OPEN SOURCE CODING

OPSC7312

POE PART 1

NOTEFY
A NOTE TAKING APP

PLANNING AND DESIGN

BY

Dylen Lutchana - ST10083924

Ashendrin Govender - St10083844

Table of contents

Introduction

Notefy is a state-of-the-art, user-friendly note-taking application designed to satisfy the needs of both personal and business users. The application's goal is to provide users with a seamless and continuous note-taking, organising, and management experience. This is accomplished by incorporating sophisticated features like offline mode, synchronisation across several devices, real-time collaboration, secure access, reminder and notification creation, and list creation. Notefy wants to be the go-to option for individuals and groups looking for a strong and flexible solution to manage their thoughts, plans, and projects. It accomplishes this by combining the best elements of well-known note-taking programs like Google Keep, Evernote, and OneNote. App Name: Notefy

Icon Design:

Innovative features included in our App: Multi-Device Synchronization: The ability to synchronise notes across several devices so consumers can access their data from anywhere.

Setting Reminders: In addition to receiving real-time notifications, the user will be able to set and generate reminders. With Notefy, users may set up reminders for their tasks, select a precise time and date, and receive notifications at the exact moment they want. They can also make reminders and receive real-time notifications. This ensures that they won't ever forget a crucial deadline.

Make lists: To help with organising chores, shopping, to-do lists, and other things, the user can make lists or checklists. To ensure efficient job management, these lists can be used in conjunction with notifications and reminders.

User Collaboration: Many users can work on the same Note in real time. A link will be sent to the collaborators email and once the link is accepted users can collaborate on the same document.

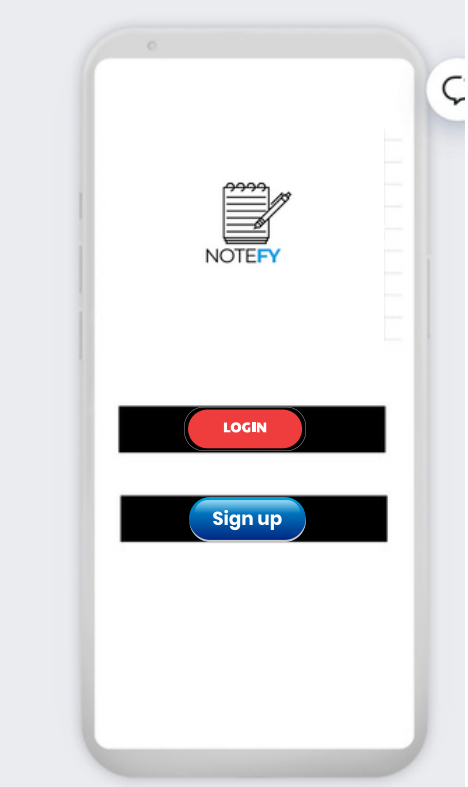
Password Protection Notes: The user will can add password protection to their private Notes. The user will be able to lock there note and access it with their password or facial recognition Offline Mode: Notefy enables users to create and modify notes without internet
connection. Upon re-establishing an internet connection, the application will instantly synchronise any modifications performed while in offline mode. This functionality is crucial for customers who travel frequently.

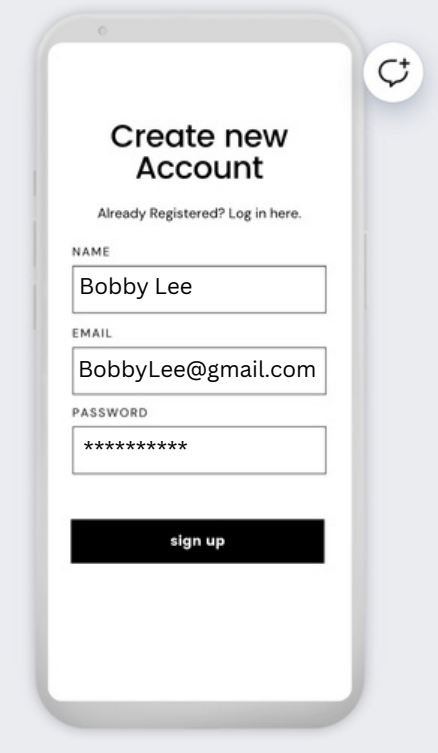
These features will guarantee that Notefy meets and surpasses user expectations by providing a full solution for organising notes and tasks in an efficient, secure, and collaborative environment.

RESTful API

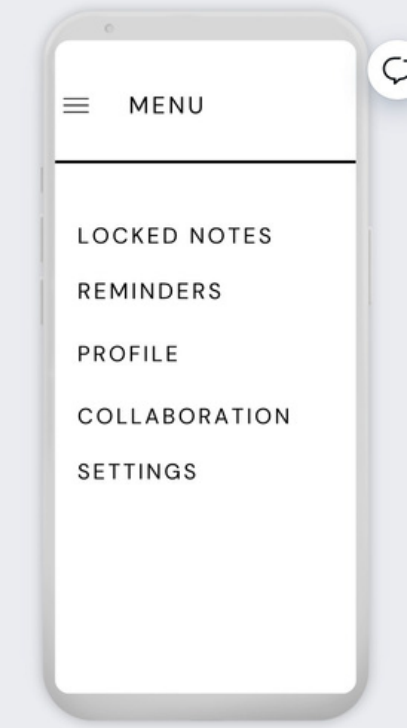
You must provide your app internet rights before you can connect it to a RESTful API. Your application will send and receive data to and from the API using an HTTP client library like Retrofit. The application will send credentials to the API for user registration and login, keeping the authentication token that is returned. User settings will work similarly, and notes made or modified within the app will be synced with the API. Data will be converted to and from JSON format for these interactions, and error handling will ensure that the system functions properly even in the event of network issues.

