

IGNITE FREAKS JOURNAL WEB APP

## A PROJECT REPORT

## *Submitted by*

**HEMA R**  **(910320104005)**

**ABDUL KAREEM** **GADHAFI A**  **(910320104001)**

**SANTHOSH M**  **(****910320104302)**

***in partial fulfilment for the award of the degree of***

# BACHELOR OF ENGINEERING

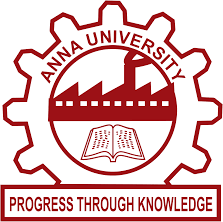
## in

COMPUTER SCIENCE AND ENGINEERING

**CHENDHURAN COLLEGE OF ENGINEERING AND TECHNOLOGY PUDUKKOTTAI - 622 507**

# ANNA UNIVERSITY: CHENNAI 600025

MAY 2024



# BONAFIDE CERTIFICATE

Certified that this project report titled **“ IGNITE FREAKS JOURNAL WEB APP ”** is the Bonafide work of **HEMA R (910320104005)** ,**ABDUL KAREEM GADHAFI A (910320104001) and SANTHOSH M (910320104302) ,** who carried out the project work under my supervision for the partial full fillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering .Certified further that to the best of my knowledge and belief the work reported here in does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion.

SIGNATURE SIGNATURE

**Mrs. M. KIRITHIKA DEVI, M.E.,Ph.D\* Mrs.PL.BABY** **SHALINI ,M.E.,**

## HEAD OF THE DEPARTMENT ASSISTANT PPROFESSOR

Dept. of Computer Science and Dept. of Computer Science and Engineering, Engineering,

Chendhuran college of Engineering Chendhuran college of Engineering

and Technology, and Technology,

Pudukkottai - 622507 Pudukkottai – 622507

The project report submitted for the viva voce held on ..............................

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We are much grateful to the management of Chendhuran College of Engineering and Technology, Pudukkottai for providing us an opportunity to undergo this project work.

We have immense pleasure and satisfaction in expressing our hearty thanks to our beloved managing trustee **Shri.AVM.SELVARAJ Chairman ,** who is the source of this great situation.

We express our sincere thanks to **Shri.R.VAIRAVAN Managing Director**, who provides support to the great success for this institution.

We are pleasure to expressing our hearty thanks to our beloved **C.E.O Dr.AVM.S.KARTHICK B.E., M.B.A .,Ph.D**., who is the source of sprit and strength of this great institution.

We express our sincere and special thanks to our Executive Director **Dr.M.PANDIKRISHNAN B.E., M.S., Ph.D**., who provided us the hope to complete this project.

We express our deep sense of gratitude and thank to our Principal **Dr.K.GANESH BABU,B.E.,M.Tech(IITM).,Ph.D(IITM**)., for permitting us to do the project work and allowing us to utilize all the facilities in the college.

We are grateful to **Mrs, M.KIRITHIKA DEVI M.E.,Ph.D\*** Head of the Department, for her valuable advice ,permission encouragement accorded to carry out this project successfully.

We express our heartfelt and special thanks to our project guide **Mrs.PL.BABY SHALINI M.E.,** Assistant professor, Department of Computer science and Engineering, our project guide for not only showing path but also guiding us throughout the path.

Finally above all , we give all glory to the **PARENTS** and **GOD,** who has been the source of spirit and strength to us throughout the project.

# 

# 

# ABSTRACT

# The "Ignite Freaks Journal App" is an innovative web-based application designed using the MERN stack (MongoDB, Express.js, React.js, and Node.js), tailored for authors to publish and submit their journals digitally. This project aims to provide a seamless and user-friendly platform that facilitates the efficient management, publication, and submission of journal documents, fostering a vibrant community of authors and readers. The application integrates a robust front-end built with React.js, offering a dynamic and responsive user interface that enhances the user experience. The back-end, developed using Node.js and Express.js, ensures secure and scalable server-side operations. MongoDB is employed as the database to store and manage data effectively, including user profiles, journal entries, and submission records, which allows for high performance and real-time data access. Key features of the "Ignite Freaks Journal App" include a streamlined submission process, real-time tracking of journal status, user authentication and authorization, and interactive tools for authors to edit and format their submissions. The app also provides mechanisms for feedback and communication between authors and the editorial team, which facilitates continual improvement of published content.

# 

# ABSTRACT ( TAMIL )

**திட்டச் சுருக்கம்**

"இக்னைட் ஃப்ரீக்ஸ் ஜர்னல் ஆப்" என்பது ஒரு புதுமையான இணைய அடிப்படையிலான பயன்பாடாகும், இது MERN ஸ்டேக்கைப் பயன்படுத்தி வடிவமைக்கப்பட்டுள்ளது (MongoDB, Express.js, React.js மற்றும் Node.js), ஆசிரியர்கள் தங்கள் பத்திரிகைகளை டிஜிட்டல் முறையில் வெளியிடுவதற்கும் சமர்ப்பிப்பதற்கும் வடிவமைக்கப்பட்டுள்ளது. இந்தத் திட்டம் ஒரு தடையற்ற மற்றும் பயனர் நட்பு தளத்தை வழங்குவதை நோக்கமாகக் கொண்டுள்ளது, இது திறமையான மேலாண்மை, வெளியீடு மற்றும் பத்திரிகை ஆவணங்களை சமர்ப்பித்தல், ஆசிரியர்கள் மற்றும் வாசகர்களின் துடிப்பான சமூகத்தை வளர்ப்பதற்கு உதவுகிறது.

பயன்பாடு React.js உடன் கட்டமைக்கப்பட்ட ஒரு வலுவான முன்-முனையை ஒருங்கிணைக்கிறது, பயனர் அனுபவத்தை மேம்படுத்தும் ஒரு மாறும் மற்றும் பதிலளிக்கக்கூடிய பயனர் இடைமுகத்தை வழங்குகிறது. Node.js மற்றும் Express.js ஐப் பயன்படுத்தி உருவாக்கப்பட்ட பின்-இறுதியானது, பாதுகாப்பான மற்றும் அளவிடக்கூடிய சர்வர் பக்க செயல்பாடுகளை உறுதி செய்கிறது. பயனர் சுயவிவரங்கள், பத்திரிகை உள்ளீடுகள் மற்றும் சமர்ப்பிப்பு பதிவுகள் உள்ளிட்ட தரவை திறம்படச் சேமிக்கவும் நிர்வகிக்கவும் தரவுத்தளமாக மோங்கோடிபி பயன்படுத்தப்படுகிறது, இது உயர் செயல்திறன் மற்றும் நிகழ்நேர தரவு அணுகலை அனுமதிக்கிறது.

"இக்னைட் ஃப்ரீக்ஸ் ஜர்னல் ஆப்" இன் முக்கிய அம்சங்களில், நெறிப்படுத்தப்பட்ட சமர்ப்பிப்பு செயல்முறை, ஜர்னல் நிலையை நிகழ்நேர கண்காணிப்பு, பயனர் அங்கீகாரம் மற்றும் அங்கீகாரம் மற்றும் ஆசிரியர்கள் தங்கள் சமர்ப்பிப்புகளைத் திருத்துவதற்கும் வடிவமைப்பதற்கும் ஊடாடும் கருவிகள் ஆகியவை அடங்கும். இந்த பயன்பாடு, ஆசிரியர்கள் மற்றும் ஆசிரியர் குழுவிற்கு இடையே கருத்து மற்றும் தகவல்தொடர்புக்கான வழிமுறைகளை வழங்குகிறது, இது வெளியிடப்பட்ட உள்ளடக்கத்தை தொடர்ந்து மேம்படுத்த உதவும.

