ACKNOWLEDGEMENT

**In performing my project**, I had to take the help and guidelines of some respected persons, who deserve my greatest gratitude. The completion of this project gives me much pleasure. I completed my project using REACT NATIVE, and I would like to show my gratitude to **Prof. Shanthi Bala**, for giving me a good guideline for the project throughout numerous consultations. I would also like to express my deepest gratitude to all those who have directly and indirectly guided me in the completion of this project.

In addition, a thank you to **Dr. G. KrishnaPriya (Faculty Advisor)**, who constantly supported us in our work & and whose passion for “underlying structure” had lasting effects.

Many people, especially my class members, have made valuable comments and suggestions on this proposal which inspired me to improve my project. Last but not least I am grateful to all the faculty members of dept. of Computer Application of Pondicherry University for their support.

ABSTRACT

HOMELEARNER is a cutting-edge mobile application designed to transform the tutoring industry by effectively connecting students, parents, and teachers. As the demand for convenient and effective tutoring solutions increases, traditional methods often fail to keep up. HOMELEARNER addresses this need by providing a user-friendly platform optimized for mobile devices. Users can effortlessly create and post tutor advertisements, explore a vast pool of tutors using smart search filters, and experience interactive sample teaching sessions. The seamless mobile booking interface, complete with calendar integration and instant confirmation notifications, ensures a smooth and efficient booking process.

Beyond its core functionalities, HOMELEARNER fosters a sense of community and inclusivity. It offers free tutoring sessions for students in need and allows experts to register as volunteer tutors, contributing their time for free video sessions. The app also includes features such as scheduling and calendar integration, background checks for tutor verification, and enhanced community support. By integrating these comprehensive features, HOMELEARNER creates an accessible, efficient, and supportive environment for all users, revolutionizing the way tutoring services are accessed and delivered.

TABLE OF CONTENTS

ACKNOWLEDGEMENT…………………………………………………………….1

ABSTRACT……………………………………………………………………..……2

1. INTRODUCTION………………………………………………………………5-6
   1. ABOUT THE PROJECT……………………………………………………...5
   2. CHARACTERISTICS………………………………………………………...6
2. PROBLEM DEFINITION AND FEASIBILITY ANALYSIS…………………7-11

2.1 PROBLEM DEFINITION……………………………………………………7

2.2 EXISTING SYSTEM…………………………………………………………8

2.3 PROPOSED SYSTEM………………………………………………………..8

2.4 FEASIBILITY STUDY……………………………………………………....9

2.4.1 TECHNICAL FEASIBILITY………………………………………….10

2.4.2 OPERATIONAL FEASIBILITY……………………………………....10

2.4.3 ECONOMIC FEASIBILITY…………………………………………...11

3. SOFTWARE REQUIREMENT SPECIFICATION………………………….....12-14

* 1. HARDWARE REQUIREMENT……………………………………………..12
  2. SOFTWARE REQUIREMENT………………………………………………13

4. SYSTEM DESIGN………………...……………………………………………14-26

* 1. MODULE DESCRIPTION…………………………………………………...14
  2. TABLE DESIGN……………………………………………………………...20
  3. ACTIVITY WORKFLOW……………..……………………………………..23
  4. UML DIAGRAM…………………………………………………….……….25

5. IMPLEMENTATION…………………..………………………………………27-29

5.1 WELCOME SCREEN…………………………………………………………27

5.2 SIGN IN/SIGN UP SCREEN………………………………………………….27

5.3 HOME SCREEN………………………………………………………………28

5.4 ADD POST FORM……………………………………………………………28

6. SYSTEM TESTING………………………..…………………………………...29-32

6.1 UNIT TESTING………………………………………………………………30

6.2 VALIDATION TESTING…………………………………………………….31

6.3 FUNCTIONAL TESTING…………………………………………………....32

7. CONCLUSION……………………………………………………………………..33

APPENDIX I: SCREENSHOTS………………….…………………………...…..34-39

BIBLIOGRAPHY……………………………………………………...……………..40

CHAPTER: 1

INTRODUCTION

* 1. ABOUT THE PROJECT

The HOMELEARNER is a cutting-edge mobile application designed to revolutionize the way students, parents, and teachers connect with qualified tutors. In the current educational landscape, the process of finding the right tutor can be arduous, relying on limited word-of-mouth recommendations or offline advertisements. Teachers, on the other hand, struggle to reach their target audience and effectively showcase their expertise. The proposed system aims to alleviate these challenges by offering a comprehensive online platform with features catering to both students/parents and teachers.

Users can register, post advertisements with specific tutor requirements, utilize advanced search filters, and assess potential tutors through sample teaching sessions. Teachers, in turn, can create profiles, post service advertisements, share sample teaching sessions, and communicate with interested students and parents. With the Personal Tutor Finding System, finding the perfect tutor becomes a hassle-free and efficient experience for all involved parties.

* 1. CHARACTERISTIC OF HOME-LEARNER

Advanced Search Capabilities: HOMELEARNER provides robust searching facilities allowing users to search for tutors based on various factors such as category, subject, exam preparation, and location. This ensures a tailored and efficient search experience.

Tutor Profile Showcase: The platform allows teachers to create detailed profiles showcasing their qualifications, expertise, and services. This helps students and parents make informed decisions based on the tutors' credentials and offerings.

User Management: HOMELEARNER efficiently manages information for both students and teachers. This includes registration details, personal information, and user-specific preferences and requirements.

Real User Feedback: The system incorporates real user feedback, enabling students and parents to read reviews and ratings of tutors. This feature enhances trust and helps users select the best possible match for their needs.

Sample Teaching Sessions: The application manages and provides access to recorded sample teaching sessions. This allows students and parents to preview a tutor's teaching style and effectiveness before committing.

Live One-to-One Sessions: HOMELEARNER supports one-to-one live tutoring sessions, facilitating real-time interaction and personalized learning experiences between students and tutors.

