An

Internship Report

On

“Educational Consultancy Apps and System”

By

Vedant Dhananjay Kahalekar

Under the Guidance

of

Ms. Mukta Shelke

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Mahatma Gandhi Mission’s College of Engineering, Nanded (M.S.)

Academic Year 2024-25

An Internship Report on

“Educational Consultancy Apps and System”

Submitted to

DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY, LONERE.

for partial fulfilment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

VEDANT DHANANJAY KAHALEKAR

Under the Guidance

of

Ms. Mukta Shelke

(DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAHATMA GANDHI MISSION’S COLLEGE OF ENGINEERING

NANDED(M.S.)

Academic Year 2024-25

CERTIFICATE

This is to certify that the internship entitled

"Educational Consultancy Apps and System"

Being submitted by ***MR. VEDANT DHANANJAY KAHALEKAR*** *to the Dr. Babasaheb Ambedkar Technological University, Lonere, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bonafide work carried out by him/her under my supervision and guidance. The matter contained in this report has not been submitted to any other university or institute for the award of any degree.*

Ms. Mukta Shelke

Guide

Dr. Mrs. A. M. Rajurkar

H.O.D

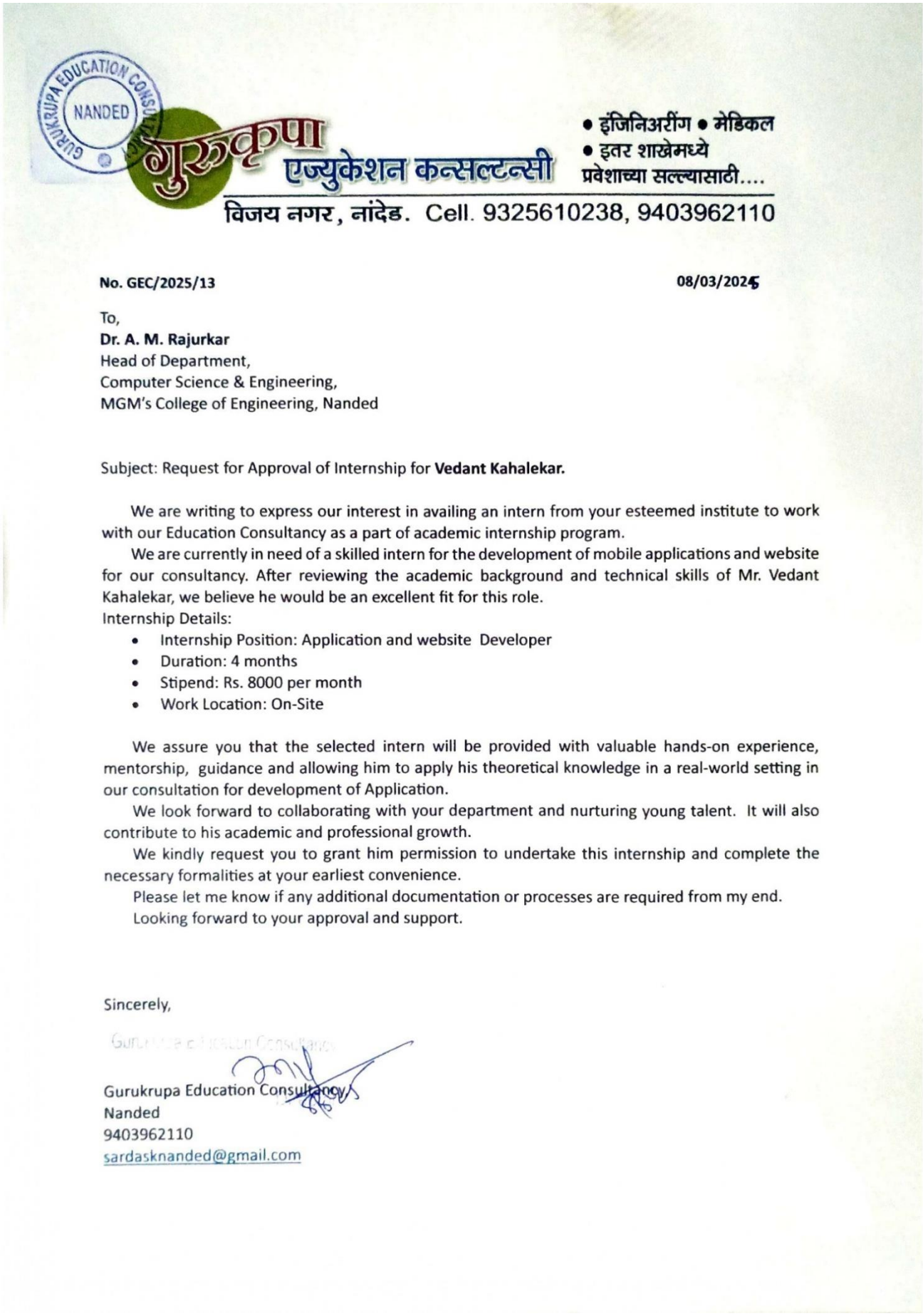
Computer Science & Engineering

Dr. Mrs. G.S. Lathkar

Director

MGM's College of Engineering, Nanded.

OFFER LETTER



GURUKRUPA EDUCATION CONSULTANCY NANDED

गुरुकृपा एज्युकेशन कन्सल्टन्सी

• इंजिनिअरींग • मेडिकल
• इतर शाखेमध्ये
प्रवेशाच्या सल्ल्यासाठी....

विजय नगर, नांदेड. Cell. 9325610238, 9403962110

No. GEC/2025/13 **08/03/2025**

To,
Dr. A. M. Rajurkar
Head of Department,
Computer Science & Engineering,
MGM's College of Engineering, Nanded

Subject: Request for Approval of Internship for **Vedant Kahalekar.**

We are writing to express our interest in availing an intern from your esteemed institute to work with our Education Consultancy as a part of academic internship program.

We are currently in need of a skilled intern for the development of mobile applications and website for our consultancy. After reviewing the academic background and technical skills of Mr. Vedant Kahalekar, we believe he would be an excellent fit for this role.

Internship Details:

- Internship Position: Application and website Developer
- Duration: 4 months
- Stipend: Rs. 8000 per month
- Work Location: On-Site

We assure you that the selected intern will be provided with valuable hands-on experience, mentorship, guidance and allowing him to apply his theoretical knowledge in a real-world setting in our consultation for development of Application.

We look forward to collaborating with your department and nurturing young talent. It will also contribute to his academic and professional growth.

We kindly request you to grant him permission to undertake this internship and complete the necessary formalities at your earliest convenience.

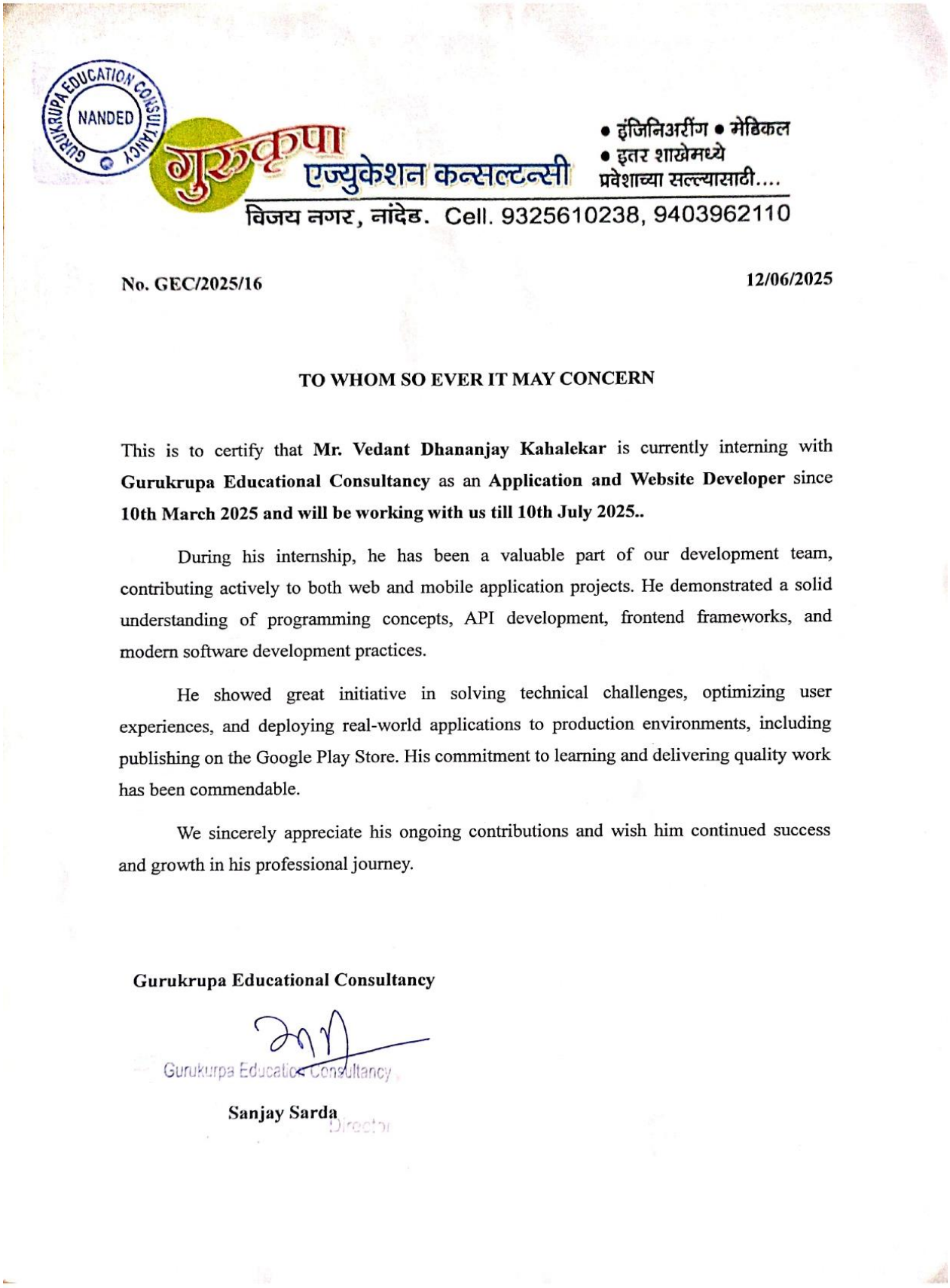
Please let me know if any additional documentation or processes are required from my end.

Looking forward to your approval and support.

Sincerely,

Gurukrupa Education Consultancy
Nanded
9403962110
sardasknanded@gmail.com

EXPERIENCE LETTER



GURUKRUPA EDUCATION CONSULTANCY NANDED

गुरुकृपा एज्युकेशन कन्सल्टन्सी

• इंजिनिअरींग • मेडिकल
• इतर शाखेमध्ये
प्रवेशाच्या सल्ल्यासाठी....

विजय नगर, नांदेड. Cell. 9325610238, 9403962110

No. GEC/2025/16 **12/06/2025**

TO WHOM SO EVER IT MAY CONCERN

This is to certify that **Mr. Vedant Dhananjay Kahalekar** is currently interning with **Gurukrupa Educational Consultancy** as an **Application and Website Developer** since **10th March 2025 and will be working with us till 10th July 2025..**

During his internship, he has been a valuable part of our development team, contributing actively to both web and mobile application projects. He demonstrated a solid understanding of programming concepts, API development, frontend frameworks, and modern software development practices.

He showed great initiative in solving technical challenges, optimizing user experiences, and deploying real-world applications to production environments, including publishing on the Google Play Store. His commitment to learning and delivering quality work has been commendable.

We sincerely appreciate his ongoing contributions and wish him continued success and growth in his professional journey.

Gurukrupa Educational Consultancy

Gurukrupa Educational Consultancy

Sanjay Sarda
Director

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my internship guide, **Ms. Mukta Shelke** for her invaluable support, guidance, and encouragement throughout the duration of this project. Her profound knowledge and expertise have been instrumental in the successful completion of this work. Her patience and willingness to assist me at every step have greatly enriched my learning experience. Her constructive feedback and insightful suggestions have not only helped me overcome challenges but also motivated me to strive for excellence.

We gladly take this opportunity to thank **Dr. A. M. Rajurkar** (Head of Computer Science & Engineering, MGM's College of Engineering, Nanded).

We are heartily thankful to **Dr. G. S. Lathkar** (Director, MGM's College of Engineering, Nanded) for providing facility during progress of project also for her kindly help, guidance and inspiration.

Last but not least we also thankful to all those who help directly or indirectly to develop this project and complete it successfully.

With Deep Reverence

VEDANT DHANANJAY KAHALEKAR

(B-Tech CSE-I)

ABSTRACT

This internship report outlines the experience and contributions made during a three-month internship period (March 10, 2025 – June 10, 2025) as an Application and Web Developer at Gurukrupa Educational Consultancy. The primary focus of the internship was the end-to-end development of two mobile applications—JEE Saral and NEET Saral—designed to assist students in exploring engineering and medical college options based on their entrance exam ranks.

Throughout the internship, a strong emphasis was placed on full-stack development, involving backend API creation in PHP with MySQL, integration of email verification using PHPMailer, and frontend development using React Native with Expo. The applications were successfully deployed on the Google Play Store, including user authentication systems, advanced search and filter functionalities, and responsive UI designs optimized for both mobile and web platforms.

In addition to development, the internship involved tasks such as debugging, performance optimization, and Play Store publishing. This report highlights the technical challenges encountered, solutions implemented, and the skills gained in modern software development practices. The internship provided hands-on industry experience and laid a strong foundation for future professional growth in the field of software engineering.

Beyond technical contributions, this internship offered valuable exposure to real-world workflows, cross-functional collaboration, and client-oriented development. Working under the guidance of experienced mentors helped sharpen both my problem-solving abilities and communication skills. The experience has significantly enhanced my confidence in building scalable applications and strengthened my commitment to continuous learning in the ever-evolving tech landscape.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Overview of the Internship

Internships play a vital role in bridging the gap between theoretical learning and real-world industry practices. They provide students with a platform to apply academic knowledge to live projects, explore their domain of interest in a practical setting, and develop key professional and technical skills. As a part of the B.Tech curriculum in Computer Science and Engineering at MGM's College of Engineering, Nanded, I was fortunate to undertake my academic internship at Gurukrupa Educational Consultancy, a firm focused on educational services and mobile-first solutions for students preparing for competitive exams in India.

The internship began on 10th March 2025 and is scheduled to conclude on 10th July 2025, spanning a total of four months. Throughout this duration, I was involved in a variety of real-time development tasks, particularly focused on mobile application and backend development. My primary responsibilities included working on two mobile apps—JEE Saral and NEET Saral—designed to simplify college prediction and admission guidance for students appearing for JEE and NEET exams. Additionally, I contributed to the design and implementation of a custom PHP backend for user management and API integration.

The internship offered me the chance to work in a collaborative team environment and exposed me to several important aspects of the software development lifecycle, such as planning, development, testing, deployment, and user feedback incorporation. By engaging in these processes, I was able to enhance not only my technical competencies in tools like React Native, Expo, PHP, and MySQL, but also improve my problem-solving, communication, and time management skills.

Moreover, working in a client-focused consultancy environment helped me understand how technology is leveraged to solve real-world problems and add value to end users. The projects I worked on were not just academic simulations—they had actual users and measurable impact, which made the experience both challenging and rewarding. I also gained firsthand exposure to production-level deployment practices, including API

hosting, database integration, authentication mechanisms, and UI/UX optimization for mobile platforms.

Overall, this internship has been a significant milestone in my academic journey, providing a solid foundation for my future career in the tech industry. It helped me gain confidence in applying my classroom knowledge to practical challenges and deepened my interest in full-stack and mobile application development.

1.2 Company Profile

Gurukrupa Educational Consultancy is a career guidance and consultancy firm that assists students in selecting the right educational path, especially in the fields of engineering and medicine. Based in India, the organization focuses on helping aspirants of JEE (Joint Entrance Examination) and NEET (National Eligibility cum Entrance Test) to identify suitable colleges based on their ranks, categories, quotas, and other relevant parameters.

The consultancy not only offers offline services but has also expanded into the digital space to reach a broader audience. As part of this digital transformation, the company has initiated the development of mobile and web applications such as JEE Saral and NEET Saral, which were the primary focus areas during my internship.

Fig 1.1 Company Logo

Key Services Offered by the Company:

- Personalized career counseling for medical and engineering aspirants.
- Guidance on college selection based on entrance ranks.
- Information on quotas, cutoffs, and category-wise admissions.
- Mobile and web-based platforms to support digital consultancy.

1.3 Objectives of the Internship

The primary objective of my internship was to contribute meaningfully to the development of web and mobile solutions that simplify the college selection process for students. The internship was designed with both learning and delivery outcomes in mind. The key goals were as follows:

- To develop and deploy fully functional mobile applications (JEE Saral and NEET Saral).
- To understand and contribute to the **full-stack development lifecycle**, including frontend, backend, and deployment.
- To work in a collaborative environment and follow real-world software development practices.
- To gain hands-on experience with technologies like **React Native, PHP, MySQL**, and tools such as **Expo**, **Hostinger**, and **PHPMailer**.
- To apply UI/UX design principles for building responsive and user-friendly interfaces.