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO.**

**ABSTRACT** iv

**ABSTRACT(TAMIL)**  v

**LIST OF FIGURES** viii

**LIST OF ABBREVIATION** ix

**1 INTRODUCTION 1**

* 1. SCOPE OF THIS PROJECT 1

1.2 OBJECTIVES OF THIS PROJECT 2

**2 REQUIREMENTS 4**

2.1 ENVIRONMENT 4

2.2 DATABASE 5

**3 INSTALLATIONS 7**

3.1 NODE JS 8

3.2 NODE PACKAGE MANAGER (NPM) 9

3.3 CREATE LOCALHOST 9

**4 APPLICATION DESIGN**  **11**

4.1 SEQUENCE DIAGRAM 11

4.2 USE CASE DIAGRAM 12

4.3 STATE DIAGRAM 12

**5**  **TESTING 13**

5.1 UNIT TESTING 13

5.2 INTEGRATION TESTING 14

5.3 SYSTEM TESTING 16

**6 SAMPLE CODE 18**

6.1 INDEX JS 18

6.2 COMPONENTS 19

6.2.1 HOME 19

6.2.2 ABOUT US 21

6.2.3 AIM AND SCOPE 22

6.2.4 ARCHIVES 23

6.2.5 ONLINE SUBMISSION 25

6.2.6 CONTACT US 30

6.3 MONGODB CONNECTION 31

**7** **OUTPUT SCREEN** 35

**8 CONCLUSIONS 37**

8.1 FUTURE SCOPE 37

**REFERENCES 38**

## 

## **LIST OF FIGURES**

## **FIGURE NO. NAME OF FIGURE PAGE NO.**

## 4.1 SEQUENCE DIAGRAM 11

## 4.2 USE CASE DIAGRAM 12

## 4.3 STATE DIAGRAM 12

## 5.1 UNIT TESTING LIFE CYCLE 14

## 

## 5.2 TYPES OF INTEGRATION TESTING 15

## 5.3 TYPES OF SYSTEM TESING 17

## 7.1 HOME PAGE 35

## 7.2 ONLINE SUBMISSION 35

## 7.3 ARCHIVES PAGE 36

## 7.4 DATABASE COLLECTIONS 36

## LIST OF ABBREVIATION

## **MERN** MongoDB, Express.js, React.js, Node.js

## **JS** JavaScript

## **NPM** Node Package Manager

## **API** Application Programming Interface

## **JSON** JavaScript Object Notation

## **CLI** Command-Line Interface

## **IDE** Integrated Development Environment

## **CMD** Command Prompt

## **UI** User Interface

## **UX** User Experience

## **DB** Database

## 

**CHAPTER - 1**

# INTRODUCTION

The "Ignite Freaks Journal App" is a sophisticated digital platform designed to revolutionize the way journal entries are published and managed within academic and creative communities. Leveraging the robust capabilities of the MERN stack (MongoDB, Express.js, React.js, and Node.js), this application offers an all-encompassing solution for the submission, review, and publication of scholarly articles and creative writings. The app addresses a critical need for a more streamlined, efficient, and user-friendly process that facilitates interaction between authors, reviewers, and readers. Key features include a secure login system, an intuitive submission interface, a dynamic reviewing module, and a comprehensive archival system for easy access to published works. The introduction of this platform is intended to simplify the complexities often associated with academic publishing, reduce turnaround times for publication, and provide a scalable solution that can grow with the needs of the user community. Through the "Ignite Freaks Journal App," we aim to empower authors and editors, enhance academic collaboration, and increase the visibility and accessibility of valuable intellectual content.

# 1.1 SCOPE OF THIS PROJECT

# The scope of the "Ignite Freaks Journal App" project encompasses the development of a robust digital platform tailored for the publication and management of journal entries. This application, built using the MERN stack (MongoDB, Express.js, React.js, and Node.js), aims to facilitate a seamless interface for authors to submit and revise their work, and for reviewers and editors to manage the review process effectively. The platform will support user registration and authentication, journal submission with metadata inputs, a comprehensive review and approval workflow, and an archival system for published journals. Moreover, the application will provide a responsive design to ensure accessibility on various devices, implement rigorous security measures to protect user data and comply with data protection regulations, and offer an online submission system that includes status tracking for users. While the initial release will focus on essential features and English language support, future enhancements may include integration with academic databases, additional language options, and more advanced content management capabilities. The deliverables for this project include the deployed web application, complete source code, detailed documentation for system setup and use, and a project report that outlines all developmental aspects and technical specifications.

# 1.2 OBJECTIVES OF THIS PROJECT

# The objectives of the "Ignite Freaks Journal App" project are multifaceted, aiming to streamline and enhance the academic and creative publication process through a user-friendly digital platform. A primary goal is to provide a seamless mechanism for authors to submit their journal entries, which includes the capability for uploading documents, categorizing content, and managing revisions. The platform will also support a structured review process, enabling reviewers and editors to efficiently assess submissions, provide feedback, and finalize publications. Another key objective is to facilitate easy access and retrieval of published content through a well-organized archive that offers robust search functionality. The application aims to improve the overall user experience for all stakeholders involved, including authors, editors, and readers, by ensuring intuitive navigation and responsive design. Additionally, maintaining high security standards to safeguard user data and intellectual property is a crucial objective. Ultimately, the project seeks to deliver a reliable and scalable digital journal management system that promotes academic dialogue and knowledge dissemination in the digital age.

# CHAPTER - 2

# REQUIREMENTS

**Front-End Requirements**

**Responsive Design:**

**Objective:** Ensure that the web application is responsive and offers an optimal viewing and interaction experience across various devices, including desktops, tablets, and smartphones.

**Technology:** Employ React.js with CSS frameworks like Material-UI or Ant Design for responsive components, and media queries to enhance custom responsiveness.

**Testing:** Conduct comprehensive responsive testing using browser-based tools and physical device testing to ensure consistent usability across different screen sizes.

Browser Developer Tools:

**Objective:** Effectively utilize browser developer tools to debug, optimize, and ensure reliable front-end performance of the web application.

**Skills:** Mastery in using Chrome DevTools or similar tools in other browsers (Firefox, Safari) to diagnose issues, optimize performance, and debug JavaScript and React components.

**Application:** Integrate these tools into the daily development workflow to continuously improve code quality and front-end functionality.

**Back-End Requirements**

**Databases:**

**Objective:** Implement MongoDB for efficient management of the application’s data, taking advantage of its flexibility with schema-less structure which is ideal for handling dynamic, unstructured data.

**Technology:** Use MongoDB along with Mongoose for schema definition, data modeling, and transaction management to enhance database interactions.

**Skills:** In-depth knowledge of MongoDB operations, performance tuning, and security practices, ensuring data integrity and access speed.

**API Development:**

**Objective:** Build robust RESTful APIs using Node.js and Express.js to handle requests between the front-end and the database effectively.

**Technology:** Utilize Express.js framework for routing and middleware capabilities, ensuring seamless data flow and efficient request handling.

Best Practices: Implement JWT for secure API authentication and authorization, apply CORS policies for security, and use tools like Postman for API testing and documentation.

**2.1 ENVIRONMENT**

For the "Ignite Freaks Journal Web App" project, built using the MERN stack, it's essential to have a robust environment setup that supports the full capabilities of MongoDB, Express.js, React.js, and Node.js. The development environment is best set up on operating systems such as Windows, macOS, or Linux, depending on developer preference and compatibility requirements. Development should be facilitated through a powerful IDE or code editor such as Visual Studio Code, which provides comprehensive support for JavaScript and Node.js, along with helpful extensions for React and MongoDB. The project requires Node.js to be installed, with NPM (Node Package Manager) used to manage and install project dependencies. React.js serves as the front-end framework, making use of its ecosystem including tools like Create React App for scaffolding new projects, and Babel plus Webpack for transpiling and bundling the application. MongoDB should be set up either locally for development purposes or using a cloud provider like MongoDB Atlas for both development and production environments, ensuring data persistence and scalability.

For the back-end, Express.js runs within a Node.js server, ideally managed via process managers like PM2, which helps in monitoring and keeping the server running continuously. API testing can be efficiently handled through Postman or similar tools, whereas front-end functionality can be tested using browser-based tools and frameworks such as Jest for unit testing and React Testing Library for React components.Source control is managed via Git, with repositories hosted on platforms like GitHub or GitLab, facilitating version control and collaborative development. The deployment of the application can be executed on cloud platforms such as Heroku, AWS, or Azure, which support the MERN stack's requirements and provide scalability options for handling increased traffic and data load. This comprehensive environment setup ensures that the development, testing, and production stages of the "Ignite Freaks Journal Web App" are efficient, scalable, and well-supported across different stages of the development lifecycle.

**2.2 DATABASE**

For the "Ignite Freaks Journal Web App" built with the MERN stack, the optimal choice for database management is MongoDB, a NoSQL database that complements the flexibility and scalability demands of modern web applications. MongoDB is particularly well-suited for handling large volumes of unstructured and semi-structured data which is typical in dynamic web environments like online journals.In this project, MongoDB will be utilized to store and retrieve all data related to user profiles, journal entries, comments, and other interactive features. The database will support high read and write speeds, ensuring that users experience minimal latency when accessing and posting content. MongoDB’s schema-less nature allows for easy modifications in the data structure, which is advantageous during iterative development and when deploying updates to the application.The integration of MongoDB with Express.js in the back-end is streamlined through Mongoose, an Object Data Modeling (ODM) library for MongoDB and Node.js. Mongoose provides a straightforward schema-based solution to model application data. It includes built-in type casting, validation, query building, and business logic hooks which makes the development process more intuitive and efficient.

