

TASK MANAGER USING FLUTTER

INTERDISCIPLINARY PROJECT

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

by

CHINTHALAPUDI SRI RAMA GOKULA (41120047)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY-1 UNIVERSITY BY UGC

Accredited "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 CHINTHALAPUDI SRI RAMA GOKULA(41120047) who carried out the project entitled "Task Manager" under my supervision from January 2024 to May 2024.

Internal Guide

Dr.L.MARY GLADENCE,M.E.,Ph.D.,

Head of the Department

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

Submitted for Viva voce Examination held on 08/05/24

Internal Examiner

815

External Examiner

DECLARATION

I, CHINTHALAPUDI SRI RAMA GOKULA hereby declare that the Project Report entitled "Task Manager" done by me under the guidance of **Dr. L. Mary Gladence, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 08/05/24

PLACE: Chennai

Ch. Sri Rama Gokula
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.L.MARY GLADENCE,M.E.,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



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 **Chinthalapudi Sri Rama Gokula (Reg No: 41120047)** 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, **Chinthalapudi Sri Rama Gokula** has closely worked as a part of the **FLUTTER** by carrying out the project entitled "**Task Manager**".

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

Best Regards,

Ravikumar S
CEO & Managing Director
Hackwit Technologies Pvt Ltd



CERTIFICATE OF COMPLETION



ABSTRACT

Efficient task management is vital for personal and professional productivity. This abstract explores the development of a task manager application using Flutter, a versatile and robust framework for building cross-platform apps. Leveraging Flutter's rich widget library and customizable UI components, the task manager offers a seamless user experience across various devices. Key features include task categorization, priority settings, deadline reminders, and progress tracking. Through intuitive design and smooth navigation, users can easily create, organize, and monitor their tasks, enhancing productivity and time management skills. The utilization of Flutter ensures not only a visually appealing interface but also facilitates rapid development and deployment across Android, iOS, and web platforms, catering to diverse user preferences and increasing accessibility. Crafting a comprehensive task manager utilizing the Flutter framework presents a dynamic approach to addressing modern productivity needs. This abstract delves into the intricacies of such an endeavour, highlighting the fusion of Flutter's robust UI capabilities with efficient task management functionalities. By harnessing Flutter's rich widget library and reactive programming paradigm, the task manager offers an immersive user experience characterized by fluid navigation and visually appealing interfaces. Central to its design philosophy is the seamless integration of task creation, categorization, and scheduling features, enabling users to effortlessly organize their responsibilities. Leveraging Flutter's cross-platform compatibility ensures broad accessibility across diverse mobile devices, enhancing user reach and convenience. Furthermore, the task manager's synchronization capabilities with cloud services foster data continuity and accessibility, empowering users to stay productive anytime, anywhere. Through this abstract, we explore the technical nuances and user-centric design principles underpinning the Flutter-based task manager, poised to redefine personal productivity paradigms in the digital age.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	vii
	LIST OF ABBREVIATIONS	ix
	LIST OF FIGURES	xi
1	INTRODUCTION	1
1.1	Flutter	2
1.2	Dart	3
1.3	Back End	4
1.4	Problem Statement	5
1.5	Objectives	7
2	METHODOLOGY	10
2.1	Initial ProjectRequirements	10
2.2	Proposed SystemArchitecture	13
2.3	Flow Diagram	17
2.4	Modules	19
2.4.1	User Management	19
2.4.2	Security	20
2.4.3	creating Tasks	21
2.4.4	Organizing Tasks	22
2.4.5	Updating the tasks and deleting tasks	23
2.5	Development and Deployment Setup	24

3	RESULTS AND DISCUSSION	27
4	CONCLUSION AND FUTURE WORK	32
4.1	Conclusion	32
4.2	Future Work	33
	REFERENCES	34
	APPENDICES	35
A.	Sample Code	35
B.	Screen Shots	37

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	Name of the Figure No.	pg no.
2.1	Initial Project Requirements	10
2.2	Proposed Application Architecture	13
2.2	Android Studio Architecture	15
2.3	Flow diagram	17
2.4	Modules	19
2.5	Development and Deployment setup	24
3.1	Add Task	27
3.2	adding task display	28
3.3	Task Added	30
3.4	Task deleted	31

CHAPTER 1

INTRODUCTION

In today's fast-paced world, staying organized and productive is paramount. With the exponential rise in digital distractions and multitasking demands, individuals seek efficient solutions to manage their tasks effectively. Enter the task manager developed using Flutter a cutting-edge framework renowned for its versatility and seamless cross-platform capabilities. This task manager represents a fusion of intuitive design and powerful functionality, crafted to streamline task management workflows for users across diverse devices and operating systems. Harnessing Flutter's robust UI toolkit and reactive programming model, this application promises not only a visually engaging experience but also unparalleled performance and responsiveness. As we delve deeper into its features and development journey, we uncover how this Flutter-powered task manager stands poised to revolutionize the way individuals organize, prioritize, and accomplish their tasks in the digital era

In an age where time is precious and productivity is paramount, task management becomes the cornerstone of efficiency. Enter the task manager developed with Flutter – a versatile and powerful framework for building dynamic cross-platform applications. This task manager represents a convergence of sleek design and robust functionality, tailored to meet the diverse needs of users navigating the complexities of modern life. With Flutter's flexible UI components and seamless performance, this application offers a seamless user experience across both iOS and Android platforms. Through innovative features such as intuitive task organization, customizable reminders, and real-time synchronization, users can seamlessly In

our fast-paced digital age, juggling numerous tasks and responsibilities has become the norm. Enter the task manager crafted with Flutter – a versatile framework renowned for its ability to create beautiful, responsive apps for multiple platforms. This task manager isn't just about organizing to-do lists; it's a dynamic solution designed to elevate productivity and efficiency. With Flutter's flexible design system and native performance, this application delivers a consistent experience across Android and iOS devices, ensuring users can stay productive no matter their preferred platform.

1.1. FLUTTER

Flutter, a robust cross-platform UI toolkit developed by Google, is at the heart of Task Manager, shaping the application's foundation and user interface. The utilization of Flutter brings a paradigm shift to the traditional app development approach, offering a single codebase that transcends platform-specific limitations. This innovation allows Task Manager to deliver a visually captivating and responsive user interface, adeptly adjusting to both iOS and Android environments.

The efficiency of Flutter lies in its ability to streamline the development process, significantly reducing the complexities associated with managing separate codebases for different platforms. Developers benefit from a cohesive workflow, resulting in accelerated development cycles and enhanced productivity. Moreover, Flutter's hot reload feature enables real-time code changes and instant visual feedback, facilitating quick iterations and adjustments.

One of the key advantages of Flutter is its capability to ensure a consistent and high-quality user experience across diverse devices.

By leveraging Flutter's rich set of customizable widgets, Task Manager achieves a cohesive design language and smooth interactions, irrespective of the underlying operating system. Users can seamlessly transition between devices without compromising on the application's performance or aesthetics.