The primary objective of my internship at Gurukrupa Education Consultancy was to contribute effectively to the design, development, and deployment of web and mobile solutions that assist students in making informed decisions during their college selection process. This internship offered a balanced blend of practical exposure and learning opportunities, enabling me to apply academic knowledge in real-world scenarios while simultaneously enhancing my technical and soft skills.

One of the main goals was to develop and deploy fully functional mobile applications, specifically the JEE Saral and NEET Saral apps. These platforms are designed to help students predict their potential college admissions based on their exam ranks and explore various institutions across categories, quotas, and locations. Working on these apps allowed me to engage in all stages of the software development lifecycle—from ideation and UI design to coding, testing, and deployment.

The internship also focused on providing an in-depth understanding of full-stack development, including both the frontend and backend components. On the frontend, I worked extensively with React Native and Expo to create responsive, cross-platform

mobile applications. On the backend, I contributed to a PHP-based authentication API that included database management using MySQL, user session handling, and secure email communication using PHPMailer.

Another significant objective was to function as part of a collaborative team environment, gaining insight into how software projects are executed in professional settings. I followed real-world practices such as modular coding, version control, debugging, and performance optimization, while also learning the importance of communication and feedback in agile development workflows.

Hands-on experience with tools like Hostinger (for deployment), Composer (for PHP dependency management), and Expo Application Services (EAS) helped me understand practical aspects of hosting, builds, and releases. I was also introduced to UI/UX design principles, ensuring that the interfaces I worked on were user-friendly, aesthetically appealing, and responsive across devices.

In summary, the internship was structured to help me grow as a developer by solving real-world problems using current technologies, understanding the dynamics of a production environment, and contributing to solutions that have direct educational value for students across India.

1.4 Scope of Work

The scope of the internship at **Gurukrupa Educational Consultancy** was extensive and multifaceted, encompassing the core areas of mobile application development, backend API engineering, deployment processes, and active team collaboration. The tasks I undertook during the internship reflected real-world development cycles and provided me with a comprehensive understanding of both technical implementation and project delivery.

Mobile App Development

One of the primary areas of focus was mobile application development. I contributed to building two key Android applications: JEE Saral and NEET Saral, both developed using React Native with the Expo framework. These applications were designed to assist students in selecting colleges based on their entrance exam ranks and preferences. My responsibilities included creating and integrating multiple UI screens, such as user login, registration, dashboards, result search pages, and advanced filters. The

development process emphasized responsive design, intuitive navigation, and a smooth user experience across a wide range of Android devices. Throughout the development, I gained valuable insight into mobile UI component libraries, navigation handling, and state management in React Native.

Backend Development

In addition to frontend work, I also developed and maintained several RESTful APIs using PHP and MySQL. These APIs handled critical backend functionalities including user authentication, session management, college data retrieval, and secure email communication. I integrated PHPMailer to support automated email verification and transactional messages, ensuring a secure and seamless onboarding experience for users. The backend code was modularized, documented, and optimized for performance and scalability, reflecting best practices in secure and maintainable PHP development. I also worked on managing the MySQL database schema and queries to support real-time data needs of the mobile apps.

Deployment and Testing

The internship also offered hands-on experience in application deployment. I was responsible for preparing the mobile apps for deployment to the Google Play Store, which involved signing builds using Expo Application Services (EAS), configuring the apps' privacy policies, setting up metadata and screenshots for the store listing, and ensuring that all compliance checks were met. Additionally, I engaged in rigorous manual testing to identify bugs and performance issues across various Android devices. Using feedback from users and internal reviews, I conducted several rounds of debugging and optimization to enhance the overall app stability.

Additional Contributions

Beyond core development, I contributed to improving the overall workflow of the team. This included writing README documentation and backend API references to help future developers understand the system architecture. I actively participated in weekly sprint meetings and daily stand-up sessions, where I shared progress updates, discussed challenges, and collaborated on planning tasks for upcoming sprints. I also resolved Git conflicts, maintained proper branching strategies, and ensured timely pushes to the

shared repository. These collaborative efforts enhanced my understanding of agile development practices and version control in a team setting.

In summary, the scope of the internship extended across full-stack application development, testing, documentation, and team coordination. This comprehensive exposure allowed me to strengthen my technical foundation while gaining valuable experience in delivering real-world solutions.

1.5 Day-to-Day Activities

During the course of my internship at Gurukrupa Educational Consultancy, my daily responsibilities were both dynamic and instructive, reflecting the typical workflow of a full-stack development team. I was actively engaged in tasks that spanned the software development lifecycle, from planning and coding to testing and deployment. Each day brought new challenges and opportunities to learn, apply, and refine my technical and collaborative skills.

A regular part of my routine involved attending daily stand-up meetings and weekly scrum sessions. These meetings were crucial in aligning the team's objectives, identifying ongoing issues, and setting short-term goals. During these sessions, I shared updates on my current tasks, discussed any blockers I encountered, and collaborated with team members to find timely solutions. These interactions not only kept the development process transparent but also familiarized me with Agile methodologies and iterative development practices.

On the technical front, I actively contributed to both frontend and backend development. My frontend tasks primarily involved developing interactive UI components using React Native. I worked on features such as user registration and login screens, navigation drawers, dashboards, and results filtering pages. Attention was given to implementing responsive layouts and ensuring seamless user experiences across various devices. I regularly used Axios and Fetch APIs to connect the frontend with backend services, enabling real-time data retrieval and updates from the server.

In parallel, I worked on backend tasks including the creation and refinement of RESTful APIs using PHP and MySQL. These APIs powered various functionalities within the apps such as user authentication, data storage, and retrieval of college-related information. I was responsible for handling server-side logic, optimizing queries, and

ensuring secure data transactions. I also collaborated with the team to manage API versioning and endpoint documentation, ensuring clarity and consistency for frontend integration.

Testing and quality assurance formed another essential part of my daily work. I conducted manual testing of newly implemented features, checking for usability issues, design inconsistencies, and device-specific bugs. I responded promptly to feedback from early users and internal testers, debugging code and fixing issues to ensure a smooth and reliable user experience. This iterative process improved my ability to diagnose problems quickly and deliver efficient solutions.

As development progressed, I was involved in deployment tasks, including the publishing of the JEE Saral and NEET Saral applications on the Google Play Store. I prepared required assets such as app icons, banners, and screenshots, and wrote app descriptions and version histories for the store listings. I also managed build signing, version control, and privacy policy linking, ensuring full compliance with Play Store guidelines. with practical exposure to working in a collaborative development environment and enhanced both my technical proficiency and professional discipline.

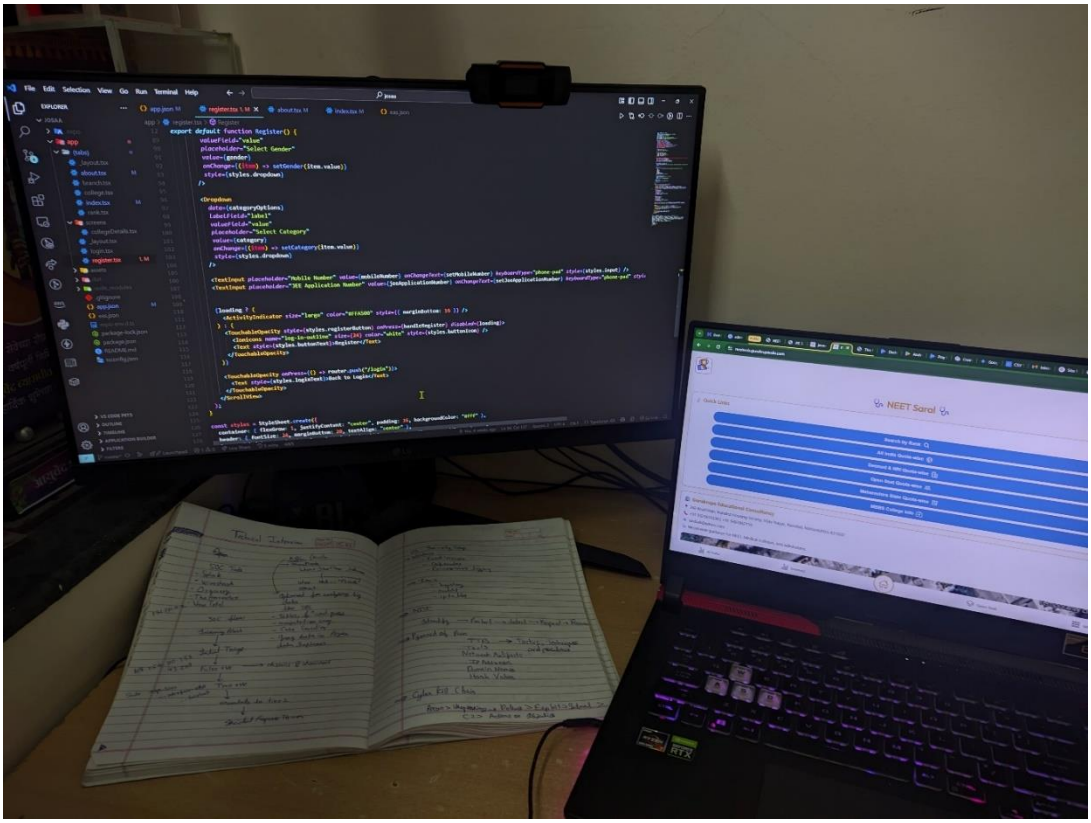


Fig 1.2 My Workspace

Chapter 2

SYSTEM ARCHITECTURE & TECHNOLOGY STACK

2.1 Overview of Applications Developed

During my internship at Gurukrupa Educational Consultancy, I was primarily involved in the design, development, and deployment of two cross-platform mobile applications: **JEE Saral** and **NEET Saral**. Both applications aim to assist students in identifying potential colleges based on their entrance exam scores—JEE (for engineering aspirants) and NEET (for medical aspirants). The apps are designed to provide students and parents with a simplified, intuitive, and data-driven platform to explore college admission possibilities.

JEE Saral focuses on candidates of the Joint Entrance Examination and includes features like rank-based prediction of eligible colleges, branch selection, and access to motivational resources and consultancy support. Users can filter colleges by category (IIT, NIT, IIIT, etc.), view program-specific cutoff ranks, and navigate directly to consultancy resources.

NEET Saral, on the other hand, is tailored for NEET aspirants seeking medical admissions across different states and quotas in India. It includes features such as quota-wise browsing (All India, Deemed/NRI, MH State, etc.), college cutoff listings, and rank-based filtering by category and location. It offers a user-friendly experience to help students shortlist colleges using their entrance exam ranks, while also integrating detailed college data and intuitive UI components.

Both apps were built using **React Native with Expo**, ensuring compatibility across Android and web platforms. They share a similar design philosophy but are uniquely tailored in functionality, data integration, and user flow.

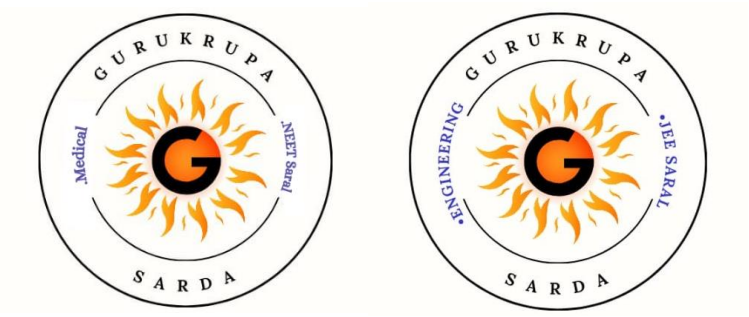


Fig 2.1 App Logos

2.2 System Architecture

The system architecture of the JEE Saral and NEET Saral mobile applications is based on a modular, scalable client-server model. This design enables clear separation between user-facing components (frontend) and data-handling services (backend), enhancing maintainability, scalability, and development efficiency. The architecture is built with an emphasis on ease of use, reliability, and performance, catering specifically to students navigating the complex process of college selection.

Frontend Architecture (Client Side)

The frontend of both applications is developed using React Native integrated with Expo, which allows for building performant mobile applications using JavaScript and React. One of the key advantages of using Expo is its abstraction of native configuration, making development and testing faster, especially for cross-platform deployment.

React Native provides a component-driven architecture, where each screen and UI section is encapsulated as a reusable component. The applications leverage file-based routing using Expo Router, which enables structured navigation by mapping folders and files directly to screens and routes in the app. This method enhances scalability and simplifies onboarding for new developers.

In terms of navigation, the apps implement both Tab Navigation and Stack Navigation using the React Navigation library. The Tab Navigator is used to organize high-level sections of the app, such as All India Quota, Deemed/NRI, Open Seats, and Maharashtra State (specific to NEET Saral). These sections are persistently accessible through a bottom tab bar for quick user access. On the other hand, Stack Navigation allows for hierarchical navigation, such as drilling down into college detail pages from filtered lists or performing multi-step tasks like registration and verification.

State within the application is managed using React Hooks, such as useState for local component state and useEffect for managing side effects like API calls. Additionally, persistent storage is implemented via the @react-native-async-storage/async-storage library. This is used for storing login sessions and user preferences, allowing users to resume from where they left off.

The applications are built with a responsive design philosophy. Using platform-aware stylesheets and conditional rendering techniques, UI components automatically adjust to different screen sizes and orientations, ensuring a consistent experience on both phones and tablets. Touch interactions, scroll views, and accessibility features are also incorporated to meet usability standards.

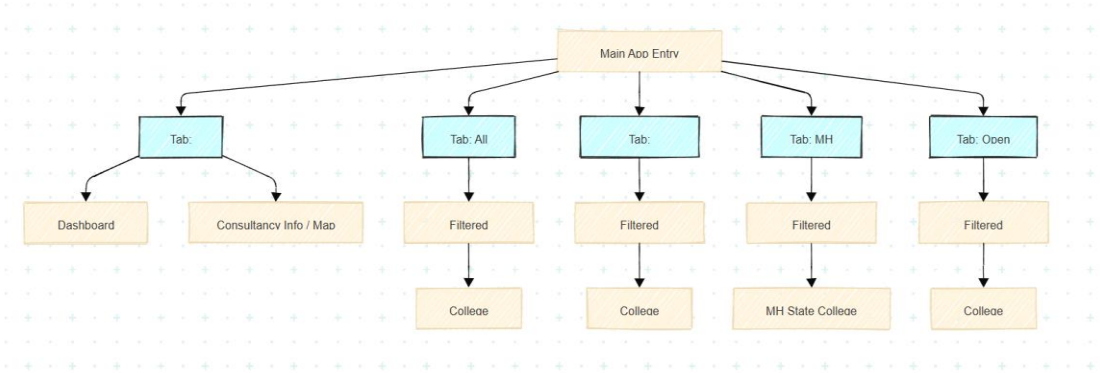


Fig 2.2 App Structure

Backend Architecture (Server Side)

The backend of the applications is implemented using PHP, deployed on a Hostinger shared hosting environment. The server follows a RESTful API architecture, providing well-defined endpoints for handling authentication, user data, filtering logic, and email notifications. The backend acts as the intermediary between the client-side mobile app and the MySQL database.

A modular directory structure is followed on the server to keep different logic segments isolated and easier to manage. For instance, user session management is handled by dedicated scripts like login.php and register.php. These scripts are responsible for validating credentials, checking for existing users, hashing passwords, and generating login tokens.

Data transactions—such as retrieving college lists or applying filters based on exam scores, ranks, seat types, and quotas—are handled through specialized endpoints like getusers.php and filter.php. These APIs accept query parameters and return structured JSON responses, which are consumed by the frontend. This ensures minimal data transfer and quick load times, especially important for users on mobile networks.

For communication features, the backend integrates PHPMailer, a popular library for sending emails via SMTP. Scripts like smtp_mailer.php are used to send verification

emails during user registration or notifications based on application status. The PHPMailer setup is configured securely with proper authentication and encryption protocols to avoid spam issues and ensure deliverability.

The database connection and interaction layer is managed through a central script db.php. This script includes reusable functions for connecting to the MySQL server, executing queries, and returning results. To maintain security and ensure the integrity of user data, the backend system follows several industry-standard security practices. One of the primary measures is input sanitization, where all user inputs—whether from registration forms, search filters, or login credentials—are cleaned and validated before being processed. This prevents malicious content from being executed or stored in the database. Additionally, the backend uses parameterized queries while interacting with the MySQL database. Parameterized queries help separate SQL code from data, thereby preventing SQL injection attacks—a common security vulnerability where attackers attempt to manipulate SQL statements through untrusted input. By adhering to these practices, the system upholds a robust defense against common web vulnerabilities and ensures safe and reliable data transactions between users and the server.

System Workflow and Integration

The overall workflow of the system is designed to ensure seamless interaction between the frontend and backend components of the mobile applications. This follows a request-response model where the frontend acts as the client, making asynchronous API calls to the backend server. These calls are typically triggered based on specific user actions such as logging in, registering, applying filters to view specific colleges, or requesting detailed information about a particular institution.

To implement these requests, libraries like Axios and the native Fetch API in JavaScript are used. These tools allow the React Native frontend to send HTTP requests (typically POST or GET) to the appropriate PHP scripts hosted on the backend server. For example, when a user selects filters like category, quota, or state, the frontend sends a request to filter.php, which processes these parameters, queries the MySQL database, and returns the filtered list of colleges. Similarly, login and registration functionalities communicate with login.php and register.php, respectively, to authenticate users and create new accounts.

