Project Report

MODERN METHODS IN SOFTWARE ENGINEERING

(ID2207)

KTH ROYAL INSTITUTE OF TECHNOLOGY

Group : 13

Name : Md Sakibul Islam
Name : Md Ahsanul Karim
Name : Singvalliyappa Velayutham

Date : 29 October, 2025

1. User Stories

1.1 Login

Login**:** “When logging in, the SEP user authenticates with a password to perform actions available only to their assigned access level.”
Time Estimate: GUI: 2 hours , Logic: 2.5 hours

1.2 Event Request Submission

Event Request Submission**:** A Customer Service Officer enters the client’s event request information, and submits it to the Senior Customer Service Officer so that the request can move to the next step in the workflow.
Time Estimate: GUI: 4 hours , Logic: 8 hours

1.3 Request Feasibility and Routing

Request Feasibility and Routing : A Senior Customer Service Officer logs into the system, reviews the event request, rejects it if not feasible, or forwards it to the Financial Manager so that the budget can be properly evaluated.
Time Estimate: GUI: 0.25 hours , Logic: 2 hours

1.4 Budget Feedback and Forwarding

Budget Feedback and Forwarding : A Financial Manager logs into the system, reviews the event request and its budget

details, and forwards it to the Administration Manager so that financial decisions can support the approval process.
Time Estimate: GUI: 0.25 hours , Logic: 2 hours

1.5: Final Approval Decision

Final Approval Decision : An Administration Manager logs into the system, approves or rejects the event request, and returns the approval decision to the Senior Customer Service Officer, allowing the client to be notified.
Time Estimate: GUI: 0.25 hours , Logic: 1 hours

1.6: Business Meeting Scheduling

Business Meeting Scheduling : A Senior Customer Service Officer logs into the system and schedules a business meeting with the client after approval, allowing detailed event requirements to be collected.
Time Estimate: GUI: 4 hours , Logic: 4 hours

2.1 - Service / Manager Team creation:

Service / Manager Team creation: “Service/Production manager creates a sub-team from all the employees available and gives the team a name . Only people in the team will be able to view it ”.
Time Estimate : GUI : 2 hours , Logic : 2hours

2.2 - Task Assignment to Team

Task Assignment to team: “The manager can create a new task and assign it to the team”.

Time Estimate : GUI : 1 hours , Logic : 1hours

2.3 - View tasks

View Tasks: “Members in the team can view the task”.
Time Estimate : GUI : 1hours , Logic : 1hours

2.4 - Add comments

Add Comments: “Members in the team can add comments , comments are replies follow a flat hierarchy for simplicity”.
Time Estimate : GUI : 2 hours , Logic : 2hours

3.1 - HR Request Form Creation

HR Request Form Creation: "A Service Manager or Production Manager opens the HR request form. The manager fills details for staff role neeeded, quantity, justification, event reference to ask the HR team for extra members"
Time Estimate: GUI: 2 hour, Logic: 0.5 hours

3.2 - Send HR Request

Send HR Request to HR Manager: "The Service Manager or Production Manager reviews details and submits the request. The system forwards it to the HR Manager, for further decision that new recruitment or outsourcing is required for an event."
Time Estimate: GUI: 0.25 hour, Logic: 1 hour

3.3 - View and Process HR Request

HR Manager Views and Processes Request: "The HR Manager sees incoming HR requests, and checks current employee records and schedules. If available staff meet the requirements, the manager assigns them. If not, the HR Manager proceeds to initiate a job advert or vendor contact for outsourcing."
Time Estimate: GUI: 0.5 hour, Logic: 2 hours

3.4 - Change HR Request Status

Change HR Request Status: “HR Manager reviews each HR request and updates its status so the Production/Service Manager always has current info. If staff are assigned, the status changes to ‘Recruited’. If hiring is ongoing, it is ‘Under Review’. If the request cannot be fulfilled, the status is updated to ‘Denied’.”
Time Estimate: GUI: 0.25 hour, Logic: 1 hour

3.5 - Check HR Request

Check HR Request: "A Production or Service Manager views the status for their HR request in the system. He/she can follow the request if the recruitment was successful or not.”
Time Estimate: GUI: 0.25 hour, Logic: 1 hour

4.1 - Financial Request Form Creation

Financial Request Form Creation: “A Service Manager or Production Manager opens the financial request form. The Service Manager or Production Manager fills in the event reference, the additional budget needed, and provides a detailed justification for the extra budget.”
Time Estimate: GUI: 1 hour, Logic: 2 hours

4.2 - Send Financial Request

Send Financial Request : "After the financial request form is completed, the Service Manager or Production Manager reviews the details and submits the request for approval. The system forwards the request to the Financial Manager."
Time Estimate: GUI: 0.25 hour, Logic: 1 hour

