A

Theme Based Project Report

on

NOTES DEPOT
SIMPLIFIED LEARNING RESOURCE

Submitted for partial fulfilment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

By

Beeram Vasanth Kumar Reddy (2451-22-733-311)
Mohd Zunaid Khan(2451-22-733-316)
J. Jayasurya (2451-22-733-314)

Under the guidance of

M. Madhuri
Asst. Prof
Department of CSE

MATURI VENKATA SUBBA RAO(MVSR) ENGINEERING COLLEGE
(An Autonomous Institution)
Department of Computer Science and Engineering
(Affiliated to Osmania University & Recognized by AICTE)
Nadergul, Balapur Mandal, Hyderabad – 501 510
Academic Year: 2022-2023

MATURI VENKATA SUBBA RAO(MVSR) ENGINEERING COLLEGE
(An Autonomous Institution)
Department of Computer Science and Engineering
(Affiliated to Osmania University & Recognized by AICTE)
Nadergul, Balapur Mandal, Hyderabad – 501 510

CERTIFICATE

*This is to certify that the Theme Based project work entitled "**NOTES DEPOT SIMPLIFIED LEARNING RESOURCE**" is a bonofide work carried out by **Beeram Vasanth Kumar Reddy (2451-22-733-311), MOHD ZUNAID KHAN (2451-22-733-316), J.JAYASURYA (2451-22-733-314)** in partial fulfilment of the requirements for the award of degree of **Bachelor of Engineering** in **Computer Science and Engineering** from **Maturi Venkata Subba Rao(MVSR) Engineering College,** affiliated to OSMANIA UNIVERSITY, Hyderabad, during the Academic Year 2022-2023. under our guidance and supervision.*

The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide	**Project Coordinators**	**Head of the Department**
M. Madhuri	Meduri Anupama	Prof J Prasanna Kumar
Assistant Professor	Associate Professor	Professor & Head
Department of CSE	M.V.R. Jyothi Sree	Department of CSE
MVSREC.	Assistant Professor	MVSREC.
	I Navakanth	
	Assistant professor	

External Examiner

DECLARATION

This is to certify that the work reported in the present Theme Based project entitled **"NOTES DEPOT SIMPLIFIED LEARNING RESOURCE"** is a record of bonafide work done by us in the Department of Computer Science and Engineering, M.V.S.R. Engineering College, Osmania University. The reports are based on the work done entirely by us and not copied from any other source.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

Beeram Vasanth Kumar Reddy	Mohd. Zunaid Khan	J. Jayasurya
(2451-22-733-311)	(2451-22-733-316)	(2451-22-733-314)

ACKNOWLEDGEMENT

We would like to express our sincere gratitude and indebtedness to our project guide Ms. **M. Madhuri, Assistant Professor** for his/her valuable suggestions and interest throughout the course of this project.

We are also thankful to our principal **Dr. G. Kanaka Durga** and **Prof. J Prasanna Kumar,** Professor and Head, Department of Computer Science and Engineering, MVSR Engineering College, Hyderabad for providing excellent infrastructure and a nice atmosphere for completing this project successfully as a part of our B.E. Degree (CSE).

We convey our heartfelt thanks to the lab staff for allowing us to use the required equipment whenever needed.

Finally, we would like to take this opportunity to thank our family for their support through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

Beeram Vasanth Kumar Reddy (2451-22-733-311)

Mohd. Zunaid Khan (2451-22-733-316)

J. Jayasurya (2451-22-733-314)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VISION

To impart technical education of the highest standards, producing competent and confident engineers with an ability to use computer science knowledge to solve societal problems.

MISSION

To make the learning process exciting, stimulating and interesting.
To impart adequate fundamental knowledge and soft skills to students.
To expose students to advanced computer technologies in order to excel in engineering practices by bringing out the creativity in students.
To develop economically feasible and socially acceptable software.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

The Bachelor's program in Computer Science and Engineering is aimed at preparing graduates who will:-

PEO-1: Achieve recognition through demonstration of technical competence for successful execution of software projects to meet customer business objectives.

PEO-2: Practice life-long learning by pursuing professional certifications, higher education or research in the emerging areas of information processing and intelligent systems at a global level.

PEO-3: Contribute to society by understanding the impact of computing using a multidisciplinary and ethical approach.

Program Specific Outcomes(PSOs)

PSO1: Demonstrate competence to build effective solutions for computational real-world problems using software and hardware across multi-disciplinary domains.

PSO2: Adapt to current computing trends for meeting the industrial and societal needs through a holistic professional development leading to pioneering careers or entrepreneurship.

CERTIFICATION

Statement of Achievement

JavaScript Essentials 1 (JSE)

The graduate of the *JavaScript Essentials 1 (JSE)* course, administered by the undersigned instructor, and provided by **Cisco Networking Academy®** in collaboration with **OpenEDG JavaScript Institute**:

- knows the syntax of the core JavaScript language to a degree that allows them to work with variables, operators, control flow, and functions;
- knows the basics of the JavaScript data types system, distinguishing between primitive and complex types, and is able to choose a type adequate to their needs;
- thinks algorithmically and can analyze a problem using a programmatic conceptual apparatus;
- knows how a program is interpreted and executed in an actual computer environment, and can design, develop, and improve very simple JavaScript programs;
- can interpret and handle basic exceptions related to errors in program execution;
- understands a programmer's work in the software development process, and the role of fundamental development tools.

BEERAM VASANTH KUMAR REDDY

Student

MVSR Engineering College

Academy Name

India	**19 Jan 2024**
Location	Date

Laura Quintana
Laura Quintana
VP & General Manager, Cisco Networking Academy

www.netacad.com | www.javascriptinstitute.com

JavaScript Institute OpenEDG JavaScript Institute Authorized Academy Program OPENEDG

Statement of Achievement

JavaScript Essentials 1 (JSE)

The graduate of the *JavaScript Essentials 1 (JSE)* course, administered by the undersigned instructor, and provided by **Cisco Networking Academy®** in collaboration with **OpenEDG JavaScript Institute**:

- knows the syntax of the core JavaScript language to a degree that allows them to work with variables, operators, control flow, and functions;
- knows the basics of the JavaScript data types system, distinguishing between primitive and complex types, and is able to choose a type adequate to their needs;
- thinks algorithmically and can analyze a problem using a programmatic conceptual apparatus;
- knows how a program is interpreted and executed in an actual computer environment, and can design, develop, and improve very simple JavaScript programs;
- can interpret and handle basic exceptions related to errors in program execution;
- understands a programmer's work in the software development process, and the role of fundamental development tools.