UI Design



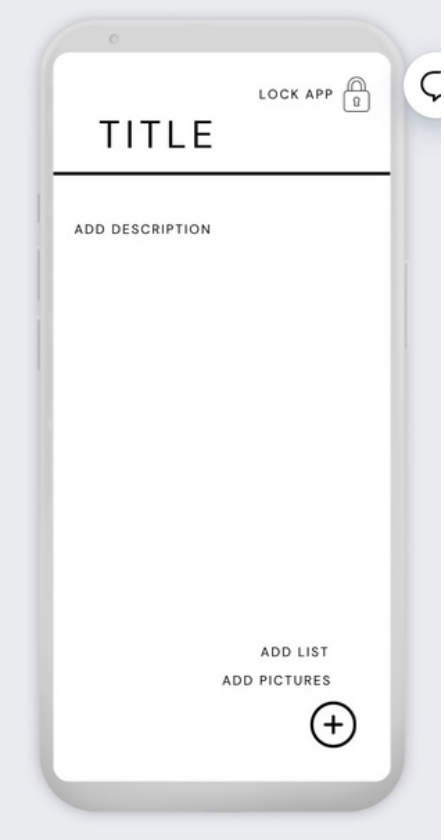


Welcome to the Notefy App, if you are an existing User Login or if you are a new user Sign Up. Sign up by entering your name, email and password.

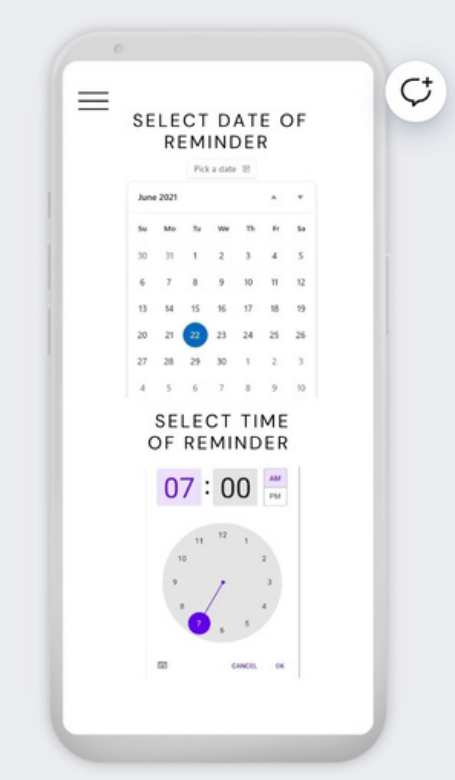


When you login to the app you will be taken to the home page where you can create a new note, create a reminder or view your previously stored notes and reminders.

By Clicking the 3 lines on the top you will be taken to the menus page where you can open settings, locked notes, profile etc.

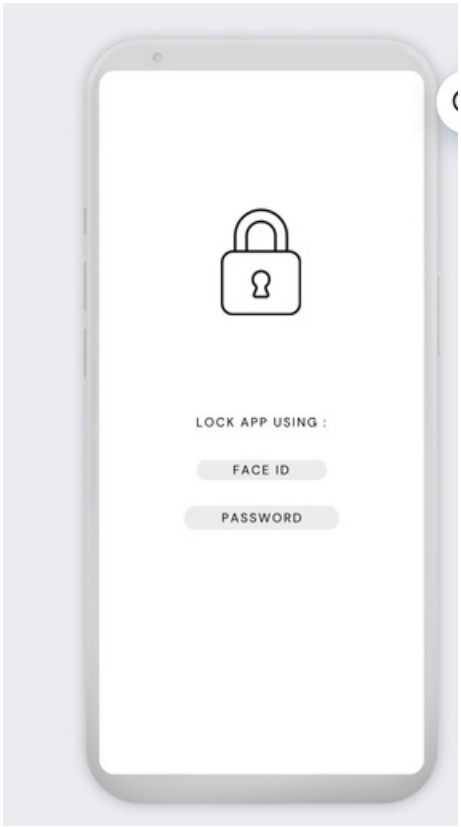


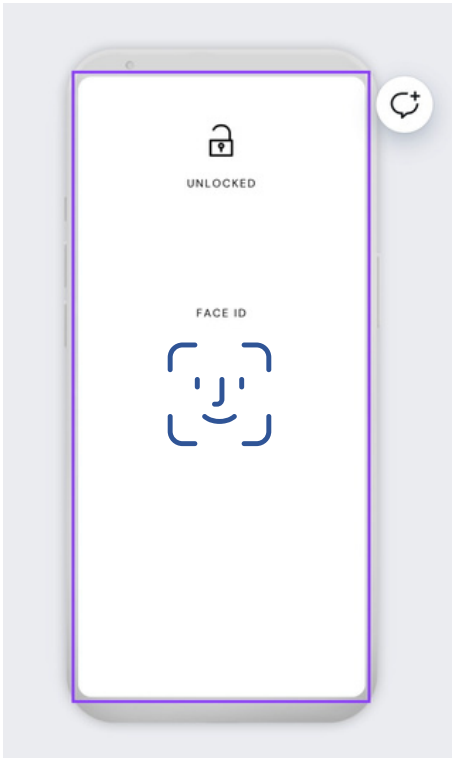
When you click on new note, you are taken to this page where you can create your own note. You can add pictures to your notes.

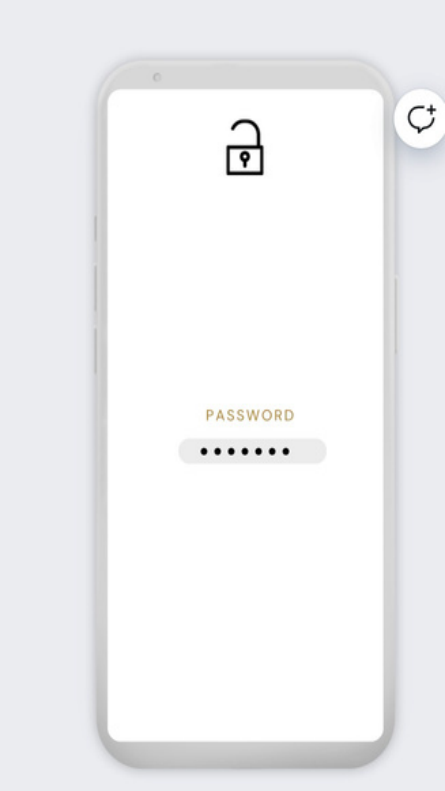


When you click on Reminders: You can create your own reminders.