1.2. DART

i)Unified Codebase:

Task Manager Hub relies on Dart, a client-optimized programming language, for implementing backend logic and server-side operations. 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:

Dart's support for asynchronous programming is crucial for handling concurrent operations efficiently. This feature ensures that Task Manager can seamlessly manage multiple tasks, such as handling user requests, database operations, and external API calls, without causing delays or disruptions in the application's responsiveness.

iii)Scalability:

Dart's scalability is instrumental in accommodating the potential growth of Men. As the user base expands and the demand for managing tasks increases, 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:

Dart's strong typing system enhances code reliability and maintainability in Task Manager. By catching potential errors during the development phase, Dart enables developers to write robust and error-resistant code, reducing the likelihood of runtime issues and enhancing the overall stability of the application.

1.3. BACK END

i)Real -time Data Synchronization:

For a backend in a Flutter application, you would typically use a server-side technology such as Node.js, Django, Flask, or any other backend framework. This backend would handle tasks like managing data storage, processing business logic, and serving data to the Flutter application through APIs

ii) User Security:

Encrypt sensitive user data stored on the device, such as authentication tokens or locally cached task information. Use secure storage mechanisms like the Flutter Secure Storage package to store encrypted data securely. Implement secure session management to prevent session hijacking or fixation attacks.

1.4. PROBLEM STATEMENT

In today's fast-paced world, individuals often find themselves overwhelmed with numerous tasks, appointments, and deadlines across various aspects of their personal and professional lives. Managing these tasks efficiently and staying organized can be challenging, leading to stress, missed deadlines, and decreased productivity. Traditional methods such as handwritten to-do lists or basic task management tools often lack the functionality and flexibility required to meet modern-day demands.

To address these challenges, there is a need for a comprehensive task manager application that provides users with a centralized platform to plan, organize, prioritize, and track their tasks effectively. The task manager application should offer a user-friendly interface accessible across multiple devices, enabling seamless synchronization of tasks and real-time updates. It should empower users to create customizable task lists, set deadlines, assign priorities, categorize tasks, and collaborate with others when necessary. Additionally, the application should incorporate features such as reminders, notifications, calendar integration, progress tracking, and data analytics to enhance productivity and time management.

The task manager application should prioritize user privacy and security, implementing robust authentication mechanisms, data encryption, and compliance with relevant privacy regulations. It should be scalable, allowing for the management of tasks at both individual and team levels, with the flexibility to adapt to evolving user needs and preferences. Moreover, the application should be platform-agnostic, supporting mobile, web, and desktop environments to cater to a diverse user base.

To address these challenges, there's a critical need for a modern, intuitive, and feature-rich task manager application. This application should empower users to effortlessly create, organize, prioritize, and track their tasks and projects in a centralized and accessible platform. It should offer seamless synchronization across multiple devices, enabling users to stay productive whether they're at their desk or on the go. Moreover, the task manager should support collaboration features, allowing individuals and teams to share tasks, assign responsibilities, set deadlines, and communicate effectively within the application.

Security and privacy are paramount concerns in today's digital landscape. Therefore, the task manager application must prioritize robust security measures to safeguard user data against unauthorized access, data breaches, and malicious attacks. This includes implementing strong authentication mechanisms, encrypting sensitive information, securing communication channels, and adhering to privacy regulations to ensure user trust and confidence in the platform.

Additionally, the task manager application should offer intelligent features such as reminders, notifications, task categorization, search functionality, and insightful analytics to help users better understand their productivity patterns and make informed decisions. It should be highly customizable to accommodate diverse user preferences and workflows, catering to both individual users and organizations with varying needs and priorities.

1.5. OBJECTIVES

The primary objective of the task manager application is to provide users with a comprehensive and intuitive platform for managing their tasks and projects effectively. This involves enabling users to create, organize, prioritize, and track tasks seamlessly, thereby enhancing their productivity and reducing the likelihood of missed deadlines or overlooked responsibilities. One key objective is to offer a centralized hub where users can consolidate all their tasks and projects, regardless of their complexity or urgency. By providing a single interface for task management, the application aims to streamline users' workflow, minimize distractions, and facilitate better focus on essential tasks. Another objective is to promote collaboration and communication among individuals and teams by facilitating the sharing of tasks, assignment of responsibilities, setting of deadlines, and exchanging of updates and feedback within the application. This collaborative functionality aims to foster teamwork, improve coordination, and ensure that everyone stays aligned towards common goals.

Key Challenges:

i) Integration of Diverse Features:

The task manager integrates essential features such as task creation, organization, prioritization, and tracking. Users can easily create new tasks, categorize them based on project or category, set deadlines, and monitor progress towards completion.

ii) Inclusive Platform Design:

Designing an inclusive platform for a task manager using Flutter involves creating an environment that prioritizes accessibility, usability, and diversity to ensure that all users, regardless of their background, abilities, or preferences, can effectively engage with the application.

iii) Security and Privacy:

Ensuring robust security measures to protect user data, particularly in the context of managing tasks.

Adhering to privacy regulations and standards to build trust among users regarding the confidentiality of their information.

iv) Scalability and Performance:

Designing the app architecture to accommodate potential growth in user base and content without compromising performance. Optimizing the performance of Task Manager to deliver a responsive and efficient experience, especially as the user community expands.

v) Flutter Codebase Maintenance:

Managing the complexity of a Flutter-based codebase to ensure code readability, maintainability, and scalability. Keeping the codebase up-to-date with the latest Flutter releases and best

practices to leverage new features and improvements. Organize the codebase into modular components, such as widgets, screens, services, and models, following principles like separation of concerns and single responsibility. Modular architecture promotes code reuse, scalability, and maintainability.

CHAPTER 2

METHODOLOGY

2.1. INITIAL PROJECT REQUIREMENTS

The initial project requirements for Task Manager encompass various aspects such as functionalities, technologies, and packages.

i)Project Scope:

- 1) Develop a mobile application named "Task Manager".
- 2) Create a platform that help to us to manage our tasks efficiently
It can help us to add, delete task very efficiently
- 3) Address both academic and job-related tasks within the platform.

ii)Technologies:

- 1) Flutter: Utilize Flutter for cross-platform mobile app development, ensuring a consistent user experience on both iOS and Android.
- 2) Dart: Employ the Dart programming language as the primary language for Flutter app development.

iii)Functionalities:

- 1) Task Creation: Allow users to create new tasks with titles, descriptions, priorities.
- 2) Task Organization: Enable users to organize tasks into lists, folders, or categories based importance of project and custom criteria.
- 3) Task Editing and Deleting: Allow users to edit existing tasks to update information such as title, description also provide the ability to delete tasks when they're no longer needed.

iv)Packages Used:

i)get:

Version: 4.6.6

Purpose: State management and navigation framework.

ii) flutter/material.dart

Version: 3.19

Purpose: This is the core package provided by Flutter for building UIs.

iii)google_fonts:

Version:2.1.0

Purpose: Easily use any of the google fonts in your flutter app

iv)flutter_bloc

Version: 7.0.0

Purpose: state management library based on the BLoC(Business Logic Component)pattern

v)flutter_svg:

Version:0.23.0

Purpose: Render SVG(Scable Vector Graphics)images in Flutter.

vi)flutter_map:

Version:0.13.0

Purpose: Interactive,customizable maps for Flutter applications

vii) flutter_spinkit:

Version: 5.1.0

Purpose: Collection of loading indicators animated with flutter.

viii)flutter_localizations

Version: 0.9.3

Purpose: Flutter package containing localization and internationalization support.

ix)toast:

Version: 0.3.0

Purpose: Displaying simple toast notifications.

x)fluttoast:

Version: 8.0.7

Purpose: Displaying customizable toast messages.