The returned data, usually in JSON format, is then parsed and rendered dynamically in the application. The frontend displays this information using custom-designed UI components tailored to mobile devices, ensuring a clean and intuitive user experience. Special care is taken to handle loading states, error responses, and empty result sets to enhance usability.

To maintain a robust integration between these layers, testing plays a vital role. Postman is used extensively during development to manually test backend endpoints. This helps in verifying the correctness of the API logic, examining request and response formats, and identifying potential issues such as improper parameter handling or SQL errors. On the frontend side, testing is carried out using Android Studio emulators as well as physical Android devices to ensure that the UI behaves consistently across different screen sizes and OS versions.

Another crucial part of the workflow is version control, which is managed through Git. This allows for efficient tracking of changes across both frontend and backend codebases. By maintaining separate branches and pushing updates regularly to the remote repository (GitHub), the development team ensures that features can be developed independently, merged systematically, and rolled back if needed. This practice not only enhances team collaboration but also helps in maintaining the integrity of the code across various modules.

The architecture also supports scalability. Since the API endpoints are modular and RESTful, new features or filters can be added with minimal changes to the overall codebase. This makes the system future-proof and adaptable to new requirements or data structures.

User Interaction Flow

A simplified flow of interaction within the JEE Saral and NEET Saral applications is structured to ensure a seamless and intuitive user experience, following a consistent client-server communication pattern. Below is a typical sequence of operations that highlights how different system components coordinate to fulfill user actions:

1. **User launches app → Login/Register:** Upon launching the app, users are greeted with an onboarding screen followed by a login/register interface. First-

time users are prompted to register by entering their name, email, and password, while returning users can simply log in using existing credentials.

2. **Frontend sends credentials → Backend validates via login.php:** When the user submits login information, the frontend app packages the credentials into a POST request using fetch or Axios and sends it to the backend login.php endpoint. The backend authenticates the input by querying the MySQL database for matching records and responds with a success or failure message along with session tokens or user data.

3. **User selects filter options → Filter API returns matching colleges:** Once logged in, users can access a range of filter options—such as exam type, quota, category, state, rank, and percentile. These inputs are transmitted to the filter.php API, which processes the parameters and runs optimized SQL queries to return a tailored list of colleges that meet the user's criteria.

4. **User views college details → Data fetched from MySQL and displayed:** When a user selects a college from the filtered list, a detail view is rendered showing essential information such as available courses, closing ranks, fee structure, and seat categories. This data is retrieved from the MySQL database through dedicated data retrieval endpoints (getcollege.php or similar), which are accessed using dynamic route parameters.

5. **User receives confirmation email → Sent via PHPMailer:** In scenarios like new registration or successful form submission, the backend system sends a transactional email using PHPMailer configured with SMTP settings. This ensures users receive verification links, password reset options, or confirmation messages, thereby enhancing reliability and engagement.

This interaction pipeline illustrates a highly modular and responsive system where every user action corresponds to a well-defined backend operation. The decoupling of frontend and backend logic not only improves maintainability but also enables smoother debugging and feature scaling. The application architecture also supports asynchronous operations, allowing users to interact with the UI while backend tasks complete in parallel.

Moreover, by maintaining consistent API patterns and clearly defined endpoints, the system ensures a secure and efficient data exchange. Authentication is safeguarded through validated sessions, and data payloads are sanitized to prevent SQL injection or malformed requests. This flow collectively results in an application ecosystem that is both robust and user-friendly, ultimately achieving the intended objective of assisting students in making informed college decisions.

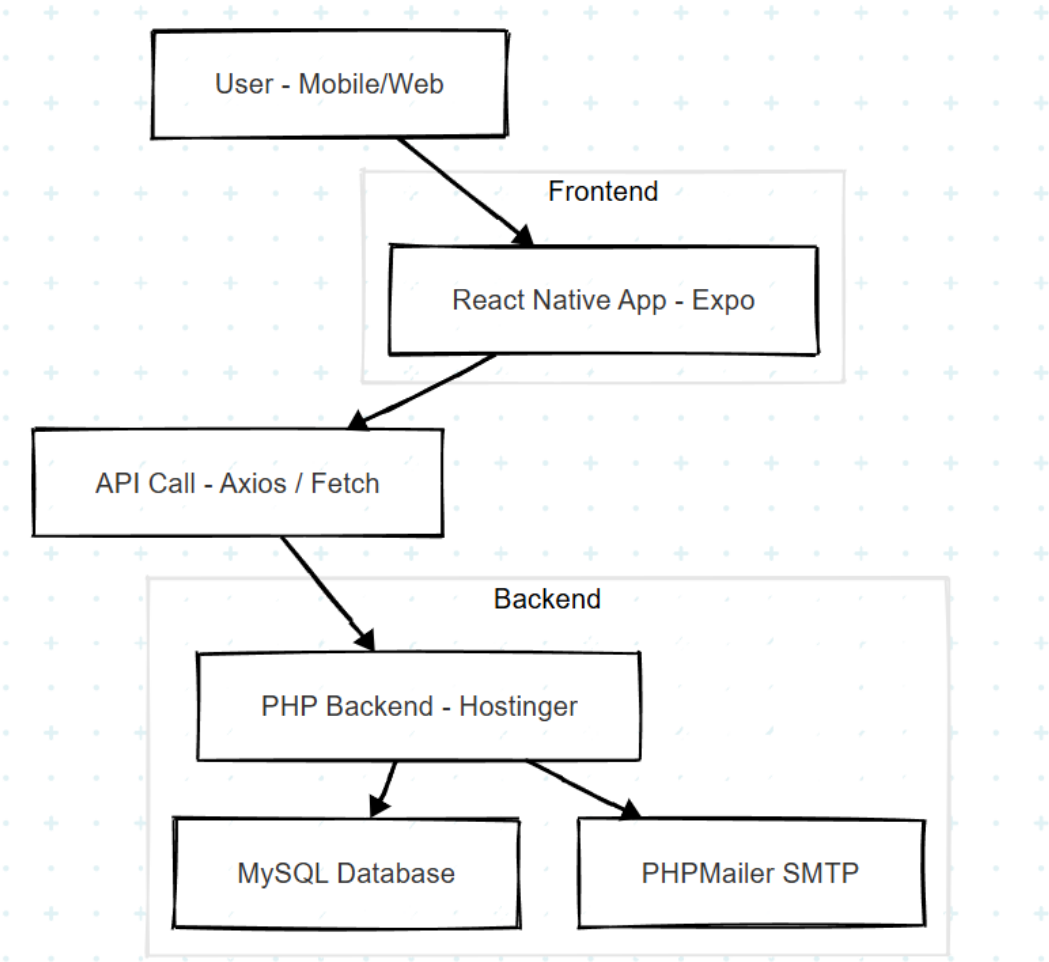


Fig 2.3 Client-Server Architecture

2.3 Technology Stack Used

The development of JEE Saral and NEET Saral applications involved the strategic selection of a full-stack technology suite tailored to efficiently meet the project's goals within the internship duration. The tech stack spanned frontend and backend technologies, database management, email integration, deployment platforms, and development tools. This multi-layered architecture enabled the team to build

responsive, scalable, and maintainable applications while keeping developer productivity and user experience as top priorities.

Frontend Technologies

1) React Native with Expo

React Native was the core framework used to develop the mobile applications. With Expo, the development process became more efficient due to the managed workflow that simplifies setup, configuration, and publishing. Expo's hot-reloading and over-the-air updates allowed for faster debugging and iterative development. The ability to use a single codebase for both Android and web platforms helped reduce overhead and increased consistency across devices.

2) JavaScript & TypeScript

JavaScript served as the primary scripting language throughout the development of both the JEE Saral and NEET Saral applications. Its non-blocking, asynchronous capabilities and event-driven nature made it highly suitable for building interactive mobile applications where responsiveness and real-time updates were key. JavaScript's extensive ecosystem of libraries, such as Axios, React Native Paper, and various utility packages, also contributed to rapid feature implementation and iteration during the internship period.

In scenarios where a higher level of code reliability and maintainability was essential—particularly in complex modules dealing with data validation, API interaction, and user authentication—the use of TypeScript was introduced. TypeScript extends JavaScript by adding static type definitions, which helped prevent common runtime errors, made the code more self-documenting, and enhanced the development experience by offering better tooling support such as autocompletion and compile-time error checking.

This dual approach allowed for both speed and stability. JavaScript was used in fast-evolving UI components, animations, and layouts, while TypeScript was employed in more critical areas where accurate data structure definitions and predictable behavior were important. For instance, when designing the interfaces for filtered college listings or the schema for storing user session data, TypeScript ensured that incorrect data formats or missing properties would be caught early during development.

Furthermore, this hybrid model also facilitated smoother team collaboration. Developers working on different parts of the application could understand and integrate code faster due to the clarity offered by TypeScript's interfaces and type annotations. It also aided in onboarding and code reviews, as the explicit definitions reduced ambiguity about the expected input and output of functions or components.

Over time, adopting TypeScript in more parts of the codebase became a best practice, especially as the application grew and the interdependencies between components became more complex. This approach not only improved code reliability but also set a strong foundation for potential future enhancements, making the project more robust and scalable.

3) Expo Router

To manage navigation across multiple screens and flows in the JEE Saral and NEET Saral applications, Expo Router was chosen as the primary routing solution. Expo Router brings the benefits of file-based routing, similar to modern web frameworks like Next.js, into the React Native environment. This approach allowed each screen to be defined in its own file, reducing complexity and improving modularity in the codebase. By mapping routes directly to the file system, developers could easily identify and modify the navigation structure without needing to manually configure route paths or nested navigators, as is common in traditional navigation setups.

The routing system was designed to reflect the logical flow of the applications. Key screens such as login, registration, dashboard, search filters, college details, and user profile were structured in separate directories, allowing for better organization. For example, all NEET-specific and JEE-specific flows were compartmentalized under distinct folders, enabling reusability and easier maintenance. This clear separation also helped avoid confusion when working on shared modules, especially when multiple developers were contributing to the same repository.

One of the standout advantages of using Expo Router was its seamless integration with Stack and Tab Navigators, enabling the construction of nested navigation flows. Tabs were used to segment high-level features (like Deemed Universities, AIQ, and State Quotas), while stack navigation facilitated transitions between sub-screens, such as navigating from a list of colleges to individual college details. This hierarchical structure improved user experience by offering intuitive, layered navigation.

Additionally, Expo Router facilitated deep linking, making it easier to implement features like push notifications or in-app redirections. This became particularly useful in scenarios where users received links via email (e.g., for verification or updates) and needed to be routed directly to specific sections of the app. Because the routing mirrored the folder structure of the codebase, these deep links could be configured with minimal effort.

From a collaborative standpoint, the use of file-based routing significantly enhanced team productivity. New screens could be created and tested independently without disrupting other parts of the navigation flow. This made the development process more scalable, especially when new features were being introduced rapidly. It also streamlined code reviews, as routing logic was inherently tied to the folder structure, making changes more predictable and transparent.

4) UI Libraries: React Native Paper & Vector Icons

React Native Paper provided material design components such as buttons, cards, and dialogs, helping maintain a uniform design language throughout the app. Vector Icons were used to visually enhance the interface, improving usability and intuitive interaction, especially for action-driven tasks like search or back navigation.

5) Async Storage

Security and session persistence were managed on the client side using @react-native-async-storage/async-storage. This enabled storage of JWT tokens and user preferences, ensuring a smoother user experience by avoiding repeated logins and retaining filter settings across sessions.

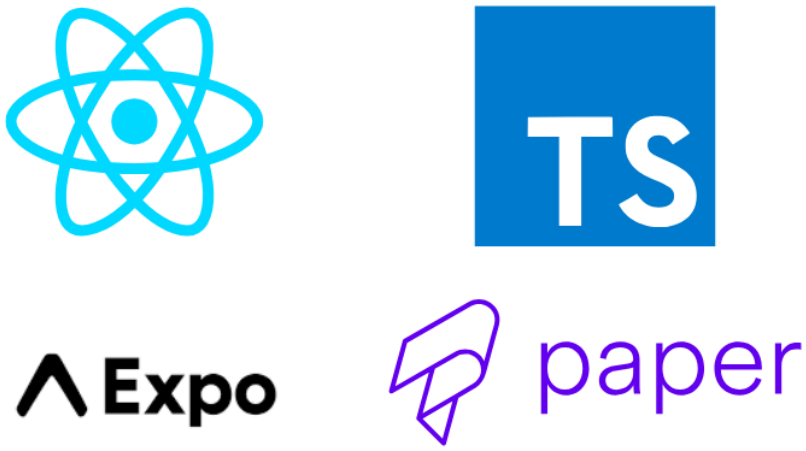


Fig 2.4: Frontend Technologies

Backend Technologies

1) PHP

The backend of the JEE Saral and NEET Saral applications was constructed using PHP, a widely adopted server-side scripting language known for its simplicity, flexibility, and deep integration capabilities with web servers and relational databases. One of the key reasons for choosing PHP was its compatibility with the Hostinger hosting environment, which offered a shared server setup that seamlessly supported PHP out of the box. This eliminated the need for additional configuration or dependencies, enabling quicker deployment and stability throughout the internship project.

PHP played a central role in orchestrating all server-side operations. It was responsible for handling user authentication flows, such as login, registration, and session management. Scripts like login.php and register.php ensured that user credentials were securely validated and stored, with additional validation layers to check for duplicate entries, input formatting, and basic security practices such as password hashing. These foundational elements ensured a robust user experience and laid the groundwork for further secure API development.

Beyond authentication, PHP was also used extensively to power the application's data filtering capabilities. This was particularly important for retrieving college-related data based on multiple dynamic filters selected by the users. The script filter.php played a crucial role in taking frontend inputs like state, rank, category, and college type, then dynamically generating SQL queries to fetch matching results from the MySQL database. The flexibility of PHP made it easy to process incoming HTTP requests, parse JSON payloads, and respond with structured data that could be directly consumed by the frontend.

The language's lightweight execution was particularly well-suited to the shared hosting environment used during the internship. Hostinger, while affordable and easy to configure, comes with limited computing resources compared to dedicated or cloud-based servers. PHP’s efficient runtime and small memory footprint ensured that even under constrained resources, API requests were served quickly and reliably without timeouts or overloads. This proved essential during development and testing when multiple requests were being made from different devices and emulators simultaneously.

To further enhance backend functionality, PHP was integrated with third-party libraries like PHPMailer for email services. This allowed the apps to send verification and transactional emails, such as confirmation messages or password reset instructions. These features added a layer of professionalism and usability to the apps while keeping the backend code modular and maintainable.

2) RESTful APIs

The backend of the JEE Saral and NEET Saral applications was structured around RESTful API principles, ensuring a modular and standardized way for the frontend to interact with the server. These APIs followed widely accepted HTTP conventions such as GET, POST, and DELETE, which not only aligned with best practices in full-stack development but also enhanced interoperability and maintainability of the codebase. Each endpoint was dedicated to a specific task and was implemented with a clear separation of concerns, which significantly improved the readability and reusability of the server-side code.

For instance, login.php was responsible for verifying user credentials by cross-checking input data against encrypted values stored in the MySQL database. Once validated, a session token or a success flag was returned in a standardized JSON format. Similarly, register.php managed user onboarding, including storing user data securely and triggering email verification through PHPMailer integration. These discrete endpoints made it easy to debug issues, add new features, or extend existing functionality without disrupting the overall architecture.

Another core endpoint, filter.php, was designed to retrieve college information based on dynamic user-defined filters such as state, quota, category, and exam rank. It accepted parameters through POST requests, executed structured SQL queries, and returned well-formatted JSON arrays. This approach not only allowed for real-time filtering but also enabled a highly personalized and responsive user experience.

To ensure data consistency and security, input validation and sanitization were implemented at each endpoint. This reduced the risk of SQL injection, malformed requests, and other vulnerabilities. The consistent use of JSON for all responses allowed the frontend—developed in React Native—to easily parse data and update the UI accordingly, reducing the overhead in data transformation logic.

Furthermore, the API layer was built to be stateless, meaning each request was handled independently with no reliance on prior interactions. This design allowed the backend to scale more effectively and simplified future deployment to cloud-based environments if needed. The architecture also supported asynchronous operations, making it compatible with modern frontend libraries like Axios and the Fetch API, which rely on promises for smooth data fetching and error handling.

3) PHPMailer for Email Services

PHPMailer was integrated for sending user verification emails, password reset links, and confirmation messages. SMTP settings were configured to ensure email deliverability, and the use of PHPMailer allowed advanced configurations like HTML email templates and multi-recipient handling.

Fig 2.5: Backend Technologies

Database Management

MySQL served as the backbone of the application's data storage and management system. As a widely adopted relational database management system (RDBMS), MySQL was chosen for its proven reliability, robust feature set, and seamless integration with PHP. Its compatibility with shared hosting services like Hostinger also made it a practical choice for deploying a live application environment within the constraints of the internship project. The use of MySQL ensured that data operations were not only efficient but also secure and maintainable.

The database schema was carefully structured to handle multiple categories of data, including user registration details, login credentials, application session states, and a large volume of college-related data. Each table was normalized to reduce data redundancy and maintain consistency throughout the dataset. For example, rather than duplicating college names across records, unique IDs were assigned and referenced through foreign keys. This approach allowed for more efficient data updates and ensured referential integrity between related tables.

A key component of the system involved filtering and sorting large datasets based on user-defined parameters such as entrance exam rank, category (General, OBC, SC/ST, etc.), quota (All India, State, NRI), and seat type. To enable high performance in such scenarios, indexing strategies were employed on frequently queried columns. This significantly reduced query execution time and enhanced the responsiveness of the mobile applications. Users were able to retrieve tailored college lists in real-time, despite the complexity of the filtering logic in the backend.

Security was another crucial consideration in the integration of MySQL. All interactions between the PHP backend and the database were performed using prepared statements. This mitigated the risk of SQL injection attacks, a common security vulnerability in database-driven applications. Prepared statements ensured that user-supplied input was safely parameterized before being executed, adding an essential layer of protection to the authentication and data retrieval workflows.

In addition to performance and security, efforts were made to maintain the scalability of the database. The structure was designed to accommodate potential growth in both the number of users and the volume of college data over time. New fields or relational mappings could be added without disrupting the existing architecture, thanks to the use of a modular table design and normalized data relationships.

Communication Layer

Axios and Fetch APIs: On the frontend, Axios and native fetch APIs were employed for making asynchronous HTTP requests. Axios, with its promise-based structure and automatic JSON parsing, was used primarily in complex request flows like registration and filtering. For simpler GET requests, fetch provided a lightweight alternative. Both ensured seamless integration between the frontend UI and backend services.

Headers such as Content-Type and Authorization were dynamically set during requests to maintain secure communication, especially for authenticated routes.

Development & Deployment Tools

1) Expo Go & EAS Build

Expo Go played a vital role throughout the development lifecycle of the JEE Saral and NEET Saral applications. As a companion app for Expo-based projects, it enabled seamless real-time testing across a variety of Android devices without requiring complex builds. During the early stages of development, this tool significantly sped up the feedback loop by allowing developers to preview changes instantly on physical devices simply by scanning a QR code. This feature proved especially helpful during UI development and component testing, where visual iteration was crucial.

As the applications matured and reached a more stable phase, the deployment process transitioned to EAS Build (Expo Application Services). EAS Build allowed for the generation of signed APK (Android Package) files suitable for submission to the Google Play Store. Unlike traditional native Android development, which requires setting up Android Studio, SDKs, and emulators, EAS Build offered a fully cloud-based solution. Developers could trigger builds via command-line tools, specify app configurations using eas.json, and automatically generate production-ready artifacts. This drastically simplified the release process and removed the dependency on heavyweight local development environments.

One major advantage of using EAS Build was the ability to automate multiple aspects of the deployment pipeline. Environment variables could be managed securely through the Expo dashboard, build profiles could be customized for development, preview, or production modes, and assets were bundled and optimized during the cloud build process. The consistency of builds, regardless of the developer's local system, helped maintain uniformity in release quality and reduced the potential for platform-specific bugs.

Furthermore, EAS Build supported app signing and versioning, which are critical steps for publishing on the Google Play Store. The process included the creation and management of keystores for secure signing of APKs, as well as auto-incrementing version codes to comply with Play Store requirements. With these automated features

in place, the team could confidently release updates, knowing that versioning and signing were handled reliably.

After generating the APKs, the final step involved uploading the build to the Google Play Console. Here, metadata such as app descriptions, icons, screenshots, and privacy policies were configured. The platform also allowed internal testing through tracks like "Alpha" and "Beta," making it easier to collect feedback before going live. Once approved, the apps were published and became available to users across India.

2) Hostinger

As part of the deployment process, Hostinger was chosen as the primary hosting platform for the backend services, owing to its affordability, reliability, and beginner-friendly interface. The backend, built using PHP and MySQL, was seamlessly deployed on a shared hosting plan, which provided the necessary features to host API scripts, manage a relational database, and enable external client connections from the mobile applications.

One of Hostinger's biggest advantages during the internship was its intuitive cPanel interface, which allowed easy management of all backend components without requiring complex DevOps setups. Through this control panel, I was able to manage the server-side environment, upload project files, set environment variables, manage domains and subdomains, and monitor server resource usage. The platform also supported custom cron jobs, which could be scheduled to perform periodic tasks such as cleaning logs or triggering batch email notifications via PHP scripts.

For deploying code, FTP (File Transfer Protocol) was used, allowing me to upload and update the backend PHP scripts directly from my development machine. Hostinger's FTP access was secured with credentials and supported file permission configurations, helping maintain code integrity and access control. This allowed quick iteration and testing of features, especially during the API integration phase with the frontend mobile apps.

Database management was carried out using phpMyAdmin, a web-based graphical interface for MySQL. This tool was particularly helpful for importing large datasets of college information, writing SQL queries for testing API responses, managing user

tables, and monitoring database performance. phpMyAdmin also enabled me to quickly debug query issues and validate data integrity without writing scripts from scratch.

Despite being a shared hosting platform (as opposed to a dedicated or cloud server), Hostinger met the performance demands of the project. With optimized PHP scripts, proper caching, and efficient MySQL queries, the server was able to respond to client requests quickly and consistently. Response times remained well within acceptable limits, even when multiple users were filtering large college datasets simultaneously.

Hostinger also allowed configuration of essential backend services like SMTP, which was crucial for integrating PHPMailer into the system. By configuring SMTP credentials through Hostinger's mail settings, I enabled secure and authenticated email delivery for user verification and notifications, without needing third-party email APIs.

3) Google Play Console

Publishing mobile applications on the Google Play Store is a critical phase in the development lifecycle, turning a locally developed app into a widely accessible product. During the internship, a major part of the deployment process involved preparing and releasing both JEE Saral and NEET Saral on the Google Play Store. This process required not only technical configuration but also compliance with Google's app policies and user experience standards.

The first technical prerequisite for Play Store deployment is app signing. This involves generating a secure keystore file and signing the APK or AAB file with it before upload. This cryptographic signature verifies the integrity and ownership of the application, ensuring that future updates can only be pushed by the original developer. For the internship apps, Expo's EAS Build service was used to manage app signing in a secure and automated manner, ensuring compliance with Google's modern AAB (Android App Bundle) requirements.

Creating an engaging and informative Play Store listing was also essential. Each app required the preparation of a complete metadata package, including the app name, short and long descriptions, category selection, and keywords to enhance discoverability. Additionally, a set of high-resolution screenshots demonstrating key features such as search functionality, college listings, and filter options were designed and uploaded. These assets help users understand the app's purpose and interface before installation.

A vital part of Play Store publishing is ensuring transparency around user data and privacy. As both applications collected minimal user data (mainly for authentication and preferences), privacy policies were written and hosted externally, with links included in the Play Console under the "App Content" section. The data safety section was carefully completed to reflect what data is collected, whether it is shared, and how it is secured. Age-based content ratings were configured through Google's questionnaire, resulting in an appropriate rating that allowed distribution to the target student demographic.

Each new release of the application was linked to a unique versionCode (an incrementing integer used by Android to identify new versions) and versionName (a string shown to users, such as "v1.0.1"). Careful versioning ensured that updates could be delivered without conflict and users could benefit from bug fixes or feature enhancements. During the internship, this helped support rapid iteration—multiple builds were tested and updated during beta phases before reaching production status.

After deployment, the Play Console served as a central dashboard for managing and monitoring app performance. It provided real-time data on downloads, user reviews, device compatibility, and crash analytics. This post-launch phase was critical for identifying edge-case bugs, analyzing user behavior, and planning future updates. Tools such as ANR (Application Not Responding) reports and stack traces were used to diagnose and fix stability issues across different Android versions and devices.

Before releasing to a broader audience, both apps were initially published to internal and closed testing tracks. This allowed internal testers, mentors, and stakeholders to

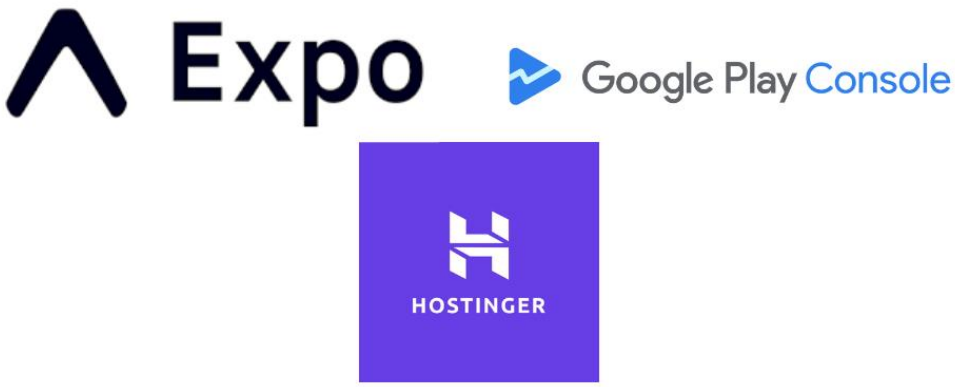


Fig 2.6: Development and Deployment Tools

review the application and provide feedback. These closed test phases were invaluable in catching usability issues, UI inconsistencies, and bugs that may have been missed during local testing.

Integration and Workflow

The success of the JEE Saral and NEET Saral mobile applications was not solely dependent on the selection of individual technologies, but also on how well those technologies were integrated and managed throughout the development lifecycle. Smooth coordination between frontend, backend, and database layers was achieved through clear architectural planning, consistent communication protocols, and disciplined development practices.

A key aspect of seamless integration was the use of consistent and predictable RESTful API schemas. The endpoints were designed to return structured JSON responses with standard status codes, making it easier for frontend developers to implement error handling and data parsing. Input validation, query filtering, and pagination were managed server-side in PHP, which kept client-side logic clean and focused on presentation and interaction.

The APIs were built with modularity in mind—each endpoint was designed to handle a specific task such as user login, college search, or data filtering. This approach reduced the complexity of debugging and future maintenance, while also enabling reusability across different components of the application.

Git was at the heart of the project's version control strategy. The entire codebase was hosted on GitHub, which enabled efficient collaboration between developers and mentors. Each new feature or bug fix was implemented in a separate branch and merged into the main branch using pull requests. This not only ensured code review before integration but also provided a history of changes that made rollback or auditing simple.

GitHub Issues were used to track bugs, enhancement requests, and development tasks. Milestones and labels helped in organizing sprints, prioritizing tasks, and maintaining a clear roadmap. This created a sense of accountability and transparency throughout the internship project.

Development followed an iterative and incremental model, where new features were introduced in small, manageable chunks. After writing code, the changes were tested

on physical and virtual devices using Expo Go, which provided a near-instant feedback loop. This rapid testing ability helped identify layout shifts, functionality glitches, or platform-specific inconsistencies early in the development cycle.

Features such as college filters, admission cut-off data rendering, and authentication logic were first prototyped in controlled branches. Once verified, they were merged into the main branch and included in production-ready builds generated via EAS Build. This iterative cycle ensured that the application remained stable and continuously improved over time.

All layers of the application—from frontend to backend to database—worked in harmony due to the enforced discipline around interface contracts and data structures. Axios (on the frontend) and PHP (on the backend) were connected via clearly defined REST endpoints, ensuring that input/output expectations were met on both sides.

Fig 2.7: Integration Tools

The result was a robust, reliable, and user-friendly mobile application that allowed students to explore, search, and analyze college admission data intuitively. By filtering based on location, branch, and rank range, students could identify suitable colleges with ease. The app's responsiveness and performance stemmed directly from this strong foundation of clean integration, collaborative development, and disciplined workflow.

2.4 Tools & Platforms

The successful development and deployment of the JEE Saral and NEET Saral applications were made possible through the use of various tools and platforms that streamlined the development workflow, improved testing efficiency, and enabled real-world distribution of the apps. This section details the core platforms used throughout the internship, highlighting their purpose and impact.

Expo (React Native Platform)

Expo played a pivotal role in the development of the JEE Saral and NEET Saral applications during the internship. It is an open-source framework and platform built around React Native that abstracts much of the underlying complexity involved in building mobile applications for both Android and iOS. By using Expo's managed workflow, the development team was able to focus more on application logic, UI/UX, and feature implementation rather than getting entangled with native build configurations, dependency linking, or environment-specific bugs.

One of the standout advantages of using Expo was its ability to streamline the end-to-end development and deployment process. From setting up a new project to testing on real devices and deploying to the Play Store, Expo offered a complete toolchain that significantly accelerated development time and reduced overhead.

Key Benefits and Features of Expo

1. Expo Go App – Real-Time Device Testing

The Expo Go app, available on the Play Store, allowed developers to instantly test their application on physical devices by simply scanning a QR code. This eliminated the need for building APKs manually or configuring USB debugging. As a result, changes made to the code were reflected in real-time, making the development cycle highly interactive and efficient. This feature was particularly useful in a fast-paced internship environment, where rapid iteration and feedback were crucial.

2. File-Based Routing with Expo Router

Using Expo Router, a file-based routing system, navigation between screens was drastically simplified. Instead of configuring navigation manually with stack and tab navigators, routes were automatically derived from the directory structure. This not

only improved code readability but also made scaling the project easier, especially as more screens and components were added during the internship.

3. Asset and Font Management

Expo provided built-in support for managing assets such as custom fonts, icons, and splash screens. Integrating these elements was as straightforward as placing them in the designated project folders and referencing them in the configuration files. This allowed the app to maintain a polished and cohesive brand identity, without the need for third-party plugins or additional linking steps typically required in bare React Native setups.

4. EAS Build and Submit Workflow

Expo's EAS (Expo Application Services) Build and Submit tools were essential for producing release-ready APK and AAB files. Unlike the older expo build command, EAS supports modern app bundles and native code configuration when needed. The internship made use of EAS Build for creating production APKs and submitting them directly to the Google Play Store using EAS Submit. This end-to-end CI/CD-style deployment process minimized errors and ensured consistency between local testing and production releases.

5. OTA Updates and Development Tools

Expo also supports Over-the-Air (OTA) updates, allowing bug fixes and small updates to be pushed to users without requiring a full Play Store update. This capability was explored during the internship for minor patches and UI improvements. Additionally, tools like expo-dev-client and development menus enhanced debugging capabilities, enabling smooth interaction with logs, network calls, and asynchronous storage.

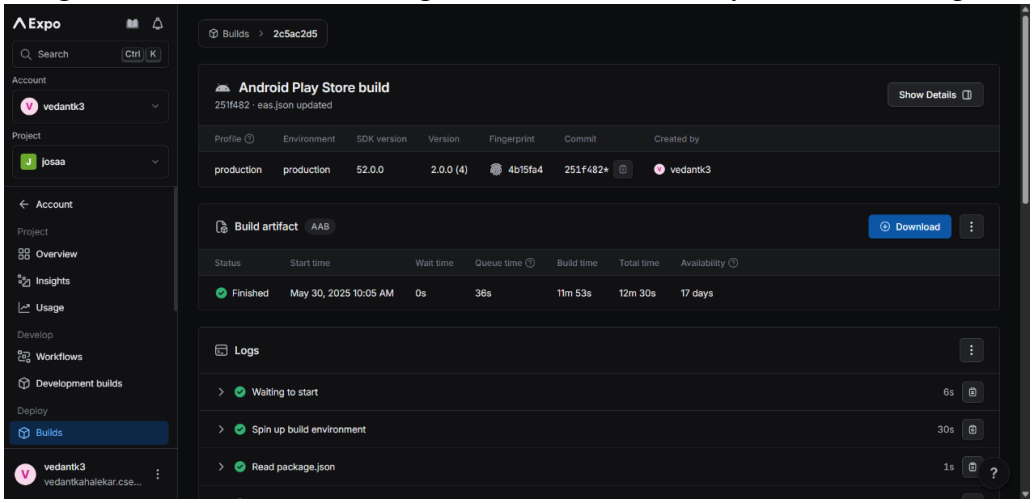


Fig 2.8: Expo EAS Dashboard

Hostinger

A critical component of the development and deployment infrastructure for the JEE Saral and NEET Saral applications was the backend hosting platform. For this purpose, Hostinger was selected based on a combination of cost-effectiveness, compatibility with PHP, ease of setup, and robust tools for managing web applications and databases. Its developer-friendly interface and support for essential web technologies made it an ideal choice during the internship period, especially for rapid prototyping and production deployment.

The decision to go with Hostinger stemmed from its strong support for LAMP stack technologies (Linux, Apache, MySQL, PHP), which aligned perfectly with the backend architecture of the applications. Hostinger offered shared hosting plans that included features like PHP version management, built-in MySQL database creation, and SSH/FTP access—all of which enabled quick backend setup without requiring extensive DevOps experience. Its affordability was also a factor, making it accessible for internship-scale projects where enterprise-grade hosting might have been overkill.

During the internship, multiple PHP scripts were created to support core backend functionality. These included authentication endpoints such as login.php and register.php, which processed user input, validated credentials, and returned appropriate status messages to the mobile apps. Additionally, data-fetching endpoints like getusers.php and filter.php were hosted on the same server, allowing the frontend to query user or college data dynamically based on search filters, categories, or user input.

