NORTHEASTERN UNIVERSITY

COLLEGE OF ENGINEERING

EVENT MANAGEMENT SYSTEM FOR NORTHEASTERN UNIVERSITY

EMGT5220 – ENGINEERING PROJECT MANAGEMENT, Sec 03

Spring 2025

Instructor: Bajracharya, Sharad

Team 2: Collective Solutions

Point Of Contact: Antony, Samson

Email: antony.s@northeastern.edu

Mobile: +1(857) 800-5850

Team Members:

Anand Girish, Aditya

Antony, Samson

Bhilave Ganesh, Shruti

Dave, Shreya

George Lal, Alan

Kalghatgi, Arnavi

Keluskar, Aarya

`

TABLE OF CONTENTS

EXECUTIVE SUMMARY **3**

1.0 INTRODUCTION 4

2.0 PURPOSE AND OBJECTIVES 4

2.1 PURPOSE **4**

2.2 PRIMARY OBJECTIVES **4**

3.0 TECHNICAL OVERVIEW 4

3.1 STUDENT WORKFLOW **5**

3.2 NEU CLUB WORKFLOW **5**

3.3 ADMIN WORKFLOW **6**

3.4 TECHNICAL SPECIFICATIONS AND SYSTEM COMPONENTS **6**

3.4.1 FRONTEND DEVELOPMENT 7

3.4.2 BACKEND DEVELOPMENT 7

3.4.3 USER AUTHENTICATION & ROLE-BASED ACCESS CONTROL (RBAC) 8

3.4.4 EVENT MANAGEMENT API DEVELOPMENT 8

3.4.5 NOTIFICATION SYSTEM 9

3.5 QUALITY ASSURANCE **9**

3.5.1 TESTING STRATEGY 10

3.5.2 PERFORMANCE & SECURITY TESTING 10

3.5.3 AUTOMATED & MANUAL TESTING 10

3.5.4 USER ACCEPTANCE TESTING (UAT) 10

3.5.5 POST-DEPLOYMENT MONITORING 10

3.6 COMPLIANCE **10**

4.0 IMPLEMENTATION PLAN 10

4.1 WORK BREAKDOWN STRUCTURE (WBS) **12**

4.3 RESPONSIBILITY CHART **12**

4.4 RESOURCE ALLOCATION **12**

4.5 STAKEHOLDERS **12**

5.0 EXECUTION PLAN 13

5.2 PROJECT AUDITING **15**

5.3 PROJECT CLOSURE **15**

6.0 RISK ASSESSMENT MANAGEMENT PLAN 17

6.1 Identification and Analysis of Risks **17**

6.2 Risk Categories **17**

`

6.3 Risk Register **18**
6.4 Continuous Risk Monitoring **19**
7.0 FINANCIAL PLAN WITH BUDGET 20
7.1 High-Level Summary **20**
9.0 REFERENCES 28
APPENDIX A 29
APPENDIX B: SCHEDULE 35
APPENDIX C: RACI MATRIX 44
APPENDIX D: BUDGET JUSTIFICATION 45
APPENDIX E: RESOURCE ALLOCATION 47

TABLE OF FIGURES

Figure 1: Student Workflow Diagram 5
Figure 2 : NEU Club workflow diagram 5
Figure 3 : Admin workflow Diagram 6
Figure 4 : Architecture Diagram 9
Figure 5 : Risk Matrix 26

LETTER OF TRANSMITTAL

Date: 04/11/2025

Prof. Sharad Bajracharya
Northeastern Graduate School of Engineering
130 Snell Engineering
360 Huntington Avenue
Boston, MA 02115

Dear Prof. Sharad Bajracharya,

We are pleased to present to you the project proposal for the Event Management System for Northeastern University. This report provides a detailed and comprehensive plan of developing a centralized platform tailored to support students, clubs and organizers to enhance event search, planning, coordination and execution of events on-campus.

The proposal includes details of the objectives, technical approaches, project timeline, financial requirement of the project, risk and mitigation strategies to ensure a timely and efficient project execution and deployment. The Event Management System aims to improve event and information accessibility and foster student engagement at the University.

We hope that the proposal meets your expectations and aligns with the vision of Northeastern University of leveraging technology to enhance services and engagement on-campus. The guidance and feedback we received throughout this semester has been valuable in refining our approach towards planning and managing the project.

Sincerely,
Anand Girish, Aditya
Antony, Samson
Bhilave Ganesh, Shruti
Dave, Shreya
George Lal, Alan
Kalghatgi, Arnavi
Keluskar, Aarya

`

EXECUTIVE SUMMARY

The Event Management System for Northeastern University is a project designed to create a centralized mobile application for students, clubs and organizations to streamline event organization, participation and engagement. This one-stop platform enables communication among the organizers and students, ensuring that the events are easily accessible. The application features event dashboards, event creation, reminders, RSPVs, recommendations, and calendar syncing for registered events. By integrating a user-friendly interface and adhering to the compliance policies, this application fosters seamless accessibility, enhances student engagement and improves the overall experience for the University community.

This application is designed for iOS and Android devices, ensuring widespread accessibility for users. The project follows a well-structured Work Breakdown Structure (WBS) classified into five key phases. The Project Initiation phase defines the scope, purpose and objectives of the project. It also includes technical, financial and operational feasibility study and team formation. The Project Planning phase defines the requirements, system architecture and design, collaboration setup, risk assessment and mitigation strategies, scheduling and budgeting, and compliance check. The Execution phase focuses on developing the core application, integrating features and conducting testing and user feedback. The Monitoring phase ensures that the application is successfully deployed, tested by users and documentation for future reference. Finally, the Closure phase focuses on post-launch support, reviews, project documentation, project handover and closure.

The project is allocated with a strategically planned budget of approximately $2,39,829.70, to ensure efficient execution of all the development phases. The major cost allocation of $1,93,315 is dedicated to labour costs supporting a skilled workforce responsible for design, development and implementation of the application. Software tools are allocated with a sum of $3,312 ensuring security, system reliability and overall system performance with hardware allocated with a budget of $21,400. Additionally, a 10% contingency fund of $21,802.70 is reserved to address unforeseen challenges. To ensure a structured and efficient utilization of funds, each project phase has a fixed budget allocated to it.

To ensure smooth coordination and execution of the project, the team utilizes various project management tools and methodologies like Microsoft Project to create Gannt Charts to efficiently manage the timeline. A RACI matrix is created to ensure accountability by clearly designating roles and responsibilities to various tasks within the project team. A risk register is maintained to identify potential risks and ranked based on their likelihood of occurrence and impact and strategies for mitigation are defined. The execution of the project is monitored and controlled by regular sprint meetings, reviews and change management practices. Additionally, rigorous testing, user feedback collection and implementation ensures high-quality delivery of the project to the University addressing the specific needs and requirements.

Ultimately, this project delivers a consolidated unified platform that enhances campus engagement by helping students navigate and register for events and for clubs to plan and organize the event seamlessly by getting the head count via RSPV feature and better manage future events based on student feedback By enabling better event planning, organizing, and interactive engagement, this event management system encourages networking, improves student involvement, and enhances overall engagement within the University community.

`

1.0 INTRODUCTION

Currently, Northeastern University students find it difficult to stay informed about activities on-campus due to scattered communication channels like emails, banners, and social media. It is difficult for clubs to effectively engage and for students to stay informed without a centralized system to locate events, send out reminders, and provide updates.

To address this issue, we will develop a centralized event management application accessible on iOS and Android platforms, where students can browse events, express interest, get timely reminders and personalized recommendations. Moreover, the clubs can effectively promote their events and boost the participation.

With a user-friendly design, secure authentication, and interactive features, this application will ensure that the students find a one-stop application that keeps them informed of the events on-campus. Unlike the scattered information about the events on social media, this dedicated platform is more reliable for event discovery and engagement.

2.0 PURPOSE AND OBJECTIVES

2.1 PURPOSE

The purpose of this project is to consolidate all information pertaining to the different activities hosted on campus by various student clubs and organizations in one platform. Students usually need to manually follow the different social media pages of the clubs and organizations to stay informed about these activities or they need to be an active member of the clubs to stay in the loop. For example, there's an Instagram page for the activities happening at the Curry Student Center, a separate one for Northeastern Explore Program, another one for Northeastern University's Office of Graduate Studies and may separate ones for Culture specific updates. It is practically impossible for students to follow each page and stay informed about all the events which might especially be of interest to them. They get to know about these events from word of mouth. This project addresses the need for a single platform to provide all this information. This specially makes it easier for the new incoming students to know about everything happening on campus. This platform will help them get acclimatized to the new environment comfortably.

2.2 PRIMARY OBJECTIVES

1. **Develop a Unified Digital Platform –** Create a centralized mobile application (iOS & Android) that consolidates all on-campus events information from all student clubs and organizations, ensuring seamless accessibility for all the users.

2. **Implement Comprehensive Event Management Features –** Integrate event listings, notifications, reminders, and calendar synchronization to streamline event tracking and enhance student engagement.

3. **Scalability, Security, and real-time Synchronization** – Build a robust and secure backend to handle large event data volumes while ensuring real-time updates and maintaining user data privacy and objectives.

3.0 TECHNICAL OVERVIEW

The technical overview provides a high-level summary of the application's architecture, components, and key functionalities. It serves as a blueprint for how the app is built, how different components interact, and how it ensures performance, security, and scalability.

`

3.1 STUDENT WORKFLOW

Students begin by signing up via Northeastern SSO. Once logged in, they can explore a list of registered clubs, browse their profiles, and view upcoming events. Students can also engage with events by viewing event details, RSVPing to confirm attendance, and receiving notifications about event updates. They can also like the event after attending it.

Through the app, students stay informed about club activities, receive notifications about new events, and announcements. This streamlined workflow ensures a seamless experience for students to discover, engage with, and contribute to various club activities within the app.

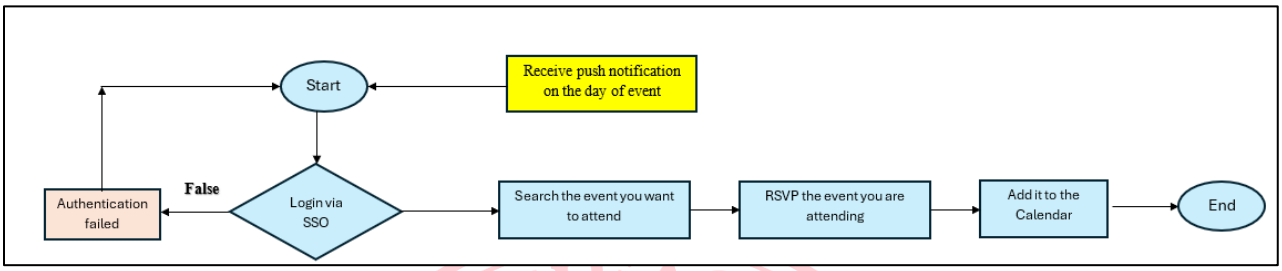


Figure 1: Student Workflow Diagram

3.2 NEU CLUB WORKFLOW

Clubs begin by submitting a registration request through the app, providing necessary details such as name, purpose, and contact information. This request is then sent to the admin (the platform administrator) for review. Once the admin approves or rejects the registration, a notification is sent to the club informing them of the decision.
Approved clubs can create and manage their profiles by updating details, adding images, and providing contact information. They can also organize events by adding event details such as name, date, time, location, and description.

Clubs can also leverage interactive features to enhance engagement with their events. Users have the option to RSVP for events, allowing clubs to track attendance and plan accordingly. Clubs can monitor these reactions to gauge interest in their events and content, helping them understand what resonates most with their audience. This feedback mechanism allows clubs to refine their activities and improve future events based on user preferences.

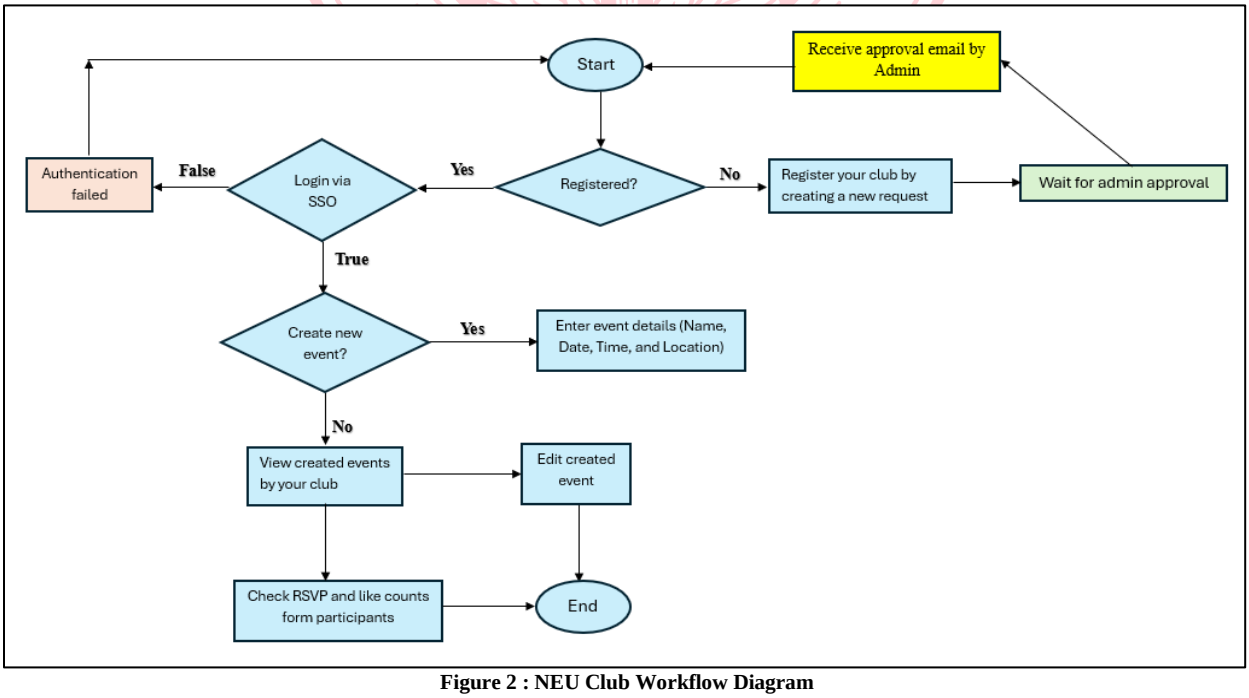


Figure 2 : NEU Club Workflow Diagram

`

3.3 ADMIN WORKFLOW

The admin begins by logging in to our app through Northeastern SSO. Once authenticated, access to the admin dashboard is granted based on their authorization status. Within the dashboard, the admin can manage clubs by viewing a list of registered users along with relevant details. Additionally, the admin can approve or reject new club registration requests. After deciding, a notification is automatically sent to the club, informing them whether their registration request has been accepted or rejected.

