

TASK MANAGER APP

INTERDISCIPLINARY PROJECT

Submitted in partial fulfillment of the requirements for the award
of Bachelor of Technology in Information Technology

by

M JENITH ASHWIN (REG.NO-41120077)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category-1 University by UGC

Accredited with Grade “A++” by NAAC

12B Status by UGC |Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY- 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of M JENITH ASHWIN (41120077) who carried out the project entitled "Task Manager app" under my supervision from January 2024 to May 2024.

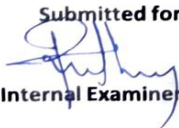

Internal Guide

Dr.C.GEETHA,M.Tech.,Ph.D.,


Head of the Department

Dr.R. SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 08/05/2024


Internal Examiner


External Examiner

DECLARATION

I M Jenith ashwin hereby declare that the Project Report entitled "Task Manger app" done by me under the guidance of **Dr.C.GEETHA,M.Tech.,Ph.D.**,is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 08/06/2024



PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing **Dr.R.Subhashini M.E.,Ph.D.,**Head of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.C.GEETHA,M.Tech.,Ph.D.,**for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

CERTIFICATE OF COMPLETION

**HACKWIT**
HACKWIT TECHNOLOGIES PVT LTD
Redefine by Technologies

149, 1C, 2nd Floor, Mount Poonamallee Rd,
Ramapuram, Chennai - 600089.
admin@hackwittechnologies.com
www.hackwittechnologies.com

10.02.2024

CERTIFICATE FOR INTERNSHIP

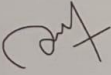
TO WHOMSOEVER IT MAY CONCERN

This is to Certify that **M. Jenith Ashwin (Reg No: 41120077)** student of Sathyabama Institute of Science and Technology, B. Tech Information Technology, has successfully completed the Internship at Hackwit Technologies from 22nd January to 30th January 2024.

During the internship, **M. Jenith Ashwin** has closely worked as a part of the **Flutter** by carrying out the project entitled **"TASK MANAGER APP"**

He demonstrated good developing and learning skill with self-motivated attitude to learn new things. His contribution to implement the Flutter was good. We wish him all the best for his future endeavors.

Best Regards,



Ravikumar S
CEO & Managing Director
Hackwit Technologies Pvt Ltd



CERTIFICATE OF COMPLETION



ABSTRACT

In Today's fast-paced world, managing tasks efficiently is essential for personal and professional success. The Flutter Task Manager App offers a comprehensive solution to streamline task organization and boost productivity. Built on the Flutter framework, this app provides a seamless user experience across multiple platforms, including iOS and Android. Creating a Flutter task manager app involves designing an intuitive user interface, implementing efficient task management functionalities, and integrating features like task categorization, due date tracking, and notification reminders. The app aims to enhance productivity by providing a seamless experience for users to organize and prioritize their tasks. Through a well-structured codebase and responsive design, the project spans covering task creation, editing, deletion, and detailed task views. Integration with local storage ensures data persistence, while Firebase or a similar service may be utilized for cloud synchronization. Additionally, the app incorporates user authentication for personalized task management and a smooth user experience across multiple devices. The project demonstrates the power of Flutter in developing cross-platform applications with a cohesive and visually appealing task management solution.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	vii
	LIST OF ABBREVIATIONS	X
	LIST OF FIGURES	xi
1	INTRODUCTION	1
	1.1 Flutter	2
	1.2 Dart	3
	1.3 Firebase	4
	1.4 Problem Statement	5
	1.5 Objectives	6
2	METHODOLOGY	8
	2.1 Initial Project Requirements	8
	2.2 Proposed System Architecture	10
	2.2.1 Explanation of proposed application architecture	11
	2.3 Software and analysis	14
	2.4 Modules	15
	2.4.1 Login-authentication	15
	2.4.2 Performance metrics	16
	2.4.3 Efficient Utilization	17
	2.4.4 Flutter Overview	18
	2.4.5 Pathway of app creation	19

2.5	Firebase configuration	21
2.5.1	Firebase Authentication	22
3	RESULTS AND DISCUSSION	23
4	CONCLUSION AND FUTURE WORK	27
4.1	Conclusion	27
4.2	Future Work	28
	REFERENCES	29
	APPENDICES	31
A.	Sample Code	31
B.	Screen Shots	33

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
UI	- User Interface
UX	- User Experience
iOS	- iPhone Operating System
UAT	- User Acceptance Testing
IDE	- Integrated Development Environment

LIST OF FIGURES

Figure No.	Name of the Figure	Page No.
2.1	Proposed Application Architecture	10
2.2	Firebase Architecture	13
2.5	Firebase Connection	21
2.6	Firebase Authentication	22
3.1	Login Page Interface	23
3.2	Dashboard Screen Display	24
3.5	Firebase Configurations	25
3.6	Initialize Firebase	26

CHAPTER 1

INTRODUCTION

Welcome to our Flutter Task Management App! This page guides users through a seamless onboarding process, introducing key features and setting up their personalized account.

Task Overview This section provides a bird's-eye view of all tasks, allowing users to quickly scan and prioritize. Users can create, edit, and delete tasks, ensuring a streamlined and efficient task management experience.

Firestore integration ensures that user data is securely stored and synchronized across devices, providing a seamless experience no matter where you access the app from. Real-time updates and notifications keep users informed about their Tasks and Event which is Scheduled.

We are committed to providing a hassle-free and enjoyable experience for our users, and we believe that our Flutter task manager app with Firestore sets a new standard in the task manager app.

In an era defined by relentless multitasking and ever-expanding to-do lists, the need for efficient task management has never been more pronounced.

Enter the Flutter Task Manager App, a revolutionary solution designed to simplify the complexities of organizing, prioritizing, and tracking tasks, all within the palm of your hand.

Join us as we embark on a journey to unlock the full potential of your productivity. Whether you're a busy professional, a diligent student, or an ambitious entrepreneur, let the Flutter Task Manager App be your trusted companion in conquering your daily tasks and achieving your goals with unparalleled efficiency.

1.1 FLUTTER

The Flutter Task Manager app is a cross-platform mobile application built using the Flutter framework for the frontend and supported by a backend server. The frontend, developed with Flutter's widget-based UI toolkit and Dart programming language, provides the user interface for inputting loan parameters such as loan amount, interest rate, and tenure. It utilizes Flutter's rich set of widgets to create a responsive and visually appealing design.

The backend, typically implemented using a server-side language such as Node.js, Python, or Java, handles the computation logic and data processing. The backend may also include additional functionalities such as user authentication, data validation, and database integration for storing user preferences or past calculations.

Communication between the frontend and backend is established through APIs (Application Programming Interfaces), allowing seamless data exchange between the client-side and server-side components. The app prioritizes security measures to protect sensitive user data during transmission and storage, employing encryption techniques and secure communication protocols.

