

TUTORKEY – AN ONLINE TUTOR FINDER APP USING FLUTTER AND FIREBASE

PROJECT REPORT

Submitted by

SAFWA NIZAMUDEEN

TKM23MCA-2052

to

TKM College of Engineering

Affiliated to

The APJ Abdul Kalam Technological University

In partial fulfilment of the requirements for the award of the Degree

of

MASTER OF COMPUTER APPLICATION



DEPARTMENT OF COMPUTER APPLICATIONS

**Thangal Kunju Musaliar College of Engineering
Kerala**

NOVEMBER 2024

DECLARATION

I undersigned hereby declare that the project report **TUTORKEY**, submitted for partial fulfilment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of **Prof. Sheera Shamsu**. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Kollam

11/11/2024

SAFWA NIZAMUDEEN

DEPARTMENT OF COMPUTER APPLICATION
TKM COLLEGE OF ENGINEERING, KOLLAM



CERTIFICATE

This is to certify that, the report entitled **TUTORKEY** submitted by **SAFWA NIZAMUDEEN(TKM23MCA-2052)** to the **APJ Abdul Kalam Technological University** in partial fulfilment of the requirements for the award of the Degree of **Master of Computer Applications** is a bonafide record of the project work carried out by her under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor(s)

Mini Project Co-ordinator

ACKNOWLEDGEMENT

First and foremost, I thank GOD almighty and our parents for the success of this project. I owe sincere gratitude and heart full thanks to everyone who shared their precious time and knowledge for the successful completion of my project. I am extremely grateful to **Prof. Natheera Beevi M**, Head of the Department, Department of Computer Application, for providing us with the best facilities. I would like to thank my project guide and project coordinator **Prof. Sheera Shamsu**. I profusely thank all other faculty members in the department and all other members of TKM College of Engineering, for their guidance and inspiration throughout my course of study. I owe thanks to my friends and all others who have directly or indirectly helped me in the successful completion of this project.

SAFWA NIZAMUDEEN

ABSTRACT

TutorKey is a mobile application developed using Flutter and Dart for the frontend and Firebase for backend support, designed to bridge the gap between learners seeking private tutoring and qualified tutors across various subjects. This digital platform aims to simplify the process of finding, booking, and managing tutoring sessions through a seamless, cross-platform experience. TutorKey supports two user roles: tutors and tutees, each with a dedicated dashboard. Tutors can manage profiles, set hourly tuition fees, specify subjects, and schedule daily availability slots. Tutees can browse tutors based on subject expertise, view tutor profiles, and book real-time sessions for the current day, with an option to specify session locations.

Firebase integration provides secure authentication, real-time data synchronization, and session status management. Both tutors and tutees can monitor booking stages, including pending, ongoing, unpaid, completed, and rejected sessions. TutorKey thus addresses key challenges, such as tutor credibility, scheduling conflicts, and session management, fostering an organized and accessible private tutoring ecosystem. Future enhancements may include a review and rating system to further support user decision-making and enrich the platform's functionality.

TABLE OF CONTENTS

1. INTRODUCTION	1
1.1 Existing System	2
1.2 Problem Statement.....	3
1.3 Proposed System	3
1.4 Objectives	4
2. LITERATURE SURVEY	6
2.1 Purpose of Literature Survey	6
2.2 Related Works	6
3. METHODOLOGY	9
3.1 Architecture	10
3.1.1 System Design	10
3.2 Software Requirements and Specifications	11
3.2.1 Flutter	12
3.2.2 Firebase	13
3.2.3 Google Chrome	14
3.2.4 Visual Studio Code	15
3.2.5 Android Studio	16
3.2.6 Dart	17
3.3 Dependencies	18
3.3.1 Firebase Core.....	19
3.3.2 Firebase Auth.....	19
3.3.3 Cloud Firestore	20
3.4 Hardware Requirements	20
4. RESULTS AND DISCUSSIONS	22
4.1 Splash Screen	23
4.2 Login and Sign up Page.....	24
4.3 Tutor Dashboard.....	25
4.3.1 Profile and Scheduling Section	26
4.3.2 Tutor Sessions.....	28

4.4 Tutee Dashboard.....	31
4.4.1 Profile and Browse Section	32
4.4.2 Tutor Sessions.....	34
5.CONCLUSION	36
6.FUTURE ENHANCEMENTS	39
REFERENCES.....	41

LIST OF FIGURES

3.1.1 System Design.....	11
4.1 Splash Screen	23
4.2(a) Login Page	24
4.2(b) Sign up Page.....	24
4.3 Tutor Dashboard	25
4.3.1(a) Manage Tutor Profile	26
4.3.1(b) Manage Tutor Fee.....	26
4.3.1(c) Schedule	26
4.3.1(d) Subject	26
4.3.2(a) Pending Session.....	28
4.3.2(b) Ongoing Session.....	28
4.3.2(c) Unpaid Session	29
4.3.2(d) Completed Session	29
4.3.2(e) Rejected Session	29
4.4 Tutee Dashboard.....	31
4.4.1(a) Manage Tutee Profile.....	32
4.4.1(b) Browse Tutors	32
4.4.1(c) Tutor Selection	33
4.4.1(d) Session Booking	33

CHAPTER 1

INTRODUCTION

In recent years, digital platforms have transformed how individuals access services, including education and private tutoring. As academic demands grow, learners often seek personalized assistance to improve their understanding of subjects, enhance academic performance, and develop lifelong learning skills. However, finding reliable and qualified tutors remains a challenge due to limitations in accessibility, lack of transparency in tutor qualifications, and the difficulty of scheduling sessions that fit both tutor and student availability.

TutorKey is designed to bridge this gap by providing a streamlined, user-friendly platform that connects learners, or tutees, with experienced tutors. Built using Flutter and Dart for the frontend and Firebase for backend integration, TutorKey simplifies the tutor-tutee relationship by offering role-specific dashboards that enhance the tutoring experience for both parties. Tutors can create and manage profiles detailing their qualifications, experience, subject expertise, and hourly tuition rates, giving learners valuable information to make informed choices. Additionally, TutorKey allows tutors to set up daily availability in three time slots, enabling flexible scheduling and making it easy for learners to book real-time sessions.

Through TutorKey, learners can browse tutors based on subject expertise, view tutor profiles, check slot availability, and submit booking requests. This real-time booking process ensures that both parties have clarity on session timings and expectations, reducing the likelihood of scheduling conflicts. The platform also offers a well-defined session status system with categories such as pending, ongoing, unpaid, completed, and rejected, providing a transparent and organized session management experience.

TutorKey not only meets the growing need for accessible, quality private tutoring but also ensures that sessions are well-coordinated and transparent, fostering an environment of trust and reliability. By harnessing the benefits of Firebase's real-time capabilities, TutorKey provides a secure, reliable tutoring platform that enhances access to academic support, aiming to promote academic growth and make quality education accessible to learners anywhere.

1.1 Existing System

In traditional tutoring systems, learners often rely on manual search methods to find a suitable tutor, such as through word-of-mouth recommendations, public noticeboards, or school bulletin boards. While these methods can occasionally yield positive results, they are generally limited by geography, requiring learners and tutors to be in the same locality. Additionally, this process is often time-consuming and lacks reliability, as learners may have difficulty finding tutors with the right qualifications or availability. This limited accessibility can hinder the learning process, especially for students seeking specific subject expertise or flexible scheduling options.

Some online platforms have attempted to address these needs by providing basic tutor search functionalities, but most fall short in offering a comprehensive solution. Fragmented in nature, these platforms might allow users to search for tutors by subject or location but often lack real-time features that would make tutoring more efficient and convenient. For instance, few platforms provide the capability to book sessions instantly or manage multiple sessions in a streamlined way. Without these functionalities, learners and tutors frequently experience scheduling conflicts, difficulty tracking session progress, and confusion around session statuses, creating barriers to effective tutoring.

Another significant limitation of existing systems is the lack of verification processes to ensure tutor reliability and credibility. Many online platforms do not rigorously verify tutor credentials, which can create trust issues among learners. This absence of quality control not only discourages learners but also undermines the value of online tutoring as a whole, as learners are left uncertain about the qualifications and experience of the tutors they engage with.

Certain tutoring methods also lack an efficient session management system. Without structured features for tracking session statuses—such as pending, ongoing, completed, or paid sessions—users find it difficult to stay organized, leading to missed payments, scheduling mishaps, and incomplete sessions. This gap in session management results in an unproductive and often frustrating experience for both learners and tutors. A modernized, well-structured system is needed to address these challenges, ensuring that tutoring sessions are efficiently coordinated, transparent, and beneficial for all parties involved.

1.2 Problem Statement

TutorKey addresses several core issues faced by learners and tutors in traditional and existing digital tutoring platforms. First, learners often struggle to find qualified tutors who align with their academic needs, possess verified credentials, and offer flexible availability. Without a structured platform, the process of locating and engaging reliable tutors can be timeconsuming and challenging. Additionally, the lack of an effective scheduling and availability system often leads to coordination difficulties. Tutors and learners encounter frequent conflicts, such as double bookings or missed sessions, due to unclear scheduling, resulting in an inefficient tutoring experience.

The absence of reliable scheduling tools also contributes to an inefficient experience. Many platforms lack real-time scheduling, leading to scheduling conflicts, repeated communications, and missed sessions, which distract from the learning process. Tutors, in particular, struggle with managing their availability due to the absence of flexible scheduling tools, further complicating the process. Session status tracking is another issue with traditional methods, as many platforms don't offer a clear structure to track session progress, payments, and outcomes. This can lead to confusion over unpaid sessions and missed payments. For tutors, this lack of organization makes it difficult to manage their workload and financial records, while for learners, it limits their ability to monitor their progress and prepare for future lessons.

TutorKey aims to address these challenges by providing a digital platform that enhances accessibility, reliability, and organization. With features such as user-provided tutor profiles, real-time scheduling, session tracking, and academic progress tools, TutorKey ensures a seamless and transparent experience for both learners and tutors. By simplifying the process of finding and engaging tutors based on user-provided information, it fosters a more efficient and productive tutoring environment.

1.3 Proposed System

TutorKey is designed to overcome the limitations of traditional and existing tutoring platforms by providing a streamlined, feature-rich solution tailored for both tutors and learners. Built with Flutter for the frontend and Firebase for backend integration, TutorKey offers a

responsive and user-friendly platform that connects students with qualified tutors through rolespecific dashboards.

The platform simplifies the tutoring process by allowing tutors to manage profiles, set their availability, and select subjects they teach. Tutees can browse tutor profiles, check availability in real time, and book sessions instantly, reducing scheduling conflicts. Tutors can set daily availability across three time slots, which are updated in real time, making it easy for tutees to book immediate sessions.