For development purposes, MongoDB can be run locally on a developer's machine, while for production, it can be hosted on cloud platforms like MongoDB Atlas. Atlas offers managed MongoDB services that include automated scaling, backup, and monitoring services, which significantly reduce the management overhead and enhance the reliability and security of the database.This setup ensures that the "Ignite Freaks Journal Web App" leverages a robust, scalable, and flexible database environment that is capable of supporting the app's functionality and user growth efficiently.

\\

**CHAPTER – 3**

**INSTALLATIONS**

**Front-End Installation** (ID

**Text Editor or Integrated Development Environment E):**

**Purpose:** Essential for writing and managing your code efficiently.

**Options:** Choose from popular editors such as Visual Studio Code, Atom, or Sublime Text, all of which support JavaScript and React.js development. Visual Studio Code is particularly recommended due to its extensive range of extensions for JavaScript, React, and Node.js, which can enhance productivity and streamline the development process.

**Web Browser:**

Purpose: Necessary for testing and debugging the web application.

Recommendations: Install a modern web browser like Google Chrome, Mozilla Firefox, or Microsoft Edge. Google Chrome is highly recommended for its comprehensive Dev Tools, which are incredibly useful for debugging JavaScript and monitoring network activity.

**Back-End Installation**

**Programming Language:**

**Primary Language:** Node.js is the backbone for server-side operations.

**Installation:** Download and install Node.js directly from the official Node.js website. Installing Node.js will also install npm (Node Package Manager), which is crucial for managing the back-end libraries and dependencies.

**Server Framework:**

**Framework:** Express.js, a minimal and flexible Node.js web application framework, provides a robust set of features for web and mobile applications.

**Setup:** Install Express.js via npm by running npm install express in your project directory. This command sets up Express along with any other necessary Node.js packages.

**3.1 NODE JS**

For the "Ignite Freaks Journal Web App" utilizing the MERN stack, Node.js serves as a foundational element. This runtime environment is essential for running the server-side code of the application, providing a scalable and efficient platform for handling web server tasks. The inclusion of Node.js enables the use of JavaScript on the backend, creating a uniform programming language across both the front and backend, which simplifies development and reduces context switching for developers. Node.js interacts seamlessly with Express.js, a server framework chosen for its minimalist structure and flexibility, to manage HTTP requests, route handling, and middleware integration efficiently. The application utilizes Express to structure its server setup, manage API endpoints, and handle middleware functions for tasks like error handling and request processing. This setup allows for high performance and easy scalability, catering well to the dynamic needs of a web application focused on user interactivity and real-time data handling.To integrate Node.js into the "Ignite Freaks Journal Web App," it is installed via its official website, ensuring compatibility with additional tools and libraries like NPM (Node Package Manager). NPM facilitates efficient management of project dependencies, and the installation of numerous packages necessary for further enhancing the app’s functionality, such as Mongoose for database interactions, and body-parser for parsing incoming request bodies. This cohesive environment ensures that the app is robust, maintainable, and capable of evolving with user needs and technological advancements.

**3.2 NODE PACKAGE MANAGER(NPM)**

For the "Ignite Freaks Journal Web App" developed using the MERN stack, the Node Package Manager (NPM) plays a critical role in managing the software dependencies required for both the front-end and back-end development processes. NPM is the default package manager for Node.js, allowing developers to install, update, and manage reusable modules from the npm registry. The effectiveness of NPM in this project cannot be overstated; it streamlines the installation of key packages like React for the frontend, Express for the backend server framework, Mongoose for MongoDB database interaction, and many other utilities such as Babel for JavaScript transcompiling, Webpack for module bundling, and various testing frameworks like Mocha and Chai. These packages are vital for creating a dynamic and responsive journal web application. NPM also enhances project maintainability through its version control capabilities. By defining package versions in the **package.json** file, developers can ensure consistent environments across all stages of deployment, from development to production. This eliminates issues arising from package updates or discrepancies between development environments, facilitating a more stable and predictable development process.

# 3.3 CREATE LOCALHOST

Express enables developers to quickly configure a server environment on their local machines, which acts as the backbone for both the serving of static files and the handling of various API requests necessary for the journal app. To create a localhost environment for this project, you begin by initializing a new Node.js project with NPM, which sets up the package.json file. You then install Express via NPM and write a simple server script.

Here’s a basic outline on setting up your local server with Express:

**Initialize your project:**

Run npm init -y in your project directory to create a package.json file, which will manage all dependencies.

**Install Express:**

Execute npm install express to add Express to your project.

Create your server file:

Create a file named server.js in your project directory.

**Run your server:**

Start your server by running node server.js in the command line.

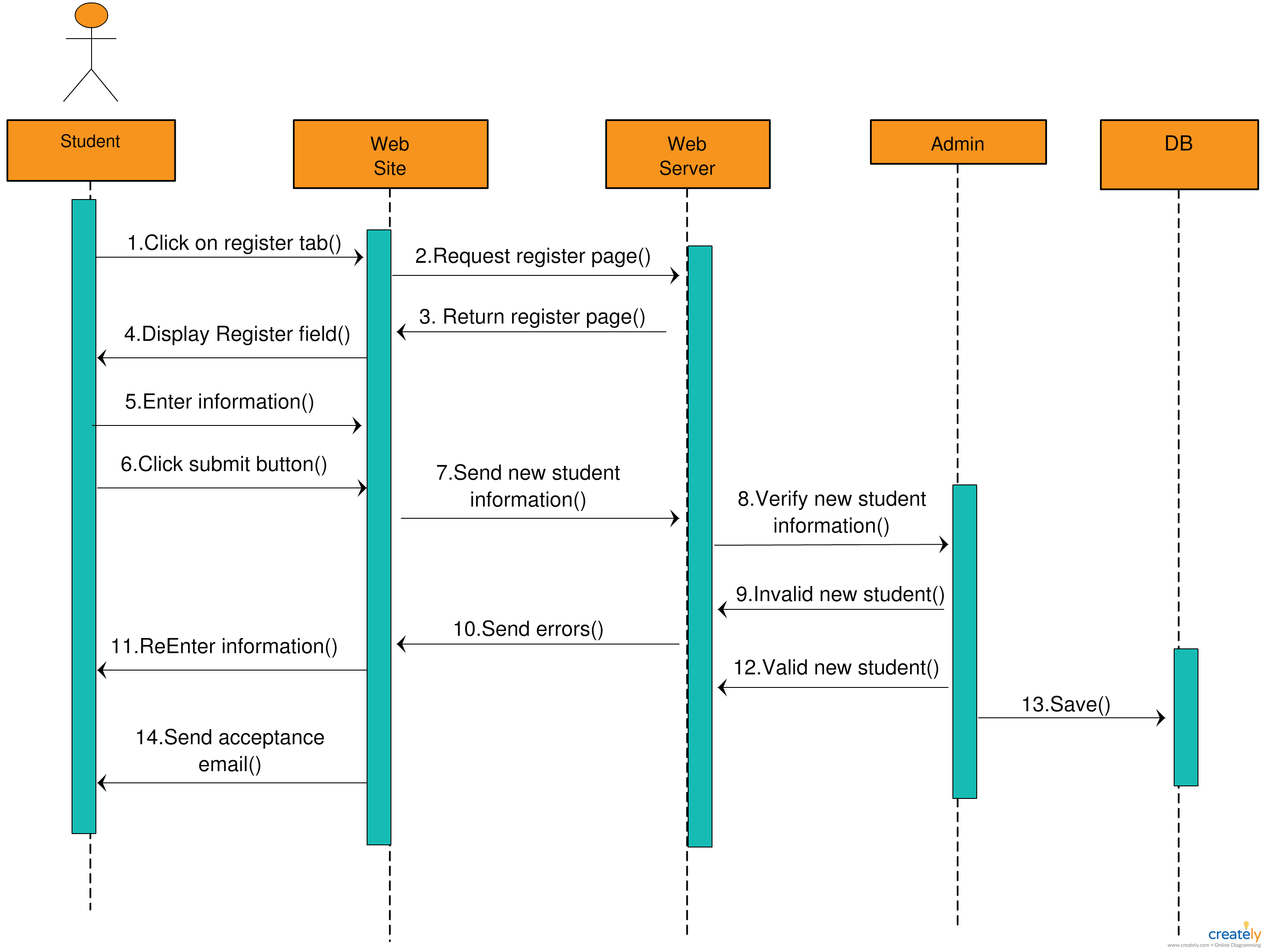
Access your application by opening a web browser.

