

SKILL SWAP

INTERDISCIPLINARY PROJECT

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Technology Degree in Information Technology

by

JUDITH SHALOM JOY J (41120082)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category-1 University by UGC

Accredited with Grade “A++” by NAAC

12B Status by UGC |Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI- 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 **JUDITH SHALOM JOY J (Reg No.:41120082)** who carried out the project entitled "**FLUTTER DEVELOPMENT**" under my supervision from January 2024 to May 2024.


Internal Guide

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



Head of the Department

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

Submitted for Viva voce Examination held on 08.05.2024


Internal Examiner


External Examiner

DECLARATION

I JUDITH SHALOM JOY J (Reg.No.41120082) hereby declare that the Project Report entitled "FLUTTER DEVELOPMENT" is done by me under the guidance of Dr.C.Geetha, M.Tech., Ph.D. and is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 08.05.2024

PLACE: Chennai

A handwritten signature in blue ink, appearing to read 'Judith', with a stylized flourish extending from the end.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

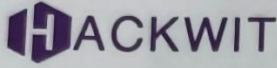
I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

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

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.C.Geetha , M.Tech., Ph.D.** for her valuable guidance, suggestions and constant encouragement which paved the 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 FOR INTERNSHIP:

 HACKWIT HACKWIT TECHNOLOGIES PVT LTD Redefine by Technologies	149, 1C, 2nd Floor, Mount Poonamallee Rd, Ramapuram, Chennai - 600089. admin@hackwittechnologies.com www.hackwittechnologies.com
---	---

10.02.2024

CERTIFICATE FOR INTERNSHIP

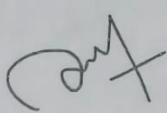
TO WHOMSOEVER IT MAY CONCERN

This is to Certify that **Judith Shalom Joy J (Reg No: 41120082)** 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, **Judith Shalom Joy J** has closely worked as a part of the **Flutter** by carrying out the project entitled **"Skill Swap"**.

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



COURSE COMPLETION CERTIFICATE:



ABSTRACT

Flutter, a cross-platform UI toolkit developed by Google, has gained significant traction in the realm of mobile application development. This abstract presents a comprehensive overview of Flutter development, covering its architecture, key features, advantages, and applications.

Flutter utilizes the Dart programming language and a reactive framework to create visually appealing, fast, and fluid user interfaces for Android, iOS, web, and desktop platforms from a single codebase. Its widget-based architecture enables developers to build highly customizable UIs with ease, facilitating rapid prototyping and iteration.

One of the main advantages of Flutter is its hot reload feature, which allows developers to instantly view changes in the application during development without restarting the entire application, thus accelerating the development process and enhancing productivity.

Moreover, Flutter's extensive widget library provides developers with a rich set of pre-designed UI components, enabling them to create beautiful and consistent user experiences across different platforms. Additionally, Flutter offers native performance by compiling Dart code to native machine code, resulting in high-performance applications.

Furthermore, Flutter's strong community support, documentation, and growing ecosystem of packages and plugins contribute to its popularity among developers worldwide. From startups to tech giants, various companies have embraced Flutter for building innovative mobile applications across different domains such as e-commerce, education, healthcare, gaming, and more.

In conclusion, Flutter represents a compelling option for mobile application development, offering a combination of productivity, performance, and versatility. As the technology continues to evolve and mature, it is poised to play a significant role in shaping the future of cross-platform development.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	vii
	LIST OF FIGURES	x
1	INTRODUCTION	11
	1.1 Key Objectives	11
	1.2 System Overview	12
2	METHODOLOGY	15
	2.1 Existing System	15
	2.2 Proposed System	17
	2.3 Requirement Specification	17
	2.4 System Architecture	23
3	RESULTS AND DISCUSSION	28
	3.1 Results	28
	3.2 Output Screenshots	30

4	CONCLUSION AND	34
	FUTURE WORK	
4.1	Conclusion	34
4.2	Future Work	35
	REFERENCES	36

LIST OF FIGURES

Figure No.	Name of the Figure	Page No.
2.4.1	System Architecture	23
3.1.1.1	Home Page Code	28
3.1.2.1	Login Page Code	29
3.2.1.1	Login Page	30
3.2.2.1	Search Page	31
3.2.3.1	Home Page	32
3.2.4.1	Profile Page	33

CHAPTER 1

INTRODUCTION

INTRODUCTION:

In an era where smartphones have become ubiquitous companions in our daily lives, developing applications that seamlessly run across multiple platforms has become imperative. Flutter, an open-source UI software development kit created by Google, has emerged as a powerful solution for crafting beautiful, natively compiled applications for mobile, web, and desktop.

This comprehensive guide aims to provide developers, both novice and experienced, with a deep dive into Flutter development. Whether you're a seasoned mobile app developer looking to expand your skill set or a newcomer intrigued by the possibilities of cross-platform development, this resource will serve as your roadmap to mastering Flutter.