A key feature of TutorKey is its efficient session tracking system. Sessions are organized into clearly defined categories: Pending, Ongoing, Unpaid, Completed, and Rejected. This structure ensures both tutors and tutees can easily track the status of each session, manage payments, and stay organized. Tutees can monitor their learning progress while tutors can manage their workload effectively. TutorKey also eliminates the need for lengthy verification processes. Tutors can join the platform quickly by self-reporting their qualifications and experience, allowing for fast onboarding and immediate access to available tutors. While the platform relies on user-provided information, the structured and transparent profile system ensures that tutees can make informed decisions when selecting a tutor based on qualifications and expertise.

Built with real-time features powered by Firebase, TutorKey enhances the learning experience by offering flexible scheduling, clear session tracking, and a transparent payment process. The platform's intuitive design and user-friendly interface make it accessible from any device, ensuring a seamless tutoring experience. TutorKey is also designed to grow with future enhancements, such as tutor verification and multi-day scheduling, providing a flexible and reliable platform for both tutors and tutees.

1.3 Objectives

The primary objective of TutorKey is to create a streamlined and efficient platform that facilitates a seamless connection between tutors and students. The platform is designed to address the key challenges faced in traditional and existing tutoring systems, making it easier for both tutors and learners to engage with each other in a flexible, organized, and transparent manner. The specific objectives of TutorKey include:

- **Enabling Easy Access to Tutoring Services:** TutorKey aims to simplify the process of finding and booking tutoring sessions by offering an intuitive platform. Tutees can easily browse through available tutors, view their qualifications and availability, and book sessions in real time. This accessibility ensures that students can quickly connect with the right tutor for their learning needs, eliminating time-consuming search and scheduling processes.
- **Providing Efficient Profile and Session Management:** The platform creates role-specific dashboards that allow tutors and tutees to manage their profiles and sessions with ease. Tutors can input and update their qualifications, experience, hourly rates, and subject expertise, while managing their availability in real time. For tutees, the dashboard includes a browse feature, enabling them to find tutors based on specific subjects and availability. Additionally, TutorKey's session management system allows both parties to track session statuses—whether pending, ongoing, unpaid, completed, or rejected—ensuring clarity and organization in the tutoring process.
- **Encouraging Real-Time Interaction:** By supporting real-time booking and session updates, TutorKey ensures that tutors and tutees can interact and schedule sessions instantly based on availability. The platform reduces delays caused by communication gaps, making tutoring more flexible and responsive. Immediate scheduling allows for dynamic learning opportunities and ensures that students receive timely academic support when needed.
- **Laying the Foundation for Future Improvements:** TutorKey not only addresses the current gaps in tutoring systems but also provides a foundation for future growth and improvements. The platform's structure allows for the addition of advanced features such as tutor qualification verification, multi-day scheduling options, and enhanced session tracking. By continually evolving, TutorKey aims to further improve the tutoring experience and meet the evolving needs of both tutors and learners.

CHAPTER 2

LITERATURE SURVEY

A literature survey, also known as a literature review, involves analysing scholarly sources related to a particular subject. Examining the available literature, it provides a comprehensive overview of the state of the field, allowing you to identify relevant theories, approaches, and gaps in the existing body of knowledge. When conducting a literature review from an audit perspective, the main focus is on evaluating the relevant literature. This process covers information that has been published in a specific field of study and sometimes includes information published within a specific time frame.

2.1 Purpose of Literature Survey

- It gives readers easy access to research on a particular topic by selecting high quality articles or studies that are relevant, meaningful, important and valid and summarising them into one complete report.
- It provides an excellent starting point for researchers beginning to do research in a new area by forcing them to summarise, evaluate, and compare original research in that specific area.
- It ensures that researchers do not duplicate work that has already been done.
- It can provide clues as to where future research is heading or recommend areas on which to focus.
- It highlights the key findings.

2.2 Related Works

The study by Gonzales et al. (2023) provides an in-depth look at the development and evaluation of *Project Learn*, a cross-platform application created to simplify the process of finding qualified tutors. Aiming to fill gaps in accessibility to tutoring services, *Project Learn* was designed using a developmental research approach, with a focus on ISO 25010 software quality standards, which include functional suitability, usability, and security. By aligning with these standards, the application demonstrates a commitment to high-quality, user-friendly functionality, ensuring that students can easily find tutors who meet their academic needs. The

research involved a comprehensive assessment from end-users and IT experts, both of whom rated the application highly for its effectiveness in connecting students to tutors in a secure and dependable environment. Key features include a clear user interface, efficient scheduling, and transparent profile information, making it easy for students to select tutors based on subject expertise and availability. Gonzales et al. underscore that technical quality, combined with a user-centered design, plays a critical role in user trust and satisfaction. This paper advances the field by illustrating how adherence to established technical standards can improve user confidence in educational technology.[1]

Enwereji, van Rooyen, and Terblanche (2023) focus on the role of e-tutoring in distance education, with a case study of its implementation at the University of South Africa (UNISA). Their study assesses students' perceptions of e-tutoring as an essential tool for enhancing learning in remote settings, especially as students face challenges such as limited physical access to instructors and classmates. E-tutoring has shown to improve comprehension, build student confidence, and provide timely academic support, all of which contribute to better learning outcomes in a distance-learning context. However, the study also highlights significant obstacles, including unreliable internet connections, the need for e-tutors to undergo additional training, and the challenge of maintaining effective communication. The authors recommend that institutions address these issues by investing in digital infrastructure and providing regular training for e-tutors to better support students. This study sheds light on how e-tutoring not only aids in academic performance but also plays a role in fostering a supportive learning environment, crucial for student retention and success in distance education settings.[2]

Shaikh, Kasat, and Shinde (2021) emphasize that trust is fundamental to the effective operation of smart education systems, particularly given the forced shift to online learning during the COVID-19 pandemic. This study examines how trust acts as a key factor in fostering positive educational experiences within smart classrooms, especially as both faculty and students navigate new digital tools. Trust in smart education is multi-dimensional, involving attributes such as attentiveness, transparent evaluation, ethical conduct, and effective use of technology. The authors argue that these elements collectively build a sense of security and reliability for students, which is critical for engagement in online learning. Additionally, the study points out that cultural, educational, and environmental factors must be considered in developing smart education tools, as one standardized approach cannot meet the diverse needs of all students. The authors stress the importance of faculty training and digital readiness as

prerequisites for a trustworthy and functional smart education system, particularly as technology becomes a central pillar in modern education. This study contributes to the literature by highlighting that trust-building measures, such as real-time feedback, clear evaluations, and digital skill development, are essential for the success of any smart educational initiative.[3]

Chougale, Yadav, and Gaikwad (2021) explore the usage of Firebase in Android app development, highlighting its cloud-based services such as authentication, real-time database, cloud messaging, and performance monitoring. Firebase enables efficient data storage and synchronization, offering features like Firebase Realtime Database for real-time updates and Firebase Authentication for streamlined user login with social accounts. The authors emphasize Firebase's advantages over traditional Relational Database Management Systems (RDBMS), particularly in managing unstructured data. Firebase's integration with Android applications enhances development speed, performance, and scalability, with tools like Firebase Test Lab supporting cross-device testing. However, the paper also acknowledges some limitations, such as Firebase's restricted querying capabilities, suggesting the need for additional services like Elastic Search for more advanced queries. Firebase's seamless integration with Flutter for cross-platform app development makes it a suitable choice for building robust and secure applications, aligning well with the needs of modern mobile app projects.[4]

CHAPTER 3

METHODOLOGY

The TutorKey application is designed to provide a seamless and efficient platform for connecting tutors with tutees, offering an easy way to manage tutoring sessions and improve the learning experience. Built using Flutter, this cross-platform framework allows the app to run on both Android and web platforms with a single codebase, simplifying development and ensuring a consistent user experience. The application leverages Firebase as the backend to handle critical functionalities such as user authentication, real-time database management, and cloud storage, creating a reliable infrastructure for user data and session information. The core aim of the app is to provide an intuitive interface for users to find tutors, check their qualifications, and schedule sessions according to their needs, making online learning more accessible and effective.

Flutter's use in the TutorKey app offers a flexible and responsive design, which is essential for catering to different screen sizes and platforms. It enables the creation of a smooth, natively-like experience, allowing users to interact with the application effortlessly. Firebase powers the backend of the app, ensuring real-time synchronization of data and secure management of user information. Firebase Authentication is used to streamline user login and sign-up processes, allowing for easy integration of multiple authentication methods, including email and social logins. Additionally, Firebase's real-time database, Cloud Firestore, ensures that session data, tutor profiles, and tutee preferences are always up-to-date, creating a dynamic and interactive learning environment. Firebase Storage allows tutors to upload documents, resources, and qualifications, which can be accessed by tutees to assess the suitability of a tutor before booking a session.

The architecture of TutorKey has been designed with scalability and performance in mind, ensuring that the app can grow with future enhancements. It allows for smooth management of users and their interactions, from profile creation to session scheduling. Flutter's integration with Firebase makes it possible to handle complex features such as real-time messaging, session updates, and profile management efficiently. Moreover, the app's interface is optimized for ease of use, enabling users to perform tasks such as searching for tutors, booking sessions, and viewing their schedules with minimal effort. By using Firebase's cloud-based services, the app

also benefits from secure data handling and robust performance monitoring. The methodology section outlines the technical tools, frameworks, and dependencies involved in the development of TutorKey, highlighting the key components that make the app functional, reliable, and scalable for online tutoring.

3.1 Architecture

The architecture of the TutorKey system is designed to follow a client-server model, which ensures efficient communication and interaction between the mobile application (client side) and the backend (server side) infrastructure. This architecture divides the system into two primary components: the client-side (frontend) and the server-side (backend), both of which interact with each other seamlessly to ensure a smooth and effective user experience.

- **Client-side (Frontend):** This includes the mobile application built using Flutter, which communicates with Firebase services to fetch and display data. The user interfaces for both tutors and tutees are designed to be intuitive, allowing easy navigation through their profiles, sessions, and messaging functionalities.
- **Server-side (Backend):** Firebase serves as the backend platform for the application, handling user authentication, real-time data storage, and synchronization across users. Firebase's cloud infrastructure ensures scalability and security for the app's data.

3.1.1 System Design

The system design of TutorKey adopts a modular approach to ensure scalability, maintainability, and separation of concerns. The design consists of distinct layers that handle different aspects of the application, including user management, session management, and data storage. The key components of the system are as follows:

- **User Management:** This component handles user authentication through Firebase Authentication. It enables tutors and tutees to securely log in, sign up, and manage their profiles. The authentication system supports different sign-in methods, including email and password authentication, ensuring a secure user experience.
- **Session Management:** TutorKey allows users to schedule and manage tutoring sessions. This module stores session data in Firebase Cloud Firestore, ensuring that information

about upcoming sessions, session times, and other relevant details are kept in sync and accessible to both tutors and tutees.