MOHD ZUNAID KHAN

Student

MVSR Engineering College

Academy Name

India	**19 Jan 2024**
Location	Date

KANAJAM MURALI KRISHNA

Instructor

Instructor Signature

www.netacad.com | www.javascriptinstitute.com

JavaScript Institute OpenEDG JavaScript Institute Authorized Academy Program OPENEDG

Statement of Achievement

JavaScript Essentials 1 (JSE)

The graduate of the *JavaScript Essentials 1 (JSE)* course, administered by the undersigned instructor, and provided by **Cisco Networking Academy®** in collaboration with **OpenEDG JavaScript Institute**:

- knows the syntax of the core JavaScript language to a degree that allows them to work with variables, operators, control flow, and functions;
- knows the basics of the JavaScript data types system, distinguishing between primitive and complex types, and is able to choose a type adequate to their needs;
- thinks algorithmically and can analyze a problem using a programmatic conceptual apparatus;
- knows how a program is interpreted and executed in an actual computer environment, and can design, develop, and improve very simple JavaScript programs;
- can interpret and handle basic exceptions related to errors in program execution;
- understands a programmer's work in the software development process, and the role of fundamental development tools.

J JAYASURYA

Student

MVSR Engineering College

Academy Name

India	**2 Feb 2024**
Location	Date

VIKRAM NARAYANDAS

Instructor	Instructor Signature

www.netacad.com | www.javascriptinstitute.com

ABSTRACT

"Notes Depot" stands as a dedicated educational platform crafted mainly for engineering students, aiming to simplify access to crucial learning materials and tutorial resources. This web application seamlessly integrates a robust backend built with Node.js and Express, utilizing MySQL for efficient data management. Through Notes Depot, users effortlessly access and download subject-specific notes sourced from a dedicated database. Moreover, the platform offers a user-friendly interface facilitating access to tutorial videos directly fetched from YouTube. Notes Depot makes BTech learning easy by providing organized notes and informative videos in one place. And in this BUDDY a chatbot was implemented by integrating an OpenAI API key, with using this platform every student (users) can get each required resource at one place.

Beeram Vasanth Kumar Reddy (2451-22-733-311)
Mohd. Zunaid Khan (2451-22-733-316)
J. Jayasurya (2451-22-733-314)

TABLE OF CONTENTS

CONTENTS **PAGENO.s**

LIST OF FIGURES

1.INTRODUCTION

1.1 PROBLEM STATEMENT AND SCOPE

Most of the students mainly engineering students face a problem with lack of resources like Notes because most of us will neglect at initial stage and during exams we all will be in rush and Start browsing for notes tutorial classes. The lack of a centralized platform for accessing educational resources poses a significant challenge for learners who struggle to find high-quality and comprehensive study materials. Existing platforms may offer limited content or subscription for notes so "Notes Depot" stands as a dedicated educational platform crafted mainly for engineering students, aiming to simplify access to crucial learning materials and tutorial resources. This web application seamlessly integrates a robust backend built with Node.js and Express, utilizing MySQL for efficient data management. Through Notes Depot, users effortlessly access and download subject-specific notes sourced from a dedicated database. Moreover, the platform offers a user-friendly interface facilitating access to tutorial videos directly fetched from YouTube. Notes Depot makes BTech learning easy by providing organized notes and informative videos in one place. And in this BUDDY a chatbot was implemented by integrating an OpenAI API key, with using this platform every student (users) can get each required resource at one place.

The scope of the project encompasses three primary objectives: Firstly, to offer a comprehensive repository of educational notes spanning various subjects, ensuring users have access to relevant study materials tailored to their specific academic needs. Secondly, the platform will provide a curated collection of tutorial videos, accessible through user searches, offering supplementary learning resources to enhance understanding and mastery of key concepts. Additionally, the project aims to incorporate a user-friendly "Buddy" feature, allowing users to seek assistance and clarification on academic queries, fostering a supportive learning environment. Together, these elements constitute a multifaceted educational platform designed to empower users with resources and support to excel in their academic endeavors.

1.2 APPLICATION AREAS

1. **Academic Institutions:** Notes Depot can be used by colleges, and universities to provide a centralized platform for students to access study materials, lecture notes

2. **Remote Learning and Distance Education:** In remote learning environments or distance education programs, Notes Depot can provide students with access to course materials, lectures, and resources, enabling remote participation and engagement in the learning process.

3. **Online Learning Platforms:** Online learning platforms can integrate the notes depot project to provide additional study materials and resources to their users. It can enhance the overall learning experience by offering a diverse range of educational content.

4. **Self-Study and Exam Preparation:** Individuals pursuing self-study or preparing for competitive exams can benefit from the project by accessing comprehensive notes and tutorial videos on various subjects. It serves as a convenient resource for review and revision purposes.

1.3 USERS

The target audience for the Notes Depot platform includes students, educators, self-learners, and anyone seeking access to educational resources to support their learning journey. The platform caters to users across various academic levels, disciplines, and areas of interest. The application users are mainly engineering students and it is also designed mainly for them.

2. SYSTEM REQUIREMENTS

2.1 TOOLS AND TECHNOLOGIES

Technologies

H1.HTML CSS JS to create a dynamic page.

2.NODE JS: (Node) is an Open Source, cross-platform runtime environment for executing JavaScript code.

3.XAMPP MYSQL: storing of data.

4.API (application program interface):

1.Youtube API

2.OpenAI API

Tools

1.Vs code editor : Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control.

2.2 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements

Server: A web server to host the project's backend application and database.

Storage: Sufficient storage space to store notes and other media files.

Network Infrastructure: Stable internet connectivity to ensure uninterrupted access to the platform.

