

Daily News App

INTERDISPLINARY PROJECT

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

by

ASHLYN TIMOTHEY D(41120031)



**DEPARTMENT OF INFORMATION TECHNOLOGYSCHOOL
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-600119

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 **ASHLYN TIMOTHEY D (41120031)** who carried out the project entitled "**DAILY NEWS APP**" under my supervision from January 2024 to May 2024.

Internal Guide

Dr.P.JEYANTHI, M.E., 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

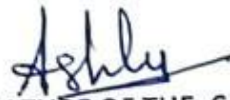
External Examiner

DECLARATION

I Ashlyn Timothy D (41120031) hereby declare that the Project Report entitled "DAILY NEWS APP" done by me under the guidance of Dr.P.JEYANTHI, M.E., Ph.D., is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 8/5/24

PLACE: CHENNAI


SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

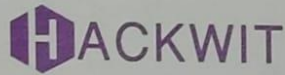
I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

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.P.JEYANTHI, M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved wayfor 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 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 **Ashlyn Timothy D (Reg No: 41120031)** 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, **Ashlyn Timothy D** has closely worked as a part of the **Flutter** by carrying out the project entitled "**Daily news**"

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, staying updated with the latest news is essential. Mobile applications have become the go-to platform for accessing news conveniently. This project focuses on the development of a Daily News App using Flutter and Dart programming languages, providing users with a seamless and intuitive interface to access news content effortlessly. The app is designed to deliver a user-friendly experience, incorporating features such as personalized news feeds, real-time updates, category-based browsing, and user preferences customization. Leveraging Flutter's cross-platform capabilities, the app ensures compatibility with both Android and iOS devices, reaching a wide user base. The backend of the app utilizes APIs to fetch news articles from reputable sources, ensuring reliability and relevance of the content presented to the users. Through efficient data management techniques, the app optimizes loading times and minimizes data usage, enhancing user satisfaction. Throughout the development process, best practices in UI/UX design, software architecture, and performance optimization are adhered to, ensuring a polished and responsive application. The project report discusses the design choices, development methodologies, challenges faced, and future enhancements for the Daily News App. Overall, the Flutter-Dart Daily News App aims to provide users with a comprehensive, efficient, and enjoyable platform for accessing news content on a daily basis, catering to the diverse needs and preferences of modern audiences.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	i
	LIST OF ABBREVIATIONS	iv
	LIST OF FIGURES	v
1	INTRODUCTION	6
	1.1 Flutter	7
	1.2 Dart	8
	1.3 Firebase	9
	1.4 Problem Statement	10
	1.5 Objectives	12
2	METHODOLOGY	15
	2.1 Initial project Requirements	15
	2.2 Proposed Application Architecture	18
	2.3 How news app request from the API service	24
	2.4 Integrate API with flutter	26
	2.5 Modules	22
	2.5.1 Profile management	29
	2.6 Development and Deployment Setup	32
3	RESULTS AND DISCUSSION	36

4	CONCLUSION AND FUTUREWORK	43
4.1	Conclusion	43
4.2	Future Work	44
	REFERENCES	44
	APPENDICES	45
A.	Sample Code	45
B.	Screen Shots	47

LIST OF ABBREVIATIONS

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

LIST OF FIGURES

FIGURE NO.	NAME OF THE FIGURE PAGE	PAGE NO.
2.1	Proposed Application Architecture	18
2.2	API	20
2.3	Integrate API with flutter	26
2.4	Project creation	32
2.5	Flutter Run	35
3.1	Injection_container.dart	36
3.2	Home page	40
3.3	Configurations	42

CHAPTER 1

INTRODUCTION

In an era dominated by rapid digitalization and information consumption, staying updated with the latest news and events has become an integral part of our daily routine. With the surge in smartphone usage and the popularity of mobile applications, the demand for accessible and personalized news delivery platforms has skyrocketed. Recognizing this trend, our project aims to address this need by developing a user-friendly and efficient Daily News Article App using the Flutter framework. Flutter, Google's UI toolkit for building natively compiled applications for mobile, web, and desktop from a single codebase, provides a seamless environment for crafting cross-platform applications. Leveraging its versatility and robust features, we embarked on the journey to create an intuitive and feature-rich news application that caters to the diverse needs of modern-day users. This project report delves into the intricate details of our endeavor, encompassing the inception, development process, features, challenges encountered, and future prospects of our Daily News Article App. Through meticulous planning, innovative design, and rigorous testing, we endeavored to create a cutting-edge application that not only delivers news content but also enhances the overall user experience through intuitive navigation, customization options, and real-time updates.

1.1. FLUTTER

Flutter, a robust cross-platform UI toolkit developed by Google, is at the heart of Daily News App, 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 Daily News App Hub 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, Daily News App 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:

Daily News App 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 Daily News App 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 Daily News App. As the user base expands and the demand for mentorship 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 Daily News App. 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. FIREBASE

i) Real -time Data Synchronization:

Firebase Cloud Firestore supports real-time data synchronization, allowing seamless updates across multiple devices. This feature is crucial for Daily News App, enabling mentors and students to experience instant changes in discussions, mentorship requests, and collaborative interactions.

ii) Firebase Authentication for User Security:

Firebase Authentication provides a secure and straightforward user authentication process. Daily News App leverages this feature to ensure that only authorized users, both mentors and students, can access the platform. Multi-factor authentication and identity verification contribute to a robust security layer.

iii) Serverless Cloud Functions:

Firebase offers serverless cloud functions that enable the execution of backend logic without managing servers. Daily News App utilizes these functions for various tasks, such as sending notifications, updating user data, and handling asynchronous operations. This serverless architecture enhances scalability and efficiency.

iv) Firebase Cloud Storage for Media Assets:

Storing media assets, such as profile pictures and discussion attachments, is streamlined with Firebase Cloud Storage. Daily News App utilizes this feature to efficiently manage and serve multimedia content, ensuring a smooth and responsive user experience for both mentors and students.

1.4.PROBLEM STATEMENT

In today's digitally-driven society, the proliferation of information sources has led to an overwhelming amount of news content available to consumers. However, amidst this abundance, users often face challenges in efficiently accessing, organizing, and staying updated with news articles that align with their interests, preferences, and reliability standards. The fragmented nature of news delivery platforms, coupled with varying levels of credibility across sources, exacerbates these challenges, resulting in:

Information Overload:

Users are inundated with a vast array of news articles from disparate sources, making it challenging to filter through the noise and identify articles relevant to their interests and concerns.

Lack of Personalization:

Existing news applications often provide limited options for personalization, offering generic news feeds that may not cater to the specific preferences and topics of individual users.

Credibility Concerns:

With the rise of misinformation and fake news, users struggle to discern the credibility and accuracy of news sources, leading to skepticism and uncertainty regarding the information they consume.

User Engagement:

Many news applications fail to provide engaging and interactive features that encourage users to actively participate in the news consumption process, resulting in passive consumption habits and reduced user engagement.