- **Data Storage and Synchronization:** Firebase's Realtime Database and Cloud Firestore are used to store and synchronize key user data, including user profiles and session information. This ensures that the application data is always up-to-date, even across devices. Firebase handles the backend logic, ensuring scalability and security.

The system design ensures that the application remains responsive and consistent across both Android and iOS platforms, thanks to the use of Flutter. This eliminates the need for platform-specific code while maintaining a seamless user experience.

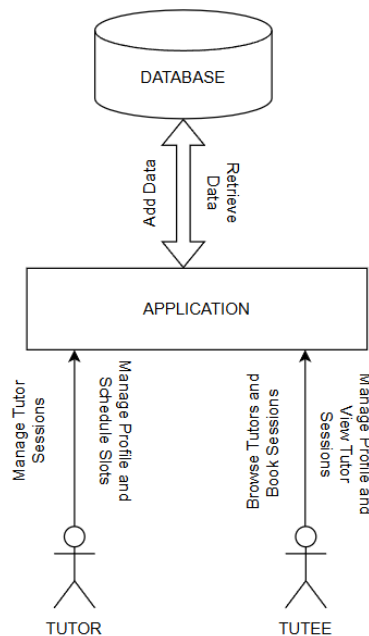


Figure 3.1.1 System Design

3.2 Software Requirements and Specifications

The software requirements for this project includes:

1. Flutter
2. Firebase

3. Google Chrome
4. VS Code
5. Android Studio
6. Dart

3.2.1 Flutter

Flutter, developed by Google, is a powerful and flexible open-source framework designed specifically for building cross-platform applications with a single codebase. This means that with Flutter, developers can write one set of code and deploy it to both Android and iOS devices, eliminating the need to write and maintain separate codebases for each platform. This efficiency makes Flutter a highly attractive choice for TutorKey, as it simplifies the development process, reduces costs, and accelerates the time to market.

A standout feature of Flutter is its widget-based architecture. In Flutter, everything is a widget—text, images, buttons, layout structures, and more are all composed of widgets. This architecture allows for a high degree of flexibility and customization, giving the development team the freedom to design each component of TutorKey’s interface to meet specific user needs. Widgets can be easily combined, customized, or even created from scratch, making it straightforward to craft unique and visually consistent interfaces for both tutors and tutees. The pre-built widgets Flutter provides are not only extensive but also follow Google’s Material Design principles, ensuring a high-quality, native-like experience on Android, while Cupertino widgets allow for an iOS-specific look and feel. This adaptability ensures that TutorKey looks professional and feels intuitive to users, regardless of their device.

Flutter’s hot reload feature is another crucial tool in the development process. This capability allows developers to instantly view the effects of code changes without restarting the entire application. When developers make adjustments to the UI, fix bugs, or add new features, they can see these changes applied immediately in the emulator or on a connected device. For TutorKey, this means faster debugging and a smoother development cycle, as developers can quickly iterate on the user interface and functionality, significantly enhancing productivity.

Moreover, Flutter’s performance is a major advantage. Unlike traditional crossplatform frameworks that rely on WebViews or other intermediate layers, Flutter compiles directly to native code. This native compilation approach ensures that TutorKey runs smoothly and

efficiently, with near-native performance, which is essential for delivering a responsive and satisfying experience to users. Whether tutors are navigating through schedules or tutees are updating their profiles, they experience minimal lag or loading delays, making the app feel seamless and engaging.

Flutter supports a vast ecosystem of plugins and packages that integrate with popular services, including Firebase, which is TutorKey's backend. This integration enables the app to easily use Firebase services for tasks like authentication, database management, and storage, simplifying backend operations and streamlining data handling between the front end and back end.

Flutter provides TutorKey with a robust toolkit that simplifies cross-platform development, enhances UI customization, and improves productivity. Its high performance, combined with a comprehensive widget system and fast development feedback, enables TutorKey to deliver an efficient, engaging, and user-friendly experience across both Android and iOS platforms.

3.2.2 Firebase

Firebase is a robust backend-as-a-service platform provided by Google that supports the development of web and mobile applications. It offers a suite of tools and services that streamline backend functionality, allowing developers to focus more on building engaging user experiences. A key feature of Firebase is its Authentication service, which simplifies user login with support for email and social logins like Google and Facebook. Firebase also provides powerful, cloud-hosted databases: Firestore for structured data and Realtime Database for applications that need data updates in real-time across multiple users.

Beyond data storage, Firebase includes Cloud Functions, which enable server-side code execution in response to app events, reducing the need for traditional backend infrastructure. Cloud Storage in Firebase is ideal for handling and serving large files such as images or videos, and Firebase Hosting offers fast, secure deployment for web applications and static content. Additionally, Firebase Analytics allows developers to monitor app performance, user engagement, and events, while Crashlytics offers detailed reports to address app crashes and

improve stability. Finally, Firebase's Machine Learning Kit makes it easy to incorporate AI features like image recognition or text analysis with minimal setup.

Together, these features make Firebase highly scalable and flexible, catering to both small projects and large-scale enterprise applications. Its seamless integration with Google's ecosystem also allows developers to leverage other services like Google Cloud, making Firebase an all-in-one platform that is well-suited to modern app development needs.

3.2.3 Google Chrome

Google Chrome plays a crucial role in the development and testing process for TutorKey, especially when it comes to debugging, performance optimization, and ensuring cross-platform functionality. With Chrome DevTools, developers have access to powerful tools for inspecting and troubleshooting the web-based components of the app. DevTools provides detailed insights into the Document Object Model (DOM) structure, CSS styling, and JavaScript execution, allowing developers to monitor and fine-tune the visual layout and interactive elements of TutorKey in real time. This is essential for verifying that the app renders correctly and behaves as intended across different screen sizes and resolutions.

For TutorKey, Chrome's Network and Performance panels are particularly valuable. The Network panel enables developers to monitor network requests and responses, providing insights into data transfer speeds and identifying any bottlenecks in data retrieval from Firebase or other backend services. The Performance panel allows developers to profile the app's behavior, tracking resource usage, load times, and frame rates. This helps pinpoint areas where the app may lag or use excessive resources, providing opportunities to optimize loading times and ensure a smooth, responsive user experience.

Furthermore, Chrome's Emulation mode allows developers to simulate various devices and network conditions directly in the browser, making it easier to see how TutorKey will perform on different mobile devices and under varying internet speeds. By testing TutorKey in Chrome, developers can assess the app's consistency across platforms, troubleshoot potential compatibility issues, and optimize for performance—all before deploying the app to actual devices. Additionally, Chrome serves as an interface for accessing the Firebase console, making it easy to manage database contents, track analytics, and configure app settings directly from the browser.

In sum, Google Chrome is an indispensable tool in the TutorKey development pipeline, offering a versatile environment for debugging, performance testing, and ensuring that the app provides a consistent and optimized experience across platforms.

3.2.4 Visual Studio Code

Visual Studio Code (VS Code) is a comprehensive and efficient code editor that serves as the foundation for TutorKey's development environment. Designed with flexibility and ease of use in mind, VS Code is equipped with an array of tools that simplify coding, testing, and collaboration. Its lightweight footprint makes it fast and responsive, even when managing extensive codebases, which is crucial for a complex application like TutorKey. One of the standout features of VS Code is its support for a variety of languages and frameworks through plugins, particularly the Flutter and Dart plugins used in TutorKey. These plugins provide essential development tools, such as real-time syntax highlighting, which makes code more readable, and intelligent code completion, which reduces the potential for syntax errors and speeds up coding by suggesting commands and variables as developers type.

The integrated debugging environment in VS Code is invaluable for identifying and resolving issues within TutorKey's code. This feature allows developers to set breakpoints, inspect variables, and step through the code, which helps in isolating and fixing issues at every stage of the app's development. For TutorKey, this streamlined debugging process is critical for ensuring that both tutor and tutee interfaces function smoothly, especially since different components, like user management and session scheduling, rely on complex, interconnected functionalities. Through this process, developers can catch and resolve potential issues early on, contributing to a more stable and polished final product.

In addition to its robust coding and debugging capabilities, VS Code supports seamless collaboration through its built-in Git version control. This functionality allows TutorKey's development team to work concurrently on different aspects of the application. Team members can push code changes, create branches for experimental features, and merge updates from others without the risk of overwriting each other's work. This is particularly important for TutorKey, where regular updates, feature additions, and bug fixes are expected. Git integration in VS Code enables team members to view and manage the project's commit history directly

within the editor, making it easy to revert changes if needed and maintain a clear, organized development timeline.

Finally, VS Code's extensive extension ecosystem further enhances its adaptability for TutorKey. Extensions like ESLint, Prettier, and project-specific tools ensure that code adheres to consistent formatting and style guidelines, maintaining high code quality and readability. For the TutorKey project, these extensions help standardize coding practices across the team, making collaboration smoother and ensuring that code remains maintainable as the app scales. Additionally, VS Code's customizable workspace settings and task automation tools, such as task runners for building and testing the app, allow developers to tailor their workflow, increasing productivity and reducing manual steps. This level of customization and control makes VS Code a powerful, flexible, and highly efficient choice for developing TutorKey, enabling the team to manage the complexities of a cross-platform app while maintaining a usercentered approach to design and functionality.

3.2.5 Android Studio

Android Studio is a powerful integrated development environment (IDE) built specifically for Android app development and is compatible with a variety of development frameworks, including Flutter. As Google's official Android IDE, it provides a suite of features tailored for mobile app development, such as a robust code editor, layout editor, and emulator, all of which streamline the process of designing, coding, testing, and deploying Android applications. With built-in support for the Dart and Flutter plugins, Android Studio offers developers essential tools for creating efficient, performant apps. Key features include a visual layout editor, which allows developers to design and preview UI elements directly within the IDE, and advanced code editing tools, including syntax highlighting, intelligent code completion, and in-depth error detection. Android Studio also includes Gradle integration for dependency management, making it easy to organize and manage the libraries needed for app functionality.

One of Android Studio's standout features is its Android Emulator, which provides an accurate simulation of Android devices, enabling developers to test their apps on various Android versions and configurations without needing physical devices. This emulator can emulate a range of device types, screen sizes, and performance parameters, giving developers

insights into how the app might behave across different user environments. Additionally, the built-in Profiler tools allow developers to monitor CPU, memory, network, and battery usage, which helps in identifying performance bottlenecks and ensuring the app's efficiency and responsiveness.

For the TutorKey project, Android Studio is particularly beneficial due to its specialized testing and deployment capabilities. The Android Emulator allows the TutorKey development team to preview and test the app across a wide array of Android devices, ensuring that the user experience is consistent for both tutors and tutees regardless of their device model. Given that TutorKey relies on Firebase for data management and real-time updates, Android Studio's dependency management through Gradle is instrumental in organizing these libraries, simplifying updates, and ensuring smooth app interactions with Firebase services.

Additionally, Android Studio's debugging and profiling tools allow the team to optimize the app's performance on Android, enhancing response times and reducing lag during common tasks like session scheduling or profile management. These tools are critical for delivering a reliable experience, as any delays or issues in these areas could impact user satisfaction. The detailed performance insights provided by Android Studio enable the TutorKey team to address potential issues early, making it a valuable asset for delivering a high-quality, consistent experience to Android users. The ability to deploy directly from the IDE to the Google Play Store further streamlines the development cycle, enabling TutorKey to reach and update its audience efficiently.

3.2.6 Dart

Dart is a robust, client-optimized programming language developed by Google, and it's specifically designed to power high-quality applications on both mobile and web platforms. As the primary language for Flutter, Dart combines expressive syntax with a focus on speed and efficiency, making it highly suitable for cross-platform development. With a syntax that resembles JavaScript, C#, and Java, Dart is both accessible and powerful, allowing developers to quickly pick up the language and build responsive, scalable applications. One of Dart's key strengths is its ability to compile to native code, meaning it doesn't rely on a bridge between the app and the platform, as other cross-platform solutions do. This direct compilation ensures smoother performance and quicker load times, providing users with a seamless experience.

Dart's asynchronous programming capabilities are crucial for managing data in TutorKey, as the app frequently communicates with Firebase for real-time data updates. Dart's asynchronous support, built on futures and streams, allows TutorKey to handle complex interactions, such as fetching session information, updating profiles, or synchronizing schedules, without blocking the main UI thread. This means users experience smooth transitions and minimal delays, which are particularly important for TutorKey's time-sensitive functions like session scheduling and profile updates.

For TutorKey, Dart's null-safety feature is highly beneficial. Null safety allows developers to avoid null reference errors—common issues that lead to crashes and bugs—by specifying whether a variable can or cannot be null. This is particularly advantageous for an educational platform like TutorKey, where data integrity and stability are essential for user trust and satisfaction. The strong typing system in Dart, combined with null safety, reduces the likelihood of unexpected runtime errors, making the app more stable and secure. This reliability ensures that users—both tutors and tutees—can interact with the app confidently, knowing that their data is secure and that the app will perform consistently.

Dart's close integration with Flutter's reactive UI framework further enhances TutorKey's performance, allowing for effective state management and high responsiveness during user interactions. Since TutorKey involves dynamic features like dashboard updates, session management, and profile customization, Dart's efficient handling of UI states ensures that changes are reflected smoothly and instantly. This compatibility allows TutorKey's development team to create a polished, highly interactive user experience, enhancing user engagement and satisfaction across both Android and iOS platforms.

3.3 Dependencies

The Dependencies for this project includes:

1. Firebase Core
2. Firebase Auth
3. Cloud Firestore

3.3.1 Firebase Core

Firebase Core is an essential dependency for the integration of Firebase into the Flutter application. It acts as the bridge between the app and Firebase, initializing the necessary connections and configurations for other Firebase services to function correctly. Without Firebase Core, the app cannot communicate with Firebase services like Authentication, Firestore, or Firebase Storage.

Once Firebase Core is added to the app, it configures the app to interact with Firebase through APIs. This includes connecting the app to Firebase servers, handling authentication processes, managing databases, and ensuring that all Firebase services are correctly set up and functional. Firebase Core also provides error handling and diagnostic tools, which are vital for identifying any issues in communication between the app and Firebase.

3.3.2 Firebase Auth

Firebase Authentication is one of the most crucial dependencies in any app that requires user management, and TutorKey is no exception. Firebase Auth provides a secure and scalable way to handle user sign-ups, logins, and identity verification. The service supports multiple authentication methods, which adds flexibility for users to choose their preferred login mechanism. The methods include:

- **Email/Password Authentication:** Users can create accounts with their email and a secure password, ensuring their data is protected with standard encryption techniques.
- **Google Authentication:** This allows users to log in with their existing Google accounts, providing a quicker and more seamless login process. This is especially useful for users who prefer to use their Google credentials without needing to remember another password.
- **Phone Authentication:** Firebase Auth also supports phone number-based authentication, where users receive a One-Time Password (OTP) via SMS to verify their identity. This is a valuable option for users who may not have email accounts or prefer phone-based security.

Firebase Auth ensures that users' personal information is securely managed. It encrypts passwords and supports OAuth2.0 for secure connections. In TutorKey, this service is used to

authenticate both tutors and tutees, ensuring that each user has access only to the relevant parts of the app, such as their profile, sessions, and other personalized features.

3.3.3 Cloud Firestore

Cloud Firestore is the real-time NoSQL database service provided by Firebase for storing and syncing data across all users. In TutorKey, Firestore plays a vital role in storing all the app's dynamic data, such as user profiles, session bookings, and tutor availability.

Cloud Firestore is designed to handle a variety of complex data structures and ensures that data can be easily scaled as the app grows. It provides a flexible, document-based structure that organizes data into collections and documents, making it easy to model different types of data like user profiles, session details, and feedback.

One of the main advantages of Firestore is its real-time synchronization feature, which ensures that whenever data is updated—whether it's a tutor updating their profile or a tutee making a booking—the changes are reflected across all devices in real-time. This is especially important for an app like TutorKey, where users need to stay updated with the latest session availability or changes in tutor profiles.

Cloud Firestore also offers offline support, meaning that even if a user temporarily loses internet connection, they can still interact with the app, and their data will sync once the connection is restored. This feature improves the app's resilience and provides a smooth experience even in areas with unreliable internet access. Firestore also provides security rules that help ensure only authorized users can access specific data. For instance, session bookings can be restricted to the relevant tutor and tutee, and profiles can be secured based on user roles.

3.4 Hardware Requirements

The development of TutorKey is carried out on high-performance hardware to ensure smooth operation during coding, debugging, and testing. The hardware used during the development phase is carefully chosen to handle the demanding tasks associated with app development, such as running emulators, managing large codebases, and utilizing Firebase services.

A high-performance laptop or desktop with a minimum of 8 GB of RAM is used for the development process. This amount of RAM allows for efficient multitasking, enabling developers to switch between development environments, testing simulators, and Firebase services without performance lags. Adequate memory is essential to run multiple applications and tools simultaneously, which is common in software development, especially when using integrated development environments (IDEs) and testing platforms.

In terms of storage, at least 256 GB of SSD storage is utilized. Solid-state drives (SSDs) offer faster read and write speeds compared to traditional hard drives, which is critical during app builds, testing, and when managing project files and dependencies. This ensures that the development process remains smooth and efficient, with minimal delays caused by slow data access speeds. Additionally, having sufficient storage capacity allows developers to store large files, such as design assets and testing data, without worrying about running out of space.

Finally, graphics capabilities, whether integrated or dedicated, are considered optional but helpful. While the app's development doesn't require intensive graphics processing, having a capable graphics system can aid in UI development and the rendering of design elements. It ensures that the visual aspects of the app are rendered smoothly during testing and design iterations, particularly when working with high-resolution images or animations.

CHAPTER 4

RESULTS AND DISCUSSIONS

The results from the development and testing phases of the TutorKey app indicate a strong performance, with the app meeting most of its intended goals for functionality, user experience, and security. The integration of Firebase services, such as Firestore for real-time data synchronization provided a robust backend infrastructure that allowed seamless interactions between tutors and tutees. The app's core features, such as user authentication, session bookings, and profile management, were successfully implemented, and the real-time syncing ensured that data was up-to-date across all devices, improving user experience and engagement.

For tutors, the app allows for easy management of their profile, scheduling availability, and subject selection. Tutors can input their personal details, including name, phone number, address, qualifications, experience, and tuition fees, making the profile management feature comprehensive. The ability to schedule availability in real-time through specific time slots ensures that tutors can efficiently manage their time and respond to booking requests promptly. The subject section, where tutors can select specific subjects they are willing to teach, allows for easy filtering and selection by tutees, enhancing the match between available tutors and student needs. The session management section for tutors, including statuses such as pending, ongoing, unpaid, completed, and rejected, works smoothly, allowing tutors to track and manage their sessions effectively.

For tutees, the app offers an intuitive interface for browsing tutors based on subjects, viewing tutor availability, and requesting sessions. The real-time booking feature ensures that only available slots can be booked, and the ability to provide a location for the tuition adds flexibility for users. The session statuses for tutees—pending, ongoing, unpaid, completed, and rejected—provide a clear overview of their requested sessions, simplifying the process of managing bookings.

The integration of Firebase for real-time data synchronization ensures that the app remains up-to-date across all devices, allowing for seamless communication between tutors and tutees. Since the app operates only online, the real-time nature of Firebase ensures that no data

is lost, and changes are reflected immediately. However, the app's lack of profile image upload functionality was noted, as well as the absence of a rating and review system, which would enhance user engagement and feedback.

In conclusion, the TutorKey app has successfully met its development goals, with strong core functionalities such as profile management, scheduling, real-time session tracking, and secure data handling. The app is positioned for continuous improvement, with future updates focused on refining the user interface and adding new features based on user needs and feedback.

