**System Design**

This section oulines the architecture, technologies, and tools used in the development of the demonstration website. It highlights how each layer of the system will interact to deliver a fully functional, responsive, and modern web application.

**Frontend Design**

**🛠️hnologies Used:**

* HTML5 – Structuring the content on the web pages.
* CSS3 – Styling and layout, including animations, transitions, and responsiveness.
* JavaScript (ES6+) – Client-side scripting to add interactivity and dynamic behaviors.
* React.js – A component-based JavaScript library for building fast, interactive UIs.

🔧 **Features:**

* Component-based architecture: UI is split into reusable React components (e.g., Navbar, Footer, LoginForm, ContactForm).
* Routing: Implemented using react-router-dom for seamless navigation across different pages without page reloads.
* State Management:
  + Local state managed via React’s useState and useEffect.
  + Optional: Context API or Redux for global state if the app scales.
* Form Handling:
  + Forms (like login, registration, and contact) are validated on the client-side.
  + React Hook Form or Formik can be used for better form management.
* UI Responsiveness:
* CSS Flexbox and Grid for layouts.
* Media queries for adapting to screen sizes.
* Mobile-first approach for styling.

**Backend Design**

**Technologies Used:**

* Node.js – JavaScript runtime for building the server-side logic.
* Express.js – Lightweight Node.js web application framework used to create RESTful APIs.

**Key Functionalities:**

* API Endpoints for:
  + User registration (POST /api/register)
  + User login (POST /api/login)
  + Contact form submission (POST /api/contact)
* Authentication:
  + Token-based authentication using JWT (JSON Web Tokens).
  + Secure password storage using bcrypt.
* Validation and Error Handling:
  + Input validation with middleware like express-validator.
  + Centralized error handling for cleaner code and easier debugging.
* CORS Handling: Enables communication between frontend and backend if hosted on different domains.

️ **Database Layer**

️ **Database: MongoDB**

* NoSQL database suitable for flexible document storage.
* Schema design using Mongoose (ODM for MongoDB).
* Collections (examples):
  + users – Stores user data like email, hashed password, name.
  + contacts – Stores data from the contact form submissions.

**Data Flow:**

* User submits data via frontend form.
* Frontend sends request to backend API.
* Backend processes data, validates, and stores it in MongoDB.
* Server returns response to frontend.

️ **UI/UX Design**

️ **Tool Used: Figma**

* Wireframes:
  + Created to visualize the layout and design of key pages before implementation.
  + Includes pages such as: Home, Login, Register, Contact, and Dashboard.
  + Emphasis on clean, minimal design with intuitive navigation.

**Design Principles:**

* User-centric design: Easy-to-navigate interface with accessibility in mind.
* Consistency: Common UI elements across all pages (e.g., buttons, colors, typography).
* Feedback Mechanisms: Visual cues for actions like form submission success/failure.

**System Flow Overview**

* User → React Frontend → API Request → Node/Express Server → MongoDB
* ↑ ↓
* UI Updates ← JSON Response ← Processed Request