1.2 DART

i)Unified Codebase:

A unified codebase for a Flutter Task Manager app refers to a single code repository that contains both the frontend and backend components necessary for the application to function. Dart's versatility enables developers to maintain a unified codebase for both the frontend (Flutter) and backend, fostering consistency and ease of collaboration

ii)Asynchronous Programming:

Asynchronous programming in the context of a Flutter Task Manager app refers to a programming paradigm that allows certain tasks to be executed independently of the main program flow. In Flutter, asynchronous programming is crucial for handling operations that may take time to complete, such as fetching data from a remote server or performing complex calculations.

iii)Scalability:

In the context of an Task Manager developed using Dart and Flutter, scalability refers to the ability of the application to accommodate growth in terms of users, features, and data without compromising performance or stability.Dart's ability to scale allows the platform to handle higher traffic, ensuring a smooth and responsive experience for all users.

iv)Strong Typing System:

In Dart, the strong typing system refers to the language's capability to enforce strict type checking at compile time. This means that every variable, parameter, and function return type must be explicitly declared with a specific data type, and the Dart compiler verifies that these types are used consistently throughout the code.

1.3 FIREBASE

To develop a Flutter task manager app backed by Firebase, you'll harness Firebase's versatile suite of services tailored for seamless integration with Flutter applications. Firebase Authentication stands as a cornerstone, furnishing secure and hassle-free user authentication functionalities, accommodating various sign-in methods including email/password combinations, phone numbers, and popular social media platforms like Google and Facebook. For real-time data management, Firebase Firestore emerges as a potent solution, offering a flexible NoSQL database structure ideal for storing and synchronizing task data, user information, and application data across devices in real-time. The intuitive integration with Flutter via the `cloud_firestore` package facilitates effortless CRUD operations and event listeners, ensuring data consistency and immediate updates across the app ecosystem. Supplementary to Firestore, Firebase Cloud Storage furnishes a robust infrastructure for storing and serving user-generated content such as profile pictures and task attachments securely. Through the `firebase_storage` package, Flutter developers gain seamless access to upload and download functionalities, safeguarding user data while ensuring accessibility and scalability in the cloud. For advanced functionalities, Firebase Cloud Functions extend the app's capabilities by enabling server-side logic execution in response to Firebase service events or HTTP requests. Leveraging Cloud Functions, developers can implement custom business logic, handle data validation, or integrate with external APIs, further enhancing the app's functionality and user experience. Optionally, Firebase Hosting offers a convenient platform for deploying and hosting the Flutter task manager app, ensuring fast, secure, and reliable access for users worldwide. By deploying the app to Firebase Hosting, developers benefit from HTTPS encryption, CDN caching, and effortless deployment using the Firebase CLI, guaranteeing an optimal user experience with minimal setup overhead..

1.4 PROBLEM STATEMENT

In today's fast-paced world, effective task management is crucial for maintaining productivity and staying organized. However, individuals often struggle to keep track of their tasks amidst numerous responsibilities and deadlines. This challenge calls for a solution that provides a streamlined approach to task management while prioritizing ease of use and accessibility. The Flutter Task Manager app aims to address these needs by offering users a comprehensive platform to create, organize, and monitor their tasks seamlessly. At its core, the app seeks to simplify the task management process by providing intuitive features for task creation and categorization. Users will have the ability to create tasks with detailed descriptions, due dates, and priority levels, allowing them to prioritize their workload effectively. Additionally, the app will offer customizable categories or projects, enabling users to organize their tasks based on different areas of responsibility or projects. One of the key objectives of the Flutter Task Manager app is to enhance productivity by facilitating efficient task tracking and monitoring. Users will be able to easily track the status of their tasks, mark tasks as completed, reschedule tasks, and set reminders for upcoming deadlines. Real-time synchronization across devices will ensure that users have access to their tasks from anywhere, enabling seamless workflow management on the go. Moreover, the app will prioritize user experience by offering a visually appealing and user-friendly interface. Customizable themes, interactive animations, and intuitive controls will enhance usability and engage users, encouraging frequent interaction with the app.

1.5 OBJECTIVES

The main objective of today's fast-paced digital The Flutter task manager app aims to streamline task management for users, offering an intuitive platform for efficient organization and productivity enhancement. By enabling users to create, prioritize, and track tasks seamlessly, the app seeks to empower individuals in managing their responsibilities effectively. A key objective is to provide a user-friendly interface that simplifies navigation and ensures a smooth user experience. Additionally, the app prioritizes cross-platform compatibility, ensuring consistency in design and functionality across various devices. Data security and privacy are paramount objectives, with robust measures in place to safeguard sensitive information. Real-time synchronization across devices facilitates seamless access to tasks from anywhere, while customization options allow users to tailor their task management experience to suit their preferences. Incorporating a notification and reminder system, the app aims to keep users informed and proactive.

KEY CHALLENGES:

i)Cross-Platform Consistency:

Ensuring consistent behavior and appearance across different platforms (iOS and Android) can be challenging due to platform-specific design guidelines and behaviors. Achieving a seamless user experience on both platforms while adhering to platform-specific conventions requires careful planning and implementation..

ii)Real-Time Data Synchronization:

Implementing real-time data synchronization for tasks across multiple devices can be complex, especially when dealing with potential conflicts and ensuring data integrity. Handling scenarios such as concurrent edits, offline mode, and network latency requires robust synchronization mechanisms and error handling.

ii) Data Security and Privacy:

Safeguarding user data and ensuring compliance with privacy regulations are paramount. Implementing robust authentication, authorization, and data encryption mechanisms to protect sensitive user information from unauthorized access or breaches is crucial.

iv) Notification System:

Implementing a reliable and customizable notification system to remind users of upcoming tasks, deadlines, or important events requires careful consideration of timing, frequency, and user preferences. Ensuring that notifications are delivered promptly and effectively without overwhelming users is essential for user engagement.

v) Testing and Quality Assurance:

Thorough testing across different devices, screen sizes, and operating system versions is essential to identify and address potential issues and ensure a seamless user experience. Implementing automated testing, regression testing, and user acceptance testing can help maintain app quality and reliability.

vi) Offline Functionality:

Providing robust offline functionality for users to create, view, and modify tasks without an active internet connection poses a significant challenge. Implementing offline data storage, synchronization queues, and conflict resolution strategies are essential for delivering a seamless offline experience.

