

COLLEGE CODE : 6109

COLLEGE NAME : JAYALAKSHMI INSTITUTE OF TECHNOLOGY

DEPARTMENT : COMPUTER SCIENCE AND ENGINEERING

STUDENT NMID : 849dbfcce321f459f0148603bb827f00

ROLL NO : 610923104317

DATE : 27-09-2025

Completed the project name as

Phase_3_TECHNOLOGY PROJECT

NAME : IBM-NJ EVENT SCHEDULER APP

SUBMITTED BY,

NAME : MAHENDIRAN M

MOBILE NO : +91 9080640249

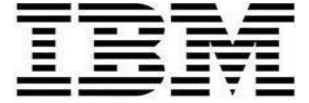
Phase3—IBM-NJ EVENT SCHEDULER APP

1. ProjectSetup

- Initialize the project repository on GitHub.
- Setup **frontend environment** using React.js (or HTML, CSS, JS with Tailwind CSS).
- Setup **backend environment** with Node.js and Express.js.
- Connect backend with a database (MongoDB/MySQL).
- Install dependencies (React Router, Axios, Express, JWT, bcrypt, Mongoose/Sequelize).
- Initialize the project environment using a modern framework (e.g., React Native/Flutter for mobile, or React/ Vue for web).
- Setup development tools, dependencies, and folder structure.
- Configure environment files and package managers.
- Integrate a task manager (like Trello, Jira) for sprint tracking.

2. Core Features Implementation

- **User Authentication:** Login, Register, and Logout with secure password hashing.
- **Event Management:**
 - Create an event (title, description, date, time, venue).
 - Update or cancel an existing event.
 - View event details in a dashboard or calendar view.
- **Event Reminders:** Notification system to alert users before event start.
- **Participant Management:** Option to invite participants (email/notification).
- **User Registration/Login** (Authentication system)
- **Create Events:** Users can schedule events by inputting title, description, time, date, and location.
- **Edit/Delete Events**
- **View Scheduled Events** in a list or calendar view.
- **Reminder/Notification System** (if possible within MVP scope)
- Basic UI/UX for navigation between views/screens.



3. DataStorage(LocalState/Database)

- **FrontendLocalState:**
 - Storetemporarydata likeevent forminputs,authenticationtokens,andUIstatesusingReact's useState/useContext or Redux.
- **Database:**
 - Usertable/collection(ID,username,email,password).
 - Eventstable/collection(ID,title, description,date,time,venue,participants).
 - Notifications/remindertable/collection(linkedtoevents).
- Use**localstatemanagement** (e.g.,Redux,React ContextAPI,orProviderforFlutter)formanagingtemporary user and event data.
- Setuppersistentstorage:
- **Frontend-onlyMVP:**UselocalStorageorAsyncStorageforstoringdatatemporarily.
- **Full-stackMVP:**Integratewithabackend(e.g.,Firebase,Node.js+MongoDB)foruserandevent data persistence.
- Defineandstructuredatamodels(Event,User).

4. TestingCoreFeatures

- Test**UserAuthentication**→userscanregister/loginsuccessfully.
- Test**EventCRUDoperations**→create,update,deleteevents.
- Test **EventReminders**→notificationstriggeratcorrecttimes.
- Test**Multi-userfunctionality** →eventssynccorrectlybetweenusers.
- UnittestingwithJest/MochaforbackendAPIs.
- Writeunit testsforcorefunctionalitieslikeeventcreation,deletion,and validation.
- Performmanualtesting ofuserflows:
- Register→Login→ScheduleEvent→ViewEvent→Edit/Delete

- Usetestingtools(e.g., Jest, FlutterTest,PostmanforAPItesting).
- Bugtrackinganditerationbasedontestresults.

5. VersionControl(GitHub)

- Maintainrepositorywith**branches**foreachfeature(e.g.,auth-feature, event-feature).
- Regularcommitswithmeaningfulmessages.
- Use**GitHubProjects/Issues**fortasktracking.
- Enable**GitHubActions**forautomatedtestinganddeploymentpipeline.
- InitializeaGitrepositoryand pushcodeto GitHub.
- Usebranchingstrategy(e.g., main,dev,feature/*)forcollaboratedevelopment.
- Regularcommitswithmeaningfulmessages.
- Ensureacleanandwell-documentedcommithistory.