To address these pressing challenges and enhance the news consumption experience for users, our project aims to develop a Daily News Article App using Flutter and Dart. This application seeks to provide a comprehensive solution.

Personalized News Feeds:

Implementing algorithms and user preferences to deliver tailored news feeds based on topics, interests, and reading habits, ensuring users receive relevant and engaging content.

Curation and Aggregation:

Curating articles from reputable sources and aggregating them within a single platform, offering users a centralized hub for accessing diverse news content while maintaining high standards of credibility and reliability.

Interactive Features:

Incorporating interactive features such as bookmarking, sharing, commenting, and personalized recommendations to enhance user engagement and foster a sense of community among users.

Real-time Updates:

Providing timely updates and notifications for breaking news stories, ensuring users stay informed about the latest developments across various topics of interest.

Seamless User Experience:

Leveraging the capabilities of Flutter to create a visually appealing, responsive, and user-friendly interface that offers smooth navigation and intuitive interactions.

By addressing these key pain points and delivering innovative features, our Daily News Article App aims to revolutionize the way users consume and engage with news content, empowering them to make informed decisions, stay connected with global events, and navigate the digital news landscape with confidence and ease.

1.5.OBJECTIVES

User Engagement:

Create an intuitive and visually appealing user interface to encourage active engagement with news articles, fostering a sense of community and interaction among users.

Personalization:

Implement algorithms to analyze user preferences and behavior, delivering personalized news feeds tailored to individual interests and reading habits.

Curation and Reliability:

Curate news articles from reputable sources while implementing measures to verify the credibility and accuracy of content, ensuring users receive trustworthy information.

Real-time Updates:

Provide timely notifications for breaking news stories and updates, keeping users informed about the latest developments across various topics.

Cross-platform Compatibility:

Develop a cross-platform application compatible with both Android and iOS devices, maximizing accessibility for users across different platforms.

Key Challenges:

i.Content Aggregation:

Integrating APIs from various news sources and handling diverse data formats pose challenges in efficiently aggregating and organizing news articles within the app.

ii. Data Management:

Managing a large volume of news articles, user preferences, and engagement data requires robust data management and storage solutions to ensure scalability and performance.

I. Personalization Algorithms:

Developing accurate and effective algorithms for analyzing user preferences and behavior to deliver personalized news feeds involves complex data processing and machine learning techniques.

II. Credibility Assessment:

Verifying the credibility and reliability of news sources and articles presents challenges in distinguishing between trustworthy and misleading content, requiring rigorous fact-checking mechanisms.

III. Performance Optimization:

Optimizing app performance and loading times, especially when dealing with large datasets and real-time updates, requires efficient resource management and caching strategies.

IV. User Engagement:

Encouraging active participation and engagement with news content poses challenges in designing compelling features and fostering community interaction within the app.

V. Platform-specific Considerations:

Addressing platform-specific differences and guidelines, such as UI/UX design patterns and performance optimizations, while ensuring a consistent user experience across different devices and platforms.

VI. Monetization Strategy:

Identifying and implementing sustainable monetization strategies, such as advertisements or subscriptions, while balancing revenue generation with user experience and engagement.

VII. Security and Privacy:

Ensuring the security and privacy of user data, especially sensitive information such

as browsing history and preferences, requires robust encryption and compliance with data protection regulations.

VIII. Feedback Integration:

Effectively collecting and incorporating user feedback to continuously iterate and improve the app, addressing user needs and preferences while maintaining stability and reliability.

CHAPTER 2

METHODOLOGY

2.1. INITIAL PROJECT REQUIREMENTS

The initial project requirements for Daily News App encompass various aspects such as functionalities, technologies, and packages.

i) Project Scope:

- 1) Develop a mobile application named "Daily News App".
- 2) News Feed: Displaying curated articles from various sources.
- 3) Save and Bookmark: Allowing users to save articles for later reading.

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.
- 3) Firebase: Integrate Firebase for backend services, including authentication, real-time database (Firestore), and cloud storage.

iii) **Functionalities:**

- 1) Display a list of curated news articles from various sources.
- 2) Organize articles by categories such as Top Stories, Technology, Sports, etc.
- 3) Implement infinite scrolling for loading additional articles as the user scrolls.
- 4) Implement a secure user authentication system for both mentors and students.
- 5) Present detailed information about each article, including headline, publication date, author, and content.
- 6) Include multimedia content such as images, videos, and embedded media.
- 7) Enable users to save articles for later reading or offline access.
- 8) Implement bookmarking functionality to bookmark favorite articles for quick access.

iv) **Packages Used:**

i. get:

Version: 4.6.6

Purpose: State management and navigation framework.

ii. **GetIt:**

GetIt is a simple service locator for Dart and Flutter projects, used for dependency injection. It allows for the registration and retrieval of dependencies throughout the application.

iii. **Dio:**

Dio is an HTTP client for Dart and Flutter, used for making HTTP requests to fetch data from remote APIs. It simplifies the process of making network requests and handling responses.

iv. **Floor:**

Floor is an SQLite database library for Flutter and Dart, providing tools for working with local databases. It offers functionalities such as database creation, schema management, and data access through Object-Relational Mapping (ORM) techniques.

I. PACKAGES USED EXPLANATION

i.dio:

This is a powerful HTTP client for Dart that allows for making network requests. It simplifies the process of sending HTTP requests and handling responses, making it a popular choice for networking tasks in Dart and Flutter applications.

ii.get_it:

This is a simple service locator for Dart and Flutter projects. It provides a way to locate and obtain dependencies from anywhere in the application without needing to pass them through constructors manually. It's particularly useful for managing global state and accessing services across different parts of the application.

iii.floor:

This is a library for working with SQLite databases in Flutter and Dart. It provides an abstraction layer over SQLite, allowing developers to define database entities and access them using DAOs (Data Access Objects). It simplifies database operations and integrates seamlessly with Flutter applications.

iv.news_api_service:

This likely represents a custom service or client for interacting with a news API. It could include methods for fetching news articles, updating article data, or performing other operations related to news content. This package might encapsulate API endpoints and authentication logic specific to the chosen news API.

v.bloc:

This package is commonly used for implementing the BLoC (Business Logic

Component) architectural pattern in Flutter applications. It helps in separating business logic from presentation, making applications more maintainable and testable. In the provided code, there are references to RemoteArticlesBloc and LocalArticlesBloc, suggesting the usage of BLoCs for managing remote and local article-related logic respectively.

Overall, these packages contribute to building a structured and maintainable Flutter application with features related to fetching, storing, and presenting daily news articles.