Tutor Availability Posting: Teachers can post their availability and specific time slots when they are free to conduct sessions. This helps in better scheduling and ensures that students can book sessions according to the tutor's availability.

CHAPTER: 2

PROBLEM DEFINITION AND FEASIBILITY ANALYSIS

* 1. PROBLEM DEFINITION

In today's fast-paced world, the need for accessible and efficient tutoring solutions is increasing. However, traditional methods of finding and connecting with tutors often fall short, leading to challenges for students, parents, and tutors alike.

CHALLENGES

Traditional tutoring methods face several significant challenges that limit their accessibility and convenience. One major issue is the requirement for physical presence and pre-scheduled sessions, which makes traditional tutoring less flexible for students with busy schedules or those living in remote locations. This rigidity in scheduling can prevent students from getting the help they need at the times they need it most, thereby hindering their academic progress and personal growth.

In addition to accessibility issues, the process of matching students with the right tutor is often inefficient. Finding a tutor who meets specific needs, such as subject expertise, teaching style, and availability, can be time-consuming and frustrating. Without a streamlined system, students and parents may rely on word-of-mouth recommendations or offline advertisements, which are not always reliable or comprehensive.

Moreover, traditional tutoring systems suffer from a lack of transparency and trust. Ensuring the credibility and qualifications of tutors is difficult without a standardized system for background checks and credential verification. The booking process itself is also cumbersome, typically involving back-and-forth communication that can lead to delays and miscommunications. Furthermore, traditional systems do not readily provide opportunities for free or volunteer tutoring, which could benefit underprivileged students. Finally, students and parents often have limited means to assess a tutor's teaching methods and effectiveness before committing to sessions, resulting in potential mismatches and dissatisfaction.

* 1. EXISTING SYSTEM

The current tutoring landscape relies heavily on traditional methods that are often inconvenient and inefficient. Students and parents face challenges in finding suitable tutors due to limited accessibility, lack of standardized credential verification, and cumbersome booking processes. Additionally, there is minimal opportunity for interactive previews of tutor capabilities, making it difficult to assess teaching quality before committing to sessions. Existing systems also do not adequately support community-based, free tutoring initiatives for underprivileged students. Overall, traditional tutoring methods fall short of providing the flexibility, transparency, and ease of use needed in today's fast-paced world.

* 1. PROPOSED SYSTEM

HOMELEARNER is a mobile application designed to enhance the tutoring experience through several innovative features. It offers streamlined advertisement posting, allowing users to create ads easily with multimedia support. The app includes smart search filters, enabling efficient tutor searches by subject, location, availability, and rating. Users can preview tutors' teaching styles through interactive sample sessions. The seamless booking experience allows for simple mobile bookings with calendar integration. HOMELEARNER supports the community by offering free tutoring sessions for students in need.

Tutors can set their availability, which integrates with Google Calendar, ensuring smooth scheduling. The app also performs background checks to verify tutors' credentials and backgrounds. Additionally, expert volunteers can register to offer free tutoring sessions. Overall, this system aims to provide a user-friendly, efficient, and reliable platform for students, parents, and tutors.

* 1. FEASIBILITY STUDY
     1. TECHNICAL FEASIBILITY

Technology Stack:

Front-End: React Native, Styled Components

Feasibility: High. Supports cross-platform development for iOS and Android with reusable code.

Back-End: Node.js, Express.js, MongoDB

Feasibility: High. Suitable for scalable, high-performance back-end services and flexible data management.

Authentication: Firebase Authentication

Feasibility: High. Provides secure and easy-to-implement user authentication.

Cloud Services: Firebase

Feasibility: High. Offers scalable backend services, real-time databases, storage, and hosting with integrated security.

Version Control: Git, GitHub

Feasibility: High. Facilitates collaborative development and efficient code management.

The HOMELEARNER project is technically feasible with the selected technology stack, including Firebase for backend services, supporting robust development and scalable infrastructure. Each feature aligns well with current industry standards, making it a promising solution for modern tutoring needs.

* + 1. OPERATIONAL FEASIBILITY

User Engagement and Adoption:

Target Audience: Students, parents, tutors, and expert volunteers.

User Experience: Intuitive mobile interface for easy access and navigation.

Marketing: Streamlined advertisement posting and community support features.

Resource Requirements:

Development Team: Small to medium-sized team proficient in React Native, Node.js, and Firebase.

Infrastructure: Firebase for scalable backend services, reducing infrastructure needs.

Support and Maintenance: Customer support system and regular updates for security and performance.

Operational Challenges:

Scalability: Firebase infrastructure handles increased loads; performance optimization is ongoing.

Security: Robust measures in Firebase and background checks ensure data protection.

Compliance: Adherence to educational and privacy regulations in different regions.

Community and Support:

Building Community: Engagement features and volunteer participation foster a supportive community.

Feedback and Improvement: Continuous user feedback collection and agile development for iterative improvements.

* + 1. ECONOMICAL FEASIBILITY

Cost-Benefit Analysis:

Initial Investment: Moderate investment is required for development, infrastructure setup, and marketing.

Revenue Streams: Potential revenue from tutor advertisement fees, premium features, and partnerships.

Cost Reduction: Utilizing Firebase for backend services reduces infrastructure and maintenance costs.

Return on Investment: Expected to achieve profitability within a reasonable timeframe due to scalable business model.

Cost Savings:

Infrastructure: Firebase reduces the need for extensive in-house infrastructure, saving on hardware and maintenance costs.

Development: Leveraging open-source technologies like React Native and Node.js minimizes development costs.

Operations: Streamlined processes and automation reduce operational overheads.

Revenue Generation:

Advertisement Fees: Revenue from tutors paying for advertisement space on the platform.

Premium Features: Additional revenue from premium features such as advanced search filters or priority listing for tutors.