1.1.KEY OBJECTIVES

1.1.1.Understanding Flutter Fundamentals: We'll start by covering the basics of Flutter, including its architecture, widgets, and how it differs from traditional development frameworks. This foundation will ensure that you grasp the core concepts necessary for building robust applications.

1.1.2.Creating Stunning User Interfaces: One of Flutter's most compelling features is its ability to create beautiful, custom user interfaces with ease. We'll explore Flutter's rich set of widgets and layout options, along with best practices for designing visually appealing and responsive UIs.

1.1.3.Managing State Effectively: State management is a critical aspect of any application, and Flutter offers several approaches to handle state in your apps. We'll delve into various state management techniques, such as `setState()`, Provider, Bloc, and Riverpod, helping you choose the right solution for your project.

1.1.4.Navigating and Routing: Navigation plays a crucial role in user experience, especially in complex applications with multiple screens.

We'll demonstrate how to implement navigation and routing in Flutter, enabling users to seamlessly navigate between different sections of your app.

1.1.5.Accessing Device Features: Flutter provides plugins for accessing device features like camera, location, storage, and more. We'll explore how to leverage these plugins to incorporate native functionality into your Flutter apps, enhancing their capabilities and user experience.

1.1.6.Testing and Debugging: Ensuring the quality and reliability of your app is paramount. We'll cover techniques for writing unit tests, widget tests, and integration tests in Flutter, as well as debugging common issues and optimizing performance.

1.1.7.Deployment and Distribution: Finally, we'll walk you through the process of deploying your Flutter app to various platforms, including iOS, Android, web, and desktop. From configuring build settings to publishing your app on app stores and distributing it to users, we'll guide you every step of the way.

1.2.SYSTEM OVERVIEW

1.2.1. Define Features and Functionality:

1.2.1.1 User Registration and Profile Creation: Allow users to create accounts and set up profiles with information about their skills, interests, and what they want to learn.

1.2.1.2 Skill Listings: Users can list the skills they are willing to teach and the skills they want to learn.

1.2.1.3 Search and Matchmaking: Implement a search feature that helps users find potential skill swap partners based on their listed skills and preferences.

1.2.1.4 Chat and Messaging: Enable a messaging system for users to communicate and coordinate skill swap sessions.

1.2.2. Design User Interface:

Create wireframes and design the user interface. Consider a clean and intuitive design that facilitates easy navigation and interaction.

1.2.3. Set Up Backend:

Choose a backend solution to store user data, skill listings, and messages. Common choices include Firebase, Node.js with Express, or Django with Python.

Implement user authentication to secure user accounts.

1.2.4. Implement User Registration and Authentication:

Use Firebase Authentication or your chosen backend to implement user registration and login functionality.

1.2.5. Build User Profiles and Skill Listings:

Allow users to create and edit their profiles, specifying their skills and the skills they are looking to acquire.

1.2.6. Develop Search and Matching Algorithm:

Implement a search algorithm that matches users based on their listed skills. Consider additional parameters like location, availability, and user preferences.

1.2.7. Integrate Messaging System:

Implement a real-time messaging system to allow users to communicate and coordinate skill swap sessions.

1.2.8. Implement Rating and Feedback System:

Add a rating and feedback system to let users provide reviews after completing a skill swap session.

1.2.9. Incorporate Notifications:

Implement push notifications to alert users about new messages, skill swap proposals, or upcoming sessions.

1.2.10. Test the App:

Conduct thorough testing to identify and fix bugs. Test the app on various devices and simulate different usage scenarios.

1.2.11. Deploy and Monitor:

Deploy the app to the desired platforms (iOS, Android) and monitor its performance. Be ready to address any issues that arise post-launch.

1.2.12. Gather User Feedback:

Encourage users to provide feedback and use it to make improvements. Consider adding features based on user suggestions.

1.2.13. Marketing and Community Building:

Market the app to potential users who are interested in skill swapping. Consider partnerships or promotional activities to grow the user base.

1.2.14. Iterate and Improve:

Continuously iterate on the app based on user feedback and changing needs. Implement updates and new features to enhance the overall user experience.

CHAPTER 2

METHODOLOGY

2.1 EXISTING SYSTEM

Skill swap applications have gained popularity as platforms for individuals to exchange expertise, services, and knowledge within their communities. While these apps offer various benefits, they also come with certain disadvantages. Let's explore some existing skill swap apps and their drawbacks:

2.1.1. TaskRabbit:

Advantages:

TaskRabbit allows users to outsource small jobs and tasks to others in their area. It covers a wide range of services, including cleaning, handyman work, and virtual assistance.

Disadvantages:

Limited to Specific Services: TaskRabbit primarily focuses on household tasks and errands, limiting its utility for skill swaps beyond these areas.

