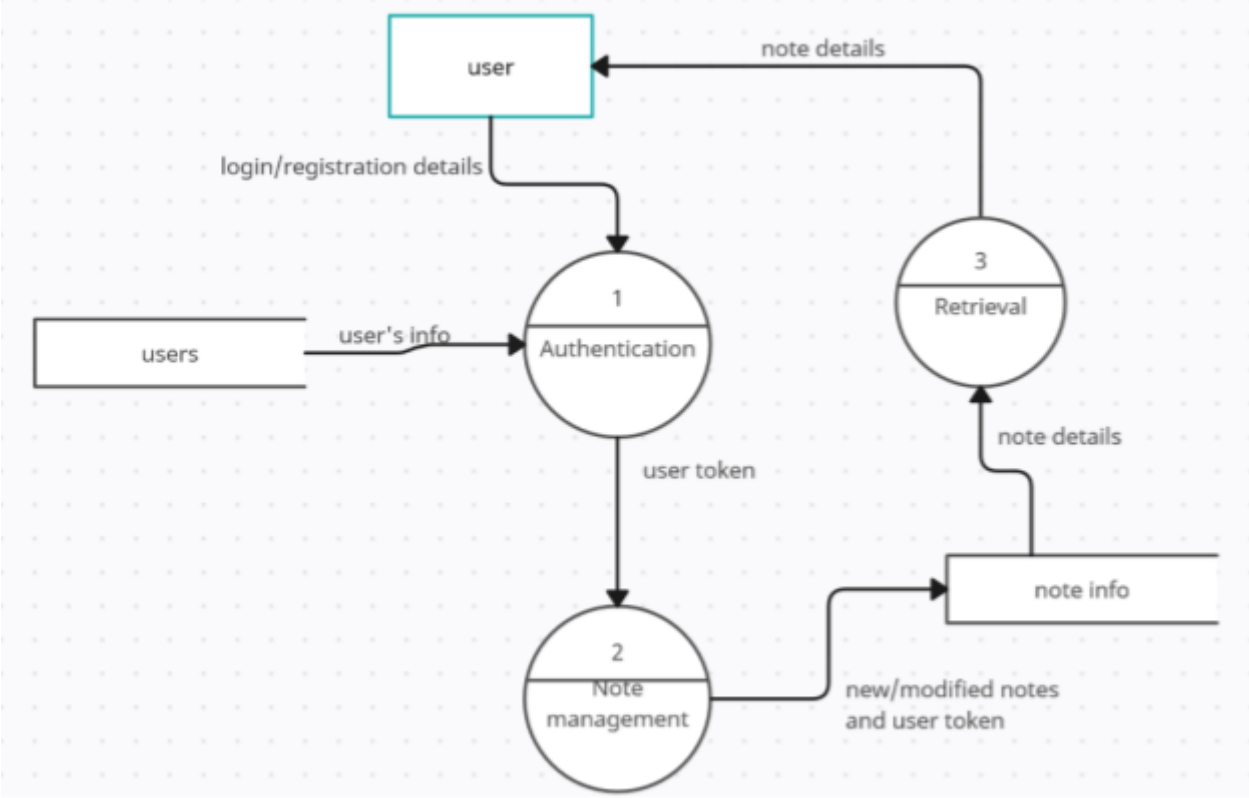

Fig 4: Level 1 DFD for User

The diagram presented above represents the Level 1 Data Flow Diagram for users depicted in the context diagram. It comprises three distinct processes, each with its own name: Authentication, Note Management, and Retrieval. The user initiates the process by providing their login/registration details to the Authentication process, which in turn authenticates the user and passes the user token to the Note Management process. The Note Management process stores the user token in the database, ensuring secure access for future use. Finally, the Retrieval process retrieves the note information from the database and generates the corresponding notes for the user, thereby facilitating a seamless user experience.