You can schedule when you want to receive the reminders notification by setting a date and time.


LOCK APP USING :
FACE ID
PASSWORD

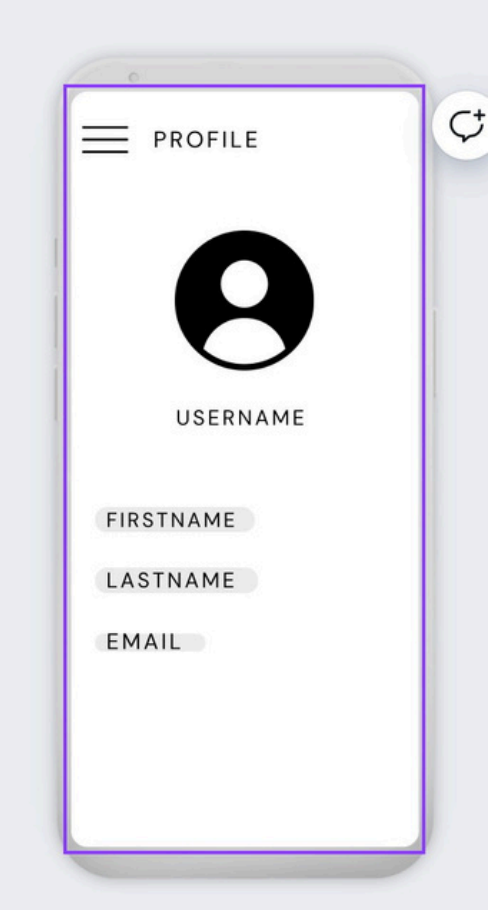

UNLOCKED
FACE ID



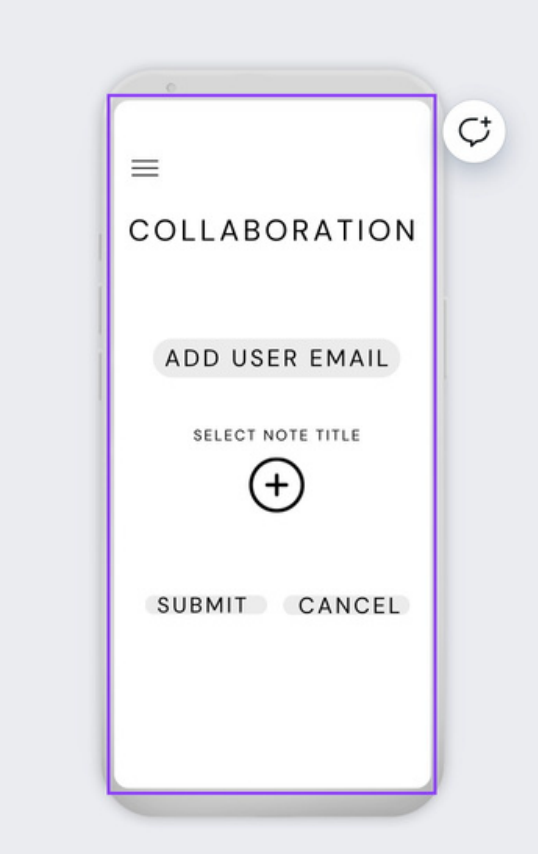
An innovative feature we have added is: Password protected Notes.

The user can Lock a note they wish to keep private, via password or facial recognition.

The user can also chose to lock the app. And they can gain access to the app by entering a password or via facial recognition.



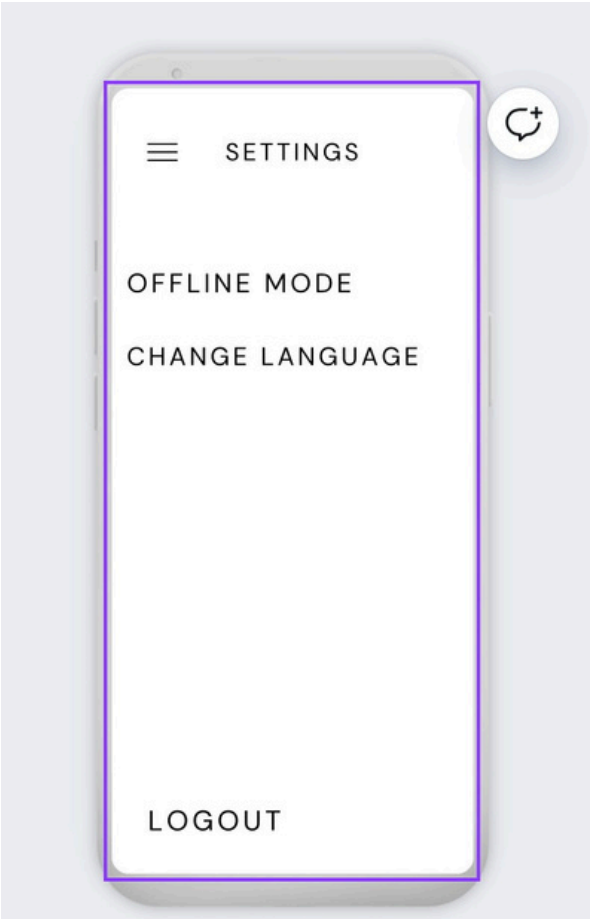
The user can view there profile from Menu. They can update their profile if they wish.



Innovative Feature: Collaboration

Multiple users Can collaborate on a Note. 1 user must enter the other persons email with whom they wish to collaborate with. That user will accept the link and can access the Note.

Then they can begin their collaboration.