Transaction Fees: The platform charges service fees for each task completed, which can reduce the overall value proposition for both service providers and users.

2.1.2. Meetup:

Advantages:

Meetup facilitates in-person gatherings and events based on shared interests and hobbies, providing opportunities for skill sharing and networking.

Disadvantages:

Limited Skill Exchange Focus: While Meetup encourages community engagement and knowledge sharing, it lacks dedicated features for direct skill swaps or service exchanges.

2.1.3. Nextdoor:**Advantages:**

Nextdoor is a neighborhood-focused social networking platform that enables residents to connect, communicate, and share resources within their local communities.

Disadvantages:

Informal Skill Exchange: While Nextdoor facilitates neighborly interactions and resource sharing, it lacks structured features for organizing formal skill swaps or service requests.

Privacy Concerns: Users may have privacy concerns about sharing personal information within their neighborhood network, potentially limiting participation in skill exchange activities.

2.1.4. Skillshare:**Advantages:**

Skillshare is an online learning platform that offers courses and tutorials on a wide range of topics, allowing users to develop new skills and knowledge.

Disadvantages:

One-Way Learning: While Skillshare provides valuable educational resources, it primarily focuses on individual skill development rather than facilitating reciprocal skill swaps between users.

Subscription Model: Access to premium content on Skillshare requires a subscription fee

2.2 PROPOSED SYSTEM

While existing skill swap apps offer valuable resources and opportunities for community engagement, they also have their limitations and drawbacks. These drawbacks may include narrow service offerings, geographic constraints, privacy concerns, subscription fees, and complex credit systems. By addressing these challenges and leveraging the strengths of existing platforms, developers can create more inclusive, user-friendly, and effective skill swap apps that empower individuals to share their expertise and talents with others in meaningful ways.

2.3 REQUIREMENT SPECIFICATION

2.3.1. Flutter App:

A Flutter app is a mobile application built using the Flutter framework, an open-source UI toolkit developed by Google. Flutter enables developers to create natively compiled applications for mobile, web, and desktop platforms from a single codebase. Here's a description of a Flutter app:

2.3.1.1. Cross-Platform Development:

Flutter allows developers to write code once and deploy it across multiple platforms, including iOS, Android, web, and desktop. This enables efficient development and maintenance of applications for diverse device ecosystems.

2.3.1.2. Reactive UI Framework:

Flutter uses a reactive UI framework, which means that the user interface reacts instantly to changes in data and state. This enables smooth animations, transitions, and interactions, resulting in a highly responsive user experience.

2.3.1.3. Widget-Based Architecture:

Flutter employs a widget-based architecture, where everything in the app is a widget. Widgets are UI elements such as buttons, text fields, and images, as well as layout elements like rows, columns, and containers.

2.3.1.4. Fast Development Cycle:

Flutter offers a fast development cycle with features like hot reload, which allows developers to see changes to the code reflected instantly on the app without restarting it. This iterative development process enhances productivity and reduces development time.

2.3.1.5. Rich Set of Widgets:

Flutter provides a rich set of pre-designed widgets for building modern and aesthetically pleasing user interfaces. These widgets are customizable and can be combined to create complex UI layouts and interactions tailored to the app's requirements.

2.3.1.6. Access to Native Features:

Flutter offers plugins and packages that provide access to native device features and functionalities such as camera, location, sensors, storage, and permissions. This allows developers to leverage native capabilities seamlessly within their Flutter apps.

2.3.1.7. Performance and Efficiency:

Flutter apps are compiled directly to native machine code, resulting in high performance and efficiency. The framework's rendering engine, Skia, ensures smooth graphics and animations, while its layered architecture minimizes overhead and maximizes performance.

2.3.1.8. Strong Community and Ecosystem:

Flutter has a vibrant and active community of developers, contributors, and enthusiasts who share knowledge, resources, and best practices.

The Flutter ecosystem includes a vast array of libraries, tools, and resources that enhance development capabilities and support various use cases.

2.3.1.9. Support for Material Design and Cupertino Styles:

Flutter offers built-in support for both Material Design, Google's design language for Android apps, and Cupertino, Apple's design language for iOS apps. This allows developers to create apps with native look and feel on each platform while maintaining a consistent brand identity.

2.3.1.10. Continuous Improvement and Updates:

Flutter is continuously evolving with regular updates, new features, and improvements. Google and the Flutter community actively contribute to the framework's development, ensuring that it remains cutting-edge and adaptable to emerging technologies and trends.

2.3.2. Visual Studio Code:

Visual Studio Code (VS Code) is an extremely useful and popular integrated development environment (IDE) for Flutter development. Here are some reasons why it's beneficial:

2.3.2.1.Lightweight and Fast:

Visual Studio Code is known for its speed and efficiency. It's lightweight compared to some other IDEs, making it quick to start up and responsive during development.

