**EXPENSE GQL**

**BACHELOR OF TECHNOLOGY IN**

**COMPUTER SCIENCE AND ENGINEERING**

By

Vimal Kumar Mishra

12111630

**Subject**

# INT222 – ADVANCE WEB DEVELOPMENT



Lovely Professional University

Phagwara, Punjab (India)

April 2024

# TABLE OF CONTENTS

**1. INTRODUCTION…………………………………………………………………………….. 2. METHODOLOGY…………………………………………………………………………...**

**2.1. Requirement Analysis …………………………………………………………………...**

**2.2. Technology Selection ……………………………………………………………………**

**2.3. Architecture Design……………………………………………………………………..**

1. **SYSTEM ARCHITECTURE……………………………………………………………….** 
   1. **Frontend Architecture…………………………………………………………………. 3.2. Backend Architecture…………………………………………………………………..**

**3.3. Data Flow………………………………………………………………………………..**

1. **TECHNOLOGY USED……………………………………………………………………..** 
   1. **MongoDB……………………………………………………………………………….. 4.2. Express JS………………………………………………………………………………. 4.3. React……………………………………………………………………………………..**
   2. **Node JS………………………………………………………………………………….**
2. **KEY FEATURE AND FUNCTIONALITIES…………………………………………….**
3. **MODULES………………………………………………………………………………….**
4. **WEB SNAPSHOTS…………………………………………………………………………**
5. **GITHUB LINK……………………………………………………………………………..**
6. **REFERENCES………………………………………………………………………………**

# 1. INTRODUCTION

In the digital age, the need for efficient and versatile tools to organize and manage information has become increasingly apparent. The edUnity project emerges as a response to this demand, aiming to provide users with a comprehensive digital educational application that facilitates the creation, organization, and retrieval of notes in a seamless manner.

The Expense GQL project represents an endeavor to harness the power of modern web technologies to offer users a robust and user-friendly solution for their note-taking needs. Built upon the MERN (MongoDB, Express.js, React, and Node.js) stack, iNotebook embodies the synergy of cutting-edge frontend and backend technologies to deliver a cohesive and responsive user experience.

At its core, the iNotebook project seeks to address several key objectives:

1. Efficiency: Streamlining the process of note-taking and organization to enhance productivity.
2. Accessibility: Providing users with anytime, anywhere access to their notes through a web-based platform.
3. Customization: Offering a range of features and functionalities to cater to diverse user preferences and requirements.

Through the implementation of CRUD (Create, Read, Update, Delete) operations, iNotebook empowers users to create, edit, and manage their notes effortlessly. Additionally, the incorporation of Bootstrap for frontend development ensures a responsive and visually appealing interface that adapts seamlessly to various devices and screen sizes.

This report serves as a comprehensive exploration of the iNotebook project, delving into its methodology, architecture, implementation, and future prospects. By dissecting the various facets of the project, we aim to provide insights into the rationale behind its development, the challenges encountered, and the solutions devised to create a robust and feature-rich digital notebook application.

# 2.METHODOLOGY

The development of the Expense GQL project followed a systematic methodology aimed at ensuring efficiency, transparency, and alignment with project objectives. The methodology encompassed several key phases, each of which played a crucial role in shaping the trajectory of the project. Below is an overview of the methodology adopted: **1- Requirement Analysis:** The project commenced with a thorough analysis of user requirements and expectations. This phase involved gathering insights into the needs and pain points of potential users regarding note-taking and organization.

1. **Technology Selection:** Once the requirements were clearly defined, the next step involved selecting the appropriate technologies for the project. The MERN stack (MongoDB, Express.js, React, and Node.js) was chosen for its versatility, scalability, and compatibility with modern web development practices.Consideration was given to factors such as performance, ease of development, community support, and alignment with project goals.
2. **Architecture Design:** With the technology stack finalized, attention turned to designing the system architecture. This phase entailed mapping out the frontend and backend components, defining their interactions, and establishing data flow patterns.

Emphasis was placed on creating a modular, decoupled architecture that would facilitate maintainability, scalability, and ease of future enhancements.

By adhering to this structured methodology, the Expense GQL project was able to navigate the complexities of software development effectively, delivering a high-quality digital notebook application that meets the needs of its users.