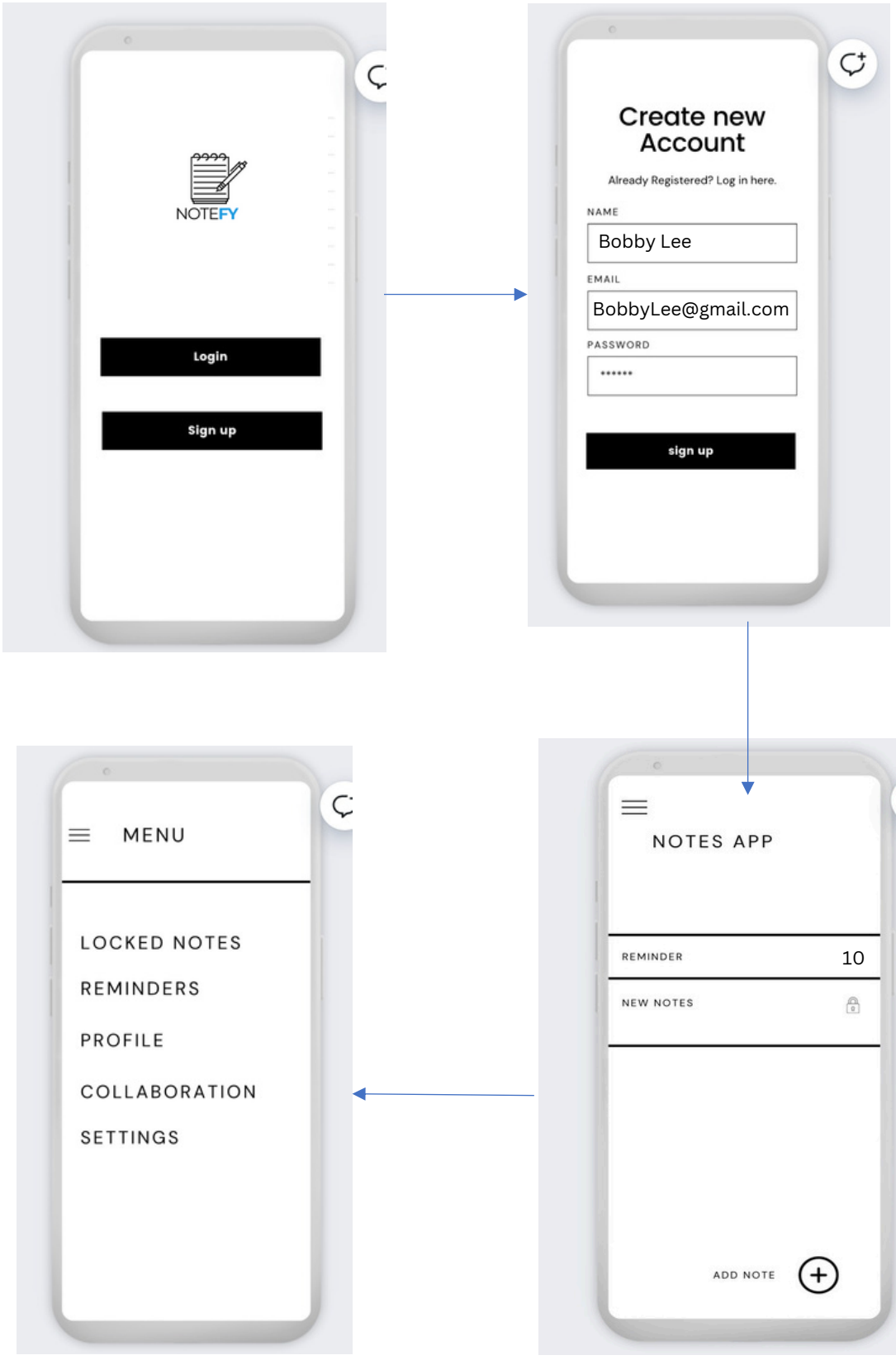


The user can access settings via Menu. They can use the app when they don’t have access to internet connection by selecting Offline Mode. When they have access to Wi-Fi again their work will sync.

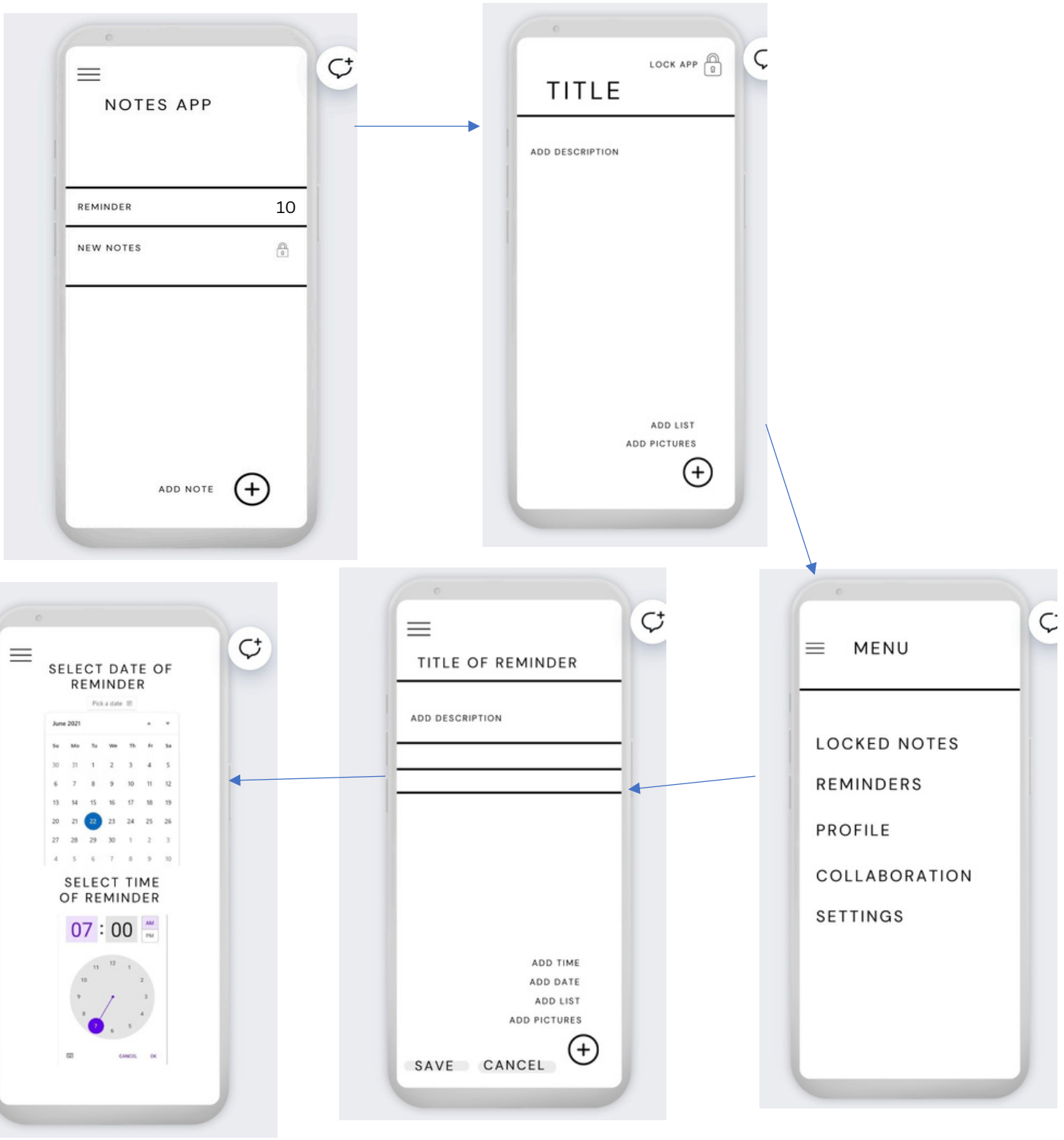
The user will also be able to change the apps language between English and Afrikaans through the Change Language Button