2.5.5 Data Flow Diagram (DFD) Level 1 for Backend Server

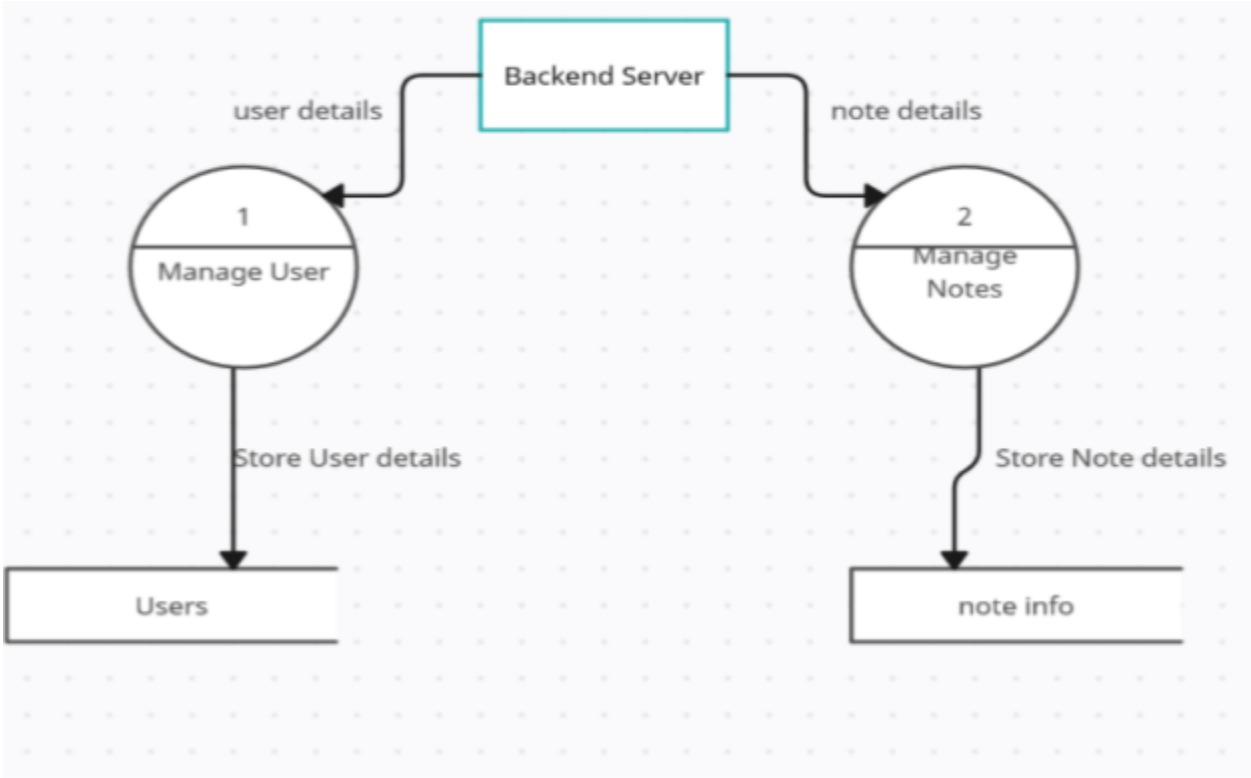

Fig 5: Level 1 DFD for Backend server

The above diagram is the level 1 DFD for the backend server of the context diagram. This level includes two processes called Manage user and Manage notes. Backend server can enter user details and note details which are processed by processes 1 and 2 respectively. The details are then stored in the Users table and note info table.

Chapter 3 : System Design

3.1 Designing System

3.1.1 Database Schema Design

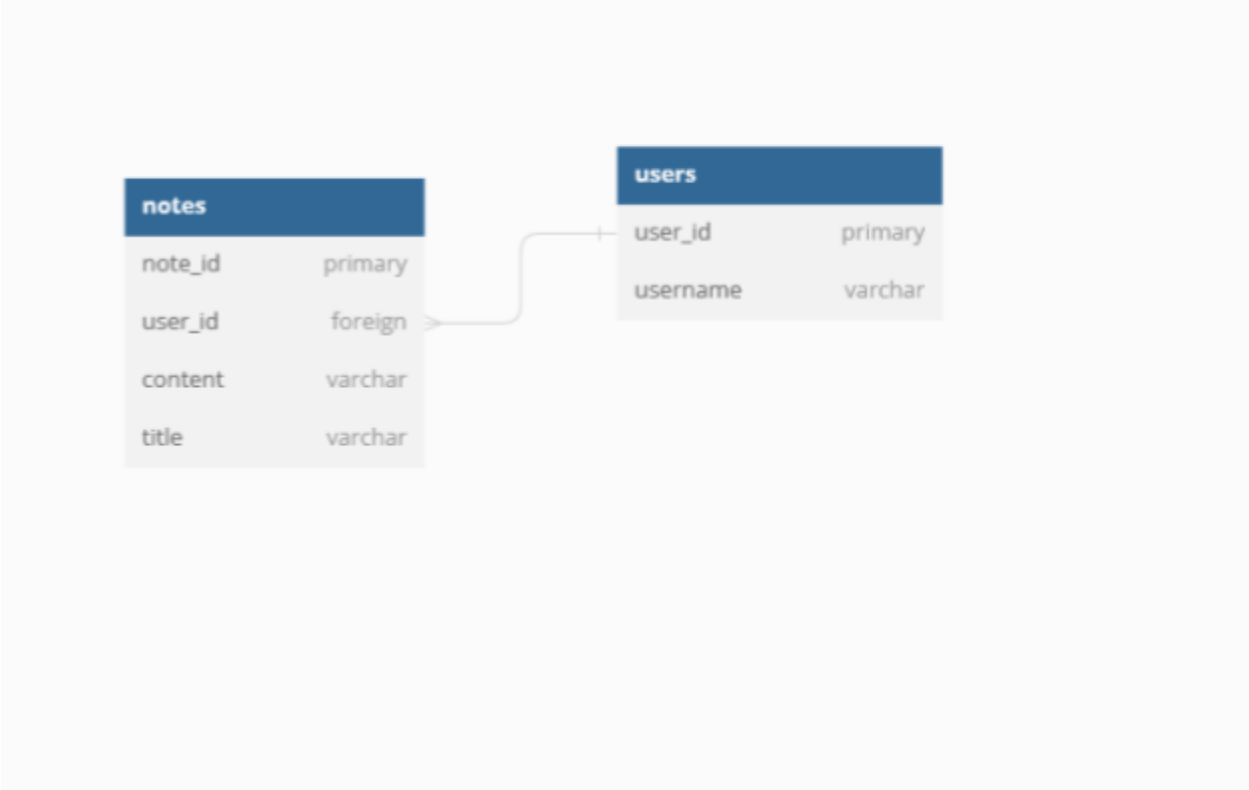

Fig 6: Database schema design

In the above figure database schema design of the online note taking system notewell is shown. There are two tables i.e. user and notes. User table contains “user_id” as the primary key and this table will store detailed information about the users.

