Project Synopsis

On

ED-TECH PLATFORM

In the partial fulfillment for the award of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

<u>Submitted by</u>

Suhani	Regd. No- 2130113
Mayank Mishra	Regd. No- 2130101
Shubham Jha	Regd. No- 2130109

Under the Guidance of

Mr. Sukhpreet Singh

Assistant Professor

(CSE Department)

Sant Longowal Institute of Engineering & Technology, Longowal - 148106

District - Sangrur, Punjab

February, 2024

Index

S.No.	Topics	Page No.
1.	Introduction	3
2.	Proposed Work and Methodology	4
3.	Objectives	5
4.	Technology Used	6
5.	Software Requirements	7
6.	References	8

Introduction

StudyNotion is a comprehensive ed-tech platform, meticulously crafted using the MERN stack, comprising ReactJS, NodeJS, MongoDB, and ExpressJS. With a mission to revolutionize education, StudyNotion is committed to delivering a seamless and interactive learning experience, fostering accessibility and engagement for students worldwide. The platform serves as a dynamic space for instructors to exhibit their expertise and connect with learners globally.

Our exploration of the technical aspects will cover the entire spectrum of StudyNotion, starting with the System Architecture – providing a high-level overview and architectural diagrams. Moving on to the Front-end, we'll delve into the intricacies of its architecture, user interface design, and the array of features and functionalities it offers, supported by an array of frameworks, libraries, and tools.

The Back-end is a crucial element, and we'll dissect its architecture, detailing features, functionalities, and the stack of frameworks, libraries, and tools employed.

The intricacies of data models and the database schema will also be explored. API Design will be scrutinized, elucidating the list of endpoints, their functionalities, and presenting sample API requests and responses.

The Deployment process is a critical part of the platform, and we'll unravel the hosting environment, infrastructure, and the deployment scripts and configuration that keep StudyNotion running seamlessly. Testing, an indispensable phase, will be thoroughly explained, covering different testing types and the frameworks and tools employed to ensure the platform's robustness.

Lastly, the roadmap for the future is illuminated in the Future Enhancements section. This segment provides insight into potential improvements, explaining how these enhancements would elevate the platform, along with estimated timelines and priorities for implementation. In summation, StudyNotion is not just an ed-tech platform but a versatile and intuitive ecosystem geared towards an immersive learning experience and a global stage for instructors to showcase their expertise. The subsequent sections will unveil the technical intricacies, giving a comprehensive understanding of the platform's capabilities and features.

Proposed Work and Methodology

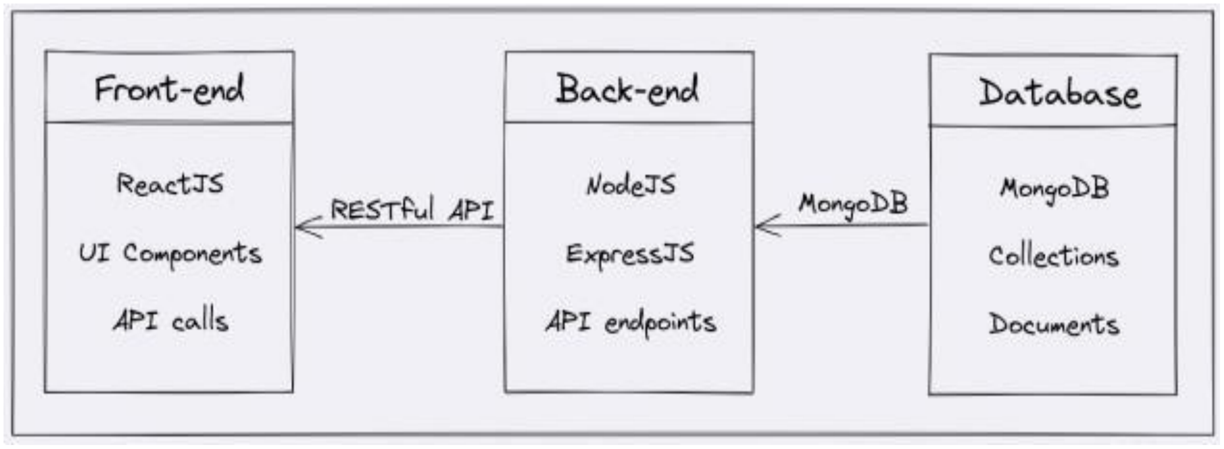
Study Notion, an ed-tech platform built on the MERN stack, is poised to redefine the educational landscape by providing a multifaceted solution for both students and instructors. The proposed work centers around the meticulous development and enhancement of the platform's capabilities to deliver a cutting-edge educational experience.