The User will be able to Logout from the app.

User Naviga)on Diagram



Create new note




LOCK APP USING :
FACE ID
PASSWORD
PROFILE
USERNAME
FIRSTNAME
LASTNAME
EMAIL
MENU
LOCKED NOTES
REMINDERS
PROFILE
COLLABORATION
SETTINGS
COLLABORATION
ADD USER EMAIL
SELECT NOTE TITLE
SUBMIT
CANCEL
SETTINGS
OFFLINE MODE
CHANGE LANGUAGE
LOGOUT

API

The Notefy API will be a RESTful service that can manage basic functions like user login, note management, synchronisation, real-time messages, and setting administration. This API will facilitate communication between the backend database and the Android Studio front end. This will guarantee that the software functions flawlessly across all platforms and online environments.

The Spring Boot framework, which works well with Android Studio to create RESTful services, will be used to build the Java API. Because of its reputation for robustness, scalability, ease of use with many databases, and security features, Spring Boot is a great choice to satisfy Notefy's needs. This framework provides a nice development experience for both the front and back sides of the application, and it integrates well with Android Studio. Database: Due of their dependability and ability to manage intricate searches, MongoDB or Firebase will serve as the primary database.

Security: Authorisation and registration for the API will be protected using JWT (JSON Web Tokens). Sensitive information, including biometric data and passwords, will be encrypted using industry-standard protocols prior to storage. In compliance with internet and mobile security best practices, the encryption process guarantees that all sensitive data within the app is securely protected. Furthermore, all devices that use Notefy will have seamless authentication and data protection thanks to the integration of these security features with Android Studio.

API Endpoints and Functionality

User Authentication

Endpoint: /auth/register

Method: POST

Description: Registers a new user with Single Sign-On (SSO) support.

Request Data: { "email": "user@example.com", "password": "hashedPassword", "provider": "google/facebook/etc.", "biometricData": "encryptedBiometricData" }
Response Data: { "status": "success", "userId": "uniqueUserId", "token": "JWT token"}

Endpoint: /auth/login

Method: POST

Description: Logs in a user using SSO or email/password and provides a JWT token.

Request Data: { "email": "user@example.com", "password": "hashedPassword", "biometricData": "encryptedBiometricData" }
Response Data: { "status": "success", "token": "JWT token" }

Endpoint: /auth/verify

Method: POST

Description: Verifies the user's biometric data during login.

Request Data: { "userId": "uniqueUserId", "biometricData": "encryptedBiometricData" }

Response Data: { "status": "verified" }

Note Management

Endpoint: /notes

Method: GET

Description: Retrieves all notes for the authenticated user.

Request Data: Authorization: Bearer <JWT token>

Response Data: { "notes": [{ "noteId": "note1", "title": "Note Title", "content": "Note Content", "tags": ["tag1", "tag2"], "createdAt": "timestamp", "updatedAt": "timestamp" }] }

Endpoint: /notes

Method: POST

Description: Creates a new note for the user.

Request Data: { "title": "New Note", "content": "Note Content", "tags": ["tag1", "tag2"], "locked": false }

Response Data: { "status": "created", "noteId": "newNoteId" }

Endpoint: /notes/{noteId}

Method: PUT

Description: Updates an existing note.

Request Data: { "title": "Updated Title", "content": "Updated Content", "tags": ["tag1", "tag2"], "locked": true }
Response Data: { "status": "updated" }

Endpoint: /notes/{noteId}

Method: DELETE

Description: Deletes a specific note.

Request Data: Authorization: Bearer <JWT token> Response Data: { "status": "deleted" }

Requirements Func)onal Requirements:

User authentication: Use Single Sign-On (SSO) to make it easy for users to sign up and log in.

For safe user authentication, use biometric identification like face recognition.

Allow people to sign up and log in with passwords they already have from big providers like Google.

Adding and Editing Notes: Users should be able to add, change, remove, and arrange notes.
Text, pictures, tasks, and voice records can all be added to notes.

For extra Security, users can lock certain notes with a password or fingerprints.

Collaboration: Allow real-time collaboration on notes so that more than one person can change a note at the same time.
Keep track of the changes that different people have made and provide version control.
Synchronisation: Make sure that all of your notes are in line on all of your devices at the same time.
Offline mode: so that people can work on their notes even when they're not connected to the internet. Sync changes when you join again.
RoomDB or SQLite can be used to store files offline on mobile devices.

Reminders and Notifications: Let users set reminders in their notes, change the time and date, and get real-time alerts. Set up a push notification system to get tips and changes in real time.