The notes table contains note details and uses note_id as the primary key. The notes table will contain the notes written by the user along with the user's user_id as the foreign key.

3.1.2 Sequence Diagram for Note creation

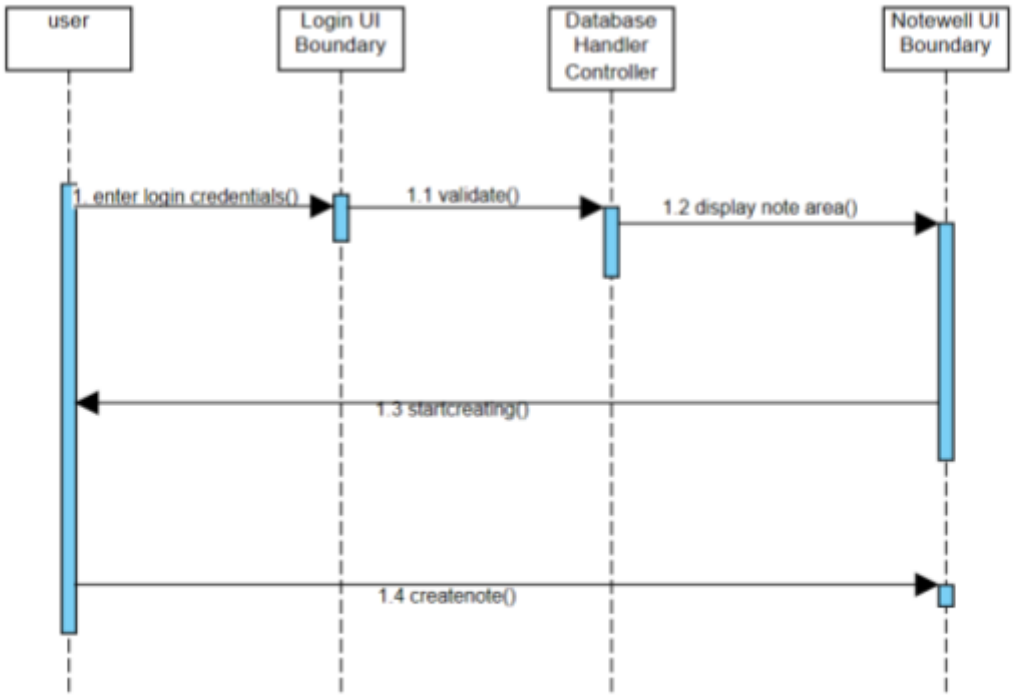

Fig 7: Sequence diagram for note creation

In the above figure there are 4 object lifelines named User, Login UI Boundary, Database Handler Controller and Notewell UI Boundary. The user sends the message "enterlogincredentials()" to the Login UI Boundary. The Login UI Boundary sends the message "validate()" to the Database Handler Controller which then sends the message "display note area()" to the Notewell UI Boundary. The Notewell UI Boundary returns the message "startcreating()" to the user. Then the user sends the message "createnote()" to the Notewell UI Boundary object.