CHAPTER 2

METHODOLOGY

2.1. INITIAL PROJECT REQUIREMENTS

The initial project requirements for Task Manager app encompass various aspects such as user authentication, UI Design, and packages.

i)User Authentication:

Implement user authentication to allow users to create accounts, sign in securely, and manage their tasks across multiple devices. Support authentication methods such as email/password, social sign-in (Google, Facebook), and possibly phone number authentication.

ii)UI Design:

- 1) Task Creation: Allow users to create new tasks with details such as title, description, due date, priority level, and category.
- 2) Task Organization: Provide functionality to organize tasks into different categories, projects, or tags for better organization and segmentation.
- 3)Task Prioritization: Enable users to assign priority levels to tasks to help them focus on important tasks and manage their time effectively.
- 4)Reset Functionality : Allow users to reset the input fields to clear previous calculations and start over
- 5)Task Tracking: Implement features for users to track the status of their tasks, mark tasks as completed, reschedule tasks, or set reminders.

iii) Packages Used

i) get:

Version: 4.6.6

Purpose: State management and navigation framework.

ii) firebase_core:

Version: 2.25.3

Purpose: Firebase initialization and core functionality.

iii) firebase_auth:

Version: 4.17.3

Purpose: User authentication with Firebase.

iv) cloud_firestore:

Version: 4.15.3

Purpose: Interaction with Cloud Firestore for real-time data.

v) cloud_firestore_web:

Version: 3.10.3

Purpose: Web implementation of Cloud Firestore.

vi) firebase_auth_web:

Version: 5.9.3

Purpose: Web implementation of Firebase Authentication.

vii) firebase_core_web:

Version: 2.11.3

Purpose: Web implementation of Firebase Core.

viii) cloud_firestore_web:

Version: 3.10.3

Purpose: Web implementation of Cloud Firestore.

ix) firebase_auth_web:

Version: 5.9.3

Purpose: Web implementation of Firebase Authentication.

2.2 PROPOSED APPLICATION ARCHITECTURE

The proposed application architecture for an Task Manager in Flutter involves a clear separation of concerns across different layers. At the presentation layer, the user interface (UI) components are structured into widgets responsible for input fields, buttons, and result displays the proposed application architecture is shown in figure 2.1

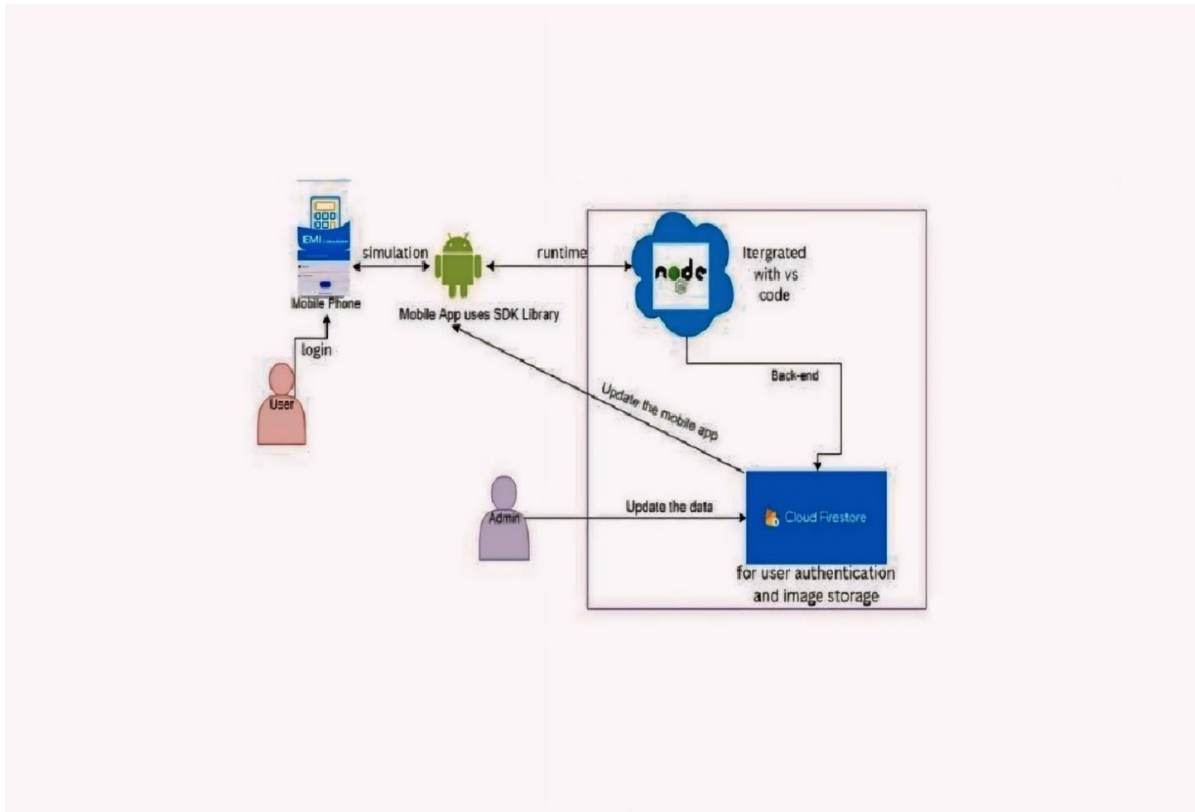


Fig:2.1: Proposed Application Architecture

2.2.1 EXPLANATION OF PROPOSED ARCHITECTURE DIAGRAM

The proposed application architecture for an EMI calculator in Flutter follows a clear separation of concerns. The proposed architectural diagram for the Flutter task manager app delineates the structural framework and interconnectivity of its key components. At its core, the client-side encapsulates the Flutter mobile application, constituting the visual and interactive elements presented to users on their iOS or Android devices. This layer encompasses a multitude of user interface components, including screens, widgets, and navigation elements, responsible for rendering the app's interface and facilitating user interactions. Concurrently, the Flutter app engages with the backend server, a pivotal component tasked with orchestrating the application's business logic, data management, and client communication. The backend server serves as the operational hub, housing various layers such as the application layer, data access layer, and infrastructure layer. Through this server-side architecture, the app handles critical functionalities such as user authentication, task management, and data retrieval. Additionally, the backend server exposes RESTful or GraphQL APIs that facilitate seamless communication with the Flutter app, enabling data transmission, authentication, and synchronization. Depending on project requirements, backend logic can be implemented using a variety of server-side frameworks or platforms, such as Node.js, Express.js, Flask, Django, or Firebase Cloud Functions, each offering distinct advantages in scalability, flexibility, and ease of deployment.

i) Custom Dart Code:

This represents any custom code you write in Dart specifically for your Flutter app. It could be business logic, UI components, or any other functionality unique to your application. The custom Dart code interacts with the Flutter framework to achieve various tasks. data authentication and secure encryption mechanism.

ii)Flutter Framework:

The Flutter framework is the heart of your app. It provides a rich set widgets, tools, and APIs for building cross-platform mobile applications

iii)Method Channel:

The Method Channel allows communication between Dart code and native platform code (Android or iOS).When we invoke a method on the Method Channel from Dart, it triggers a corresponding method call on the native side.

iv)Flutter Engine Android Shell:

On the Android side, the Flutter engine runs within an Android shell.