List Making: Users can make lists, like food lists and to-do lists, and cross things off as they're done.
Add lists to the note system to let users know about due dates or jobs that are coming up.
Support for multiple languages: offer the app's design in at least two South African languages besides English. The 2 languages will be Zulu and Afrikaans
Custom Settings: Give users ways to change how the app looks (for example, dark mode), how notifications work, and Users information.
Users should be able to change their account details, language settings, and security options all within the app.
Adding support for the REST API:

You can store and get user data by connecting the app to a REST API that talks to a private database.

All CRUD (Create, Read, Update, and Delete) actions for notes and user settings should be possible through the API.

Non-Func)onal Requirements:

Performance: Even if you have a lot of notes, the app should start quickly and work well.

Optimise the use of resources to make sure that the app runs smoothly on a range of mobile devices.

Security: Put in place strong security measures, such as encrypting private data.

Make sure fingerprint authentication is safe and effective so that people who aren't supposed to be there can't get in.

Scalability: Make sure the app can handle a lot of users and a lot of info.

Make sure that synchronisation and real-time teamwork work well, even if there are a lot of people.

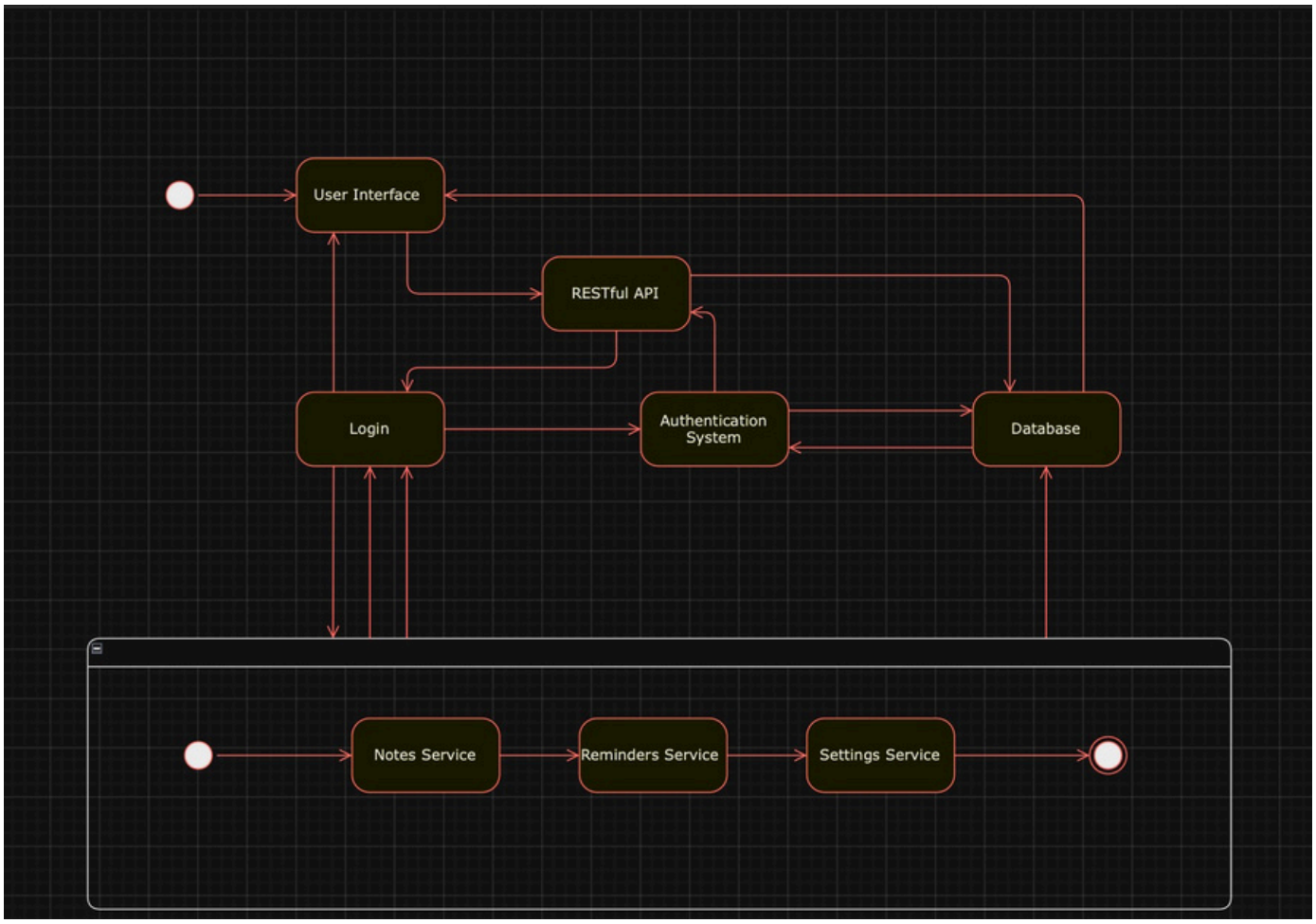
Usability: The app should have an easy-to-understand layout that lets users quickly find their way around and use its functions.

Backend and Database: A solid system, like Node.js or Django, should be used to build the REST API.

If you want to keep user info safe, use a large database like MongoDB.

For push notifications, we would use Firebase Cloud Messaging (FCM)

UML Diagram


User Interface
RESTful API
Login
Authentication System
Database
Notes Service
Reminders Service
Settings Service

Detailed Lis)ng of the data that the app needs to capture from the user and store, including data types

User Information

• UserID: String (Unique identifier for each user, typically a UUID)
• Email: String (User's email address, used for login and communication)
• Password: String (Encrypted password for user authentication)
• BiometricData: Binary (Stored encrypted data for fingerprint or facial recognition)
• Username: String (Optional, for display purposes within the app)
• PhoneNumber: String (Optional, for communication or 2FA purposes)
• ProfilePicture: Binary (Optional, for user's profile picture)
• LanguagePreference: String (Stores the preferred language for the UI)

2. Note Data

• NoteID: String (Unique identifier for each note, typically a UUID)
• UserID: String (Foreign key linking the note to its creator)
• Title: String (Title of the note)
• Content: Text (Main content of the note, could be plain text or rich text)
• CreationDate: DateTime (Timestamp of when the note was created)
• LastModifiedDate: DateTime (Timestamp of the last modification)
• Locked: Boolean (Indicates whether the note is password-protected or not)
• Password: String (Optional, encrypted password for locking the note)
• Tags: Array<String> (Optional, for categorizing notes)
• Attachments: Array<Binary> (Optional, for storing images, PDFs, etc.)