4.1 Splash Screen

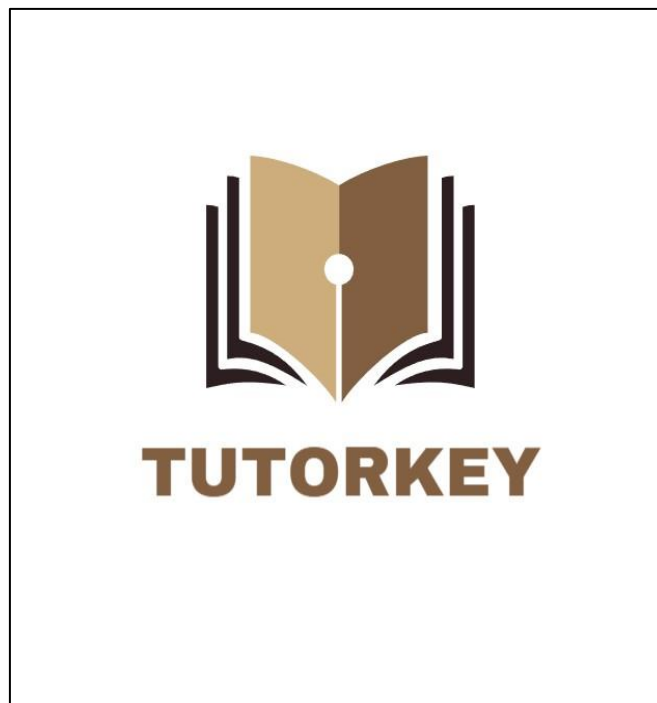


Figure 4.1 Splash Screen

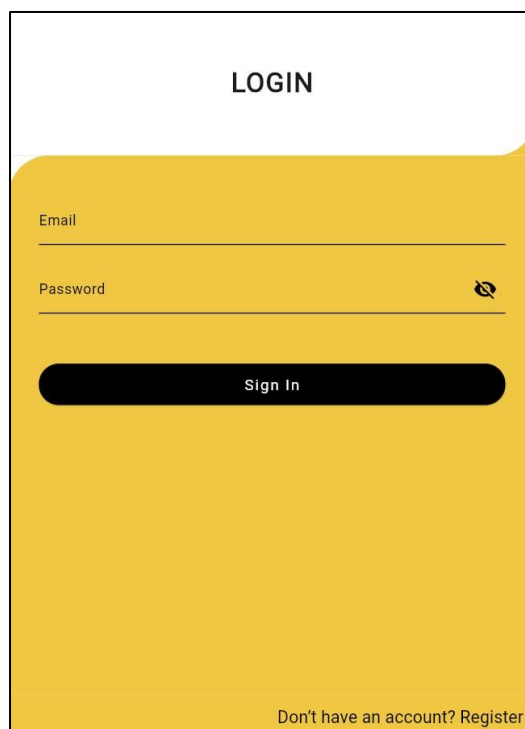
Figure 4.1 of the TutorKey app serves as the introductory screen when the app is launched. It appears briefly when the app starts up, providing users with a smooth transition into the main features of the app. The primary function of the splash screen is to showcase the app's logo and branding while the app initializes necessary resources, such as loading essential data and establishing connections to Firebase services.

The design of the splash screen is kept simple and visually appealing, featuring the

TutorKey logo with an engaging color scheme that aligns with the app's overall aesthetic. This screen sets the tone for the rest of the app and ensures that users have a seamless initial experience. While it is displayed briefly, it also provides a clear sense of the app's identity, reinforcing the brand's professionalism and user-centered approach.

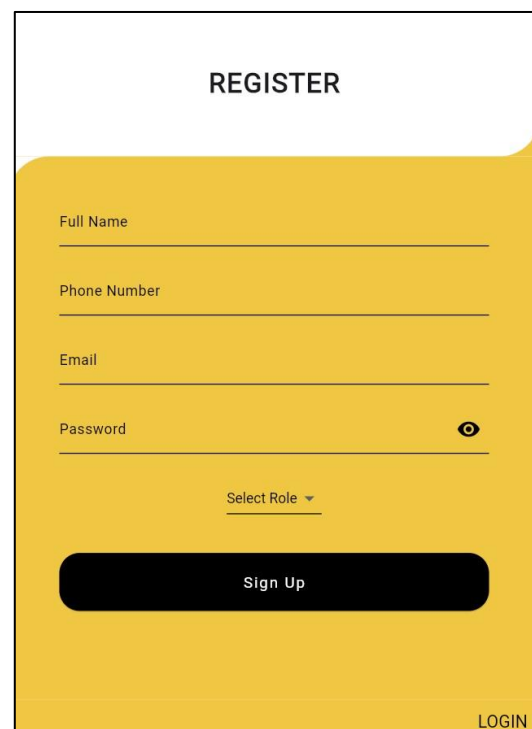
The splash screen has been optimized for quick loading and does not impact the overall performance of the app. This initial screen enhances the user experience by providing a brief moment of anticipation before the main dashboard appears, giving users time to settle into the app without feeling rushed.

4.2 Login and Sign up Page



The Login page features a white header with the word "LOGIN" in bold. Below the header is a large yellow rounded rectangle containing the login form. The form has two input fields: "Email" and "Password". The "Password" field includes a small eye icon for toggling visibility. Below the input fields is a black rounded button with the text "Sign In" in white. At the bottom of the yellow area, there is a link that says "Don't have an account? Register".

Figure 4.2(a) Login Page



The Register page features a white header with the word "REGISTER" in bold. Below the header is a large yellow rounded rectangle containing the registration form. The form has four input fields: "Full Name", "Phone Number", "Email", and "Password". The "Password" field includes a small eye icon for toggling visibility. Below the "Password" field is a dropdown menu labeled "Select Role". Below the dropdown menu is a black rounded button with the text "Sign Up" in white. At the bottom right of the yellow area, there is a link that says "LOGIN".

Figure 4.2(b) Sign up page

The Login and Sign-Up page is a crucial part of the TutorKey app, as it allows users to securely access their personalized dashboards. The page is designed to be user-friendly and straightforward, ensuring a smooth onboarding experience for both tutors and tutees.

Upon launching the app, users are presented with the login page as in Figure 4.2(a) and an option to register if they are new users. The user can login using the email and password credentials. The Sign-Up page (Figure 4.2(b)) requires users to provide essential details such as name, phone number, email, password and role selection (either tutor or tutee). This role

selection is an important feature, as it directs users to the appropriate dashboard based on their chosen role.

The design of the Login and Sign-Up page is clean and minimalistic, with clear labels and intuitive input fields. Error messages are displayed if users enter incorrect credentials, helping them quickly resolve any login issues. It plays a critical role in providing a personalized experience for both tutors and tutees, directing them to the right features based on their role.

4.3 Tutor Dashboard

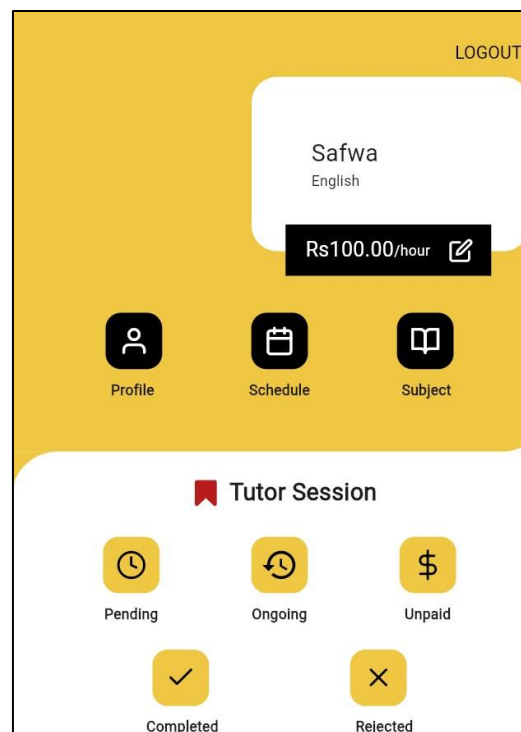
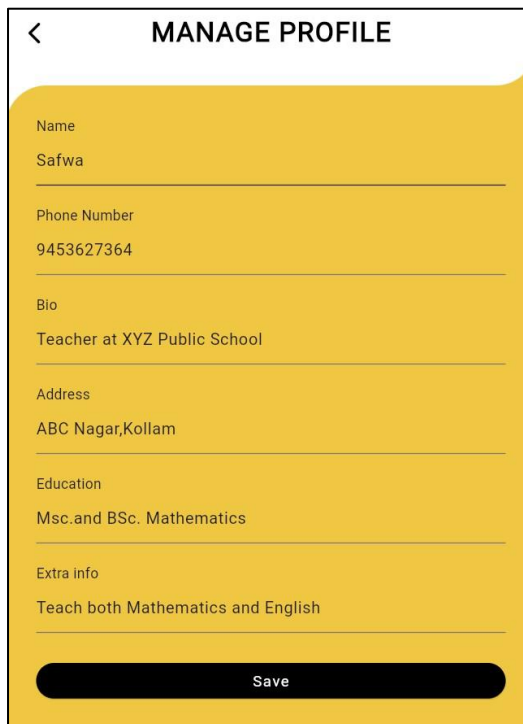


Figure 4.3 Tutor Dashboard

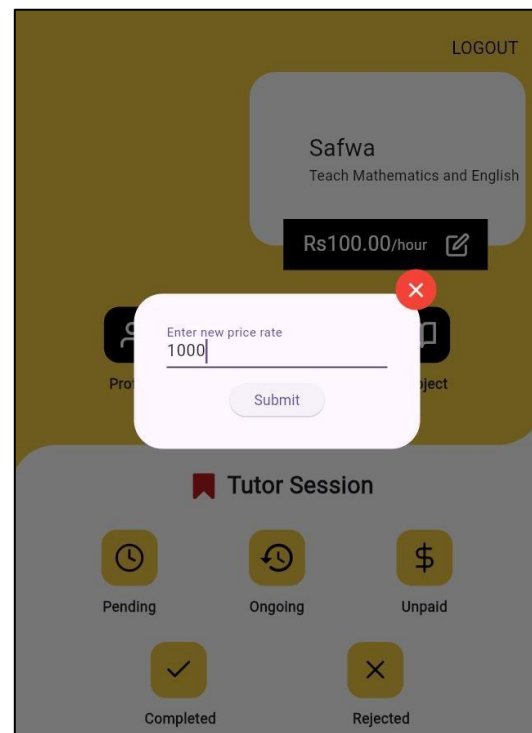
The Tutor Dashboard is the central hub for tutors, where they can manage their profiles, availability, subject selection, and track their tutoring sessions. It is designed with a clean, organized layout that allows tutors to easily navigate between different sections of the app. The dashboard is divided into two main sections: Profile and Scheduling and Tutor Sessions.

4.3.1 Profile and Scheduling Section



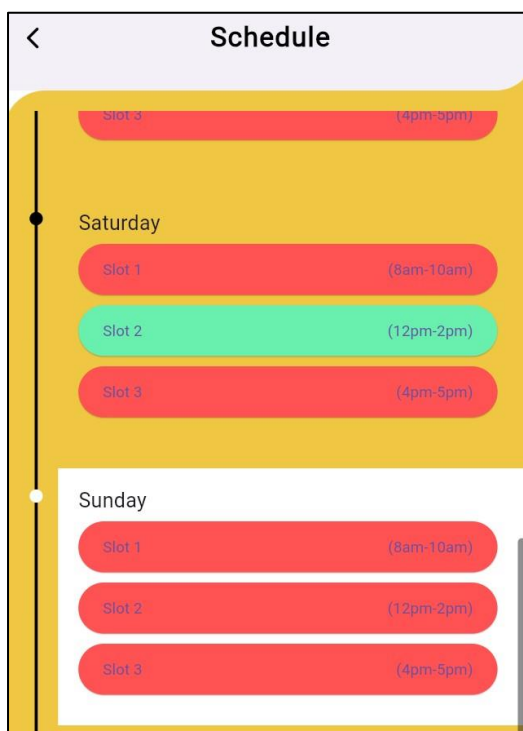
The 'MANAGE PROFILE' screen features a yellow background. It contains several input fields for user information: Name (Safwa), Phone Number (9453627364), Bio (Teacher at XYZ Public School), Address (ABC Nagar, Kollam), Education (Msc. and BSc. Mathematics), and Extra info (Teach both Mathematics and English). A black 'Save' button is at the bottom.