2.2. PROPOSED APPLICATION ARCHITECTURE

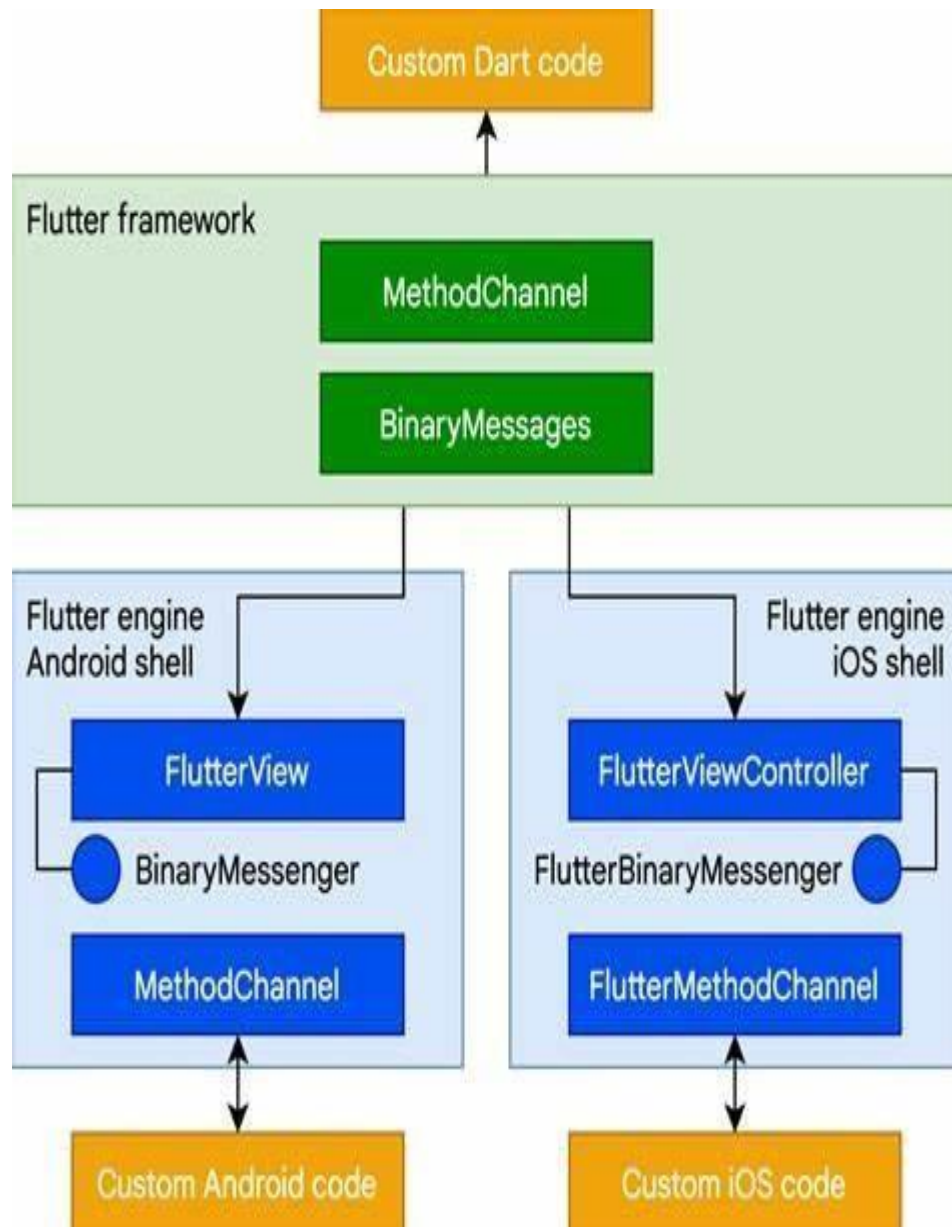


Fig:2.1: Proposed Application Architecture

1.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.

2.Flutter Framework:

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

3.MethodChannel:

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

4.BinaryMessages:

BinaryMessages is another channel for communication between Dart and native code.It enables the exchange of binary data (such as images, files, or serialized objects) between the two sides.

5.Flutter Engine Android Shell:

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

i)FlutterView: This is the Android view that displays your Flutter UI.

ii)BinaryMessenger: It handles communication between the Flutter engine and the Android platform.

6.Flutter Engine iOS Shell:

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

i)FlutterViewController: The iOS view controller responsible for displaying your Flutter UI.

ii)FlutterBinaryMessenger: Handles communication between the Flutter engine and the iOS platform.

7.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 with native features or APIs that aren't directly accessible from Flutter.

8.Android Studio:

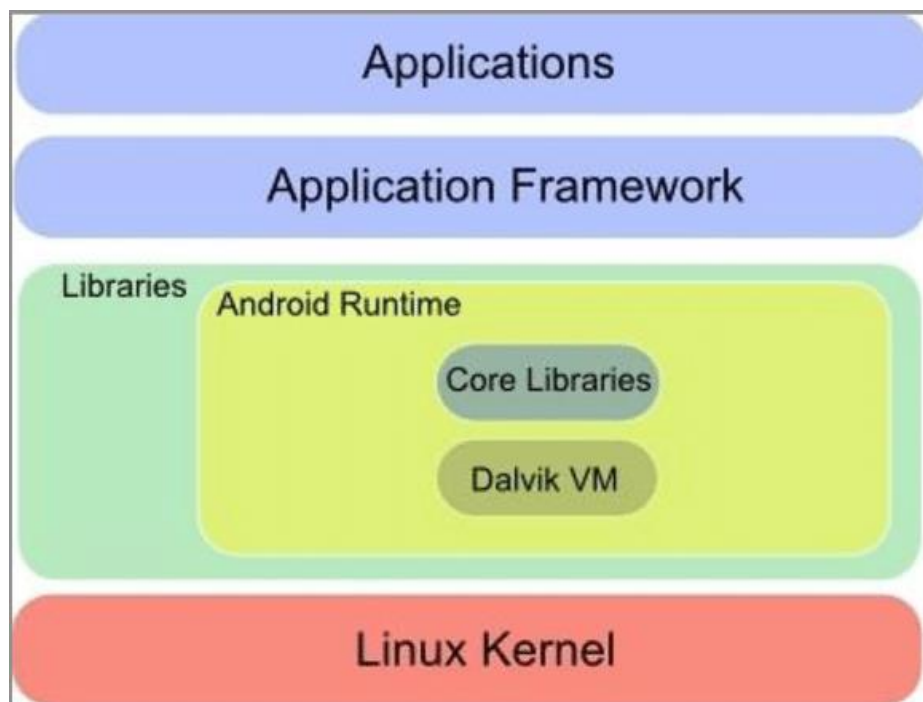


Fig:2.2: Android Studio Architecture

i)Flutter:

Flutter is a UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. It uses the Dart programming language, which is also developed by Google, known for its simplicity and efficiency.

ii)Flutter Toast:

In Flutter, a "toast" is a small, non-intrusive message that appears momentarily on the screen to provide feedback or information to the user. Flutter provides packages like `fluttertoast` that simplify the implementation of toasts in your app.

iii)Flutter Sliders:

creating moving slides or sliders is a dynamic way to engage users with interactive content. Sliders allow users to input a value by dragging a thumb along a track, providing a visual representation of a range of values. Flutter's `Slider` widget provides a simple yet customizable solution for implementing sliders in your app.

iv) SQLite database:

SQLite database through the `sql` package. Specifically, it uses functions like `open Database` to open a database connection, `execute()` to execute SQL queries, `insert` to insert data into tables, `update()` to update existing records, `raw Query` to execute raw SQL queries, and `get Databases Path` to get the path where databases are stored on the device.