At the core of the proposed work is the commitment to crafting a seamless and interactive learning experience for students. The choice of ReactJS for the front end ensures that the platform's user interfaces are not only dynamic but also responsive, catering to the diverse needs and preferences of students. The front-end architecture, adept at RESTful API communication with the back end, forms the foundation for Study Notion's goal of making education more accessible and engaging.

Simultaneously, StudyNotion envisions creating a global platform for instructors, transcending geographical boundaries. Instructors gain access to a dashboard that provides an overview of their courses, ratings, and valuable insights into the effectiveness of their teaching methods. The back end, powered by NodeJS and ExpressJS, serves as the backbone for user authentication, course management, and the secure storage of course content, ensuring that instructors can focus on delivering quality education without compromising on security or functionality.

As the platform evolves, the proposed work extends beyond the initial development phase to include ongoing enhancements. StudyNotion recognizes the importance of adapting to the evolving educational landscape and is committed to incorporating future enhancements that further elevate the platform's capabilities.

The overarching goals are underpinned by StudyNotion's commitment to a client-server architecture, with ReactJS as the client and NodeJS, ExpressJS, and MongoDB forming the robust server infrastructure. The database, built on MongoDB, provides a flexible and scalable solution for storing unstructured and semi-structured data, accommodating diverse content types such as videos, images, and PDFs.



Objective

1. **User Engagement:**

   Introduce interactive features like quizzes and discussions.

   Enhance user satisfaction through feedback and ratings.

   Implement personalized learning paths based on user behavior.

2. **InstructoConnectivity:**

   Increase the number of instructors on the platform.

   Develop instructor profiles highlighting expertise.