**CHAPTER – 4**

**APPLICATION DESIGN**

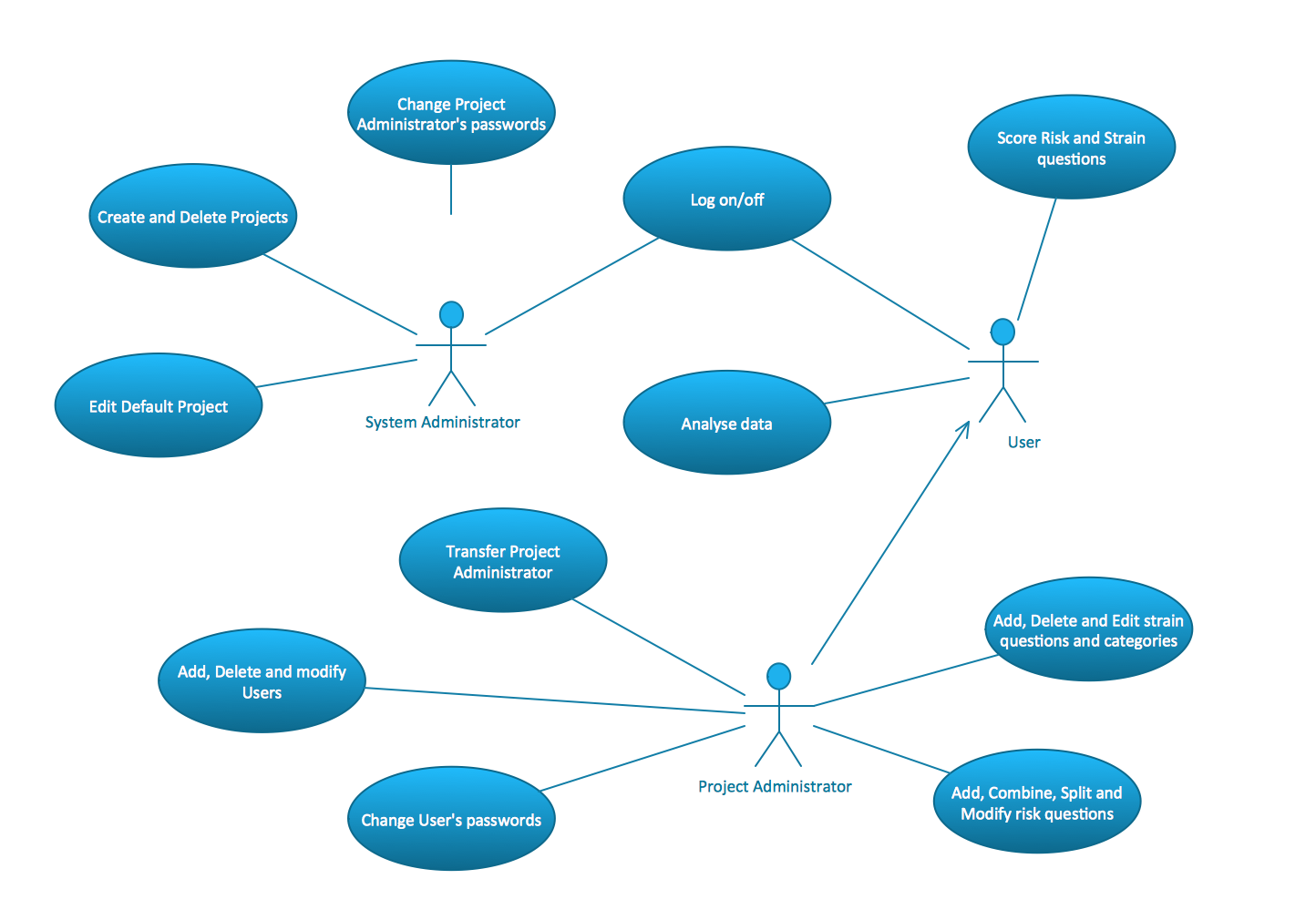
The "Ignite Freaks Journal Web App" is meticulously designed using the MERN stack to offer a comprehensive and interactive platform for journaling and personal reflection. The application design emphasizes user experience and functionality, integrating React for the front-end, Express and Node.js for the backend, and MongoDB for data management.The front-end of the application is built with React, leveraging its component-based architecture to ensure a responsive and dynamic user interface. Key UI components include a dashboard, journal entry forms, and a browsing section for past entries. React Router is employed for seamless navigation between components without reloading the page, enhancing the application's speed and user experience. The design also incorporates responsive design principles to ensure accessibility across various devices and screen sizes.

**4.1 SEQUENCE DIAGRAM**



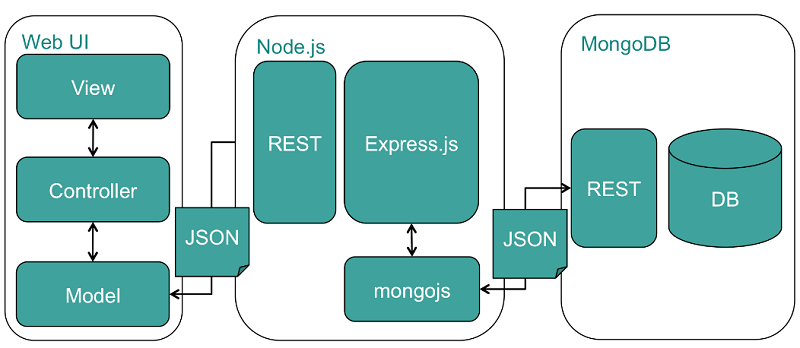
**Figure 4.1 Sequence Diagram**

**4.2 USECASE DIAGRAM**



**Figure 4.2 Use case Diagram**

**4.3 STATE DIAGRAM**



**Figure 4.3 State Diagram**

**CHAPTER – 5**

**TESTING**

For the "Ignite Freaks Journal Web App" built using the MERN stack, a comprehensive testing strategy is implemented to ensure the application's reliability, security, and performance. This strategy includes various levels of testing: unit testing, integration testing, and system testing.

**5.1 UNIT TESTING**

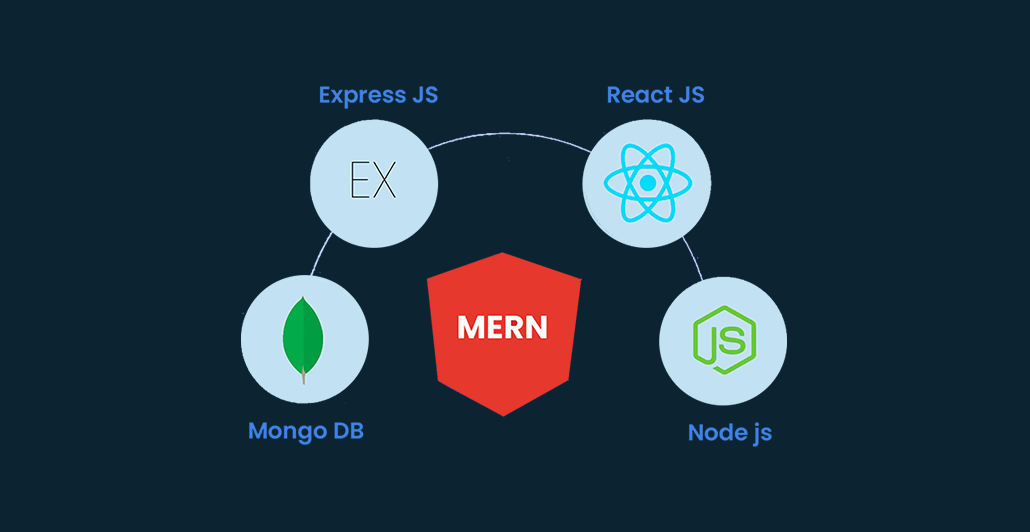
For the "Ignite Freaks Journal Web App" developed using the MERN stack, unit testing plays a critical role in ensuring each module functions correctly in isolation. To implement unit testing effectively, we utilize Jest alongside specific testing libraries tailored for both the backend and frontend components of our stack.