2.2.PROPOSED APPLICATION ARCHITECTURE

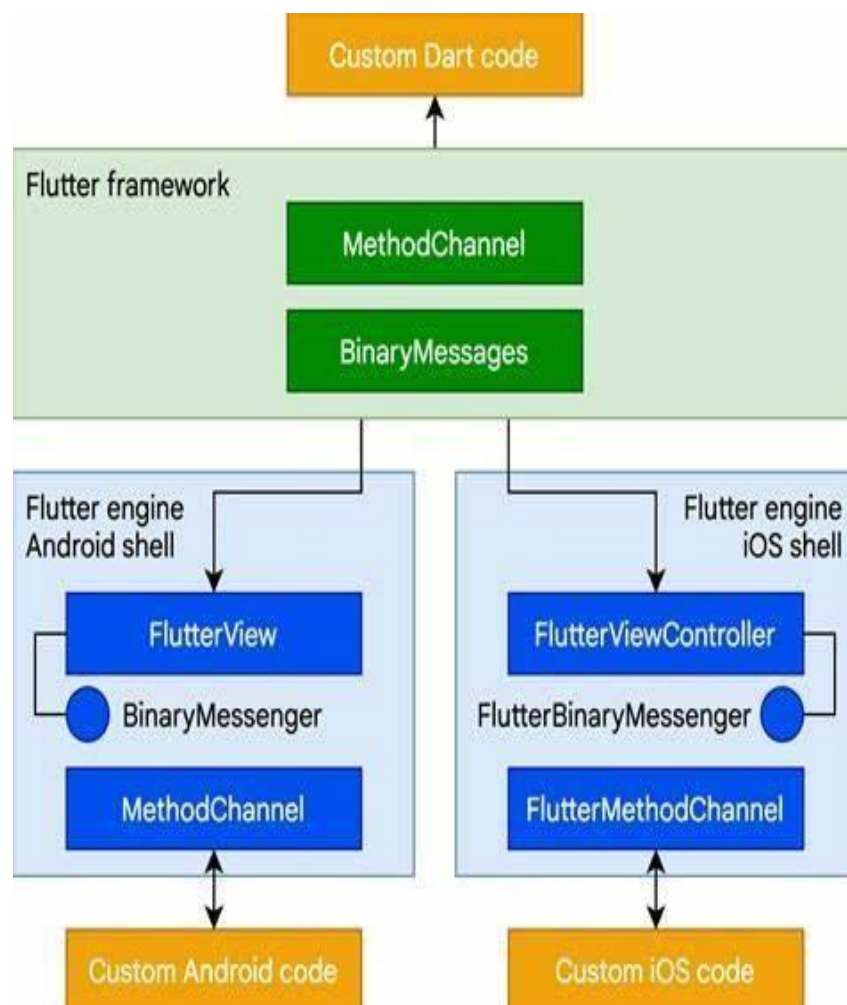


Fig:2.1 Proposed Application Architecture

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. API:



Fig:2.2 API

API

I. Introduction:

- Provide an overview of the purpose and scope of the API report, highlighting the importance of APIs in software development and their role in enabling communication and data exchange between different systems.

II. API Overview:

- Define what an API (Application Programming Interface) is and explain its significance in facilitating interaction between software components, services, and platforms.
- Describe the types of APIs commonly used, including RESTful APIs, SOAP APIs, GraphQL APIs, etc.
- Highlight the benefits of using APIs, such as improved interoperability, scalability, and flexibility in application development.

III. API Design:

- Discuss the key principles and best practices for designing APIs, including:
 - Endpoint Design: Define clear and intuitive endpoint URLs and HTTP methods for accessing resources.
 - Data Formats: Specify the use of standardized data formats such as JSON or XML for request and response payloads.
 - Error Handling: Define consistent error codes and messages for handling errors and exceptions.
 - Security Considerations: Address security concerns such as authentication, authorization, and data encryption to protect API endpoints from unauthorized access and data breaches.

IV. API Documentation:

- Emphasize the importance of comprehensive API documentation in facilitating developer onboarding and integration.
- Provide guidelines for documenting APIs, including:
 - Endpoint Documentation: Document each API endpoint with details on its purpose, parameters, request/response format, and examples.
 - Authentication and Authorization: Explain the authentication methods and access controls required to use the API securely.
 - Code Samples: Include code snippets and examples in multiple programming languages to demonstrate how to use the API.
 - API Reference: Generate an API reference documentation using tools like Swagger or OpenAPI Specification (OAS) to automate the process and ensure consistency.

V. API Testing:

- Discuss the importance of testing APIs to ensure reliability, performance, and compatibility.
- Describe the types of API tests, including unit tests, integration tests, and end-to-end tests.
- Outline strategies for API testing, such as using mocking frameworks, API testing tools, and continuous integration pipelines.
- Provide guidelines for writing effective API tests and validating API responses against expected outcomes.

VI. API Security:

- Highlight the security considerations and best practices for securing APIs against common threats such as SQL injection, cross-site scripting (XSS), and unauthorized

access.

- Discuss authentication mechanisms such as OAuth 2.0, JWT (JSON Web Tokens), and API keys for verifying the identity of clients and ensuring secure access to API endpoints.
- Address authorization controls for restricting access to sensitive resources based on user roles and permissions.
- Recommend encryption and data masking techniques for protecting sensitive data transmitted over the API.

VII. API Management:

- Discuss the importance of API management platforms for centralizing API governance, monitoring, and analytics.
- Describe the features and capabilities of API management tools, including API gateway, API lifecycle management, rate limiting, and traffic management.
- Explain how API management platforms facilitate API discovery, versioning, and monetization to support business objectives and revenue generation.

VIII. Conclusion:

- Summarize the key findings and recommendations from the API report, emphasizing the critical role of APIs in modern software development and the importance of following best practices for designing, documenting, testing, and securing APIs.
- Provide actionable insights for improving API design, implementation, and management to enhance interoperability, security, and developer experience.

2.3.HOW NEWS APP REQUEST FROM THE API SERVICE:

In a Flutter app, requesting data from an API service involves several steps, typically implemented using an HTTP client library like `http` or `Dio`. Here's how this process works in a Flutter news app:

Import HTTP Client Library:

First, you need to import the HTTP client library in your Flutter project. This can be done by adding the relevant package (`http` or `dio`) to your `pubspec.yaml` file and running `flutter pub get` to install the dependencies.

Forming the Request:

When a user interacts with the app, such as opening the news feed screen, the app needs to send a request to the API service to fetch news articles.

You create an HTTP request object that includes details such as the endpoint URL, HTTP method (GET, POST, etc.), query parameters, headers, and any request body if required.

Sending the Request:

Once the request is formed, you use the HTTP client library to send the request to the API service.

If you're using the `http` package, you use the `http.get`, `http.post`, `http.put`, or `http.delete` methods to send requests. If you're using the `Dio` package, you use the `dio.get`, `dio.post`, `dio.put`, or `dio.delete` methods.

Handling the Response:

After sending the request, the API service processes the request and sends back a response.

You await the response using asynchronous programming, as network requests are

inherently asynchronous operations.

Once the response is received, you extract the relevant data from the response object. This could be JSON data, XML data, or any other format supported by the API.

If the response indicates success (e.g., HTTP status code 200), you parse the data and update the app's UI accordingly. If there's an error, you handle it appropriately, such as displaying an error message to the user.