3.1.3 Sequence Diagram for Oauth Login

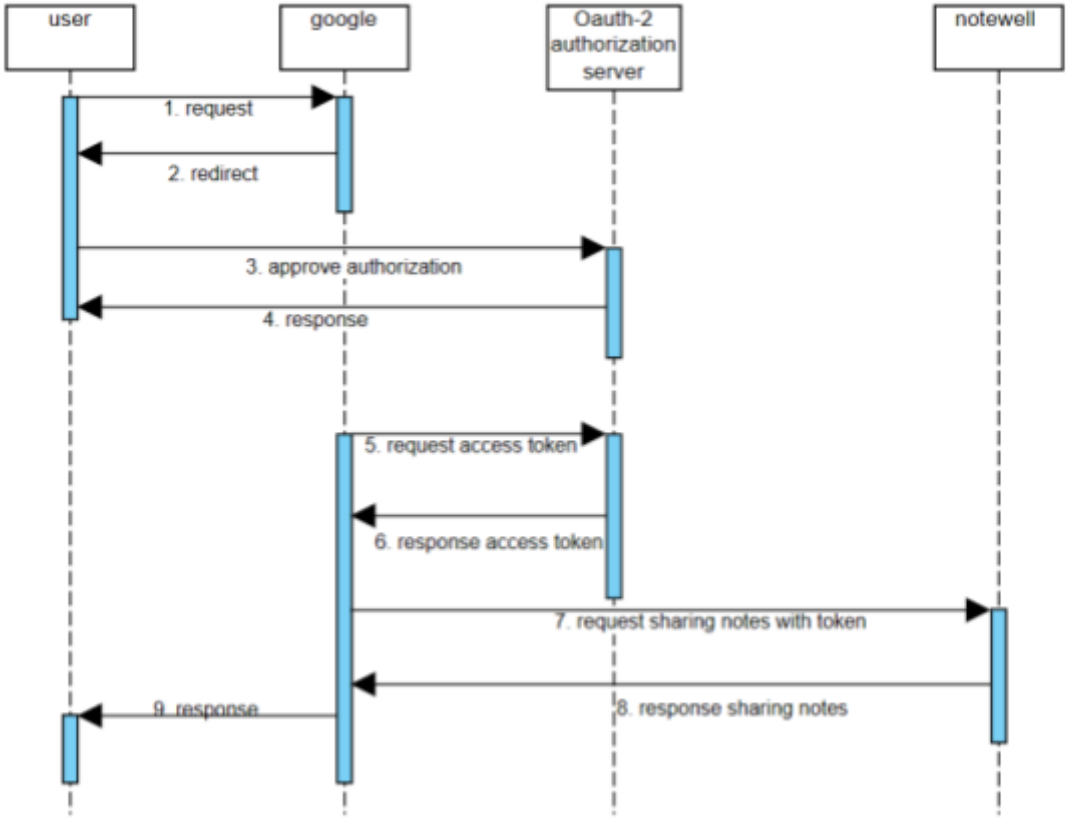

Fig 8: Sequence diagram for Oauth login

In the above figure there are 4 object lifelines named user, google, Oauth-2 authorization server and notewell. The user sends the message “request” to google which in return sends the message “redirect” redirecting to the Oauth-2 authorization server. The user then sends the message “approve authorization” to the Oauth-2 authorization server. It then returns a message “response” to the user. After the response , google then sends a message “ request access token” to the Oauth server. The Oauth server returns a message “response access token” containing the access token required to access the users notes. Google then sends a message “request sharing notes with token” to the notewell object which then returns the message “response sharing notes” to google. Google then returns a message “response” to the user containing the notes.

3.2 Methods used

3.2.1 Oauth-2

Since security is a major concern in a note taking app , this project utilizes the recently popular authorization technique called the Oauth-2. Oauth is an open authorization protocol, which

allows access to the resources of the resource owner by enabling the client applications on HTTP services. It allows sharing of resources stored on one site to another site without using their credentials. It uses username and password tokens instead.

3.2.2 How it works

The main essence of how the Oauth-2 service works is summarized in the following points.

1. The client application (the app that wants to access the user's data, in our case notewell) requests authorization from the user.
2. The user authenticates with the authorization server (the server that holds the user's credentials, in our case Google) and grants authorization to the client application.
3. The authorization server sends an access token to the client application.
4. The client application uses the access token to access protected resources (such as user data) on the resource server (the server that holds the protected data, in our case also Google).
5. The resource server validates the access token and, if it is valid, returns the requested data to the client application.
6. The client application then displays required data to the user based on the access token received.

This is the general flow when implementing an Oauth-2 system but developers can modify the flow as they see fit. There are many different ways of implementation according to the developer’s desires.

Chapter 4 : Implementation and Testing

4.1 Implementation

Implementation refers to the process of putting a plan or design into action, by taking the necessary steps to make a system, product, or service functional and available to users. It involves carrying out the design and development stages of a project, and involves tasks such as programming, testing, integration, and deployment. The goal of implementation is to ensure that the system meets the requirements and specifications set out in the design phase, and that it is functional, reliable, and easy to use. Implementation is a critical phase of the project lifecycle, as it sets the stage for ongoing maintenance, support, and optimization of the system.

4.1.1 Tools Used

We utilized a variety of programming and software development tools to create, debug, maintain, and support our web application. The tools we employed for the front-end development included HTML, CSS, JavaScript, and ReactJS. For the back-end development, we utilized Firebase. In addition to these programming languages and frameworks, we utilized various tools for debugging and testing, including Chrome DevTools and Postman. By combining these tools in a logical and effective manner, we were able to create a fully-functional and user-friendly web application that meets the needs and expectations of our users. We also utilized Figma for creating the UI designs of our web application. Figma is a powerful design tool that allows us to create and share user interface designs with ease. With Figma, we were able to quickly create wireframes, mockups, and prototypes of our application, and collaborate with team members to make design decisions. The integration of Figma into our development process allowed us to create a visually appealing and intuitive user interface that enhances the overall user experience of the web application.

4.1.1 Implementation Details of modules

The following section provides an overview of the various implementation modules utilized in the project:

4.1.1.1 React Module

The React module was implemented using the ReactJS framework for front-end development. The user interface was designed using Figma, and the layout and styling were implemented using HTML, CSS, and JavaScript. The module is responsible for rendering the user interface and handling user interactions.

To implement the module, we first designed the user interface in Figma, using a minimalist and intuitive layout that allows the user to easily view and interact with their notes. We then used ReactJS to build the front-end of the module, integrating the design elements and adding the necessary functionality to display the notes and allow the user to add, edit, or delete them.

4.1.1.2 Firebase Module

The Firebase module was implemented using Firebase for back-end development. The module is responsible for retrieving and storing data in the Realtime Database, and managing user authentication and authorization using Firebase Authentication.

To implement the module, we integrated Firebase SDKs to facilitate communication between the front-end and the Realtime Database, and implemented CRUD operations to allow the user to create, read, update, or delete notes as needed. We also used Firebase Authentication to manage user authentication and authorization, ensuring that only authorized users have access to their notes data.