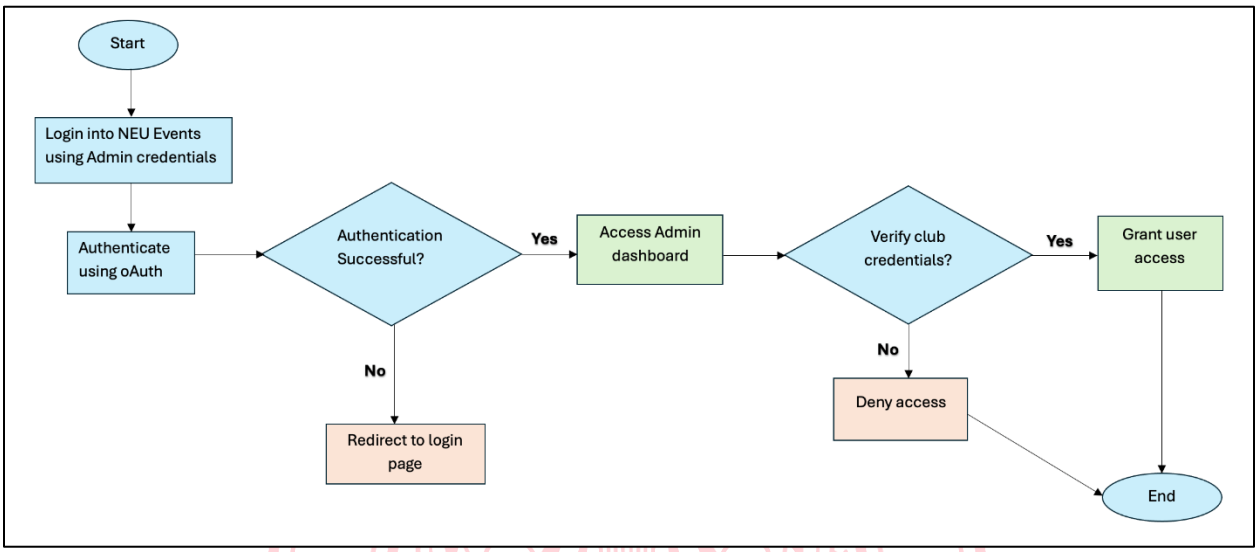


Figure 3 : Admin Workflow Diagram

3.4 TECHNICAL SPECIFICATIONS AND SYSTEM COMPONENTS

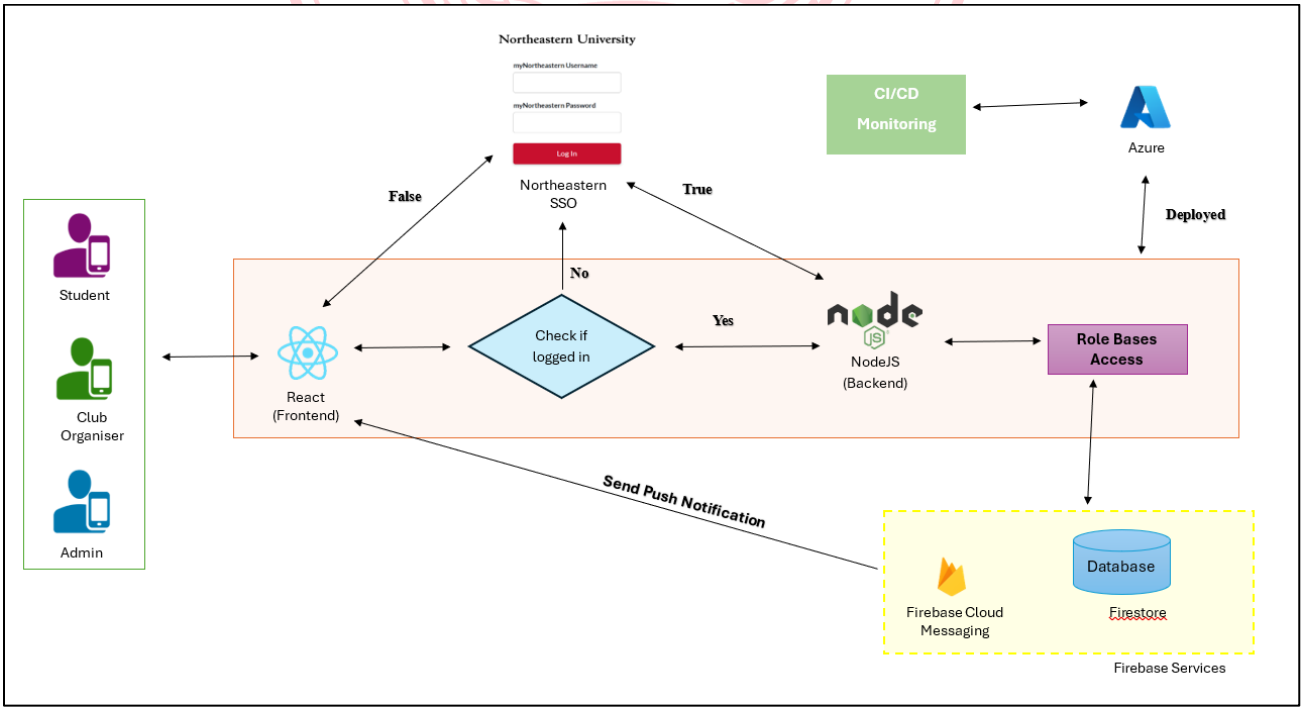


Figure 4 : Architecture Diagram

`

When a user opens the app, they begin by logging in through Northeastern University's SSO (Single Sign-On) using secure SAML authentication. Once authenticated, the app identifies their role—Student, Organizer, or Admin—and customizes access accordingly through Role-Based Access Control (RBAC).

For a student, the flow starts with browsing a curated list of events pulled from Firebase Firestore. Students can filter events by date, category, or club, and view detailed information including time, location, organizer, and images. If interested, they can RSVP directly through the app, which updates the RSVP count in real time and optionally syncs the event with their calendar. Students also receive automated reminders and real-time notifications through Firebase Cloud Messaging (FCM), ensuring they never miss an event. After attending, they can interact with the event by providing a like or dislike.

For a club organizer, the flow begins with submitting a club registration request through the app. Once approved by an admin, the organizer gains access to a dashboard where they can create and manage events, upload images, set participant limits, and track RSVP counts. Organizers also receive insights into event interest and can make edits to events even after publishing. Event changes trigger real-time updates to registered students.
The admin uses a separate interface to review and approve club registration requests, manage events, monitor app activity, and ensure compliance. They also oversee platform health, including user activity, event popularity, and system stability.

Behind the scenes, all interactions—RSVPs, notifications, profile updates, and media uploads—are synced in real time with Firebase, while the backend logic (hosted on Microsoft Azure) ensures fast and secure operations. Azure DevOps handles automated deployments and updates, maintaining system integrity throughout the app's lifecycle.

3.4.1 FRONTEND DEVELOPMENT

3.4.1.1. React Native Interface Development

The first step in building the event management platform is developing the React Native interface for event browsing, RSVP, and dashboards. This ensures that users can easily navigate through events, express their interest, and manage their interactions with the platform. The app will be designed to work seamlessly across iOS and Android, providing a consistent experience regardless of the device used.

3.4.1.2. Responsive UI/UX Implementation

A well-designed user interface (UI) and user experience (UX) is crucial for the success of the application. The platform will feature an intuitive and visually appealing design that supports smooth navigation. The User Flow Implementation will cater to students and attendees, allowing them to log in, authenticate their accounts, browse events, RSVP, receive notifications, and integrate with their calendars. On the other hand, the Admin Flow Implementation will enable event organizers and university staff to log in, manage events through a dedicated admin dashboard, and create or edit event details efficiently.

3.4.2 BACKEND DEVELOPMENT

3.4.2.1 Server Setup

The backend of the application will use Firebase Firestore, a cloud-based database, to store and manage user profiles, event details, and registrations. To connect the backend with Firestore, we use the Firebase Admin SDK, which allows secure access to the database and enables the server to read and write data efficiently. When a user registers, their information such as name and email is saved in the user's collection, while event details like title, description, and date are stored in the events collection. The backen d ensures smooth communication with Firestore by using API routes, which handle different tasks such as adding new users, retrieving event details, and managing event registrations.

To keep the data safe, Firestore Security Rules are applied. These rules control who can read or write data, ensuring that users can only access their own information. Additionally, authentication mechanisms verify user identity before allowing any sensitive actions.

`

The backend follows a structured approach, separating different functionalities into organized files, making it easier to maintain and scale as the application grows. By leveraging Firestore's real-time synchronization, any updates made in the database instantly reflect on connected devices, providing a seamless user experience. This setup ensures that the app remains fast, secure, and capable of handling large amounts of data efficiently.

3.4.2.2 Configure Firebase as the NoSQL Database for Event and User Storage

Firebase Firestore is a NoSQL database that is cloud-hosted with the capability of supporting real-time synchronization of data and automatic scaling. Data is contained in collections and documents, where there is space to store structured data such as events and users. Firebase features built-in security rules and integrated authentication to ease access control and secure the data of the users. Backend will interact with Firebase Firestore using the Firebase Admin SDK, which allows secure and efficient read and write operations to save and retrieve user profiles, event details, registrations, and other dynamic elements.

3.4.2.3 Integrate SAML authentication for secure login

Security Assertion Markup Language (SAML) is a well-known Single Sign-On (SSO) protocol that allows users to authenticate with a trusted Identity Provider (IdP) once and gain access to multiple services without re-providing credentials. This method enhances security by outsourcing authentication to an external provider, without having to store user passwords locally. In this configuration, passport-saml, a Node.js module, will be used to handle authentication requests, decrypt SAML assertions, and establish secure user sessions.

3.4.3 USER AUTHENTICATION & ROLE-BASED ACCESS CONTROL (RBAC)

3.4.3.1 Authentication

Verifies the identity of a user before granting access to the system. In this application, authentication is performed through SAML SSO, where users can authenticate using the authentication system of an institution. User data (email and role) is stored in the database during authentication, and a secure session is created using JWT (JSON Web Token). Compliance will be ensured with WCAG (Web Content Accessibility Guidelines) and FERPA (Family Educational Rights and Privacy Act).

3.4.3.2 Set up multi-role access control (student, organizer, admin)

RBAC ensures different levels of access for various users based on their roles. There are three roles in this model:

1. Student – Can search events, enrol, and provide feedback (like).
2. Organizer – Can create, edit, and manage events.
3. Admin – All-access control, user administration, and security monitoring

3.4.4 EVENT MANAGEMENT API DEVELOPMENT

The Event Management API serves as the core backend component, enabling event creation, modification, retrieval, and deletion. It is designed using Node.js with Express.js for handling API requests and Firebase Firestore as the NoSQL database. The API supports secure, scalable, and real-time event operations, ensuring seamless interaction between students, organizers, and administrators.

3.4.3.1 Develop CRUD APIs for Event Creation, Updating, Retrieval, and Deletion

The API exposes CRUD (Create, Read, Update, Delete) operations to allow users and admins to manage event data.

- Create Event: Organizers can create events by providing details such as title, description, date, time, location, category, and images. A unique event ID is generated and stored in Firestore.
- Retrieve Event(s):
  1. Single Event Lookup: Fetches details of a specific event using its unique ID.
  2. Event Listings: Supports filters (e.g., date, category) to enhance user experience.
- Update Event: Allows authorized users to modify event details. Firestore triggers notify participants of any updates.
- Delete Event: Removes an event and updates related records

`

3.4.3.2 Integrate Event Metadata (Location, Time, Organizers, Images)

The API stores and retrieves metadata to provide users with comprehensive event details:

- Location Services: Location where the event will be hosted on campus E.g. Curry Ballroom
- Times: Time when the event starts and ends in EDT
- Organizer Details: Each event links to an organizer profile, allowing attendees to view event hosts.
- Event Images & Media: Images are uploaded to Firebase Storage, ensuring efficient retrieval and display within the app.

3.4.3.3 Implement Event Categorization, RSVP Tracking, and Participant Limits

Event Categorization: Events are categorized based on type (e.g., Workshops, Socials, Sports, Hackathons) to improve discoverability. A tag-based filtering system allows users to find relevant events quickly.

- RSVP Tracking
    - Users can RSVP to events, which updates a Firestore record linking count of RSVPs to event IDs.
    - Event capacity constraints ensure that once a participant limit is reached, users are placed on a waitlist.
    - RSVP status tracking helps organizers manage attendance, and users can cancel their RSVPs if needed.
- Participant Limits
    - Each event has a maximum participant limit, preventing overbooking.
    - Admins receive notifications when events reach capacity, allowing them to increase limits if necessary.
    a. The system can send automatic notifications to waitlisted users if spots become available.

3.4.5 NOTIFICATION SYSTEM

The notification system is designed to ensure real-time updates, automated reminders, and mobile alerts for event management. Built using Firebase Cloud Messaging (FCM), Firestore, and React Native, this system enhances user engagement by delivering timely notifications.

3.4.5.1 Real-time Notifications for Event Updates

To keep users informed about changes to their events, a real-time notification mechanism is implemented. Firestore database triggers detect modifications in event details such as time, location, or description. When an update occurs, a Firebase Cloud Function retrieves the list of affected users and sends notifications via Firebase Cloud Messaging (FCM). Users receive these alerts as in-app notifications when the app is open or as push notifications when it is in the background. Firestore's real-time listeners (onSnapshot()) ensure that updates reflect instantly within the app interface.

3.4.5.2 Automated Reminders via Push Notifications and Email

To reduce missed events, an automated reminder system is scheduled using Firebase Cloud Scheduler and Cloud Functions. These functions run at predefined intervals, such as 24 hours, 1 hour, or 15 minutes before an event, to send notifications. FCM is used to push reminders directly to mobile devices, while email notifications are delivered using SendGrid or Firebase Email Extensions. Users have the option to customize their notification settings, including reminder frequency and preferred delivery channels.

3.4.5.3 Mobile Alerts Using Firebase Cloud Messaging (FCM)

Firebase Cloud Messaging (FCM) ensures seamless push notification delivery across Android and iOS. Each device generates a unique FCM token upon installation, which is securely stored in Firestore and linked to the user's account. Notifications can be targeted individually using token-based messaging or sent to event attendees using topic-based messaging.

3.5 QUALITY ASSURANCE

Quality Assurance (QA) ensures that the event management system functions smoothly, meets performance expectations, and maintains security and usability standards. Our QA approach covers multiple testing phases, automation, and continuous monitoring to deliver a reliable product.

`

3.5.1 TESTING STRATEGY

Unit Testing: Validates core functionalities like authentication, event creation, and notifications.
Integration Testing: Ensures smooth interaction between frontend, backend, and third-party integrations (Google Calendar, University SSO).
End-to-End Testing: Simulates real user journeys from login to RSVP and event reminders.
User feedback and bugs fixing: Bugs reported by users through ticket submissions on the Resource Hub provided by the library.

3.5.2 PERFORMANCE & SECURITY TESTING

Load & Stress Testing: Evaluates system performance under peak traffic and large-scale event participation.
Security Testing: Includes penetration testing, SSO authentication validation, and encryption checks to protect user data.

3.5.3 AUTOMATED & MANUAL TESTING

Automated Testing: Uses tools like Selenium and Jest to streamline UI and API testing.
Manual Testing: Performed by testers and users to identify usability issues that automation may miss.

3.5.4 USER ACCEPTANCE TESTING (UAT)

Pilot Testing: Real students and university admins test the system in actual scenarios.
Bug Fixing & Optimization: Refinements based on user feedback before final deployment.

3.5.5 POST-DEPLOYMENT MONITORING

Real-Time Error Tracking: Uses Firebase Crashlytics and Google Cloud Logging to detect and fix issues instantly.
Regression Testing: Ensures system stability after updates.
Continuous Improvement: Collects user feedback from the tickets raised for future enhancements.

3.6 COMPLIANCE

To ensure compliance with WCAG (Web Content Accessibility Guidelines) and FERPA (Family Educational Rights and Privacy Act), we will conduct accessibility and privacy audits, verifying that our system is accessible to users with disabilities and that student data is securely managed in accordance with FERPA regulations. We will also integrate clear privacy policies and user consent mechanisms, ensuring users are informed and have control over their data. Additionally, a review session with NU's IT and Legal department will be scheduled to validate cloud usage, confirm compliance with security protocols, and ensure no unauthorized third-party access to sensitive data.

4.0 IMPLEMENTATION PLAN

1. PROJECT INITIATION PHASE

This phase will establish the foundation of the project by defining key objectives, engaging stakeholders (Northeastern University Clubs and Students), and securing necessary approvals and permissions. A project kickoff meeting will be conducted to align expectations, define the scope, and assign responsibilities through a Project Charter. A Feasibility Study will assess technical and financial viability, ensuring the project aligns with university goals. Compliance approvals will be obtained to adhere to institutional policies before proceeding to the planning phase.

2. PROJECT PLANNING PHASE

This phase includes defining the requirements, compliance measures, system designs, cost estimation, resource allocation, scheduling, and risk assessments. The core functional requirements like the event listing, club onboarding, notifications, and an admin dashboard will be finalized. The system designs will be selected, with React Native for mobile development, Node.js for backend and Firebase for authentication. A detailed project schedule will be created, mapping out milestones, deliverables and dependences, and resource allocation will be done. The cost estimation for the project along with risk assessment and mitigation plans will be addressed in this phase.

`

3. PROJECT EXECUTION PHASE

a) Design - This phase focuses on creating a visual blueprint of the application for an intuitive user experience. Wireframes and high-fidelity mockups will be developed using Adobe XD and Figma for key screens i.e. event discovery, event details, student, and admin dashboard. The system architecture will define the backend, frontend, and database for seamless integration. The UI will be designed for iOS and Android compatibility, adhering to WCAG standards.

Key Activities:

1. User Research and Wireframing
2. High-Fidelity Prototyping and Visual Design

b) Development - This phase involves coding, integration, and testing of system components. The backend will be built using Node.js and Express.js with authentication, RBAC, and APIs for event management. The database schema will use Firebase for efficient storage. The front end will be developed using React Native, ensuring smooth navigation. Security enhancements such as data encryption and caching will be implemented. Finally, integration and real-time testing of APIs, event workflows, and notifications will be conducted before moving to the testing phase.

Key Activities:

1. Authentication Module
    a. Implement NEU SSO integration via Firebase Auth or custom OAuth2.0.
    b. Validate login via myNortheastern credentials and restrict access by role (student, club, admin).
2. Frontend Development - Student Flow
    a. Fetch and display all upcoming events using paginated scroll.
    b. Apply filters (date, category, club).
    c. Store RSVP events in "My Events" tab.
    d. Build in-app calendar to display RSVP'd events.
    e. Highlight dates with scheduled events.
    f. Past Events & Feedback:
    g. Enable like/dislike buttons (icon-based) with feedback storage.
3. Frontend Development - Club/Admin Flow
    a. Form to create new events with image upload, date/time, description, and tags.
    b. Table of "Your Events" with options to edit or delete events.
4. Admin Dashboard:
    a. Approve/reject onboarding requests with role update.
    b. Backend Development
5. API Layer:
    a. Build RESTful APIs using Express.js for event creation, RSVP, event listing, notifications.
    b. RBAC (Role-Based Access Control):
    c. Assign permissions for Students (view, RSVP), Clubs (create/edit events), Admins (approve clubs).
6. Firebase Integration:
    a. Firestore schema for users, events, rsvps, feedback, clubs.
    b. Cloud Functions for triggered events (e.g., sending notifications).
7. Push Notifications
    a. Configure Firebase Cloud Messaging (FCM).
    b. Create in-app notification screen with read/unread filter.
8. Security Enhancements
    a. Encrypt sensitive data in Firestore (user details, RSVP records).
    b. Enable rules to restrict unauthorized writes.
    c. Prevent duplication of RSVPs and restrict editing past events.

`

c) **Testing** - A User Acceptance Test (UAT**)** will be conducted using Firebase App Distribution with select students and event organizers to evaluate event creation, RSVP functionality, and notifications. Feedback will be gathered to refine system performance, usability, and security before the official launch.