These PHP scripts acted as the API layer, accepting HTTP requests from the frontend and returning JSON responses. Hostinger's cPanel File Manager and integrated FTP system were used to upload and maintain these scripts, enabling efficient iteration during development cycles. Quick file updates and instant accessibility from the mobile app allowed for a smooth development workflow.

The backend relied on a MySQL database hosted on Hostinger to store persistent data such as user accounts, authentication tokens, college listings, and metadata related to courses and streams. The MySQL management interface provided by Hostinger's control panel allowed the creation of tables, manual insertion of test data, and running of SQL queries for debugging purposes.

Security measures such as password hashing and prepared statements were implemented in the PHP scripts to ensure safe communication between the database and frontend. Regular backups and versioning of the database schema were also performed to minimize data loss during updates.

One of the added backend features implemented was automated email delivery for account verification, password reset, and notifications. To enable this, the PHPMailer library was integrated into the PHP backend. Hostinger allowed SMTP configuration with ease, providing details such as SMTP host, port, and authentication credentials.

Once configured, PHPMailer was used to send emails through the SMTP server, making it possible to deliver verification links to users immediately after registration. This helped simulate a real-world, production-grade user authentication system and gave experience in integrating third-party services with backend logic.

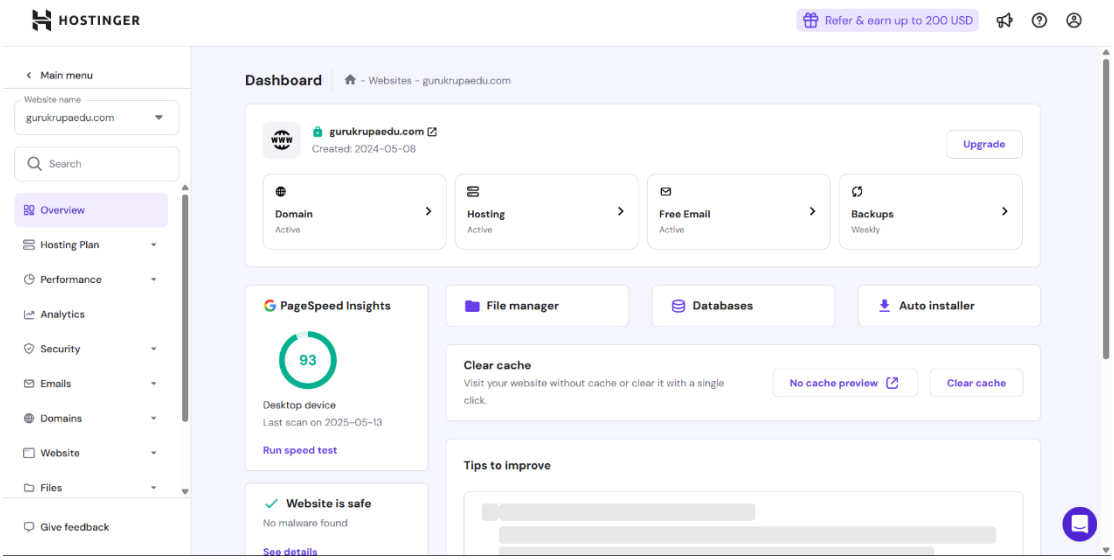


Fig 2.9: Hostinger Dashboard

Google Play Store

The ultimate goal of the mobile app development lifecycle is to make the application accessible to its intended users, and for Android platforms, the Google Play Store serves as the primary channel for app distribution. During the internship, both JEE Saral and NEET Saral were successfully published on the Google Play Store, marking a major milestone in the project. The Play Store is not only a hosting platform but also a suite of tools that ensures secure distribution, efficient updates, user analytics, and compliance with global policies on privacy and safety.

The process began by registering a Google Play Developer Account, which is a one-time paid registration that allows developers to publish apps under their own branding. This step was essential to create a formal presence for both applications and access Play Console functionalities. After account setup, separate app listings were created for JEE Saral and NEET Saral, with distinct package names, version control, and product metadata.

After development and testing phases, the applications were packaged into Android-compatible formats (APK/AAB) using EAS Build, a modern cloud-based build service provided by Expo. The builds were signed with secure keys to verify authorship and enable future version updates. These signed APKs were then uploaded through the Play Console interface, where build artifacts are validated, scanned for vulnerabilities, and prepared for release.

To make the apps visually appealing and informative on the Play Store, several visual and textual assets were prepared and uploaded. These included:

- High-resolution app icons optimized for various screen densities
- A set of in-app screenshots showcasing features like college filters, user login, and ranking views
- Feature graphics and banners for improved discoverability on certain devices
- Clear and concise title, short description, and full description fields to communicate the value proposition of each app
- Together, these elements created a professional and polished store listing that could attract and retain users at first glance.

Compliance with user data regulations is a core requirement for publishing on the Google Play Store. For both apps, externally hosted privacy policies were drafted and linked in the Play Console. These documents outlined what data is collected, how it is stored, and how user rights are protected.

The Data Safety section was carefully completed to reflect real app behavior, specifying whether personal data such as login credentials, device IDs, or analytics were collected or shared. Additionally, a Content Rating Questionnaire was submitted to generate an appropriate age-based content rating, ensuring the app would be available to its intended audience (primarily students and educators).

Once all required fields were filled and builds were uploaded, the apps were submitted for Google Play Review. This review process involved automated and manual checks for malware, policy violations, and user experience standards. After a short evaluation period, both JEE Saral and NEET Saral were approved and published, making them available for download by users across supported Android devices.

Post-deployment, the Play Console served as a centralized hub for managing app lifecycle. Features such as versioning, staged rollouts, crash analytics, and user reviews provided critical insights for continuous improvement. Future updates could be published seamlessly by uploading a new version with an incremented versionCode, allowing real-time fixes and feature upgrades without disrupting the user base.

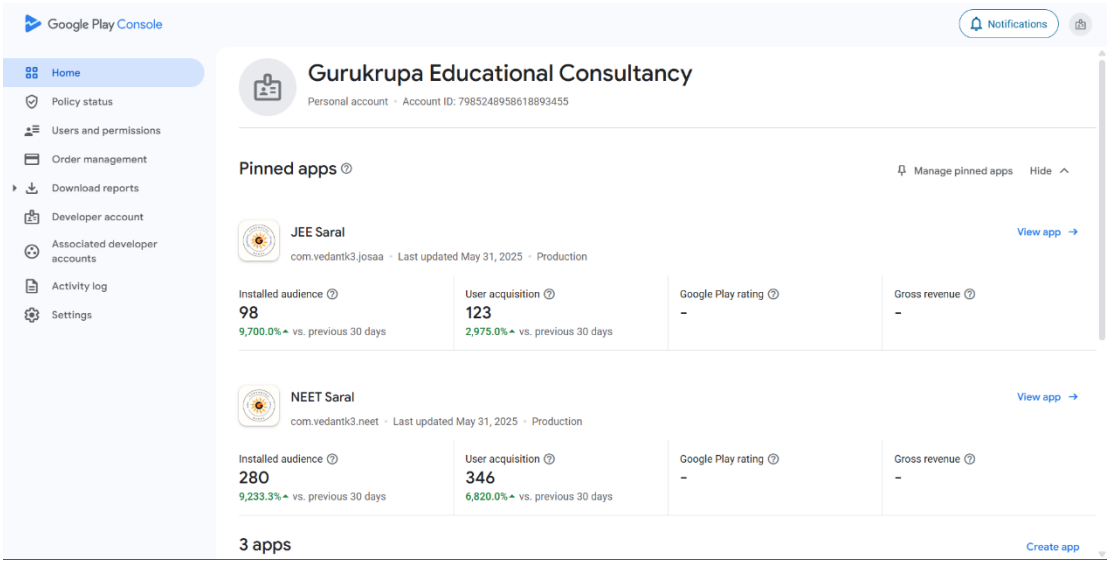


Fig 2.10: Google Play Console

2.5 APIs & Data Flow Diagram

At the core of both JEE Saral and NEET Saral applications is a robust backend system powered by a set of custom-built RESTful APIs, written in PHP and connected to a MySQL relational database. These APIs serve as the vital communication channel between the mobile frontend—developed using React Native—and the backend database, orchestrating all critical data flows throughout the application lifecycle.

Architecture and Core Functionality

The backend was designed using a modular, endpoint-driven architecture, with each PHP file acting as a microcontroller for a specific function. The architecture ensured

clean separation of concerns, allowed for easier maintenance, and made it simple to debug or update specific functions without affecting the entire system.

These APIs were responsible for executing a variety of essential operations such as user registration, login authentication, college list generation, filtering based on academic scores, and more. By following the REST architecture style, each endpoint used conventional HTTP methods like GET and POST, and returned responses in lightweight JSON format, ideal for consumption by mobile devices over varying network conditions.

Endpoint	**Method**	**Purpose**
register.php	POST	Registers a new user, validates input, stores in DB
login.php	POST	Authenticates user credentials and returns session/token
filter.php	POST	Filters college list based on user rank, quota, etc.
getusers.php	GET	Fetches all or filtered user/college data
smtp_mailer.php	POST	Sends email via SMTP for verification or alerts

Table 2.1: Key API Endpoints and their roles

Each endpoint was developed with clarity and performance in mind. Efficient SQL queries and appropriate indexing were used to minimize load times, ensuring that mobile users could fetch large datasets—such as nationwide college listings—without experiencing delays.

API Communication Flow

The integration between the frontend and backend followed a clear and predictable flow:

1) User Interaction: The user initiates an action from the app, such as entering their NEET or JEE rank and selecting specific filters like state, caste category, or gender.
2) API Request: The app triggers an HTTP request (using Axios or Fetch API) to the appropriate backend endpoint—such as filter.php—sending the input data as JSON payload.

3) Backend Processing: On receiving the request, the PHP script validates the input, performs SQL queries on the MySQL database, and compiles the result based on conditions.
4) Response Formatting: The data is packaged into a JSON array or object and returned with appropriate status codes (200 for success, 400 for invalid input, etc.).
5) Frontend Rendering: The app parses the response and dynamically renders a user-friendly list of colleges with relevant details like location, cut-off rank, and quota eligibility.

This decoupled communication system made the apps modular, scalable, and easy to maintain. Additional endpoints can be integrated seamlessly without restructuring the entire backend.

Security and Validation

While the backend system was lean and optimized, it did not compromise on essential security measures. Important safeguards included:

Input Sanitization: All incoming data was sanitized using mysqli_real_escape_string and similar techniques to protect against SQL injection attacks.

Field Validation: Both the frontend and backend performed checks to ensure required fields were present, correctly formatted, and within expected ranges.

Email Verification: The smtp_mailer.php script, using PHPMailer, sent verification emails to users during the sign-up process to validate their identity and confirm their intent.

Authentication Planning: Although initial versions used basic session-based validation, plans were discussed to implement JWT (JSON Web Tokens) for stateless and more secure authentication in future updates.

Data Flow Diagram

A Data Flow Diagram (DFD) is a visual representation of how data moves through a system, highlighting the flow of information between components such as the user interface, backend server, and database. For the JEE Saral and NEET Saral applications,

the DFD illustrates how data is requested, processed, and returned during a typical user interaction.

In our case, the data flow starts from the user, who inputs information such as exam rank, category, and quota into the React Native application. This data is sent to the backend API, where a corresponding PHP script processes the request and interacts with the MySQL database to retrieve relevant college data. Once the data is filtered and formatted, it is sent back as a JSON response to the frontend app, which then renders the appropriate results for the user.

This unidirectional yet cyclical flow ensures clarity, security, and performance by keeping each component's role clearly defined. It also supports scalability, allowing new features or endpoints to be integrated without disrupting the existing data pipeline.

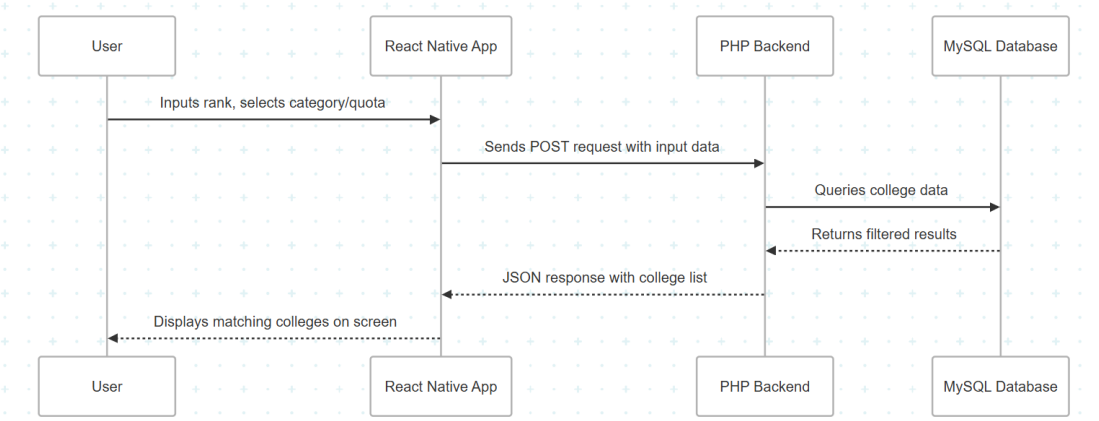


Fig 2.11: Data Flow Diagram

Chapter 3

DESIGN AND DEVELOPMENT PROCESS

3.1 Requirement Gathering and Analysis

The first and one of the most crucial phases of any software development lifecycle is requirement gathering and analysis. It lays the foundation for understanding the purpose of the application, defining the expected features, and identifying the end-users' needs. During my internship at Gurukrupa Educational Consultancy, this process involved a combination of discussions with mentors, feedback from users, and analysis of existing tools used by students preparing for JEE and NEET exams.

Stakeholder Discussions

To begin the process, I engaged in conversations with:

- The internal team at Gurukrupa Educational Consultancy, who provided insights into what students commonly struggle with during the college admission phase.
- Mentors and coordinators who outlined the core expectations from both JEE Saral and NEET Saral apps.
- Potential end-users (students and parents) through informal feedback, helping to define usability goals.
- These interactions helped shape the key features like rank-based search, quota-specific filtering, motivational content, and access to consultancy resources.

Functional Requirements Identified

During the planning and design phase of the JEE Saral and NEET Saral applications, several core functional requirements were identified to ensure the platforms effectively meet the needs of students navigating the complex college admission process. These requirements were derived from user research, discussions with consultants, and analysis of existing tools in the education domain.

1. User Registration and Login: A secure and user-friendly authentication system was considered fundamental. The applications required functionality for user registration—allowing new users to create an account using their email or phone number—and login, which authenticated users and maintained secure sessions throughout the app

experience. Session tokens or cookies were used to persist login states and protect sensitive user data. Proper validation mechanisms, such as password strength checks and duplicate account prevention, were also implemented.

2. College and Branch Search: One of the key functionalities was enabling users to search and explore colleges and specific branches based on multiple criteria. Users could input their exam rank (JEE or NEET), category (e.g., General, OBC, SC/ST), and quota type to retrieve a list of eligible colleges. The filtering system had to be responsive and dynamic, taking real-time user inputs and updating the results accordingly, ensuring users received relevant data without delay.

3. Detailed College View: For better decision-making, each college listing included a dedicated detail screen. This view presented comprehensive information such as opening and closing ranks, available branches, tuition fees, counseling rounds, seat matrix, and admission procedures. The data was organized in a clean, readable format to allow students and parents to easily compare options and identify colleges that best matched their criteria.

4. Tab-Based Navigation for Quota Segmentation: Recognizing the complexity of quota systems in Indian entrance exams, the app featured tab-based navigation for various admission quotas. Tabs were created for All India Quota (AIQ), Deemed/NRI quota, Maharashtra State quota (MH State), and Open category admissions. This separation allowed users to easily switch between different admission categories and view relevant colleges under each quota system without confusion or data overlap.

5. Intelligent Backend Filtering: The core of the search and recommendation engine relied on a powerful backend filtering logic. This logic was implemented via RESTful APIs and was capable of processing numerous variables in real time. When a user submitted a search query, the backend would validate the inputs, construct optimized SQL queries, and return only those colleges that matched the user's exact academic profile and preferences. This helped ensure accuracy, relevance, and performance.

6. Google Maps Integration: To further enhance the app's usability for students looking to visit consultancy offices in person, an integration with Google Maps API was implemented. This feature displayed the physical location of the consultancy office directly within the app, complete with navigation support.

Non-Functional Requirements

- Cross-platform compatibility (Android & Web).
- Fast loading time and smooth user experience on low-end devices.
- Clean and intuitive user interface with consistent theming.
- Responsive design to handle multiple screen sizes.
- Secure handling of user data and input validation on both ends.

Tools Used for Requirement Analysis

- Google Docs/Sheets for listing and revising feature sets.
- Wireframing tools like Figma for visual mockups.
- Postman for testing early API responses during backend planning.

11	6	Build logic for login and register functionality	Completed
12	7	Work on backend API logic (PHP + MySQL)	Completed
13	8	Start building the Rankwise screen layout	Completed
14	9	Meet supervisor to discuss app progress	Completed
15	10	Complete the Branchwise screen	Completed
16	11	Fix CORS issue in PHP backend (index.php, filter.php)	Completed
17	12	Complete the Collegewise screen	Completed
18	13	Build College Details screen component	Completed
19	14	Start filter feature logic for Rank and Branch pages	Completed
20	15	Improve logic on the Rank page	Completed
21	16	Improve logic on the Branch page	Completed
22	17	Design and code the Home page	Completed
23	18	Improve design of login and register pages	Completed
24	19	Make UI consistent across all screens	Completed

Fig 3.1: My Task List

This analysis phase enabled clear goal-setting, avoided scope creep, and ensured that every development effort aligned with solving real-world user problems.