Updating the UI:

After parsing the response data, you update the app's UI to display the fetched news articles.

This typically involves updating state variables within Flutter widgets using `setState`, `Provider`, or other state management solutions.

You may display the news articles in a list or grid view, along with additional information such as headlines, publication dates, and article summaries.

Error Handling:

It's essential to implement error handling to deal with potential issues during the request-response cycle.

This includes handling network errors (e.g., timeouts, no internet connection), server errors (e.g., HTTP status codes 4xx or 5xx), and parsing errors (e.g., invalid JSON format).

You can display error messages to the user or implement retry mechanisms to resend failed requests.

Optimization:

To improve performance and reduce network usage, you can implement caching mechanisms to store fetched data locally on the device.

Additionally, you can optimize API requests by batching multiple requests, implementing pagination, or using compression techniques to minimize data transfer.

By following these steps, your Flutter news app can effectively communicate with the API service, fetch news articles, and provide users with an up-to-date and engaging news browsing experience.

2.4.INTEGRATE API WITH FLUTTER

```
import 'package:http/http.dart' as http;
import 'dart:convert';

Future<void> fetchData() async {
  final response = await http.get(Uri.parse('https://api.example.com/data'));

  if (response.statusCode == 200) {
    // If the server returns a 200 OK response, parse the JSON data
    final jsonData = json.decode(response.body);

    // Process the data and update UI accordingly
    print(jsonData);
  } else {
    // If the server returns an error response, throw an exception
    throw Exception('Failed to fetch data');
  }
}
```

Fig:2.3 INTEGRATE API WITH FLUTTER

To integrate an API with Flutter, you can follow these general steps:

Choose an API:

Identify the API you want to integrate into your Flutter app. This could be a public API provided by a service like news articles, weather data, social media platforms, etc. Ensure that you have access to the API documentation and any required authentication credentials.

HTTP Client:

Use an HTTP client in Flutter to make requests to the API and handle responses. The

most commonly used HTTP client in Flutter is the `http` package, but you can also use other packages like `Dio` for more advanced features.

Make API Requests:

Use the chosen HTTP client to make GET, POST, PUT, DELETE, etc., requests to the API endpoints. Construct the request URLs with any required query parameters or request bodies as specified in the API documentation.

Handle Responses:

Handle the responses returned by the API requests. Parse the response data, handle any errors or status codes, and extract the relevant information needed for your app.

Update UI:

Update your Flutter app's UI based on the data retrieved from the API. Display the fetched data in lists, grids, cards, or any other UI components as per your app's design and requirements.

Error Handling:

Implement error handling mechanisms to handle network errors, timeouts, server errors, etc. Display appropriate error messages to the user and provide fallback options when necessary.

Optimization:

Optimize API requests by implementing techniques like caching, pagination, and lazy loading to improve app performance and reduce network usage.

Here's a basic example demonstrating how to make a GET request to fetch data from an API using the `http` package in Flutter:

2.5.MODULES

Algorithm

The provided code initializes dependencies using the GetIt service locator in a Flutter/Dart project. Here's an algorithmic breakdown of what the code does:

Initialize GetIt Service Locator:

Import the GetIt package and create a singleton instance of the GetIt class named `sl`.

Initialize Database:

Use the Floor library to initialize the local database named `AppDatabase` by calling the `databaseBuilder` method.

Register the `AppDatabase` instance as a singleton dependency in the service locator (`sl`).

Initialize Dio HTTP Client:

Register the Dio HTTP client as a singleton dependency in the service locator (`sl`).

Initialize News API Service:

Instantiate the `NewsApiService` class, passing the Dio HTTP client (`sl()`) as a dependency.

Register the News Api Service instance as a singleton dependency in the service locator (`sl`).

Initialize Article Repository:

Instantiate the `ArticleRepositoryImpl` class, passing the database (`sl()`) and news API service (`sl()`) as dependencies.

Register the `ArticleRepository` instance as a singleton dependency in the service locator (`sl`).

Initialize Use Cases:

Instantiate the various use case classes (`GetArticleUseCase`, `GetSavedArticleUseCase`, `SaveArticleUseCase`, `RemoveArticleUseCase`), passing

the article repository (sl()) as a dependency for each use case.

Register each use case instance as a singleton dependency in the service locator

Initialize Blocs:

Instantiate the RemoteArticlesBloc and LocalArticlesBloc classes, passing the necessary dependencies (such as the news API service, article repository, and use cases) to each bloc.

Register the bloc instances as factory dependencies in the service locator (sl), meaning a new instance will be created each time they are requested.

Initialization Function:

Define a function named initializeDependencies to encapsulate the initialization process.

The function is marked as async to handle asynchronous database initialization.

This function can be called at the application startup to initialize all dependencies before the app's main logic executes.

Overall, the code sets up dependencies for data sources, repository implementations, use cases, and blocs using the GetIt service locator, facilitating dependency injection and separation of concerns in the Flutter/Dart project.

2.5.1. PROFILE MANAGEMENT:

To implement profile management in your Flutter app using the provided dependencies and architecture, you'll need to follow these steps:

ALGORITHM:

Create User Model:

Define a User model class to represent user data such as name, email, profile picture, etc.

Profile Screen:

Create a profile screen where users can view and edit their profile information.

Display user details fetched from the database or authentication provider.

Provide options for users to update their profile picture, change their name, or modify other profile details.

Update User Repository:

Modify the `ArticleRepository` interface or implementation to include methods for managing user profile data.

Implement methods such as `getUserProfile`, `updateUserProfile`, `updateProfilePicture`, etc., in the repository.

Update Use Cases:

Modify the existing use cases or create new use cases for profile management operations.

Implement use case classes like `GetUserProfileUseCase`, `UpdateUserProfileUseCase`, `UpdateProfilePictureUseCase`, etc., to handle profile-related business logic.

Update Blocs:

Modify the existing blocs or create new blocs to handle profile-related events and states.

Implement bloc classes like `ProfileBloc` to coordinate profile management operations and update the UI accordingly.

Update UI:

Update the profile screen UI to integrate with the blocs and display user profile data.

Implement form fields and input widgets for users to edit their profile information.

Handle user interactions such as tapping buttons to update profile data or upload profile pictures.

Integrate with Database and API:

Update the database and API service to support profile management operations.

Implement database queries and API endpoints for fetching and updating user profile data.

Handle Authentication and Authorization:

Ensure that only authenticated users can access the profile management features.
Implement authentication checks to prevent unauthorized access to profile data.

Test Profile Management:

Test profile management features thoroughly to ensure they work as expected under different scenarios.

Test UI interactions, data validation, error handling, and integration with backend services.

Handle Edge Cases:

Consider edge cases such as network errors, invalid input, and unexpected server responses.

Implement error handling mechanisms to provide informative error messages to users and handle errors gracefully.