Software Requirements

Operating System: windows 10 ,11

Web Server: Apache

2.3 IO SPECIFICATION

Inputs

- User Requests: Users interact with the system by making requests through the user interface (UI), such as selecting a subject to view notes or searching for tutorial videos.
- Search Queries: Users input search queries to find specific notes or tutorial videos.
- Form Submissions: Users submit forms to perform actions like fetching notes or initiating a search.
- Click Events: Users click on links, buttons, or other interactive elements to navigate through the application or trigger specific actions.

Outputs

- Notes Display: The system outputs notes based on the user's selection of subject, year, semester, etc.
- Tutorial Videos: The system displays relevant tutorial videos based on the user's search query.
- Error Messages: If an error occurs during the process, the system outputs error messages to inform users about the issue.
- Success Messages: After successful operations, such as fetching notes or submitting forms, the system outputs success messages to confirm the action.
- Navigation Changes: The system updates the UI to reflect changes in navigation, such as moving to a different page or section within the application.

3.SYSTEM ARCHITECTURE

3.1 FLOWCHART

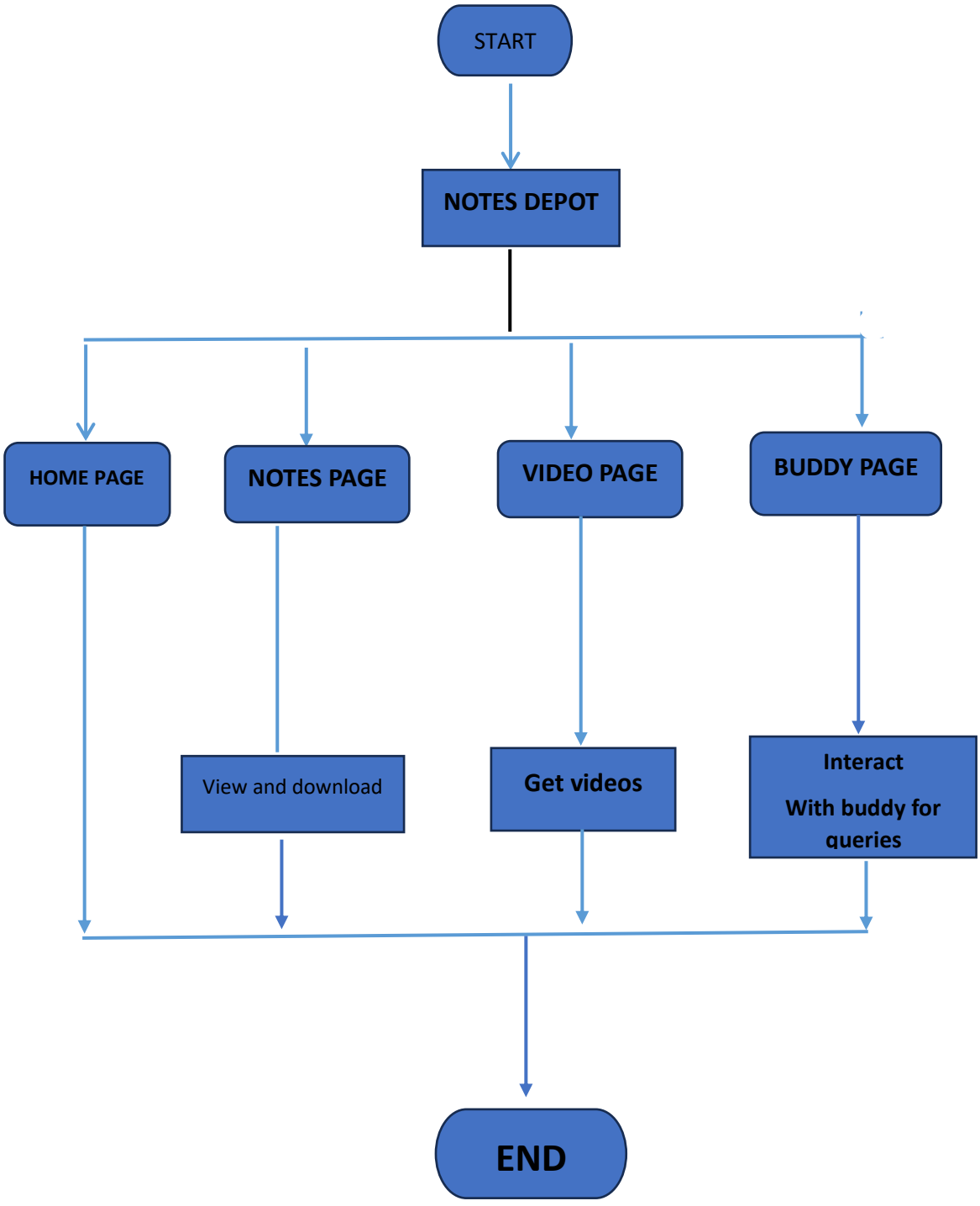

Fig.3.1.1 flowchart of notes depot

3.2 DATA FLOW DIAGRAM

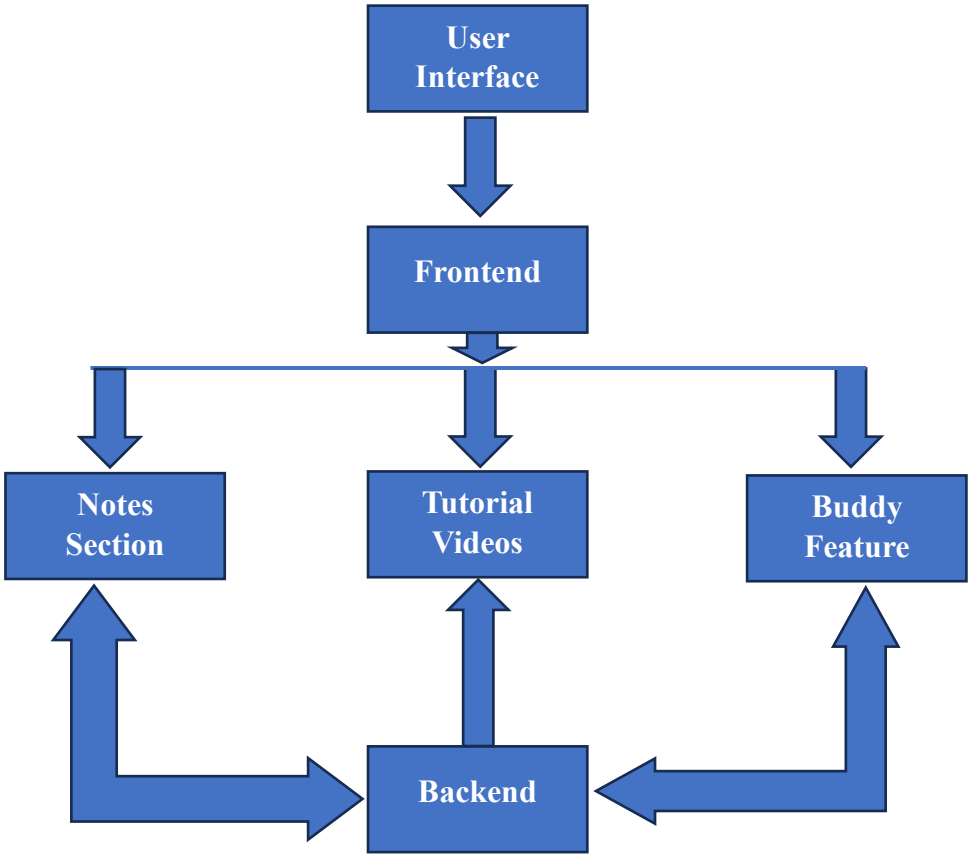

Fig.3.2.1 Dataflow diagram

Explanation:

•**User Interface:** Represents the interface through which users interact with the application.

•**Frontend:** Handles the presentation layer, including the home page, notes section, tutorial videos, and chatbot interface.

•**Backend:** Manages the server-side logic and interacts with external APIs and the database.

•**Notes Section:** Handles user requests related to accessing and fetching notes based on selected criteria such as year, semester, and subject.

•**Tutorial Videos**: Manages user requests to search for and retrieve tutorial videos using the YouTube Data API.

•**Buddy Feature:** Deals with user queries and interactions with the chatbot feature powered by the OpenAI

4.IMPLEMENTATION

4.1 ENVIRONMENTAL SETUP

The environmental setup for the Notes Depot project involves configuring the development environment with the necessary tools, technologies, and dependencies. Below is an outline of the environmental setup:

Software Requirements:

1.**Node.js:** Install Node.js, a JavaScript runtime, to execute server-side code.

a.DownloadNode.js

2.**npm (Node Package Manager):** Comes with Node.js, used to install and manage project dependencies.

a. npm Documentation

3.**Text Editor or IDE:** Choose a text editor or integrated development environment for writing code.

a. Visual Studio Code

4.**MySQL Database:** Set up a MySQL database to store and retrieve notes data.

A .Download XAMPP

Project Dependencies:

1.Install Express.js for the backend

npm install express

2.Install the OpenAI Node.js library for chatbot functionality

npm install openai

3.Install the MySQL Node.js library for database interaction

npm install mysql

4. Install CORS middleware for handling Cross-Origin Resource Sharing

npm install cors

API integration: Integration of YouTube API and OpenAI API set up required credentials and endpoints to API

Project Structure:

- notes-depot/
  - frontend/
    - index.html
    - notes.html
    - videos.html
    - about.html
    - styles.css
    - script.js
    - script1.js
  - server.mjs
  - database.js
  - package.json

4.2 MODULES AND THEIR DESCRIPTION

1.HTML/CSS/JavaScript Module:

Description: This module constitutes the frontend of the web application, responsible for creating an interactive and user-friendly interface. It combines HTML for structure, CSS for styling, and JavaScript for dynamic behavior, enabling users to access notes and tutorial videos seamlessly.

Components:

5. HTML (Hypertext Markup Language):

- Description: HTML is used to structure the content of the web pages, defining the layout and elements that users interact with. It includes tags for headings, paragraphs, lists, forms, and other elements essential for building the user interface.
- Role: HTML provides the backbone of the frontend, organizing the content and creating the structure of each page. It ensures proper semantics and accessibility, allowing users to navigate and interact with the application efficiently.

2. CSS (Cascading Style Sheets):

- Description: CSS is employed to style the HTML elements, enhancing the visual presentation and user experience. It encompasses properties and values for controlling aspects like colors, fonts, layouts, and animations.
- Role: CSS plays a vital role in defining the appearance and aesthetics of the frontend interface. It ensures consistency across pages, improves readability, and enhances the overall design, making the application visually appealing and engaging for users.

3. JavaScript:

- Description: JavaScript adds interactivity and dynamic behavior to the web pages, enabling real-time updates, form validation, DOM manipulation, and event handling. It allows for the creation of interactive features and enhances user engagement.
- Role: JavaScript acts as the behavior layer of the frontend, facilitating client-side interactions and functionality. It handles user input, responds to events, fetches data from the server asynchronously, and updates the interface dynamically, providing a rich and responsive user experience.

By integrating HTML, CSS, and JavaScript effectively, this module ensures the creation of a compelling and intuitive frontend interface for the web application. It focuses on usability, accessibility, and aesthetics, thereby enhancing user satisfaction and engagement.

2. Middleware Integration:

- **Description:** Middleware integration is a crucial aspect of the Express server module. Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application's request-response cycle. They can perform tasks such as parsing request bodies, handling CORS, serving static files, and logging requests.
- **Components:**
- **Body Parser:** Parses incoming request bodies in different formats like JSON, URL-encoded, etc. This middleware is essential for extracting data from incoming requests, enabling the server to process and respond to client requests effectively.
- **CORS (Cross-Origin Resource Sharing):** Allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served. CORS middleware handles CORS-related headers to ensure secure communication between the client and server, especially in cases where the client and server are hosted on different domains.
- **Static File Serving:** Serves static files such as HTML, CSS, and JavaScript files for the frontend interface. By configuring static file serving middleware, the Express server can serve these files directly to clients, improving performance and reducing latency.

These middleware components enhance the functionality and security of the Express server, ensuring smooth request processing and response generation for various client requests.

3.MySQL Database Module:

Description: The MySQL Database Module is responsible for managing the connection to the MySQL database and handling data operations related to notes. It serves as the backend storage mechanism, storing and retrieving information about notes, such as year, semester, subject, and associated content.