### **Backend Unit Testing**

On the backend, which is built with Node.js and Express, we employ Jest in conjunction with Supertest to test individual API endpoints. These tests focus on checking that each API function handles requests and returns the expected responses, including error handling paths. For instance, we write tests to verify that the journal entry creation endpoint correctly processes valid data and returns the expected success message, as well as tests to ensure that attempts to submit incomplete or invalid data result in the appropriate error responses.

### **Frontend Unit Testing**

For the frontend, built with React, Jest is used together with the React Testing Library. This combination allows us to effectively test React components in isolation, ensuring that they behave as expected under various states and props. For example, tests are crafted to confirm that the journal entry form renders correctly, handles user inputs as intended, and triggers the correct function upon submission.



**Figure 5.1 UNIT TESTING LIFE CYCLE**

**5.2 INTEGRATION TESTING**

The integration testing involves simulating real user scenarios to test how the frontend components communicate with the backend APIs and how the backend processes those requests, interacts with the database, and returns the appropriate responses. We employ tools like Supertest for backend API testing and Cypress for end-to-end frontend interactions, which allow us to test the complete flow from the user interface down to the database.

**Test Cases:**

Specific test cases are designed to cover all the critical interactions within the application. For example, tests verify the complete process of creating a journal entry, from filling out the form in the React application, submitting the form to the Express server, processing the request, saving the data to MongoDB, and finally confirming that the correct response is received and appropriately displayed on the frontend.

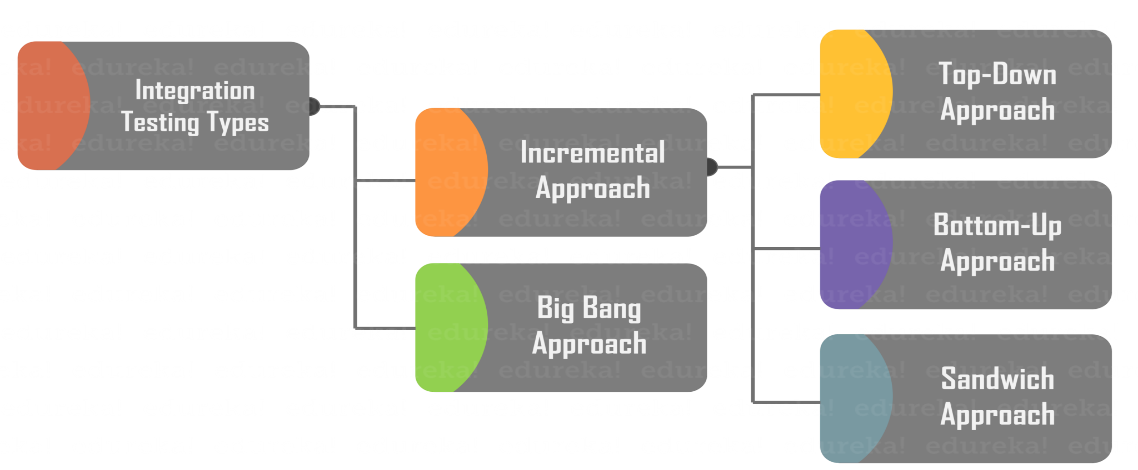
**Error Handling and Edge Cases:**

Integration tests also focus on error handling and edge cases to ensure robustness. For instance, we test how the application responds if the database is unreachable or if the data sent to the server is incomplete or malformed. These tests help verify that the application provides meaningful error messages and does not crash under unexpected conditions.

**Continuous Integration (CI):**

To facilitate frequent and reliable testing, integration tests are automated and run as part of the continuous integration pipeline using platforms like Jenkins or GitHub Actions. This ensures that every change in the codebase is tested, minimizing the chances of integration issues before deployment.

By conducting thorough integration testing, the "Ignite Freaks Journal Web App" team ensures that all parts of the system work together properly, providing a smooth, error-free user experience. This not only improves the quality of the application but also reduces the time and cost associated with manual testing and debugging in later stages of development.



**Figure 5.2 TYPES OF INTEGRATION TESTING**

**5.3 SYSTEM TESTING**

The system testing for the "Ignite Freaks Journal Web App" encompasses several key aspects, including functionality, usability, performance, and security. The tests are designed to ensure that all components integrated during development work together harmoniously across different platforms and user conditions.

**Functionality and Usability Testing**

Functionality tests verify that all the app’s features work as intended from end to end. This includes testing user interactions with the journal entry features, authentication mechanisms, data retrieval, and response handling from the backend. Usability testing focuses on the user experience aspects, such as ease of navigation, clarity of the user interface, and the responsiveness of the application on various devices.

**Performance Testing**

Performance testing is critical, particularly for ensuring that the application handles expected and peak load conditions effectively. Tools like JMeter or LoadRunner are used to simulate multiple users accessing the application simultaneously to ensure that the system remains stable and maintains a high level of performance under stress.

**Security Testing**

Security is paramount, given the nature of the personal data handled by the journal app. Security testing involves checking the application for vulnerabilities such as SQL injections, XSS (Cross-Site Scripting), and ensuring that data transmissions are encrypted using SSL/TLS. The tests also verify that user data is adequately protected both in transit and at rest.

**Automated Testing Tools**

Automated testing tools play a significant role in system testing by providing a way to repeatedly test the application across all these parameters efficiently. Selenium, for example, is used for automating browser activities for usability and functional testing, while tools like OWASP ZAP are used to automate security scans.



**Figure 5.3 TYPES OF SYSTEM TESTING**

**CHAPTER - 6**

**SAMPLE CODE**  
The "app.js" file typically refers to the JavaScript file that contains the main code for an application. Its importance lies in its role as the central hub for implementing the functionality and behavior of the application on the client-side.

**app.js:**

import React from "react";

import Class from "./Class";

import "./App.css";

import Navabar from "./Navbar";

export default function App() {

return (

<div className="container">

<Navabar />

</div>

);

}

**6.1 INDEX JS**

import React from "react";

import ReactDOM from "react-dom/client";

import "./index.css";

import reportWebVitals from "./reportWebVitals";

import App from "./App";

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);reportWebVitals();

**6.2 COMPONENTS**

**6.2.1 HOME**

import React, { useEffect, useState } from "react";

import "./Home.css";

import pic1 from "../Images/duck.jpg";

import Footer from "../Footer";

import "../About Us/About.css";

import Scrollingbar from "../Scrollingbar";

import h from "../Images/bkg.jpg";

import Topics from "../Topics/Topics";

function Home() {

const [typedText, setTypedText] = useState("");

const fullText = "IJRASTEM";

useEffect(() => {

const typeText = (index) => {

if (index < fullText.length) {

setTimeout(() => {

setTypedText((prevText) => prevText + fullText[index]);

typeText(index + 1);

}, 200); // Adjust the typing speed (milliseconds)

} else {

// Restart typing animation after a delay

setTimeout(() => {

setTypedText("");

typeText(0);

}, 2000); // Adjust the delay before restarting typing

}

};

typeText(0); // Start typing animation

return () => clearTimeout(); // Cleanup function

}, []); // Run once on mount

return (

<div className="home">

<Scrollingbar />

<div className="aim-scope">

<img src={pic1} alt="hello" className="i1" />

</div>

<div className="about-us-container">

<h2>About Us</h2>

<p>

Ignite Freaks Journal App is an international, peer-reviewed, open-access, online

journal. It aims to provide an online platform to promote excellence

and advancement in the different areas of arts, science, technologies,

education, and management (IJRASTEM).

</p></div><div>

<img src={h} alt="hi" className="kk" /></div>

<div className="aim-scope-content">

<h2>Aim & Scope</h2><p>

The primary aim is to offer academicians, researchers, and students

from all around the world specialized services for publishing research

and review articles. It is our aim to present the most innovative,

genuine, and current research findings through each of your

submissions. We handle your requirements and make sure the leading

academics in the field quickly publish your research. Priority is

given to high-quality papers that focus on fundamental ideas that have

the potential to make significant contributions to the field of

research. We publish authentic research in the fields of the arts,

science, technology, education and management.

</p>

</div>

<Topics />

<Footer />

</div>

);

}

export default Home;

**6.2.2 ABOUT US**

import React from "react";

import "./About.css";

import g from "../Images/tara.jpg";

import Footer from "../Footer";

import Scrollingbar from "../Scrollingbar";

const AboutUs = () => {

return (

<div className="about">

<Scrollingbar />

<div className="about-us-container">

<h2>About Us</h2>

<p>

Ignite Freaks Journal App is an international, peer-reviewed, open-access, online

journal. It aims to provide an online platform to promote excellence

and advancement in the different areas of arts, science, technologies,

education, and management (IJRASTEM).