4. PROJECT MONITORING PHASE

The mobile app will be deployed on Google Play Store & Apple App Store, with backend services hosted on Microsoft Azure for scalability. A CI/CD pipeline with Azure Ops will automate deployment, ensuring seamless updates and system stability. User training would be given to administrators for proper use of the platform.

Key Activities

1. Set up Azure DevOps pipelines for auto build & deployment for Android & iOS builds.
2. Backend deployment and monitoring logs.

5. PROJECT CLOSURE PHASE

Final system validation and stakeholder approval will confirm project completion, followed by documentation handover. Post-launch support, user feedback and application stability which will guide future enhancements and system optimizations.

4.1 WORK BREAKDOWN STRUCTURE (WBS)

To ensure organized execution and effective project tracking, a detailed **Work Breakdown Structure (WBS)** was developed based on PMI's five-phase project lifecycle: Initiation, Planning, Execution, Monitoring, and Closure. The WBS decomposes the project into manageable tasks and subtasks, allowing clear assignment of responsibilities, timeline estimation, and resource planning. It ensures alignment between scope, deliverables, and milestones while minimizing the risk of overlooked activities. Each phase outlines specific objectives such as stakeholder engagement, requirement gathering, UI/UX design, system development, deployment, and post-launch support. Refer to the complete WBS in Appendix A.

4.3 RESPONSIBILITY CHART

The RACI matrix is a valuable project management tool that defines the roles and responsibilities of each stakeholder in the project and also ensures that each task has been assigned to the appropriate stakeholder by categorizing them as Responsible (R) - who performs the task, Accountable (A) - who has final authority, delegates and approves, Consult (C) - subject matter experts and Informed (I) - who needs to be informed but does not actively contribute. This well-structured approach enhances efficiency by defining accountability, improves communication among stakeholders and ensures smooth project execution and delivery.
The matrix consists of tasks on one axis and stakeholders on the other, with each task assigned with R-A-C-I notations making it convenient and easy to view the responsibilities briefly. This matrix reduces misunderstandings and task overlaps, ensuring efficient collocations and transparency. Refer to Appendix for a detailed matrix.

4.4 RESOURCE ALLOCATION

To allocate resources to this project, we have created a detailed requirement chart for all available engineers. Refer to the details in Appendix E.

4.5 STAKEHOLDERS

Primary Stakeholders: Students of Northeastern University (Including Student Clubs and Organizations)

- Students are the primary users of the event management system. They directly benefit from the one-stop platform for event discovery.
- Clubs and Organizations will particularly benefit from the admin workflow that helps them create, promote and monitor events.

`

Secondary Stakeholders: University Administration and IT Department

- The University Administration play a critical role in gatekeeping role by approving events before they go live to ensure alignment with university policies, scheduling rules and resource availability. They also enforce FERPA and University IT policies, which the platform must follow.
- The IT Department ensure the system integrates smoothly with existing campus platforms (email and login systems) and meets university standards for data security and performance.

Internal Project Stakeholders:

- **Project Manager:** Oversees project execution across all phases, ensuring alignment with university policies. Manages risk assessment, stakeholder communication, and resource allocation.
- **DevOps Manager:** Manages cloud deployment, CI/CD pipelines, and system security. Ensures seamless integration with university systems and optimizes infrastructure performance.
- **UI Designer:** Responsible for overseeing the design and development of the platform's user interface, ensuring that it is visually appealing and optimized for both devices iOS and Android.
- **UX Designer:** Designs intuitive user interfaces and ensures a seamless user experience. Conducts usability testing and modifies UI elements based on student feedback. She also plays a key role during the testing phase to validate SDE interns from Khoury College through NU Works.
- **System/Database Architect Manager:** Designs database architecture and ensures efficient data storage and retrieval. Works with UI/UX and PM teams to meet system functionality requirements. She also works with UI and PM for functional requirements.
- **Risk and Compliance Manager:** Ensures compliance with university policies and legal standards, focusing on data privacy and security. Manages documentation for regulatory compliance and collaborates with IT on operational integration.
- **Tester 1 Lead:** Performs functional and non-functional testing to ensure system reliability and performance. Conducts test case execution, bug reporting, and regression testing.
- **Tester 2 Lead: Hired through NU Works:** Assists in end-to-end testing, UI/UX testing, and bug reporting. Works with the development team to validate platform stability before release.
- **Marketing and Engagement Lead (Northeastern Communication Team):** Develop strategies for student and faculty adoption through targeted outreach. Executes promotional campaigns and monitors user engagement metrics.
- **NEU Project Admin:** Responsible for internal posting in NU Works and would be identified within the existing admins. They would also be responsible for recruiting interns and scheduling interviews with PMs. They also make sure that the project follows all the policies and guidelines given by Northeastern University.
- **Full Stack Software Developers (2): Hired through NU Works:** Develops frontend and backend components using React Native and Node.js. Implements authentication, event tracking, and database integrations.
- **Planning Phase:** Conducts student and faculty surveys to identify effective engagement strategies and preferred communication channels.
- **Development Phase:** Collaborates with UI/UX design to ensure seamless onboarding and engagement mechanisms.

Vendors and Technology Partners

Below is the list of vendors and technology partners:

1. **Cloud & Infrastructure:** Microsoft Azure, Firebase
2. **Development & DevOps:** React Native, Node.js, Jenkins, GitHub Actions
3. **Authentication & Security:** Northeastern OAuth SSO
4. **Notifications & Messaging:** Firebase Cloud Messaging

5.0 EXECUTION PLAN

5.1 Project Monitoring & Control

The progress of this project will be monitored using a combination of tools, stand-up meetings and team-based reviews to ensure that the project aligns with the scope, timeline, budget and objectives of the project**.**

`

Monitoring Activities

- Key Performance Indicators and Metrics: The indicators and metrices like the development milestones, timelines, budget and schedules will be tracked and adhered to. Additionally, user feedback during testing and system reliability will be tracked. The project timeline will be tracked using the Gantt Chart.
- Progress Review and Tracking: The status updates on the tasks assigned, roadblocks if any, and requirements will be discussed in the daily/weekly stand-up meetings.
- Stakeholder Feedback: Timely user feedback from students, clubs and the University Admins to ensure that the application meets expectations and works seamlessly.

Tools for Monitoring

- Microsoft Project: The budget and schedule of the project will be tracked using Gantt Charts.
- Microsoft Teams: The communication and collaboration among team members and cross-functional teams would be via MS Teams and in-person meetings.
- Azure Monitor: The performance of the application will be tracked and monitored using Azure Monitor.
- Jira: The backlog items, issue resolution and project sprint progress will be tracked using Jira.

Control Category	Control Description	Frequency	Record/ Output	Responsibility
Feasibility Controls	Conduct technical, financial, and operational feasibility studies	One-time	Feasibility Reports	Project Manager, Technical Lead
Stakeholder Alignment	Define scope, success criteria, and communication plans	One-time	Project Charter, Meeting Minutes	Project Manager
Compliance & Security	Verify WCAG & FERPA compliance, assess university IT policies	Scheduled Reviews	Compliance Reports	IT team
Risk Management	Define risk mitigation strategies for prospective risks	Continuous	Risk Register	Project Manager, Technical Lead
Development	Conduct code reviews, security testing, and performance testing	Frequent	Code Review Reports, Test Logs	Development team
User Testing & Feedback	Conduct usability testing, UAT	Iterative	User Feedback	UI/UX Team
Deployment & Performance	Monitor system uptime, API response times, error tracking	Continuous	Performance Reports, Incident Logs	DevOps Team
Data & Access Handover	Transfer ownership of cloud resources	One-time	Access Transfer Logs	IT team
Final Handover Report	Compile system summary & confirm stakeholder transition	One-time	Final Handover Report	Project Manager, University Stakeholders
Project Closure Meeting	Review project outcomes, lessons learned, and final approvals	One-time	Meeting Minutes, Lessons Learned Report	All the Stakeholders

Monthly sprint review meetings will provide a clear picture of the progress on the project, the lessons learnt, and roadblocks to ensure proper planning for the subsequent sprints. The progress will be documented and shared with the Project Manager and Stakeholders for better alignment.

`

5.2 PROJECT AUDITING

Project auditing will be conducted to ensure the application meets technical, security, and compliance requirements.

Phase-Wise Audits

5.2.1 Performance and Scalability Audit

- **Frontend Performance:** Use VS Code to evaluate page load speed, accessibility, and UI performance of the entire application. React.js components will be optimized by reducing unnecessary re-renders and implementing lazy loading for better responsiveness.
- **Backend & Database Load Testing:** Use Azure Monitor to analyze Node.js API performance under peak load. Firebase Firestore queries will be optimized by indexing frequently used fields and reducing redundant read/write operations to improve response times.
- **CI/CD Pipeline Efficiency:** Azure DevOps build logs will be analyzed to identify slow build times, failed deployments, or misconfigurations that slow down continuous integration and delivery. Deployment processes will be streamlined by using incremental builds and caching dependencies.

5.2.2 Compliance and Accessibility Audit

- **Frontend Compliance:**
  - Run Lighthouse audits to ensure adherence to ADA (Accessibility) guidelines.
  - Conduct screen reader testing for visually impaired users.
- **Device Compatibility:**
  - Use Selenium for automated UI testing across different mobile devices.
- **API & Data Security Compliance:**
  - Validate Firebase Authentication & Role-Based Access Control (RBAC) using Postman.
  - Ensure Firebase Firestore & Cloud Storage follow encryption best practices.

5.2.3 Security and Data Protection Audit

- **Authentication & Authorization Testing:**
  - Review Firebase Authentication security with Postman.
  - Implement two-factor authentication (2FA) for admin users.
- **Data Storage & Security Measures:**
  - Ensure Firebase Firestore uses end-to-end encryption and restricts unauthorized access.
  - Conduct penetration testing to identify potential security threats.
- **Code Review & Vulnerability Testing:**
  - Use VS Code & GitHub to review code security warnings.
  - Track security issues and fixes using Jira.

5.2.4 Final Audit & Deployment Readiness

- **Comprehensive Performance Testing:**
  - Azure Monitor to track real-time app performance under high traffic.
  - Validate Firebase database performance under real-world event conditions.
- **Regulatory & Institutional Compliance:**
  - Confirm adherence to GDPR and university IT security policies.
  - Ensure all accessibility, security, and compliance issues are resolved.
- **Final Report & Stakeholder Approval:**
  - Document audit results in a detailed report for stakeholders.
  - Approve deployment after final security and performance checks.

5.3 PROJECT CLOSURE

The **Project Closure** marks the final phase of the event management platform's development, signifying the formal conclusion of all planned activities and the successful transition of ownership to Northeastern University. This phase ensures that deliverables have been met, stakeholder expectations are satisfied, and lessons learned are documented for future reference.

`

5.3.1 Final Validation and Approval

- Conduct a comprehensive review of project deliverables to ensure alignment with predefined success criteria.
- Validate that all functional and non-functional requirements have been fulfilled, including accessibility, security, and scalability standards.
- Obtain formal sign-off from all primary stakeholders, including the university administration, IT department, and student representatives.

5.3.2 Documentation Handover

- Deliver complete technical documentation including:
  - System architecture
  - API documentation
  - Database schema
  - CI/CD pipeline configuration
  - Security protocols and compliance checklists
- Provide end-user manuals and administrator guides for students, club organizers, and university staff.
- Archive all documentation in a centralized and secure repository for future access.

5.3.3 Knowledge Transfer

- Conduct structured training sessions with the Northeastern IT team and designated system administrators.
- Walk through system maintenance procedures, backup strategies, and monitoring tools (e.g., Firebase Crashlytics, Azure Monitor).
- Provide a detailed overview of incident response and recovery processes.

5.3.4 Transition of Ownership

- Transfer ownership of all cloud resources, including Firebase, Azure accounts, and GitHub repositories.
- Revoke old developer and testing credentials, issuing new access permissions to authorized university personnel.
- Share all legal and compliance-related documentation to ensure continuity in data protection and platform integrity.

5.3.5 Project Retrospective and Lessons Learned

- Organize a final project retrospective meeting with all team members and key stakeholders.
- Document challenges faced, solutions implemented, and recommendations for future similar initiatives.
- Identify opportunities for system improvements based on pilot feedback and testing results.

5.3.6 Final Handover Report

- Compile a comprehensive report detailing:
  - Project objectives and outcomes
  - Technical infrastructure
  - Compliance and audit results
  - Performance metrics and usage projections
  - Maintenance and support recommendations
- Secure approval and acknowledgment from all relevant stakeholders confirming project completion and readiness for ongoing use.

`

6.0 RISK ASSESSMENT MANAGEMENT PLAN

6.1 Identification and Analysis of Risks

The purpose of this section is to identify, analyse and prepare for potential risks that could affect the successful implementation of the Event Management System (EMS) project. Risks have been categorized into technical, operational, security, financial, compliance, and workforce-related areas. Each risk that has been identified has been assessed for its likelihood of occurrence, potential impact, severity ranking and appropriate mitigation strategies have been proposed.
The risk assessment will be continuously updated throughout the project lifecycle. Early identification and planning will help the team proactively avoid delays, ensure security and compliance, and meet quality and user satisfaction goals.

6.2 Risk Categories

A. Technical Risks

- **Integration Delays:** Delays in setting up Single Sign-On (SSO) integration with Northeastern University's authentication system. It could result in delayed user authentication implementation and overall development timeline, if not prevented.
- **Cross-Platform Compatibility Bugs:** Errors in iOS or Android deployment due to device fragmentation and potential inconsistencies across iOS and Android devices may cause UI/UX issues and as a result poor user experience can lead to low adoption rates.
- **Database Failures:** Event data updates may not reflect in real-time due to Firebase sync delays or offline caching as a result users may see outdated event info.

B. Security Risks

- **Unauthorized Access:** Failure to implement proper access control may allow users to access restricted data and can lead to breach student privacy and data security.
- **Encryption Failure:** Lack of strong encryption (e.g., AES-256) can expose student records. Sensitive user data is stored or transmitted without adequate encryption can lead to a high risk of data leaks or breaches
- **Vulnerable APIs:** Open REST APIs without authentication can be exploited. Public-facing APIs accessible without proper authentication, can lead to Vulnerability to injection and misuse attacks.

C. Operational Risks

- **Low Adoption Rate:** Students and clubs may not use the app due to lack of awareness or value proposition would lead to Failure to achieve intended engagement and purpose.
- **Inefficient Bug Fix Cycles:** QA/tester backlog or unclear ownership may slow down issue resolution can lead to prolonged bugs in turn reducing user trust.
- **Event Overload:** System may struggle to handle large-scale events or concurrent usage.

D. Financial Risks

- **Budget Overruns:** If the Project costs exceed initial estimates due to scope creep or hiring delays, it can cause delays in procurement of software licenses or testing tools.
- **Delayed Funding:** Delays from NEU in approving payments, internships, or software subscriptions can slow onboarding and progress.

E. Compliance Risks

- **FERPA Violations**: Collection or storage of student information without adequate user consent can lead to Legal and reputational consequences.
- **University IT Policy Violation**: Firebase or third-party services not aligning with NEU's IT policies can result in potential discontinuation of service.
- **App Store Approval Rejection:** Failing to meet Apple or Google data compliance policies can lead to app rejection during the submission process to the App Store (Apple) or Google Play Store, delaying launch.

F. Workforce Risks

`

- **Internship Gaps:** Delays in onboarding interns from NUWorks or Unavailability of Interns due to academic workload can lead to delays in development or testing.
- **Skill Mismatch:** Inexperienced interns may delay critical development tasks. They may not have required tech stack experience leading to longer onboarding time and rework.
- **Key Resource Burnout:** Overreliance on PM, UI/UX leads, or DevOps engineer can lead to Critical decision-making or execution delays

6.3 Risk Register

Risk Description	Likelihood	Impact	Rank (Likelihood X Impact)	Mitigation
Integration Delays	High	Medium	6	Break down tasks into smaller chunks, maintain clear communication between front-end and back-end teams, use continuous integration for faster deployment.
Cross-platform Compatibility bugs	High	Low	3	Conduct early cross-platform testing, use tools like React Native to minimize inconsistencies, implement unit and UI testing for all platforms.
Database failures	Low	Medium	2	Implement regular database backups, use automated monitoring tools like Google Cloud Monitoring, and have a database failover plan in place.
Unauthorized access	Medium	Medium	4	Implement robust authentication (MFA, Role-Based Access Control), use encryption, regularly review access logs, and conduct security audits.
Encryption failure	Low	High	3	Use proven encryption standards (AES-256), test encryption regularly, and audit cryptographic algorithms to ensure reliability and security.
Vulnerable API's	Medium	Medium	4	Regularly test APIs with security tools, enforce API versioning, perform regular security reviews, and integrate API security best practices (rate-limiting, etc.).
Low Adaption Rate	Medium	Low	2	Conduct market research and user testing before full launch, offer incentives for early adopters, and provide user education and support.
Inefficient Bug Fix cycles	High	Low	3	Implement agile practices with frequent sprints, use automated testing for early bug detection, and prioritize bug fixes based on severity.
Event Overload	Low	Low	1	Implement event moderation tools, allow limited event creation features for unverified users, and monitor system performance to handle overload.
Budget Overruns	Medium	Medium	4	Regularly track project expenses, implement cost-cutting measures, and ensure alignment of scope with available budget to avoid overruns.
Delayed Funding	Medium	Medium	4	Establish a funding timeline, have contingency funds, and ensure all stakeholders are aligned on funding requirements.
FERPA/GDPR Violations	Low	High	3	Ensure compliance with FERPA/GDPR regulations through regular audits, use data anonymization, and provide user consent management for personal data handling.
University IT Policy Violation	Low	Medium	2	Regularly review university IT policies, involve university IT department in early planning stages,

`

				and ensure that the project complies with institutional policies.
App Store Approval Rejection	Medium	Low	2	Review app store guidelines beforehand, ensure the app adheres to guidelines, and maintain thorough testing and quality assurance before submitting the app.
Internship Gap	Medium	Low	2	Plan ahead for internship needs, ensure that team members have proper documentation, and provide support for new interns during onboarding.
Skill Mismatch	Medium	Low	2	Clearly define roles and required skills during the hiring process, provide training, and offer mentorship for upskilling the team.
Key Resource Burnout	High	Medium	6	Monitor workload distribution, provide regular feedback and support, rotate tasks between team members, and ensure appropriate breaks and time off.