2.3.2.2.Flutter and Dart Support:

VS Code offers excellent support for Flutter and Dart development out of the box. It includes features like syntax highlighting, code completion, linting, debugging, and more, specifically tailored for Flutter development.

2.3.2.3.Extensions:

VS Code has a vast ecosystem of extensions that can enhance the Flutter development experience even further.

There are extensions available for formatting, code generation, state management, Flutter-specific snippets, and integration with various tools and services.

2.3.2.4.IntelliSense:

VS Code provides IntelliSense, which offers smart code completion and suggestions based on the context of your code. This helps speed up development and reduce errors by providing relevant suggestions as you type.

2.3.2.5.Debugging Tools:

VS Code includes robust debugging tools for Flutter applications. Developers can set breakpoints, inspect variables, step through code execution, and diagnose issues efficiently, making it easier to identify and fix bugs.

2.3.2.6.Version Control Integration:

VS Code seamlessly integrates with version control systems like Git, allowing developers to manage their code repositories directly within the IDE. This makes it convenient to collaborate with team members and track changes to the codebase.

2.3.2.7.Customizable:

VS Code is highly customizable, allowing developers to tailor their development environment to suit their preferences and workflow. Users can customize themes, keyboard shortcuts, and install extensions to add additional functionality as needed.

2.3.2.8.Community Support:

VS Code has a large and active community of developers who contribute to its development and provide support through forums, documentation, and tutorials. This means that developers can easily find help and resources when encountering issues or seeking advice.

2.3.2.9.Cross-Platform:

Visual Studio Code is available on multiple platforms, including Windows, macOS, and Linux, making it accessible to developers regardless of their operating system preferences.

2.3.3. Android Studio:

Android Studio is another popular integrated development environment (IDE) for Flutter development, alongside Visual Studio Code (VS Code). Here's how Android Studio is useful for Flutter development:

Official IDE for Android Development: Android Studio is the official IDE for Android app development, and since Flutter apps can be deployed to Android, it provides a seamless environment for Flutter developers targeting the Android platform.

Full Integration with Flutter: Google provides official support for Flutter in Android Studio, ensuring that Flutter projects are well-integrated and supported within the IDE. This includes features such as project setup, code editing, debugging, and deployment.

Dart and Flutter Plugins: Android Studio comes with built-in support for Dart, the programming language used for Flutter development. Additionally, it offers Flutter plugins that enhance the development experience by providing features such as code completion, linting, and formatting specifically tailored for Flutter projects.

UI Design Tools: Android Studio offers powerful UI design tools, such as the Layout Editor and Widget Inspector, which can help developers design and visualize UI layouts for their Flutter apps.

These tools provide a visual representation of the app's UI components and facilitate rapid prototyping and iteration.

Emulator and Device Management: Android Studio includes an emulator and device manager, allowing developers to run and test Flutter apps on virtual devices or physical devices connected to their development machine. This simplifies the testing process and ensures that Flutter apps behave correctly across different device configurations.

Built-in Git Integration: Android Studio seamlessly integrates with version control systems like Git, enabling developers to manage their Flutter projects, track changes, and collaborate with team members directly within the IDE.

Performance Profiling: Android Studio provides performance profiling tools that help developers identify and diagnose performance issues in Flutter apps. These tools allow developers to analyze CPU usage, memory allocation, and network activity, optimizing app performance and resource usage.

Gradle Build System: Android Studio uses the Gradle build system for building and packaging Android apps, including Flutter apps targeting Android. Developers can customize build configurations, manage dependencies, and automate build tasks to streamline the development process.

Community Support: Android Studio has a large and active community of developers who contribute to its development and provide support through forums, documentation, and tutorials. This ensures that developers can easily find help and resources when working with Flutter in Android Studio.