Partnerships: Potential revenue through partnerships with educational institutions or corporate sponsors.

CHAPTER: 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 HARDWARE REQUIREMENT

Server-Side Hardware:

Processor: Modern multi-core processor for efficient handling of backend operations.

RAM: Sufficient RAM for running Node.js, Express.js, and MongoDB concurrently.

Storage: Adequate storage space for database management and file storage.

Networking: Stable internet connectivity with high bandwidth for handling multiple concurrent connections.

Client-Side Hardware:

Smartphones/Tablets: Compatible with iOS and Android devices.

Processor: Decent processing power for the smooth running of React Native applications.

RAM: Sufficient memory for seamless multitasking and app responsiveness.

Storage: Adequate storage space for app installation and caching.

Additional Hardware:

Development Machines: Computers or laptops with necessary software development tools for frontend and backend development.

Testing Devices: Various smartphones and tablets for testing compatibility and performance across different devices and screen sizes.

3.2 SOFTWARE REQUIREMENT

Development Environment:

Code Editor: Any modern code editor such as Visual Studio Code, Sublime Text, or Atom for writing code.

Version Control: Git for version control management, with repositories hosted on platforms like GitHub.

Node.js and npm: Latest stable versions of Node.js and npm for backend development.

React Native CLI: Command Line Interface for React Native development.

Firebase CLI: Command Line Interface for Firebase deployment and management.

Backend Technologies:

Node.js: Runtime environment for executing JavaScript code on the server side.

Express.js: Web application framework for building APIs and handling HTTP requests.

MongoDB: NoSQL database for storing application data, requiring installation and configuration.

Frontend Technologies:

React Native: JavaScript framework for building cross-platform mobile applications.

Styled Components: Library for styling React components with CSS-in-JS approach.

Firebase SDK: Software Development Kit for integrating Firebase services such as Authentication and Realtime Database.

Additional Tools:

Firebase Authentication: Service for user authentication, requiring configuration and setup.

Firebase Realtime Database: Cloud-hosted NoSQL database for storing real-time data, used for chat features and real-time updates.

Firebase Cloud Storage: Cloud-based storage for storing user-uploaded files such as profile pictures.

Testing and Deployment:

Unit Testing Framework: Jest or Mocha for writing and executing unit tests.

Continuous Integration: Integration with CI/CD platforms like CircleCI or GitHub Actions for automated testing and deployment.

CHAPTER: 4

SYSTEM DESIGN

* 1. MODULE DESCRIPTION:
     1. WELCOME PAGE:

Overview:

The Welcome Page module serves as the introductory interface for users when they first launch the application. It provides an onboarding experience through a series of tutorial screens that highlight the key features and benefits of the application. This module aims to familiarize users with the application's capabilities and guide them through the initial steps of getting started.

Functionality:

Onboarding Screens: Displays a sequence of three tutorial screens to introduce users to the application.

Screen 1: "Learn Anywhere, Anytime" – Highlights the accessibility of educational materials from home.

Screen 2: "Personalized Learning" – Emphasizes the tailored learning experiences based on individual needs.

Screen 3: "Discover Your Need" – Focuses on personalized educational pathways to meet learning objectives.

Navigation Controls: Provides buttons for navigating through the onboarding screens.

Skip Button: This allows users to skip the onboarding process and go directly to the main screen.

Next Button: Moves to the next onboarding screen or navigates to the Sign-In screen if on the last screen.

Pagination Indicators: Displays dots indicating the user's current position in the onboarding sequence.

Inputs and Outputs:

Inputs: User interactions with the Skip and Next buttons.

Outputs: Navigation actions to either the main screen or the Sign-In screen.

Dependencies:

Internal: This module is connected to the Sign-In module to facilitate user authentication after the onboarding process.

External: Utilizes react-native components and @react-navigation/stack for navigation.

Design and Architecture:

Design Patterns: Implements a simple linear onboarding flow using React hooks (useState) and references (createRef).

Architectural Diagram: (Include a simple diagram if needed)

Implementation Details:

Technologies Used: Developed using React Native with core components such as View, Text, Image, TouchableOpacity, ScrollView, and Dimensions.

Code Structure:

WelcomeScreen.js: Contains the main logic and layout for the Welcome Page.

styles.js: (Inline in this case) Defines the styling for various components in the Welcome Page.

Challenges and Solutions:

Challenge: Ensuring smooth navigation and user experience across different screen sizes.

Solution: Utilized Dimensions for responsive design and ScrollView for seamless horizontal scrolling.

Testing and Validation:

Testing Methods: Manual testing for navigation flows, responsiveness, and visual consistency.

Validation Results: All navigation paths and UI elements functioned correctly during testing.

Future Enhancements:

Planned Improvements: Add animations and transitions between onboarding screens for a more engaging user experience.

Scalability and Maintenance: Designed to easily add more onboarding screens or update existing ones without major refactoring.

This module plays a crucial role in providing a welcoming and informative first impression to new users, guiding them smoothly into the main functionality of the application.

* + 1. LOGIN PAGE:

Overview:

The Login Page module is responsible for managing user authentication in the application. It provides an interface for users to sign in or sign up using their email and password. Additionally, it handles user authentication states and navigates authenticated users to the home screen.

Functionality:

Sign In and Sign Up: Allows users to either sign in if they have an existing account or sign up to create a new account.

Authentication State Management: Tracks the authentication state of users, displaying appropriate screens based on whether a user is authenticated.

Navigation: Directs authenticated users to the home screen and provides a logout function.

Inputs and Outputs:

Inputs: User email and password.

Outputs: Authentication state changes (user logged in or logged out), navigation to home screen.

Dependencies:

Internal: This module interacts with the navigation system to switch between different screens.

External: Utilizes Firebase for authentication services.

Design and Architecture:

Design Patterns: Utilizes React hooks (useState, useEffect) for state management and Firebase for authentication.

Architectural Diagram: (Include a simple diagram if needed)

Implementation Details:

Technologies Used: Developed using React Native and Firebase. Key libraries include @firebase/app for Firebase initialization and @firebase/auth for authentication.

Code Structure:

AuthScreen.js: Contains the sign-in and sign-up form along with associated logic.

AuthenticatedScreen.js: Displays user details and navigation options for authenticated users.

App.js: Main component managing state and rendering appropriate screens based on authentication status.

Challenges and Solutions:

Challenge: Handling different authentication scenarios (sign in, sign up, logout) within a single module.

Solution: Implemented conditional rendering and state management to handle different authentication scenarios smoothly.

Testing and Validation:

Testing Methods: Manual testing of user interactions, including sign-in, sign-up, and logout functionalities.

Validation Results: Verified that all authentication flows work correctly and that appropriate screens are displayed based on the user's authentication state.

Future Enhancements:

Planned Improvements: Add error handling and user feedback for authentication failures.

Scalability and Maintenance: Designed to easily incorporate additional authentication methods (e.g., OAuth) and improve user experience with minimal changes to the codebase.

This module is essential for ensuring secure access to the application, allowing users to manage their accounts, and providing a seamless transition between authenticated and non-authenticated states.

* + 1. LOGIN PAGE:

Overview

The HomeScreen module is a central component of a React Native application designed to facilitate the interaction between tutors and students. It offers a range of functionalities including viewing tutor posts, student needs, recorded sessions, and success stories. Users can also search for content, view detailed information about specific posts, add new posts, and access a menu with various options.

Key Features:

Header and Navigation

Menu Button: Toggles the visibility of a side menu containing various options.

Logo: Displays the application name "HomeLearner."

Profile Button: Opens a modal to display user profile information.

Search Functionality

Search Input: Allows users to search for specific content within the app.

Content Display

Tutor's Post: Displays a carousel of posts created by tutors.

Student's Need: Displays a carousel of posts highlighting student requirements.

Recorded Sessions: Displays a carousel of recorded video sessions available for viewing.

Success Stories: Displays a carousel of success stories from users.

Modals:

Post Details Modal: Shows detailed information about a selected post.

Add Post Modal: Allows users to create a new post by filling out a form.

Upcoming Events Modal: Displays notifications and details of upcoming events.

Menu Options:

Add Post: Opens the form to add a new post.

Videos: Highlights and scrolls to the recorded videos section.

Live Sessions, Reviews, Schedule, Contribute, About Us: Placeholder options for additional functionalities.

Footer Navigation:

Scroll to Top: Scroll the main content to the top of the screen.

Highlight Videos: Highlights the recorded videos section.

Add Post: Shortcut to open the add post form.

Upcoming Events: Opens the upcoming events modal.

User Interaction

TouchableOpacity components are used extensively to capture user interactions such as button presses.

ScrollView is utilized to enable horizontal and vertical scrolling of content.

Modal components provide a way to display overlaid content without navigating away from the current screen.

State Management

Various useState hooks are used to manage the visibility of modals, the selected product, form inputs, and other interactive states within the module.

Styling

The module employs a variety of styles defined in a StyleSheet to ensure a cohesive and visually appealing user interface, including specific styles for headers, content sections, modals, and footers.

* 1. TABLE DESIGN

This report provides a detailed description of the database tables used in our project. The database is designed to manage information about users, their posts, and video sessions effectively. The following sections outline the structure and design of each table within the database.

1. Overview:

The database is structured to support the functionalities of a tutoring platform where users can create posts for tutoring needs, upload videos of tutoring sessions, and interact with each other. The tables are designed to store user information, posts, and video details, ensuring data integrity and efficient data retrieval.

2. Database Schema:

Below is an ER (Entity-Relationship) diagram representing the tables, their columns, and the relationships between them.

1. Table Descriptions:

Table Name: Users

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| **user\_id** | INTEGER | PRIMARY KEY, AUTOINCREMENT | Unique identifier for each user. |
| **name** | TEXT | NOT NULL | The name of the user. |
| **email** | TEXT | NOT NULL, UNIQUE | The email address of the user. |
| **password** | TEXT | NOT NULL | Hashed password for the user. |
| **role** | TEXT | NOT NULL | Role of the user (**tutor** or **student**). |
| **contact** | TEXT | NULL | Contact number of the user. |

Description: Stores information about the users of the application, including tutors and students.

**Description**: Stores posts created by users, detailing their needs or offerings.

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| **post\_id** | INTEGER | PRIMARY KEY, AUTOINCREMENT | Unique identifier for each post. |
| **user\_id** | INTEGER | FOREIGN KEY REFERENCES **Users(user\_id)** | ID of the user who created the post. |
| **title** | TEXT | NOT NULL | Title of the post. |
| **details** | TEXT | NOT NULL | Detailed description of the post. |
| **contact** | TEXT | NULL | Contact information for the post. |
| **created\_at** | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Timestamp when the post was created. |

**Description**: Stores information about recorded video sessions available for viewing.

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| **video\_id** | INTEGER | PRIMARY KEY, AUTOINCREMENT | Unique identifier for each video. |
| **title** | TEXT | NOT NULL | Title of the video. |
| **source** | TEXT | NOT NULL | File path or URL of the video. |
| **created\_at** | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Timestamp when the video was uploaded. |

4. Relationships:

Users and Posts: Each user can create multiple posts. This relationship is represented by a foreign key (user\_id) in the Posts table referencing the Users table.

Users and Videos: Each user can upload multiple videos. If applicable, the Videos table can include a user\_id column to reference the user who uploaded the video.

5. Constraints and Indexes

Primary Keys: user\_id in Users, post\_id in Posts, and video\_id in Videos are primary keys ensuring each record is unique.