Components:

5. Database Connection:

- Description: Establishes a connection to the MySQL database using credentials such as host, username, password, and database name. It ensures secure and reliable communication between the backend server and the database.
- Functionality: Handles the initialization of the database connection when the server starts and manages the connection throughout the application's lifecycle. It provides error handling mechanisms to deal with connection failures and ensures optimal performance by reusing connections where possible.

2. CRUD Operations:

- Description: Defines functions to perform CRUD (Create, Read, Update, Delete) operations on the notes stored in the database. These operations enable the manipulation of note data, including adding new notes, retrieving existing notes, updating note details, and deleting obsolete notes.
- Functionality: Implements SQL queries to execute CRUD operations efficiently. It abstracts the complexity of database interactions, encapsulating them into functions that can be called by other modules or routes within the application. This ensures modularity, maintainability, and reusability of database-related code.

3. Data Schema:

- Description: Defines the structure of the database tables used to store note-related information. It specifies the fields (columns) and their data types, constraints, and relationships, ensuring consistency and integrity of the data.
- Functionality: Creates and maintains the database schema, including tables for storing notes, metadata, and any other relevant information. It ensures proper indexing and normalization to optimize query performance and minimize data redundancy.

4. Error Handling:

• Description: Implements mechanisms to handle errors and exceptions that may occur during database operations. It includes error detection, logging, and appropriate error responses to ensure robustness and reliability of the application.

• Functionality: Catches and handles database-related errors gracefully, providing informative error messages to users and developers. It logs errors for troubleshooting and debugging purposes, aiding in identifying and resolving issues quickly.

By encapsulating database-related functionality into this module, the application achieves efficient data management and persistence, enabling users to store, retrieve, and manipulate notes seamlessly. It ensures data integrity, security, and reliability, contributing to the overall functionality and performance of the application.

4.YouTube Integration Module:

Description: The YouTube Integration Module facilitates interactions with the YouTube Data API to retrieve tutorial videos based on user requests. It serves as a bridge between the application and the YouTube platform, enabling seamless integration of video content into the web application.

Functionality:

5. API Authentication:

• Manages authentication with the YouTube Data API using API keys or OAuth tokens. It ensures secure access to YouTube resources and compliance with API usage policies.

2.Video Search:

• Implements functions to search for tutorial videos based on user-defined criteria such as keywords, topics, or categories. It constructs API requests with appropriate parameters to retrieve relevant video content.

3.Video Retrieval:

• Retrieves video metadata including titles, descriptions, thumbnails, and URLs from the YouTube API response. It parses the API response to extract relevant information and prepares it for display to users.

4.Error Handling:

• Handles errors and exceptions that may occur during API interactions, such as rate limiting, quota exceeded, or network errors. It provides graceful error handling mechanisms to notify users and developers of any issues encountered.

5.Caching Mechanism:

• Implements a caching mechanism to store previously fetched video data locally, reducing the number of API calls and improving performance. It checks the cache before making new API requests and updates the cache periodically to ensure freshness of data.

By integrating with the YouTube Data API, this module enriches the web application with a vast library of tutorial videos, enhancing the learning experience for users. It provides seamless access to relevant video content directly within the application, eliminating the need for users to visit external platforms.

5.Chatbot Integration Module:

Description: The Chatbot Integration Module integrates a simple chatbot functionality into the web application, enabling users to ask questions and fetch relevant information using the OpenAI API. This module enhances user engagement and provides a conversational interface for interacting with the application.

Functionality:

1.OpenAI Integration:

• Establishes communication with the OpenAI API to leverage its natural language processing capabilities. It sends user queries to the API and processes the responses to generate relevant and contextual replies.

2.User Interaction:

• Implements a user-friendly chat interface where users can type questions or queries. It captures user input and sends it to the backend server for processing.

3.Query Processing:

- Sends user queries to the OpenAI API for natural language understanding and processing. It formulates requests in the required format and handles API responses to extract meaningful information.

4.Response Generation:

- Parses the responses from the OpenAI API to generate appropriate replies to user queries. It formats the responses for display in the chat interface, ensuring clarity and relevance.

5. Error Handling:

- Manages errors and exceptions that may occur during chatbot interactions, such as API request failures or invalid user input. It provides error messages and prompts to guide users in case of issues.

By integrating a chatbot into the web application, this module enhances user engagement and provides an interactive way for users to obtain information. Users can ask questions in natural language and receive helpful responses, making the application more user-friendly and accessible.

5.TESTING AND RESULTS

5. Unit Testing:

- Express Server Module: Unit tests are conducted to ensure that each route and middleware function in the Express server module behaves as expected. Mock data and stubs are used to simulate requests and responses for different scenarios.
- MySQL Database Module: Unit tests verify the functionality of CRUD operations and error handling in the MySQL database module. Mock database connections and queries are used to test various database interactions.
- YouTube Integration Module: Unit tests validate the integration with the YouTube Data API, ensuring that the module correctly retrieves and processes tutorial videos based on user requests.
- Chatbot Integration Module: Unit tests assess the chatbot's ability to understand user queries and generate relevant responses using the OpenAI API. Mock API responses and user inputs are used to simulate chat interactions.

2. Integration Testing:

- Integration tests evaluate the interaction between different modules of the project. This includes testing the communication between the frontend (HTML/CSS/JavaScript) and backend (Express server), as well as the integration of external APIs (YouTube Data API, OpenAI API).
- End-to-end testing is performed to validate the complete workflow of the application, from user interaction to data retrieval and display. Test scenarios cover common user actions such as searching for notes, watching tutorial videos, and interacting with the chatbot.

5.1. RESULTS

•The testing process ensures that the project meets its functional requirements and performs reliably under various conditions.
•Unit tests confirm the correctness of individual components, while integration tests verify the seamless interaction between different modules.
•The project is validated against acceptance criteria to ensure that it fulfils user expectations and delivers the intended features and functionalities.
•Test reports and logs document the testing process and results, providing insights into any identified issues or areas for improvement.
•Upon successful testing, the project is deemed ready for deployment, ensuring a high-quality and robust web application for users to access notes, tutorial videos, and chatbot assistance.

