SDLC Case Study Worksheet – Resource.CS

Project Title

Resource.CS – Centralized Learning & Tech Event Hub for CS Students

Team Name

Web Scalers

Team Members & Roles

• Muhammad Ibrahim(B23110006103) – Project Manager
• Afan Qaiser Farooqi(B2311000606) – Backend Developer (Node.js, Express.js)
• M Bilal Atif Usmani (B23110006057)– Frontend Developer (React.js)
• Abdullah Khan and Abdullah Mufeez(B23110006078,B23110006080) – UI/UX Designer (Figma)
• M Ibrahim(B23110006103)– QA Tester
• Amir Aslam (B23110006010)– Deployment & DevOps (Cloud, CI/CD)

1. Requirements Phase

Functional Requirements

1. User Registration & Login – Email/Google authentication.
2. Role Management – Support for Student and Admin roles.
3. Profile Management – Students can update semester, interests.
4. Topic-wise Resource Listing – Admin can add/update curated YouTube links, notes.
5. Search & Filter – Students can find resources by topic.
6. Tech Event Posting – Admin can post events (hackathons, webinars, jobs).
7. Notifications – System sends real-time push/email alerts.
8. Feedback System – Students can rate and comment on resources.
9. Dashboard – Students can view personalized recommended content.
10. Secure Backend API – JWT-based authentication and authorization.
11. Admin Panel – Manage users, resources, and events.
12. Database Management – Store users, resources, events, and feedback with backups.
13. Documentation – User guide and technical documentation for future teams.

Non-Functional Requirements

1. Performance – Handle 200+ concurrent students with < 2 sec response.
2. Security – HTTPS, JWT, password hashing, DB indexing.
3. Usability – Mobile-first, responsive design.
4. Scalability – Can add more categories and events easily.
5. Reliability – 99.5% uptime target.
6. Maintainability – Clean, modular MERN stack code.
7. Portability – Work on all major browsers and mobile devices.

2. Design Phase

Work Breakdown Structure (WBS)

1. Project Planning & Research
   1.1 Requirement Gathering
   1.2 Stakeholder Identification
   1.3 Technical Feasibility Study
   1.4 Project Timeline & Milestones

2. UI/UX Design
   2.1 Wireframing (Login, Dashboard, Resources, Events)
   2.2 High-Fidelity Designs (Figma)
   2.3 Responsive Prototypes
   2.4 Feedback Review

3. Frontend Development (React.js + React Native)
   3.1 Component Development (Login, Dashboard, Resource Card)
   3.2 State Management (Context API / Redux Toolkit)
   3.3 Integration with Backend APIs
   3.4 Testing & Responsiveness

4. Backend Development (Node.js + Express.js)
   4.1 API Structure & Routes (Users, Resources, Events)
   4.2 JWT Authentication
   4.3 CRUD Operations
   4.4 Middleware & Validation
   4.5 API Testing (Postman)

5. Database (MongoDB Atlas)
   5.1 Schema Design (Users, Resources, Events, Feedback)
   5.2 Indexing & Optimization
   5.3 Backup & Restore

6. Notifications & Admin Panel
   6.1 Event Notifications
   6.2 Admin Resource & Event Management

7. Testing & QA
   7.1 Unit Testing
   7.2 Integration Testing
   7.3 UI Testing

8. Deployment
   8.1 Frontend (Vercel/Netlify)
   8.2 Backend (Render/Heroku)
   8.3 DB (MongoDB Atlas)
   8.4 Final Handover

UML Use Case (High Level)

Actors: Student, Admin
Student Use Cases: Register/Login, View & Search Resources, View Events, Give Feedback, Receive Notifications
Admin Use Cases: Add/Edit/Delete Resources, Add/Edit/Delete Events, Manage Users, Monitor Feedback

3. Backend Design

Entities: Users, Resources, Events, Feedback, Notifications.
Relationships: One-to-Many relationships between User-Resource, User-Event, User-Feedback, and Resource-Feedback.