# 3.System Architecture

The system architecture of the ed Expense GQL Unity project is designed to provide a robust and scalable foundation for the seamless functioning of the application. Comprising both frontend and backend components, the architecture is crafted to facilitate efficient data flow, modularization, and maintainability. Below is an in-depth exploration of the system architecture:

1. **Frontend Architecture**:
   * + The front end of Expense GQL is built using React.js, a popular JavaScript library for building user interfaces. React's component-based architecture enables the creation of reusable UI elements, promoting modularity and code reusability.
     + The frontend architecture follows a single-page application (SPA) model, where the entire application is contained within a single HTML page, with dynamic updates rendered through JavaScript.
     + Components such as note editors, note lists, user authentication forms, and navigation menus are encapsulated as reusable React components, promoting a modular and maintainable codebase.
2. **Backend Architecture**:
   * The backend of Expense GQL is powered by Node.js and Express.js, providing a fast and scalable server-side runtime environment. Express.js, a minimalist web framework for Node.js, facilitates the creation of robust API endpoints and middleware.
   * MongoDB, a NoSQL database, serves as the backend data store, storing user information, notes, and other application data. MongoDB's flexible schema allows for the storage of structured and unstructured data, accommodating the dynamic nature of note-taking.
   * The backend architecture follows a RESTful API design, with endpoints defined for CRUD operations (Create, Read, Update, Delete) on notes, user authentication, and other functionalities.
   * Authentication and authorization mechanisms, such as JSON Web Tokens (JWT) or OAuth, may be implemented to secure API endpoints and protect user data.
3. **Data Flow**:
   * The frontend communicates with the backend through HTTP requests, typically using fetch or axios libraries to perform CRUD operations on notes and interact with user authentication endpoints.
   * When a user interacts with the frontend interface, such as creating a new note or updating an existing one, the frontend sends an HTTP request to the corresponding API endpoint on the backend.
   * The backend processes the request, performs any necessary database operations, and sends a response back to the frontend, which updates the UI accordingly.
   * Data is exchanged between the frontend and backend in JSON format, facilitating interoperability and ease of parsing.

By adhering to a well-defined system architecture, the Expense GQL project ensures the reliability, performance, and maintainability of the application, enabling users to seamlessly create, organize, and manage their notes.

# 4. Technology Used

1. **MongoDB:**

MongoDB is a widely used NoSQL database that offers flexibility, scalability, and performance for modern web applications. Here's an overview of its key features:

|  |  |  |
| --- | --- | --- |
| • | **Document-Oriented** | : MongoDB stores data in flexible, JSON-like documents, allowing for easy representation of complex data structures without the need for rigid schemas. |
|  |
| • | **Scalability**: MongoDB's distributed architecture supports horizontal scaling through sharding, enabling applications to handle large volumes of data and high traffic loads. | |
| • | **Query Language**: MongoDB's query language supports rich and expressive queries, including CRUD operations, aggregation pipelines, and geospatial queries. | |
| • | **Indexes and Replication**: MongoDB provides support for indexes and replica sets, ensuring fast query performance and high availability. | |
| • | **Document Validation**: MongoDB allows for schema validation at the database level, enabling developers to enforce data integrity and consistency. | |
| • | **Community and Ecosystem**: MongoDB boasts a vibrant community and ecosystem, with comprehensive documentation, tutorials, and support resources available for developers. | |

1. **Express.js:**

Express.js is a minimalist web framework for Node.js that simplifies the development of web applications and APIs. Here are its main characteristics:

|  |  |  |
| --- | --- | --- |
| • | **Middleware** | : Express.js utilizes middleware to handle HTTP requests and responses,  enabling developers to modularize application logic and handle common tasks such as parsing request bodies, authentication, and error handling. |
|  |
| • | **Routing**: Express.js provides a robust routing mechanism that allows developers to define URL patterns and corresponding request handlers, making it easy to create RESTful APIs and organize application logic. | |
| • | **Template Engines**: Express.js supports various template engines such as EJS, Pug (formerly known as Jade), and Handlebars, allowing developers to generate dynamic HTML content on the server side. | |
| • | **Error Handling**: Express.js provides built-in error handling middleware and error routing mechanisms, making it easy to handle errors gracefully and maintain application stability. | |
| • | **Performance**: Express.js is known for its lightweight and high-performance nature, making it well-suited for building fast and scalable web applications. | |
| • | **Middleware Ecosystem**: Express.js has a rich ecosystem of third-party middleware and plugins that extend its functionality, offering features such as authentication, session management, and caching. | |