3. List Data

• ListID: String (Unique identifier for each list, typically a UUID)
• UserID: String (Foreign key linking the list to its creator)
• Title: String (Title of the list)
• Items: Array<Object> (An array of list items, each containing ItemID, Description, Completed status

    • ItemID: String (Unique identifier for each list item)
    • Description: String (Text description of the list item)
    • Completed: Boolean (Status of whether the item is completed or not)

• CreationDate: DateTime (Timestamp of when the list was created)
• LastModifiedDate: DateTime (Timestamp of the last modification)

4. Reminder Data

• ReminderID: String (Unique identifier for each reminder, typically a UUID)
• UserID: String (Foreign key linking the reminder to its creator)
• NoteID: String (Optional, foreign key linking the reminder to a note, if applicable)
• ListID: String (Optional, foreign key linking the reminder to a list, if applicable)
• Title: String (Title of the reminder)
• Description: String (Detailed description of the reminder)
• ReminderDateTime: DateTime (The scheduled time and date for the reminder)
• Recurring: Boolean (Indicates if the reminder is recurring)
• RecurringInterval: String (Optional, interval for recurrence, e.g., daily, weekly)
• NotificationSent: Boolean (Status of whether the reminder notification has been sent)

5. Synchronization Data • SyncID: String (Unique identifier for each sync operation, typically a UUID) • UserID: String (Foreign key linking the sync operation to the user) • LastSyncTime: DateTime (Timestamp of the last successful sync) • PendingChanges: Array<Object> (An array of changes that need to be synced, each containing ChangeID, ChangeType, EntityID, Timestamp)

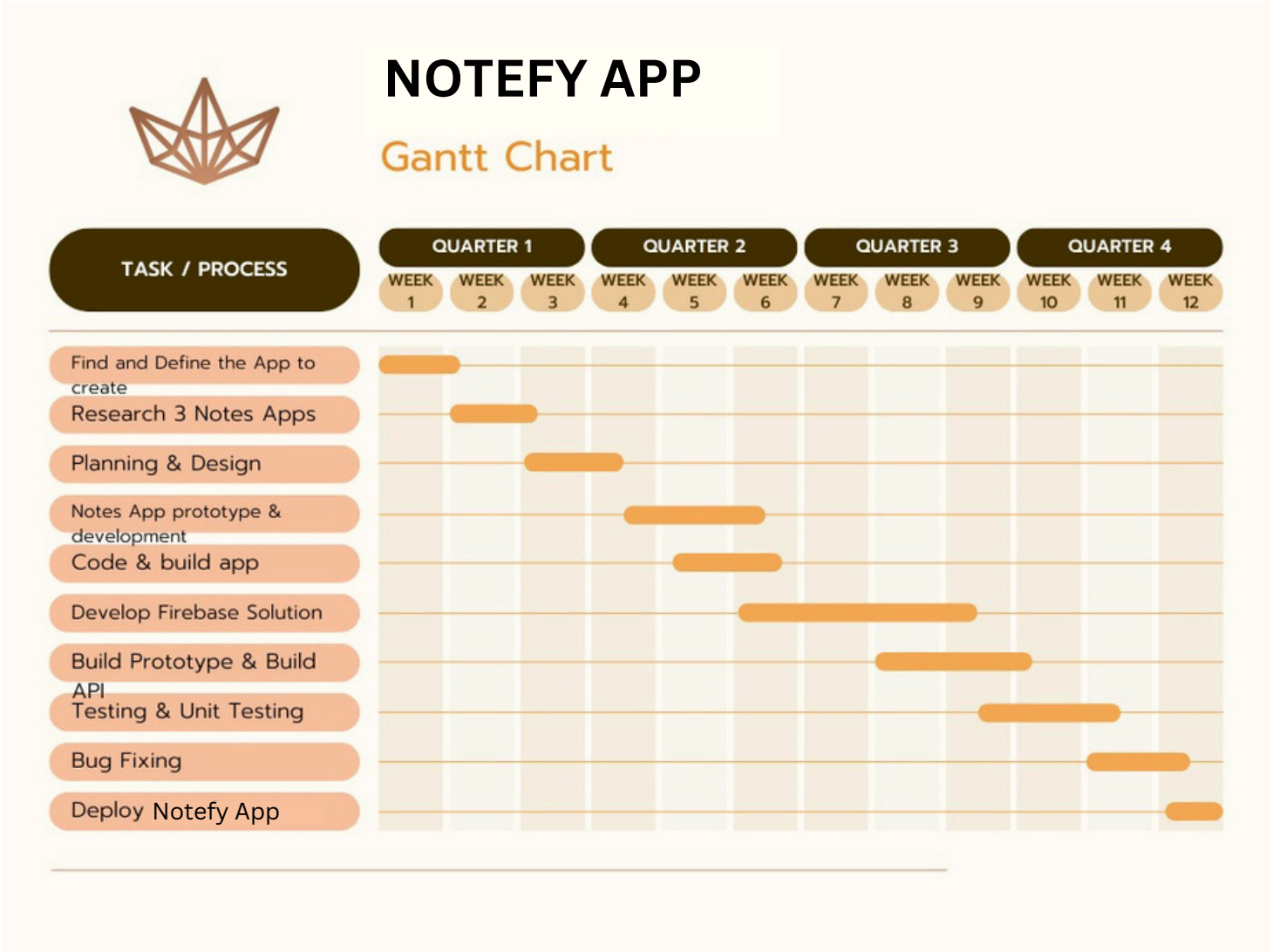
6. Settings Data

• UserID: String (Foreign key linking the settings to the user)
• Theme: String (Stores the current theme preference, e.g., dark, light)
• NotificationEnabled: Boolean (Indicates if notifications are enabled)
• LanguagePreference: String (Stores the preferred language for the UI)
• SyncOverWiFiOnly: Boolean (Indicates if sync should occur only over WiFi)
• AutoLockTimeout: Integer (Time in minutes before the app auto-locks for security)

