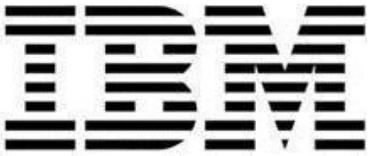

COLLEGE CODE: 9504

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

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

STUDENT NM ID: 17EB415F3BD034A830348260A3C0FB92

ROLL NO: 950423104033

DATE: 29-09-2025

Completed the project SINGLE PAGE APPLICATION Phase-04

TECHNOLOGY PROJECT NAME: IBM-FE- SINGLE PAGE APPLICATION

SUBMITTED BY,

NAME : RENISTON R

MOBILE: 8883793681

Phase 4 — Enhancements & Deployment

1 Additional Features

◈ Task 1: Role-Based Access

- Define roles: Admin, User, Guest.
- Restrict features based on roles (e.g., only Admin can delete).
- Store role info in JWT/session and check before rendering UI.

Tools: JWT, Firebase Auth, Passport.js (Node), Supabase.

◈ Task 2: Search, Filters, Sorting

- Add a **search bar** for quick lookups.
- Implement filters (date range, category, tags).
- Enable sorting (A–Z, latest–oldest).

Tools: Backend query params (?search=, ?sort=, ?filter=), frontend debounce for search.

◈ Task 3: Data Export/Import

- Export data into **CSV, PDF, Excel**.
- Allow importing CSV/Excel to update data.

Tools:

- **CSV/Excel**: papaparse, xlsx library.
- **PDF**: jspdf, pdfmake.

◈ Task 4: Notifications

- Email notifications for key events.
- In-app alerts (snackbars, modals, toast messages).
- Optional push notifications (with service workers).

Tools: Nodemailer, Firebase Cloud Messaging, Toastify.

◆ Task 5: Offline Mode (Optional)

- Cache recent data for offline use.
- Show “Offline mode” banner.

Tools: Service Workers, Workbox, IndexedDB.

2 UI/UX Improvements

◆ Task 1: Navigation & Layout

- Add sidebar or top nav bar.
- Use breadcrumbs for easy navigation.
- Mobile-first responsive design.

Tools: React Router, TailwindCSS, Bootstrap.

◆ Task 2: Theming

- Implement **dark/light mode** toggle.
- Store theme preference in localStorage.

Tools: Tailwind Dark Mode, Styled Components.

◆ Task 3: Accessibility

- Use **ARIA labels** for screen readers.
- Ensure proper color contrast (WCAG standards).
- Enable **keyboard navigation** (tab focus).

Tools: Axe DevTools, Lighthouse audits.

◆ Task 4: Animations/Micro-Interactions

- Smooth page transitions.
- Hover effects and button feedback.
- Loading spinners and skeleton screens.

Tools: Framer Motion, Lottie Animations.

3 API Enhancements

◆ Task 1: Pagination & Query Params

- Implement limit, page, sort, filter params.
- Return metadata (totalItems, pages, next, prev).

◆ Task 2: Security Features

- Add **rate limiting** to prevent abuse.
- Enable **CORS** properly.
- Use HTTPS-only cookies or JWT refresh tokens.

Tools: Express Rate Limit, Helmet.js, CORS middleware.

◆ Task 3: Error Handling

- Consistent error response format:

Json Format:

```
{
  "status": "error",
  "code": 400,
  "message": "Invalid input"
}
```

◆ Task 4: API Versioning

- Use /api/v1/ → base route.
- Prepare for future upgrades (/api/v2/).

4 Performance & Security Checks

◈ Task 1: Frontend Optimization

- Enable **lazy loading** for components.
- Use **code splitting** (React.lazy, dynamic import).
- Compress images (WebP, ImageOptim).

◈ Task 2: Backend Optimization

- Add database indexes.
- Use caching (Redis).
- Optimize queries (avoid N+1 queries).

◈ Task 3: Security Hardening

- Sanitize inputs to prevent **XSS & SQL injection**.
- Add CSRF protection (CSRF tokens).
- Use **HTTPS & SSL**.

Tools: OWASP ZAP, Burp Suite (for penetration testing).

◈ Task 4: Monitoring

- Setup error tracking.
- Collect performance metrics.

Tools: Sentry, LogRocket, Google Analytics.

5 Testing of Enhancements

◆ Unit Testing

- Test each function/component separately.
- Example: test search filtering logic.

Tools: Jest, Mocha, React Testing Library.

◆ Integration Testing

- Ensure API + frontend work correctly together.
- Example: fetch user data → render table.

Tools: Cypress, Playwright.

◆ Performance Testing

- Run load tests to simulate 1k–10k users.
- Identify bottlenecks.

Tools: Apache JMeter, k6.

◆ Security Testing

- Run vulnerability scans.
- Check for XSS, CSRF, SQL injection.

Tools: OWASP ZAP, Nmap.

◆ User Acceptance Testing (UAT)

- Share with real users/testers.
- Collect feedback and iterate.

6 Deployment

◆ Step 1: Choose Platform

- **Netlify** → Great for frontend SPAs.
- **Vercel** → Best for Next.js/React apps.
- **AWS/GCP/Azure** → Full control for production apps.

◆ Step 2: CI/CD Setup

- Automate deployment on every **git push**.
- Example: GitHub Actions → build & deploy.

◆ Step 3: Configure Environment

- Store API keys in environment variables.
- Example: .env.local for dev, secrets in platform dashboard.

◆ Step 4: Post-Deployment

- Monitor uptime (Pingdom, UptimeRobot).
- Collect crash/error reports.
- Run performance audits (Lighthouse, GTmetrix).