Foreign Keys: user\_id in Posts references user\_id in Users to maintain referential integrity.

Unique Constraints: The email column in the Users table is unique to prevent duplicate email addresses.

Indexes: Appropriate indexes are created on the primary keys and frequently queried columns to optimize performance.

6. Normalization

The database is designed to be in the Third Normal Form (3NF) to avoid data redundancy and ensure data integrity:

First Normal Form (1NF): All columns contain atomic and unique values.

Second Normal Form (2NF): All non-key attributes are fully functional and dependent on the primary key.

Third Normal Form (3NF): All attributes are only dependent on the primary key, ensuring no transitive dependencies.

* 1. ACTIVITY WORKFLOW

ACTIVITY 1: User Registration

Steps:

User accesses the registration page.

User fills in registration details (name, email, password, role).

System validates the input data.

If validation fails, an error message is displayed, and the user is prompted to correct the input.

If validation succeeds, the system creates a new user record in the database.

System sends a confirmation email to the user.

User confirms the email address.

Registration process is completed, and the user can now log in.

ACTIVITY 2: Post Creation

Steps:

User logs in to the platform.

User navigates to the "Create Post" page.

User enters post details (title, description, contact information).

System validates the entered data.

If validation fails, an error message is displayed, and the user is prompted to correct the input.

If validation succeeds, the system creates a new post record in the database.

System displays a success message, and the new post is now visible to other users.

ACTIVITY 3: Video Upload

Steps:

User logs in to the platform.

User navigates to the "Upload Video" page.

User selects a video file and enters video details (title, description).

System validates the video file and details.

If validation fails, an error message is displayed, and the user is prompted to correct the input.

If validation succeeds, the system uploads the video file and creates a new video record in the database.

System displays a success message, and the video is now available for viewing.

* 1. UML DIAGRAM:

4.4.1 Use Case Diagram:

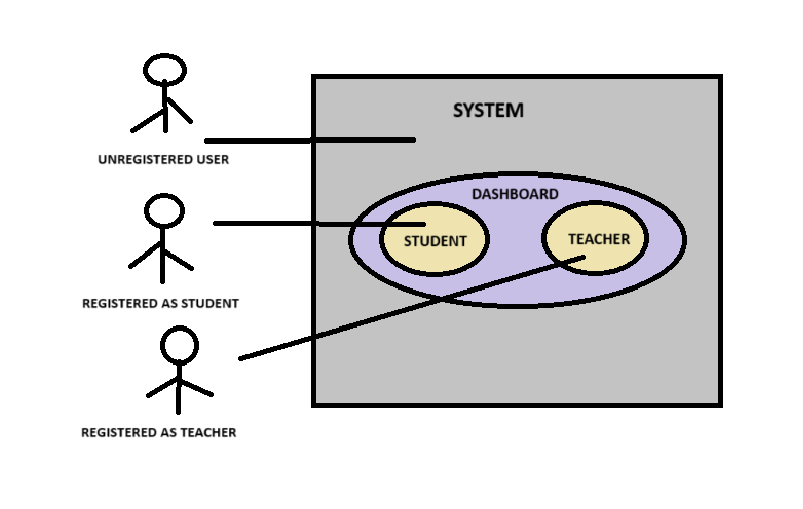


Figure Description

It can be accessible by both registered as well non-registered users. Non-register users can only see the dashboard content.

4.4.2 DFD Diagram:

4.4.2.1 Zero Level DFD

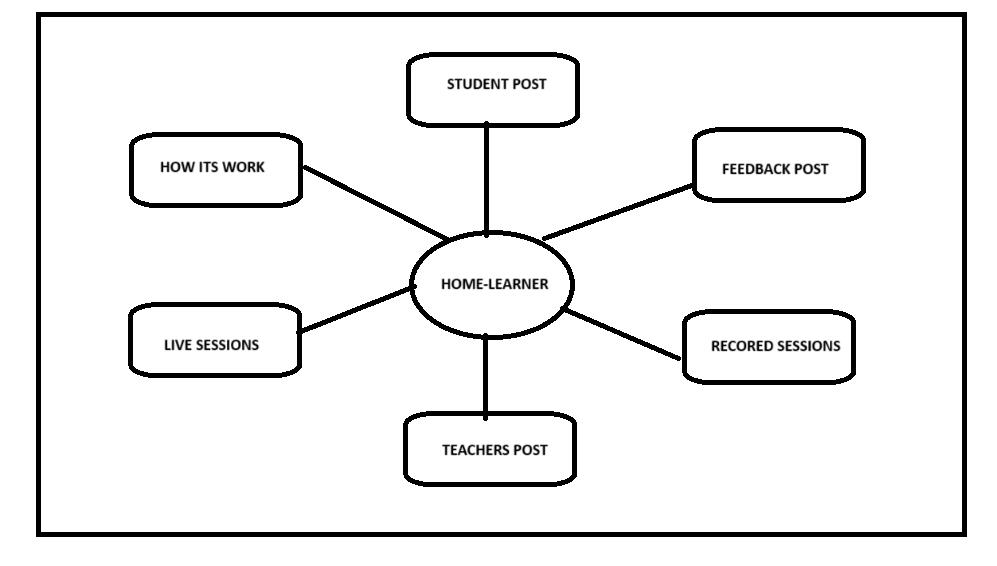


Figure Description

The first level DFD describes that the website has several functions like Students Post, Feedback Post, Recorded Sessions, Live Sessions, How Its Work etc.

4.4.2.2 First Level DFD

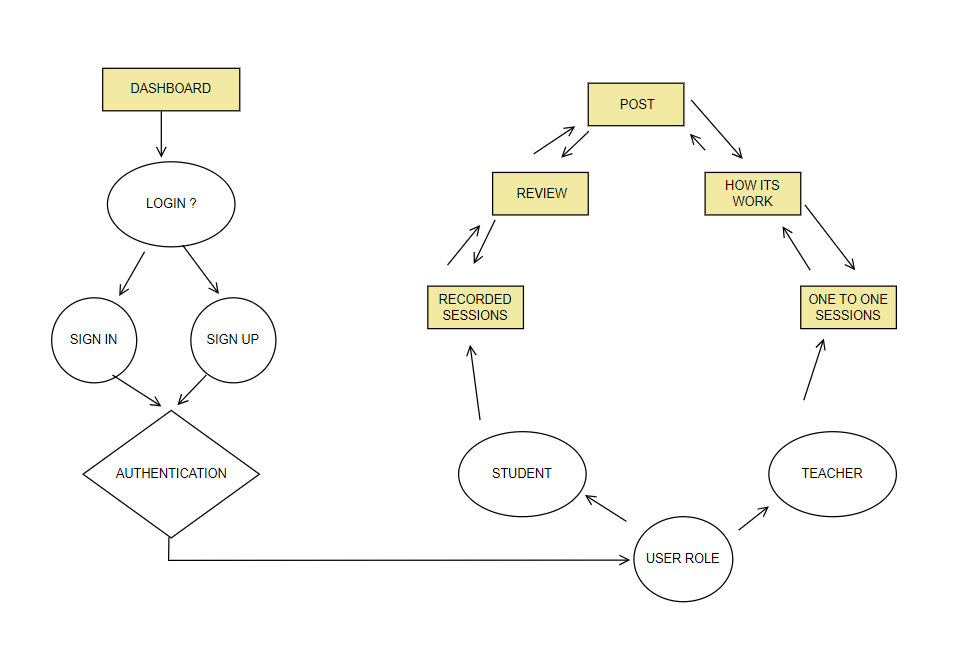


Figure Description

The second level DFD describes the basic work flow of application starting from login authentication to facilitate feature with user role like student can post their requirement, see teacher’s post, watch recorded sessions and for teachers, see student’ s post, take online sessions etc.

CHAPTER: 5

IMPLEMENTATION

* 1. WELCOME SCREEN:

The Welcome Screen of our app serves as the initial point of interaction for new users, offering them a guided introduction to your app's features. Through a series of three onboarding tutorial screens, users are presented with key aspects of your app's functionality. These screens highlight the convenience of accessing educational materials from anywhere, the personalized learning experiences tailored to individual preferences, and the ability to discover personalized educational pathways to meet specific learning objectives effectively.

Users can seamlessly navigate through these onboarding screens by swiping horizontally, engaging them in an interactive learning experience. Clear pagination indicators provide visual feedback on their progress. Additionally, the "Skip" button offers users the flexibility to bypass the tutorial if they prefer, while the "Next" button guides them through the screens, dynamically transitioning to prompt them to start using the app after the final tutorial screen. Upon completing the tutorial or opting to skip it, users are directed to the Sign-In screen. This transition is seamlessly handled by React Navigation, ensuring a smooth user experience. By effectively combining informative onboarding content with intuitive navigation and a clear call-to-action to sign in, the Welcome Screen sets the stage for users to explore and engage further with your app's features.

* 1. SIGN IN/ SIGN UP SCREEN:

The Sign-In and Sign-Up screens provide users with essential entry points into our app's ecosystem. The Sign-In screen allows existing users to access their accounts by providing their credentials, facilitating a seamless transition into the app's personalized experience. On the other hand, the Sign-Up screen offers new users the opportunity to create accounts, enabling them to unlock additional features and personalize their app experience. Both screens typically feature user-friendly interfaces, guiding users through the authentication process with clear prompts and input fields while ensuring the security of their login credentials. These screens play a crucial role in fostering user engagement and facilitating access to our app's content and functionalities.

* 1. HOME SCREEN:

The Home Screen is the central hub of the application, offering users a gateway to various features and content. It presents a user-friendly interface divided into sections, each catering to different types of content and functionalities. Through intuitive design and organization, users can easily navigate and explore the diverse offerings of the app. At the top, the header provides quick access to essential actions, such as toggling the side drawer menu for additional options and viewing user profiles. Below, the search functionality empowers users to find specific content swiftly, enhancing their overall browsing experience and enabling efficient content discovery.

The screen is structured to accommodate different types of content seamlessly. Users can browse through tutor and student posts, watch recorded sessions, read success stories, and stay updated on upcoming events or notifications. Modal dialogs offer detailed views of selected items and facilitate interactions like adding new posts, ensuring a smooth and engaging user experience. Additionally, the menu drawer provides easy access to supplementary navigation options, enhancing user convenience and accessibility. Overall, the Home Screen serves as a versatile platform for users to explore, engage with, and benefit from the app's diverse offerings effectively.

* 1. ADD POST:

The "Add Post" form on the Home Screen serves as a platform for both teachers and students to express their needs or availability within the app's community. Teachers can detail their expertise and availability for tutoring, while students can outline their learning requirements and preferred schedules. This facilitates meaningful connections between educators and learners, enhancing the app's role as a collaborative learning platform.

CHAPTER: 6

SYSTEM TESTING

System testing is a pivotal phase within the software development lifecycle, serving to validate the entire integrated system against predetermined requirements. Its fundamental objective lies in defect identification and confirmation of the system's intended functionality. A comprehensive test plan was meticulously crafted to encompass both functional and non-functional requisites, with testing executed in an environment mirroring production settings, employing tools such as Selenium for automation and JIRA for defect tracking. Test scenarios, rooted in system requirements, targeted user interactions, workflows, and edge cases, with detailed test cases specifying inputs, expected outcomes, and acceptance criteria, ensuring rigorous validation.

