Learning Platform Project

This report outlines the proposed development of a comprehensive learning platform catering to students, teachers, and administrators. The platform will facilitate user management, course registration, test participation, progress tracking, communication, and secure access control.

Problem Statement

The client requires a robust learning platform with dedicated apps for students, teachers, and an admin dashboard. This platform needs to address user roles, course management, testing, progress monitoring, communication, and secure access control mechanisms.

Proposed Solution

We propose a scalable and secure learning platform built with the following technologies:

- **Frontend:** React/React-Native for mobile apps, Next.js for the admin dashboard, and TypeScript for enhanced code maintainability.
- **Backend:** NestJS for server-side rendering and API development.
- **Database:** SQL database (e.g., MySQL or PostgreSQL) for data storage and management.

Product Requirements

- **Authentication/Authorization:** Secure login with distinct user roles (student, teacher, admin) and access control mechanisms (JWT tokens, RBAC).
- **Course Management:** Students can register for up to 4 courses from a predefined list.
- **Testing:** Students can participate in tests created by teachers.
- **Progress Tracking:** Personalised dashboards for students to visualise their progress in each course.
- **Communication:** Facilitate messaging between students and teachers within the platform.
- **Teacher Access Control:** Teachers can only access courses they are assigned to teach.
- **Admin Privileges:** Admins can manage teachers (including adding new ones) and have full platform access.

Architecture

- **Microservices Architecture:** Independent, loosely coupled services (e.g., User Service, Course Service, Test Service) for improved scalability and maintainability.
- **API Gateway:** Centralised API gateway to handle all communication between frontend apps and backend services.

Tech Stack

- **Frontend:** React/React-Native, Next.js, TypeScript
- **Backend:** NestJS
- **Database:** SQL (PostgreSQL), NoSQL (MongoDB)
- **Cloud Hosting:** Cloud providers like AWS, GCP, or Azure
- **Domain Provider:** Namecheap or Hostinger
- **Analytics:** Google Analytics and Posthog

Database Schemas and Models

Database tables will be designed for users, courses, tests, results, messages, and roles with appropriate relationships and data types. Corresponding models will be implemented in NestJS entities.

Roadmap Timeline

1. **User Stories:** Define detailed user stories for each functionality, prioritising features based on client needs.
2. **Design:** Create user interface mockups and prototypes for each app and the admin dashboard.
3. **Development:** Implement the frontend and backend services based on the chosen tech stack and architecture.
4. **Testing:** Conduct thorough unit, integration, and user acceptance testing to ensure platform functionality and stability.

Hosting and Deployment

- We will utilise cloud platforms like Google Play Store for the apps and AWS for hosting the admin and Render or Railway for the backend services, as well as the database.
- Choose a domain name provider like Namecheap or Hostinger and configure DNS settings.

Analytics

- Integrate Google Analytics and Posthog for user behaviour tracking, performance monitoring, and platform optimization.
- Get data on DAU, WAU and MAU.

Limitations

- Platform scalability and performance depend on the chosen hosting infrastructure and database size.
- Robust security measures are essential to protect user data and prevent unauthorised access.

Storage

- Storage requirements will depend on the number of users, courses, and content uploaded to the platform. Cloud providers offer scalable storage solutions.

Cost of Implementation

- Development costs will depend on the project complexity, team size, and development duration. The proposed team size is 1 designer, 1 backend developer and 2 frontend developers. The designer would be designing screens for 3 different apps and the cost of that will be five hundred thousand naira (NGN 500,000), the backend developer's fee in set at eight hundred thousand naira (NGN 800,000), and the frontend developers' services will incur seven hundred thousand naira (NGN 700,000) each, totaling at two million, seven hundred thousand naira (NGN 2,7000,000).
- Hosting costs will vary based on the chosen cloud platform, resource utilisation, and storage needs. Typically, AWS would be ideal for the frontend as they have a one-year free offer for Amplify and subsequent monthly charges may fall short of five dollars (US$5). A Google Developer's Account is required to host an application on the Play Store, same for the Apple Store. Google requires a one-time non-refundable fee of twenty-five dollars (US$25), while the Apple Store requires an annual fee of ninety-nine dollars (US$99). Apple Store also requires an extensive verification and testing process before the app can be released to the store. Hosting the backend requires a storage bucket and a database, the former typically being the more expensive. A serverless option would be ideal to mitigate against paying for a continuous running service. An estimated value can only be provided based on initial usage. Using an SQL database will increase efficiency but comes at a higher cost, which a NoSQL database may have limitations but pays off in terms of expenditure management.

Additional Notes

- We will consider implementing additional features like gamification, downloadable resources, and forum discussions for a more engaging learning experience.
- Ensure compliance with relevant data privacy regulations.