Project Design Phase
Solution Architecture

Date	15 April 2025
Team ID	SWTID1743870329
Project Name	Personal Expense Tracker
Maximum Marks	2 Marks

Solution Architecture:

- The solution architecture consists of multiple layers working together to provide a seamless, secure, and scalable user experience. The app follows a modular, RESTful, and component-based architecture with a clear separation of concerns.

1. Client-Side (Frontend) – React.js

- **Responsibilities:**
    - Provides a responsive and interactive user interface
    - Handles routing with React Router
    - Sends API requests to the backend via Axios/Fetch
    - Displays visual insights using libraries like Chart.js or Recharts
- **Components:**
    - Dashboard (summary view)
    - Expense form (add/edit)
    - Expense list (filtered by date/category)
    - Budget manager
    - Report generator
    - Authentication (login/register forms)

- **State Management:**
    - React Hooks (useState, useEffect)
    - Optionally, Redux or Context API for global state (e.g., user session)

🔹 2. Server-Side (Backend) – Node.js + Express.js

- **Responsibilities:**
    - Handles all business logic
    - Processes API requests and routes them accordingly
    - Validates input data
    - Authenticates and authorizes users (JWT)
    - Interfaces with MongoDB for CRUD operations
- **API Endpoints:**
    - `/api/auth/register` – User registration
    - `/api/auth/login` – User login
    - `/api/expenses` – Add/view/edit/delete expenses
    - `/api/budgets` – Set and retrieve budget limits
    - `/api/reports` – Generate summaries and insights
    - `/api/users` – Profile management

🔹 3. Database – MongoDB (with Mongoose)

- **Responsibilities:**

    - Stores all persistent data (users, expenses, budgets, etc.)
    - Offers schema flexibility for evolving needs
- **Collections:**
    - `users`: userId, name, email, password (hashed), settings
    - `expenses`: expenseId, userId, category, amount, date, vendor, payment method
    - `budgets`: budgetId, userId, category, limit, time range
    - `reports`: generated summaries or saved snapshots (optional)

🔹 4. Authentication & Security

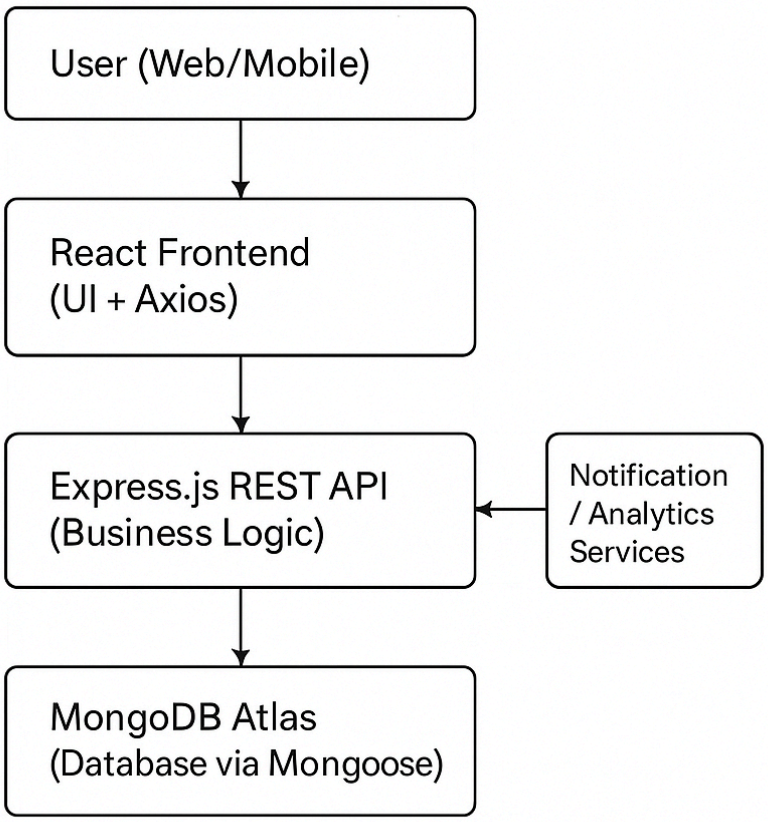
- **JWT (JSON Web Tokens):** Used for stateless user authentication
- **bcrypt.js:** Password hashing and storage
- **Helmet.js:** Sets secure HTTP headers
- **CORS Middleware:** Controls cross-origin access
- **Role-Based Access Control:** Optional – Admin/Business user handling

🔹 5. Deployment Architecture

- **Frontend Deployment:**
    - **Vercel / Netlify**
- **Backend Deployment:**
    - **Render / Railway / Heroku / AWS EC2**
- **Database Hosting:**

    - **MongoDB Atlas** (Cloud-based)
- **CI/CD (optional):**
    - GitHub Actions or GitLab CI for continuous integration & deployment

Logical flow :


User (Web/Mobile)
React Frontend (UI + Axios)
Express.js REST API (Business Logic)
Notification / Analytics Services
MongoDB Atlas (Database via Mongoose)