5.2 SCREENSHOTS

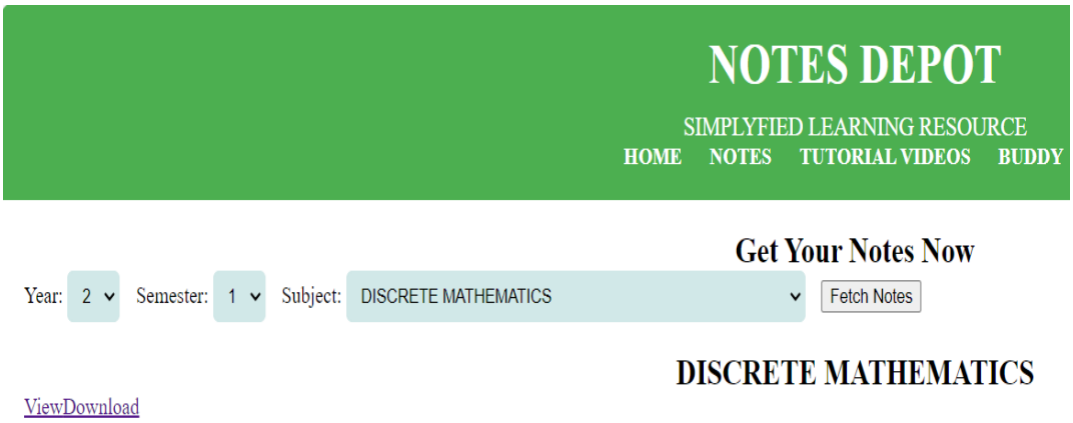

Fig 5.2.1 screenshot of notes page

In this page when user select their required year ,semester, subject and click on fetch notes, it will displays the notes with view and download option

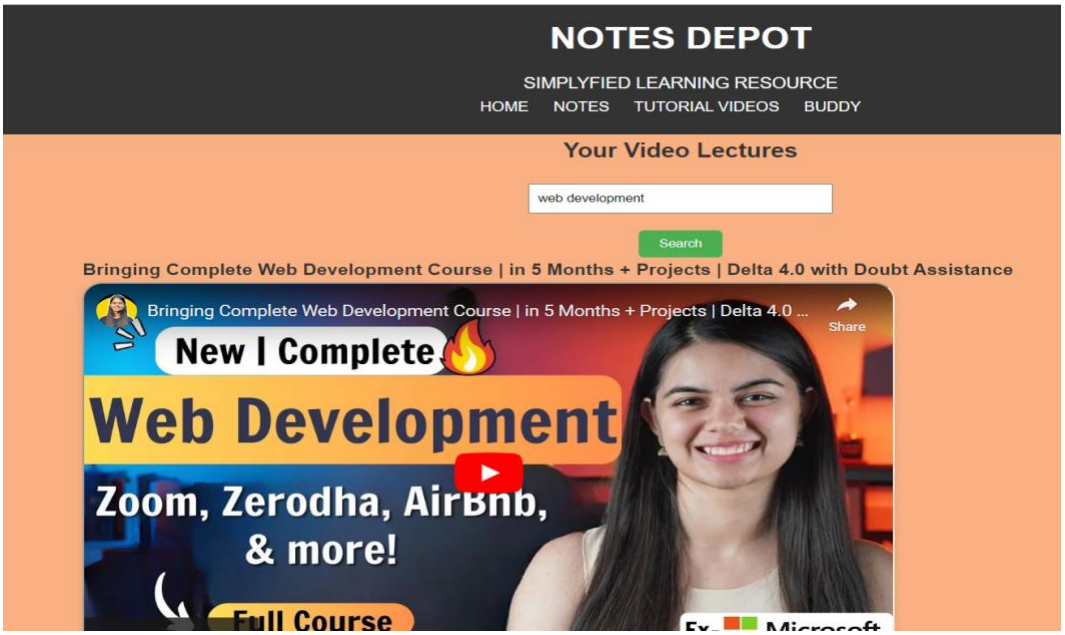

Fig 5.2.1 screenshot of videos page

In this page user can get their required tutorial video by searching using search box it will gives the results as above.

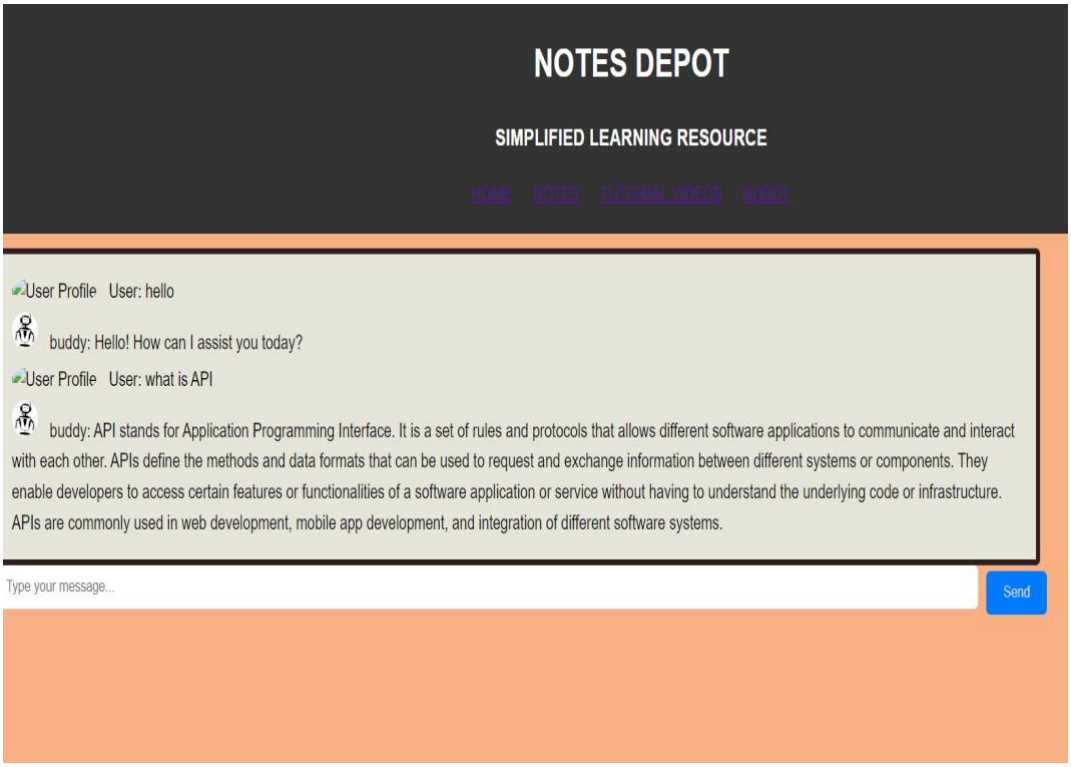

Fig 5.2.3 screenshot of buddy page

In this user can interact with buddy and solve their questions and the reply from buddy will be based on users question as show as above.