The execution of tests followed a phased approach, commencing with smoke tests and progressing through functional, regression, and performance assessments. Test data was meticulously prepared to cover diverse usage scenarios, guaranteeing exhaustive test coverage. Defect management was streamlined through JIRA, with defects categorized by severity and systematically tracked until resolution. Despite challenges such as time constraints and intermittent environment instability, strategic measures such as prioritizing critical test cases and bolstering environment monitoring were implemented for effective mitigation. Ultimately, system testing culminated in a successful validation of functionality and performance, unearthing critical defects that were promptly rectified. While boasting a commendable 90% pass rate out of 200 executed test cases, the recommendation for post-deployment load testing stands as a testament to the commitment to ensuring scalability and ongoing project quality improvement.

* 1. TESTING:

Testing in the Software Development Life Cycle (SDLC) is a critical phase where the developed software is evaluated to ensure it meets the specified requirements and is free of defects. This phase is essential to deliver a reliable, efficient, and high-quality product to the end-users.

Purpose of Testing:

The primary purposes of testing in SDLC are:

Verification: Ensure the software correctly implements specified functionalities.

Validation: Confirm the software meets the user's needs and requirements.

Defect Identification: Detect and fix defects in the software to improve quality.

Performance Evaluation: Assess the performance, security, and usability of the software.

* + 1. UNIT TESTING:

Unit testing, a fundamental aspect of software development, involves the meticulous examination of individual components or modules to ascertain their proper functioning. Typically undertaken by developers during the coding phase, its primary objective is to ensure the correctness and reliability of each unit.

The process encompasses several key steps, starting with the identification of the unit to be tested, followed by the creation of specific test cases with defined inputs and expected outputs. Subsequently, tests are executed using a designated unit testing framework, with any detected defects promptly addressed through debugging and correction. The iterative nature of re-running tests ensures the validation of fixes and guards against the emergence of new issues. Through this systematic approach, unit testing not only facilitates early bug detection, thereby reducing the cost and effort associated with later issue resolution, but also simplifies debugging processes and elevates overall code quality by promoting better coding practices and fostering code reliability.

* + 1. VALIDATION TESTING:

Validation testing stands as a pivotal phase within the software testing continuum, serving to confirm that the final product aligns with specified requirements and fulfills its intended purpose. Positioned typically at the culmination of the development cycle, its mandate is to ensure readiness for release to end-users, safeguarding against any deviations from the envisioned functionality.

The validation process encompasses multifaceted objectives, including the confirmation of user and business requirements, the evaluation of software usability in real-world scenarios, and the assurance of user satisfaction through the delivery of expected outcomes. Rooted in meticulous requirement analysis, it progresses through the development of comprehensive test cases, execution in production-like environments, and meticulous result comparison against predefined criteria. Feedback mechanisms integrated within this process allow for iterative refinement, ensuring that any identified issues are promptly addressed and rectified.

Various forms of validation testing augment this process, ranging from user acceptance testing (UAT) conducted by end-users to alpha and beta testing, which respectively involve internal teams and a limited cohort of external users. Through its systematic execution, validation testing not only serves as a barometer for product quality, flagging deficiencies that may have eluded earlier testing phases, but also acts as a conduit for enhancing user satisfaction by ensuring the software aligns closely with user expectations and requirements.

* + 1. FUNCTIONAL TESTING:

Functional testing stands as a cornerstone within software testing methodologies, dedicated to validating that a software system operates in accordance with specified requirements. Its focal point lies in meticulously assessing the functionality of an application, scrutinizing inputs against expected outputs to ensure seamless performance across diverse scenarios. The overarching objectives of functional testing encompass guaranteeing the correct execution of all specified functions, validating adherence to functional requirements, and swiftly identifying any deviations in software behavior from the expected norm. The procedural framework entails a systematic progression from requirement analysis to the development of exhaustive test cases, culminating in the execution of tests within a meticulously prepared testing environment. The subsequent analysis of test results facilitates the identification and reporting of defects, followed by iterative retesting cycles aimed at resolving issues and validating corrective measures.

Functional testing encompasses various specialized forms, including smoke testing, sanity testing, regression testing, integration testing, and user acceptance testing (UAT). Each type serves distinct purposes within the validation spectrum, from preliminary checks of basic functionality to comprehensive assessments of user satisfaction. Collectively, these testing approaches ensure not only the quality assurance of the software but also the fulfillment of functional requirements, ultimately contributing to enhanced user satisfaction and the delivery of a product aligned with user expectations.

CHAPTER:7

CCONCLUSION

The HOMELEARNER project is a comprehensive mobile application designed to revolutionize the tutoring industry by seamlessly connecting students, parents, and teachers. It successfully implements key features such as streamlined advertisement posting, smart search filters, interactive sample sessions, scheduling and calendar integration, background checks, and community support. The software requirement specification phase outlined the necessary hardware and software requirements, while the system design phase detailed module descriptions, table design, activity workflow, and UML diagrams. The implementation phase included creating the welcome screen, sign-in/sign-up screen, home screen, add post screen, and user review screen, followed by rigorous system testing to ensure optimal performance.

The HOMELEARNER project demonstrates strong technical, operational, and economic feasibility, with a robust technology stack and scalable business model. It addresses traditional tutoring challenges like limited accessibility, inefficient tutor matching, lack of transparency, cumbersome booking processes, and inadequate preview of tutor capabilities. By providing a flexible, efficient, and trustworthy platform, HOMELEARNER aims to become a leading solution in the tutoring industry. In conclusion, the project combines innovative features, user-friendly design, and thorough testing to deliver a cutting-edge mobile application that meets the evolving needs of the tutoring sector, fostering a supportive learning environment and promoting community-based initiatives.