2.4. SYSTEM ARCHITECTURE

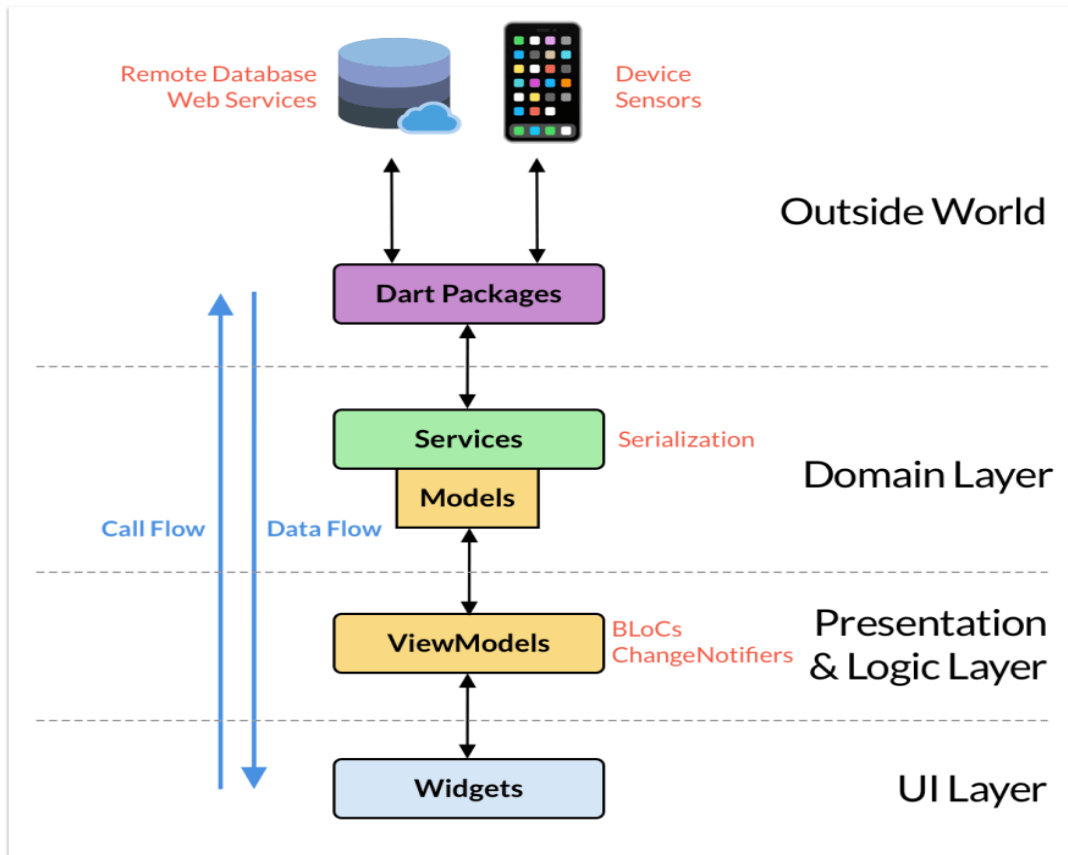


Fig 2.4.1 System Architecture

Fig 2.4.1. refers to the designing of the system architecture for a Skill Swap Flutter app involves several components to ensure smooth functioning, scalability, and security. Here's a high-level overview:

2.4.1..Outside World:

The outside world includes the device sensors, the remote database web services and dart packages. Device sensors include the device that we use like mobile phones, tablets, iPod, iPhone which has the application installed in it. The remote database web services provides services for data retrieval, authentication (login page), identifying partners that can help you etc. Dart packages are included in the code for utilizing the particular purpose of that packages.

2.4.2.Domain Layer:

In the system architecture of a Skill Swap Flutter app, the domain layer plays a crucial role in defining the business logic and rules governing the application's core functionalities. The domain layer encapsulates the application's domain model, which represents the real-world entities and their relationships within the app.

Implement domain services to encapsulate complex business logic and operations that involve multiple domain entities.

For example, services for user authentication, skill listing management, messaging, transaction handling, etc.

Domain services coordinate interactions between domain entities and enforce business rules.

Implement domain events to capture significant changes or occurrences within the domain.

Domain events can be used to trigger asynchronous processes, update other parts of the system, or notify external systems about domain-related actions.

Examples of domain events in a Skill Swap app may include SkillCreatedEvent, MessageSentEvent, TransactionCompletedEvent, etc.

2.4.3.Presentation and Logic Layer:

ViewModels play a crucial role in managing the presentation logic and state of the UI components. ViewModels act as an intermediary between the UI layer (views/widgets) and the business logic layer (domain entities, services, repositories). They facilitate the separation of concerns and help in keeping UI code clean, reusable, and easy to maintain.

Create ViewModel classes for each screen or UI component in the app that requires business logic and state management.

For example, you might have `UserViewModel`, `SkillListViewModel`, `MessageViewModel`, etc., depending on the features and screens of the app.

2.4.4.UI Layer:

In a Skill Swap Flutter app, the UI layer is responsible for presenting the user interface and interacting with users. It comprises various widgets and screens that users interact with to navigate the app, view content, input data, and perform actions. Here's how you can structure the UI layer:

2.4.4.1.Widgets:

Break down the UI into reusable widgets that represent individual UI components such as buttons, text fields, cards, etc.

Use Flutter's built-in widgets or create custom widgets as needed to match the app's design and functionality.

Widgets should be modular, composable, and customizable to promote code reuse and maintainability.

2.4.4.2.Screens:

Define screens to represent different views or sections of the app, such as login/signup screens, home screen, skill listing screen, messaging screen, etc.

Each screen typically consists of a scaffold widget as its root, with various UI widgets arranged within it to create the desired layout and functionality.