By following this algorithm and integrating profile management features into your Flutter app, you can provide users with a seamless experience for managing their profile information within the app

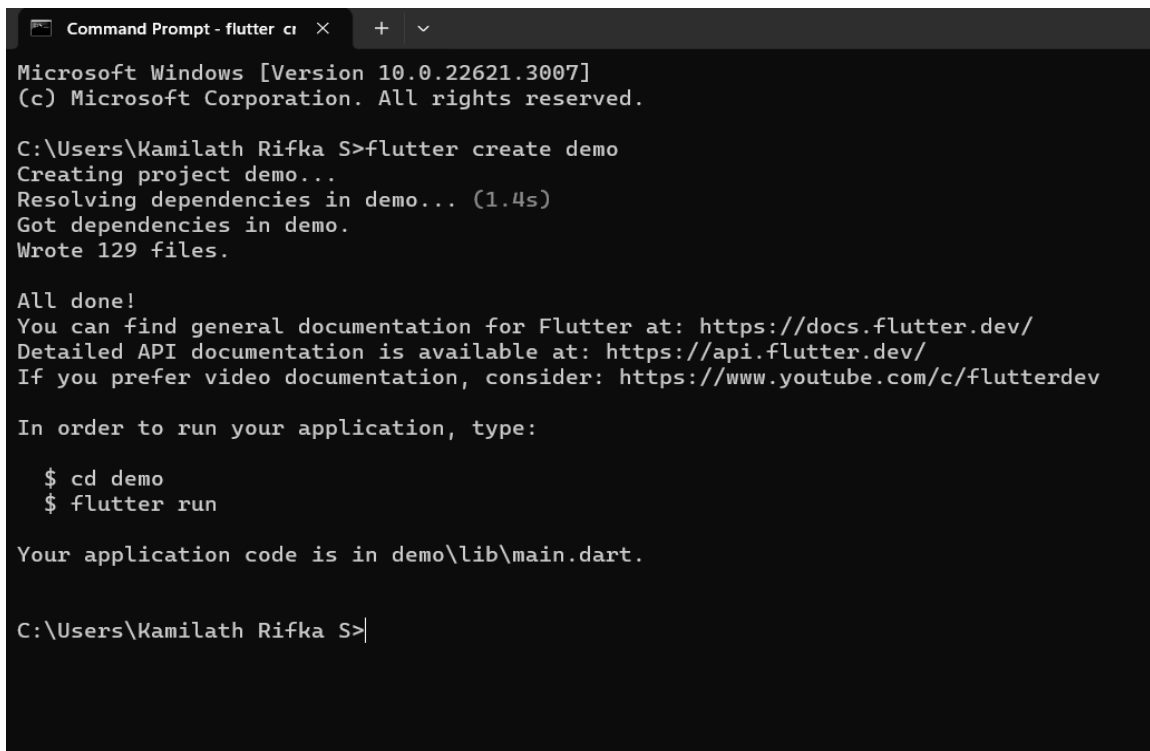
2.6.DEVELOPMENT AND DEPLOYMENT SETUP

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

A screenshot of a Windows Command Prompt window titled "Command Prompt - flutter c1". The window shows the output of the command "flutter create demo". The text in the prompt is as follows:

```
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kamilath Rifka S>flutter create demo
Creating project demo...
Resolving dependencies in demo... (1.4s)
Got dependencies in demo.
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 demo
$ flutter run

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

C:\Users\Kamilath Rifka S>
```

Fig:2.4 Project Creation

3. Dependency Management:

Dio Package (package:dio/dio.dart):

Dio is an HTTP client for Dart that simplifies making HTTP requests and handling responses. It's used for sending requests to the remote API service to fetch news articles.

GetIt Package (package:get_it/get_it.dart):

GetIt is a simple service locator for Dart and Flutter projects. It's used for dependency injection, allowing objects to be registered and retrieved from a central location (the service locator) throughout the app.

Floor Database (AppDatabase from new_app_clean_arch/features/daily_news/data/data_sources/local/app_database.dart):

Floor is a library that simplifies SQLite database operations in Flutter and Dart. The AppDatabase class is the local database used for storing articles and other local data within the app.

News API Service (NewsApiService from new_app_clean_arch/features/daily_news/data/data_sources/remote/news_api_service.dart):

The NewsApiService class is a wrapper around the remote API service for fetching news articles. It's responsible for making HTTP requests to the API and handling responses.

Article Repository (ArticleRepository from new_app_clean_arch/features/daily_news/data/repository/article_repository.dart):

The ArticleRepository interface defines methods for interacting with article data, such as fetching articles from the local database or the remote API. The ArticleRepositoryImpl class implements this interface.

Use Cases:

Use cases are classes that encapsulate the business logic of specific operations within the app. In this code, use cases like GetArticleUseCase, GetSavedArticleUseCase, SaveArticleUseCase, and RemoveArticleUseCase are used to perform operations related to fetching, saving, and removing articles.

Blocs:

Blocs (Business Logic Components) are classes that manage the state of specific parts of the app and handle events and data transformations. In this code, RemoteArticlesBloc and LocalArticlesBloc are used for managing the state of remote and local articles, respectively.

These dependencies are registered and initialized in the initializeDependencies function using the GetIt service locator (sl). This function sets up the dependency injection framework for the app, allowing these dependencies to be easily accessed and used throughout the app.

4. Development Tools:

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

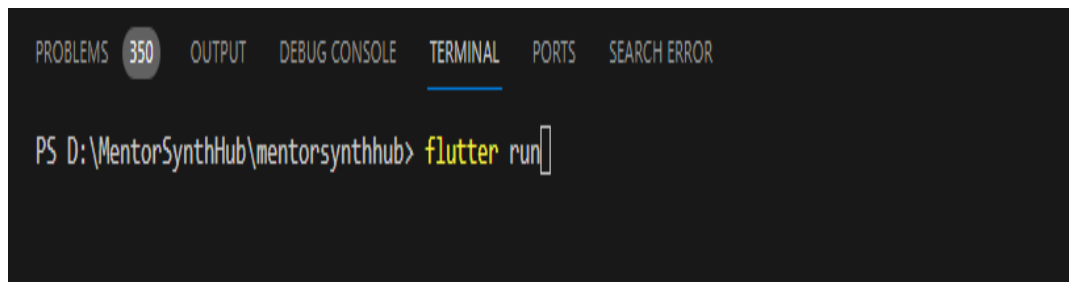


Fig:2.5 Flutter Run

5.Deployment Process:

API used for hosting the web version of the application. Seamless deployment using Flutter and API commands.

6.Supported Platforms:

Cross-platform support for Android, iOS, and web.

7.Integrated Development Environment (IDE):

Developers can choose between Visual Studio Code or AndroidStudio for coding.

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) `Injection_container.dart` : This file typically contains code responsible for setting up and configuring the dependency injection framework.