   Simplify content creation, curation, and sharing tools for instructors

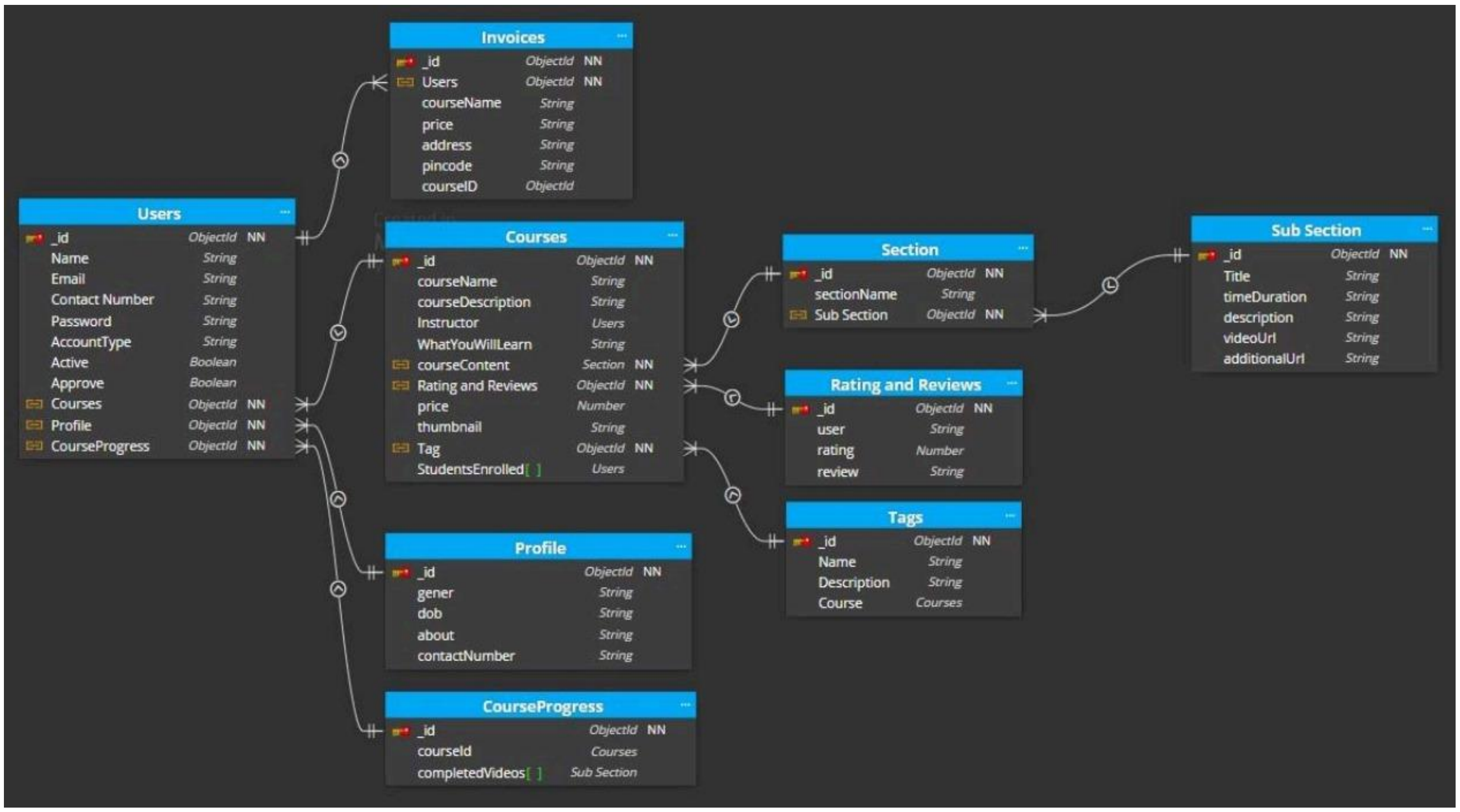
3. **Platform Reliability:**

   Implement robust error handling and monitoring.

   Conduct performance testing for scalability.

   Explore cloud-based solutions for efficient scaling.

Project Schema



Technology used

1. **Front-end Technology:**
    - ReactJS: A popular JavaScript library for building dynamic and responsive user interfaces.
    - CSS and Tailwind: Styling frameworks to enhance the look and responsiveness of the user interface.
    - Redux: A state management library for React, used to manage the application's state efficiently.
    - VSCode: A popular code editor used as the development environment for the front end.
2. **Back-end Technology:**
    - NodeJS: A JavaScript runtime used as the primary framework for building scalable and robust server-side applications.
    - ExpressJS: A web application framework for Node.js, simplifying the process of building web applications.
    - MongoDB: A NoSQL database providing flexible and scalable data storage for unstructured and semi-structured data.
    - JWT (JSON Web Tokens): Used for secure authentication and authorization.
    - Bcrypt: Employed for password hashing, adding an extra layer of security to user data.
    - Mongoose: An Object Data Modeling (ODM) library used for interacting with MongoDB using JavaScript.
3. **Other Tools and Services:**
    - Cloudinary: A cloud-based media management service used for storing and managing media content, including images, videos, and documents.
    - Razorpay: Integrated for seamless payment handling within the platform.
    - Vercel: Used for front-end hosting, providing a fast and scalable environment.
    - Render or Railway: Cloud-based hosting services for back-end applications built with Node.js and MongoDB.
    - MongoDB Atlas: A fully managed cloud database service used for hosting the MongoDB database.

Software Requirement

1. **Development Tools:**
    - Utilize Node.js and npm for server-side development, ensuring that the chosen versions are stable and well-supported.
    - Leverage Express.js for building a scalable and efficient back-end web application.
    - Employ ReactJS for front-end development, emphasizing modular and reusable component structures.
2. **API Design and Testing:**
    - Use dedicated tools like Postman for designing and testing RESTful APIs, ensuring the reliability of data exchange between front-end and back-end.
    - Implement robust frameworks like Jest for JavaScript to conduct unit testing, ensuring the stability and functionality of the application.
    - Prioritize thorough documentation of API endpoints, request-response formats, and error handling procedures for both development and maintenance purposes.
3. **Deployment and Hosting:**
    - Opt for hosting services like Vercel, Render, or Railway for the front end and MongoDB Atlas or a dedicated server for the database.
    - Implement proper configuration for deployment scripts and environment variables to streamline the deployment process.
    - Conduct thorough testing of the deployed application to identify and address any performance or compatibility issues.
4. **Security Measures:**
    - Integrate SSL certificates for secure data transmission, enhancing the overall security of the platform.
    - Implement JWT for secure authentication, ensuring that user credentials are handled safely.
5. **Design and Development Tools:**
    - Leverage a version control system like Git for effective collaboration and code management.
    - Utilize Visual Studio Code or a similar code editor for efficient and streamlined development.
6. **Documentation:**
    - Develop comprehensive documentation for developers and system administrators, providing clear guidelines for setup, maintenance, and troubleshooting.
    - Establish a user-friendly wiki or knowledge base for end-users, explaining the platform's features and providing guidance on usage.
    - Prioritize continuous updates to the documentation to reflect any changes or enhancements made to the platform.

REFRENCES

- **ReactJS:** https://reactjs.org/
- **NodeJS:** https://nodejs.org/en/
- **ExpressJS:** https://expressjs.com/
- **MongoDB:** https://docs.mongodb.com/
- **Railway:** https://railway.app/docs
- **MongoDB Atlas:** https://docs.atlas.mongodb.com/
- **Git:** https://git-scm.com/doc
- **Postman:** https://learning.postman.com/