</p>

</div>

<div>

<img src={g} alt="hi" className="kk" />

</div>

<Footer />

</div>

);

};

export default AboutUs;

**6.2.3 AIM AND SCOPE**

import React from "react";

import "./aim.css";

import h from "../Images/bkg.jpg";

import Footer from "../Footer";

import Scrollingbar from "../Scrollingbar";

function AimScope() {

return (

<div className="aim">

<Scrollingbar />

<div className="aim-scope-content">

<h2>Aim & Scope</h2>

<p>

The primary aim is to offer academicians, researchers, and students

from all around the world specialized services for publishing research

and review articles. It is our aim to present the most innovative,

genuine, and current research findings through each of your

submissions. We handle your requirements and make sure the leading

academics in the field quickly publish your research. Priority is

given to high-quality papers that focus on fundamental ideas that have

the potential to make significant contributions to the field of

research. We publish authentic research in the fields of the arts,

science, technology, education and management.

</p>

</div>

<div>

<img src={h} alt="hi" className="kk" />

</div>

<Footer />

</div>

);

}

export default AimScope;

**6.2.4 ARCHIVES**

import React, { useState, useEffect } from "react";

import axios from "axios";

import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";

import { faDownload } from "@fortawesome/free-solid-svg-icons";

import "./archives.css";

function Archives() {

const [allFiles, setAllFiles] = useState(null);

useEffect(() => {

fetchFiles();

}, []);

const fetchFiles = async () => {

try {

const response = await axios.get("http://localhost:3000/get-files");

setAllFiles(response.data.data);

} catch (error) {

console.error("Error fetching files:", error);

}

};

const downloadFile = (fileUrl) => {

const anchor = document.createElement("a");

anchor.href = fileUrl;

anchor.download = "downloaded.docx"; // Set desired file name with extension

document.body.appendChild(anchor);

anchor.click();

document.body.removeChild(anchor);

};

return (

<div>

<div className="uploaded">

<div className="output-div">

{allFiles ? (

allFiles.map((file) => (

<div className="inner-div" key={file.\_id}>

<h6>Title: {file.title}</h6>

<button

className="btn-download"

onClick={() =>

downloadFile(

`http://localhost:3000/files/${file.word}`

//https://journal-app-backend-3466.onrender.com

)

}

>

<FontAwesomeIcon icon={faDownload} /> Download Word

</button>

</div>

))

) : (

<p>No files available</p>

)}

</div>

</div>

</div>

);

}export default Archives;

**6.2.5 ONLINE SUBMISSION**

import "./OnlineSubmission.css";

import React, { useState } from "react";

import axios from "axios";

import Footer from "../Footer";

function OnlineSubmission() {

const [topic, setTopic] = useState("");

const [title, setTitle] = useState("");

const [name, setName] = useState("");

const [corrA, setCorrA] = useState("");

const [corrAE, setCorrAE] = useState("");

const [mobile, setMobile] = useState("");

const [pages, setPages] = useState(0);

const [affiliation, setAffiliation] = useState("");

const [file, setFile] = useState(null);

const [successMessage, setSuccessMessage] = useState("");

const handleTopicChange = (event) => {

setTopic(event.target.value);

};

const handleTitleChange = (event) => {

setTitle(event.target.value);

};

const handleNameChange = (event) => {

setName(event.target.value);

};

const handleCorrAChange = (event) => {

setCorrA(event.target.value);

};

const handleCorrAEChange = (event) => {

setCorrAE(event.target.value);

};

const handleMobileChange = (event) => {

setMobile(event.target.value);

};

const handleAffiliationChange = (event) => {

setAffiliation(event.target.value);

};

const handlePagesChange = (event) => {

setPages(event.target.value);

};

const handleFileChange = (event) => {

setFile(event.target.files[0]);

};

const handleSubmit = async (event) => {

event.preventDefault();

const formData = new FormData();

formData.append("topic", topic);

formData.append("title", title);

formData.append("name", name);

formData.append("corrA", corrA);

formData.append("corrAE", corrAE);

formData.append("mobile", mobile);

formData.append("affiliation", affiliation);

formData.append("pages", pages);

formData.append("file", file);

try {

const response = await axios.post(

"http://localhost:3000/upload-files",

formData,

{

headers: {

"Content-Type": "multipart/form-data",

},

}

);console.log(response.data);

setSuccessMessage("File uploaded successfully!");

// Reset form fields after successful submission

setTopic("");

setTitle("");

setName("");

setCorrA("");

setCorrAE("");

setMobile("");

setPages(0);

setAffiliation("");

setFile(null);

} catch (error) {

console.error("Error uploading file:", error);

// Handle error

}

};

return (

<div>

<div className="goose">

<h2>Submit Your Information</h2>

<form onSubmit={handleSubmit}>

<label htmlFor="topic"> Topic / Domain:</label>

<br />

<input

type="text"

id="topic"

value={topic}

onChange={handleTopicChange}

required

/> <label htmlFor="title"> Title:</label>

<br />

<input

type="text"

id="title"

value={title}

onChange={handleTitleChange}

required

/> <label htmlFor="name"> Name:</label>

<br />

<input

type="text"

id="name"

value={name}

onChange={handleNameChange}

required

/>

<label htmlFor="corrA"> Corresponding Author Name:</label>

<br />

<input

type="text"

id="corrA"

value={corrA}

onChange={handleCorrAChange}

required

/> <label htmlFor="corrAE"> Corresponding Email:</label>

<br />

<input

type="email"

id="corrAE"

value={corrAE}

onChange={handleCorrAEChange}

required

/><label htmlFor="mobile"> Mobile:</label>

<br />

<input

type="tel"

id="mobile"

value={mobile}

onChange={handleMobileChange}

required

/><label htmlFor="affiliation"> Affiliation:</label>

<br />

<input

type="text"

id="affiliation"

value={affiliation}

onChange={handleAffiliationChange}

required

/><label htmlFor="pages"> No. of Pages in Document:</label>

<br />

<input

type="number"

id="pages"

value={pages}

onChange={handlePagesChange}

required

/><label htmlFor="file">Attach Word File:</label>

<br />

<input

type="file"

id="file"

onChange={handleFileChange}

accept=".doc,.docx"

required

/><button type="submit">Submit</button>

</form>

{/\* Success Message Popup \*/}

{successMessage && (

<div className="success-message">

<p>{successMessage}</p>

</div>

)}

</div>

<Footer />

</div>

);

}

export default OnlineSubmission;

**6.2.6 CONTACT US**

import React from "react";

import "./contact.css";

import k from "../Images/contact.jpg";

import { FaEnvelope, FaMobileAlt, FaMapMarkerAlt } from "react-icons/fa";

import Footer from "../Footer";

const Contact = () => {

return (

<div>

<div className="contact-container">

<h1>Contact Us</h1>

<div className="contact-item">

<FaEnvelope />

<span>Email: your.email@example.com</span>

</div>

<div className="contact-item">

<FaMobileAlt />

<span>Contact Number: +91-9488058361</span>

</div>

<div className="contact-item">

<FaMapMarkerAlt />

<span>

No.6/9, Kamarajapuram 17th Street 1st Cross, Pudukkottai-622001,

Tamilnadu, India

</span>

</div>

<div>

<img src={k} alt="hi" className="kl" />

</div>

</div>

<Footer />

</div>

);};

export default Contact;

**6.3 MONGO DB CONNECTION**

The specific functionality of a "connectiondb.js" file can vary depending on the context and the specific requirements of the application. However, based on common conventions and practices in web development, a "connectiondb.js" file typically includes code related to establishing a connection to a database.

