Software Design and Engineering

Lab Document

High Level Purpose Statement:	The goal of this lab was to explore a new web framework—in this case, Node.js with Express—to create a deployable project. Instead of working with a complex Java/Maven system with numerous dependencies and security/CORS challenges, I built a simple, visually appealing web application. This project, "Smart Schedule Tool," demonstrates how to create a full-stack application with a minimal REST API and a polished, responsive frontend using EJS templates, Bootstrap, custom CSS animations, and dynamic visual features. The project is designed for rapid prototyping and educational purposes.
Experimental Design:	**Project Setup** • I created a Node Express project in IntelliJ IDEA. • The project is organized with a clear separation between server code (Express) and frontend assets (EJS templates, static CSS). • I used Lowdb for persistent data storage, enabling basic CRUD functionality. • I integrated express-ejs-layouts for consistent layouts (navigation bar and footer) across all pages. **Frontend Development** • I used EJS for templating and Bootstrap for styling. • The homepage now includes: o A striking hero section with a dynamic, rotating background that features a gradient overlay and 3D animations. o Animated text and 3D effect buttons that navigate to meaningful pages. • Additional sections include: o A **Features** area with three interactive feature cards: "Easy Management," "Real-Time Updates," and "Customizable." o A **Testimonials** section showcasing user feedback. • Each feature page is fully functional: o **Easy Management:** Displays a filterable table of courses. o **Real-Time Updates:** Polls the API every 5 seconds to simulate live data. o **Customizable:** Allows the user to change the app's background color dynamically. • Custom CSS animations and vibrant color schemes were applied to create a captivating user experience. **Backend Development**

	• I built a simple REST API endpoint (/api/courses) that supports full CRUD operations on course data stored persistently using Lowdb. • The Express server serves both the API and the static frontend, eliminating CORS issues. • I implemented routes for: o Viewing all courses. o Adding a course via a form. o Editing a course via a form. o Feature pages for "Easy Management," "Real-Time Updates," and "Customizable." • The backend uses Nanoid to generate unique IDs for courses. **Deployment & Tools** • I used Node.js and npm for dependency management and running the project. • IntelliJ IDEA was my primary development environment. • The project can be started with simple commands (npm install and npm start), and it is fully runnable locally. • The entire app is designed to be deployed on platforms such as Heroku, Netlify, or AWS with minimal configuration.
Resources Available:	**Documentation:** Official documentation for Node.js, Express, EJS, Bootstrap, npm, and Lowdb. **Development Tools:** IntelliJ IDEA (with integrated terminal), Git, and GitHub. **Online Tutorials:** Various YouTube videos and articles on Node Express and modern web design. **Previous Lab Experiences:** Lessons learned from a complex Java/Maven project guided my decision to use a simpler stack.
Time Estimate:	I spent about 10 hours on this project: • **2-3 hours** setting up the Node Express project in IntelliJ. • **3-4 hours** developing and refining the frontend with EJS, Bootstrap, and custom CSS animations. • **2-3 hours** implementing the REST API, persistent storage with Lowdb, and integrating CRUD operations. • Additional time for debugging, testing, and refining the user experience.
Experiment Notes:	**Initial Challenges:** My previous experience with a complex Java/Maven project involved numerous dependencies and security/CORS issues, which slowed down development and made rapid prototyping difficult. These challenges led me to explore a simpler framework. **Transition to Node Express:** I chose Node Express for its lightweight nature and ease of use. Setting up the project in IntelliJ IDEA was straightforward, and using

	EJS templates with Bootstrap allowed me to quickly build a visually appealing interface. This shift enabled me to focus more on design and functionality rather than extensive configuration. **Persistent Data and CRUD:** By integrating Lowdb, I added persistent storage for course data. This allowed me to implement full CRUD operations (Create, Read, Update, Delete) for managing courses. I also used Nanoid to generate unique IDs for each course, which improved data management and testing. **Frontend Enhancements:** I significantly improved the user interface by: • Creating a dynamic hero section with a rotating background and 3D animations. • Adding interactive buttons that lead to meaningful pages (View Courses, Add Course, Edit Course, and feature detail pages). • Designing feature pages with vibrant backgrounds and engaging interactive elements. For example, the "Easy Management" page now displays a filterable table of courses; the "Real-Time Updates" page simulates live updates via periodic API polling; and the "Customizable" page allows users to change the background theme. • Applying custom CSS transitions and animations to create smooth 3D effects, enhancing the overall visual appeal.
Results:	**Functional Application:** The final application runs on http://localhost:3000. The homepage features a dynamic hero section, vibrant animated buttons, and multiple sections for features and testimonials. **REST API:** The API endpoint at /api/courses supports full CRUD operations, allowing course data to be persistently stored and manipulated. **Fully Functional Feature Pages:** • **Easy Management:** Displays a filterable table of courses. • **Real-Time Updates:** Polls the API for live data updates. • **Customizable:** Allows users to change the background color theme. **Visual Appeal:** The app features dynamic 3D effects, vibrant gradients, and smooth animations that create a captivating user experience. **Ease of Deployment:** The project can be started with simple npm commands, making it ideal for educational purposes and rapid prototyping.
Consequences for the Future:	**Scalability:** This project serves as a solid foundation for future enhancements such as integrating a real database, implementing full user authentication, and adding real-time features using WebSockets.

	Learning and Experimentation: The simplicity and modularity of Node Express, combined with EJS and Bootstrap, provide a flexible base for further exploration in full-stack web development. **Deployment:** With its minimal setup and straightforward codebase, I can easily deploy this application to cloud platforms for real-world use.