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-based system that involved Maven and numerous dependencies, 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 beautiful, responsive static frontend using EJS templates, Bootstrap, and custom CSS. The project avoids complex CORS and security issues, making it ideal for rapid prototyping and educational purposes.
Experimental Design:	Project Setup: • I created a Node Express project in IntelliJ IDEA. • The project is structured with a clear separation between server code (Express) and frontend assets (EJS templates and static CSS). Frontend Development: • I used EJS for templating and Bootstrap for styling. • The homepage includes a striking hero section with a background image, overlay, animated text, and multiple buttons. • Additional sections include feature cards and a testimonials section—all designed with custom CSS animations and vibrant colors. Backend Development: • I built a simple REST API endpoint (/api/courses) that returns sample course data. • The Express server serves both the API and the static frontend, eliminating CORS issues. • Express-ejs-layouts was integrated to manage a common layout for all pages, including a navigation bar and footer. Deployment & Tools: • Node.js and npm were used for dependency management and running the project. • IntelliJ IDEA served as the development environment. • The entire project is easily runnable via simple commands (npm install and npm start).
Resources Available:	Documentation: Node.js, Express, EJS, Bootstrap, and npm official documentation. Development Tools: IntelliJ IDEA, integrated terminal, Git, and GitHub. Online Tutorials: Various YouTube videos and online articles on Node Express and modern web design.

	Previous Lab Experiences: Lessons learned from a more complex Java/Maven project guided the decision to switch frameworks for simplicity.
Time Estimate:	I estimated spending about 10 hours on this project: • 2-3 hours setting up the Node Express project in IntelliJ. • 3-4 hours developing the frontend with EJS and Bootstrap, including design enhancements. • 2-3 hours on building the REST API and integrating it with the frontend. • Additional time for debugging and testing the deployment.
Experiment Notes:	Initial Challenges: The earlier project involved Maven, Java, and various security and CORS issues, which proved complex and time-consuming. Transition to Node Express: Switching to Node Express greatly simplified the development process. The lightweight framework allowed me to focus on creating a visually appealing and functional interface without getting bogged down by security configurations. Frontend Enhancements: The use of Bootstrap and custom CSS animations significantly improved the aesthetics of the application. Adding multiple buttons that navigate to meaningful routes (view courses, add course, edit course, and feature details) provided an interactive user experience. Lessons Learned: Simplicity in design can reduce configuration overhead. Node Express, combined with EJS and Bootstrap, offers a fast and efficient way to build and deploy a web application.
Results:	Functional Application: The final application runs on http://localhost:3000. The backend API is accessible at /api/courses, and the frontend displays a modern, vibrant homepage with a hero section, features, testimonials, and navigation buttons. Visual Appeal: The project now boasts a dynamic gradient background, animated sections, and attractive call-to-action buttons, making it visually appealing. Ease of Deployment: The project can be run with simple commands, and the Express server handles both API and static content, avoiding CORS complications.
Consequences for the Future:	Scalability: This project serves as a foundation for future enhancements, such

	as integrating a real database, implementing user authentication, or expanding the REST API. Learning and Experimentation: The simplicity of Node Express allows for rapid prototyping and iterative design. Future labs could explore adding more advanced features, such as real-time updates using Socket.IO or deploying to cloud platforms. Deployment: With its minimal setup and straightforward codebase, this project can be easily deployed on platforms like Heroku, Netlify, or AWS for real-world applications.