APPENDIX: SCREENSHOTS

|  |  |
| --- | --- |
| Figure Description:  WELCOME SCREEN:  The welcome screen greets users with an engaging tutorial, introducing key features of our application to make a lasting first impression. | Figure Description:  SIGN IN/ SIGN UP SCREEN:  The Sign In/Sign Up screen offers a seamless entry point, allowing users to quickly access or create their accounts with ease. |
| Figure Description:  USER AUTHENTICATION:  After sign-in or sign-up, the User Authentication screen verifies credentials and directs users to their profile with "HOME"  & “LOGOUT” button. | Figure Description:  ERROR SCREEN:  The Error Screen alerts users with clear messages when they are not registered or encounter other issues, guiding them to resolve the problem. |

|  |  |
| --- | --- |
| Figure Description:  HOME SCREEN:  The Home Screen features sections for tutor posts, student requests, video sessions, and user reviews, all organized with top headings and bottom navigation for easy access. | Figure Description:  MENU BAR SCREEN:  The Menu Bar includes options for adding posts, viewing videos, accessing live sessions, checking reviews, scheduling, contributing, and learning about us. |

|  |  |  |
| --- | --- | --- |
| Figure Description:  ADD POST FORM:  The Add Post Form allows students and teachers to post their needs with user-friendly input fields, making it easy to share requests or offers. | | Figure Description:  UPCOMING VIEW:  The Upcoming Notifications screen displays alerts about events or important notices, keeping users informed and updated. |
| Figure Description:  DETAILED VIEW OF POST:  A detailed view of a post includes the user's name, title, and contact details. This information provides context about the author and facilitates direct communication. It is essential for understanding the author's background and engaging with them effectively. | Figure Description:  PLAYING VIDEO:  users can access recorded video sessions and play them within the "Recorded Session" section. It offers convenient playback functionality, allowing users to review content at their own pace. This feature enhances accessibility to past sessions, fostering continuous learning and engagement within the platform. | |

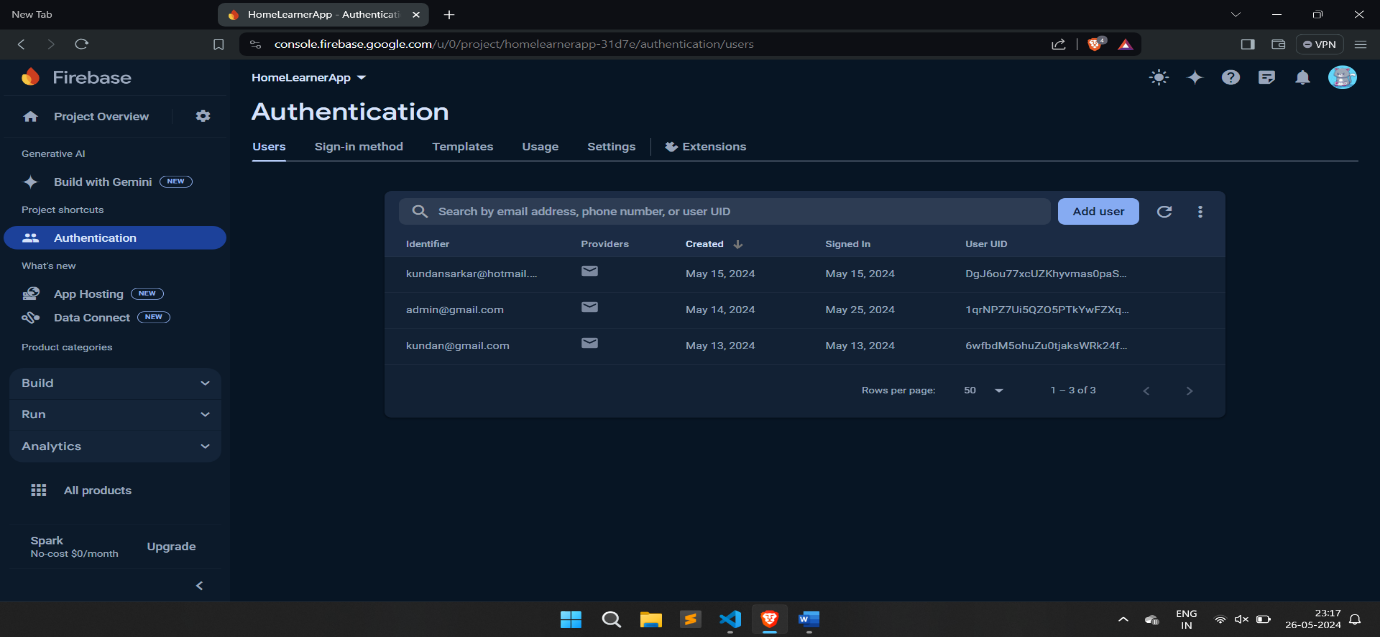


Figure Description:

AUTHENTICATION TABLE IN FIREBASE:

The Authentication Table in Google Firebase contains comprehensive details of registered users, ensuring secure storage and efficient management of authentication data. This table plays a crucial role in verifying user identities and maintaining the integrity of the user base.

BIBLIOGRAPHY

YouTube:

"THAPA TECHNICAL” for Installation and App setup.

(<https://youtu.be/9Q_LX94r3rg?si=hpKj0yKE0BDZJ6JU>)

"FLAIRA” for Onboarding welcome page.

(<https://youtu.be/pI4hMSoU7f0?si=cOPqU3PW6sCxiLgc>)

“CEBRAIL” for User authentication with Firebase.

(<https://youtu.be/MGBfr3WwIyw?si=5UZdKhNwa8c1ZW0n>)

Online Resources:

Canvas Designing

Official React Native Documentation)

Books:

Learning React Native: Building Native Mobile Apps with JavaScript. By Eisenman, Bonnie.

ChatGPT:

Source Recommendations: ChatGPT suggested relevant academic papers, articles, and resources based on the HOMELEARNER project abstract.

Clarifying Technical Terms and Concepts: ChatGPT clarified technical terms like React Native, Node.js, MongoDB, aiding my understanding of the report.

\* \* \* \* \* \* \*