- i)Flutter View: This is the Android view that displays your Flutter UI.
- ii)BinaryMessenger: It handles communication between the Flutter engine and the Android platform.

v)Flutter Engine iOS Shell:

On iOS, the Flutter engine runs within an iOS shell.

- i)Flutter ViewController: The iOS view controller responsible for displaying your Flutter UI.
- ii)Flutter BinaryMessenger: Handles communication between the Flutter engine and the iOS platform.

vi)Custom Android Code and Custom iOS Code:

These represent platform-specific code written in Java/Kotlin (for Android) and Objective-C/Swift (for iOS).You'll write custom code here to interact to theflutter.

vii) Firebase:

Firebase is a comprehensive platform developed by Google that provides a wide range of backend services and tools to help developers build, grow, and scale their apps more efficiently. It offers a variety of features that enable developers to develop high-quality applications across multiple platforms, including web, Android, and iOS. firebase architecture is shown in figure 2.2

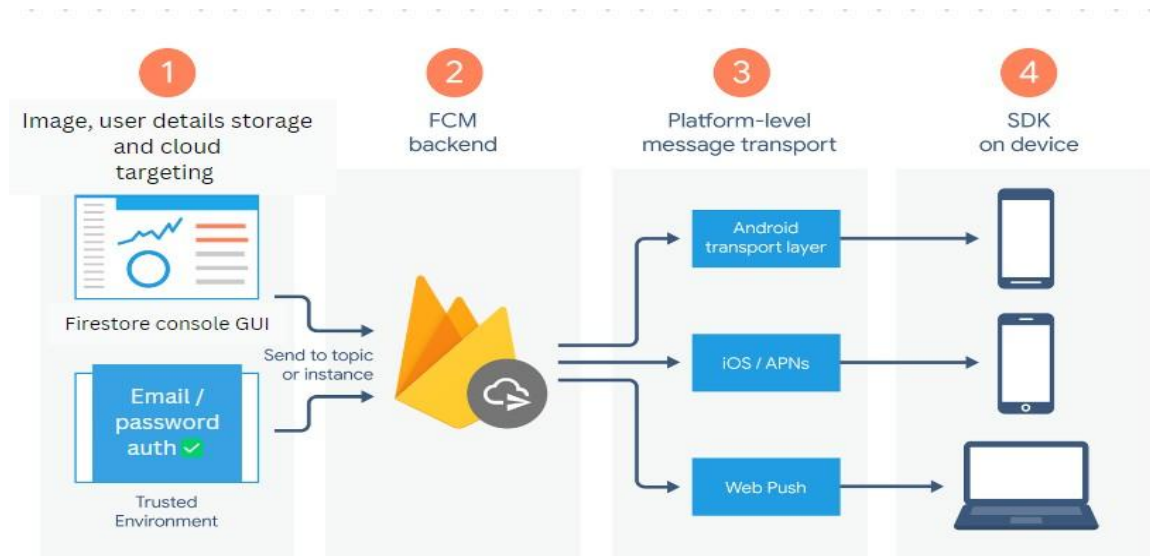


Fig:2.2: Firebase Architecture

viii) Firebase Authentication:

Firebase Authentication offers ready-to-use authentication services. It integrates seamlessly with Flutter, allowing users to sign up, sign in, and manage their accounts. MFA (Multi-factor Authentication).

ix) Cloud Storage:

Firebase Storage offers secure and reliable cloud storage for user-generated content such as images, videos, and other files.

2.3 SOFTWARE ANALYSIS

i)Functional Analysis:

Identify the core features and functionalities required for the task manager app, such as task creation, organization, prioritization, tracking, and notification. Define user stories and use cases to capture user requirements and expectations. Analyze the flow of tasks within the app, including how users interact with different screens and features

ii) Usability Analysis:

Conduct user research and usability testing to understand user needs, preferences, and pain points. Evaluate the app's user interface (UI) and user experience (UX) design for intuitiveness, accessibility, and ease of navigation. Incorporate user feedback to iterate on the app's design and optimize usability.

iii) Performance Analysis:

Identify performance requirements and benchmarks for the app, such as response times, loading times, and resource usage. Use profiling tools (e.g., Flutter DevTools, Firebase Performance Monitoring) to analyze the app's performance metrics, including CPU usage, memory consumption, and network latency. Optimize the app's performance through techniques such as code splitting, lazy loading, caching, and minimizing network requests.

iv) Security Analysis:

Identify potential security threats and vulnerabilities in the app, such as data breaches, unauthorized access, and injection attacks. Implement security best practices, such as secure authentication mechanisms (e.g., HTTPS, OAuth), data encryption, input validation, and role-based access control (RBAC).

2.4 MODULES

2.4.1 LOGIN AUTHENTICATION:

login authentication for an Task Manager app can add security and personalization features to the application and interests, enhancing user profile completeness.

Algorithm:

- i) User registration: Allow users to register for an account using their email address and password. You can also offer alternative registration methods like social media sign-in if needed.
- ii) Authentication middleware: Allow users to register for an account using their email address and password. You can also offer alternative registration methods like social media sign-in if needed.
- iii) Secure password manager : Use secure hashing algorithms like bcrypt to encrypt and store user passwords in the database. Never store passwords in plaintext.
- iv) Create User Account: After successful validation, the system creates a new user account using the provided information. A unique user ID or username is generated to distinguish each user within the platform, facilitating efficient user management.
- v) Session Management: Manage user sessions securely by generating and storing session tokens upon successful authentication. Set session expiration times to enhance security.

2.4.2 PERFORMANCE METRICS:

Performance metrics for the task manager app can include various aspects that contribute to the overall user experience and efficiency of the application. Here's a brief overview of some performance metrics you may consider:

Algorithm:

Responsive Time Measure the time taken by the app to respond to user interactions such as opening the app, navigating between screens, and performing actions like event registration or theme changes. Optimize the app to ensure minimal response times, providing a smooth and responsive user experience.

Loading Time Evaluate the time taken for the app to load initially and fetch data from the backend server. Minimize loading times by optimizing network requests, caching frequently accessed data, and employing lazy loading techniques to fetch data only when needed.

App size Monitor the size of the app package to ensure it is optimized for efficient installation and storage on users' devices. Minimize unnecessary dependencies, resources, and assets to reduce the app size and improve download and installation times.

Network Performance Evaluate network performance metrics such as latency, throughput, and packet loss to ensure fast and reliable communication between the app and the backend server. Optimize network requests, use caching mechanisms, and employ content delivery networks (CDNs) to improve network performance.