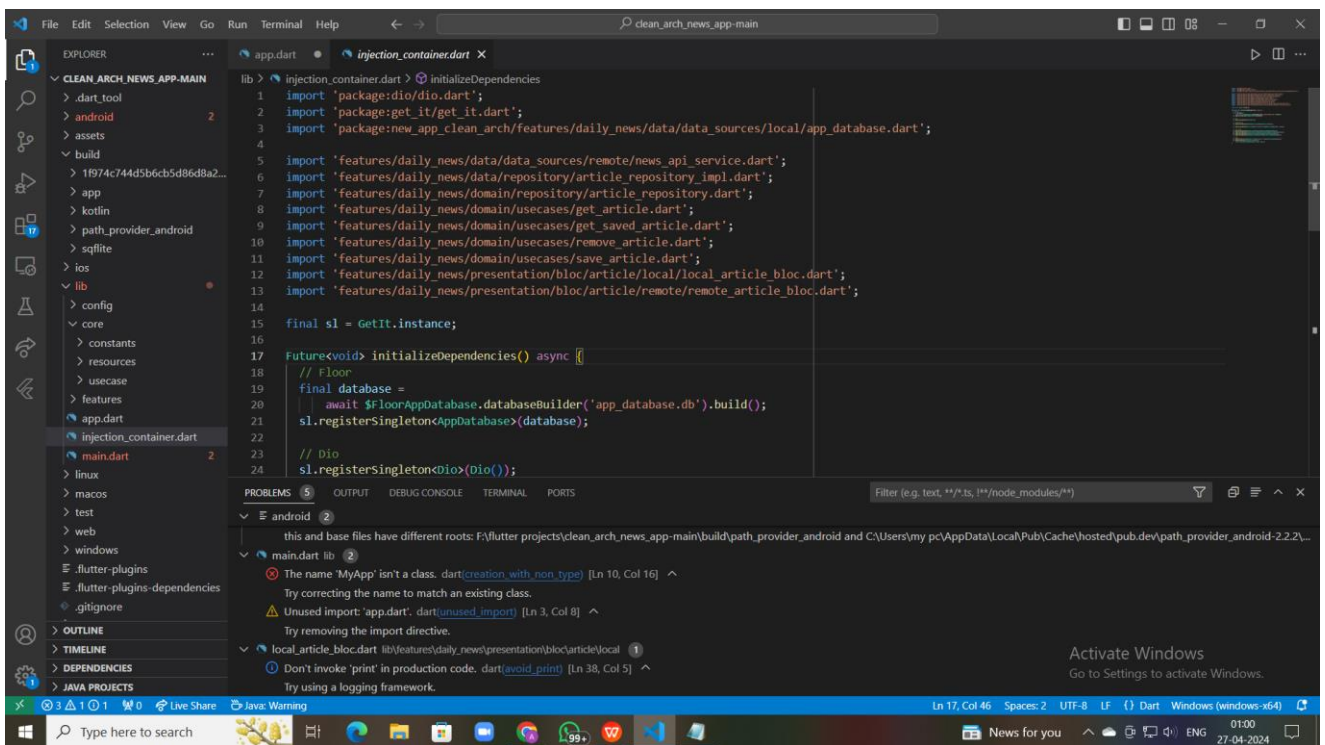


Fig:3.1 injection_container.dart

iii) `homePage.dart`: This file represents the main content of your app after a successful login or signup. It contains `messages`, `profilePage`, `feedbackPage` and `mentorSelection Page` with various features.

iv) The file is likely responsible for displaying and editing user profiles. It includes information about the user, such as their name, profile picture, and additional details.

v) `messagePage.dart`:

This file could represent a page where users can exchange messages.

vi) `message_model.dart`: This file defines the structure of a `Message` model.

vii) `messageForm.dart`: This file represents a form for users to send messages.

ABOUT:

A daily news app developed using Flutter offers users a convenient way to stay informed about current events, breaking news, and topics of interest. Here's an overview of what a typical daily news app developed using Flutter might include:

User Interface:

The app features a clean and intuitive user interface designed to provide easy navigation and an enjoyable reading experience.

It may include screens such as:

News Feed: A feed displaying a list of latest news articles sorted by date or relevance.

Categories: Options for users to browse news articles by categories such as politics, sports, technology, entertainment, etc.

Search: A search functionality allowing users to search for specific news articles based on keywords or phrases.

Article Details: Detailed view of individual news articles, including headlines, images, publication dates, and full article content.

Content Management:

The app fetches news articles from one or more external sources via APIs.

It may integrate with popular news APIs such as News API, RSS feeds, or custom APIs provided by news organizations.

Articles are fetched asynchronously and displayed in real-time, ensuring that users have access to the latest news updates.

Features and Functionality:

Bookmarking: Users can save articles they find interesting for later reading.

Sharing: Built-in sharing functionality allows users to share articles with friends and on social media platforms.

Offline Reading: Articles can be cached locally for offline reading, ensuring users can access content even without an internet connection.

Push Notifications: Optionally, the app can send push notifications to users to alert them about breaking news or important updates.

Settings: Users may have the option to customize app settings such as preferred news sources, notification preferences, font size, etc.



FCC Reinstates Net Neutrality In A Blow To Internet Service Provi...

Major internet providers once again will have to

✓ 2024-04-25T16:16:00Z



Analysis | More details in the national conspiracy to overtur...

Indictments in Arizona and new information in Michigan

✓ 2024-04-25T15:49:42Z



2024 NFL Mock Draft: J.J. McCarthy to Vikings, Xavier Worth...

The final mock before the real thing

✓ 2024-04-25T15:20:00Z



Rubrik: Microsoft-backed startup prices IPO at \$32 per share t...

Rubrik's offering had received about 20 times as

✓ 2024-04-25T15:19:15Z



University protests spread across the U.S.

Fig:3.2 HOME PAGE

i) 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.