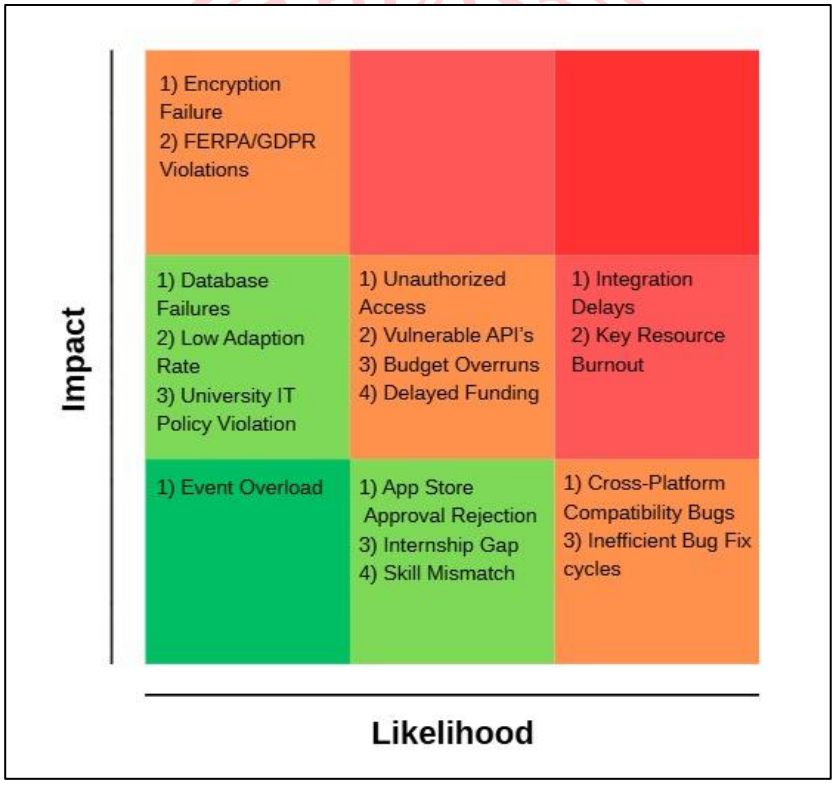


Figure 5: Risk Matrix

6.4 Continuous Risk Monitoring

Risks will be reassessed bi-weekly during sprint retrospectives. Jira and Airtable will be used for live tracking. Red flags (rank > 6) will be escalated to the Compliance and Operations Manager (Arnavi) and Project Manager (Samson).

`

7.0 FINANCIAL PLAN WITH BUDGET

7.1 High-Level Summary

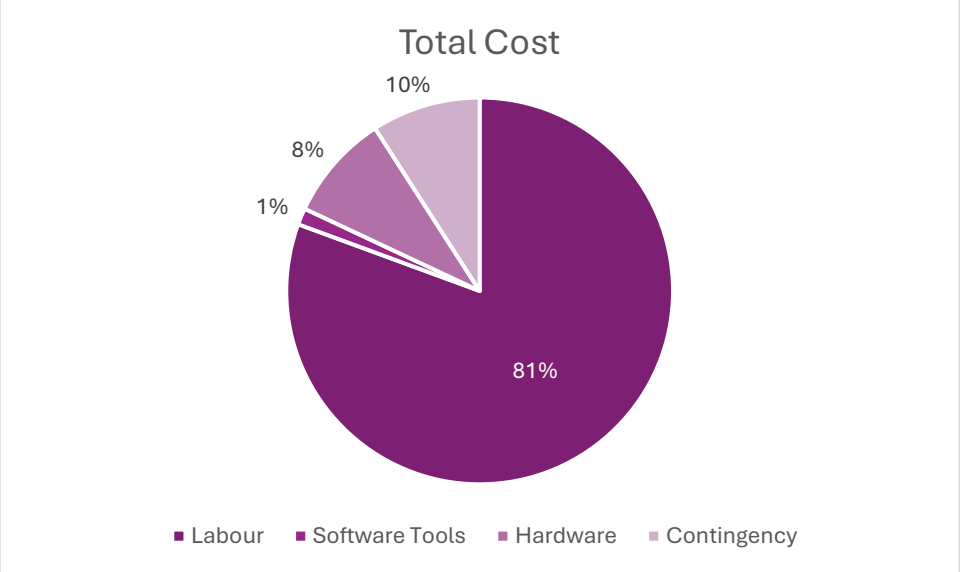
Below is a high-level financial plan that outlines the budget framework, spanning five distinct phases. This step-by-step process from first requirements gathering through the ultimate launch and maintenance ensures that each detail is addressed with precision to benefit the Northeastern community.

- **Key Cost Components:**
  - **Project Management & Planning** – Managing schedules, stakeholder communication, and risk.
  - **Design & Prototyping** – UI/UX design services, prototyping tools, and user testing.
  - **Development & Integration** – Software development, cloud hosting, and integration with existing university systems.
  - **Testing & Quality Assurance** – QA tools, test environment setup, and user acceptance testing.
  - **Launch & Maintenance** – Deployment, training, ongoing support, and updates.
- **Contingency Reserve:** 10 % of the total budget to address unforeseen costs.

Table: High-Level Summary

Resources	Total Cost
Labour	$ 1,93,315.00
Software Tools	$ 3,312.00
Hardware	$ 21,400.00
Contingency	$ 21,802.70
Total	**$ 2,39,829.70**

Figure: Budget Summary Chart



`

The Pie Chart above shows the Breakdown of the total budget that would be needed to build the NEU Events application. This includes labour costs which is about 81%, software tools of 1%, hardware required which is 8% for the project and contingency costs of 10%.

Labor Cost Table:

Title	Cost/Hour
Project Manager	$55.00/hr
UI Designer	$45.00/hr
UX Designer	$45.00/hr
System Architect	$70.00/hr
Compliance Manager	$50.00/hr
UAT Tester	$50.00/hr
DevOps Engineer	$65.00/hr
NU Admin	$50.00/hr
Software Developer 1	$28.00/hr
Software Developer 2	$28.00/hr
Performance Tester	$45.00/hr

The above table shows the breakdown of our labour cost. Based on the role designation the hourly rate of labour is given. Since all the phases would not require all the resources at all times, the phase wise breakdown has been calculated.

Hardware Cost Breakdown Table:

Hardware	Quantity	Cost	Total Cost
Laptop	12	$ 1,500.00	$ 18,000.00
Apple phone	2	$ 700.00	$ 1,400.00
Android phone	2	$ 550.00	$ 1,100.00
Ipad	1	$ 500.00	$ 500.00
Android Tablet	1	$ 400.00	$ 400.00
Total Hardware Cost			$ 21,400.00

The above table specifies the hardware costs required. A total of 12 laptops with the cost of $1,500 each, amounting to $18,000. Additionally, two Apple phones and two Android phones costing $700 and $550 each, respectively, totalling up to $2,500. The table also includes one iPad for $500 and one Android tablet for $400. Altogether, the total hardware investment sums up to $21,400.

Software Pricing Breakdown:

Tool	Usage	Cost (Estimated)	Final Cost	Remarks
React.js	Frontend framework for building the UI	Free (Open Source)	$ -	Free to use
Node.js	Backend runtime for running JavaScript code	Free (Open Source)	$ -	Free to use
Firebase Services (Firebase	NoSQL cloud database	Free (50K reads, 20K writes per day), then pay-as-you-go	$ 1,800.00	Will be using the blaze plan

`

Database, Cloud Storage)				
Selenium	Automated UI testing	Free (Open Source)	$ -	Free to use
Jest	JavaScript testing framework for unit tests	Free (Open Source)	$ -	Free to use
Azure App Service	Hosting React app on Azure	Free (Basic Plan), then ~$13/month for Standard	$ 1,200.00	Will be using the standard plan
Azure DevOps	CI/CD alternative to Jenkins	Free (Basic), $6/user for advanced plans	$ 72.00	Opting for the $6/month plan
GitHub	Code repository and version control	Free (Public repo), $4/user for private repos (Pro Plan)	$ -	Free to use
Azure Monitor	Performance monitoring	Free (Basic), then pay-as-you-go	$ 240.00	Will opt a $20/month plan since we will have small–medium workload
Postman	API testing and debugging	Free (Basic), then $12/month for Pro	$ -	Free to use
Jira	Project management & issue tracking	Free (Up to 10 users), then ~$8/user/month	$ -	Free to use
Microsoft Teams	Team collaboration & communication	Free (Basic), then ~$4/user/month for Teams Essentials	$ -	Free to use
VS Code	Code editor for development	Free (VS Code), WebStorm ($5/month for students)	$ -	Free to use

The Tool Cost Breakdown table provides an overview of the technologies and services used in the project, along with their respective costs and purposes. Most of the tools, such as React.js, Node.js, Selenium, Jest, GitHub, Postman, Jira, Microsoft Teams, and VS Code, are free and open-source, making them cost-effective choices for development, testing, collaboration, and project management. The major costs are associated with Firebase Services at $1,800, Azure App Service at $1,200, Azure DevOps at $72, and Azure Monitor at $240, totalling $3,312.00. These expenses reflect the selection of advanced cloud services to support backend hosting, deployment, and monitoring under scalable plans.

Miscellaneous cost breakdown:

Task ID	Task Description	Misc. Cost (USD)	Remarks
1.1.4	Conduct student and faculty surveys to identify event management pain points and user needs.	$100	Allocated for survey materials and incentives. Covers printing flyers or survey forms and small rewards for participants (e.g. $5–$10 gift cards) to boost response rates. This encourages sufficient feedback from students/faculty during the feasibility study phase.
1.2.2	Conduct project kickoff meeting with stakeholders (Northeastern University and students) to align expectations.	$150	Budget for a stakeholder meeting. Covers light refreshments and snacks for attendees and any printed handouts or meeting materials. Ensures a welcoming environment and engages student and university stakeholders from the start. (Venue is on campus, so no rental cost.)

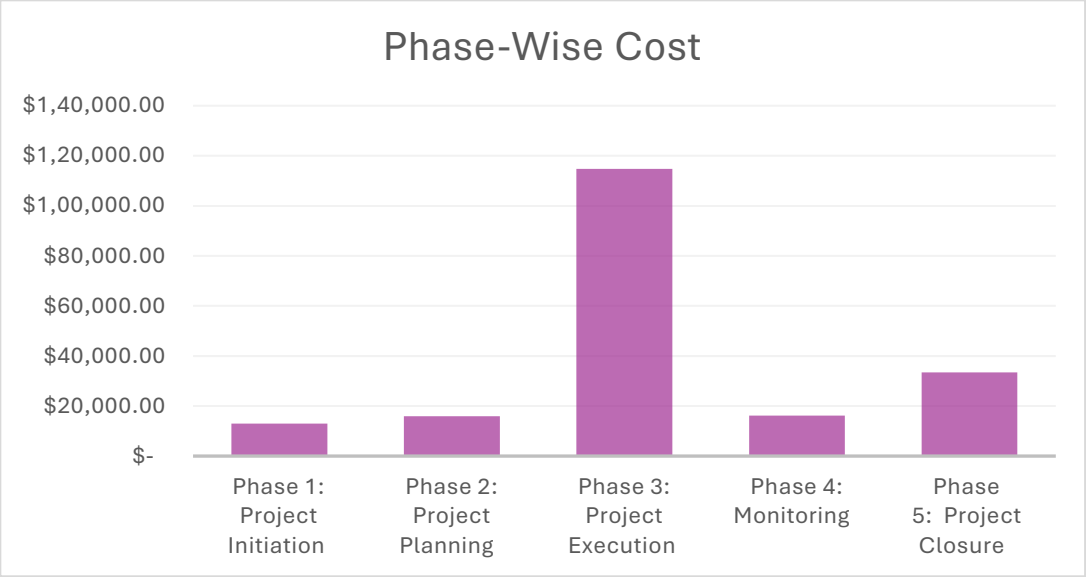
`

3.1.1.2	Design high-fidelity mock-ups using Figma and Adobe XD.	$50	Minor software tool cost for UI/UX design. The team may use Figma's Professional plan (~$15–$20 per editor/month) or an Adobe Creative Cloud subscription for XD. This cost covers 1–2 months of subscription needed to create high-fidelity prototypes with advanced features beyond free tiers.
3.1.5.2	Conduct usability testing with students and event admins.	$100	Usability testing sessions budget. Provides refreshments for test participants and small thank-you gifts (e.g. campus café vouchers) to encourage participation. Also covers any printing of test scenarios or feedback forms. Ensures real users (students, event admins) are motivated to give useful feedback.
3.2.4.3	End to End Testing	$50	Cross-platform/device testing expenses. Allocated for accessing a testing service or devices to simulate end-to-end usage on multiple platforms. For example, a one-month BrowserStack "Desktop & Mobile" subscription (~$39) allows testing the mobile app on various real devices, ensuring compatibility.
3.3.2.1	Load Testing to understand if system can function properly if the users increase.	$200	Cloud resource costs for performance testing (load, stress, scalability). Covers short-term usage of cloud servers or a load-testing service to simulate high user traffic. For instance, scaling up Azure instances or using a load test tool may incur fees to generate heavy load over several test runs.
3.3.4.1	Pilot Testing with Student & Admin Groups	$150	Misc. costs for UAT/pilot sessions with real users. Covers hosting a small event or focus group for student and admin testers – e.g. pizza or refreshments and swag (T-shirts or gift cards) as appreciation. This encourages participation and thorough testing in a real-world scenario before full launch.
4.1.1.2	Deploy backend services	$450	Cloud hosting costs for backend deployment on Azure. Covers approximately 6 months of running the Node.js backend in the cloud. (Azure App Service in a basic/standard tier costs roughly $50–$75 per month
4.1.2	Release mobile application on Google Play Store & Apple App Store	$125	Developer account fees for publishing the app. Google Play Console requires a one-time $25 registration. Apple's App Store program costs $99 per year. These total covers enrolling in both programs so the team can publish the Android and iOS versions of the event app.
4.1.3.1	Install and configure Jenkins server	$80	Server resources for CI/CD pipeline. Jenkins software is free, but this covers a small cloud VM or container to host the Jenkins server. For example, an Azure VM (Standard B1s) costs about $8–$10 per month; budgeting for ~8–9 months ensures the CI/CD runs throughout development and release.
4.2.1	Create step-by-step user guides tailored for different roles (students, event organizers, university admins).	$75	Printing and material costs for user documentation. Covers designing and printing a limited number of user guide booklets or quick start leaflets for each role. While guides will also be available digitally, having some printed copies for training sessions and key stakeholders (admins/organizers) is useful.

`

4.2.2	Conduct virtual training sessions via Zoom for event organizers and university admins.	$15	Zoom Pro subscription for one month to host training sessions without time limits. The Pro plan is about $14.99/month. This ensures that virtual training workshops for club organizers and staff can run longer than 40 minutes and include all features (recording, Q&A) needed for effective onboarding.
5.1.3.1	Conduct periodic surveys to gather feedback from students and administrators.	$100	Post-launch feedback collection budget. Covers incentives for survey respondents (similar to initial survey) or a subscription to a premium survey tool if needed. Small gift cards or raffle prizes encourage users to provide feedback on the live system, improving response rates for continuous improvement efforts.
5.2.1	Conduct project retrospective to discuss successes and challenges.	$100	Team wrap-up meeting expenses. Allocated for a modest project closure celebration or retrospective meetings with the project team. Covers refreshments or a team lunch as a thank-you for the members' work, creating a comfortable setting to reflect on lessons learned and document best practices for the future.
5.2.2	Prepare a final project report summarizing key learnings and outcomes.	$50	Printing and binding the final report for stakeholders. While the report will be delivered electronically, this covers producing a few high-quality printed copies (colour prints, binding) for the department or sponsor archives. It ensures the university has a tangible record of the project outcomes and learnings.

Table: Phase-Wise Cost



`

Phase 1: Research & Requirements

Allocated Budget: $12,930 (For detailed budget, refer to APPENDIX D: Budget Justification)
This foundational phase involves understanding the unique needs of all stakeholders involved in campus event management.

Key Activities:
Stakeholder Interviews: Engage with a broad set of stakeholders including students, faculty, administrative staff, and event organizers to gather qualitative insights into current challenges and expectations.
System Analysis: Assess existing tools or platforms in use to identify integration points, functional gaps, and user experience issues.
Requirements Gathering: Compile a comprehensive list of both functional requirements like event creation, RSVP, calendar syncing and technical requirements like compatibility with Single Sign-On, database structure.
Documentation: Consolidate all findings into a clearly defined Requirements Specification Document that will guide the design and development phases.