3.2 UI/UX Design Strategy

User Interface (UI) and User Experience (UX) design play a vital role in determining how effectively users can navigate an application and achieve their goals. Given that the target users of JEE Saral and NEET Saral include students and parents with varying levels of tech familiarity, the design strategy focused on simplicity, clarity, and responsiveness.

Design Goals

The primary goals of the UI/UX design were:

- To create a clean and minimalistic layout that reduces user cognitive load.
- To ensure a mobile-first approach, optimizing screens for small devices.
- To provide a consistent navigation experience using tabs and intuitive icons.
- To implement interactive feedback through loading indicators, empty state messages, and visual cues.
- To maintain a consistent color palette and typography across the apps.

Wireframing and Mockups

Early-stage wireframes were created using Figma, outlining the core screens:

- Splash Screen and Login/Register
- Home Dashboard
- Rank Predictor Form
- College and Branch Search Filters
- College Detail View
- Informational Sections (About, Disclaimer, Contact)

These mockups helped align the development process with visual expectations.

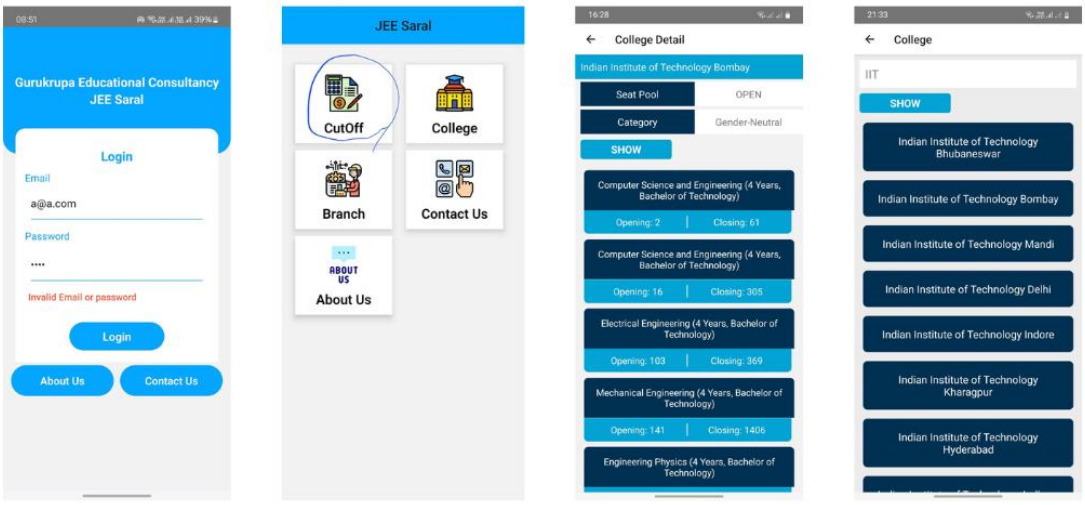


Fig 3.2 Initial App Design

Design Elements and Libraries Used

- **React Native Paper**: For standard UI components like buttons, cards, and text inputs with material design principles.
- **@expo/vector-icons**: To display recognizable icons for actions like search, home, info, and filters.
- **Custom Fonts and Branding**: Fonts were loaded using Expo's expo-font and applied consistently throughout the app.
- **Splash Screen and Adaptive Icons**: Configured using app.json for a polished startup experience and consistent branding.

Mobile Responsiveness

The UI was tested on devices with different screen sizes to ensure scalability. React Native's flexible styling system allowed conditional styling for tablets and small phones, avoiding layout breaks or overlapping elements.

UX Enhancements

- **Search and Filter Inputs**: Dropdowns and pickers were used to minimize typing and reduce errors.
- **Real-Time Feedback**: Loading animations, toast messages, and error prompts improved user feedback.
- **Navigation Structure**: File-based routing made it easier to maintain and update screens without disrupting the flow.

The design strategy successfully balanced functionality with usability, resulting in an experience that felt intuitive even for first-time users.

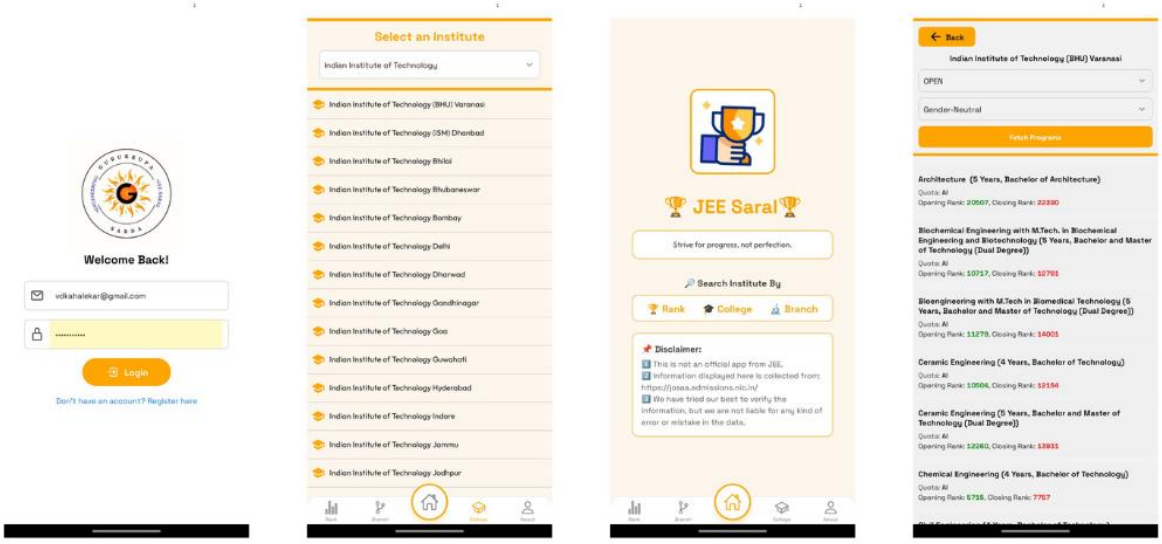


Fig 3.3 Final UI Design JEE Saral

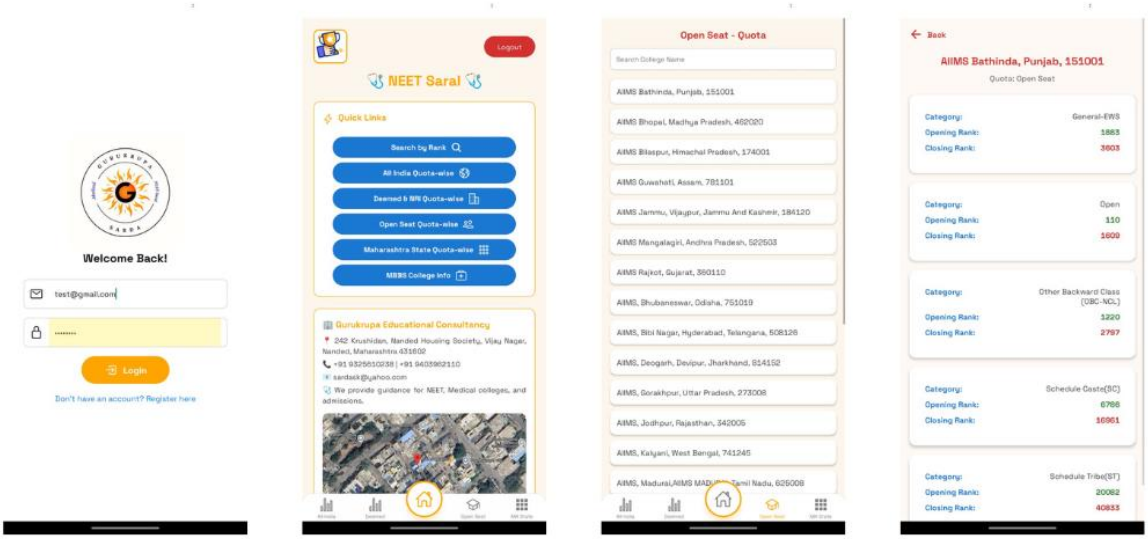


Fig 3.4 Final UI Design NEET Saral

3.3 Backend API Development

The backend of both JEE Saral and NEET Saral applications was built using PHP, with a focus on creating lightweight, modular, and scalable RESTful APIs. These APIs acted as the bridge between the frontend app and the MySQL database, handling user data, rank-based filtering, and college information retrieval.

Development Approach

A modular, file-based structure was followed, where each API endpoint was separated into a dedicated PHP file. This structure made debugging and deployment straightforward, especially on a shared hosting environment like Hostinger.

File Structure:

- register.php – Registers new users into the database.
- login.php – Authenticates users and returns a session or response.
- filter.php – Handles input for rank, category, and quota; fetches matching colleges.
- getusers.php – Fetches full or filtered user/college records.
- smtp_mailer.php – Sends emails using PHPMailer.

Authentication System

A simple username/password login system was implemented using the login.php and register.php scripts. User credentials were stored securely in the database, and each login request was validated against the stored values.

```php
if (!empty($obj['email']) && !empty($obj['password'])) {
    $email = $obj['email'];
    $password = md5($obj['password']); // You can switch to password_hash() later for more security

    $query = sprintf(
        "SELECT status, name, neet_no FROM neet_user WHERE email='%s' AND password='%s' LIMIT 1",
        $mysqli->real_escape_string($email),
        $mysqli->real_escape_string($password)
    );

    $result = $mysqli->query($query);

    if ($result->num_rows == 0) {
        http_response_code(401);
        $response['Error'] = 'Invalid credentials';
    } else {
        $user = $result->fetch_assoc();

        if ($user['status'] == 0) {
            http_response_code(403); // Forbidden
            $response['Error'] = 'Your account is not approved. Please contact admin.';
        } else {
            http_response_code(200);
            $response['user']['Status'] = true;
            $response['user']['name'] = $user['name'];
            $response['user']['neet_no'] = $user['neet_no'];
            $response['Success'] = true;
        }
    }
} else {
    http_response_code(400);
    $response['Error'] = 'Missing email or password';
}
```

Fig 3.5 Login.php Code Snippet

Database Design and Queries

- **Database**: MySQL hosted on Hostinger.
- **Tables**: Included user data, college records, cutoff ranks, and program details.
- **Query Structure**: Used SELECT, JOIN, and WHERE clauses to efficiently filter colleges based on multiple parameters such as rank, category, quota, and state.

All API responses were sent in JSON format, making them easy to parse on the client side. Sample structure:


```
[
  {
    "id": "1",
    "institute_name": "Indian Institute  of Technology Bhubaneswar",
    "program_name": "Civil Engineering (4 Years, Bachelor of Technology)",
    "quota": "AI",
    "seat_type": "OPEN",
    "gender": "Gender-Neutral",
    "opening_rank": "9106",
    "closing_rank": "14782"
  },
  {
    "id": "2",
    "institute_name": "Indian Institute  of Technology Bhubaneswar",
    "program_name": "Civil Engineering (4 Years, Bachelor of Technology)",
    "quota": "AI",
    "seat_type": "OPEN",
    "gender": "Female-only (including Supernumerary)",
    "opening_rank": "18286",
    "closing_rank": "23024"
  },
  {
    "id": "3",
    "institute_name": "Indian Institute  of Technology Bhubaneswar",
    "program_name": "Civil Engineering (4 Years, Bachelor of Technology)",
    "quota": "AI",
    "seat_type": "EWS",
    "gender": "Gender-Neutral",
    "opening_rank": "2062",
    "closing_rank": "2267"
  },
```


Fig 3.6 API JSON Responses

3.4 Frontend Development

The frontend for both JEE Saral and NEET Saral applications was built using React Native under the Expo framework, which significantly accelerated cross-platform development for Android and web. The focus was on creating a responsive, fast, and intuitive interface that allowed users to navigate various features with ease.

Development Environment Setup

The foundation of the development process for both the JEE Saral and NEET Saral applications began with configuring a robust and scalable development environment. This phase was crucial for ensuring a smooth development lifecycle, facilitating faster iterations, and simplifying the testing and deployment processes.

To kick things off, the Expo CLI was installed and used to scaffold the React Native project. Expo, known for its managed workflow, proved to be an ideal choice due to its simplicity, powerful toolset, and wide community support. With just a few commands, a fully functional development workspace was ready, complete with default files, folder structures, and essential dependencies.

One of the most valuable features leveraged from Expo was hot reloading. This enabled real-time previewing of code changes without restarting the app or rebuilding the entire project. As developers made changes to the UI or logic, the updates were instantly reflected on connected devices or emulators. This significantly accelerated development speed and improved productivity, especially during UI design and testing phases.

Another major advantage of Expo's ecosystem was its streamlined configuration process. Customizing app elements like the splash screen, app icons, and status bar appearance was handled through the app.json configuration file. Permissions for features such as camera, notifications, and location services could be declared effortlessly, eliminating the need to manually edit native files such as AndroidManifest.xml or Info.plist. This greatly reduced setup complexity, especially for developers unfamiliar with native Android or iOS development.

When it came time to test and distribute the apps, EAS Build (Expo Application Services) played a key role. Using this cloud-based build service, Android APK files were generated directly from the source code, without requiring a local Android Studio installation. EAS Build also managed app signing and allowed for multiple build profiles (e.g., development, staging, production), which simplified testing and release cycles.

Fig 3.7 Expo CLI Running JEE Saral

Major Functional Components

- **Login & Registration**: Integrated with backend APIs using **Axios** to handle authentication and token storage via AsyncStorage.
- **Search & Filter Forms**: Implemented dropdowns (from react-native-element-dropdown) for selecting rank, quota, and category.
- **Tabs & Navigation**: Bottom tab layout allowed users to switch between quota types with visual clarity.
- **College Detail Views**: Presented filtered results with relevant cutoff data, college name, category, and quota info.

State Management

State was managed using built-in React hooks:

- useState() for local UI state.
- useEffect() for fetching and updating data from APIs.
- AsyncStorage for persistent session management (login tokens).

UI/UX Optimization

- Used ScrollView and FlatList for rendering long lists of colleges efficiently.
- Applied conditional rendering for loading indicators, error messages, and empty state prompts.
- Ensured responsive design by using dynamic dimensions and StyleSheet from React Native.

Code Reusability

Common components such as headers, buttons, and cards were extracted into the components/ folder, promoting clean code and DRY (Don't Repeat Yourself) principles.

This frontend setup ensured that the apps were not only functional but also visually clean, responsive, and ready for real-world usage with smooth navigation and reliable data handling.

3.5 Integration and Feature Implementation

The final stage in the development workflow involved integrating the backend APIs with the frontend interface and implementing core features in both JEE Saral and NEET Saral. While both apps shared a similar codebase and architectural pattern (React Native + Expo + PHP backend), their features and user journeys were uniquely tailored to their respective audiences: engineering aspirants for JEE Saral and medical aspirants for NEET Saral.

API Integration

- The app used Axios to make HTTP requests to the PHP APIs hosted on Hostinger.

- Authentication requests (login, registration) were routed to login.php and register.php.
- Filtering requests were routed to filter.php, where user rank, category, and quota preferences were processed server-side.

```
axios.post('https://gurukrupaedu.com/xxx/xxxxx.php', {
  rank: userInput.rank,
  category: selectedCategory,
  quota: selectedQuota
});
```

Fig 3.8: Example Axios call

JEE Saral Features Implemented

1. Rank-Based College Prediction

- Users input their JEE rank and apply filters like program, category, or quota.
- The app sends a request to the backend, which returns eligible engineering colleges based on cutoff data.

2. Branch-Wise Search

- A feature to search colleges offering a specific engineering branch (e.g., CSE, ECE, ME).
- The result screen displayed program-specific data like opening and closing ranks.

3. Motivational Dashboard

- Curated motivational quotes and affirmations displayed at login and on the home screen.
- Helped maintain a positive mindset for students during stressful entrance periods.

4. Consultancy & Maps Integration

- Access to Gurukrupa's consultancy services via a CTA.
- Integrated Google Maps to show the office location.