2.3. FLOW DIAGRAM

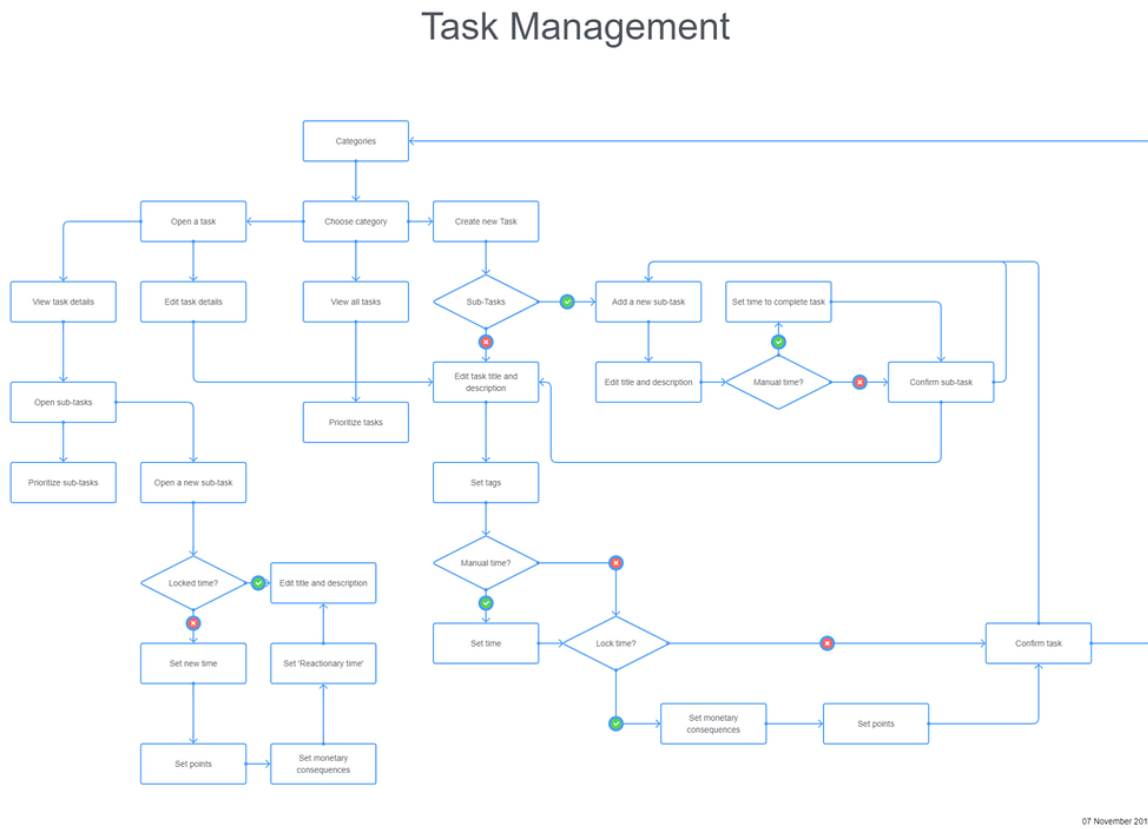


Fig:2.3: Flow Diagram

The following steps are used to describe the above flow diagram:

- i) The user opens the mobile app.
- ii) In “fig 2.3” The users can see the option to create the task .

- iii)** In “fig2.3” After started creating task use can see the options in like enter title, enter task description, select task type, select task status.
- iv)** When user starts creating task first they will add task title and it should be minimum five characters should be we can see in fig(2.3)
- v)** In “fig2.3” After creating title we should create description of task and it should be minimum 10 characters
- vi)** In “fig 2.3” After creating description we should give task type in our app there are four types those are urgent important ,urgent not important, not urgent important, not urgent not important.
- vii)** After that we should set task status so that we can get one idea which task is completed and which is on going and which is incomplete and press on save option that In “fig2.3”
- viii)** In task status totally 3 types completed, incomplete, ongoing.
- ix)** After creating task we can delete the task just by dragging side.
- x)** In “fig 2.3” If you want to update the task just press on task which you want to update and make changes and again save it.

2.4. MODULES

2.4.1.USER MANAGEMENT:

Flutter Task Manager is a powerful and intuitive application designed to help you streamline your daily tasks and manage your busy schedule effortlessly. Built using Flutter, Google's UI toolkit for building natively compiled applications, our app provides a seamless user experience across multiple platforms, including Android and iOS

Algorithm:

- i) Efficient Task Creation: Quickly add new tasks with just a few taps. Our app supports adding task titles, descriptions, due dates, and task types, allowing you to capture all the necessary details in one place.

- ii)Flexible Task Management: Edit, delete, or update tasks on the go. Whether you need to modify task details or reschedule deadlines, our app makes task management a breeze.

- iii)Sync Across Devices: Keep your tasks in sync across multiple devices seamlessly. With our cloud-sync feature, your tasks are securely stored and accessible from any device, ensuring you never miss a beat, whether you're at home, in the office, or on the go.

- iv)Task Organization: Easily create, categorize, and prioritize your tasks to stay on top of your to-do list. With our intuitive interface, you can organize tasks into customizable categories and mark them as complete when finished.

2.4.2. SECURITY:

Developed using Flutter, a versatile UI toolkit, and powered by SQLite, a reliable and secure database engine, our app prioritizes the protection of your sensitive information without compromising on functionality.

Algorithm:

i) Secure Data Storage: Rest easy knowing that your task data is securely stored using SQLite, a lightweight and reliable database engine. With SQLite, your data is encrypted and protected, safeguarding your privacy and sensitive information.

ii)Secure: We are committed to providing a secure and reliable task management experience. Our team continuously monitors for security vulnerabilities and releases regular updates to patch any identified issues promptly. This ensures that your app remains resilient against emerging threats and vulnerabilities.

iii)Safety: With SQLite, your data is encrypted and protected, safeguarding your privacy and sensitive information. This makes it extremely difficult for malicious actors to decipher or manipulate your data, even if they gain access to the underlying database files.

2.4.3. CREATING TASKS:

Enable users to create tasks efficiently. Any type of tasks can be added to your task manager.