Figure 4.3.1(a) Manage Tutor Profile



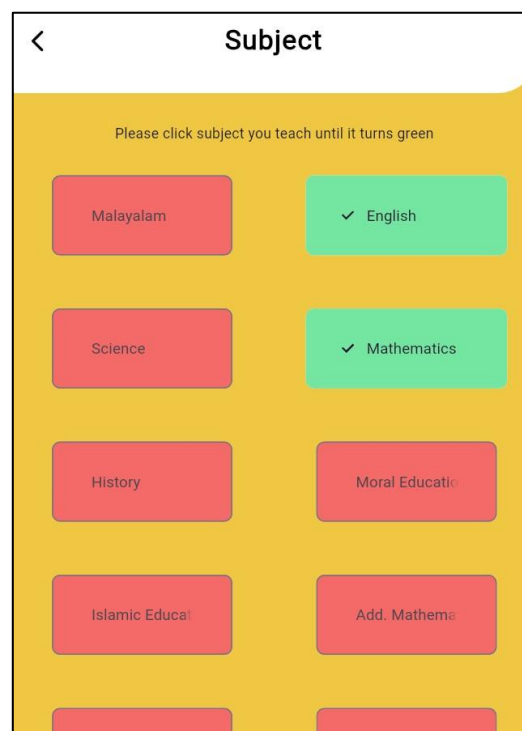
The 'Manage Tutor Fee' screen has a dark olive green background. It displays the tutor's name 'Safwa' and subject 'Teach Mathematics and English' with a current rate of 'Rs100.00/hour'. A modal dialog is open for 'Enter new price rate' with '1000' entered and a 'Submit' button. Below, a 'Tutor Session' section shows icons for Pending, Ongoing, Unpaid, Completed, and Rejected sessions. A 'LOGOUT' link is in the top right.

Figure 4.3.1(b) Manage Tutor Fee



The 'Schedule' screen shows a weekly calendar. Saturday has three slots: Slot 1 (8am-10am), Slot 2 (12pm-2pm), and Slot 3 (4pm-5pm). Sunday also has three slots: Slot 1 (8am-10am), Slot 2 (12pm-2pm), and Slot 3 (4pm-5pm). Slots are represented by colored rounded rectangles.

Figure 4.3.1(c) Schedule



The 'Subject' screen has a yellow background and instructs the user to 'Please click subject you teach until it turns green'. It displays a grid of subject buttons: Malayalam, English (checked), Science, Mathematics (checked), History, Moral Education, Islamic Education, and Add. Mathematics.

Figure 4.3.1(d) Subject

In the Profile and Scheduling section of the Tutor Dashboard, tutors can manage all aspects of their profile, including personal details, availability, tuition rates, and subject expertise. This section is divided into multiple areas, each catering to a specific aspect of the tutor's profile, ensuring that they can maintain accurate information and easily manage their teaching schedule. This comprehensive setup allows tutors to effectively present their qualifications and manage their availability for potential sessions.

In Figure 4.3.1(a), tutors can update essential personal information such as their name, phone number, address, qualifications, and teaching experience. Having up-to-date and accurate information helps tutors attract suitable tutees and ensures that communication between tutors and tutees is smooth and reliable. Additionally, tutors can include relevant qualifications and experience to showcase their expertise, building trust and credibility with tutees. This structured profile section gives tutors a complete view of their professional details and offers an easy way to edit or add information as needed, making profile updates both efficient and straightforward.

In the Profile Management section (Figure 4.3.1(b)), tutors also have the ability to set and update their tuition fee per hour. This feature allows tutors to adjust their hourly rate based on their qualifications, experience, and market demand. By providing a field for entering their tuition fee, tutors can keep their pricing transparent and competitive. Changes made to the tuition rate are automatically reflected in the profile, ensuring that potential tutees see accurate pricing when browsing tutor options. This flexibility empowers tutors to manage their rates independently, helping them align their fee with their level of expertise and the subjects they teach.

Tutors can schedule their availability for sessions in the Scheduling section (Figure 4.3.1(c)). This feature allows tutors to select specific time slots when they are available to teach, providing a straightforward way to manage their teaching hours and avoid scheduling conflicts. The slots are pre-configured, enabling tutors to simply select the times that fit their availability each day. This not only aids tutors in maintaining a consistent schedule but also allows tutees to book sessions only during these available slots, which helps ensure that sessions are efficiently managed in real time.

The Subject Section (Figure 4.3.1(d)) enables tutors to select the subjects they are capable of teaching, providing a clear view of their areas of expertise. Subjects are displayed

in a grid format, with each subject initially shown in a red grid box. When a tutor clicks on a subject to indicate that they are available to teach it, the grid turns green, signaling that the subject is active and available for tutees to book. This visual indication of selected subjects not only makes it easy for tutors to manage their subject offerings but also allows tutees to quickly identify which subjects a tutor is available to teach. This feature simplifies the matching process between tutors and tutees, ensuring that tutees have access to tutors specializing in their required subjects.

4.3.2 Tutor Sessions

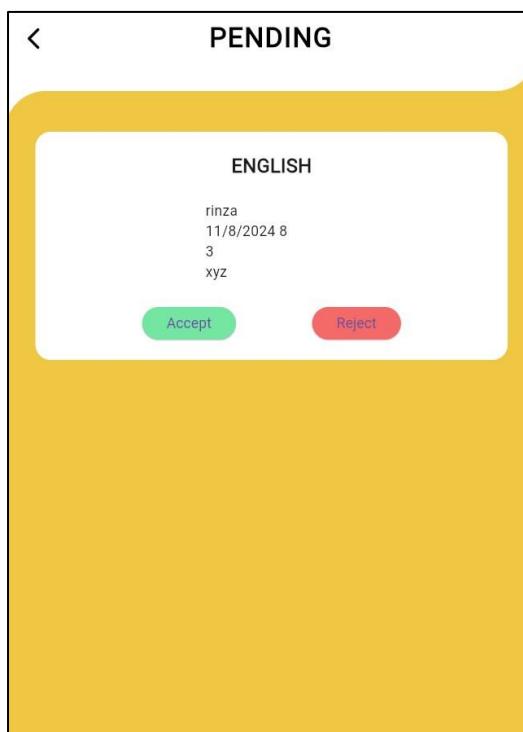


Figure 4.3.2(a) Pending Session

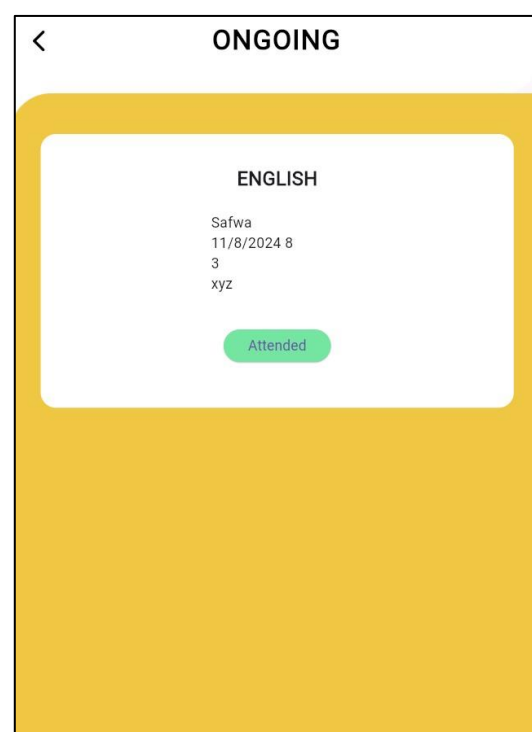


Figure 4.3.2(b) Ongoing Session

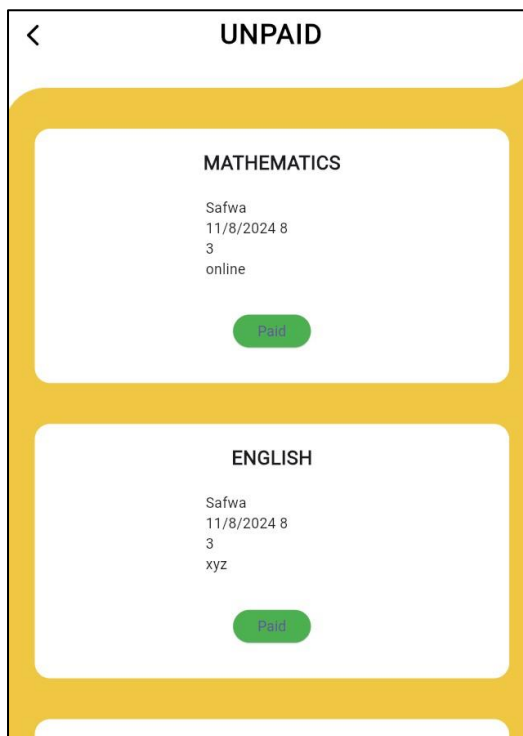


Figure 4.3.2(c) Unpaid Session

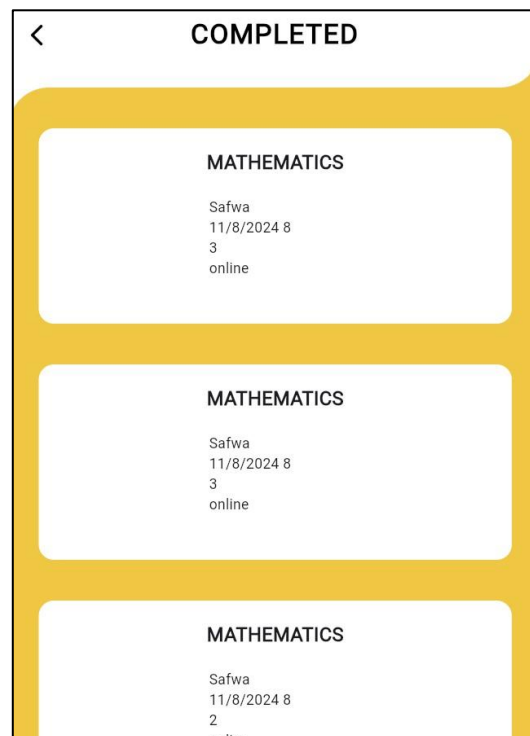


Figure 4.3.2(d) Completed Session

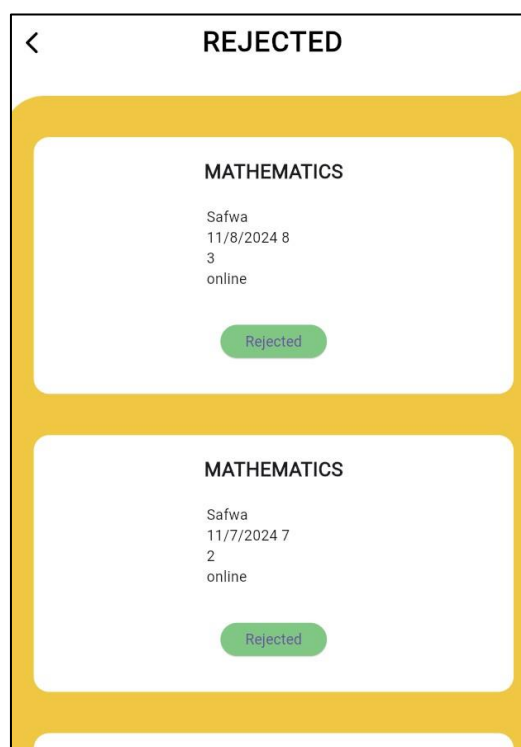


Figure 4.3.2(e) Rejected Session

The Tutor Session Section in the Tutor Dashboard provides a structured and intuitive layout for tutors to manage all aspects of their tutoring sessions, making it easy to track, review,

and update the status of each session. This section is organized into five main containers—Pending, Ongoing, Unpaid, Completed, and Rejected—each serving a specific purpose within the session management workflow. This streamlined approach helps tutors handle their sessions efficiently, ensuring that all tasks, from booking requests to payment confirmations, are easily accessible.