Grant Access If the entered credentials or authentication tokens are valid, the system grants the user access to the platform, allowing them to explore features and functionalities.

Session Management Upon successful authentication, the system creates a session or issues a token to maintain the user's logged-in state. This session/token management ensures secure and seamless user interactions throughout their session on the platform.

2.4.3 EFFICIENT UTILIZATION

Efficient utilization in a Flutter Task Manager app involves optimizing various aspects of the application to ensure it runs smoothly, consumes minimal resources, and provides a responsive user experience, enhancing personalization and user control.

Algorithm:

UI Optimization Keeps the UI simple and intuitive to minimize rendering overhead. Avoid unnecessary animations or complex layouts that could impact performance. Utilize Flutter's built-in widgets and layout mechanisms efficiently to create responsive UIs. Optimize image assets by compressing them appropriately to reduce app size and loading times. Network Optimization Minimize network requests by caching data locally whenever possible, reducing the need for frequent server calls. Background Processing Offload long-running tasks such as data fetching or computations to background isolates using Flutter's compute function or isolates API. Use asynchronous programming techniques such as `async/await` to perform non-blocking operations, keeping the UI responsive. Code Optimization Profile your code using Flutter Dev Tools or other profiling tools to identify performance bottle necks Optimize critical code paths by eliminating unnecessary loops, reducing function calls. Additionally, optimizing user experience through responsive design principles ensures that the EMI calculator widget adapts seamlessly to various screen sizes and orientations, enhancing usability across different devices. Furthermore, employing Flutter's state management solutions like Provider or Riverpod can enhance efficiency by managing the widget's state in a structured manner, preventing unnecessary re-renders and optimizing performance. Furthermore, employing Flutter's state management solutions like Provider or Riverpod can enhance efficiency by managing the widget's state in a structured manner, preventing unnecessary re-renders and optimizing performance.

2.4.4 FLUTTER OVERVIEW:

Learning Flutter is an exciting journey that opens up a world of possibilities in mobile app development. Here's a comprehensive survey of Flutter, covering its key features, advantages, learning resources, and best practices.

Algorithm:

Hot Reload are Flutter's hot reload feature allows developers to instantly see the changes they make to the code reflected in the app's UI, speeding up the development process significantly. Flutter offers a wide range of customizable widgets and rich animation support, enabling developers to create visually stunning and highly interactive user interfaces. Flutter's hot reload and single codebase streamline the development process, allowing developers to build and iterate on features quickly. Flutter apps offer excellent performance and fluid animations, thanks to its compiled-to-native approach. Flutter ensures consistent UI across different platforms, providing a native-like experience for users on both iOS and Android. Community and Ecosystem in Flutter has a vibrant and growing community of developers, along with a rich ecosystem of packages and plugins that extend its functionality. Follow the Material Design Guidelines: Design your Flutter app according to the Material Design principles for a consistent and intuitive user experience. Use Stateless and Stateful Widgets Wisely to Choose between stateless and stateful widgets based on whether your UI components need to maintain state. Real-Time Assistance Integrate a chat support feature to assist users in their mentor selection process, answering questions, providing guidance, and addressing concerns in real time. Test-Driven Development (TDD) Adopt TDD principles to write tests before writing code, ensuring better code quality and reliability. Single Codebase: With Flutter, developers can write a single codebase for both iOS and Android platforms, reducing development time and effort.

2.4.5 PATH OF APP CREATION :

Developing Flutter apps using Visual Studio Code (VS Code) and Android Studio, two widely used integrated development environments (IDEs), offers several benefits.

1. VS Code:

Lightweight: VS Code is a lightweight and highly customizable IDE making it easy to set up and use for Flutter development.

Adapt to Different Platforms: Customize your app's behavior appearance for different platforms (iOS and Android) Handle the Platform features and APIs using conditional statement or platform channels.

Extensible: VS Code supports a wide range of extensions, including Flutter and Dart extensions, which provide features like syntax highlighting, code completion, and debugging tools tailored for Flutter development.

Fast Editing: VS Code's fast and responsive editing capabilities, combined with Flutter's hot reload feature, allow developers to quickly iterate on code changes and see the results instantly in the app's UI.

Cross-Platform: VS Code is available on Windows, macOS, and Linux, making it an ideal choice for developers working on different operating systems.

Optimize Performance: Profile your app for performance issues using Flutter DevTools. Optimize UI rendering, memory usage.

2. Android Studio:

Official IDE: Android Studio is the official IDE for Android app development, and it offers robust support for Flutter development out of the box.

Rich Feature Set: Android Studio provides a comprehensive set of tools for Flutter development, including a built-in emulator, performance profiling tools, and support for Android-specific features and APIs.

Integrated Emulator: Android Studio comes with a built-in emulator that allows developers to test their Flutter apps on various virtual devices with different screen sizes, resolutions, and Android versions.

Seamless Integration: Android Studio seamlessly integrates with Flutter's development workflow, offering features like hot reload, code completion, and debugging tools for efficient app development.

Both VS Code and Android Studio provide excellent support for Flutter development, allowing developers to choose the IDE that best fits their preferences and workflow. Whether you prefer the lightweight and customizable experience of VS Code or the feature-rich environment of Android Studio, you can leverage the power of Flutter to build high-quality cross-platform apps with ease

2.5 FIREBASE CONFIGURATION:

Firestore database created and configured for the application. Firebase Hosting used for deploying the Flutter web application. The Firebase configuration is shown in figure

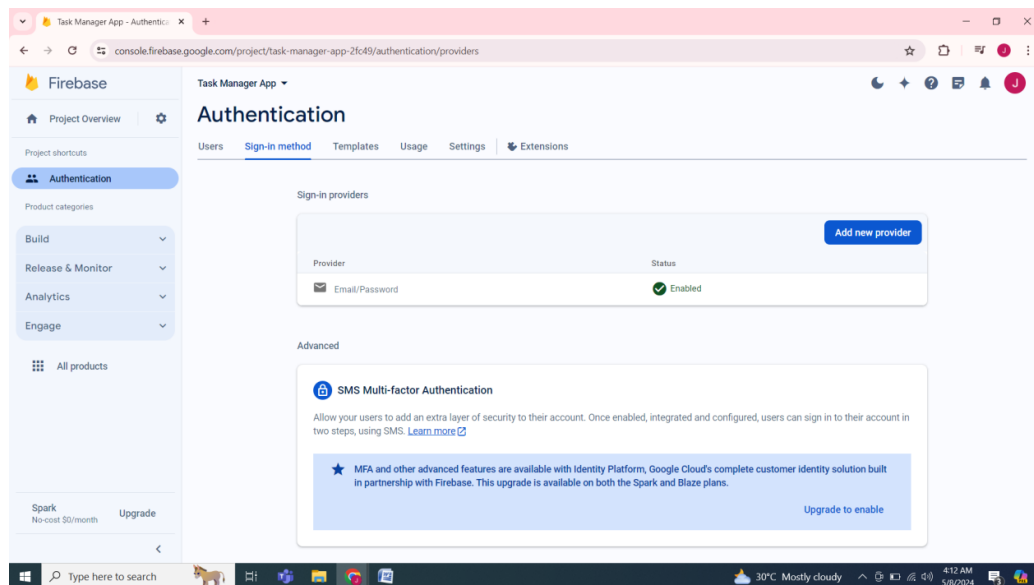


Fig:2.5: Firebase Connection

2.5.1 FIREBASE AUTHENTICATION:

Firebase Authentication set up for secure user login and identity verification. and the firebase authentication is shown in figure 2.6