Algorithm:

i) Creating Tasks: After started creating task use can see the options like enter title, enter task description, select task type, select task status. This class should include properties such as title, description, due date, priority, and completion status.

ii) Editing Tasks: Allow users to edit existing tasks. When a user selects a task from the list, navigate to a screen with a pre-filled form containing the task details. Update the task object with the edited details and save it .

iii) Organizing Tasks: Easily create, categorize, and prioritize your tasks to stay on top of your to-do list. With our intuitive interface, you can organize tasks into customizable categories and mark them as complete when finished.

2.4.4. ORGANIZING TASKS:

Organizing tasks in a Flutter task manager app involves providing users with tools and features to categorize, prioritize, and group their tasks efficiently.

Algorithm:

i) Categories: Allow users to categorize tasks into different groups or labels. For example, users might have categories like "Work," "Personal," "Shopping". Implement a feature where users can assign each task to one or more categories.

ii) Drag: Implement a drag-and-drop interface that allows users to reorder tasks manually. This feature enables users to prioritize tasks by rearranging them in the task list according to their preference.

iii) Customization Options: Provide users with customization options to personalize their task organization preferences. Allow users to customize category names, priority levels, sorting criteria, and other settings according to their workflow and preferences.

iv) Task Details Screen: Create a screen to display detailed information about a selected task. This screen should show all the properties of the task, such as title, description, due date, and completion status.

2.4.5. UPDATING TASKS AND DELETING TASKS:

When a user selects a task from the task list, navigate to a screen where they can view and edit the task details. Populate the edit screen with the current details of the selected task, allowing the user to make changes as needed.

Algorithm:

- i) Provide input fields or widgets for modifying task properties such as title, description, due date, priority, etc. Upon submitting the changes, update the corresponding task object in your data model with the new values.

- ii) Optionally, you can display a confirmation dialog to confirm the update before saving the changes. Once the task is successfully updated, navigate back to the task list screen to reflect the changes.

- iii) Implement a mechanism for users to delete tasks from their task list. Add a delete button or gesture to each task item in the task list. When the user triggers the delete action for a task, prompt them with a confirmation dialog to confirm the deletion.

2.5. DEVELOPMENT AND DEPLOYMENT SETUP

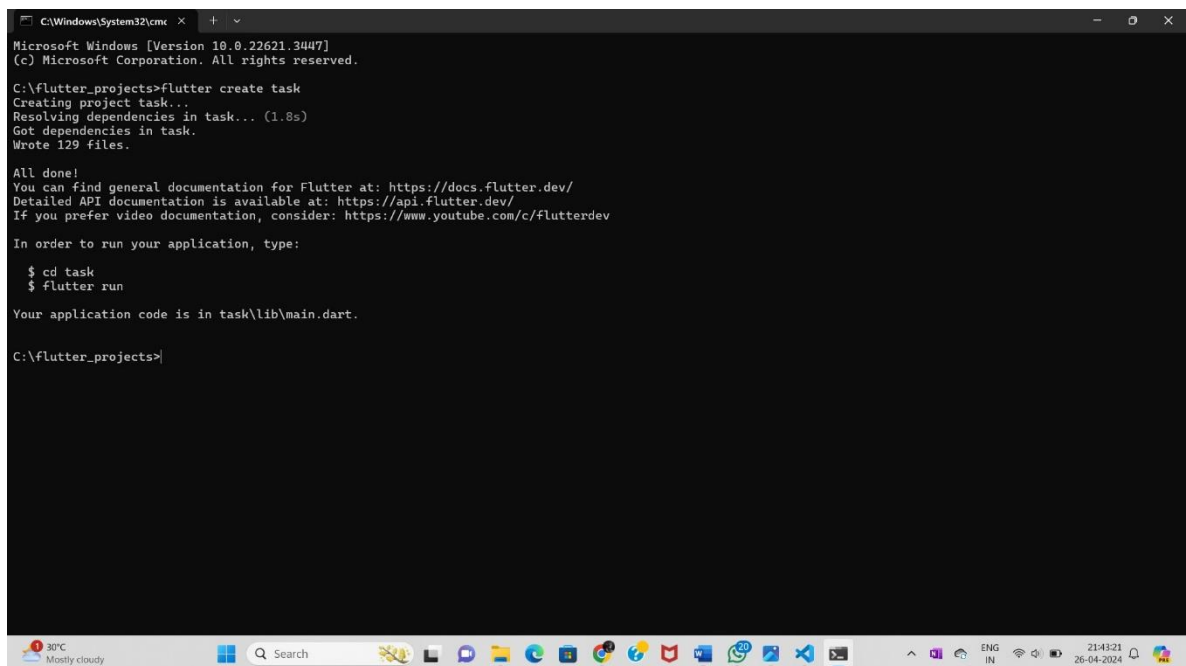
1. Development Environment:

Flutter and Dart need to be installed on the developer's machine.

IDEs like Visual Studio Code or Android Studio for coding.

2. Create Project:

>>flutter create project_name



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.3447]
(c) Microsoft Corporation. All rights reserved.

C:\flutter_projects>flutter create task
Creating project task...
Resolving dependencies in task... (1.8s)
Got dependencies in task.
Wrote 129 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider: https://www.youtube.com/c/Flutterdev

In order to run your application, type:

$ cd task
$ flutter run

Your application code is in task\lib\main.dart.

C:\flutter_projects>
```

Fig:2.4: Project Creation

3.Dependency Management:

Project dependencies specified in the pubspec.yaml file.

Key packages include:

- flutter pub get in the terminal to download and install the specified packages
- Flutter pub upgrade command to upgrade dependencies to their latest.
- Flutter generates a pubspec.lock file.
- Other packages like get, toast, image_picker, etc., for additional functionalities.

4.SQLITE:

SQLite does not require a separate server process to operate. It is self-contained, meaning the entire database engine is integrated into the application. This makes SQLite easy to deploy and manage, as there is no need to install, configure, or manage a database server.