Pending Sessions (Figure 4.3.2(a)): The Pending container shows all new session requests sent by tutees. Tutors can review each request and choose to either accept or reject it based on their availability and preferences. This section is crucial for initial session management, allowing tutors to stay on top of incoming requests and make timely decisions. The clear distinction of pending sessions helps tutors prioritize new requests and ensure they are responsive to potential bookings, supporting positive engagement with tutees from the outset.

Ongoing Sessions (Figure 4.3.2(b)): Once a session request has been accepted, it moves to the Ongoing container. This section lists all sessions that are scheduled to be conducted or are in progress, providing a clear view of upcoming commitments. After each session is completed, the tutor can mark it as “Attended,” signifying that the session was successfully held. This status update is essential for tracking session completion and serves as a reference for both tutors and tutees. The Ongoing container allows tutors to maintain visibility over their active sessions and ensures that they have easy access to upcoming appointments and attendance tracking.

Unpaid Sessions (Figure 4.3.2(c)): The Unpaid container tracks all completed sessions for which payment is still pending. This feature provides a dedicated area for tutors to follow up on outstanding payments, ensuring that they are compensated for their time and efforts. Tutors have the option to mark a session as “Paid” once payment is received, automatically updating the session’s status and removing it from the Unpaid list. This process streamlines payment tracking and minimizes the likelihood of missed payments, providing both tutors and tutees with a transparent and reliable way to manage session fees.

Completed Sessions (Figure 4.3.2(d)): The Completed container houses a record of all sessions that have been fully conducted and, if necessary, marked as paid. This section serves as an archive of the tutor’s past sessions, offering a comprehensive history of completed tutoring engagements. The Completed container provides tutors with a quick reference for reviewing past sessions, making it easy to recall previous interactions with tutees. This

historical log of completed sessions also enables tutors to analyze their past work patterns and potentially identify trends in their tutoring schedule.

Rejected Sessions (Figure 4.3.2(e)): The Rejected container lists all sessions that the tutor has chosen not to accept. This section helps tutors keep track of declined requests, providing a record of sessions they were unable to accommodate. This record is useful in case tutors wish to follow up with tutees or reschedule at a later date, and it also allows them to manage their availability effectively by maintaining transparency with tutees regarding unaccepted bookings.

4.4 Tutee Dashboard

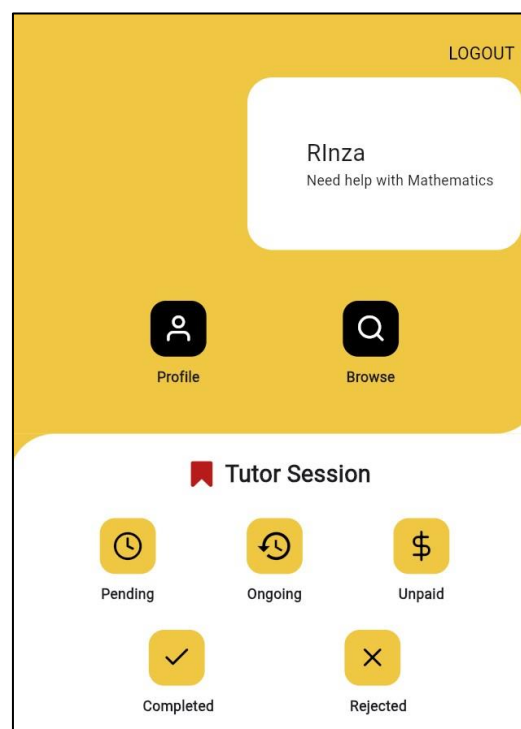


Figure 4.4 Tutee Dashboard

The Tutee Dashboard is the main interface for tutees, designed to facilitate an easy and streamlined experience for finding and booking tutors. The dashboard is divided into two main sections: Profile and Browse Tutors and Session Management.

The Tutee Dashboard is designed to be intuitive and easy to navigate, with clear sections and responsive elements that simplify the process of finding, booking, and managing sessions

with tutors. The real-time synchronization ensures that session statuses and tutor availability are always up-to-date, giving tutees an efficient way to manage their tutoring experience within the app.

4.4.1 Profile and Browse Section

< **MANAGE PROFILE**

Name
Rlnza

Phone Number
9847365443

Bio
Need help with Mathematics

Address
ABC Nagar, Kollam

Education
9th standard

Extra info
Student at LMN Public School

Save

Figure 4.4.1(a) Manage Tutee Profile

< **BROWSE**

Malayalam English

Science Mathematics

History Moral Education

Islamic Education Add. Mathematics

Figure 4.4.1(b) Browse Tutors

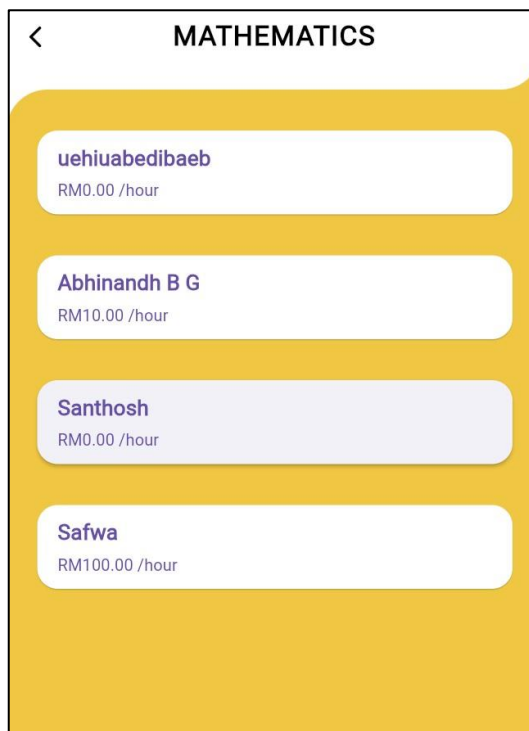


Figure 4.4.1(c) Tutor Selection

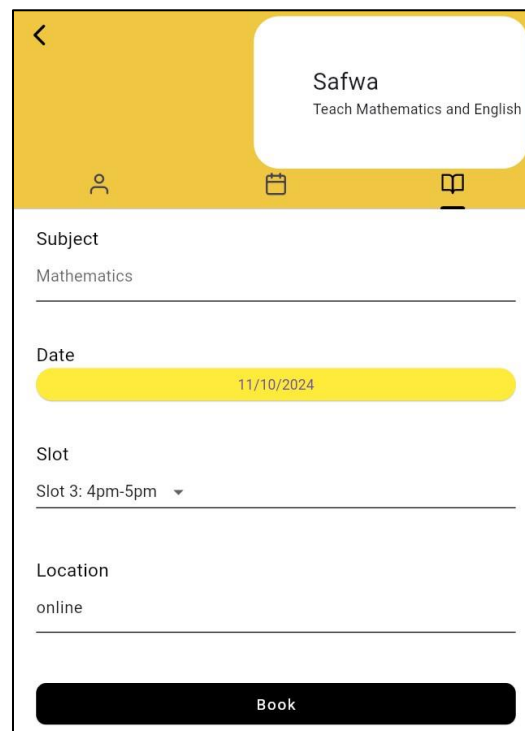


Figure 4.4.1(d) Session Booking

In the Profile and Browse Tutors section of the Tutee Dashboard, tutees are provided with tools to manage their profile and explore available tutors based on their tutoring needs.

The Profile Management feature allows tutees to personalize their account by adding or updating essential details, including name, phone number, address, and any additional information they wish to share (Figure 4.4.1(a)). Having accurate profile information benefits both parties, as tutors can access relevant details that may assist in arranging tutoring sessions.

The Browse Tutors functionality is a core part of the Tutee Dashboard, allowing tutees to find suitable tutors by filtering based on specific subjects (Figure 4.4.1(b)). When a tutee selects a subject, the app generates a list of tutors who are qualified and available to teach that subject (Figure 4.4.1(c)). Each tutor entry in the list displays essential information, including the tutor's name, hourly tuition fee, and subject expertise. This feature allows tutees to quickly compare tutors based on their experience, specialization, and fees, helping them make informed decisions about who might best meet their academic needs.

Once a tutee identifies a potential tutor, they can click on the tutor's profile to view a detailed profile page. This page provides an in-depth overview of the tutor's qualifications,

experience, and available time slots for the day. The availability section is displayed as specific time slots, indicating when the tutor is free to hold a session. This real-time availability feature ensures that tutees can book sessions only when tutors are ready to teach, avoiding scheduling conflicts.

When a tutee decides to book a session, they select an available time slot for the current day, as the app currently supports real-time, same-day booking only. In addition to selecting the time slot, the tutee can provide a specific location for the session in the input field, allowing for either in-person or virtual arrangements, depending on the tutor's preferences and capabilities. Upon completing these steps and confirming the booking, a booking request is sent directly to the selected tutor(Figure 4.4.1(d)).

This request appears in the tutor's Pending Session list within their dashboard, where they can either accept or reject the booking. Once the tutor accepts, the session moves forward in the tutee's dashboard under Ongoing Sessions, indicating that the booking has been confirmed and is awaiting completion.