The use of Firebase and Realtime Database allowed us to build a scalable and reliable back-end for our application, with real-time synchronization of data across clients. The integration of Firebase Authentication also ensured that the application is secure and user data is protected.

4.2 Test Cases for Integration Testing

After the completion of unit testing, we proceeded to perform integration testing on our application. The purpose of this phase was to verify that the different components of the system were functioning correctly when integrated with each other. To achieve this, we designed and executed a series of Test Cases for Integration Testing, which ensured that the various modules of the application were interacting with each other as expected and that the system was working as a whole. Through this phase, we were able to identify and resolve any issues related to the integration of the different modules, ensuring that the application was functioning as intended.

Presented below is a table that provides details of the testing process:

S.N	Test Case id	Test Description	Input Test Data	Expected Result	Actual Result	Status
1	TC 01	Enter Valid Login Detail (Google Credential for Oauth)	Username: xyz@google.com Password: **********	It should redirect to homepage	It displayed Home page	PASS
2	TC 02	Select Gmail Account	Selects google account	It should redirect the user to the homepage, where they can view their notes that were previously saved under their account	Successful redirection to the homepage, the application displayed the notes that were previously saved under the user's account	PASS
3	TC 03	Notes should not have only title	Enters Title without content	New notes should not be created	New note was not created	PASS
4	TC 04	Note Deletion	User clicks on Delete button	Note should be removed permanently	Note was removed permanently	PASS

Table 1: Table for details of the testing process

4.3 Test Cases for SystemTesting

The next phase of testing after Integration Testing was System Testing. The aim of this phase was to evaluate the application as a whole and test the system's compliance with the specified requirements. To achieve this, we developed a set of Test Cases for System Testing, which covered all the features and functionalities of the application. Through this phase, we were able to validate that the application was operating as intended and that it met the specified requirements. Any issues or bugs that were discovered during this phase were logged, reported, and resolved before the application was released to the end-users.

S.N	Test Case id	Test Description	Input Test Data	Expected Result	Actual Result	Status
1	TC 01	Validation of user login functionality and overall application performance.	Enter Login credential/Choose google profile. Inputs title and content.	After being redirected to the homepage, the user was able to access the user login page. Once logged in, the user had the ability to create notes and save them for future use.	After a successful login, the user was able to create notes, and the application automatically saved them to the database.	PASS

CHAPTER 5: RESULT ANALYSIS

5.1 Introduction

All the objectives of the project are met and the methods used in this project are working as expected. The users are able to register to the system and sign into the system. After signing into the system they can successfully view their previously saved notes, create new notes and edit said notes. The notes are safely stored into the firestore database and are able to be retrieved when necessary or when the user logs in.

5.2 Screenshots

The screenshots of some of the main components of the systems are shown below:

5.2.1 Sign In page

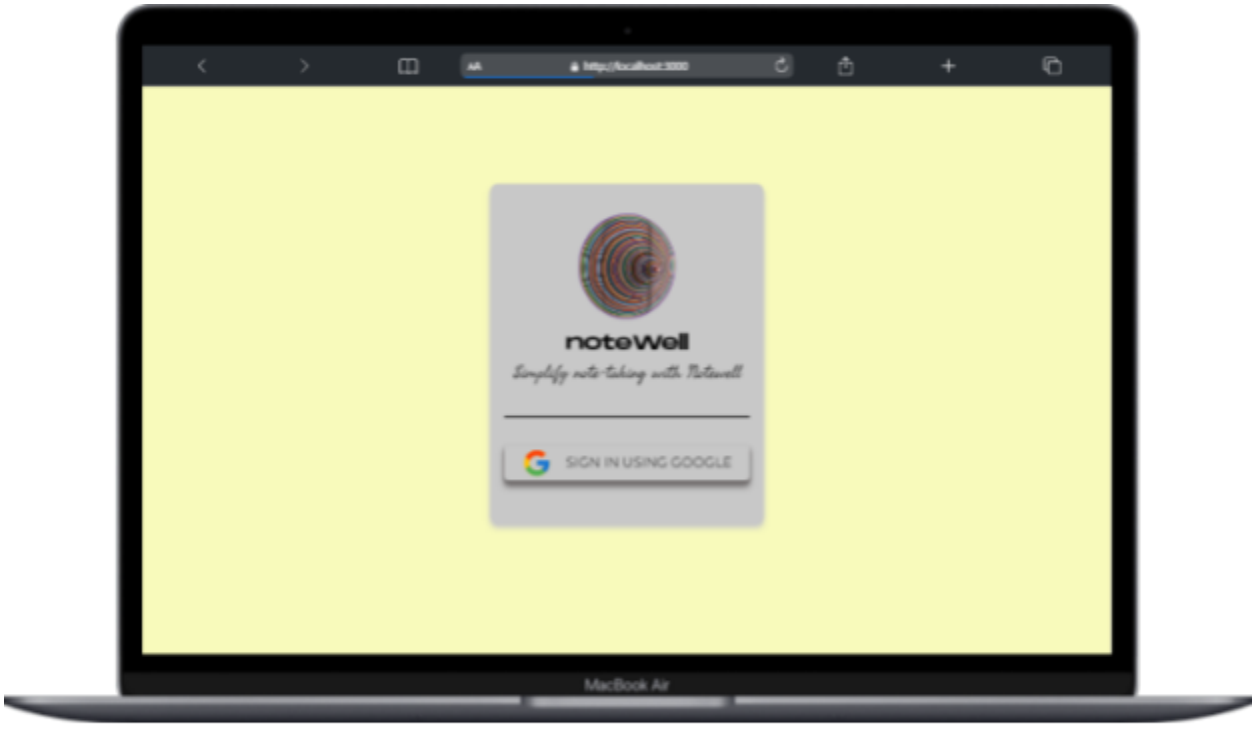

Fig 9: Sign in Page

This is the first page the user sees after visiting the website if he hasn't signed in before or hasn’t logged out from the previous session. The user can clearly see the sign in button on the webpage.

As the website is made responsive, the signin page on some other screens is shown below:

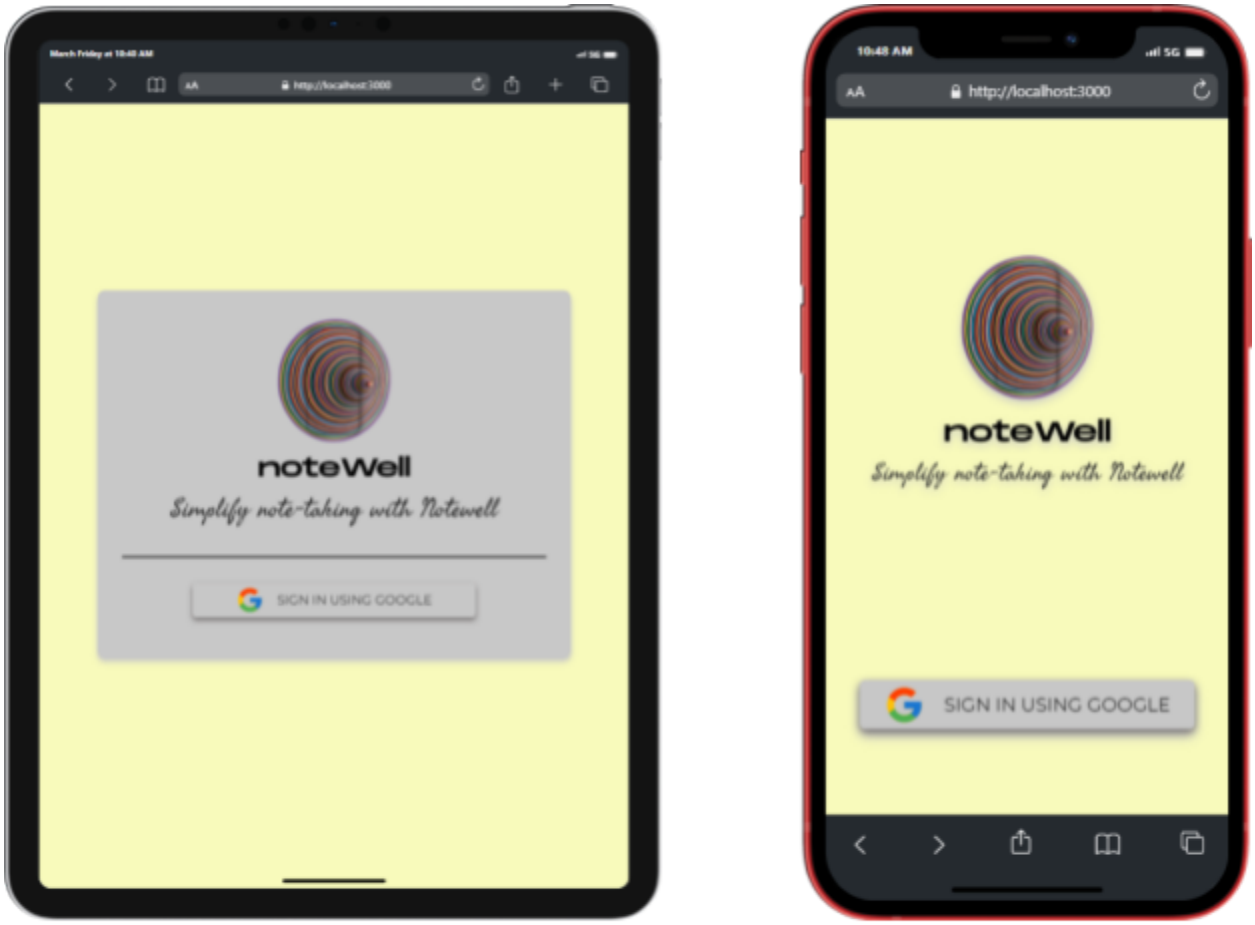

Fig 10: Sign in page on tablet and mobile

The above left picture shows the sign in page on a tablet. As we can see the sign in page is being shown as expected. While the picture on the right shows the sign in page on a typical handheld mobile.

5.2.2 Home Page

After the user has successfully signed in to the website, the user is then redirected to the home page i.e. the note creating page. Here the user can create notes, view notes and delete ntes if necessary. All of the options are clearly presented and there should be no visible confusion while using the service.

A picture of the home page is shown below on different screen sizes.