```
import 'package:sqflite/sqflite_api.dart';
import 'package:taskly/data/datasource/sql_constants.dart';
import 'package:taskly/data/datasource/task_data_source.dart';
import 'package:taskly/domain/models/task.dart';
import 'package:path/path.dart' as path;
import 'package:sqflite/sqflite.dart' as sql;
import 'package:taskly/domain/models/task_type.dart';
import 'package:taskly/data/datasource/transformer.dart';
```

Fig:2.5: sqlite Connection

5.Development Tools:

Flutter commands used for testing and building the application.Real-time testing on Android and iOS devices.



Fig:2.7: Flutter Run

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)routes.dart: It will navigate the user to the home screen of the app. It will navigate the user to the task screen where they can add a new task.

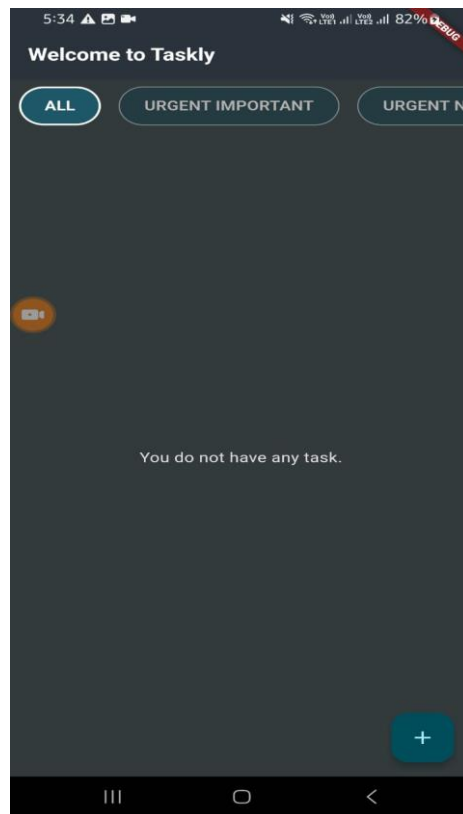


Fig:3.1: Add Task

iii) Quickly add new tasks with just a few taps. Our app supports adding task titles, descriptions, due dates, and task types, allowing you to capture all the necessary details in one place.

iv) Edit, delete, or update tasks on the go. Whether you need to modify task details or reschedule deadlines, our app makes task management a breeze.

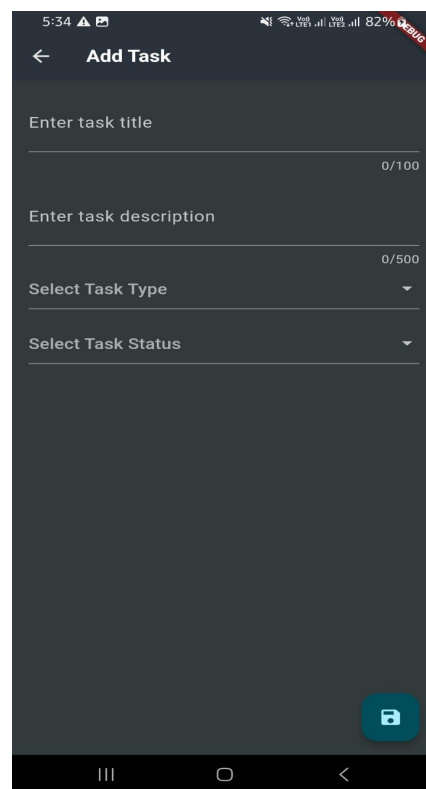


Fig:3.2: Adding Task Display

vi) flutter/material.dart: This package provides the necessary widgets and utilities for building Material Design-compliant apps in Flutter.

vii) It extends StatelessWidget and overrides the build() method to define the application's UI.

viii) In this code, ProviderScope is used to wrap the entire application with a scope for managing providers.

ix) The theme property of MaterialApp is customized using ThemeData.dark().copyWith() to create a dark theme.

x) This function navigates to the HomeScreen by replacing the current route in the navigation stack with a new MaterialPageRoute containing the HomeScreen

xi) Use the flutter pub upgrade command to upgrade dependencies to their latest versions while respecting version constraints specified in the pubspec.yaml file.

xii) sqflite: A SQLite database plugin for Flutter that provides APIs for interacting with SQLite databases on both iOS and Android.



Fig:3.3: Task Added

xiii) provider.dart, Fluttertoast.showToast An extended version of the fluttertoast package with additional features such as toast position customization, toast animation customization, and support for custom toast widgets.

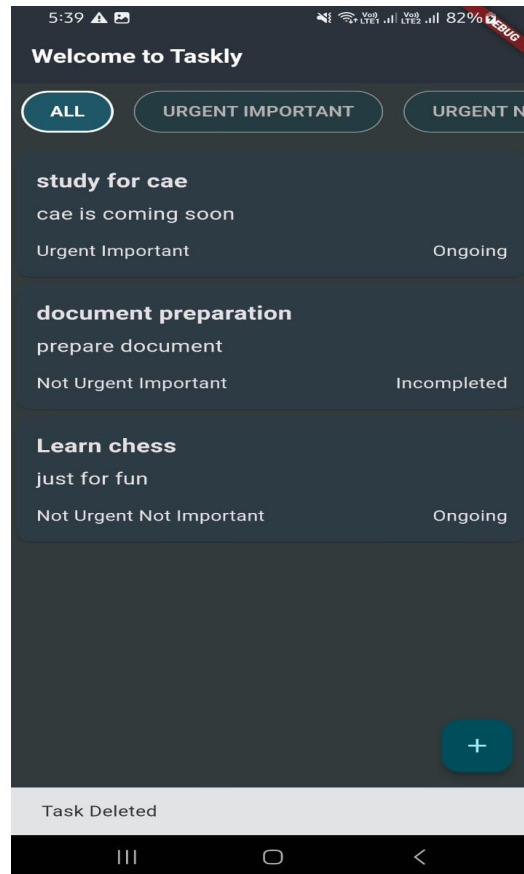


Fig:3.4: Task Deleted

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1. CONCLUSION

In conclusion, the development of a task manager application using Flutter offers a promising solution for efficiently managing tasks across various platforms. By leveraging the capabilities of Flutter, developers can create a seamless and intuitive user experience with a single codebase for both iOS and Android platforms.