2.4.4.3.Navigation:

Implement navigation between screens using Flutter's navigation system, which includes Navigator and MaterialApp.

Define routes for each screen and navigate between them using push, pop, or named routes based on user actions or app logic.

2.4.4.4.State Management:

Choose an appropriate state management solution based on the complexity of your app and your team's preferences.

Options include Provider, Riverpod, Bloc, Redux, MobX, or even just setState for simpler apps.

Manage UI state such as user authentication state, loading indicators, error messages, etc., using the chosen state management solution.

2.4.4.5.Data Binding:

Use data binding techniques to bind UI elements to data from ViewModels or other sources.

For example, use ValueNotifier, StreamBuilder, Consumer (if using Provider), or similar widgets to automatically update UI elements when data changes.

2.4.4.6.UI Styling and Theming:

Define a consistent UI design and apply styling and theming throughout the app to maintain a cohesive look and feel.

Utilize Flutter's theming capabilities to define colors, fonts, text styles, and other visual properties globally or for specific widgets/screens.

2.4.4.7.User Input Handling:

Implement user input handling to capture user interactions such as taps, swipes, text input, etc.

Use GestureDetector, InkWell, TextField, and other input widgets to handle user input events and respond accordingly.

2.4.4.8.Responsive Design:

Design UI layouts to be responsive and adaptive to different screen sizes and orientations.

Utilize Flutter's layout widgets like MediaQuery, LayoutBuilder, Expanded, Flexible, etc., to create responsive UI designs that look good on various devices.

2.4.4.9.Localization and Internationalization:

Support multiple languages and locales by implementing localization and internationalization features.

Use Flutter's intl package or similar solutions to provide translations for different languages and adapt UI text and formats based on the user's locale.

2.4.4.10.Testing:

Write UI tests using Flutter's testing framework (e.g., widget tests, integration tests) to ensure that UI elements and interactions behave as expected.

Test UI responsiveness, navigation, input handling, state management, and other UI-related functionalities.

CHAPTER 3

RESULTS AND DISCUSSION

3.1.RESULTS

3.1.1.Homepage code:

Fig 3.1.1.1 refers to the home page which consists of login and sign up with authentication where the existing user logs in and the new user does sign up registering with their name, age, email(optional), location, skills and designation.

Once the user logs in he/she will be directed to a page where the actual application starts.

```
import 'package:demo/secondPage.dart';
import 'package:demo/signupPage.dart';
import 'package:flutter/material.dart';
import 'package:flutter/src/widgets/container.dart';
import 'package:flutter/src/widgets/framework.dart';
import 'package:get/get.dart';

void main() {
  runApp(MyHomePage());
}

class MyHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Login Page',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(
          title: Text('SKILL SWAP',textAlign: TextAlign.center,),
          leading: Icon(
            IconData(0x0053, fontFamily: 'CustomIcons'), // 0x0053 is the Unicode code point for the "S" character
            size: 24,
          ),
        ),
        body: Padding(
          padding: const EdgeInsets.all(20.0),
        ),
      ),
    );
  }
}
```

Fig 3.1.1.1. Home Page

3.1.2.Second Page:

Fig 3.1.2.1. refers to the login the user will be directed to a secondpage of the application where the user will find a bottom navigation bar containing a homepage, search page and profile page. Using the matching algorithm the user will be given suggestions for their required skills and as well as those required in their skills.

If the user clicks any profile that matches their requirement, a chat box will be opened so that the user can ask their doubts and get cleared. As soon as their chat ends the chats will be deleted and contact will be end.

```
import 'package:flutter/material.dart';

Run | Debug | Profile
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'SKILL SWAP',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ), // ThemeData
      home: SecondPage(),
    ); // MaterialApp
  }
}

class SecondPage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State< SecondPage> {
  int _selectedIndex = 0;

  final List<Widget> _pages = [
```

Fig 3.1.2.1 Login Page

3.2.Output Screenshots

3.2.1.Login Page

Fig 3.2.1.1. refers to the login page has login as well as sign up option for the new user. Login has authentication (i.e, if wrong password or username is typed the login will show error message and will be unable to login).