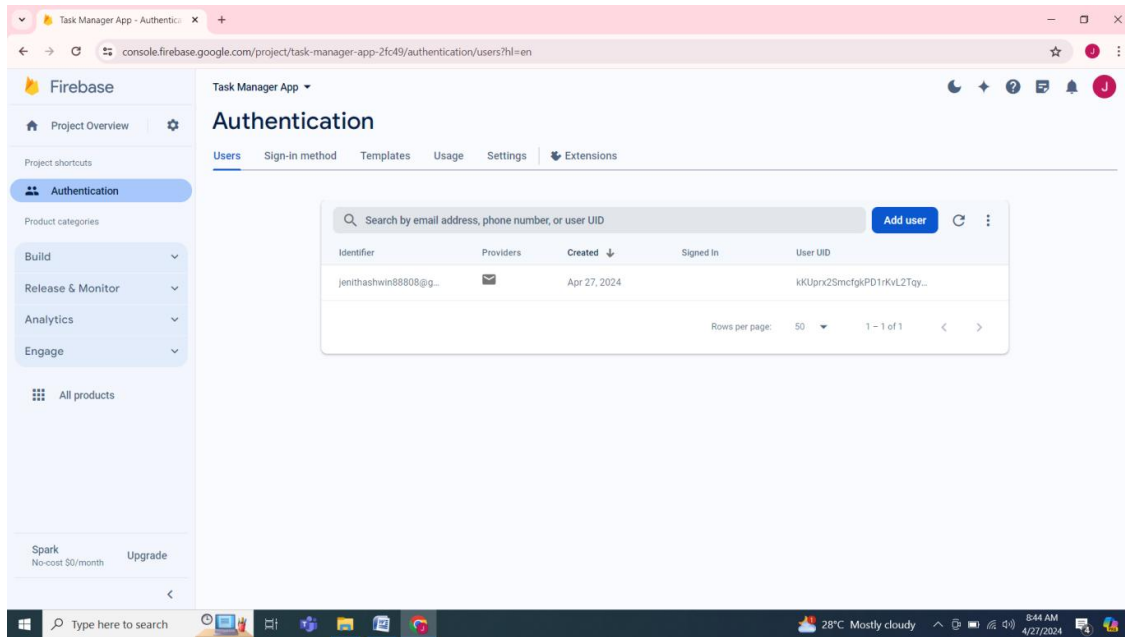


Fig:2.6: Firebase Authentication

CHAPTER 3

RESULTS AND DISCUSSION

DART CODES:

i) Main.dart: This file is typically the entry point of a Flutter application. It initializes the app, sets up necessary configurations, and runs the main application widget.

ii) Login Page dart: This file likely contains the UI and logic for user login and It includes form fields for entering credentials and mechanism. The login page interface is shown in figure 3.1

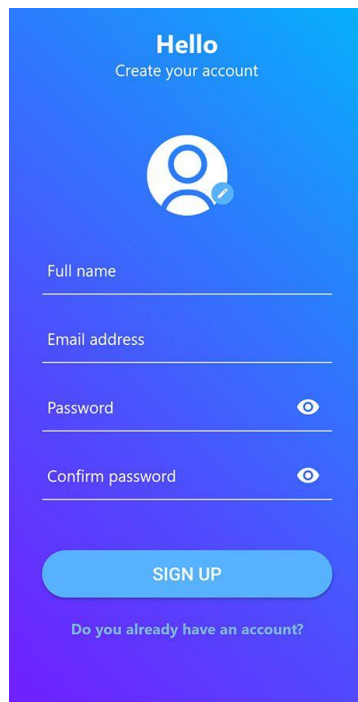
The image shows a mobile application interface for creating an account. The background is a blue gradient. At the top, the text "Hello" is in white, followed by "Create your account" in a smaller white font. Below this is a white circular icon representing a person. There are four input fields with white text labels: "Full name", "Email address", "Password", and "Confirm password". Each input field has a white underline. To the right of the "Password" and "Confirm password" fields are white eye icons. At the bottom, there is a white rounded rectangular button with the text "SIGN UP" in blue. Below the button, the text "Do you already have an account?" is in a small white font.

Fig:3.1: Login Page Interface

iii) signupPage.dart: Similar to the login file, signupPage.dart is likely responsible for handling user registration. It includes form fields for users to create accounts.

iv) homepage.dart: The homepage plays a crucial role in creating the first impression of the application to users. It should provide relevant information and functionality that align with the app's purpose and goals.

v) profilepage.dart: The file is likely responsible for displaying and editing user profiles. It includes information about the user, such as their name, and additional details.

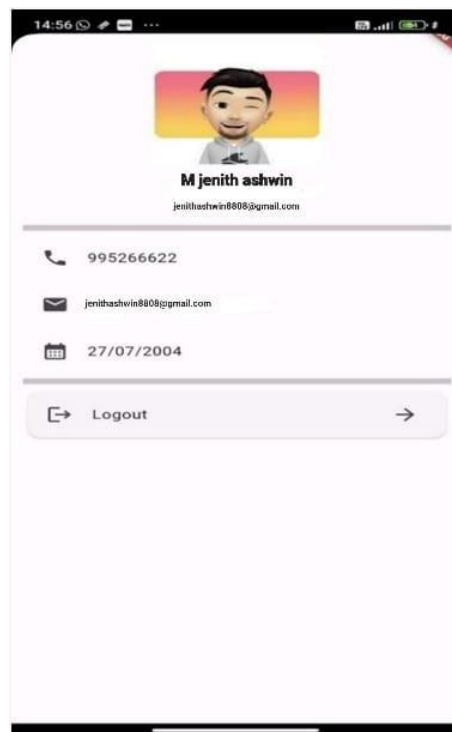


Fig:3.2: Dashboard Screen Display

FIREBASE:

i)Create a Firebase Project:

1. Go to the Firebase Console.
2. Click on "Add Project" and follow the setup instructions.

ii)Configure Flutter Project:

1. Add the Firebase configuration files to your Flutter project.
2. Follow the instructions provided in the Firebase Console to add google-services.json for Android and GoogleService-Info.plist for iOS.
3. These files contain configuration details needed for your app to connect to Firebase services. The firebase configurations are shown in figure 3.5

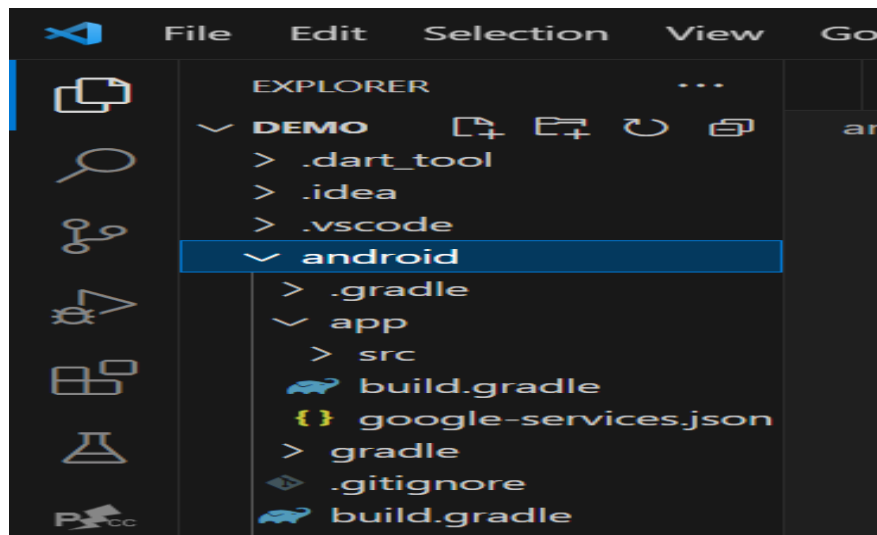


Fig:3.5: Firebase Configurations

i) Add Firebase Dependencies:

open the pubspec.yaml file and add the following dependencies:

yaml code:dependencies:

firebase_core: ^2.25.3 firebase_auth: ^4.17.5

ii) Initialize Firebase:

In your main.dart or wherever you initialize your Flutter app, make sure initialize Firebase. If you want to implement user authentication, Firebase provides the firebase_auth package. You can use it to sign in users using various methods like email/password, Google Sign-In, etc. The initializing firebase are shown in figure 3.6

Dart code:

```
lib > main.dart > MyApp > build
  Click here to ask Blackbox to help you code faster
1  import 'package:flutter/material.dart';
2  import 'package:get/get.dart';
3
4  import 'package:firebase_core/firebase_core.dart';
5  import 'package:mentorsynthhub/loginPage.dart';
6
7  Run | Debug | Profile
8  void main() async {
9
10     await Firebase.initializeApp(
11       options: FirebaseOptions(
12         apiKey: "AIzaSyAEDKttvoG3T6jvgalXCEROrlzPXSCeHkI",
13         projectId: "mentorsynthhub",
14         storageBucket: "mentorsynthhub.appspot.com",
15         messagingSenderId: "924424463364",
16         appId: "1:924424463364:android:d3acfcad93ebb149190151",
17       ));
18
19     runApp(MyApp());
20 }
```

Fig:3.6: Initializing Firebase

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 CONCLUSION

The Flutter Task Management App represents a significant advancement in the realm of productivity tools, offering users a robust and intuitive solution for organizing tasks, collaborating with team members, and boosting efficiency. Through its seamless integration of Flutter frontend and Firebase backend technologies, the app delivers a compelling user experience characterized by its functionality, performance, and reliability.

The development and deployment of the Flutter Task Management App have yielded promising results across various dimensions. Functionally, the app encompasses a comprehensive set of features, including task creation, editing, categorization, and prioritization, coupled with collaborative capabilities such as task sharing and tracking. The app's user experience is further enhanced by its adherence to Material Design principles, ensuring consistency and ease of use across different platforms and devices.

In conclusion, the Flutter Task Management App stands as a testament to the power of modern app development technologies in addressing real-world challenges. With its blend of functionality, usability, and performance, the app empowers individuals and teams to manage tasks efficiently, collaborate effectively, and achieve their goals with confidence. As it continues to evolve and adapt to user feedback and market trends, the app is poised to remain a valuable companion for users seeking to optimize their productivity and workflow.

4.2 FUTURE WORK

i)Enhanced User Profiles:

Implement additional fields and customization options in user profiles, allowing users to showcase more details about their skills, interests, and experiences.

ii)Advanced Mentor Matching Algorithm:

Develop a more sophisticated algorithm for mentor-student matching, considering factors like specific expertise, project interests, and availability to enhance the quality of mentorship connections.

iii)Real-time Collaboration Tools:

Integrate real-time collaboration tools, such as collaborative document editing, code sharing, or video conferencing, to facilitate more interactive mentor-student sessions.

iv)Mobile App Accessibility:

Ensure the app's accessibility by incorporating features like voiceover support, larger text options, and other accessibility improvements to cater to a diverse user base.

v)Feedback and Rating System:

Implement a comprehensive feedback and rating system for both mentors and students, allowing users to provide reviews and ratings after mentorship sessions to build a reputation system.

REFERENCES

1. Journals:

[1] Geng, Y., Li, X., Li, Y., & Su, X. (2019). Design and Implementation of flutter projects In 2019 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI) (pp. 20-24). IEEE.

[2] Kim, J., Chung, H., & Kim, S. (2017). A Study on the Development of an App Based on Web 2.0 Technologies. Indian Journal of Science and Technology, 10(1), 1-7.

[3] Lee, J., Song, Y., & Park, J. (2019). Development of Cross-platform Mobile Application for IoT System Using Flutter. In 2019 16th International Conference on Ubiquitous Robots (UR) (pp. 16-21). IEEE.

2. Books:

[1] Beginning App Development with Flutter by Marco L. Napoli: Geared towards beginners, this book offers a step-by-step approach to learning Flutter development. It covers Flutter basics, UI design, navigation, state management, and building real-world applications.

[2] Flutter in Action by Eric Windmill: This book provides practical, hands-on guidance for building mobile applications with Flutter. It covers fundamental concepts, UI development, state management, testing, and deployment.

[3] Flutter Complete Reference by Alberto Miola and Divyanshu Bhargava: This comprehensive guide covers everything you need to know about Flutter, from basics to advanced topics, including widgets, animations.