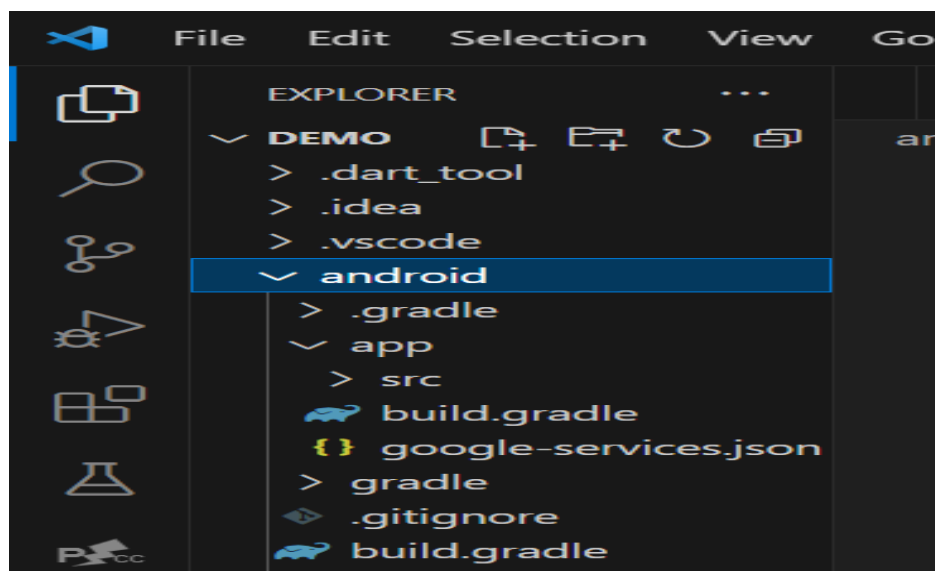
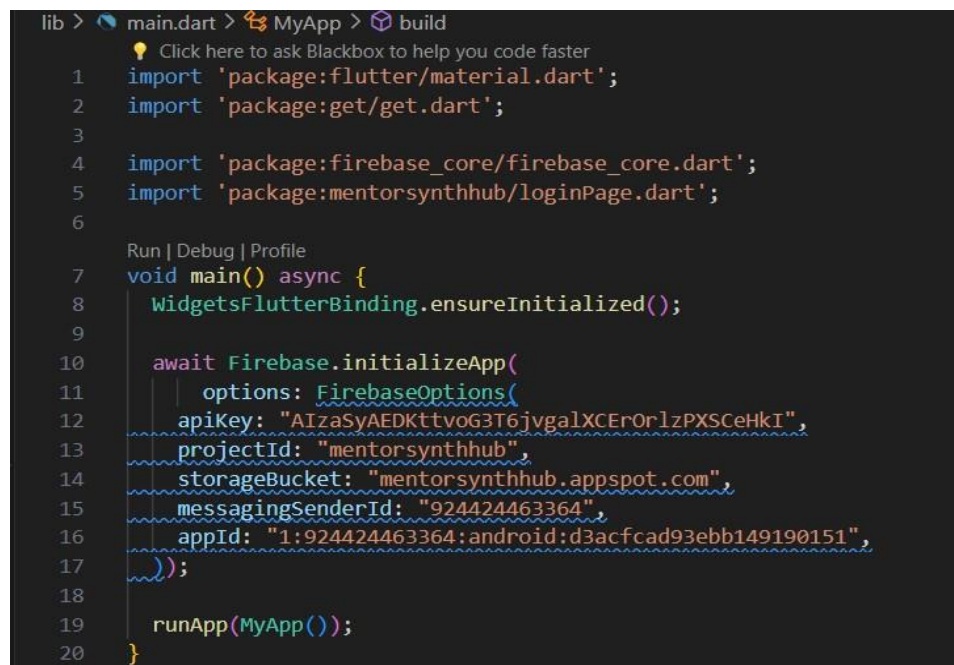


Fig:3.3 Configurations

ii) Initialize Firebase:

In your main.dart or wherever you initialize your Flutter app, make sure to 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.