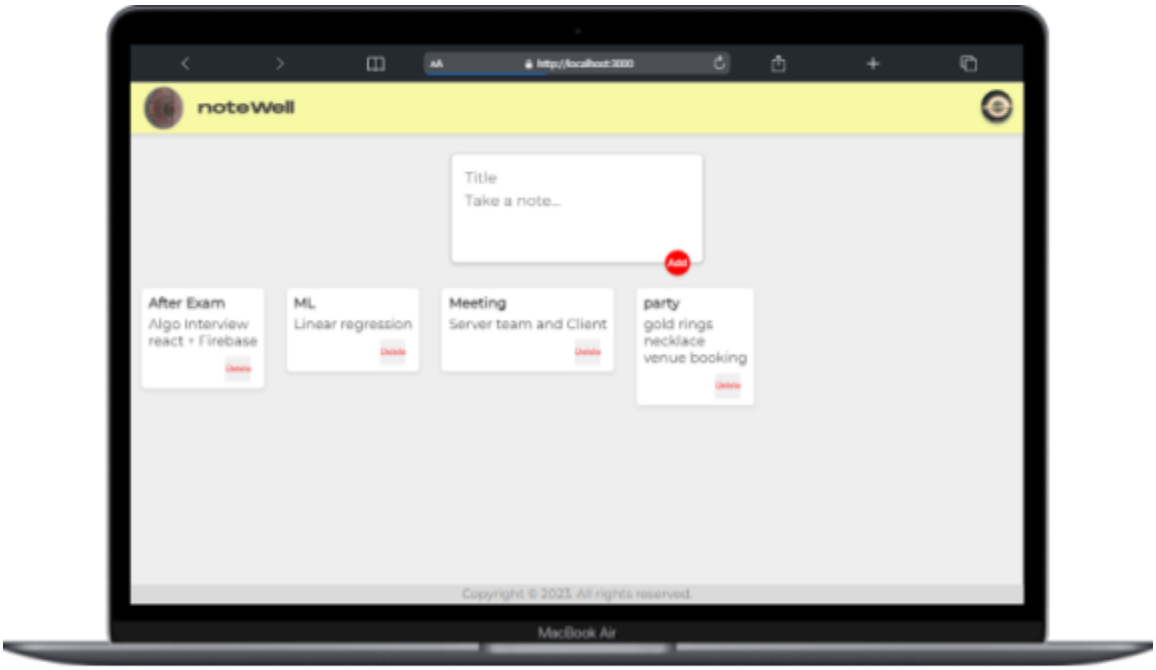

Fig 11: Homepage UI

The picture above shows the homepage on a desktop. As we can see the user has created multiple notes. The note creation is clearly visible and the notes align themselves below the creation area. The pictures below represent the homepage on a mobile and tablet simultaneously to present the responsive capabilities of the notewell service.

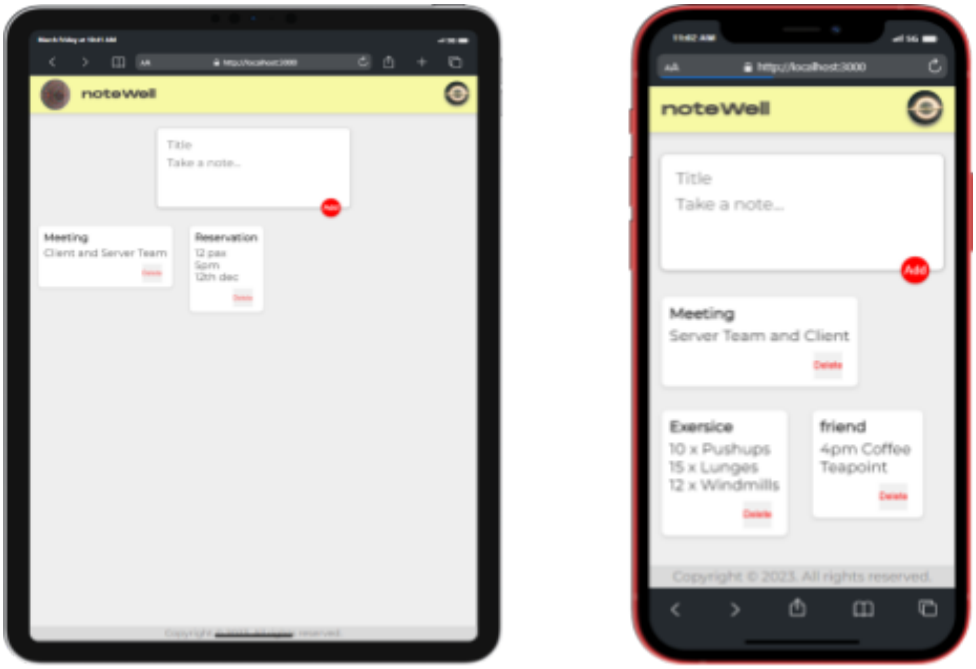

Fig 12: Homepage UI on tablet and Mobile

5.2.3 Logout Button

If the user wants to sign out after taking notes, he can easily sign out of the service by just clicking the user profile which then creates a drop down menu with the logout option. A picture of this process is shown below.