Deliverables:
- Finalized Requirements Specification Document
- Initial Project Plan and Timeline
- Stakeholder Feedback Summary

Phase 2: Design & Prototyping

Allocated Budget: $15,920 (For detailed budget, refer to APPENDIX D: Budget Justification).
This phase focuses on translating the requirements into intuitive and engaging user experiences that meet stakeholder expectations.

Key Activities:
Wireframes & User Flows: Create low-fidelity wireframes and map out user flows for each system interaction like RSVP for events, creating new listings.
Prototypes: Develop detailed, interactive UI/UX prototypes for mobile interfaces.
Feedback & Iteration: Conduct sessions with key stakeholders to gather feedback and validate usability before finalizing designs.

Deliverables:
- Stakeholder-Approved UI/UX Designs
- Clickable Prototypes for Demonstration
- Summary Report of Feedback and Iteration Outcomes

Phase 3: Development & Integration

Allocated Budget: $ 114,830 (For detailed budget, refer to APPENDIX D: Budget Justification)
This is the most resource-intensive phase, where design transforms into a functional, integrated application.

Key Activities:
Front-End and Back-End Development: Build responsive interfaces for mobile platforms, while establishing robust back-end services to manage events, users, and data.
Database and Security Setup: Define database schemas, implement encryption protocols, and integrate secure authentication systems like Single Sign-On.
Feature Development: Implement core modules such as event RSVP, dynamic calendar views, push/email notifications.
System Integration: Ensure smooth interaction with other university systems, such as student portals and communication tools.

Deliverables:
- Functional Application Modules
- Integrated System with Security and SSO
- Development Documentation

`

Phase 4: Testing & Quality Assurance

Allocated Budget: $16,145 (For detailed budget, refer to APPENDIX D: Budget Justification
Before launch, the system must be rigorously tested to ensure stability, security, and usability.

Key Activities:
Comprehensive Testing: Conduct unit tests, system integration tests, and performance/load testing to validate code quality and resilience under high usage.
User Acceptance Testing (UAT): Roll out a pilot test with a small group of end-users to simulate real-world usage and gather final feedback.
Bug Resolution & Compliance: Address identified issues, ensure adherence to security standards like FERPA compliance, and document test results.

Deliverables:
- QA-Approved Release
- Testing Reports & Bug Fix Logs
- UAT Feedback Summary

Phase 5: Launch & Maintenance

Allocated Budget: $33,490 (For detailed budget, refer to APPENDIX D: Budget Justification)
The final phase ensures smooth deployment, onboarding, and long-term support for the Event Management System.

Key Activities:
System Deployment: Launch the fully functional system in the production environment with necessary infrastructure scaling and monitoring tools in place.
Training & Onboarding: Provide training sessions to familiarize users with the platform's features.
Ongoing Maintenance: Monitor system performance, resolve post-launch issues, and prepare for future enhancements and periodic updates.

Deliverables:
- Live Event Management System
- User Training & Support Materials
- Maintenance

By following this structured five-phase plan, Northeastern University can implement an Event Management System that is financially sound and thoroughly tested. Continuous monitoring and utilization of funds through each phase will ensure a smooth deployment and long-term sustainability of the system.

8.0 TEAM CREDENTIALS

Team Member	Contribution
Anand Girish, Aditya	Aditya Girish Anand is a graduate student pursuing his master's in engineering management at Northeastern University. He has 1.5 years of experience in Business Development at an event tech platform named SortMyScene. He has collaborated with cross-functional teams across different sectors. In this project, Aditya contributed to the Introduction and References section by defining why, what and who this project is applicable to. He also played a role in developing and editing the Work Breakdown Structure for front-end development. He completed the Stakeholder's section and helped in Resource Allocation. Additionally, he made sure that the presentation and the handouts were printed on time.
Antony, Samson	Samson is a master's student in Engineering Management at Northeastern University with a background in Supply Chain and Data Analytics. He's worked with cross-functional teams throughout his career, putting his interdisciplinary skills to good use.

`

	In this project, Samson focused on defining the purpose and scope while aligning business and consumer goals. He put together schedules and Gantt charts, handled financial planning, worked on resource allocation, and built a risk register to track and visualize critical risks. He also helped create the Work Breakdown Structure (WBS), kept communication smooth across the team, and took the lead on drafting and formatting the final proposal to make sure everything was clear and well-organized.
Bhilave, Shruti	Shruti is a graduate student pursuing a master's in engineering management. She worked as a Software Developer II at Walmart Global Tech for 2.5 years after completing her undergraduate degree in B.Tech Computer Engineering. Drawing on her strong technical skills, critical thinking and business skills she took the lead in developing the Work Breakdown Structure (WBS) along with the implementation plan, execution plan and technical overview (Figure 1,2 and 4). She played a key role in defining project planning, project execution and project monitoring phase. She helped in resource allocation and identifying the stakeholders. In addition to managing coordination and communication across team members, she prepared the final project presentation for her part, proactively developed both a functional demo of the application and a comprehensive handout, ensuring all key deliverables and objectives were clearly articulated.
Dave, Shreya	Shreya Dave is a graduate student pursuing her master's in engineering management with a strong foundation in digital product management. She has extensive experience in developing user-centric applications, having previously worked on New Product Introduction (NPI) projects that optimized data accuracy and operational efficiency. In this project, Shreya led the development of the Work Breakdown Structure (WBS), drafting the implementation plan, allocating the resources, categorization of the risks for Event Management System (EMS). She has also crafted the content strategy for the project report, presentation and handouts and conceptualized the design elements. She played a key role in defining the project scope, ensuring clear deliverables, and allocating responsibilities by distributing tasks among team members for UI/UX design, backend and frontend development, and testing.
George Lal, Alan	Alan George Lal is a Mechanical Engineer pursuing his master's in engineering management at Northeastern University. He has 1 year of experience as a Project Associate at the Indian Institute of Science. Throughout his career, he has worked in cross-functional teams where he has had to use his multidisciplinary skills. In this project, Alan contributed to defining the introduction and created the table of contents. And worked on the deployment phase, formatting, and the Work Breakdown Structure (WBS). Additionally, he also worked on creating the Financial Planning and Budget overview.
Kalghatgi, Arnavi	Arnavi Kalghatgi is a Mechanical Engineer pursuing her master's in engineering management at Northeastern University. She has prior experience of 2.8 years as a Product Design Engineer in an automotive industry. She hones skills in cross-functional collaborations, product design, development, and quality validation alongside release management. She is passionate about driving engineering solutions coupled with innovation and aspires to delve into Product Management. In this project, Arnavi has worked on the RACI matrix creation – accounting tasks to specific roles, defining the introduction, considering various aspects on why, how, what and who the application will address and contributed to the planning phase of work breakdown structure. Additionally, worked on the Monitoring Phase of Execution Plan,

`

	Letter of transmittal, Executive Summary, Project Handout creation and Project presentation
Keluskar, Aarya	Aarya Keluskar is a graduate student pursuing a master's in engineering management at Northeastern University. She has prior experience as Software Developer at Mercedes-Benz Research and Development India for 1.5 years prior to which she has completed her undergraduate degree in Computer Science & Engineering. In this project, Aarya has worked on the Business and Consumer objectives and also contributed to the Work Breakdown Structure focusing on the monitoring phase and closure phase. Additionally, she has worked on Technical overview focusing on Admin workflow along with the workflow diagram, front-end development and backend development. She has worked on compliance, the control table in the execution phase and has worked on the financial plan including the phase-wise distribution of labour costs. She has worked on the creation of Handout and presentation.

9.0 REFERENCES

[1] Uptech Team. App development for iOS and Android: A complete guide. Uptech. Retrieved from https://www.uptech.team/blog/app-development-for-ios-and-android

[2] Innvonix Tech Solutions. (2023, May 12). Why choose React Native for mobile app development? Medium. Retrieved from https://medium.com/@innvonixtechsolutions/why-choose-react-native-for-mobile-app-development-31e8ba23af5e

[3] Saviant Consulting. 7 benefits of Azure web apps development. Saviant Consulting. Retrieved from https://www.saviantconsulting.com/blog/7-benefits-azure-web-apps-development.aspx

[4] *Boston, MA Tech + Startup salaries*. Boston, MA Tech Salaries 2025 | Built In. (n.d.). https://builtin.com/salaries/location/boston-ma

[5] USA salary database: Compare & estimate salary for any Job. (n.d.). https://www.ziprecruiter.com/Salaries

[6] Company salaries | glassdoor. (n.d.-a). https://www.glassdoor.ie/Salaries/index.htm

`

APPENDIX A:

	TASKS
1	**Phase 1: Project Initiation**
2	**1.1 Feasibility Study and Preliminary Research**
3	1.1.1 Conduct a technical feasibility study to assess system requirements (scalability, integration feasibility, security measures).
4	1.1.2 Conduct a financial feasibility study to estimate development costs and ROI
5	1.1.3 Evaluate operational feasibility by verifying usability within university systems (authentication, event management integration)
6	1.1.4 Conduct student and faculty surveys to identify event management pain points and user needs.
7	**1.2 Project Kickoff & Scope Definition**
8	1.1.1 Define scope, purpose, and success criteria of the project
9	1.1.2 Conduct project kickoff meeting with stakeholders (Northeastern university and students) to align expectations
10	1.1.3 Establish Project communication channels and collaboration methods
11	1.1.4 Develop a Stakeholder engagement plan
12	**1.3 Project Team Formation**
13	1.3.1 Define project roles and responsibilities (UI/UX, development, testing, compliance)
14	1.3.2 Assign team members to specific project tasks based on expertise
15	1.3.4 Set up regular project meetings for progress updates and issue resolution
16	**Phase 2: Project Planning**
17	2.1 Requirements Gathering
18	2.1.1 Define functional requirements (event creation, RSVP, notifications, admin roles).
19	2.1.2 Identify non-functional requirements (performance, security, scalability).
20	2.1.3 Establish key performance indicators (KPIs - System Performance & Uptime, Scalability & Load Handling, Notification Effectiveness) for project success.
21	**2.2 System Architecture & Design Planning**
22	2.2.1 Finalize technology stack: React Native (frontend), Node.js (backend), Firebase (database)
23	2.2.2 Define system components, including database structure and API design
24	2.2.3 Plan system integrations (Google Calendar, university SSO authentication).
25	**2.3 Scheduling and Resource Allocation**
26	2.3.1 Develop a timeline with milestones and deadlines
27	2.3.2 Identify and allocate human resources (functional manager, developers, UI/UX designers, testers, compliance)
28	2.3.3 Post job descriptions and project requirements on NU Works Portal of Northeastern University to recruit developers and NU Admin and Test Lead
29	2.3.4 Engage with Northeastern University faculty to identify students with relevant expertise for internship
30	2.3.5 Conduct interviews and skill assessments to select candidates for different project roles.
31	**2.4 Collaboration Setup**
32	2.4.1 Provide access to Jira, Slack and development environments.

`

33	2.4.2 Conduct an initial onboarding meeting to align the team with project scope, timelines, and responsibilities
34	2.4.3 Assign supervisors to track performance and guide project interns
35	**2.5 Risk Assessment & Mitigation**
36	2.5.1 Identify project risks, including technical issues, adoption resistance, and security threats
37	2.5.2 Check for Northeastern University's IT security framework to ensure data encryption, secure authentication, and access control
38	2.5.3 Develop contingency plans for high-risk scenarios
39	2.5.4 Allocate risk ownership and implement monitoring strategies
40	**2.6 Compliance Check**
41	2.6.1 Check for compliance - WCAG for accessibility and FERPA for student data
42	2.6.2 Check for privacy policy and user consent integration
43	2.6.3 Schedule a review session with NU's IT and Legal departments to validate cloud usage and no unauthorized third-party access
44	**Phase 3: Project Execution**
45	**3.1 Design**
46	**3.1.1 Develop UI/UX wireframes**
47	3.1.1.1. Create low-fidelity wireframes for all key screens
48	3.1.1.2 Design high-fidelity mock-ups using Figma and Adobe XD
49	**3.1.2. Define User Flow (User and Admin Flow)**
50	3.1.2.1 Implement user authenticate via SSO (University ID)
51	3.1.2.2 Design Feature Event Discovery
52	3.1.2.2.1 Browse events via categories, search, or featured events. Filter by date, category, or club association
53	3.1.2.3 Design Features Event Details and RSVP
54	3.1.2.3.1 View event details (time, location, organizer, description). RSVP with confirmation and calendar sync
55	3.1.2.3.2 Design Features Notifications & Reminders
56	3.1.2.3.2.1 Receive push/email notifications for upcoming events. Get reminders for RSVP'd events
57	**3.1.3 Define Admin Flow**
58	3.1.3.1 Create Admin Dashboard Access (club organizers vs university admin)
59	3.1.3.2 Event Creation & Management feature
60	3.1.3.2.1 Design access for organizers to create/edit/delete events
61	3.1.3.2.2 Design access for organizers to upload event posters, descriptions, and participant limits.
62	3.1.3.3 RSVP & Participant Management
63	3.1.3.3.1 Define feature to view even registrations
64	3.1.3.3.2 Design a feature to approve/reject requests for limited seat events
65	**3.1.4 Define Database Schema and Backend API Structure**
66	3.1.4.1 Structure user profiles, event details, RSVP records, and notifications.
67	3.1.4.2 Design APIs for event creation, retrieval, and user authentication

`

68	3.1.4.3 Implement Security protocols (OAuth, JWT authentication, role-based access control)
69	**3.1.5 Prototype Development**
70	3.1.5.1 Develop an interactive prototype for evaluation
71	3.1.5.2 Conduct usability testing with students and event admins
72	3.1.5.3 Feedback collection and implementation for better user experience
73	**3.2 Development**
74	**3.2.1 Backend Development**
75	**3.2.1.1 Backend Server Setup**
76	3.2.1.1.1 Set up Node.js with Express.js as the backend framework
77	3.2.1.1.2 Configure Firebase as the NoSQL database for event and user storage
78	3.2.1.1.3 Implement OAuth & JWT-based authentication for secure login
79	**3.2.1.2 User Authentication & Role-Based Access Control (RBAC)**
80	3.2.1.2.1 Integrate Northeastern University SSO (Single Sign-On) authentication
81	3.2.1.2.2 Set up multi-role access control (student, organizer, admin).
82	3.2.1.2.3 Encrypt user credentials and session data.
83	**3.2.1.3 Event Management API Development**
84	3.2.1.3.1 Develop CRUD APIs for event creation, updating, retrieval, and deletion
85	3.2.1.3.2 Implement event categorization, RSVP tracking, and participant limits
86	3.2.1.3.3 Integrate event metadata (location, time, organizers, images)
87	**3.2.1.4 Notification & Reminder System**
88	3.2.1.4.1 Develop real-time notifications for event updates
89	3.2.1.4.2 Implement automated reminders via push notifications and email.
90	3.2.1.4.3 Integrate Firebase Cloud Messaging (FCM) for mobile alerts
91	**3.2.1.5 Development Error Tracking System**
92	**3.2.1.5.1 Real-Time Error Tracking**
93	3.2.1.5.1.1 Implement Google Cloud Logging to monitor API errors and backend failures.
94	3.2.1.5.1.2 Use Firebase Crashlytics for real-time mobile app crash reporting.
95	3.2.1.5.1.3 Integrate Sentry or LogRocket for frontend and session tracking.
96	**3.2.1.5.2 Automated Alerts & Notifications**
97	3.2.1.5.2.1 Configure Google Cloud Monitoring to send real-time alerts for critical errors.
98	3.2.1.5.2.2 Integrate notifications with Slack, and Email for quick response.
99	3.2.1.5.2.3 Categorize errors by severity levels (Critical, High, Medium, Low) for efficient prioritization.
100	**3.2.1.5.3 Logging & Debugging**
101	3.2.1.5.3.1 Enable structured logging in Google Cloud Logging for API requests, database queries, and function executions.
102	3.2.1.5.3.2 Utilize Firestore query profiling to optimize database performance.
103	3.2.1.5.3.3 Set up Cloud Trace & Cloud Profiler to analyze system bottlenecks.
104	**3.2.1.5.4 Incident Response & Recovery**