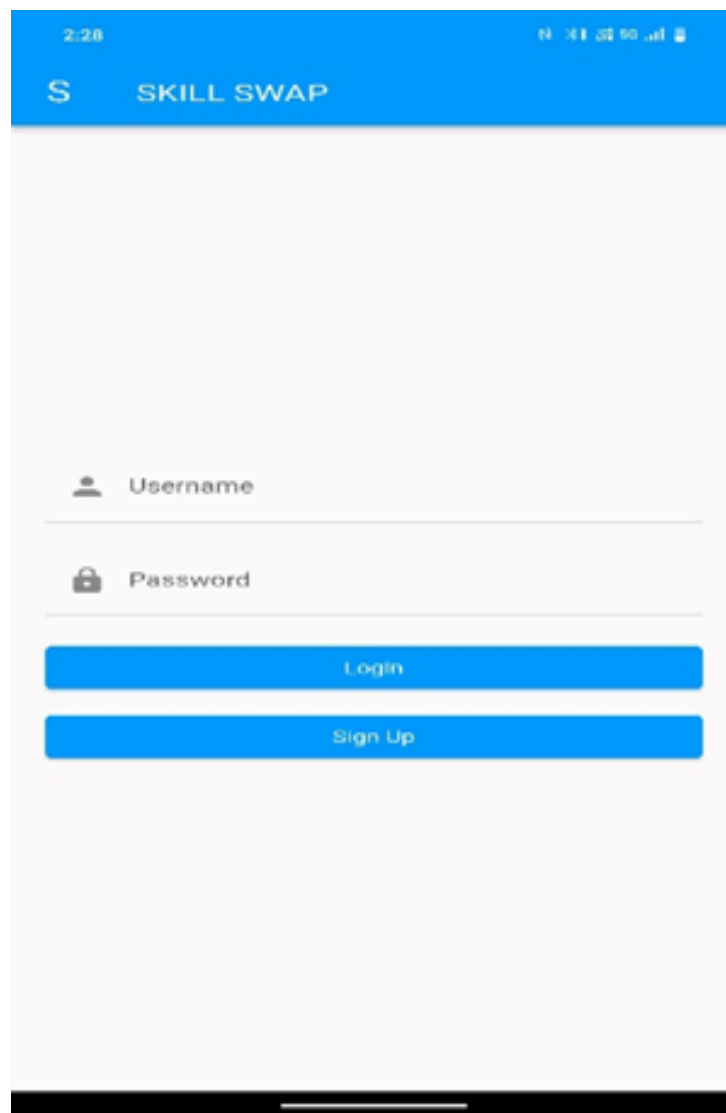


Fig 3.2.1.1 Login Page

3.2.2.Search Page

Fig 3.2.2.1. refers to the search page consists of popular searches and a search bar where we can search and find the profile that can rectify our doubts in the topic they are skilled in.

It searches and gives you a list of profiles who all are expert in particular domain so that we can have a conversation with them.



Fig 3.2.2.1 Search Page

3.2.3.Home Page

Fig 3.2.3.1. refers to the home page in the main page when we enter the application after login or signup. It has the list of suggestions through matching algorithm. Based on your input in the sign up (your requirements) page this matching algorithm gives you suggestions. When a view card of any user is clicked a chatbox opens.