4.3 - Review Financial Request

Financial Request Review : "The Financial Manager receives the new financial request. The Financial Manager examines the requested amount, checks the justification, and may contact the requesting manager for clarification.
Time Estimate: GUI: 0.5 hour, Logic: 2 hours

4.4 - Update Financial Request Status

Status Update for Financial Request : “The Financial Manager makes a decision, he/she updates the status of the financial request to either “Approve” if the budget is added or “Reject” if he/she thinks to deny the request or the client refuses to add the budget." The system automatically sends a notification to the original Service Manager or Production Manager with the updated status.
Time Estimate: GUI: 0.25 hour, Logic: 0.5 hour

4.5 - Check Financial Request Status

Financial Request Status Check : "Service Manager or Production Manager view the status of their submitted financial request. If approved, the budget becomes available for the event. If rejected, the manager reviews feedback and decides on the next steps."
Time Estimate: GUI: 1 hour, Logic: 1 hour

2. Release Planning

User Story Name	Value	Risk	Iteration
Login	High	High	1
Event Request Submission Story	High	High	1
Request Feasibility and Routing	Medium	Low	1
Budget Feedback and Forwarding	High	Medium	1
Final Approval Decision	High	Medium	1
Business Meeting Scheduling	Medium	Low	1
Team Creation	High	Low	2
Task Assignment	High	Medium	2
View Tasks	High	Low	2
Add Comments	Medium	High	2
HR Request Form Creation	High	High	3
Send HR Request	Medium	Low	3
View and Process HR Request	High	Medium	3
Change HR Request Status	Medium	Low	3
Check HR Request	Medium	Medium	3
Financial Request Form Creation	High	High	4
Send Financial Request	High	Medium	4
Review Financial Request	High	High	4
Update Financial Request Status	Medium	Medium	4
Check Financial Request Status	Medium	Low	4

	High Value	Medium Value	Low Value
High Risk	7	2	0
Medium Risk	4	2	1
Low Risk	4	2	0

3. Metaphor

Metaphor	System
Airport	Event planning company (SEP)
Air Traffic Controller	Service Manager / Production Manager
Flight	Event request or scheduled event
Flight Plan Submission	Client's event request entry by Customer Service
Pre-Flight Check	Senior Customer Service Officer's feasibility review
Ground Crew Assignment	Allocation of staff and resources
Runway Assignment	Resource (venue/service/personnel) scheduling
Control Tower Clearance	Approval/rejection at key workflow gates (Admin/Finance)
Taxi to Runway	Meeting scheduling and detailed preparation
Flight Takeoff	Event Execution commencement
Arrival & Feedback	Event closure and feedback collection

4. Iteration Management

Our development was divided into three iterations , each iteration being a bit more complex than the last. In the first iteration, we concentrated on implementing the login functionality and creating the initial forms for event creation. This allowed us to establish the basic authentication flow and ensure that users could submit event-related data correctly. In the second iteration, we built the team management module and refined event behavior, such as linking events to teams and managing their interactions. Finally, in the third iteration, we developed the frontend interface and connected it to the

backend through API integrations. This helped us test the complete workflow from user actions on the interface to database responses, ensuring that all components worked together seamlessly.

5. Test-Driven Pair-Programming

We followed a test-driven pair programming approach by first writing small example tests for our database and user operations. To begin, we broke down the system into controllers and models. The models are Users (employees,events,teams etc) and the controllers are responsible for each of the use cases. Based on this plan, we wrote tests to verify the basic functionality of each part. All these controllers have corresponding tests defined for them. Once the tests were defined, we started implementing the classes and functions step by step. Every time a piece of code was ready, we ran the tests to see if it behaved as expected. If the tests failed, we debugged and refined the implementation until everything passed. Throughout this process, both members of the team contributed equally to writing, testing, and debugging the code.

6. Applied Refactoring

At the beginning, we implemented all the routes and logic within a single file, which quickly became messy and difficult to manage as the project grew. During the latter part of the second iteration, we decided to refactor the codebase by separating concerns into modules, controllers, and routers. This restructuring made the code much cleaner and easier to maintain, while also improving collaboration between team members. Each module was assigned a specific responsibility, and the router files were used to organize the API endpoints systematically. This refactoring process significantly improved readability, scalability, and debugging efficiency.

7. Acceptance Test

Test case name	**Login**
Expected actions	1. Open the system URL in a browser. 2. The login interface should appear. 3. Enter the email ”ahsan@example.com” in the Email field. 4. Enter the password “123456” in the Password field.