`

105	3.2.1.5.4.1 Establish a rollback strategy for failed deployments.
106	3.2.1.5.4.2 Assign an on-call support team to handle production issues.
107	3.2.1.5.4.3 Implement post-mortem analysis for major incidents and apply improvements.
108	**3.2.1.5.5 Backup & Disaster Recovery**
109	3.2.1.5.5.1 Automate daily backups for Firestore and Cloud Storage.
110	3.2.1.5.5.2 Ensure redundant cloud deployments to minimize downtime risks.
111	3.2.1.5.5.3 Develop a disaster recovery plan for critical data loss scenarios.
112	**3.2.2 Frontend Development**
113	3.2.2.1 Build React Native interface for event browsing, RSVP, and dashboards
114	3.2.2.2 Implement a responsive UI/UX design for smooth navigation (iOS and Android)
115	3.2.2.2.1 User Flow Implementation for student and attendee
116	3.2.2.2.1.1 Implement login and authentication
117	3.2.2.2.1.2 Implement event dashboard
118	3.2.2.2.1.3 Implement event discovery, RSVP, notifications, and calendar integration
119	3.2.2.2.2 Admin Flow Implementation for organizers and university staff (login, authentication, admin dashboard, event creation, editing, analytics and reports)
120	3.2.2.2.2.1 Implement login and authentication
121	3.2.2.2.2.2 Implement admin dashboard
122	3.2.2.2.2.3 Implement event creation, editing, analytics and reports
123	**3.2.3 Security and Performance Enhancement**
124	3.2.3.1 Implement encryption, authentication security, and access control policies
125	3.2.3.1.1 Implement AES-256 encryption to securely store sensitive data in Firebase Firestore and Cloud Storage.
126	3.2.3.1.2 Implement Role-Based Access Control (RBAC)
127	3.2.3.1.3 Implement Multi-Factor Authentication (MFA): Require additional verification for admin logins
128	3.2.3.2 Optimize database queries, API calls, and caching mechanisms for performance
129	**3.2.4 Integration & Unit Testing**
130	3.2.4.1 API Testing for backend and frontend synchronization
131	3.2.4.2 UI Testing
132	3.2.4.2.1 Conduct Backend API testing using Postman
133	3.2.4.2.2 Perform component-level testing for UI elements
134	3.2.4.2.3 Validate authentication security, event creation, and RSVP functionalities
135	3.2.4.3 End to End Testing
136	**3.3 System Testing and Feedback Implementation**
137	**3.3.1 Functional Testing**
138	3.3.1.1 Unit Testing for all the features we are implementing in our app
139	3.3.1.2 Integration Testing to check if frontend and backend is working as expected
140	3.3.1.3 API Testing for CRUD operations
141	**3.3.2 Performance & Load Testing**

`

142	3.3.2.1 Load Testing to understand if system can function properly if the users increase
143	3.3.2.2 Stress Testing on a particular event which has many registrations
144	3.3.2.3 Scalability Testing to check system's ability to scale up or down as the workload increases or decreases
145	**3.3.3 Security Testing**
146	3.3.3.1 Authentication & Authorization Testing for SSO
147	3.3.3.2 Data Encryption & Compliance Testing for user personal data
148	3.3.3.3 Penetration Testing to check systems compatibility with external malicious attacks
149	**3.3.4 User Acceptance Testing (UAT)**
150	3.3.4.1 Pilot Testing with Student & Admin Groups
151	3.3.4.2 Bug Fixes & Optimizations
152	**3.3.5 Regression Testing**
153	3.3.5.1 Re-testing Core Functionalities
154	3.3.5.2 Performance Re-Evaluation of the entire app
155	**Phase 4: Monitoring**
156	**4.1 Deployment & Go-Live**
157	**4.1.1 Deploy backend services to Microsoft Azure for cloud hosting**
158	4.1.1.1 Set up Microsoft Azure account and configure resources
159	4.1.1.2 Deploy backend services
160	4.1.1.3 Configure security settings
161	4.1.1.4 Test the backend services
162	**4.1.2 Release mobile application on Google Play Store & Apple App Store**
163	**4.1.3 Set up a CI/CD pipeline with Jenkins for automated updates**
164	4.1.3.1 Install and configure Jenkins server
165	4.1.3.2 Test CI/CD pipeline for automated updates
166	4.1.3.3 Monitor pipeline performance and troubleshoot issues
167	**4.2 User Training & Documentation**
168	4.2.1 Create step-by-step user guides tailored for different roles (students, event organizers, university admins).
169	4.2.2 Conduct virtual training sessions via Zoom for event organizers and university admins.
170	**Phase 5: Project Closure**
171	**5.1 Post-Launch Support**
172	5.1.1 Monitor application stability, responsiveness, and crash reports
173	5.1.2 Track bug reports using JIRA
174	5.1.3 Collect user feedback for future enhancements and updates
175	5.1.3.1 Conduct periodic surveys to gather feedback from students and administrators.
176	5.1.3.2 Organize feedback review meetings with stakeholders for prioritization.
177	5.1.3.3 Implement continuous improvements based on collected feedback.
178	**5.2 Project Review & Documentation**

`

179	5.2.1 Conduct project retrospective to discuss successes and challenges
180	5.2.2 Prepare a final project report summarizing key learnings and outcomes
181	5.2.3 Archive all project documentation for future reference and scalability
182	**5.3 Project Handover to University**
183	**5.3.1 Transition Plan**
184	5.3.1.1 Define a structured handover process with key milestones.
185	5.3.1.2 Assign responsibilities for maintaining and supporting the system.
186	**5.3.2 Knowledge Transfer**
187	5.3.2.1 Conduct detailed training sessions for the university IT team
188	5.3.2.2 Provide in-depth documentation, including system architecture, APIs, and maintenance procedures.
189	**5.3.3 Data & Access Handover**
190	5.3.3.1 Transfer ownership of cloud resources (Firebase, Azure, etc.).
191	5.3.3.2 Revoke old access credentials and issue new ones.
192	**5.3.4 Final Handover Report**
193	5.3.4.1 Compile a summary of system
194	5.3.4.2 Ensure all stakeholders acknowledge the successful transition
195	**5.4 Project Closure Meeting**

`

APPENDIX B: SCHEDULE

Task Name	Duration	Start	Finish
NEU EMS	**224 days**	**Mon 3/17/25**	**Thu 1/22/26**
Phase 1: Project Initiation	**10 days**	**Mon 3/17/25**	**Fri 3/28/25**
Feasibility Study	**3 days**	**Mon 3/17/25**	**Wed 3/19/25**
Conduct a technical feasibility study to assess system requirements (scalability, integration feasibility, security measures).	3 days	Mon 3/17/25	Wed 3/19/25
Conduct a financial feasibility study to estimate development costs, ROI, and potential funding sources	3 days	Mon 3/17/25	Wed 3/19/25
Evaluate operational feasibility by verifying usability within university systems (authentication, event management integration)	3 days	Mon 3/17/25	Wed 3/19/25
Conduct student and faculty surveys to identify event management pain points and user needs.	3 days	Mon 3/17/25	Wed 3/19/25
Project Kickoff & Scope Definition	**5 days**	**Thu 3/20/25**	**Wed 3/26/25**
Define scope, purpose, and success criteria of the project	2 days	Thu 3/20/25	Fri 3/21/25
Conduct project kickoff meeting with stakeholders (Northeastern University and students) to align expectations	1 day	Mon 3/24/25	Mon 3/24/25
Establish communication channels and collaboration methods	1 day	Tue 3/25/25	Tue 3/25/25
Develop a Stakeholder engagement plan	1 day	Wed 3/26/25	Wed 3/26/25
Project Formation	**2 days**	**Thu 3/27/25**	**Fri 3/28/25**
Define project roles and responsibilities (UI/UX, development, testing, compliance)	1 day	Thu 3/27/25	Thu 3/27/25
Set up regular project meetings for progress updates and issue resolution	1 day	Fri 3/28/25	Fri 3/28/25
Assign team members to specific project tasks based on expertise	1 day	Fri 3/28/25	Fri 3/28/25
Phase 2: Project Planning	**16 days**	**Mon 3/31/25**	**Mon 4/21/25**
Requirements Gathering	**2 days**	**Mon 3/31/25**	**Tue 4/1/25**
Define functional requirements (event creation, RSVP, notifications, admin roles)	2 days	Mon 3/31/25	Tue 4/1/25

`

Task Name	Duration	Start	Finish
Identify non-functional requirements (performance, security, scalability)	1 day	Mon 3/31/25	Mon 3/31/25
Establish key performance indicators (KPIs) for project success	2 days	Mon 3/31/25	Tue 4/1/25
System Architecture & Design Planning	**3 days**	**Wed 4/2/25**	**Fri 4/4/25**
Finalize technology stack: React Native (frontend), Node.js (backend), Firebase (database)	2 days	Wed 4/2/25	Thu 4/3/25
Define system components, including database structure and API design	2 days	Wed 4/2/25	Thu 4/3/25
Plan system integrations (Google Calendar, university SSO authentication)	1 day	Fri 4/4/25	Fri 4/4/25
Scheduling and Resource Allocation	**5 days**	**Mon 4/7/25**	**Fri 4/11/25**
Develop a timeline with milestones and deadlines	1 day	Mon 4/7/25	Mon 4/7/25
Identify and allocate human resources (functional manager, developers, UI/UX designers, testers, compliance)	1 day	Mon 4/7/25	Mon 4/7/25
Post job descriptions and project requirements on NU Works Portal of Northeastern University to recruit functional manager, developers, UI/UX designers, compliance	1 day	Tue 4/8/25	Tue 4/8/25
Engage with Northeastern University faculty to identify students with relevant expertise for internship	1 day	Tue 4/8/25	Tue 4/8/25
Conduct interviews and skill assessments to select candidates for different project roles.	4 days	Tue 4/8/25	Fri 4/11/25
Collaboration Setup	**1 day**	**Mon 4/14/25**	**Mon 4/14/25**
Provide access to Jira, Slack and development environments.	1 day	Mon 4/14/25	Mon 4/14/25
Conduct an initial onboarding meeting to align the team with project scope, timelines, and responsibilities	1 day	Mon 4/14/25	Mon 4/14/25
Assign supervisors to track performance and guide project interns	1 day	Mon 4/14/25	Mon 4/14/25
Risk Assessment & Mitigation	**3 days**	**Tue 4/15/25**	**Thu 4/17/25**
Identify project risks, including technical issues, adoption resistance, and security threats	1 day	Tue 4/15/25	Tue 4/15/25

`

Task Name	Duration	Start	Finish
Check for Northeastern University's IT security framework to ensure data encryption, secure authentication, and access control	1 day	Wed 4/16/25	Wed 4/16/25
Develop contingency plans for high-risk scenarios	1 day	Wed 4/16/25	Wed 4/16/25
Allocate risk ownership and implement monitoring strategies	1 day	Thu 4/17/25	Thu 4/17/25
Compliance Planning	**2 days**	**Fri 4/18/25**	**Mon 4/21/25**
Check for accessibility compliance - WCAG and FERPA for student data	1 day	Fri 4/18/25	Fri 4/18/25
Check for privacy policy and user consent integration	1 day	Fri 4/18/25	Fri 4/18/25
Schedule a review session with NU's IT and Legal departments to validate cloud usage and no unauthorized third party access	1 day	Mon 4/21/25	Mon 4/21/25
Phase 3: Project Execution	**158 days**	**Tue 4/22/25**	**Thu 11/27/25**
Design	**41 days**	**Tue 4/22/25**	**Tue 6/17/25**
Develop UI/UX wireframes	**8 days**	**Tue 4/22/25**	**Thu 5/1/25**
Create low-fidelity wireframes for all key screens	3 days	Tue 4/22/25	Thu 4/24/25
Design high-fidelity mock-ups using Figma and Adobe XD	5 days	Fri 4/25/25	Thu 5/1/25
Define User Flow (User and Admin Flow)	**10 days**	**Fri 5/2/25**	**Thu 5/15/25**
Authenticate users via SSO (University ID)	2 days	Fri 5/2/25	Mon 5/5/25
Design Feature Event Discovery	**2 days**	**Tue 5/6/25**	**Wed 5/7/25**
Browse events via categories, search, or featured events. Filter by date, category, or club	2 days	Tue 5/6/25	Wed 5/7/25
Design Features Event Details and RSVP	**6 days**	**Thu 5/8/25**	**Thu 5/15/25**
View event details (time, location, organizer, description). RSVP with confirmation	3 days	Thu 5/8/25	Mon 5/12/25
Design Features Notifications & Reminders	**3 days**	**Tue 5/13/25**	**Thu 5/15/25**
Receive push/email notifications for upcoming events. Get reminders for RSVP'd events	3 days	Tue 5/13/25	Thu 5/15/25
Define Admin Flow	**5 days**	**Fri 5/16/25**	**Thu 5/22/25**
Create Admin Dashboard Access (club organizers vs university admin)	1 day	Fri 5/16/25	Fri 5/16/25

`

Task Name	Duration	Start	Finish
Event Creation & Management feature	**2 days**	**Mon 5/19/25**	**Tue 5/20/25**
Design access for organizers to create/edit/delete events	2 days	Mon 5/19/25	Tue 5/20/25
Design access for organizers to upload event posters, descriptions, and participant limits	2 days	Mon 5/19/25	Tue 5/20/25
RSVP & Participant Management	**2 days**	**Wed 5/21/25**	**Thu 5/22/25**
Define feature to view event registrations	2 days	Wed 5/21/25	Thu 5/22/25
Define Database Schema and Backend API Structure	**8 days**	**Fri 5/23/25**	**Tue 6/3/25**
Structure user profiles, event details, RSVP records, and notifications	3 days	Fri 5/23/25	Tue 5/27/25
Design APIs for event creation, retrieval, and user authentication	3 days	Wed 5/28/25	Fri 5/30/25
Implement Security protocols (OAuth, JWT authentication, role-based access control)	2 days	Mon 6/2/25	Tue 6/3/25
Prototype Development	**10 days**	**Wed 6/4/25**	**Tue 6/17/25**
Develop an interactive prototype for evaluation	4 days	Wed 6/4/25	Mon 6/9/25
Conduct usability testing with students and event admins	3 days	Tue 6/10/25	Thu 6/12/25
Feedback collection and implementation for better user experience	3 days	Fri 6/13/25	Tue 6/17/25
Development	**84 days**	**Wed 6/18/25**	**Mon 10/13/25**
Backend Development	**84 days**	**Wed 6/18/25**	**Mon 10/13/25**
Backend Server Setup	**5 days**	**Wed 6/18/25**	**Tue 6/24/25**
Set up Node.js with Express.js as the backend framework	2 days	Wed 6/18/25	Thu 6/19/25
Configure Firebase as the NoSQL database for event and user storage	2 days	Fri 6/20/25	Mon 6/23/25
Implement OAuth & JWT-based authentication for secure login	1 day	Tue 6/24/25	Tue 6/24/25
User Authentication & Role-Based Access Control (RBAC)	**4 days**	**Wed 6/25/25**	**Mon 6/30/25**
Implement University SSO (Single Sign-On) authentication	2 days	Wed 6/25/25	Thu 6/26/25

`

Task Name	Duration	Start	Finish
Set up multi-role access control (student, organizer, admin)	2 days	Wed 6/25/25	Thu 6/26/25
Encrypt user credentials and session data	2 days	Fri 6/27/25	Mon 6/30/25
Event Management API Development	**9 days**	**Tue 7/1/25**	**Fri 7/11/25**
Develop CRUD APIs for event creation, updating, retrieval, and deletion	3 days	Tue 7/1/25	Thu 7/3/25
Implement event categorization, RSVP tracking, and participant limits	3 days	Fri 7/4/25	Tue 7/8/25
Integrate event metadata (location, time, organizers, images)	3 days	Wed 7/9/25	Fri 7/11/25
Notification & Reminder System	**4 days**	**Mon 7/14/25**	**Thu 7/17/25**
Develop real-time notifications for event updates	2 days	Mon 7/14/25	Tue 7/15/25
Implement automated reminders via push notifications and email	2 days	Mon 7/14/25	Tue 7/15/25
Integrate Firebase Cloud Messaging (FCM) for mobile alerts	2 days	Wed 7/16/25	Thu 7/17/25
Development Error Tracking System	**20 days**	**Fri 7/18/25**	**Thu 8/14/25**
Real-Time Error Tracking	**8 days**	**Fri 7/18/25**	**Tue 7/29/25**
Implement Google Cloud Logging to monitor API errors and backend failures.	3 days	Fri 7/18/25	Tue 7/22/25
Use Firebase Crashlytics for real-time mobile app crash reporting.	2 days	Wed 7/23/25	Thu 7/24/25
Integrate Sentry or LogRocket for frontend and session tracking.	3 days	Fri 7/25/25	Tue 7/29/25
Automated Alerts & Notifications	**7 days**	**Fri 7/18/25**	**Mon 7/28/25**
Configure Google Cloud Monitoring to send real-time alerts for critical errors	3 days	Fri 7/18/25	Tue 7/22/25
Integrate notifications with Slack and Email for quick response	2 days	Wed 7/23/25	Thu 7/24/25
Categorize errors by severity levels (Critical, High, Medium, Low) for efficient prioritization	2 days	Fri 7/25/25	Mon 7/28/25
Logging & Debugging	**8 days**	**Tue 7/29/25**	**Thu 8/7/25**
Enable structured logging in Google Cloud Logging for API requests, database queries, and function executions	3 days	Tue 7/29/25	Thu 7/31/25