4.4.2 Tutor Sessions

The Session Management section in the Tutee Dashboard provides tutees with a comprehensive overview of all their tutoring sessions, allowing them to easily track each booking's status. This section is organized into five distinct containers—Pending, Ongoing, Unpaid, Completed, and Rejected—each representing a different stage in the session lifecycle. This clear structure helps tutees manage their sessions, payments, and interactions with tutors seamlessly.

- **Pending:** The Pending container displays all tutoring session requests that the tutee has sent to tutors. When a tutee books a session by selecting an available time slot and submitting a request, it is placed in this section, where it awaits the tutor's response. The tutor has the option to either accept or reject the session request. This pending status helps the tutee keep track of all requests that are awaiting confirmation, reducing uncertainty and making it easy to follow up if needed.
- **Ongoing:** Once a tutor accepts a booking request, the session moves from Pending to the Ongoing container. This section shows all confirmed sessions that are scheduled but

have not yet been completed. Having an Ongoing section allows the tutee to clearly see which sessions are active and scheduled to take place soon. After each session is conducted, the tutor can mark it as "attended," which will update the status in both the tutor's and tutee's dashboards.

- **Unpaid:** The Unpaid container displays completed sessions for which payment has not yet been processed. This feature is crucial for tutees, as it serves as a reminder for any outstanding payments. Tutors are also aware of these unpaid sessions on their dashboard, allowing them to verify the payment status. Once the tutor marks the session as "Paid," the session is automatically removed from the Unpaid list, ensuring that both tutors and tutees can keep track of the payment status transparently.
- **Completed:** The Completed section contains all tutoring sessions that have been successfully conducted and, if applicable, paid for. Once a session is marked as attended and payment is confirmed, it moves into the Completed list. This section acts as a record of past sessions, enabling the tutee to review their learning history and access details about previous interactions with tutors. It is especially helpful for tutees who may want to rebook sessions with the same tutor based on past experiences.
- **Rejected:** The Rejected container lists all session requests that were declined by tutors. This section provides tutees with visibility into any requests that could not be fulfilled, either due to scheduling conflicts or other reasons on the tutor's end. Having a Rejected section ensures that tutees are fully aware of each booking request's outcome and can plan accordingly, whether by reaching out to other tutors or rescheduling with the same tutor if possible.

Each of these containers in the Session Management section helps the tutee navigate the entire booking process efficiently, from initial requests to completed sessions and payment confirmation. The layout is designed for easy viewing, with clear labels and organized status indicators. This structure enhances the user experience by keeping session information readily accessible, while also providing transparency and accountability for both tutors and tutees throughout each session's lifecycle.

CHAPTER 5

CONCLUSION

The conclusion of the TutorKey project emphasizes how the app effectively addresses the challenge of connecting tutors and tutees through the use of advanced technology, particularly Firebase's suite of services. By utilizing Firebase, the app can efficiently manage core functions such as user authentication, real-time data synchronization, and secure data storage. These technical capabilities are essential for ensuring that TutorKey runs smoothly and reliably. For instance, the real-time data synchronization feature ensures that all changes—whether it's booking a session or updating a tutor's availability—are instantly reflected across the app, providing users with a seamless experience.

The app's backend services are seamlessly integrated with the front-end user interface, which is intuitive and user-friendly. This ensures that both tutors and tutees can easily navigate the platform and utilize its features without being overwhelmed by complex tools or interfaces. Whether it's managing profiles, scheduling sessions, or monitoring session statuses, the app simplifies each of these processes, making the experience pleasant and efficient for all users.

A standout feature of TutorKey is its dual-dashboard system, which tailors the experience for both tutors and tutees. For tutors, the dashboard allows them to easily manage their hourly rates, session schedules, and track the real-time status of each session. This minimizes the time spent on administrative tasks, which in turn lets tutors focus more on their teaching. By organizing information into clear categories like Pending, Ongoing, Unpaid, Completed, and Rejected sessions, both tutors and tutees are kept up to date on the status of every interaction, ensuring transparency and accountability in the tutoring process.

For tutees, the platform is designed to simplify the process of finding tutors, booking sessions, and tracking the progress of those sessions. The clean design ensures that users don't feel overwhelmed by technicalities and can focus on the core goal—learning. This dual-focus approach makes TutorKey not just a tool for tutors but a comprehensive, intuitive platform for tutees as well.

Although TutorKey offers a comprehensive solution, the conclusion also acknowledges areas where the app can evolve further. First, the introduction of a rating and review system would allow tutees to provide feedback on their tutoring experience, helping others make more

informed decisions when selecting a tutor. This system could also be a valuable tool for tutors, who can showcase their reviews to attract new clients.

Second, expanding the session scheduling options would accommodate a wider variety of user needs, including those who operate in different time zones or have irregular schedules. By allowing more flexibility in how users schedule sessions, TutorKey could become even more accessible and attractive to a global audience.

Third, adding an in-app messaging feature would create an integrated communication channel for tutors and tutees. Currently, users may need to rely on external communication tools, but in-app messaging would streamline interactions, making it easier to confirm session details, clarify doubts, or provide feedback without leaving the platform. These improvements would enhance the overall functionality and appeal of the app.

The report highlights the crucial role that Firebase's cloud-based technology plays in powering TutorKey. Firebase provides the necessary infrastructure to handle real-time data synchronization, user authentication, and secure data storage without requiring the app's developers to manage complex server-side infrastructure. This is particularly important for real-time applications like TutorKey, where the timely exchange of information—such as session updates or scheduling changes—needs to occur without delays.

Firebase's scalability is another critical feature that supports the app's growth. As more users join the platform, Firebase can handle the increased load without compromising performance, ensuring a smooth and reliable experience for all users. This allows TutorKey to grow alongside its user base and incorporate new features as needed, without the fear of outgrowing its infrastructure.

The TutorKey app exemplifies the power of cloud-based technology in creating a reliable, scalable, and user-friendly platform for tutoring. The use of Firebase as the backend infrastructure ensures that the app can deliver real-time, secure, and scalable solutions for both tutors and tutees. The dual-dashboard system, real-time session tracking, and user-centric design set the app apart as an effective tool for managing tutoring sessions. Furthermore, by identifying areas for potential enhancement—such as a rating system, expanded scheduling options, and in-app messaging—the report outlines the app's pathway for continued improvement.

As TutorKey evolves, it has the potential to become an essential tool in the educational landscape, offering greater convenience, structure, and support for both tutors and tutees. The app's adaptability and scalability ensure that it can meet the demands of the future and remain a valuable resource in an increasingly digital world.

CHAPTER 6

FUTURE ENHANCEMENTS

In future iterations, TutorKey can be enhanced with a range of additional features aimed at improving the user experience, expanding functionality, and meeting evolving user needs.

Some key areas for potential enhancement include:

- **Rating and Review System:** Introducing a rating and review feature would allow tutees to provide feedback on tutoring sessions, helping tutors understand their strengths and areas for improvement. This feature would also enable tutees to make informed choices when selecting a tutor by viewing previous feedback from other students, enhancing transparency and trust within the platform.
- **Advanced Scheduling Options:** Expanding the scheduling system to include options for multi-day or recurring sessions could accommodate users who require long-term tutoring arrangements. This improvement would provide more flexibility for both tutors and tutees, making it easier to schedule consistent sessions that align with users' longterm learning goals.
- **In-App Messaging System:** Adding a real-time messaging feature would enable tutors and tutees to communicate directly within the app, streamlining coordination for session details, last-minute changes, and general inquiries. This feature could replace the need for external communication channels, making it more convenient for users to stay connected within the TutorKey app.
- **Offline Functionality:** Enabling limited offline functionality would allow users to view essential information, such as profiles, schedules, or past sessions, even when internet connectivity is intermittent. Once the connection is restored, any actions taken offline could be synced automatically, improving reliability and user satisfaction in regions with unstable network access.
- **Enhanced Tutor Verification Process:** Implementing a tutor verification process, where qualifications and credentials are verified by an external source, could increase the credibility of the platform and give tutees greater confidence in their tutor choices. This could be achieved through document uploads or linking with trusted verification services.

- **Push Notifications and Reminders:** Integrating a notification system would allow tutors and tutees to receive real-time alerts for important updates, such as new session requests, booking confirmations, payment reminders, and upcoming sessions. This feature would enhance engagement by keeping users informed and minimizing missed appointments.
- **Expanded Payment Tracking and Invoicing:** Adding functionality for detailed payment tracking, invoicing, and potentially integrating third-party payment processing options would make TutorKey more convenient and professional for tutors. This would support secure transactions and give tutors and tutees better visibility over their financial interactions within the app.
- **Enhanced Search and Filtering Options:** Adding advanced search and filtering options for tutees, such as filtering by tutor rating, availability, subject specialization, or hourly rate, would streamline the process of finding the right tutor. This feature would save time and make it easier for users to find tutors that precisely match their preferences and needs.
- **Session Recording and Note-Taking Feature:** Incorporating a session recording feature (if privacy policies allow) or a shared note-taking section would enable tutors and tutees to review session content post-meeting. For virtual or online sessions, session recording could allow tutees to revisit lesson material, and shared notes would serve as a study resource, improving retention and reinforcing learning.
- **Resource Sharing and Learning Materials:** Adding a section where tutors can upload resources like PDF worksheets, slides, or reading materials for their tutees could add significant value. This feature would make TutorKey not only a booking tool but also a resource hub, facilitating ongoing learning outside of tutoring sessions.

These future enhancements would further solidify TutorKey as a comprehensive tutoring platform, improving usability, functionality, and trustworthiness. By addressing these potential areas for improvement, TutorKey can continue to grow and adapt, meeting the demands of both tutors and tutees while positioning itself as a robust tool in the educational technology space.

REFERENCES

- [1] **Gonzales, Alexandria & Beringuela, Kristine & Rudio, Danice & Cantalejo, Jan & Bautista, Reyce & Olipas, Cris Norman & Villegas, Andrew.** (2023). Project Learn: The Development and Assessment of a Cross-Platform Tutor Finder. *European Journal of Theoretical and Applied Sciences*. 1(4). 15-27.
- [2] **Prince Chukwuneme Enwereji , Annelien van Rooyen and Alet Terblanche.** (2023). Exploring Students' Perceptions on Effective Online Tutoring at a Distance Education Institution. *The Electronic Journal of e-Learning*. 21(04). 366-381.
- [3] **Naim Shaikh, Kishori Kasat, Mahesh Shinde.** (2021). Trust among faculty and students as an essential element of Smart Education System. *Journal of Contemporary Issues in Business and Government*. 27(03). 1568-1575.
- [4] **Chougale, Pankaj & Yadav, Vaibhav & Gaikwad, Anil & Student, Bharati & Vidyapeeth,.** (2022).Firebase-Overview and Usage. *Journal of Engineering and Technology Management*. 03(12). 2582-5208.