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    WidgetsFlutterBinding.ensureInitialized();
10
11    await Firebase.initializeApp(
12      options: FirebaseOptions(
13        apiKey: "AIzaSyAEDKttvoG3T6jvgalXCeOrlzPXSCeHkI",
14        projectId: "mentorsynthhub",
15        storageBucket: "mentorsynthhub.appspot.com",
16        messagingSenderId: "924424463364",
17        appId: "1:924424463364:android:d3acfcad93ebb149190151",
18      ));
19    runApp(MyApp());
20  }
```

Fig:3.4 Initializing Firebase

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1. CONCLUSION

In conclusion, developing a daily news app using Flutter and Dart offers a powerful solution for delivering timely and relevant news content to users in a fast, intuitive, and engaging manner. By leveraging the capabilities of Flutter, developers can create cross-platform applications with a native look and feel, ensuring a seamless user experience across different devices. With features such as a customizable news feed, category browsing, search functionality, and article bookmarking, users can easily access the latest news articles tailored to their interests. The integration of push notifications further enhances user engagement by providing timely updates on breaking news and important developments.

Additionally, the app's performance is optimized through efficient data fetching, caching mechanisms, and asynchronous operations, ensuring smooth navigation and fast loading times. Security measures are implemented to safeguard user data and privacy, enhancing trust and confidence in the app. Overall, a daily news app developed using Flutter and Dart not only meets the demands of modern news consumption but also showcases the capabilities of Flutter as a versatile framework for building high-quality mobile applications. With its rich feature set, intuitive user interface, and robust performance, such an app serves as an indispensable tool for keeping users informed and engaged in today's fast-paced world.

4.2. FUTURE WORK

Implement advanced personalization algorithms to tailor the news feed to each user's preferences and browsing history.

Offer customization options for users to specify their preferred topics, sources, and types of news content. Improve the search functionality by integrating natural language processing (NLP) techniques for more accurate and relevant search results. Implement advanced filtering options such as date range, location-based filtering, and content type filtering. Introduce interactive elements such as polls, quizzes, and user comments to increase user engagement and foster community interaction within the app. Enable social features like user profiles, following other users, and sharing favorite articles with friends. Integrate support for multimedia content such as audio news briefings, video clips, and live streams to provide users with diverse media formats for consuming news. Ensure the app complies with accessibility standards and guidelines to make it usable for all users, including those with disabilities. Implement features such as screen reader support, voice commands, and adjustable text sizes for improved accessibility. Expand language support and offer localized content in different languages to cater to a more diverse audience. Explore monetization opportunities such as in-app advertisements, sponsored content, premium subscriptions, or affiliate marketing partnerships.

Implement analytics tools to track user engagement and behavior for optimizing monetization strategies. Enhance offline functionality by implementing features such as offline reading, offline search, and offline content caching for seamless access to news articles without an internet connection. Explore integration with emerging technologies such as augmented reality (AR) and virtual reality (VR) to create immersive news experiences and storytelling formats. Experiment with blockchain technology for secure and transparent content distribution and authentication.

REFERENCES

- [1] Arendt, F., & Karadas, N. (2017). Content analysis of mediated associations: “An automated text-analytic approach”. *Communication Methods and Measures*, 11(2), 105-120.
- [2] Chenggang Zhen, Peng Cheng. “Construction of campus trading platform based on third-party online payment” 2nd International Conference on industrial and Information Systems, IEEE,2010
- [3] Dashtestani, R. (2016). Moving bravely towards mobile learning: “Iranian students” use of mobile devices for learning English as a foreign language. *Computer Assisted Language Learning*, 29(4), 815–832.
- [4] ESL students “perceptions of mobile applications for discipline-specific vocabulary acquisition for academic purposes” , volume 13 , p. 102 - 117
Posted: 2021
- [5] Ganapathy, M., Shuib, M., & Azizan, S. N. (2016). Malaysian ESL students “perceptions on the usability of a mobile application of grammar test”: A case study of ESL undergraduates in University Sains Malaysia. 3L: The Southeast Asian Journal of English Language Studies, 22(1), 127–140.
- [6] Haizheng Li and Han Zhang, “ How People Select Their Payment Methods in Online Auctions? An Exploration of eBay Transactions”, *Proceedings of the 37th Hawaii International Conference on System Sciences* – 2004
- [7] *Journal for Research Scholars and Professionals of English Language Teaching* , volume 4 , issue 22 , p. 1 - 5
Posted: 2020
- [8] *Language Teaching Research Quarterly* , volume 20 , p. 1 - 18
Posted: 2020
- [9] Mandasari, B. (2020). “The Impact of Online Learning on Students” *Academic Performance on Business Correspondence Course. EDUTEC: Journal of Education and Technology*, 4(1),

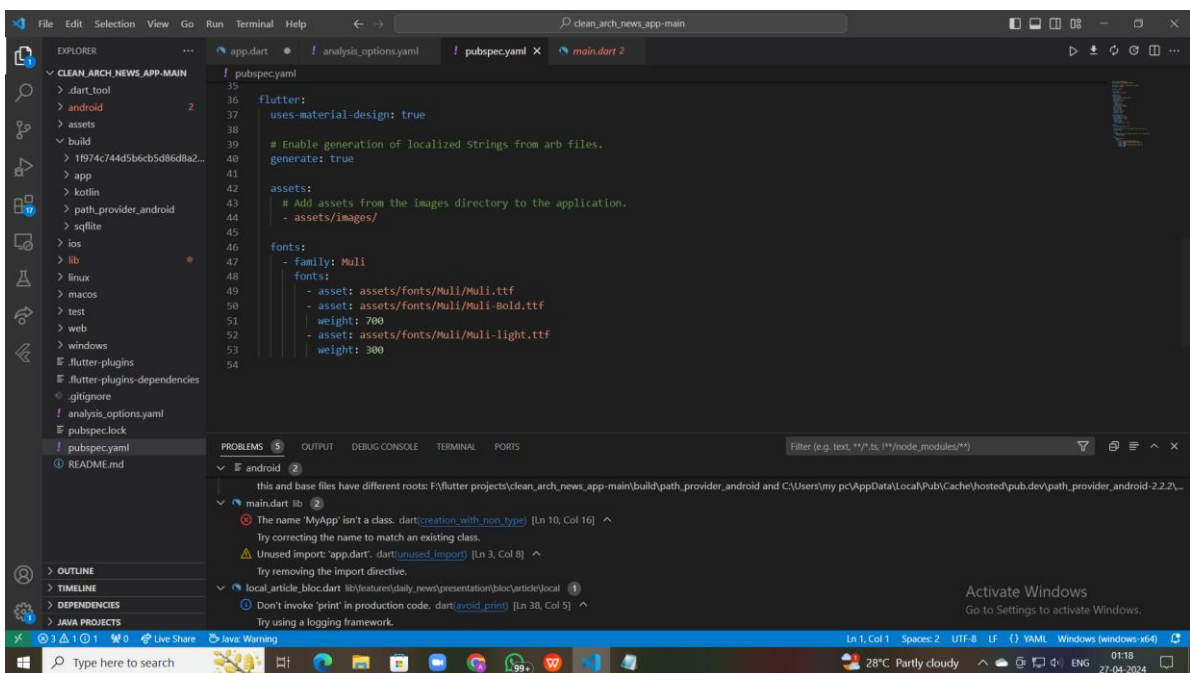
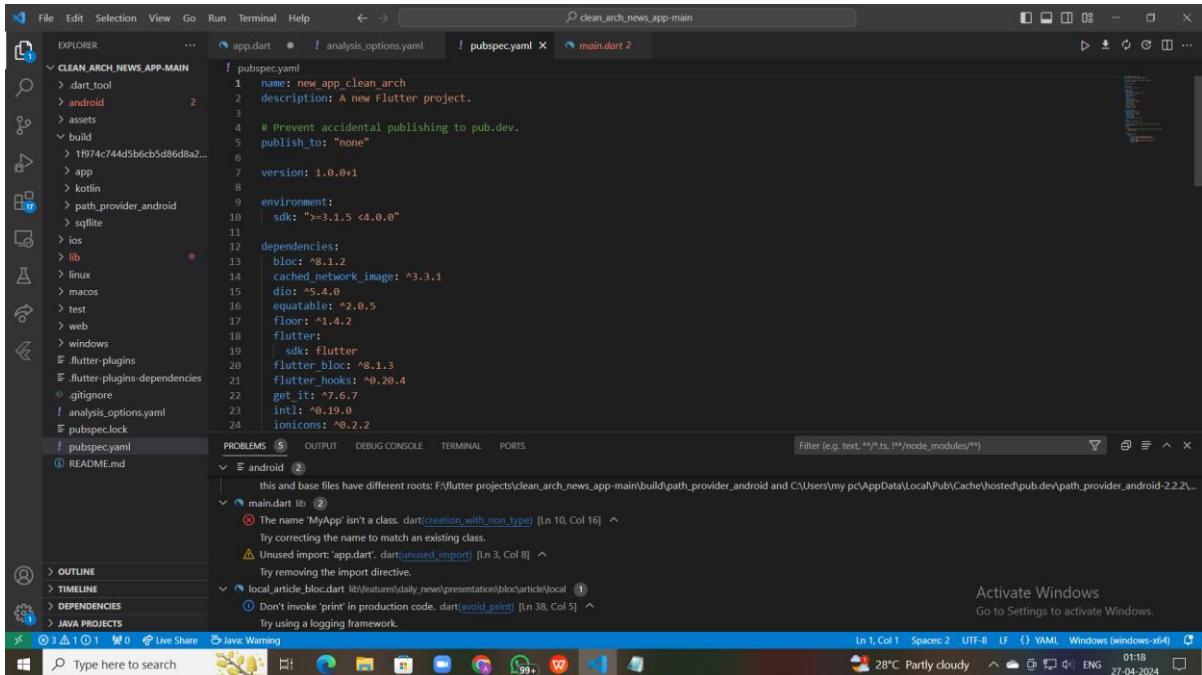
APPENDICES

A.SAMPLE CODE

i)Main 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
Run | Debug | Profile
7  void main() async {
8    WidgetsFlutterBinding.ensureInitialized();
9
10   await Firebase.initializeApp(
11     options: FlutterOptions(
12       apiKey: "AIzaSyAEDKttvoG3T6jvga1XCErOrlzPXScEhKI",
13       projectId: "mentorsynthhub",
14       storageBucket: "mentorsynthhub.appspot.com",
15       messagingSenderId: "924424463364",
16       appId: "1:924424463364:android:d3acfcad93ebb149190151",
17     );
18
19   runApp(MyApp());
20 }
```

ii) pubspec.yaml



B.SCREEN SHOTS

i)home Page



ii) Saved Article

00:41

66%



Analysis | More details in the national conspiracy to overturn the 2020 election - The Washington Post

🕒 2024-04-25T15:49:42Z



Indictments in Arizona and new information in Michigan shows the scale of the effort to keep Donald Trump in power.

Last month, the overhauled leadership of the Republican Party took a step aimed at keeping what Donald Trump insists occurred in 2020 that the election was somehow stolen by fraud or subversion, whic... [+4635 chars]



Article saved successfully!

iii) Server loading