**connectiondb.js:**

const express = require("express");

const app = express();

const path = require("path");

const mongoose = require("mongoose");

const cors = require("cors");

const WordSchema = require("./wordDetails"); // Assuming wordDetails.js exports the WordSchema model

// Middleware

app.use(express.json());

app.use(

cors({

origin: "\*",

credentials: true,

methods: ["GET", "POST", "PUT", "DELETE"],

})

);

app.use("/files", express.static("files"));

// MongoDB connection

const mongoUrl ="mongodb+srv://admin:admin@journal.p6kw1o3.mongodb.net/?retryWrites=true&w=majority&appName=Journal";

mongoose.connect(mongoUrl)

.then(() => console.log("Connected to database"))

.catch((e) => console.log("Database connection error:", e));

// Multer configuration for file uploads

const multer = require("multer");

const storage = multer.diskStorage({

destination: function (req, file, cb) {

cb(null, "./files");

},

filename: function (req, file, cb) {

const uniqueSuffix = Date.now();

cb(null, uniqueSuffix + file.originalname);

},

});

const upload = multer({ storage: storage });

// Routes

app.post("/upload-files", upload.single("file"), async (req, res) => {

try {

const { title, topic } = req.body;

const fileName = req.file.filename;

await WordSchema.create({ title, topic, word: fileName }); // Assuming 'word' field in the schema for Word file

res.send({ status: "ok" });

} catch (error) {

console.error("File upload error:", error);

res.status(500).json({ status: "error", error: error.message });

}

});

app.get("/get-files", async (req, res) => {

try {

const data = await WordSchema.find({});

res.send({ status: "ok", data });

} catch (error) {

console.error("Error fetching files:", error);

res.status(500).json({ status: "error", error: error.message });

}

});

app.get("/", (req, res) => {

res.send("success!!");

});

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

console.log(`Server started on port ${PORT}`);

});

**wordDetails.js:**

const mongoose = require("mongoose");

const WordDetailsSchema = new mongoose.Schema(

{ word: String,topic: String, title: String,

},

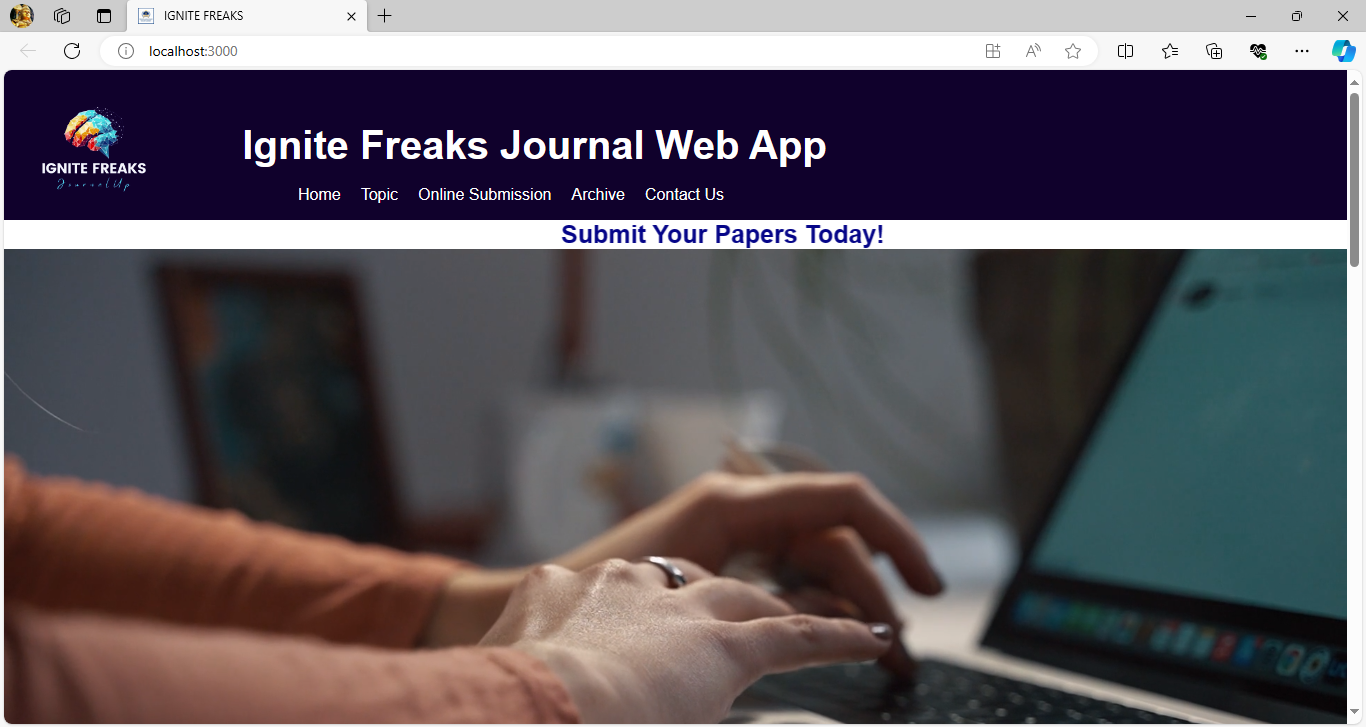
{ collection: "WordDetails" }

);module.exports = mongoose.model("WordDetails", WordDetailsSchema)

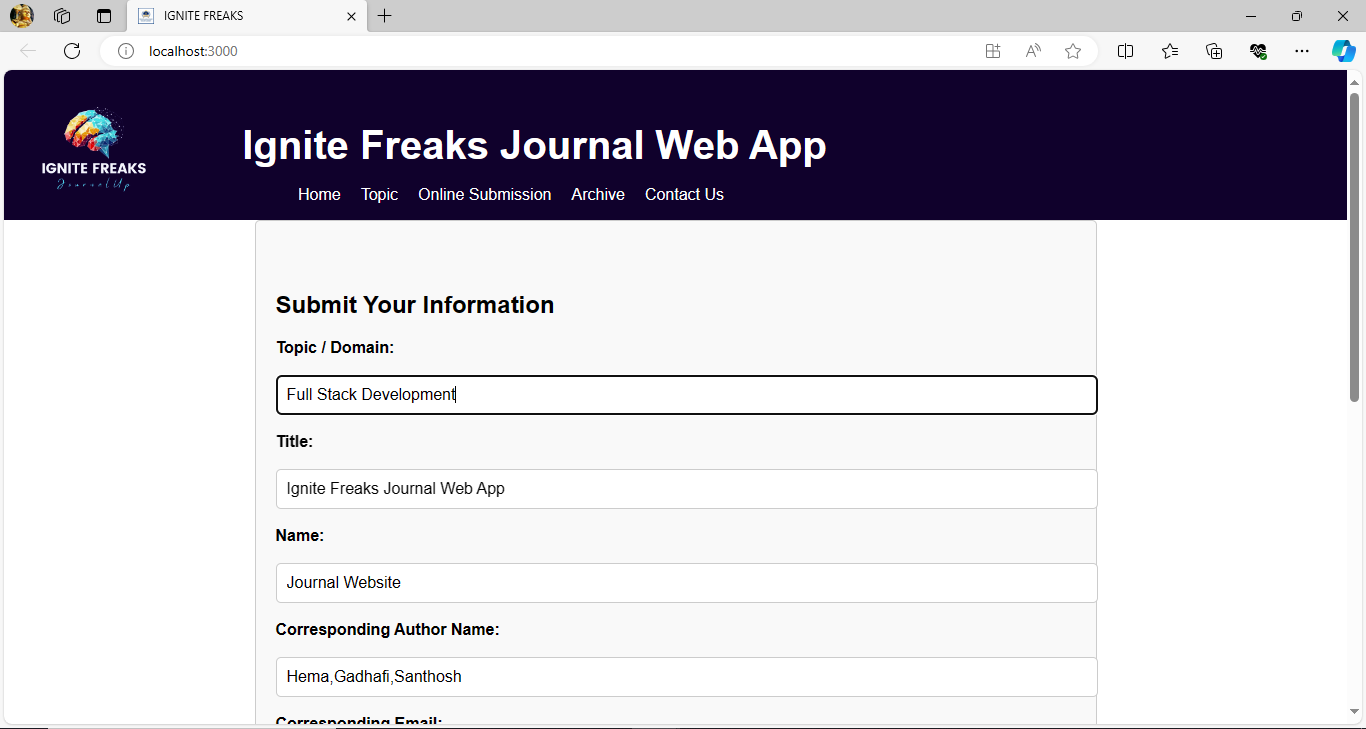
\

**CHAPTER – 7**

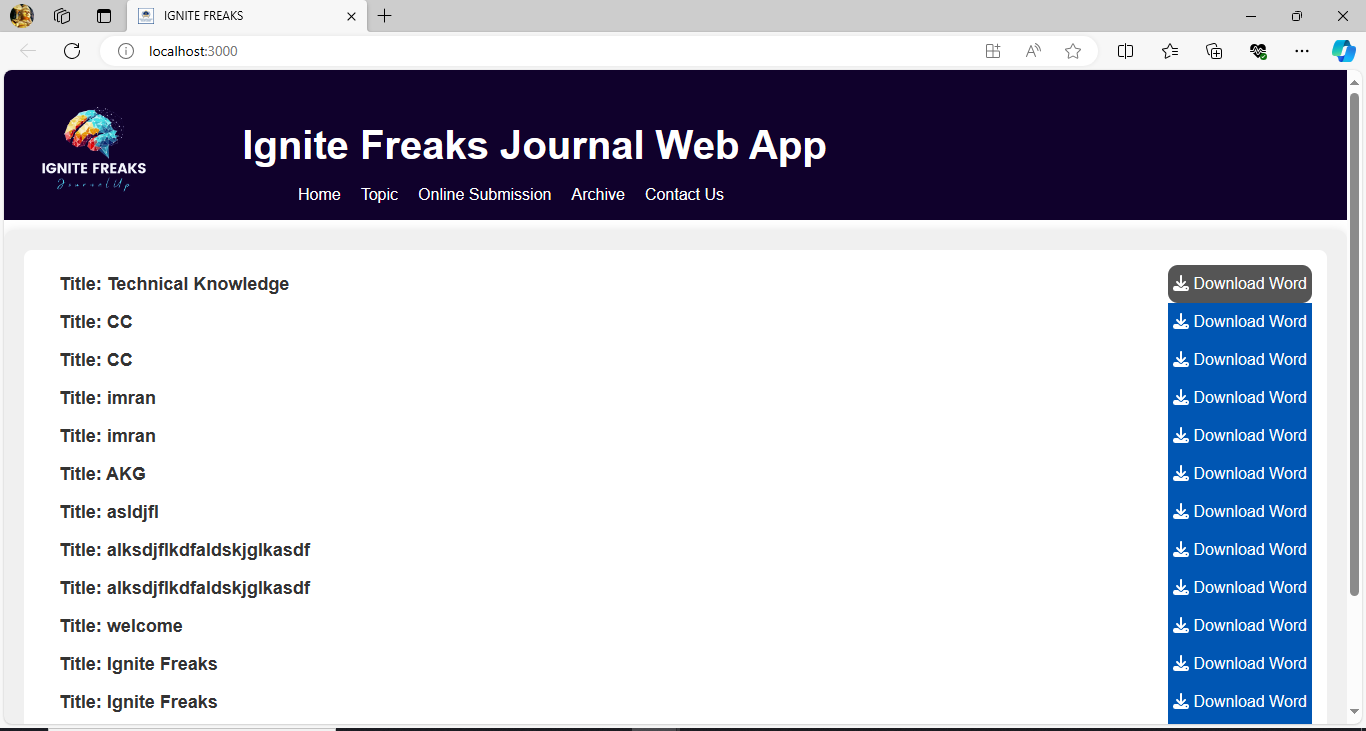
**OUTPUT SCREEN**

­

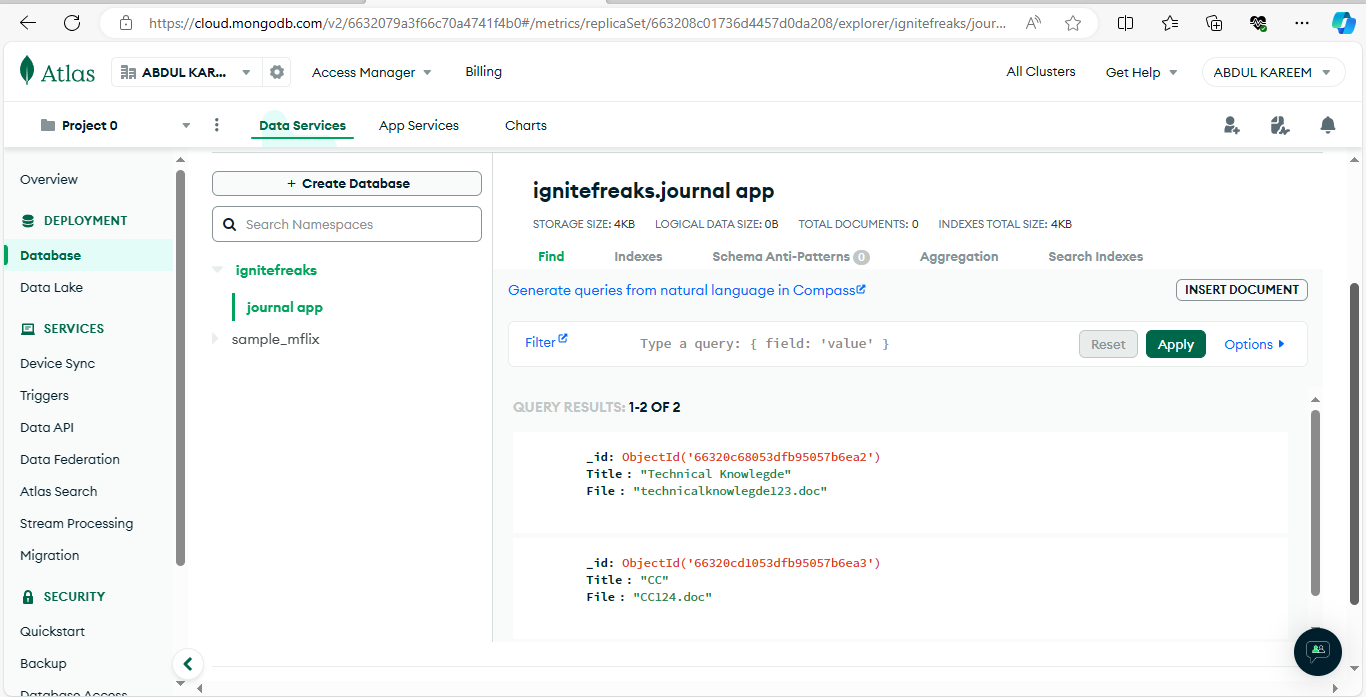
**Figure 7.1** **Home Page**



**Figure7.2 Online Submission**



**Figure 7.3 Archives Page**



**Figure 7.4 Database Collections**

**CHAPTER – 8**

**CONCLUSIONS**

The "Ignite Freaks Journal Web App" project has successfully created a dynamic and user-friendly platform that serves as a comprehensive resource for academic and research communities. By integrating various functionalities such as topic exploration, online submission, archival access, and editorial insights, this web application stands out as a pivotal tool in the dissemination of knowledge and information.

Throughout the development of the "Ignite Freaks Journal Web App", the project team adhered to modern web development practices to ensure a responsive and accessible user interface. The use of React, combined with effective state management and routing, provided a robust foundation for building a scalable and maintainable web application. This ensures that users have a seamless experience navigating through the different sections of the journal, whether they are submitting research papers or browsing through past publications.

**8.1 FUTURE SCOPE**

The future scope for the "Ignite Freaks Journal Web App" is vast and promising, offering numerous avenues for enhancement and expansion. In the forthcoming phases, the introduction of artificial intelligence and machine learning algorithms could revolutionize the way users interact with the platform, enabling personalized article recommendations and intelligent search functionalities that adapt to user preferences and reading habits. Moreover, expanding the scope of the platform to include multimedia content such as podcasts and video interviews with authors and researchers can greatly enrich the user experience. Another significant area for development is the integration of collaborative tools, such as forums and discussion boards, which would facilitate academic discussions and networking among users. Additionally, implementing advanced analytics to track user engagement and feedback can provide valuable insights for continuous improvement of the platform. There is also potential to enhance mobile accessibility and develop dedicated apps to increase the platform’s reach and usability. Overall, the future enhancements aim not only to enhance the functionality and user experience of the "Ignite Freaks Journal Web App" but also to ensure it remains at the forefront of digital academic publishing technology.

**REFERENCES**

1. Zawacki-Richter O, Marín VI, Bond M, Gouverneur F. Systematic review of research on artifcial intelligence applications in higher education–where are the educators? Int J Educ Technol High Educ. 2019;16:39.

2. Lemay DJ, Baek C, Doleck T. Comparison of learning analytics and educational data mining: A topic modeling approach. Comput Edu: Artif Intell. 2021;2:100016.

3. Jacob J, Jha K, Kotak P, Puthran S. Educational data mining techniques and their applications. In 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), 2015;1344–1348.

4. Shabbir J, Anwer T. Artifcial intelligence and its role in near future. arXiv preprint arXiv:1804.01396 (2018).

5. Ras Z. Advances in intelligent information systems. NY: Springer; 2018.

6. Wiederhold G. The roles of artifcial intelligence in information systems. J Intell Info Syst. 1992;1:35–55.