4. Development Phase

Pseudo-code Example for Resource Upload:

```
FUNCTION addResource(title, description, category, link, adminId)
   Validate adminId and check role = 'admin'
   Validate inputs (title, link, category)
   Save resource in database with createdAt timestamp
   Return success message + resourceId
END FUNCTION
```

5. Testing Phase

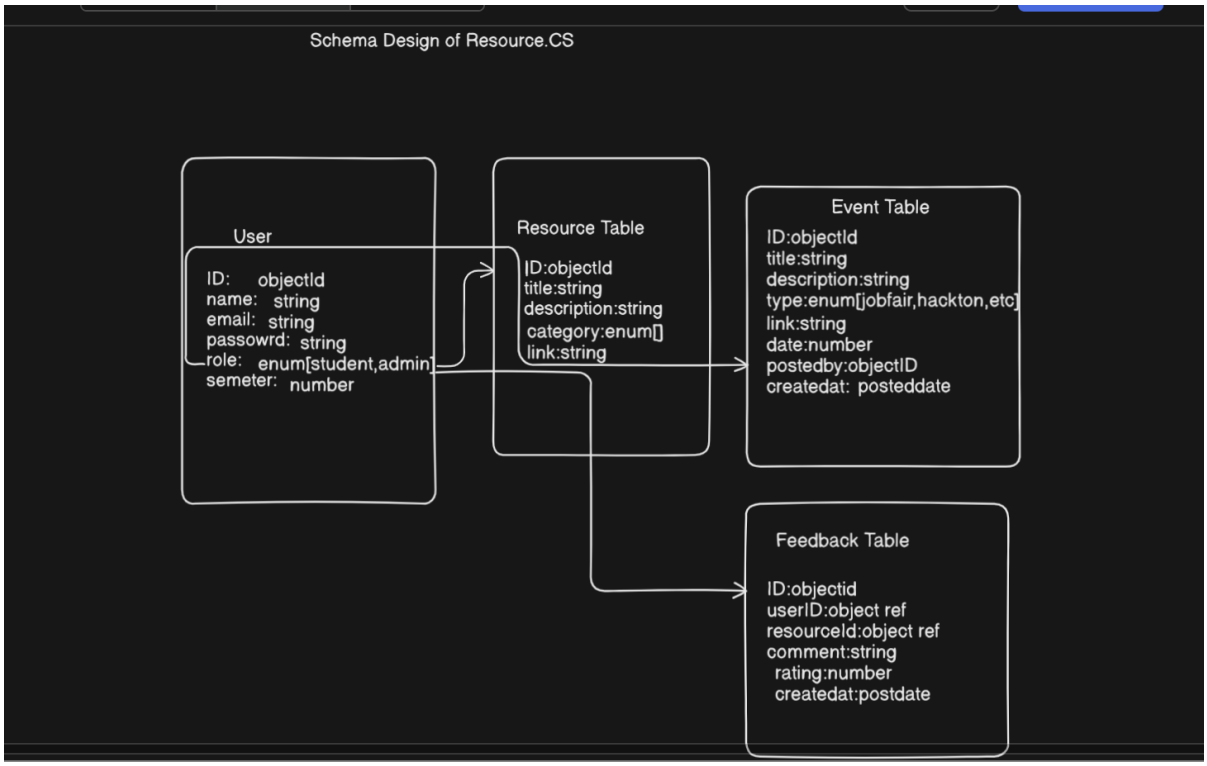
Test Case ID	Description	Input	Expected Output	Result

TC-01	Verify user login	Email+Password	Redirect to dashboard	Pass
TC-02	Add new resource	Resource title + link	Resource stored & listed	Pass
TC-03	Event notification	Post new event	Students receive notification	Pass

6. Reflection

Most Challenging Phase: Design Phase – because WBS, ERD, and UML had to cover web requirements.
Best SDLC Model: Agile – since we can release MVP first and add features in sprints.
Requirement Derivation: Functional requirements came from student pain points (scattered resources, missed events). Non-functional requirements focus on scalability, security, and usability.

7. Schema Design



Design link : [Schema design link]