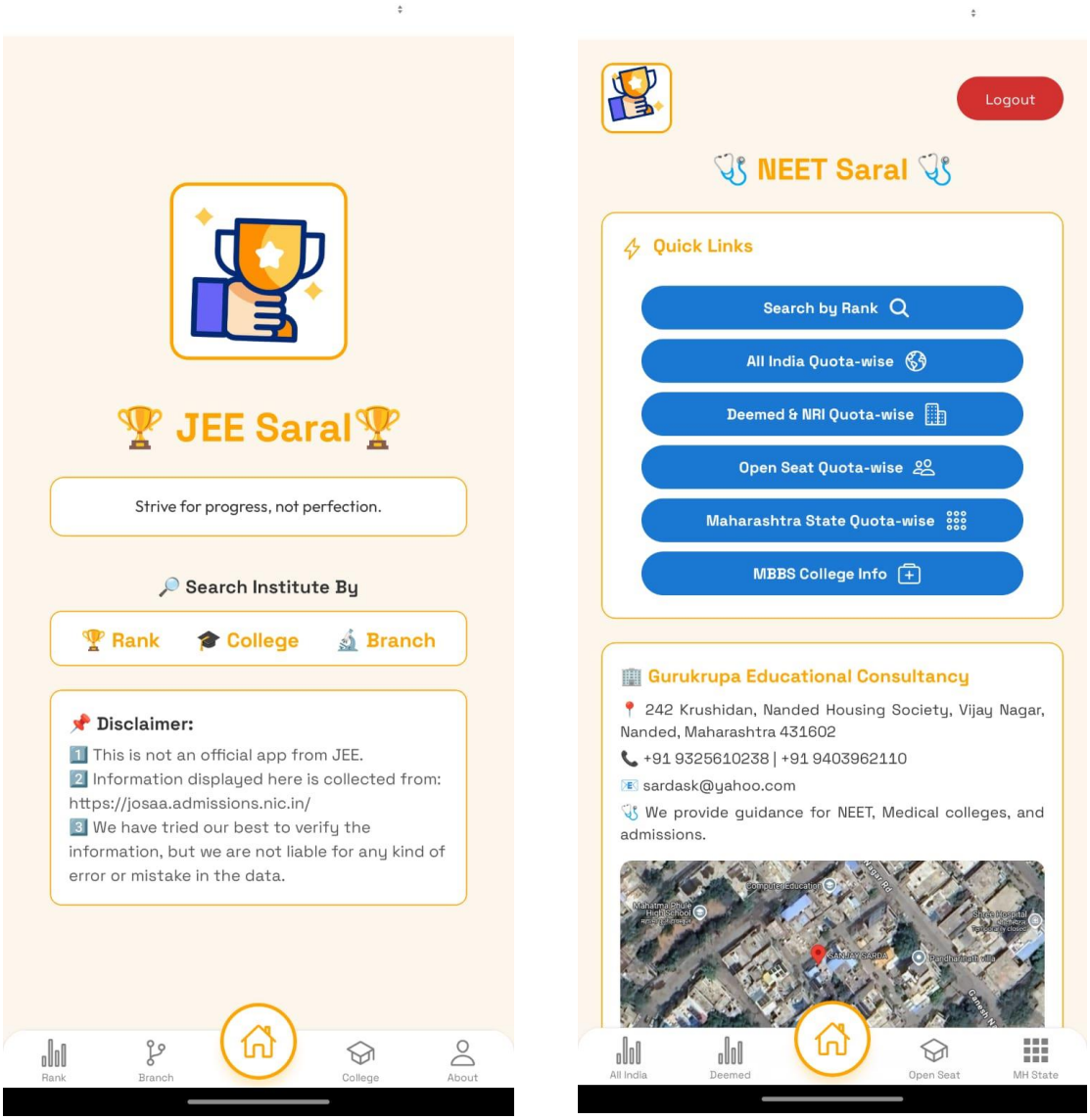


Fig 3.9: Features of JEE Saral and Neet Saral

NEET Saral Features Implemented

1. Quota-Wise Browsing

- The app included dedicated tabs for major NEET quotas:
    - All India (AIQ)

    - Maharashtra State Quota
    - Deemed/NRI Quota
    - Open Seat Quota
- Each tab dynamically fetched data using filter.php with parameters like quota, category, and rank.

2. Detailed College Info

- Clicking on a college entry showed closing rank, seat matrix, and quota-category-specific info.
- Layout adapted dynamically to include program type (MBBS, BDS), location, and cutoff ranks.

3. Filtering by Rank, Category, and State

- A form-based interface allowed users to enter their rank and select their reservation category (e.g., OBC, SC, EWS).
- Results were refined based on these filters to help students make informed choices.

4. Static Informational Pages

- Legal disclaimers, contact info, and an About section were added to clarify data sources and consultancy credibility.

5. Login/Register with Persistence

- Secure login and token storage using AsyncStorage for future sessions.
- Handled auto-redirects and logout functionality using layout guards in Expo Router.

Chapter 4

TESTING AND DEPLOYMENT

4.1 Testing Methodologies

Testing was a crucial part of the development process for both JEE Saral and NEET Saral applications. Since these apps were targeted at real users and published on the Play Store, rigorous testing ensured functionality, reliability, and a smooth user experience. A mix of manual testing, unit testing (where applicable), and integration testing was adopted during the development cycle.

1. Manual Testing

Manual testing was the primary method used throughout the internship, especially in the absence of a formal QA team. This involved:

- Testing each screen for layout consistency across multiple Android devices and resolutions.
- Validating form inputs, dropdown selections, and button interactions.
- Navigating across screens to test route transitions, deep links, and back stack behavior.
- Verifying data flow by entering various combinations of ranks, quotas, and categories.

Common bugs identified manually:

- Filters not resetting between screen transitions.
- Improper error messages on failed API calls.
- Splash screen hanging on slower devices.

2. Unit Testing

Although the internship timeline was limited, basic unit testing was conducted for reusable functions and utility logic. Examples included:

- Validating the formatting of rank input fields.
- Checking null/empty conditions before sending API requests.

- Testing utility functions in the utils/ folder (e.g., string sanitization, rank range validators).

Unit tests were written using:

- **Jest** (available via jest-expo) – for testing isolated JavaScript functions.
- **Manual mocks** – to simulate API responses without hitting the backend.

3. Integration Testing

Integration testing was performed manually to ensure different modules worked together cohesively. Examples include:

- Verifying login + token storage + protected route access.
- Testing API call + response parsing + frontend rendering in a complete flow.
- Navigating from home dashboard → quota tab → college list → detail view and checking data continuity.

Each major use case was tested from input to result, simulating a real user journey from onboarding to college selection.

Test Devices Used

- Android Smartphones (real devices): Samsung Galaxy A16, Moto G40 Fusion
- Emulators via Android Studio and Expo Go
- Web version (Expo Web) tested in Chrome and Firefox

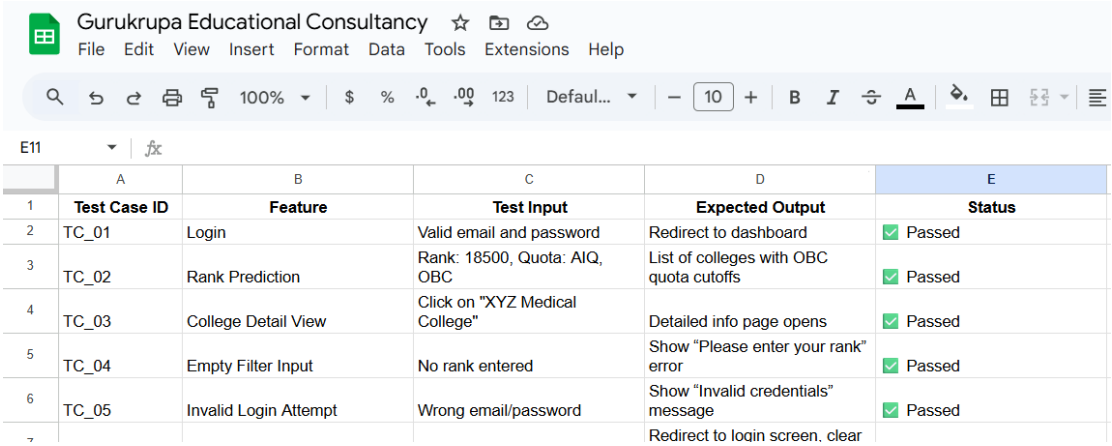
Testing ensured that the app was robust enough for real-world deployment, even in varying network and device conditions.

4.2 Test Cases and Bug Fixes

After the core features of both applications were implemented, structured testing was carried out using manually designed test cases. These test cases helped identify functional flaws, UI glitches, and data-related inconsistencies. Each module was verified using multiple input scenarios, ensuring the apps worked reliably under real-world usage conditions.

Test Case Design

Test cases were manually documented using spreadsheets, with the following standard format:



	A	B	C	D	E
1	**Test Case ID**	**Feature**	**Test Input**	**Expected Output**	**Status**
2	TC_01	Login	Valid email and password	Redirect to dashboard	✅ Passed
3	TC_02	Rank Prediction	Rank: 18500, Quota: AIQ, OBC	List of colleges with OBC quota cutoffs	✅ Passed
4	TC_03	College Detail View	Click on "XYZ Medical College"	Detailed info page opens	✅ Passed
5	TC_04	Empty Filter Input	No rank entered	Show "Please enter your rank" error	✅ Passed
6	TC_05	Invalid Login Attempt	Wrong email/password	Show "Invalid credentials" message	✅ Passed
				Redirect to login screen, clear	

Fig 4.1: Screenshot Of Test Case Spreadsheet

Bug Tracking Process

Identified bugs were tracked using a shared Google Sheet and resolved in weekly sprints. Common categories of bugs included:

- **Functional Bugs**: API not returning data for certain filters, login loop on session timeout.
- **UI/UX Bugs**: Overlapping dropdowns, misaligned icons, long text overflow on small screens.
- **Integration Bugs**: API failure not being handled properly on slow connections or timeout.

Example Bugs and Fixes:

- *Bug*: Filtered college list showed stale data when switching between tabs. *Fix*: Added useFocusEffect to reload data on tab switch.
- *Bug*: Splash screen not transitioning smoothly on low-end devices. *Fix*: Reduced image size and optimized font loading using useFonts() hook.
- *Bug*: Empty search returned blank screen instead of user feedback. *Fix*: Added conditional rendering with "No results found" message and retry option.

User Feedback Loop

Initial builds were shared with a small test group (students and internal team), and their feedback helped uncover real-use bugs such as:

- Delayed API responses during high traffic
- Typo in college names
- Confusion with navigation flow

These suggestions were noted and integrated into subsequent builds.

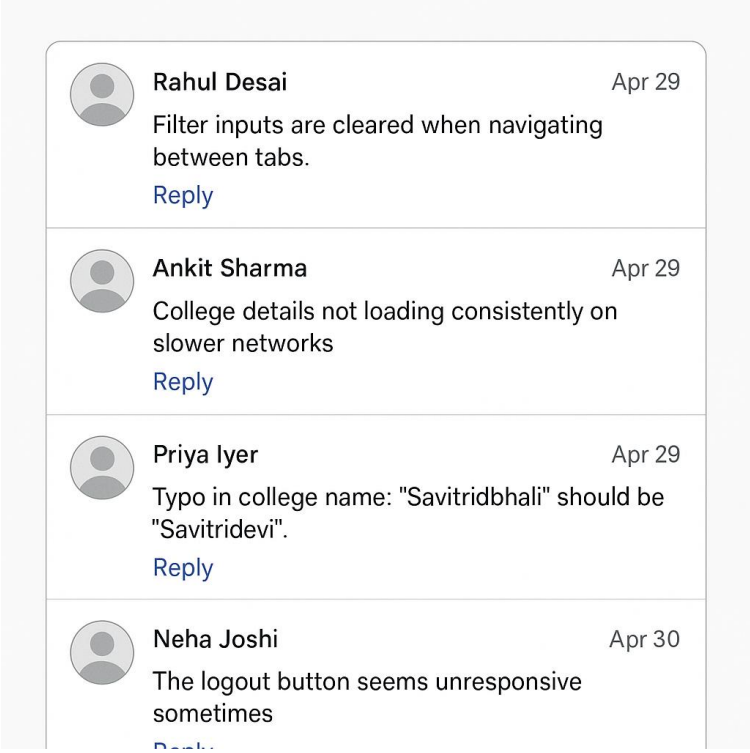


Fig 4.2 User Closed Testing Feedback

4.3 Optimization Techniques Used

Optimizing the performance of both **JEE Saral** and **NEET Saral** was a critical part of the development process, particularly because the applications were aimed at a broad user base—including students using entry-level smartphones and accessing data on slower networks. To ensure smooth functionality and a responsive user experience, several performance optimization techniques were applied at both the frontend and backend levels.

One of the first strategies implemented was **lazy loading** for screens and UI components. Using React Native's capabilities along with the Expo framework, screens

were configured to render only when navigated to, significantly improving the initial loading time of the application. Moreover, the UI was modularized into small, manageable components to reduce the rendering overhead and improve code maintainability.

From the API integration perspective, significant effort was made to minimize unnecessary network requests. Rather than making multiple calls for every user interaction, filters and user inputs were batched into a single API request using Axios. Furthermore, logic was introduced to cache previously retrieved data where appropriate, which helped reduce server load and improved performance on repeat interactions. Axios was also configured with custom timeout settings and basic retry logic to gracefully handle slow or unreliable network conditions, thereby preventing abrupt app crashes or blank screens during poor connectivity.

On the asset side, all images used in the apps were compressed and optimized for mobile using the WebP format, and large assets were scaled down to avoid memory overuse. **Fonts**, which are often overlooked, were loaded efficiently using Expo's useFonts() hook to ensure that the UI rendered properly only after font resources were ready—this helped maintain visual consistency without causing flash-of-unstyled-text (FOUT) issues.

When dealing with dynamic data lists—especially for college listings—**FlatList** was used with performance-focused props such as initialNumToRender and windowSize. This ensured that only visible items were rendered at any given time, significantly improving scroll performance. Additionally, React's memo() function was used to wrap several stateless UI components, avoiding unnecessary re-renders when props hadn't changed.

The UI was also made fully responsive using percentage-based layouts and device-specific breakpoints. Conditional styling using React Native's Platform module helped fine-tune layouts for Android, iOS (if needed in the future), and web. Special attention was given to handling empty states and edge cases—for example, if a user entered a rank that did not match any college, a meaningful "No results found" message was shown instead of leaving the screen blank.

Lastly, significant work went into minimizing the final APK size for faster download and installation via the Play Store. Unused libraries, redundant images, and unnecessary

console logs were stripped out of the production build. The splash screen was optimized to appear instantly with a lightweight image and minimal animation, reducing perceived load time.

These optimization techniques collectively resulted in apps that were fast, efficient, and user-friendly—regardless of device capability or internet speed. This not only improved usability but also played a key role in earning positive feedback from early users and test groups.

4.4 Deployment to Play Store

After completing development and internal testing of both JEE Saral and NEET Saral, the next major milestone was deploying the applications to the Google Play Store, making them publicly available to Android users. The deployment process was an important learning experience, as it involved working with production builds, app store policies, metadata preparation, and release management.

The deployment pipeline was managed using EAS (Expo Application Services), which simplified the process of building and signing APK files for production. After thoroughly testing the apps on real devices using the Expo Go app, a production build was generated using the command-line interface provided by Expo. The APK was signed automatically through EAS and downloaded for upload to the Play Store. This approach eliminated the need for complex keystore management and manual signing, streamlining the deployment process for a solo or small-team developer.

Before uploading the apps to the Play Store, certain prerequisites had to be fulfilled. A Google Play Console developer account was created, and separate app listings were made for JEE Saral and NEET Saral. Each listing required a package name, app icon, and various assets such as a high-resolution feature graphic, screenshots from the app, and a brief promotional video (optional). Careful attention was paid to writing clear and concise descriptions for each app, highlighting their purpose, features, and target audience. Keywords were also optimized to help improve discoverability on the Play Store.

In addition to the creative assets, Google mandates several policy requirements for apps. This included filling out the Data Safety form, which required declaring the types of data collected and stored by the app—such as name, email address, and rank-related

inputs. A Privacy Policy was drafted and hosted externally, then linked in the app listing. The apps were also reviewed for compliance with Google's content rating system, which rated both apps as suitable for all age groups since they did not contain sensitive or offensive content.

Once all required fields were completed and the signed APK uploaded, the apps were submitted for Play Store review. The review process typically takes between 24–48 hours. Fortunately, both apps were approved without issues and were published successfully on their first attempt. After publication, internal testing tracks were set up to allow further post-launch testing with trusted users before pushing future updates to the production track.

Maintaining the app post-deployment also involved using the Play Console to monitor user reviews, crash analytics, and update rollouts. Feedback from users was closely monitored, and any bugs or feature requests were noted for implementation in future updates.

Publishing these apps was a key highlight of the internship. It provided end-to-end exposure to not just development but also release management and public distribution—an essential skill for real-world app developers.