The cross-platform nature of Flutter reduces development time and effort, allowing for faster deployment and updates. Additionally, the robust widget library and customizable UI elements enable developers to design a visually appealing and feature-rich task manager application. However, challenges such as ensuring data security, optimizing performance, and addressing compatibility issues across different devices may arise.

Despite these challenges, the versatility and flexibility of Flutter provide ample opportunities for building a high-quality task manager application that meets the needs of modern users. Overall, the adoption of Flutter for developing a task manager application holds great potential for streamlining task management processes and enhancing productivity in various domains.

4.2. FUTURE WORK

- 1.** Looking ahead, there are several avenues for future work in the development of task manager applications using Flutter. One promising direction is the integration of advanced features such as natural language processing (NLP) for task input and management. By incorporating NLP capabilities, users can interact with the task manager using natural language commands, making task creation and management more intuitive and efficient.

- 2.** Additionally, enhancing collaboration and communication features within the app, such as real-time task sharing and team collaboration functionalities, could further improve productivity and teamwork among users.

- 3.** Moreover, continuous optimization of the app's performance, responsiveness, and user experience across different devices and platforms will be essential to ensure seamless operation and user satisfaction. By incorporating NLP capabilities, users can interact with the task manager using natural language commands, making task creation and management more intuitive and efficient.

- 4.** Furthermore, exploring the integration of machine learning algorithms for personalized task recommendations and predictive analytics could provide valuable insights to users and help optimize task management workflows.

REFERENCES

Journals:

- A Comparison of Performance and Looks Between Flutter and Applications, Matilda Olsson June 13th, 2020.
- A flashcard mobile application development with Flutter Tung Huynh Haaga- Helia University of Applied Sciences Bachelor's Thesis 2021 Bachelor of Business Administration.

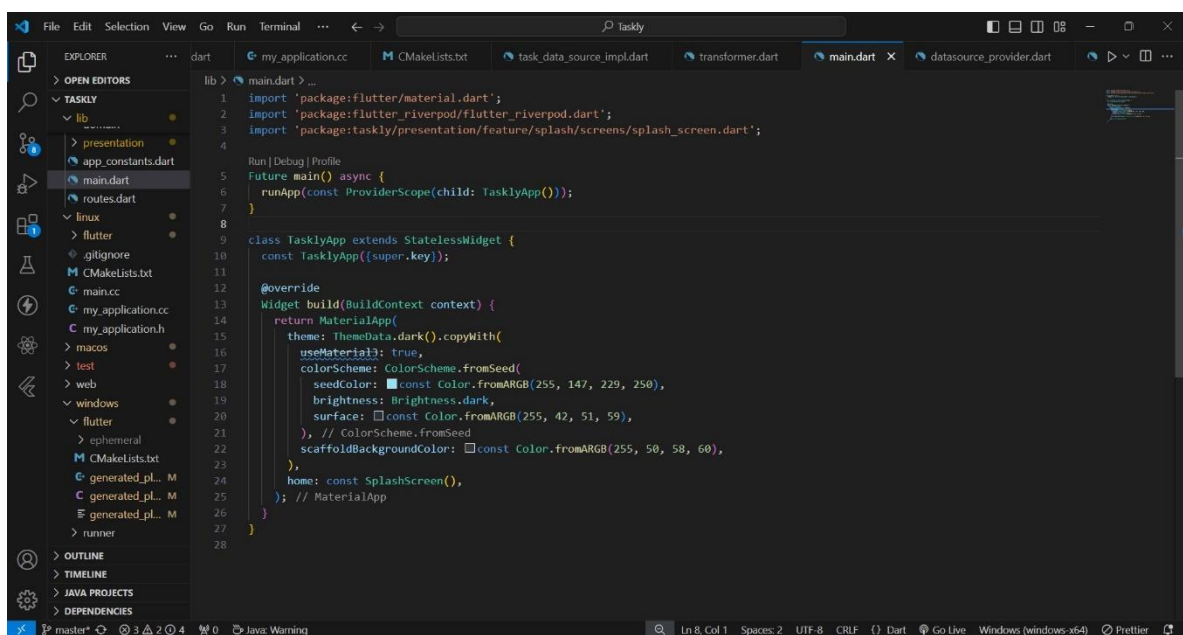
Books:

- "Building Virtual Communities for Knowledge Exchange" M. Davis, 2021.
- "Flutter for Mobile App Development in Education" K. Patel, 2019.
- "Fostering Collaboration between Academia and Industry" S. Williams, 2021.
- "Hackathons and Learning: A Practical Approach" , : E. Rodriguez, 2020.
- "How Technology is Transforming Mentorship in Education" , John Doe .

APPENDICES

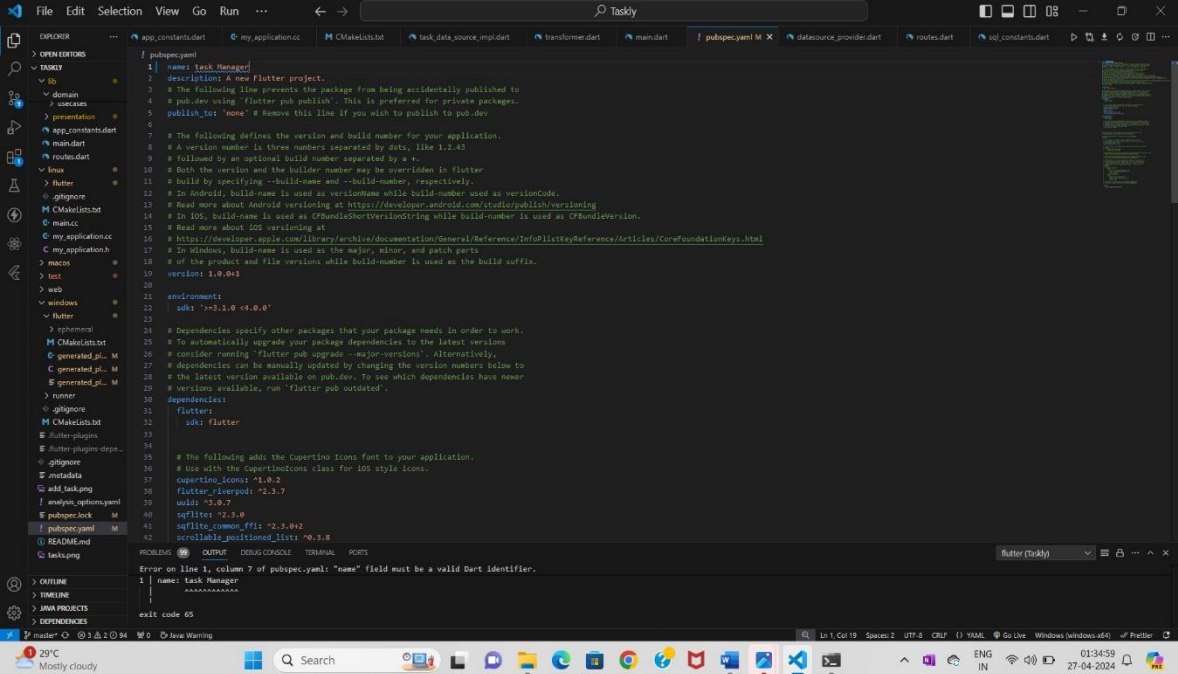
A.SAMPLE CODE

i)Main Code:



```
lib > main.dart > ...
1 import 'package:flutter/material.dart';
2 import 'package:flutter_riverpod/flutter_riverpod.dart';
3 import 'package:taskly/presentation/feature/splash/screens/splash_screen.dart';
4
5 Run | Debug | Profile
6 Future main() async {
7   runApp(const ProviderScope(child: TasklyApp()));
8 }
9
10 class TasklyApp extends StatelessWidget {
11   const TasklyApp({super.key});
12
13   @override
14   Widget build(BuildContext context) {
15     return MaterialApp(
16       theme: ThemeData.dark().copyWith(
17         useMaterial3: true,
18         colorScheme: ColorScheme.fromSeed(
19           seedColor: const Color.fromARGB(255, 147, 229, 250),
20           brightness: Brightness.dark,
21           surface: const Color.fromARGB(255, 42, 51, 59),
22           // ColorScheme.fromSeed
23           scaffoldBackgroundColor: const Color.fromARGB(255, 58, 60),
24         ),
25       home: const SplashScreen(),
26     ); // MaterialApp
27   }
28 }
```

ii)pubspec.yaml



The screenshot shows an IDE window with the `pubspec.yaml` file open. The file content is as follows:

```
1 name: Task Manager
2 description: A new Flutter project.
3 # The following line prevents the package from being accidentally published to
4 # pub.dev using "flutter pub publish". This is preferred for private packages.
5 publish_to: 'none' # Remove this line if you wish to publish to pub.dev
6
7 # The following defines the version and build number for your application.
8 # A version number is three numbers separated by dots, like 1.2.43.
9 # followed by an optional build number separated by a +.
10 # Both the version and the build number may be overridden in flutter
11 # build by specifying --build-name and --build-number, respectively.
12 # In Android, build-name is used as versionName while build-number used as versionCode.
13 # Read more about Android versioning at https://developer.android.com/studio/publish/versioning
14 # In iOS, build-name is used as CFBundleShortVersionString while build-number is used as CFBundleVersion.
15 # Read more about iOS versioning at
16 # https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html
17 # In Windows, build-name is used as the major, minor, and patch parts
18 # of the product and file versions while build-number is used as the build suffix.
19 version: 1.0.0+1
20
21 environment:
22   sdk: '>=3.1.0 <4.0.0'
23
24 # Dependencies specify other packages that your package needs in order to work.
25 # To automatically upgrade your package dependencies to the latest versions
26 # consider running "flutter pub upgrade --major-versions". Alternatively,
27 # dependencies can be manually updated by changing the version numbers below to
28 # the latest version available on pub.dev. To see which dependencies have newer
29 # versions available, run "flutter pub outdated".
30 dependencies:
31   flutter:
32     sdk: flutter
33
34 # The following adds the Cupertino Icons font to your application.
35 # Use with the CupertinoIcons class for iOS style icons.
36 cupertino_icons: ^1.0.2
37 flutter_riverpod: ^2.3.7
38 uuid: ^3.0.7
39 sqflite: ^2.3.0
40 sqflite_common_ffi: ^2.3.0+2
41 sqflite_common_ffi_wasm: ^0.9.0
42
43 # An asset of an image file to be included with the application.
44 assets:
45   - assets/images/
46
47 # An asset of a font file to be included with the application.
48 fonts:
49   - family: sans-serif
50     fonts:
51       - asset: assets/fonts/sans-serif.ttf
52       - asset: assets/fonts/sans-serif.woff2
53       - asset: assets/fonts/sans-serif.woff
54       - asset: assets/fonts/sans-serif.woff2
55       - asset: assets/fonts/sans-serif.woff
56       - asset: assets/fonts/sans-serif.woff2
57       - asset: assets/fonts/sans-serif.woff
58       - asset: assets/fonts/sans-serif.woff2
59       - asset: assets/fonts/sans-serif.woff
60       - asset: assets/fonts/sans-serif.woff2
61       - asset: assets/fonts/sans-serif.woff
62       - asset: assets/fonts/sans-serif.woff2
63       - asset: assets/fonts/sans-serif.woff
64       - asset: assets/fonts/sans-serif.woff2
65       - asset: assets/fonts/sans-serif.woff
66       - asset: assets/fonts/sans-serif.woff2
67       - asset: assets/fonts/sans-serif.woff
68       - asset: assets/fonts/sans-serif.woff2
69       - asset: assets/fonts/sans-serif.woff
70       - asset: assets/fonts/sans-serif.woff2
71       - asset: assets/fonts/sans-serif.woff
72       - asset: assets/fonts/sans-serif.woff2
73       - asset: assets/fonts/sans-serif.woff
74       - asset: assets/fonts/sans-serif.woff2
75       - asset: assets/fonts/sans-serif.woff
76       - asset: assets/fonts/sans-serif.woff2
77       - asset: assets/fonts/sans-serif.woff
78       - asset: assets/fonts/sans-serif.woff2
79       - asset: assets/fonts/sans-serif.woff
80       - asset: assets/fonts/sans-serif.woff2
81       - asset: assets/fonts/sans-serif.woff
82       - asset: assets/fonts/sans-serif.woff2
83       - asset: assets/fonts/sans-serif.woff
84       - asset: assets/fonts/sans-serif.woff2
85       - asset: assets/fonts/sans-serif.woff
86       - asset: assets/fonts/sans-serif.woff2
87       - asset: assets/fonts/sans-serif.woff
88       - asset: assets/fonts/sans-serif.woff2
89       - asset: assets/fonts/sans-serif.woff
90       - asset: assets/fonts/sans-serif.woff2
91       - asset: assets/fonts/sans-serif.woff
92       - asset: assets/fonts/sans-serif.woff2
93       - asset: assets/fonts/sans-serif.woff
94       - asset: assets/fonts/sans-serif.woff2
95       - asset: assets/fonts/sans-serif.woff
96       - asset: assets/fonts/sans-serif.woff2
97       - asset: assets/fonts/sans-serif.woff
98       - asset: assets/fonts/sans-serif.woff2
99       - asset: assets/fonts/sans-serif.woff
100      - asset: assets/fonts/sans-serif.woff2
```

An error message is displayed at the bottom of the editor:

```
Error on line 1, column 7 of pubspec.yaml: "name" field must be a valid Dart identifier.
```

The error points to the `name: Task Manager` line. The IDE interface includes a sidebar with a file explorer, a top toolbar with various icons, and a bottom status bar showing the current file and line information.

B.SCREEN SHOTS

i)Add Task

5:34 82% DEBUG

← Add Task

Enter task title 0/100

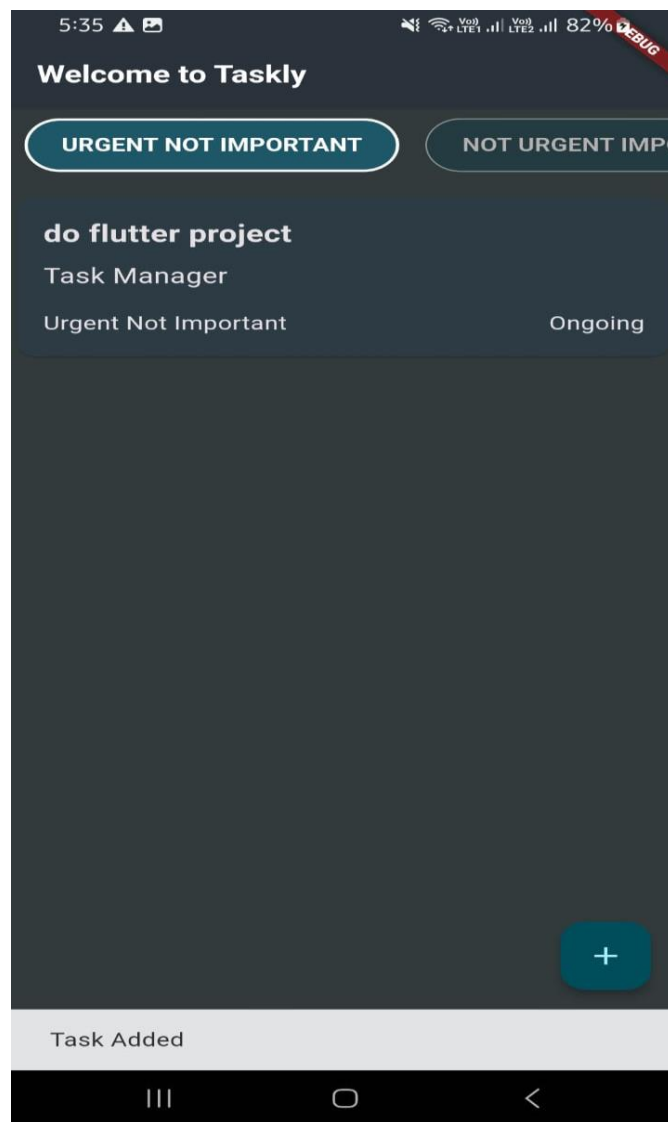
Enter task description 0/500

Select Task Type ▼

Select Task Status ▼

Save

ii) Task added



iii)Task Deleted