1. **React:**

React is a JavaScript library for building user interfaces, developed by Facebook. It is widely used for creating dynamic and interactive web applications. Here's what sets React apart:

|  |  |  |
| --- | --- | --- |
| • | **Component-Based** | : React follows a component-based architecture, where UI elements  are encapsulated as reusable components. This promotes modularity, code reusability, |
| and maintainability. |
| • | **Virtual DOM**: React utilizes a virtual DOM (Document Object Model) to efficiently update the UI in response to changes in application state. This minimizes DOM manipulation overhead and improves performance. | |
| • | **JSX Syntax**: React introduces JSX (JavaScript XML), a syntax extension that allows developers to write HTML-like code directly within JavaScript. This enhances readability and enables the creation of complex UI structures with ease. | |
| • | **Unidirectional Data Flow**: React enforces a unidirectional data flow, where data flows down from parent components to child components via props. This simplifies data management and makes it easier to reason about application state. | |
| • | **React Hooks**: React Hooks are a feature introduced in React 16.8 that allows developers to use state and other React features without writing class components. Hooks promote code reuse and improve code organization. | |
| • | **React Ecosystem**: React has a thriving ecosystem with a wide range of libraries, tools, and community-driven projects available for building React applications. This includes state management libraries like Redux, routing solutions like React Router, and UI component libraries like Material-UI. | |

1. **Node.js:**

Node.js is a server-side JavaScript runtime that enables developers to build scalable and highperformance web applications. Here's what makes Node.js unique:

|  |  |  |
| --- | --- | --- |
| • | **Event-Driven Architecture** | : Node.js uses an event-driven, non-blocking I/O model,  which allows it to handle concurrent requests efficiently. This makes it well-suited for time applications that require high concurrency and low latency. |
| building real- |
| • | **NPM Ecosystem**: Node.js comes with npm (Node Package Manager), a package manager that provides access to a vast ecosystem of open-source libraries and modules.  This makes it easy to integrate third-party functionality into Node.js applications. | |
| • | **Asynchronous Programming**: Node.js employs asynchronous programming techniques, such as callbacks, promises, and async/await, to handle I/O operations asynchronously.  This prevents blocking the event loop and ensures optimal resource utilization. | |
| • | **Scalability**: Node.js is highly scalable, thanks to its non-blocking, event-driven architecture. It can handle a large number of concurrent connections with minimal resource consumption, making it suitable for building microservices and distributed systems. | |
| • | **Cross-Platform Compatibility**: Node.js is cross-platform, meaning it can run on various operating systems, including Windows, macOS, and Linux. This makes it easy to develop and deploy Node.js applications across different environments. | |
| • | **Community and Support**: Node.js has a vibrant community and extensive documentation, with a wealth of tutorials, articles, and forums available for developers. This community-driven support ensures that developers have access to resources and assistance when building Node.js applications. | |

|  |
| --- |
|  |
|

|  |  |  |
| --- | --- | --- |
| **5.** | **Key Features and Functionality of Expense GQL** |  |

The Expense GQL application is designed to provide users with a versatile and intuitive platform for organizing and managing their notes effectively. Here are the key features and functionalities offered by iNotebook:

* 1. **User Authentication**: • Expense GQL incorporates a robust user authentication system that allows users to securely sign up, log in, and manage their accounts.
     + Users can create personalized accounts with unique usernames and passwords, ensuring data privacy and security.
  2. **CRUD Operations**: • Expense GQL facilitates CRUD (Create, Read, Update, Delete) operations on notes, allowing users to perform essential actions such as creating new notes, viewing existing notes, updating note content, and deleting unwanted notes.
     + The application provides intuitive user interfaces for performing CRUD operations, ensuring a seamless and hassle-free user experience.
  3. **Cross-Platform Compatibility**:
     + Expense GQL is designed to be cross-platform compatible, allowing users to access their notes seamlessly across different devices and operating systems.
     + The application may offer native mobile apps for iOS and Android devices, as well as a web-based interface for desktop users, ensuring a consistent user experience across all platforms.

By incorporating these key features and functionalities, Expense GQL aims to provide users with a comprehensive and user-friendly platform for managing their notes efficiently and effectively. Whether for personal use, academic pursuits, or professional endeavors, Expense GQL empowers users to stay organized, productive, and in control of their digital workspace.

# 8.Github Link

https://github.com/VimalKMGithub/GraphQLApp

# 9. References

**Google, YouTube**

**React Documentation:** [Getting Started – React (reactjs.org)](https://legacy.reactjs.org/docs/getting-started.html)

**Node Js Documentation:** [Index | Node.js v21.7.3 Documentation (nodejs.org)](https://nodejs.org/docs/latest/api/)

**Express Js Documentation:** [Express routing (expressjs.com)](https://expressjs.com/en/guide/routing.html) **MongoDB Documentation:** [MongoDB Documentation](https://www.mongodb.com/docs/) **,etc.**