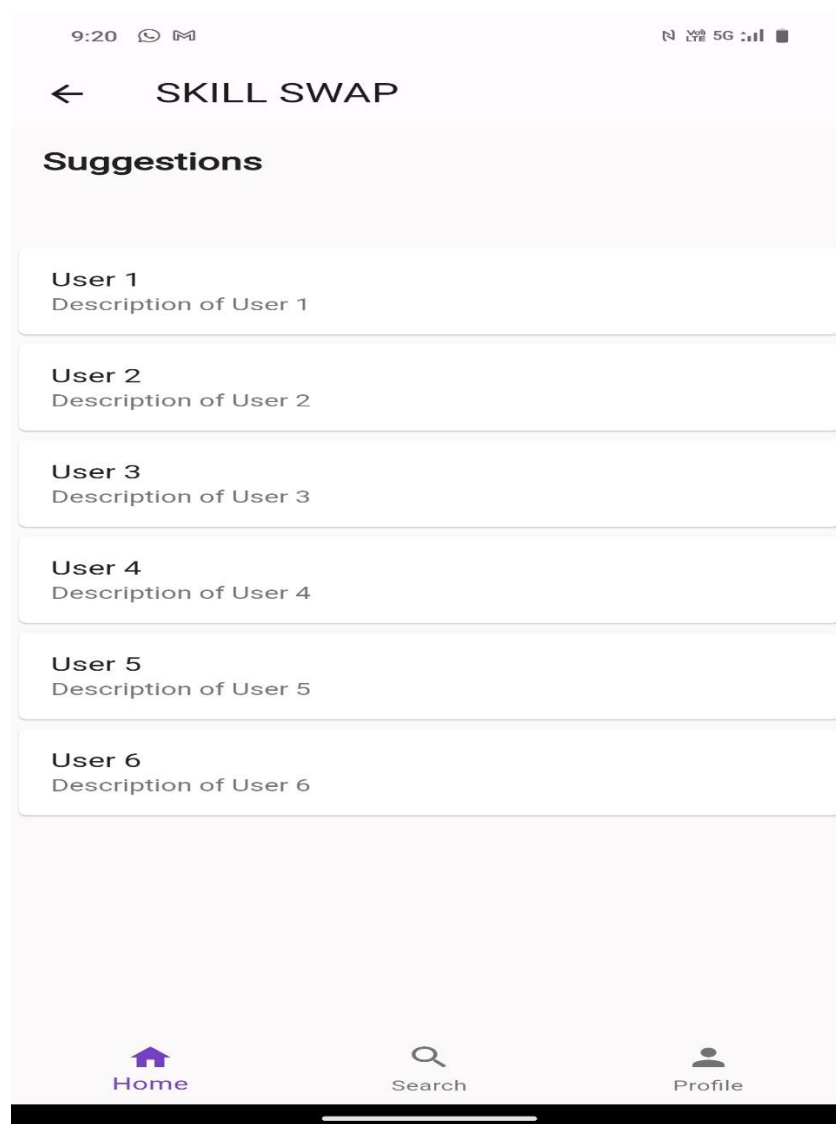


Fig 3.2.3.1.Home Page

3.2.4.Profile Page

Fig 3.2.4.1.refers to the profile page where all the details of an user reside. It consists of information given by the user at the time of signup. Since its based on our skills, everyday people might improve their skills and may expand their knowledge so there is an option here to edit your profile whenever there is a new skill to add or update your company name anything.



Fig 3.2.4.1. Profile Page

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 CONCLUSION

In conclusion, the Skill Swap Flutter app presents a dynamic platform where individuals can exchange skills, knowledge, and experiences within a vibrant community. By leveraging the power of Flutter, we've crafted an intuitive and responsive application that seamlessly connects users with opportunities for personal growth and collaboration.

Through its user-friendly interface and robust features, the app empowers users to both share their expertise and discover new skills in an interactive and supportive environment.

Whether you're a seasoned professional looking to mentor others or a newcomer eager to learn, Skill Swap provides a space for meaningful connections and fruitful exchanges.

Furthermore, by embracing the Flutter framework, we've ensured cross-platform compatibility, allowing users to access the app seamlessly across various devices. This accessibility enhances the reach and impact of Skill Swap, enabling users from diverse backgrounds to come together and enrich each other's lives through shared learning experiences.

As we continue to refine and expand the app's capabilities, we remain committed to fostering a thriving community where individuals can grow, connect, and inspire one another.

With Skill Swap, the possibilities for learning and collaboration are endless, empowering users to unlock their full potential and achieve their goals.

4.2.FUTURE WORKS

4.2.1.Advanced Matching Algorithms: Implement more sophisticated algorithms to enhance the skill matching process, taking into account factors such as user preferences, proficiency levels, and location to provide more accurate and relevant matches.

4.2.2.In-App Messaging System: Introduce an in-app messaging system to facilitate communication between users, allowing them to discuss skill exchange details, coordinate meetups, and provide feedback on sessions.

4.2.3.User Ratings and Reviews: Enable users to rate and review each other based on their skill exchange interactions, fostering a trustworthy and transparent community where members can build reputations and establish credibility.

4.2.4.Integration with Social Media Platforms: Incorporate social media integration to enable users to share their skill exchange experiences, invite friends to join the platform, and expand their network beyond the app.

4.2.5.Monetization Features: Explore monetization opportunities such as premium memberships, sponsored content, or in-app advertisements to generate revenue and sustain the growth of the platform.

4.2.6.Enhanced Profile Customization: Allow users to personalize their profiles with additional information, including portfolios, certifications, and interests, to provide more context about their skills and expertise.

4.2.7.Community Events and Workshops: Organize virtual or in-person events, workshops, and seminars within the app to promote skill development, foster community engagement, and provide opportunities for networking and collaboration.

REFERENCES

- Shady Boukhary, "Flutter Clean Architecture Pub Package", April 2019, [online] Available: https://pub.dev/packages/flutter_clean_architecture.
- Philip A. Laplante, "Requirements engineering for software and systems", Boca Raton, Florida: CRC Press, pp. 168, 2014, ISBN 9781466560819.
- Nilesh Limbore, "AN ANALYTICAL STUDY OF INDIAN AGRICULTURE CROP PRODUCTION AND EX-PORT WITH REFERENCE TO WHEAT", Review of Research 2249-894X Impact points 3.1402(UIF), vol. 4, no. 11, 2015.
- H. Patel and D. Patel, "Survey of Android Apps for Agriculture Sector", *International Journal of Information Sciences and Techniques*, vol. 6, pp. 61-67, 2016.
- Yasir Babiker Hamdan and A. Sathesh's, "Study of Program Sizing and Data Will be based estimation by Machine Learning Taxonomy", *Journal of Wireless Communication Technologies and Computing*, vol. 3, no. 4, pp. 303-313, 2021.
- Ashutosh Kumar Yadav and Vyom Kulshreshtha, *the world wide Journal of Studies in Engineering Management and Science*, vol. 4, no. 1, pp. 88-92, 2021.