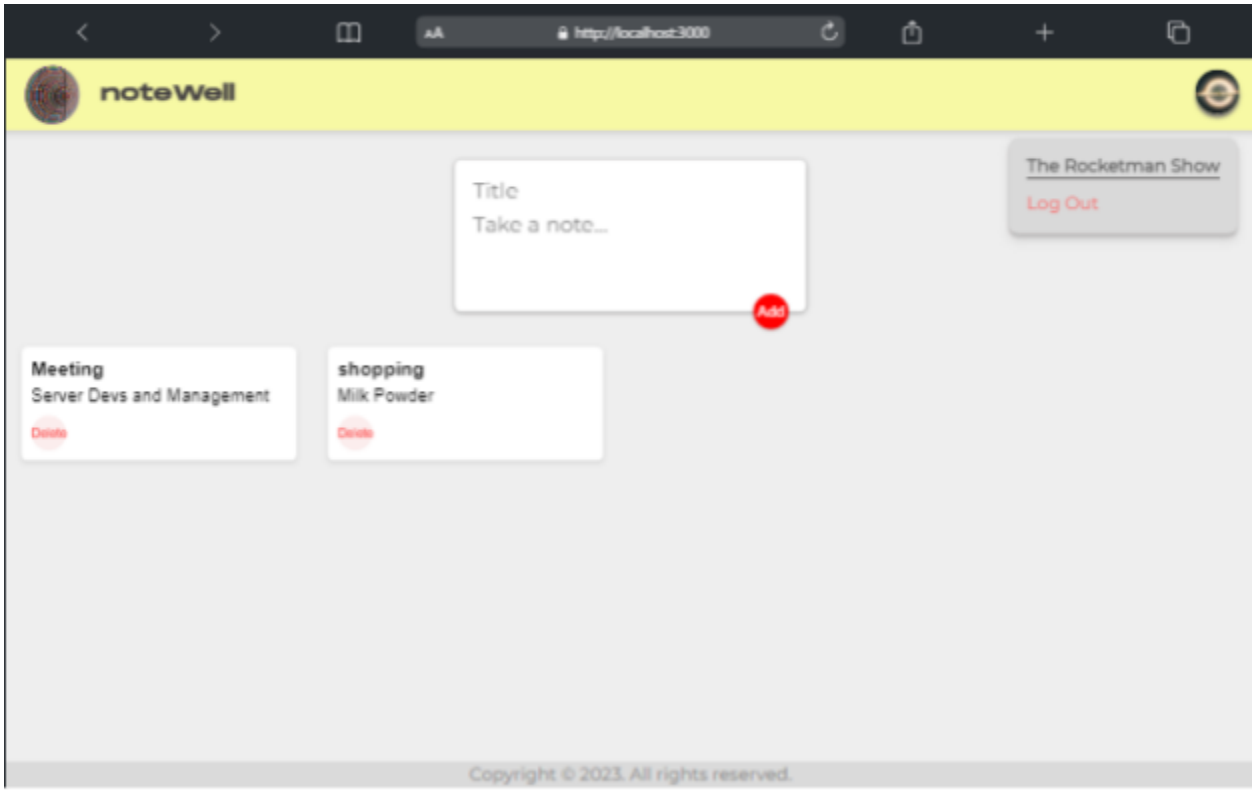

Fig 13: Logout button

As we can see the logout button is clearly visible. After pressing the logout button, the users locally stored data is cleared and is redirected to the sign in page again.

CHAPTER 6: Conclusion and Future Changes

6.1 Conclusion

The development of the Notewell application has been successfully completed. The primary objective of the application was to provide users with a simple and efficient way to create, organize and access their notes. Throughout the development process, we employed various tools and technologies to ensure the successful implementation of the application. The application was tested using various methodologies, including Unit Testing, Integration Testing, and System Testing, to ensure it functioned as intended and met the specified requirements. Overall, the application was found to be efficient, reliable, and user-friendly. As a result, we believe that the application will be well-received by the target user demographic.

6.2 Future Updates

Moving forward, there are several updates and features that can be added to the Notewell application to enhance the user experience. One potential update is the addition of a voice memo feature, which would allow users to record their thoughts and ideas on the go. Another potential feature is the ability to update notes with pictures, enabling users to capture visual information alongside their notes. Additionally, the application can be improved with collaborative note-taking, allowing users to share notes and work together on projects. These updates and features would further enhance the usability and functionality of the application, making it even more appealing to users. Overall, there is significant potential for the Notewell application to grow and evolve with future updates and features.

CHAPTER 7: References

[1] "Sequence Diagram of Interaction with Our OAuth2 Authorization RESTful Feed Sharing Service." ResearchGate, https://www.researchgate.net/figure/Sequence-Diagram-of-Interaction-with-Our-OAuth2-authorization-RESTful-Feed-Sharing-Service_fig3_272823002

[2] Bihis, Charles. Mastering OAuth 2.0. Packt Publishing, 2015.

[3] Tutorialspoint. OAuth 2.0 Overview. Available online: https://www.tutorialspoint.com/oauth2.0/oauth2.0_overview.htm

[4] Stack Overflow. How to send value from one component to another. Available online: https://stackoverflow.com/questions/61266952/how-to-send-value-from-one-component-to-another

[5] React Documentation. Available online: https://reactjs.org/docs/getting-started.html

[6] Firebase Documentation. Available online: https://firebase.google.com/docs