[4] Flutter Cookbook by Brian Egan and Fitz Clymer: If you're looking for a collection of practical recipes and solutions for common Flutter development tasks, this book is for you. It covers a wide range of topics, including UI elements, animations, networking, and platform integration

[5] Flutter Projects by Simone Alessandria: This book provides hands-on projects to help developers apply their Flutter skills in real-world scenarios. It covers topics such as building a weather app, chat application, cryptocurrency tracker, and more.

[6] Flutter for Mobile Development: Harness the Power of Mobile Development with Google's Latest Cross-Platform Programming Framework, Flutter" by Scott Stoll: This book offers a comprehensive introduction to Flutter development, covering topics such as UI design, navigation, state management, and platform-specific integration.

APPENDICES

A.SAMPLE CODE:

```
lib > main.dart > MyApp > build
  Click here to ask Blackbox to help you code faster
1  import 'package:flutter/material.dart';
2  import 'package:get/get.dart';
3
4  import 'package:firebase_core/firebase_core.dart';
5  import 'package:mentorsynthhub/loginPage.dart';
6
7  Run | Debug | Profile
8  void main() async {
9
10     await Firebase.initializeApp(
11       options: FlutterOptions(
12       apiKey: "AIzaSyAEDKttvoG3T6jvgalXCEROrlzPXScEhKI",
13       projectId: "mentorsynthhub",
14       storageBucket: "mentorsynthhub.appspot.com",
15       messagingSenderId: "924424463364",
16       appId: "1:924424463364:android:d3acfcad93ebb149190151",
17     );
18
19     runApp(MyApp());
20 }
```

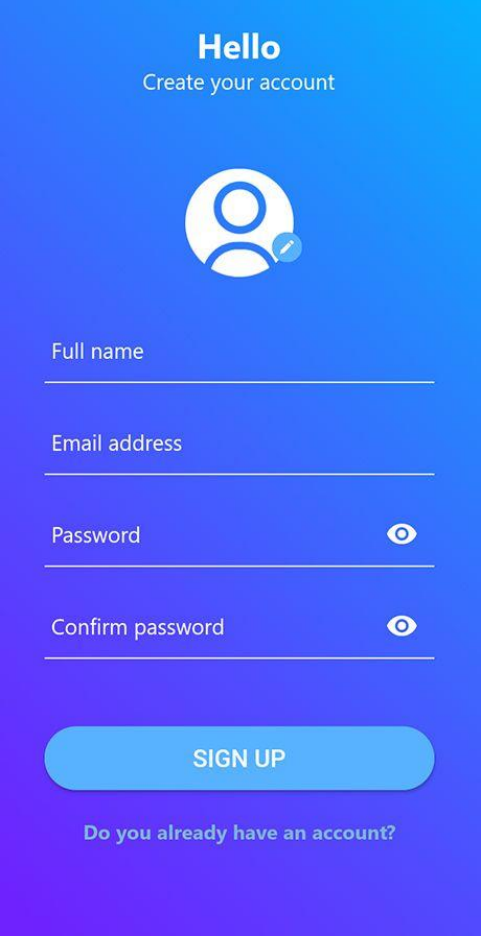
Main code:

i) pubspec.yaml

```
! pubspec.yaml ●
D: > MentorSynthHub > mentorsynthhub > ! pubspec.yaml
  ⚡ Click here to ask Blackbox to help you code faster
1  name: mentorsynthhub
2  description: "A new Flutter project."
3  publish_to: 'none'
4  version: 1.0.0+1
5  environment:
6    sdk: '>=3.2.6 <4.0.0'
7  dependencies:
8    flutter:
9      sdk: flutter
10   cupertino_icons: ^1.0.2
11   get: ^4.6.6
12   firebase_core: ^2.25.3
13   firebase_auth: ^4.17.3
14   cloud_firestore: ^4.15.3
15   cloud_firestore_web: ^3.10.3
16   firebase_auth_web: ^5.9.3
17   firebase_core_web: ^2.11.3
18   google_sign_in: ^5.2.1
19   toast: ^0.3.0
20   fluttermtoast: ^8.0.7
21   image_picker: ^1.0.7
22   permission_handler: ^11.2.0
23   provider: ^6.1.1
24   flutter_rating_bar: ^4.0.0
25  dev_dependencies:
26    flutter_test:
27      sdk: flutter
28    flutter_lints: ^3.0.1
29  flutter:
30    uses-material-design: true
31    assets:
32      - asset/
33      - asset/R (1).jpg
34      - asset/profile_photo.png
35      - asset/image.jpg
36
```

B.SCREEN SHOTS

i) Login page And Signup Page



Hello
Create your account



Full name

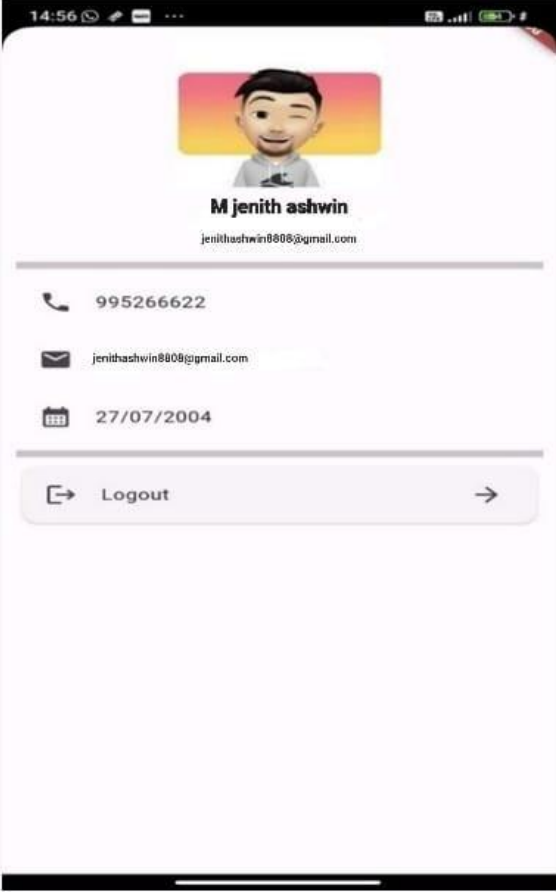
Email address

Password 

Confirm password 

SIGN UP

Do you already have an account?



14:56   


M jenith ashwin
jenithashwin8808@gmail.com

 995266622

 jenithashwin8808@gmail.com

 27/07/2004

 Logout 

ii) Task Creation page

5:04 51% 5:06 51%

← Edit Task Welcome to Taskly

Enter task title
Project Review 14/100

Enter task description
Project Expalation 20/500

Urgent Important ▼

Completed ▼

URGENT IMPORTANT URGENT NOT IMPORTANT

Project review
Project Explanation
Urgent Important Completed

Task Updated