7. Authentication Data (For SSO)

• SSOProvider: String (Name of the Single Sign-On provider, e.g., Google)
• SSOToken: String (Token received from the SSO provider)

GANTT Chart


NOTEFY APP
Gantt Chart
TASK / PROCESS
QUARTER 1
QUARTER 2
QUARTER 3
QUARTER 4
WEEK 1
WEEK 2
WEEK 3
WEEK 4
WEEK 5
WEEK 6
WEEK 7
WEEK 8
WEEK 9
WEEK 10
WEEK 11
WEEK 12
Find and Define the App to create
Research 3 Notes Apps
Planning & Design
Notes App prototype & development
Code & build app
Develop Firebase Solution
Build Prototype & Build API
Testing & Unit Testing
Bug Fixing
Deploy Notefy App

Conclusion

It is anticipated that Notefy's development would provide a highly functional and secure note-taking application that meets the various needs of its users. The application will leverage the robust features of Android Studio and integrate a RESTful API built with Spring Boot to deliver a consistent experience across numerous devices. Important features like offline capability, biometric identification, multi-device synchronisation, and real-time collaboration will further establish Note Sync as an exceptional tool for both personal and professional use.

To sum up, Note Sync aims to be a cutting-edge, intuitive, and version-specific note-taking application that not only satisfies but also exceeds the expectations of its users. In the competitive world of digital note-taking apps, a solution that combines innovative features, strong security, and a reliable backend will stand out.

(2550 Words)

References

1. Bigelow, S.J. and Gillis, A.S. (2024) ***What is Rest Api (restful API)?***, ***App Architecture***. Available at: https://www.techtarget.com/searchapparchitecture/definition/RESTful-API (Accessed: 22 August 2024).
2. Juviler, J. (2024) ***Rest apis: How they work and what you need to know***, ***HubSpot Blog***. Available at: https://blog.hubspot.com/website/what-is-rest-api (Accessed: 22 August 2024).
3. Keeports, A. (2024) ***Understanding how to utilize a REST API***, ***Cleo***. Available at: https://www.cleo.com/blog/blog-knowledge-base-what-is-rest-api (Accessed: 22 August 2024).
4. Unknown (2023a) ***Declare your app's data use : App quality : android developers***, ***Android Developers***. Available at: https://developer.android.com/privacy-and-security/declare-data-use (Accessed: 22 August 2024).
5. Unknown (2023b) ***Why a multi-function ipaas is the future of apis***, ***Software AG***. Available at: https://www.softwareag.com/en_corporate/resources/api/wp/multi-function-ipaas.html?utm_source=google&utm_medium=cpc&utm_campaign=aim_api-intg&utm_region=hq&utm_subcampaign=stg-1&utm_content=stg-1_whitepaper_why-a-multifunction-ipaas-is-future-apis&gad_source=1&gbraid=0AAAADLqeyVgHUgjVi9fSVOVHLMri1X83&gclid=Cj0KCQjww5u2BhDeARIsALBuLnPJaVfv8Bpnr4WYTWyQKgn1SQUPQo5oEUqfhGHyUgd8O8_C5O6T5XUaAiMAEALw_wcB (Accessed: 22 August 2024).
6. Yuskevych, M. (2024) ***A guide to writing non-functional requirements for an app: Perpetio***, ***Mobile Application Design & Development Expertise***. Available at: https://perpet.io/blog/a-guide-to-writing-non-functional-requirements-for-your-mobile-app/ (Accessed: 22 August 2024).
7. To create the Mock up/ App Prototype and the Gantt Chart we used Canva: (No date) ***Free design tool: Presentations, video, social media | CANVA***. Available at: https://www.canva.com/ (Accessed: 22 August 2024).
8. To create UML we used Draw.io: https://app.diagrams.net

Project Development with AI Tools

The Notefy software evolved from conceptualising needs to implementing sophisticated features including real-time collaboration, biometric identification, and cross-device synchronisation. In this procedure, AI technologies significantly improved productivity,

accuracy, and project
quality.

Project requirements understanding:

The initial problem was to adequately define and comprehend project requirements. AI technologies were used to analyse and comprehend the complicated characteristics and functions to be implemented. AI assisted in organising the project to address all important components by breaking down requirements. The process involved defining API endpoints, data models, and security procedures. We used AI to acquire a greater understanding of the project scope and guarantee no important features were omitted. The early usage of AI formed a firm basis for the development process.

Code Snippet Creation and Improvement:

In the development phase, AI techniques were used to generate code snippets, especially for Spring Boot backend development. AI established RESTful API code structures, integrated JWT for safe authentication, and encrypted data. Our AI-generated snippets provided a foundation for customising the code to match the needs of Notefy. The AI proposed efficient offline synchronisation using Firebase and safe user verification using biometric data.

This sped development and eliminated mistakes by using AI to generate code that followed best practices in software development.

AI technologies proved useful in debugging and resolving issues throughout the development phase.

Integrating features like real-time alerts, SSO login, and multi-language support was challenging, leading to complications. AI was utilised to swiftly detect and provide remedies for data security and API integration issues. AI helped optimise synchronisation logic for offline data and cloud database, maintaining data consistency across devices. This feature has considerably reduced debugging time and helped the team maintain a constant development pace.

AI tools were used to prepare project documentation and reports, in addition to coding. This involved creating concise explanations of project objectives, techniques, and outcomes. AI facilitated systematic information organisation, resulting in a professional and clear study report. In the AI-assisted documentation process, summaries and technical explanations were created to simplify complicated ideas. In conclusion, integrating AI techniques in the

Notefy project yielded significant benefits.
AI significantly streamlined the development process by comprehending project requirements, producing code, addressing issues, and providing documentation. The program was well-structured, secure, and feature-rich, meeting all criteria. Using AI improved project productivity, accuracy, and creativity in the end product.