`

Task Name	Duration	Start	Finish
Utilize Firestore query profiling to optimize database performance	2 days	Fri 8/1/25	Mon 8/4/25
Set up Cloud Trace & Cloud Profiler to analyze system bottlenecks	3 days	Tue 8/5/25	Thu 8/7/25
Incident Response & Recovery	**7 days**	**Tue 7/29/25**	**Wed 8/6/25**
Establish a rollback strategy for failed deployments	3 days	Tue 7/29/25	Thu 7/31/25
Assign an on-call support team to handle production issues	2 days	Fri 8/1/25	Mon 8/4/25
Implement post-mortem analysis for major incidents and apply improvements	2 days	Tue 8/5/25	Wed 8/6/25
Backup & Disaster Recovery	**6 days**	**Thu 8/7/25**	**Thu 8/14/25**
Automate daily backups for Firestore and Cloud Storage	3 days	Thu 8/7/25	Mon 8/11/25
Ensure redundant cloud deployments to minimize downtime risks	3 days	Thu 8/7/25	Mon 8/11/25
Develop a disaster recovery plan for critical data loss scenarios	3 days	Tue 8/12/25	Thu 8/14/25
Frontend Development	**21 days**	**Fri 8/15/25**	**Fri 9/12/25**
Build React Native interface for event browsing, RSVP, and dashboards	7 days	Fri 8/15/25	Mon 8/25/25
Implement a responsive UI/UX design for smooth navigation (iOS and Android)	**14 days**	**Tue 8/26/25**	**Fri 9/12/25**
User Flow Implementation for student and attendee (login, authentication, dashboard, event discovery, RSVP, notifications, and calendar integration)	**6 days**	**Tue 8/26/25**	**Tue 9/2/25**
Implement login and authentication for student and attendee	2 days	Tue 8/26/25	Wed 8/27/25
Implement event dashboard	2 days	Thu 8/28/25	Fri 8/29/25
Implement event discovery, RSVP, notifications, and calendar integration	2 days	Mon 9/1/25	Tue 9/2/25
Admin Flow Implementation for organizers and university staff (login, authentication, admin dashboard, event creation, editing, analytics, and reports)	**8 days**	**Wed 9/3/25**	**Fri 9/12/25**

`

Task Name	Duration	Start	Finish
Implement login and authentication	2 days	Wed 9/3/25	Thu 9/4/25
Implement Admin Dashboard	3 days	Fri 9/5/25	Tue 9/9/25
Implement event creation, editing, analytics and reports	3 days	Wed 9/10/25	Fri 9/12/25
Security and Performance Enhancement	**8 days**	**Mon 9/15/25**	**Wed 9/24/25**
Implement encryption, authentication security, and access control policies	**5 days**	**Mon 9/15/25**	**Fri 9/19/25**
Implement AES-256 encryption to securely store sensitive data in Firebase Firestore and Cloud Storage.	2 days	Mon 9/15/25	Tue 9/16/25
Implement Role-Based Access Control (RBAC)	2 days	Wed 9/17/25	Thu 9/18/25
Implement Multi-Factor Authentication (MFA): Require additional verification for admin logins	1 day	Fri 9/19/25	Fri 9/19/25
Optimize database queries, API calls, and caching mechanisms for performance	**3 days**	**Mon 9/22/25**	**Wed 9/24/25**
Integration & Unit Testing	**13 days**	**Thu 9/25/25**	**Mon 10/13/25**
API Testing for backend and frontend synchronization	3 days	Thu 9/25/25	Mon 9/29/25
UI Testing	**5 days**	**Tue 9/30/25**	**Mon 10/6/25**
Conduct Backend API testing using Postman	2 days	Tue 9/30/25	Wed 10/1/25
Perform component-level testing for UI elements	2 days	Thu 10/2/25	Fri 10/3/25
Validate authentication security, event creation, and RSVP functionalities	1 day	Mon 10/6/25	Mon 10/6/25
End to End Testing	5 days	Tue 10/7/25	Mon 10/13/25
System Testing and Feedback Implementation	**33 days**	**Tue 10/14/25**	**Thu 11/27/25**
Functional Testing	**7 days**	**Tue 10/14/25**	**Wed 10/22/25**
Unit Testing	3 days	Tue 10/14/25	Thu 10/16/25
Integration Testing	2 days	Fri 10/17/25	Mon 10/20/25
API Testing	2 days	Tue 10/21/25	Wed 10/22/25
Performance & Load Testing	**6 days**	**Thu 10/23/25**	**Thu 10/30/25**
Load Testing	2 days	Thu 10/23/25	Fri 10/24/25
Stress Testing	2 days	Mon 10/27/25	Tue 10/28/25
Scalability Testing	2 days	Wed 10/29/25	Thu 10/30/25

`

Task Name	Duration	Start	Finish
Security Testing	**6 days**	**Fri 10/31/25**	**Fri 11/7/25**
Authentication & Authorization Testing	2 days	Fri 10/31/25	Mon 11/3/25
Data Encryption & Compliance Testing	2 days	Tue 11/4/25	Wed 11/5/25
Penetration Testing	2 days	Thu 11/6/25	Fri 11/7/25
User Acceptance Testing (UAT)	**9 days**	**Mon 11/10/2**	**Thu 11/20/25**
Pilot Testing with Student & Admin Groups	4 days	Mon 11/10/2	Thu 11/13/25
Bug Fixes & Optimizations	5 days	Fri 11/14/25	Thu 11/20/25
Regression Testing	**5 days**	**Fri 11/21/25**	**Thu 11/27/25**
Re-testing Core Functionalities	2 days	Fri 11/21/25	Mon 11/24/25
Performance Re-Evaluation	3 days	Tue 11/25/25	Thu 11/27/25
Phase 4: Monitoring	**20 days**	**Fri 11/28/25**	**Thu 12/25/25**
Deployment & Go-Live	**13 days**	**Fri 11/28/25**	**Tue 12/16/25**
Deploy backend services to Microsoft Azure for cloud hosting	**7 days**	**Fri 11/28/25**	**Mon 12/8/25**
Set up Microsoft Azure account and configure resources	1 day	Fri 11/28/25	Fri 11/28/25
Deploy backend services	2 days	Mon 12/1/25	Tue 12/2/25
Configure security settings	2 days	Wed 12/3/25	Thu 12/4/25
Test the backend services	2 days	Fri 12/5/25	Mon 12/8/25
Release mobile application on Google Play Store & Apple App Store	1 day	Tue 12/9/25	Tue 12/9/25
Set up a CI/CD pipeline with Azure DevOps for automated updates	**5 days**	**Wed 12/10/25**	**Tue 12/16/25**
Install and configure Azure DevOps server	2 days	Wed 12/10/2	Thu 12/11/25
Test CI/CD pipeline for automated updates	2 days	Fri 12/12/25	Mon 12/15/25
Monitor pipeline performance and troubleshoot issues	1 day	Tue 12/16/25	Tue 12/16/25
User Training & Documentation	**7 days**	**Wed 12/17/2**	**Thu 12/25/25**
Create step-by-step user guides tailored for different roles (students, event organizers, university admins)	3 days	Wed 12/17/25	Fri 12/19/25
Conduct virtual training sessions via Zoom for event organizers and university admins.	4 days	Mon 12/22/25	Thu 12/25/25

`

Task Name	Duration	Start	Finish
Phase 5: Project Closure	**20 days**	**Fri 12/26/25**	**Thu 1/22/26**
Post-Launch Support	**8 days**	**Fri 12/26/25**	**Tue 1/6/26**
Monitor application stability, responsiveness, and crash reports	5 days	Fri 12/26/25	Thu 1/1/26
Track bug reports using JIRA	3 days	Fri 12/26/25	Tue 12/30/25
Collect user feedback for future enhancements and updates	**5 days**	**Wed 12/31/25**	**Tue 1/6/26**
Conduct periodic surveys to gather feedback from students and administrators	5 days	Wed 12/31/25	Tue 1/6/26
Organize feedback review meetings with stakeholders for prioritization	5 days	Wed 12/31/25	Tue 1/6/26
Implement continuous improvements based on collected feedback	5 days	Wed 12/31/25	Tue 1/6/26
Project Review & Documentation	**4 days**	**Wed 1/7/26**	**Mon 1/12/26**
Conduct project retrospective to discuss successes and challenges	1 day	Wed 1/7/26	Wed 1/7/26
Prepare a final project report summarizing key learnings and outcomes	2 days	Thu 1/8/26	Fri 1/9/26
Archive all project documentation for future reference and scalability	1 day	Mon 1/12/26	Mon 1/12/26
Project Handover to University	**7 days**	**Tue 1/13/26**	**Wed 1/21/26**
Transition Plan	**2 days**	**Tue 1/13/26**	**Wed 1/14/26**
Assign responsibilities for maintaining and supporting the system	2 days	Tue 1/13/26	Wed 1/14/26
Knowledge Transfer	**3 days**	**Tue 1/13/26**	**Thu 1/15/26**
Conduct detailed training sessions for the university IT team	2 days	Tue 1/13/26	Wed 1/14/26
Provide in-depth documentation, including system architecture, APIs, and maintenance procedures	1 day	Thu 1/15/26	Thu 1/15/26
Data & Access Handover	**2 days**	**Fri 1/16/26**	**Mon 1/19/26**
Transfer ownership of cloud resources (Firebase, Azure, etc.)	1 day	Fri 1/16/26	Fri 1/16/26
Revoke old access credentials and issue new ones	1 day	Mon 1/19/26	Mon 1/19/26
Final Handover Report	**2 days**	**Tue 1/20/26**	**Wed 1/21/26**
Compile a summary of system	1 day	Tue 1/20/26	Tue 1/20/26
Ensure all stakeholders acknowledge the successful transition	1 day	Wed 1/21/26	Wed 1/21/26
Project Closure Meeting	**1 day**	**Thu 1/22/26**	**Thu 1/22/26**

`

APPENDIX C: RACI MATRIX

ID	Task	Project Manager	Compliance and Operations Managers	UI/UX designers	Software developers	DevOps	System Architect	Testers	Northeastern University (Admin)	Northeastern University (Clubs)	Northeastern University (Students)
1	**1.0 Project Initiation Phase**										
2	1.1 Discuss Project scope, purpose and objectives of the project	R	A			I	I		C	C	
3	1.2 Identify stakeholders	R,A	C				I		C	C	
4	1.3 Develop a communication plan	R,A	C	I	I	C	C	I	I		
5	1.4 Conduct initial meetigns to define technical feasibility	R,A	C	I	I	C	C	I			
6	1.5 Conduct meetings to define financial feasibility for development cost and I	R,A	C						I		
7	1.6 Conduct meetings to define operational feasibility	A	R	C	C	C	C	I			
8	1.7 Conduct student and faculty surveys to identify user needs and pain points	A	C	I	I	I	R		C	C	C
9	1.8 Define Success Criteria and Key Performance Indicators(KPIs)	R	C		I	I	A		C		
10	1.9 Team formation, defining & allocating responsibilities and scheduling stand-up meetings	R,A	C	I	I	I	I	I			
11	**2.0 Project Planning Phase**										
12	2.1 Define functional and non-functional requirements	A	C	C	C	C	R		C	C	
13	2.2 Define the System Architecture and Design plan	I	C	I	I	C	R,A	I			
14	2.3 Create Project Schedule and Resource allocation	R,A	C	I	I	I	C	I			
15	2.4 Develop Collaboration Setup	A	C	I	I	I	R	I			
16	2.5 Perform Risk Assessment and Mitigation Planning	R	A	I	I	C	C	I			
17	2.6 Establish Compliance and Legal Requirements	A	R	I	I	I	I	I	C	C	
18	**3.0 Project Execution**										
19	**3.1 Design**										
20	3.1.1 Design UI/UX wireframes	I		R	I	I	C	I			
21	3.1.2 Define user flow in the application	I		R	C	I	C	I			
22	3.1.3 Define admin flow in the application	I		R	C	I	C	I			
23	3.1.4 Define Database Schema and Backend API Structure	I		I	R	C	C	I			
24	3.1.5 Design security protocols (NEU OAuth, JWT authentication, RBAC)	I	C	I	R	C	R	I			
25	3.1.6 Design an interactive prototype for evaluation	I		R	C	I	C	I			
26	3.1.7 Conduct usability testing	I	C	I	I	C	I	R			
27	3.1.8 Feedback collection and implementation	I		R	I	I	I	I	C	C	C
28	**3.2 Development**										
29	3.2.1 Develop backend server setup and functions	I	C	I	R	C	A	I	C		
30	3.2.2 Set-up Firebase Database & Authentication and encryption protocols	I	C	I	R	I	A	I	C		
31	3.2.3 User Authentication & Role-Based Access Control (RBAC)	I	C	I	R	I	A	I	C		
32	3.2.4 Event Management API Development	I	C	I	R	I	A	I	C		
33	3.2.5 Develop Error tracking	I	C	I	R	I	A	R			
34	3.2.6 Integrate Automated Alerts and Notifications	I	C	I	R	I	A	I	C		
35	3.2.7 Logging and Debugging	I		I	R	C	A	R	C	C	C
36	3.2.8 Incident Response & Recovery			I	I	R,A	C		C	C	C
37	3.2.9 Backup & Disaster Recovery			I	I	R	A	R	C	C	C
38	3.2.10 Front end development	I		I	R	I	A	I			
39	3.2.11 Security and Performance Enhancement	I		I	R	C	A	I			
40	3.2.12 Integration & Unit Testing	I		I	R	A	C	R			
41	3.2.13 System Testing and Feedback Implementation	I		I	R	I	A	R	I	I	I

`

42	**4.0 Monitoring and Deployment Phase**									
43	4.1 Deploy backend services to Microsoft Azure for cloud hosting	I		I	C	R	A	I		
44	4.2 Release mobile application on Google Play Store & Apple App Store	I	C	I	R	A	C	I	I	I
45	4.3 Set up a CI/CD pipeline with Jenkins for automated updates	I		I	R	A	I	C	I	I	I
46	4.4 User training and documentation manual and virtual training sessions	I		R	C	A	C	C	I	I	I
47	**5.0 Project Closure**									
48	5.1 Monitor application performace stability and responsiveness	I		I	C	R	A	C	C	C	C
49	5.2 Collect user feedback for future enhancements and updates	I		I	I	R	A	I	C	C	C
50	5.3 Track bug fixes and updates using JIRA	I		I	I	R	A	C			
51	5.4 Prepare final project report and insights	I	C	C	C	R	A	C			
52	5.5 Archive all project documentation for future reference and scalability	I	C	C	C	R	A	C			
53	5.6 Define handover process and allocation responsibility for support and manatainance of the system	A,R	C	I	I	I	C	I	I		
54	5.7 Knowledge transfer to University IT team about architecture, API and maintainance procedures	I	C	C	C	R	A	C	I		
55	5.8 Data & Access Handover	A			C	C	R		I	I	I
56	5.9 Final Project handover to University	A	C		C	C	R		I	I	I
57	5.10 Project closure meeting	A	R	I	I	C	C	I			
58	5.11 Post - Launch Performance Analaytics and Optimization	I				R	A	C	I	I	I

APPENDIX D: BUDGET JUSTIFICATION

ID	Task Description	Labour - No. of people	Labour - Combined Work Hours	Labour Rate/hour (USD)	Labour Estimated Cost (USD)	Miscellaneous (USD)	Total Estimated Cost (USD)
1	Phase 1: Project Initiation		368				
2	1.1 Feasibility Study and Preliminary Research		264				
3	1.1.1 Conduct a technical feasibility study to assess system requirements (scalability, integration feasibility, security measures).	1	32	$ 70.00	$ 2,240.00	$ 500.00	$ 2,740.00
4	1.1.2 Conduct a financial feasibility study to estimate development costs and ROI	1	24	$ 50.00	$ 1,200.00	$ 300.00	$ 1,500.00
5	1.1.3 Evaluate operational feasibility by verifying usability within university systems (authentication, event management integration)	1	24	$ 70.00	$ 1,680.00		$ 1,680.00
		1	24	$ 50.00	$ 1,200.00	$ 200.00	$ 1,400.00
6	1.1.4 Conduct student and faculty surveys to identify event management pain points and user needs.	1	40	$ 55.00	$ 2,200.00		$ 2,200.00
		1	40	$ 45.00	$ 1,800.00		$ 1,800.00
		1	40	$ 45.00	$ 1,800.00		$ 1,800.00
		1	40	$ 30.00	$ 1,200.00	$ 900.00	$ 2,100.00
7	1.2 Project Kickoff & Scope Definition		64				
8	1.2.1 Define scope, purpose, and success criteria of the project	1	16	$ 55.00	$ 880.00	$ 500.00	$ 1,380.00
9	1.2.2 Conduct project kickoff meeting with stakeholders (Northeastern university and students) to align expectations	1	16	$ 55.00	$ 880.00		$ 880.00
		1	16	$ 50.00	$ 800.00	$ 300.00	$ 1,100.00
10	1.2.3 Establish Project communication channels and collaboration methods	0					
11	1.2.4 Develop a Stakeholder engagement plan	1	8	$ 55.00	$ 440.00		$ 440.00
		1	8	$ 50.00	$ 400.00	$ 900.00	$ 1,300.00
12	1.3 Project Team Formation		40				
13	1.3.1 Define project roles and responsibilities (UI/UX, development, testing, compliance)	1	16	$ 55.00	$ 880.00	$ 500.00	
14	1.3.2 Assign team members to specific project tasks based on expertise	1	16	$ 55.00	$ 880.00	$ 300.00	$ 1,180.00
15	1.3.4 Set up regular project meetings for progress updates and issue resolution	1	8	$ 55.00	$ 440.00	$ 200.00	$ 640.00
					Total Phase Cost		**$ 22,140.00**