	5. Click the “Login” button.
Expected results	1. The system validates the entered credentials. 2. If valid, a message “Login successful!!” appears. 3. The system stores the JWT token in local storage. 4. The user is redirected to the main interface (Event Form page).
Test result	Successful

Test case name	**Create Client Event Request**
Expected actions	1. Click on “New Request” in the sidebar to open the Event Form. 2. Fill in all required fields with valid data: a. Client Record Number: “CR-001” b. Client Name: “Tech Innovators AB” c. Event Type: “Workshop” d. Description: “AI Integration Workshop for corporate teams” e. Expected Number: “50” f. Budget: “15000” g. From: “2025-11-10” h. To: “2025-11-12” 3. Optionally fill or leave blank optional fields: i. Decorations: “Stage backdrop, lighting setup, flower arrangements.” j. Food / Drinks: “Buffet lunch and evening snacks for 120 people.” k. Filming / Photos: “Professional videographer and photographer.” l. Music: “Background instrumental playlist and closing ceremony music.” m. Posters / Artwork: “Event banners, company logo posters, digital screens.” n. Computer-Related Issues: “Two projectors, sound system, and registration tablets.” o. Other Needs: “Parking permits and guest transport arrangement.” 4. Click the “Submit” button.
Expected results	1. The system validates that all required fields contain valid data.

	2. Numeric fields (“Expected Number”, “Budget”) accept only positive integers ≥ 1. 3. Dates are validated (“From” ≤ “To”). 4. The system automatically assigns default text "N/A" for optional empty fields. 5. The system sends the event request data to the backend API (/events) with an authorization header containing the JWT token. 6. The backend stores the event record successfully in the database. 7. A success message “Request submitted successfully!” appears. 8. The form clears, and the user is redirected to the “Client Requests List” page showing the newly created request.
Test result	Successful

8. Daily Meeting

Meeting Dates	09/10/2025
Participants	Sakibul, Ahsan, Singvalliyappa
Meeting Notes	1. Summary of our activities in previous day: a. Completed writing and reviewing all the HR user stories for staff recruitment, sending requests, status updates, and checks. b. Integrated feedback from the last group meeting to clarify the workflow graphs and HR documentation. 2. Today expected actions: a. Schedule interviews with selected candidates. b. Update recruitment request status and notify corresponding managers. 3. Problems: a. A few candidates declined interview invites and need to expand outreach.
Comments	Expected outcome for today: Complete the HR request form draft and finalize the logic for status changes and notifications.

Meeting Dates	14/10/2025

Participants	Singvalliyappa, Ahsan, Sakibul
Meeting Notes	1. Summary of our activities in previous day: a. Finished creating the initial database tables for financial requests and set up basic UI wireframes. b. Reviewed last week's flow to confirm all financial user stories are covered in our backend schema. 2. Today expected actions: a. We'll start developing the Finance Manager user and connect it to the backend authentication logic. b. We'll refine the financial request page UI to match the wireframe, and link it to our database. 3. Problems: a. We found a bug in the form validation script. It isn't saving all fields correctly, so it needs debugging before testing.
Comments	Expected outcome for today: Ensure every financial request is thoroughly analyzed and that teams receive clear instructions on submission standards. Aim for better consistency and fewer errors tomorrow.

9. Conclusion

The realization of this project demonstrates how the principles of Extreme Programming (XP) and Test-Driven Development (TDD) can streamline system implementation and improve clarity throughout the development process. By focusing on small iterations, frequent testing, and continuous feedback, the XP and TDD approaches allowed us to build a functional event management workflow system that is both reliable and easy to refine. Compared to the Object-Oriented Programming (OOP) approach, which emphasizes complex relationships between classes and higher abstraction, XP promotes simplicity and direct collaboration, making the system easier to understand and adapt during rapid development. Through systematic testing and incremental realization, the project achieved its objectives efficiently, ensuring functionality, maintainability, and user satisfaction.

10. Setup and Execution Instructions

This system consists of two main components:

- Backend (Server): Handles authentication, database operations, and event request APIs.

- Frontend (Client): Provides the user interface for manager login, event creation, and request management.

1. Prerequisites

Before running the system, ensure the following are installed:

- **Node.js** (version 18 or higher)
- **npm** (comes with Node.js)
- **MongoDB** (running locally or accessible remotely)

2. Backend Setup

- Open a terminal and navigate to the folder: “npm install”
- Create a .env file inside the backend directory and add:
“ PORT=8000
MONGO_URI=mongodb://localhost:27017/eventdb
JWT_SECRET=your_secret_key “

- Start the MongoDB server (if running locally): mongod
- Run the backend server: npm start
- The backend will run at “http://localhost:8000”

3. Frontend Setup

- Open another terminal and navigate to the frontend folder: “cd frontend”
- Install dependencies: “npm install”
- Start the development server: “npm run dev”
- The frontend will run at “http://localhost:5173”