6.CONCLUSION AND FUTURE ENHANCEMENTS

6.1 CONCLUSION

Notes Depot project has been developed with the primary goal of providing users with a simplified learning resource that encompasses notes, tutorial videos, and an interactive chatbot. The utilization of modern web development technologies such as Node.js, HTML, CSS, and JavaScript ensures a responsive and engaging user interface.

The HTML/CSS/JavaScript module plays a crucial role in the project by implementing the frontend, allowing users to seamlessly interact with the platform. The intuitive design enhances user experience, making it easy to navigate and access educational content.

The MySQL Database Module manages the storage and retrieval of notes, ensuring efficient CRUD operations. This database integration enables users to access educational materials based on their preferences, such as year, semester, and subject.

The YouTube Integration Module facilitates the retrieval of tutorial videos using the YouTube Data API. Users can search for specific topics, and the system fetches relevant videos, enriching the learning experience with multimedia content.

The Chatbot Integration Module adds an interactive dimension to the project, allowing users to pose questions and receive information from the OpenAI API. This feature enhances user engagement and provides a personalized learning experience.

Throughout the development process, rigorous testing has been conducted to ensure the reliability and performance of the platform. Iterative testing and improvements have been applied to address any identified issues and enhance overall functionality.

In summary, the Notes Depot project stands as a versatile educational platform, catering to a wide range of users. It leverages technology to create an inclusive learning environment, making educational resources easily accessible and promoting continuous learning. The project has undergone thorough testing, and its dynamic features contribute to a positive and interactive educational experience.

6.2 FUTURE ENHANCEMENTS

1.Content management system(for admins)

2.Progressive Web App (PWA): Convert the application into a Progressive Web App (PWA) to enable offline access, push notifications, and other native-like features on mobile devices.

3.Advanced Chatbot Functionality: Enhance the chatbot integration by incorporating natural language processing (NLP) capabilities to provide more accurate and context-aware responses. Implement features such as sentiment analysis to understand user queries better and tailor responses accordingly.

4.Interactive Note-Taking: Introduce interactive note-taking functionality, allowing users to highlight key points, add annotations, and create personal study guides directly within the platform. These notes can be saved to user profiles for future reference.

5.Community Engagement Features: Foster a sense of community among users by incorporating social features such as discussion forums, group study sessions, or peer-to-peer tutoring platforms. Encourage collaboration and knowledge sharing among users to enrich the learning experience.

6.Content Expansion: Continuously expand the repository of notes and tutorial videos to cover a broader range of subjects and topics. Collaborate with educational institutions, subject matter experts, and content creators to curate high-quality educational resources across various disciplines.

7.Performance Optimization: Continuously optimize the platform's performance to ensure fast loading times, smooth navigation, and scalability as user traffic grows. Implement caching mechanisms, code minification, and server-side optimizations to enhance overall responsiveness and reliability.

By incorporating these future enhancements, Notes Depot can evolve into a comprehensive educational platform that empowers users to engage in immersive and personalized learning experiences across diverse subjects and disciplines.

APPENDEX

SAMPLE SOURCE CODE:

Server.js

```
import express from 'express';
import dotenv from 'dotenv';
import bodyParser from 'body-parser';
import cors from 'cors';
import { OpenAI } from 'openai';
import * as db from './database.js';
import path from 'path';
dotenv.config();
const openai = new OpenAI({
   key: process.env.OPENAI_API_KEY,
   engine: "text-davinci-003",
});
const app = express();
const PORT = process.env.PORT || 5000;
app.use(cors());
app.use(express.json());
app.use(express.static('frontend'));
app.use(bodyParser.json());
app.use(express.static(path.join(new URL('.', import.meta.url).pathname, 'public')));
app.get('/', (req, res) => {
   res.sendFile(__dirname + '/about.html');
});



app.post('/', async (req, res) => {
   try {
      const userInput = req.body.userInput;

      const response = await openai.chat.completions.create({
         model: "gpt-3.5-turbo",
```

```
            messages: [{ role: "user", content: userInput }],
            temperature: 0,
            max_tokens: 2048,
            top_p: 1,
            frequency_penalty: 0.5,
            presence_penalty: 0,
        });

        res.status(200).send({
            bot: response.choices[0].message.content
        });
    } catch (err) {
        console.log(err);
        res.status(500).send({ err });
    }
});
// API endpoint to fetch distinct years
app.get('/api/years', async (req, res) => {
    try {
        const years = await db.getDistinctYears();
        res.json(years);
    } catch (error) {
        console.error('Error fetching years:', error);
        res.status(500).json({ error: 'Internal server error' });
    }
});

// API endpoint to fetch semesters based on the selected year
app.post('/api/semesters', async (req, res) => {
    const { year } = req.body;
    try {
        const semesters = await db.getDistinctSemesters(year);
        res.json(semesters);
```

```
    } catch (error) {
        console.error('Error fetching semesters:', error);
        res.status(500).json({ error: 'Internal server error' });
    }
});

// API endpoint to fetch subjects based on the selected year and semester
app.post('/api/subjects', async (req, res) => {
    const { year, semester } = req.body;
    try {
        const subjects = await db.getDistinctSubjects(year, semester);
        res.json(subjects);
    } catch (error) {
        console.error('Error fetching subjects:', error);
        res.status(500).json({ error: 'Internal server error' });
    }
});

// API endpoint to fetch and display notes based on the selected year, semester, and subject
app.post('/api/notes', async (req, res) => {
    const { year, semester, subject } = req.body;
    try {
        const notes = await db.getNotes(year, semester, subject);
        res.json(notes);
    } catch (error) {
        console.error('Error fetching notes:', error);
        res.status(500).json({ error: 'Internal server error' });
    }
});
```

```
// API endpoint to serve PDF files
app.get('/api/pdf/:subject', async (req, res) => {
    const { subject } = req.params;
    try {
        const pdfPath = await db.getPdfPathBySubject(subject);

        // Construct an absolute path to the PDF file
        const absolutePath = path.resolve(new URL('../public',
import.meta.url).pathname, pdfPath);
  // Send the file with the correct absolute path
        res.sendFile(absolutePath);
    } catch (error) {
        console.error('Error serving PDF file:', error);
        res.status(500).json({ error: 'Internal server error' });
    }
});
app.listen(5000, () => console.log('Server is running on http://localhost:5000'));
```