16	Phase 2: Project Planning		328				
17	2.1 Requirements Gathering		112				
18	2.1.1 Define functional requirements (event creation, RSVP, notifications, admin roles).	1	16	$ 45.00	$ 720.00		$ 720.00
		1	16	$ 45.00	$ 720.00		$ 720.00
		1	16	$ 70.00	$ 1,120.00	$ 500.00	$ 1,620.00
19	2.1.2 Identify non-functional requirements (performance, security, scalability).	1	16	$ 70.00	$ 1,120.00		$ 1,120.00
		1	16	$ 50.00	$ 800.00	$ 300.00	$ 1,100.00
20	2.1.3 Establish key performance indicators (KPIs - System Performance & Uptime, Scalability & Load Handling, Notification Effectiveness) for project success.	1	16	$ 55.00	$ 880.00		$ 880.00
		1	16	$ 70.00	$ 1,120.00	$ 200.00	$ 1,320.00
21	2.2 System Architecture & Design Planning		48				
22	2.2.1 Finalize technology stack: React Native (frontend), Node.js (backend), Firebase (database)	1	16	$ 70.00	$ 1,120.00	$ 700.00	$ 1,820.00
23	2.2.2 Define system components, including database structure and API design	1	16	$ 70.00	$ 1,120.00	$ 500.00	$ 1,620.00
24	2.2.3 Plan system integrations (Google Calendar, university SSO authentication).	1	16	$ 70.00	$ 1,120.00	$ 300.00	$ 1,420.00
25	2.3 Scheduling and Resource Allocation		80				
26	2.3.1 Develop a timeline with milestones and deadlines	1	16	$ 55.00	$ 880.00	$ 900.00	$ 1,780.00
27	2.3.2 Identify and allocate human resources (functional manager, developers, UI/UX designers, testers, compliance)	1	8	$ 55.00	$ 440.00		$ 440.00
		1	8	$ 50.00	$ 400.00	$ 700.00	$ 1,100.00
28	2.3.3 Post job descriptions and project requirements on NU Works Portal of Northeastern University to recruit functional manager, developers, UI/UX designers, compliance specialists and testers.	1	8	$ 50.00	$ 400.00	$ 500.00	$ 900.00
29	2.3.3 Engage with Northeastern University faculty to identify students with relevant expertise for internship	1	8	$ 50.00	$ 400.00	$ 300.00	$ 700.00
30	2.3.4 Conduct interviews and skill assessments to select candidates for different project roles.	1	32	$ 50.00	$ 1,600.00	$ 200.00	$ 1,800.00
31	2.4 Collaboration Setup		32				
32	2.4.1 Provide access to Jira, Slack and development environments.	1	8	$ 55.00	$ 440.00	$ 700.00	$ 1,140.00
		1	8	$ 50.00	$ 400.00		$ 400.00
33	2.4.2 Conduct an initial onboarding meeting to align the team with project scope, timelines, and	1	8	$ 55.00	$ 440.00	$ 500.00	$ 940.00
34	2.4.3 Assign supervisors to track performance and guide project interns	1	8	$ 55.00	$ 440.00	$ 300.00	$ 740.00
35	2.5 Risk Assessment & Mitigation		56				
36	2.5.1 Identify project risks, including technical issues, adoption resistance, and security threats	1	16	$ 55.00	$ 880.00	$ 900.00	$ 1,780.00
		1	16	$ 50.00	$ 800.00		$ 800.00
37	2.5.2 Check for Northeastern University's IT security framework to ensure data encryption, secure authentication, and access control	1	16	$ 50.00	$ 800.00	$ 700.00	$ 1,500.00
38	2.5.3 Develop contingency plans for high-risk scenarios	1	8	$ 50.00	$ 400.00	$ 500.00	$ 900.00
39	2.5.4 Allocate risk ownership and implement monitoring strategies	0			$ -	$ 300.00	$ 300.00
					Total Phase Cost		$ 27,560.00

`

40	Phase 3: Project Execution		2872							
41	3.1 Design		872							
42	3.1.1 Develop UI/UX wireframes	1	80	$ 45.00	$ 3,600.00	$ 700.00	$ 4,300.00			
		1	80	$ 45.00	$ 3,600.00		$ 3,600.00			
43	3.1.2. Define User Flow (User and Admin Flow)	1	72	$ 45.00	$ 3,240.00		$ 3,240.00			
		1	72	$ 45.00	$ 3,240.00		$ 3,240.00			
		1	24	$ 50.00	$ 1,200.00		$ 1,200.00			
		1	24	$ 25.00	$ 600.00		$ 600.00			
		1	24	$ 25.00	$ 600.00	$ 200.00	$ 800.00			
44	3.1.3 Define Admin Flow	1	8	$ 50.00	$ 400.00		$ 400.00			
		1	72	$ 45.00	$ 3,240.00		$ 3,240.00			
		1	72	$ 45.00	$ 3,240.00	$ 500.00	$ 3,740.00			
45	3.1.4 Define Database Schema and Backend API Structure	1	24	$ 70.00	$ 1,680.00		$ 1,680.00			
		1	72	$ 25.00	$ 1,800.00		$ 1,800.00			
		1	72	$ 25.00	$ 1,800.00		$ 1,800.00			
		1	24	$ 45.00	$ 1,080.00		$ 1,080.00			
		1	24	$ 45.00	$ 1,080.00		$ 1,080.00			
		1	24	$ 50.00	$ 1,200.00	$ 900.00	$ 2,100.00			
46	3.1.5 Prototype Development	1	80	$ 45.00	$ 3,600.00		$ 3,600.00			
		1	24	$ 50.00	$ 1,200.00	$ 200.00	$ 1,400.00			
47	3.2 Development		1688							
48	3.2.1 Backend Development	1	384	$ 25.00	$ 9,600.00	$ 200.00	$ 9,800.00			
		1	384	$ 25.00	$ 9,600.00		$ 9,600.00			
		1	344	$ 65.00	$ 22,360.00		$ 22,360.00			
		1	56	$ 55.00	$ 3,080.00		$ 3,080.00			
49	3.2.2 Frontend Development	1	80	$ 25.00	$ 2,000.00		$ 2,000.00			
		1	80	$ 25.00	$ 2,000.00		$ 2,000.00			
		1	40	$ 45.00	$ 1,800.00		$ 1,800.00			
		1	40	$ 45.00	$ 1,800.00	$ 500.00	$ 2,300.00			
50	3.2.3 Security and Performance Enhancement	1	16	$ 65.00	$ 1,040.00		$ 1,040.00			
		1	40	$ 25.00	$ 1,000.00		$ 1,000.00			
		1	40	$ 25.00	$ 1,000.00	$ 500.00	$ 1,500.00			
51	3.2.4 Integration & Unit Testing	1	88	$ 50.00	$ 4,400.00		$ 4,400.00			
		1	40	$ 65.00	$ 2,600.00		$ 2,600.00			
		1	16	$ 45.00	$ 720.00		$ 720.00			
		1	40	$ 45.00	$ 1,800.00	$ 300.00	$ 2,100.00			
52	3.3 System Testing and Feedback Implementation		312							
53	3.3.1 Functional Testing	1	56	$ 50.00	$ 2,800.00		$ 2,800.00			
		1	24	$ 45.00	$ 1,080.00	$ 700.00	$ 1,780.00			
54	3.3.2 Performance & Load Testing	1	48	$ 45.00	$ 2,160.00	$ 900.00	$ 3,060.00			
55	3.3.3 Security Testing	1	32	$ 40.00	$ 1,280.00		$ 1,280.00			
		1	48	$ 45.00	$ 2,160.00	$ 200.00	$ 2,360.00			
56	3.3.4 User Acceptance Testing (UAT)	1	40	$ 50.00	$ 2,000.00	$ 300.00	$ 2,300.00			
57	3.3.5 Regression Testing	1	32	$ 50.00	$ 1,600.00		$ 1,600.00			
		1	32	$ 45.00	$ 1,440.00	$ 700.00	$ 2,140.00			
						Total Phase Cost	$ 1,22,520.00			
58	Phase 4: Monitoring		272							
59	4.1 Deployment & Go-Live		144							
60	4.1.1 Deploy backend services to Microsoft Azure for cloud hosting	1	48	$ 65.00	$ 3,120.00	$ 700.00	$ 3,820.00			
		1	24	$ 70.00	$ 1,680.00		$ 1,680.00			
		1	16	$ 50.00	$ 800.00		$ 800.00			
		1	8	$ 45.00	$ 360.00		$ 360.00			
61	4.1.2 Release mobile application on Google Play Store & Apple App Store	1	8	$ 65.00	$ 520.00	$ 700.00	$ 1,220.00			
62	4.1.3 Set up a CI/CD pipeline with Jenkins for automated updates	1	40	$ 65.00	$ 2,600.00	$ 500.00	$ 3,100.00			
63	4.2 User Training & Documentation	1	56	$ 55.00	$ 3,080.00		$ 3,080.00			
		1	56	$ 45.00	$ 2,520.00		$ 2,520.00			
		1	16	$ 45.00	$ 720.00	$ 700.00	$ 1,420.00			
						Total Phase Cost	$ 18,000.00			
64	Phase 5: Project Closure		632							
65	5.1 Post-Launch Support		304							
66	5.1.1 Monitor application stability, responsiveness, and crash reports	1	40	$ 70.00	$ 2,800.00		$ 2,800.00			
		1	40	$ 50.00	$ 2,000.00	$ 700.00	$ 2,700.00			
		1	40	$ 45.00	$ 1,800.00		$ 1,800.00			
67	5.1.2 Track bug reports using JIRA	1	40	$ 50.00	$ 2,000.00		$ 2,000.00			
		1	40	$ 45.00	$ 1,800.00	$ 500.00	$ 2,300.00			
68	5.1.3 Collect user feedback for future enhancements and updates	1	64	$ 55.00	$ 3,520.00		$ 3,520.00			
		1	40	$ 50.00	$ 2,000.00	$ 300.00	$ 2,300.00			
69	5.2 Project Review & Documentation		64							
70	5.2.1 Conduct project retrospective to discuss successes and challenges	1	16	$ 55.00	$ 880.00	$ 300.00	$ 1,180.00			
71	5.2.2 Prepare a final project report summarizing key learnings and outcomes	1	32	$ 55.00	$ 1,760.00	$ 200.00	$ 1,960.00			
72	5.2.3 Archive all project documentation for future reference and scalability	1	16	$ 55.00	$ 880.00	$ 900.00	$ 1,780.00			
73	5.3 Project Handover to University		264							
74	5.3.1 Transition Plan	1	32	$ 55.00	$ 1,760.00	$ 500.00	$ 2,260.00			
75	5.3.2 Knowledge Transfer	1	40	$ 55.00	$ 2,200.00		$ 2,200.00			
		1	40	$ 45.00	$ 1,800.00		$ 1,800.00			
		1	40	$ 45.00	$ 1,800.00		$ 1,800.00			
		1	16	$ 70.00	$ 1,120.00	$ 900.00	$ 2,020.00			
76	5.3.3 Data & Access Handover	1	16	$ 65.00	$ 1,040.00		$ 1,040.00			
		1	16	$ 45.00	$ 720.00		$ 720.00			
		1	8	$ 55.00	$ 440.00		$ 440.00			
		1	8	$ 50.00	$ 400.00	$ 300.00	$ 700.00			
77	5.3.4 Final Handover Report	1	24	$ 55.00	$ 1,320.00		$ 1,320.00			
		1	24	$ 50.00	$ 1,200.00	$ 700.00	$ 1,900.00			
78	5.4 Project Closure Meeting									
						Total Phase Cost	$ 38,540.00			
					TOTAL BUDGET		$ 2,28,760.00			

`

APPENDIX E: RESOURCE ALLOCATION

	Resource Name	Work	Details	3/16	3/23	3/30	4/6	4/13	4/20	4/27
						April				May
1	**Project Manager**	**504 hrs**	Work	40h	32h	48h	16h	16h	8h	24h
2	UI Designer	536 hrs	Work	40h			16h			
3	UX Designer	440 hrs	Work	40h			16h			
4	**System Architect**	**256 hrs**	Work	56h			72h	24h		
5	Compliance Manager	176 hrs	Work	24h			16h			
6	UAT Tester	320 hrs	Work							
7	DevOps Engineer	512 hrs	Work							
8	Intern	40 hrs	Work	40h						
9	NU Admin	192 hrs	Work	24h	16h	8h			40h	24h
10	Software Developer 1	600 hrs	Work							
11	Software Developer 2	600 hrs	Work							
12	Performance Tester	296 hrs	Work							

	Resource Name	Work	Details	5/4	5/11	5/18	5/25	6/1	6/8	6/15
								June		
1	**Project Manager**	**504 hrs**	Work	16h						
2	UI Designer	536 hrs	Work		32h	40h	16h	40h	40h	24h
3	UX Designer	440 hrs	Work		32h	40h	16h	40h	40h	24h
4	**System Architect**	**256 hrs**	Work							
5	Compliance Manager	176 hrs	Work	40h			24h			
6	UAT Tester	320 hrs	Work							
7	DevOps Engineer	512 hrs	Work							
8	Intern	40 hrs	Work							
9	NU Admin	192 hrs	Work						8h	
10	Software Developer 1	600 hrs	Work				24h			
11	Software Developer 2	600 hrs	Work				24h			
12	Performance Tester	296 hrs	Work							

	Resource Name	Work	Details	6/22	6/29	7/6	7/13	7/20	7/27	8/3
					July				August	
1	**Project Manager**	**504 hrs**	Work							
2	UI Designer	536 hrs	Work	32h	24h	16h	40h	24h		
3	UX Designer	440 hrs	Work	32h	24h					
4	**System Architect**	**256 hrs**	Work	8h	16h					
5	Compliance Manager	176 hrs	Work			24h				
6	UAT Tester	320 hrs	Work				24h			
7	DevOps Engineer	512 hrs	Work						16h	
8	Intern	40 hrs	Work							
9	NU Admin	192 hrs	Work							
10	Software Developer 1	600 hrs	Work	8h	40h	24h		16h	40h	40h
11	Software Developer 2	600 hrs	Work	8h	40h	24h		16h	40h	40h
12	Performance Tester	296 hrs	Work							

	Resource Name	Work	Details	8/10	8/17	8/24	8/31	9/7	9/14	9/21
							September			
1	**Project Manager**	**504 hrs**	Work							
2	UI Designer	536 hrs	Work							
3	UX Designer	440 hrs	Work							
4	**System Architect**	**256 hrs**	Work							
5	Compliance Manager	176 hrs	Work							
6	UAT Tester	320 hrs	Work							
7	DevOps Engineer	512 hrs	Work			24h	40h	40h	40h	40h
8	Intern	40 hrs	Work							
9	NU Admin	192 hrs	Work							
10	Software Developer 1	600 hrs	Work	40h	40h	40h	40h	40h	32h	
11	Software Developer 2	600 hrs	Work	40h	40h	40h	40h	40h	32h	
12	Performance Tester	296 hrs	Work							

`

	Resource Name	Work	Details	October 9/28	10/5	10/12	10/19	10/26	November 11/2	11/9
1	**Project Manager**	**504 hrs**	Work	24h	32h					
2	UI Designer	536 hrs	Work					16h	24h	
3	UX Designer	440 hrs	Work					16h	24h	
4	**System Architect**	**256 hrs**	Work							
5	Compliance Manager	176 hrs	Work							
6	UAT Tester	320 hrs	Work							
7	DevOps Engineer	512 hrs	Work	40h	40h	40h	24h		16h	
8	Intern	40 hrs	Work							
9	NU Admin	192 hrs	Work							
10	Software Developer 1	600 hrs	Work	24h	32h		16h	40h	40h	24h
11	Software Developer 2	600 hrs	Work	24h	32h		16h	40h	40h	24h
12	Performance Tester	296 hrs	Work							

	Resource Name	Work	Details	11/16	11/23	December 11/30	12/7	12/14	12/21	January 12/28
1	**Project Manager**	**504 hrs**	Work							
2	UI Designer	536 hrs	Work		16h					
3	UX Designer	440 hrs	Work							
4	**System Architect**	**256 hrs**	Work							
5	Compliance Manager	176 hrs	Work						32h	
6	UAT Tester	320 hrs	Work	32h	32h	40h	40h			24h
7	DevOps Engineer	512 hrs	Work	32h	8h					
8	Intern	40 hrs	Work							
9	NU Admin	192 hrs	Work							
10	Software Developer 1	600 hrs	Work							
11	Software Developer 2	600 hrs	Work							
12	Performance Tester	296 hrs	Work		16h	40h	8h	40h	40h	16h

	Resource Name	Work	Details	1/4	1/11	1/18	1/25	February 2/1	2/8	2/15
1	**Project Manager**	**504 hrs**	Work					16h		
2	UI Designer	536 hrs	Work				16h	40h		
3	UX Designer	440 hrs	Work				16h	40h		
4	**System Architect**	**256 hrs**	Work		8h	16h			40h	
5	Compliance Manager	176 hrs	Work		8h	8h				
6	UAT Tester	320 hrs	Work	40h	8h				40h	40h
7	DevOps Engineer	512 hrs	Work		32h	40h	24h			
8	Intern	40 hrs	Work							
9	NU Admin	192 hrs	Work							
10	Software Developer 1	600 hrs	Work							
11	Software Developer 2	600 hrs	Work							
12	Performance Tester	296 hrs	Work	24h	8h	8h			40h	40h

	Resource Name	Work	Details	2/22	March 3/1	3/8	3/15	3/22	April 3/29	4/5
1	**Project Manager**	**504 hrs**	Work	40h	40h	40h	40h	40h	24h	8h
2	UI Designer	536 hrs	Work					40h		
3	UX Designer	440 hrs	Work					40h		
4	**System Architect**	**256 hrs**	Work					16h		
5	Compliance Manager	176 hrs	Work							
6	UAT Tester	320 hrs	Work							
7	DevOps Engineer	512 hrs	Work						16h	
8	Intern	40 hrs	Work							
9	NU Admin	192 hrs	Work	40h					24h	8h
10	Software Developer 1	600 hrs	Work							
11	Software Developer 2	600 hrs	Work							
12	Performance Tester	296 hrs	Work						16h	