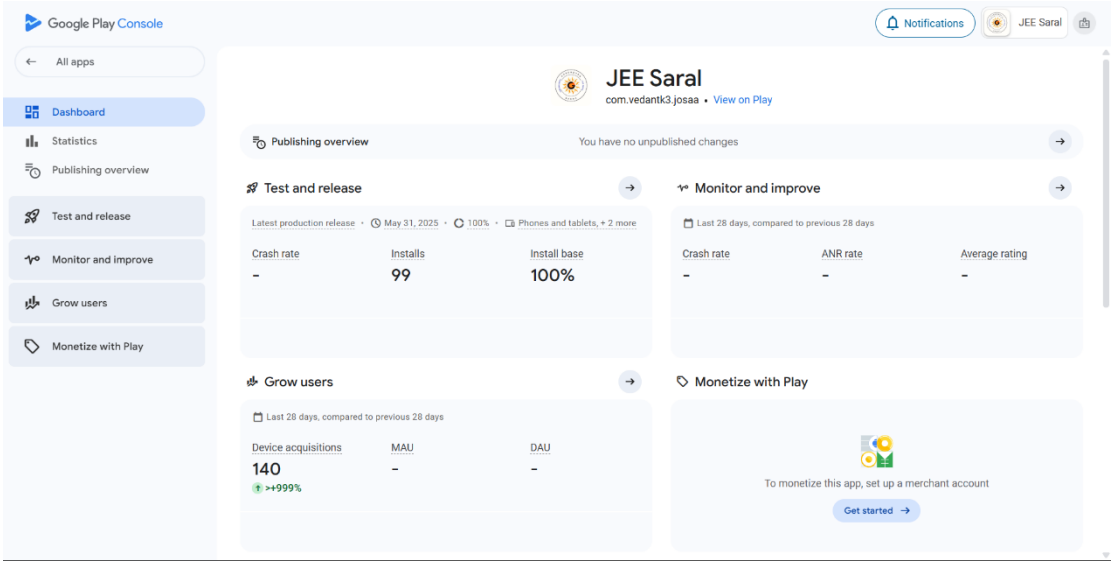


Fig 4.3: JEE Saral Google Play Console Page

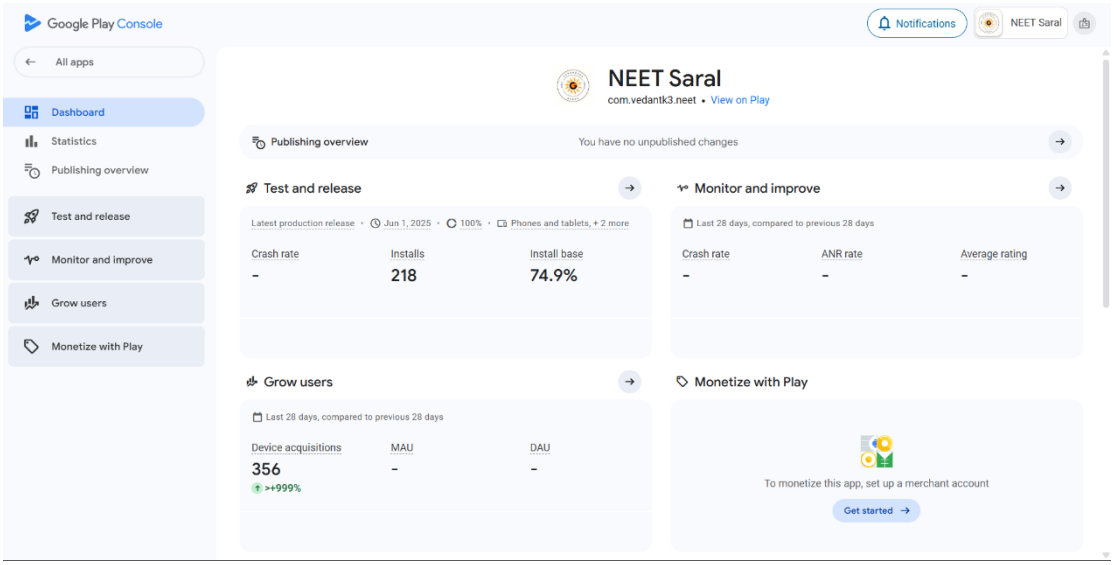


Fig 4.3: NEET Saral Google Play Console Page

4.5 Hosting and Backend Configuration

The backend of both JEE Saral and NEET Saral was hosted using Hostinger, a popular shared hosting service that offers PHP and MySQL support out-of-the-box. Hostinger provided an accessible and cost-effective solution for deploying lightweight PHP-based APIs, which were central to the functionality of both applications.

The deployment of backend services began with setting up the file structure on the hosting environment. Using Hostinger's built-in File Manager and FTP access, the complete PHP backend—including files such as login.php, register.php, filter.php, getusers.php, and smtp_mailer.php—was uploaded to the public directory. The folder structure was carefully organized to reflect separation between API scripts, static resources (like college logos), and configuration files.

Database configuration was another crucial aspect of the setup. A MySQL database was created via Hostinger's control panel. Tables were designed to store user information, rank details, cutoff records, and college metadata. SQL import tools were used to populate initial datasets that the apps would rely on during filtering operations. The db.php file was configured to securely connect to the remote database using the server's credentials, and was designed to be reusable across all API endpoints.

To ensure the backend was accessible to the mobile applications, public-facing URLs were generated for each API endpoint (e.g., https://gurukrupaedu.com/api/xxxxx.php). These URLs were then integrated directly into the frontend via Axios calls. During

testing, CORS and hosting-specific permission issues were handled through Hostinger's PHP configuration settings.

One of the key integrations was email functionality, handled by PHPMailer, a popular mailing library in PHP. SMTP configuration for Hostinger's mail server was set up with authentication credentials to allow outgoing emails for tasks like registration confirmations or user queries. The script smtp_mailer.php was tested using a sandbox file (test_smtp.php) before integration with the frontend app.

Security practices were followed wherever possible within the shared hosting constraints. API input validation was implemented to prevent SQL injection, and sensitive connection details were placed in separate files to avoid direct exposure. Folder permissions were configured to prevent unauthorized access to key directories or scripts.

Regular backups were maintained using Hostinger's scheduled backup tools, ensuring that the database and PHP scripts could be restored if needed. Additionally, the Hostinger dashboard allowed real-time monitoring of server load and performance, which was useful during the app's initial launch phase.

By the end of deployment, the backend was stable, secure, and fully integrated with the mobile applications. Hostinger's reliable uptime ensured uninterrupted service for end-users, while the PHP-MySQL combination provided fast and efficient processing of user requests.

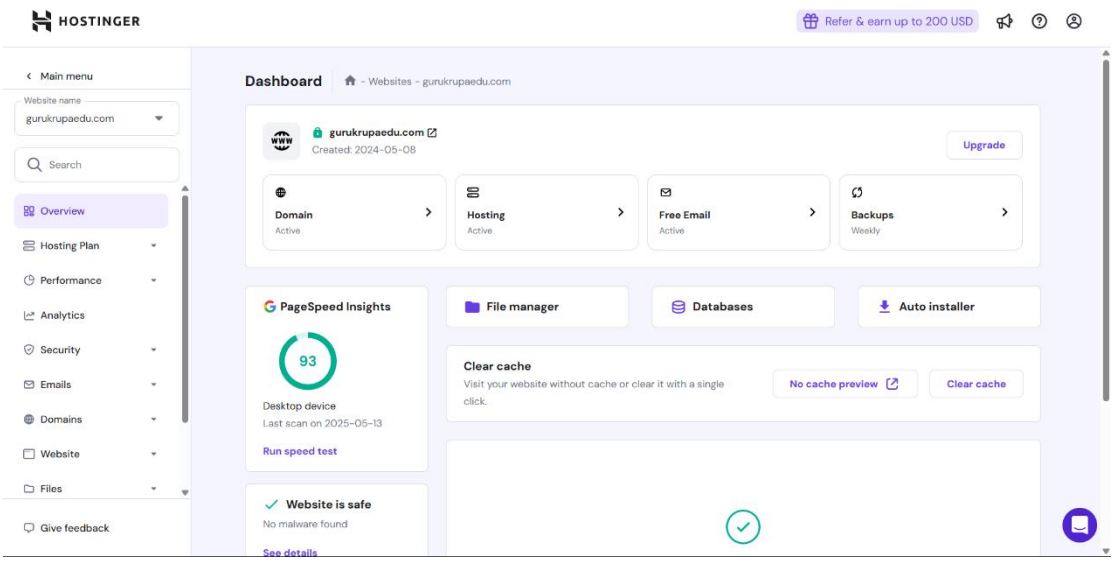


Fig 4.4: Hostinger Dashboard

Chapter 5

INTERNSHIP OUTCOMES AND LEARNINGS

5.1 Deliverables and Deployment Outcomes

Over the course of the internship, the primary objective was to deliver two functional, user-friendly, and fully deployed mobile applications—JEE Saral and NEET Saral—alongside a supporting backend infrastructure. Both apps were built from the ground up using React Native with Expo and PHP-based backend APIs hosted on Hostinger. The applications were developed, tested, and successfully published on the Google Play Store during the internship timeline.

The first major deliverable was the JEE Saral app, designed to assist engineering aspirants in exploring eligible colleges based on their Joint Entrance Examination (JEE) ranks. This app included modules such as rank-based college prediction, branch search, detailed college views, and integration with consultancy services. Similarly, the NEET Saral app was developed for medical aspirants, featuring advanced filters for quotas (All India, Deemed, State, Open), category-based college browsing, and rank-based search functionality.

In addition to the mobile applications, a complete backend API system was developed using PHP and MySQL. This system handled user registration, login authentication, data filtering, and college information retrieval. Secure email communication was enabled through PHPMailer using SMTP configuration, allowing the system to send verification emails to newly registered users.

A web-based Admin Dashboard was also delivered for both JEE Saral and NEET Saral. These dashboards displayed user statistics in real-time, such as total registered users, approval status, and pending actions. This provided the internal team at Gurukrupa Educational Consultancy with tools to manage user data and monitor app adoption without requiring technical intervention.

The deployment outcomes were also a significant success. Using Expo's EAS build service, both apps were built, signed, and uploaded to the Google Play Store, making them publicly available to Android users across India. The apps met all Play Store requirements, including data safety disclosures, privacy policy links, and high-

resolution graphic assets. Importantly, both apps were approved on the first submission, indicating technical readiness and policy compliance.

These deliverables collectively reflect a full-stack contribution—from interface design to API development to Play Store publishing—representing the complete software development lifecycle. Each component of the system was thoroughly tested, documented, and aligned with the goals of Gurukrupa Educational Consultancy to make student guidance more accessible and efficient.

5.2 Performance Metrics and User Impact

Measuring the real-world usage and growth of the applications was an essential step in evaluating the success of this internship project. By the end of the internship period, both JEE Saral and NEET Saral had not only been successfully published on the Google Play Store but had also shown promising adoption and engagement from users. Metrics were tracked via the Google Play Console and internal Admin Dashboards built for user management.

According to Play Store analytics, JEE Saral achieved an installed audience of 98 users, with a user acquisition count of 123, reflecting a 9,700% increase over the previous 30 days. NEET Saral saw even greater traction, reaching an installed audience of 280 users, with 346 total user acquisitions, representing a 9,233.3% growth. This dramatic increase in user acquisition shortly after deployment validated the relevance and usefulness of both applications in their respective domains.

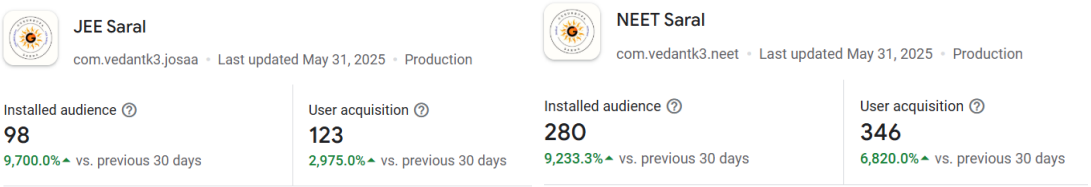


Fig 5.1: Google Play Console metrics for JEE Saral and NEET Saral

These insights not only confirmed the market demand but also indicated successful user onboarding and app discoverability on the Play Store. Although formal user ratings and revenue data were not yet available at the time of writing, the download numbers alone suggested a healthy initial user base with room for future growth and engagement.

Internally, the **Admin Dashboards** reflected a near-perfect onboarding and approval rate. The **JEE Saral dashboard** showed **124 total users**, all of whom were marked as approved with **0 pending requests**. Similarly, **NEET Saral** registered **170 total users**, again with all users marked as approved and no pending actions. This indicated that the registration system, user verification process, and overall backend integration were functioning correctly and reliably.

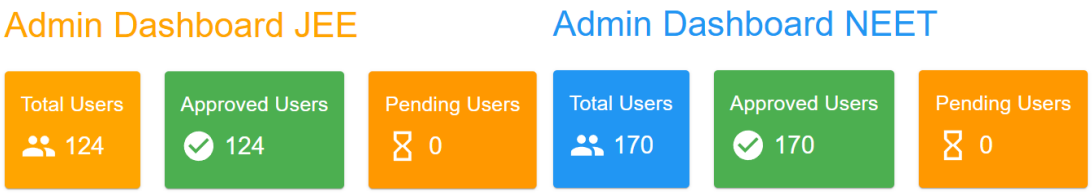


Fig 5.2 Total Users till date

These performance metrics confirmed not just that the apps were technically sound, but also that they were being actively used by real students in the target demographic. The ability to track user growth both externally (via Play Console) and internally (via the dashboards) provided valuable feedback and a foundation for future development—such as adding analytics, improving personalization, and eventually collecting app ratings.

5.3 Technical Learnings and Personal Growth

This internship offered a well-rounded and immersive learning experience, allowing me to grow significantly as both a developer and a professional. Through the complete development lifecycle of two real-world applications, I was able to bridge the gap between theoretical concepts learned in academics and the practical demands of building, testing, and deploying production-grade software.

Technical Learnings

One of the most valuable technical skills I developed was full-stack mobile app development using React Native with Expo. I learned how to build structured, modular apps with intuitive navigation, interactive forms, dynamic list rendering, and platform-aware design using React Native components. I also gained hands-on experience with integrating third-party libraries like react-native-paper, element-dropdown, and @expo/vector-icons to streamline UI creation and improve user experience.

On the backend side, I honed my skills in PHP development by creating RESTful APIs for user management, college filtering, and data handling. Working with MySQL

improved my understanding of relational database design, query optimization, and schema structuring for real-time filtering of large datasets. Additionally, integrating PHPMailer to implement SMTP-based email verification helped me understand how email protocols work and how to secure communication features in a live environment.

One of the highlights was deploying apps on the Google Play Store using Expo's EAS Build system. I gained confidence in signing APKs, setting up Play Store listings, and ensuring compliance with data privacy and app content guidelines. I also became proficient with tools like Axios for making API calls, AsyncStorage for session persistence, and Postman for backend testing.

This internship helped solidify my grasp on end-to-end architecture—covering everything from frontend UI logic to backend processing, data flow, error handling, deployment, and real-time analytics integration.

Personal and Professional Growth

Beyond the technical aspect, I developed a strong set of soft skills that are essential in a professional setting. Working independently on multiple modules while staying aligned with the consultancy's goals improved my time management, task prioritization, and problem-solving under deadlines. Communicating with mentors and presenting weekly updates helped sharpen my professional communication and documentation abilities.

Collaborating on the needs of real users—students and parents—gave me a deeper appreciation for user-centric design and the importance of clarity, simplicity, and accessibility in educational tech tools.

Additionally, resolving bugs and incorporating user feedback helped me build a mindset of iterative improvement—a crucial attribute for any software developer working in production environments.

CONCULSION

The three-month internship at Gurukrupa Educational Consultancy has been an incredibly enriching journey, blending technical growth with meaningful real-world application. Tasked with developing and deploying two full-scale mobile applications—JEE Saral and NEET Saral—I had the opportunity to experience the entire software development lifecycle, from conceptualization and design to backend API creation and deployment on the Google Play Store.

This internship not only allowed me to sharpen my skills in React Native, PHP, MySQL, and Expo, but also introduced me to crucial tools and practices such as RESTful API development, user authentication systems, real-time data filtering, and integration of email services using PHPMailer. I also gained hands-on experience in app optimization, session management, Play Store compliance, and performance tuning—skills that are highly valuable in a production environment.

Beyond the technical aspects, the internship also facilitated my growth in time management, problem-solving, client-focused development, and iterative improvement. Regular collaboration with mentors and feedback loops helped me understand the importance of clarity in communication, usability in design, and accountability in delivery. The creation of admin dashboards for both apps also reinforced the need for operational tools that support post-launch maintenance and user engagement.

The impact of the work is evident through the strong user acquisition metrics, steady growth in active installations, and a fully functional backend supporting hundreds of users in each application. These results serve as both validation of the technical implementation and evidence of the utility these apps offer to students navigating the complexities of college admissions.

REFERENCES

[1] J. Samama, *Security Technologies for the World Wide Web*. Boston, MA: Artech House, 2008.

[2] R. W. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 8th ed. New York: McGraw-Hill Education, 2015.

[3] M. Fowler, *Patterns of Enterprise Application Architecture*. Boston: Addison-Wesley, 2002.

[4] D. Crockford, *JavaScript: The Good Parts*. Sebastopol, CA: O'Reilly Media, 2008.

[5] T. Yamano, "An Empirical Study on React Native Application Performance," in *Proceedings of the 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 1–10, 2021.

[6] K. A. Hafeez, M. Khan, and A. Anjum, "Comparative analysis of PHP frameworks for scalable web applications," *International Journal of Computer Applications*, vol. 142, no. 5, pp. 10–16, 2016.

[7] M. L. Murphy, "Using MySQL for scalable backend architectures," *IEEE Software*, vol. 27, no. 6, pp. 84–89, Nov.-Dec. 2010.

[8] G. T. Heineman and W. T. Councill, *Component-Based Software Engineering: Putting the Pieces Together*. Boston, MA: Addison-Wesley, 2001.

[9] Expo.dev, "Building with Expo and React Native," Expo Documentation, 2024. [Online].

[10] PHPMailer, "PHPMailer Library for PHP Email," GitHub Repository, 2024. [Online].