COLLEGE CODE: 9504

COLLEGE NAME: DR.G.U.POPE COLLEGE OF ENGINEERING

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

STUDENT NM ID: 17EB415F3BD034A830348260A3C0FB92

ROLL NO: 950423104033

DATE: 15-09-2025

Completed the project SINGLE PAGE APPLICATION Phase-02

TECHNOLOGY PROJECT NAME: IBM-FE- SINGLE PAGE APPLICATION

SUBMITTED BY,

NAME : RENISTON R

MOBILE: 8883793681

Phase 2 — Solution Design & Architecture

Tech Stack Selection:

To build a scalable and responsive Single-Page Application (SPA), the following technology stack has been selected:

Frontend:

React.js – Component-based UI library for building dynamic interfaces.

React Router – Enables smooth client-side routing.

Tailwind CSS – Utility-first CSS framework for styling and responsiveness.

Backend:

Node.js with Express.js – Handles server-side logic and API endpoints.

Database:

MongoDB (NoSQL) – Stores structured and semi-structured data efficiently.

Other Tools:

Axios – For sending and receiving HTTP requests.

JWT (JSON Web Token) – For authentication and session management.

GitHub – Version control and collaboration.

This stack is chosen because it supports fast development, modularity, scalability, and real-time updates suitable for SPA.

UI Structure / API Schema Design:

UI Structure:

Header (Navbar) - Contains navigation links and branding.

Main Content Area - Displays views dynamically (e.g., Dashboard, Profile, Tasks).

Sidebar (Optional) - Provides shortcuts or filters.

Footer - General info and quick links.

API Schema:

User API:

POST/api/users/register - Register a new user.

POST/api/users/login Login user with authentication.

GET /api/users/: id - Retrieve user details.

Task API (example module for SPA):

GET /api/tasks - Fetch tasks.

POST/api/tasks - Add task.

PUT/api/tasks/:id Update task.

DELETE/api/tasks/:id-Delete task.

Data Handling Approach:

Frontend Data Handling:

State managed using React Hooks (useState, useEffect).

Global data handled with Context API or Redux if needed.

API integration with Axios for CRUD operations.

Backend Data Handling:

Input validation with Express middleware.

Database communication using Mongoose.

Secure authentication using JWT & bcrypt.

Persistence:

Data stored in MongoDB.

User session maintained using JWT in browser localStorage.

Component / Module Diagram:

Frontend Components:

App.js Main app and router setup.

Navbar.js-Navigation bar.

Footer.js - Footer section.

Dashboard.js Main content/dashboard.

TaskList.js Task display.

TaskForm.js Add/edit tasks.

Auth.js Handles login & register.

Backend Modules:

server.js Entry point for Node/Express.

routes/ - User and task APIs.

MongoDB schemas (User, Task). models/

controllers/API logic.

middlewares/ - Auth, validation.

Basic Flow Diagram: