COLLEGECODE : 6109

COLLEGENAME : JAYALAKSHMI INSTITUTE OF TECHNOLOGY

DEPARTMENT : COMPUTER SCIENCE AND ENGINEERING

STUDENT NM- ID :849dbfcce321f459f0148603bb827f00

ROLL NO : 610923104317

DATE : 05-10-2025

Completed the project name as

Phase_4_TECHNOLOGY PROJECT

NAME : IBM-NJ EVENT SCHEDULER APP

SUBMITTED BY,

NAME : Mahendiran M

MOBILE NO :+91 9080640249

Phase4 —IBM-NJ EVENT SCHEDULER APP

1. Additional Features

In this stage, new features are added based on user feedback, project goals, or feature backlog. Some examples may include:

- Recurring Events: Option to schedule weekly/monthly events.
- Notifications/Reminders: Email or in-app notifications for upcoming events.
- Event Categories: Allow users to filter or tag events (e.g., personal, work, education).
- Dark Mode: Enhancing accessibility and user preferences.
- Purpose: Improve functionality and usability of the app for a broader range of use cases.

2. UI/UX Improvements

User Interface and User Experience are polished to make the app more intuitive, engaging, and accessible.

- Responsive Design: Ensuring the app looks and works well across devices (mobile, tablet, desktop).
- Consistent Styling: Use of a clean design system or UI framework (e.g., Tailwind, Material UI).
- User-Friendly Forms: Simplifying the event creation and editing process.

- Animations & Transitions: Smooth visual effects to enhance interactivity.
- Purpose: Boost user satisfaction, retention, and accessibility.

3. API Enhancements

Enhance backend or third-party API integration to support additional functionality and better performance.

- Optimized API Responses: Reduced payload sizes, efficient querying.
- New Endpoints: For features like event sharing, admin moderation, etc.
- Rate Limiting: To prevent abuse and improve security.
- Authentication/Authorization Improvements: Role-based access, token refresh.
- Purpose: Ensure the backend can scale, remain secure, and support new features.

4. Performance & Security Checks

Before deployment, a thorough check is done to ensure the app performs efficiently and is protected against vulnerabilities.

- Performance: Use tools like Lighthouse to audit and optimize load times, render speed.
- Code Splitting: To reduce bundle size.

- Security:

- Input validation and sanitization.
- HTTPS enforcement.
- Environment variables for sensitive data.
- Preventing common threats (XSS, CSRF, etc.).
- Purpose: Provide a fast, stable, and secure experience to users.

5. Testing of Enhancements

Conduct rigorous testing to ensure all new changes work as intended and do not break existing functionality.

- Unit Testing (e.g., using Jest)
- Integration Testing (ensuring front-end and back-end communicate properly)
- End-to-End Testing (e.g., using Cypress or Playwright)
- User Acceptance Testing (UAT) with a few real users or testers.
- Purpose: Maintain app stability and confidence before deployment.

6. Deployment

Finally, the fully tested and enhanced version of the app is deployed to a cloud hosting platform:

Platforms:

- Netlify: Good for JAMstack apps and static site deployment.
- Vercel: Optimized for React/Next.js applications.
- Other Cloud Platforms: AWS, Firebase, or Heroku for full-stack apps.
- CI/CD Integration: Configure auto-deploy from GitHub for continuous updates.
- Domain Setup: Custom domain mapping and SSL setup.