Database.js

```
const mysql = require('mysql');
const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: '', // Enter your MySQL password here
    database: 'notesdepot'
});
connection.connect((error) => {
    if (error) {
        console.error('Error connecting to database:', error);
        return;
    }
    console.log('Connected to database');
});
```

```
function query(sql, params) {
    return new Promise((resolve, reject) => {
        connection.query(sql, params, (error, results) => {
            if (error) {
                reject(error);
                return;
            }
            resolve(results);
        });
    });
}
async function getDistinctYears() {
    const sql = 'SELECT DISTINCT year FROM notes';
    const rows = await query(sql);
    return rows.map(row => row.year);
}
async function getDistinctSemesters(year) {
    const sql = 'SELECT DISTINCT semester FROM notes WHERE year = ?';
    const rows = await query(sql, [year]);
    return rows.map(row => row.semester);
}
async function getDistinctSubjects(year, semester) {
    const sql = 'SELECT DISTINCT subject FROM notes WHERE year = ? AND semester = ?';
    const rows = await query(sql, [year, semester]);
    return rows.map(row => row.subject);
}
async function getNotes(year, semester, subject) {
    const sql = 'SELECT * FROM notes WHERE year = ? AND semester = ? AND subject = ?';
    const rows = await query(sql, [year, semester, subject]);
    return rows;
}
```

```
async function getPdfPathBySubject(subject) {
    const sql = 'SELECT pdf_path FROM notes WHERE subject = ?';
    const rows = await query(sql, [subject]);
    if (rows.length > 0) {
        return rows[0].pdf_path;
    }
    throw new Error('PDF not found');
}

module.exports = {
    getDistinctYears,
    getDistinctSemesters,
    getDistinctSubjects,
    getNotes,
    getPdfPathBySubject
};
```

Script.js

```
document.addEventListener('DOMContentLoaded', () => {
  const yearSelect = document.getElementById('year');
  const semesterSelect = document.getElementById('semester');
  const subjectSelect = document.getElementById('subject');
  const notesContainer = document.getElementById('notes-container');
  const noteForm = document.getElementById('noteForm');
// Function to fetch distinct years from the server and populate the year select dropdown
  const fetchYears = async () => {
      try {
          const response = await fetch('/api/years');
          const years = await response.json();
          yearSelect.innerHTML = ''; // Clear previous options
          years.forEach(year => {
             const option = document.createElement('option');
              option.value = year;
              option.textContent = year;
```

```
            yearSelect.appendChild(option);
        });
    } catch (error) {
        console.error('Error fetching years:', error);
    }
};
// Function to fetch semesters based on the selected year and populate the semester select dropdown
const fetchSemesters = async () => {
    const selectedYear = yearSelect.value;
    try {
        const response = await fetch('/api/semesters', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ year: selectedYear })
        });
        const semesters = await response.json();
        semesterSelect.innerHTML = ''; // Clear previous options
        semesters.forEach(semester => {
            const option = document.createElement('option');
            option.value = semester;
            option.textContent = semester;
            semesterSelect.appendChild(option);
        });
    } catch (error) {
        console.error('Error fetching semesters:', error);
    }
};

// Function to fetch subjects based on the selected year and semester and populate the subject select dropdown
```

```
const fetchSubjects = async () => {
    const selectedYear = yearSelect.value;
    const selectedSemester = semesterSelect.value;
    try {
        const response = await fetch('/api/subjects', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ year: selectedYear, semester: selectedSemester })
        });
        const subjects = await response.json();
        subjectSelect.innerHTML = ''; // Clear previous options
        subjects.forEach(subject => {
            const option = document.createElement('option');
            option.value = subject;
            option.textContent = subject;
            subjectSelect.appendChild(option);
        });
    } catch (error) {
        console.error('Error fetching subjects:', error);
    }
};

// Function to fetch and display notes based on the selected year, semester, and subject
const fetchNotes = async () => {
    const selectedYear = yearSelect.value;
    const selectedSemester = semesterSelect.value;
    const selectedSubject = subjectSelect.value;
    try {
        const response = await fetch(`/api/notes`, {
            method: 'POST',
            headers: {
```

```
            'Content-Type': 'application/json'
          },
          body: JSON.stringify({ year: selectedYear, semester: selectedSemester, subject: selectedSubject })
        });
        const notes = await response.json();
        notesContainer.innerHTML = ''; // Clear previous notes
        notes.forEach(note => {
          const noteContainer = document.createElement('div');

          // Display subject name
          const subjectName = document.createElement('h2');
          subjectName.textContent = note.subject;
          noteContainer.appendChild(subjectName);

          // View button
          const viewButton = document.createElement('a');
          viewButton.href = `/api/pdf/${encodeURIComponent(note.subject)}`;
          viewButton.textContent = 'View';
          viewButton.target = '_blank'; // Open in a new tab
          noteContainer.appendChild(viewButton);

          // Download button
          const downloadButton = document.createElement('a');
          downloadButton.href = `/api/pdf/${encodeURIComponent(note.subject)}`;
          downloadButton.download = 'note.pdf';
          downloadButton.textContent = 'Download';
          noteContainer.appendChild(downloadButton);

          notesContainer.appendChild(noteContainer);
        });
      } catch (error) {
        console.error('Error fetching notes:', error);
```

```
    }
  };

  // Event listeners for select dropdowns
  yearSelect.addEventListener('change', fetchSemesters);
  semesterSelect.addEventListener('change', fetchSubjects);

  // Event listener for form submission
  noteForm.addEventListener('submit', (event) => {
    event.preventDefault();
    fetchNotes();
  });

  // Fetch years when the page loads
  fetchYears();
});
```