CSE-6324-004-ADV TOPS SOFTWARE ENGINEERING

Team - 09

Academiverse

Final Report

Prepared by:

Ekta Patel - 1002166089

Hetvi Joshi - 1002145106

Jainik Gadara - 1002144181

Param Shah - 1002159523

Dr. Farnaz Farahanipad

Part 1: Project Initiation

Project Proposal:

Client Requirements:

1. User-friendly interface for both professors and students
2. Secure authentication system with role-based access
3. Course management capabilities
4. Assignment and quiz functionality
5. Real-time grading system
6. Communication features between students and professors

Objectives:

1. Streamline course management processes
2. Improve student-professor communication
3. Provide real-time access to learning materials
4. Enable efficient assignment submission and grading
5. Create a collaborative learning environment

Scope of the project:

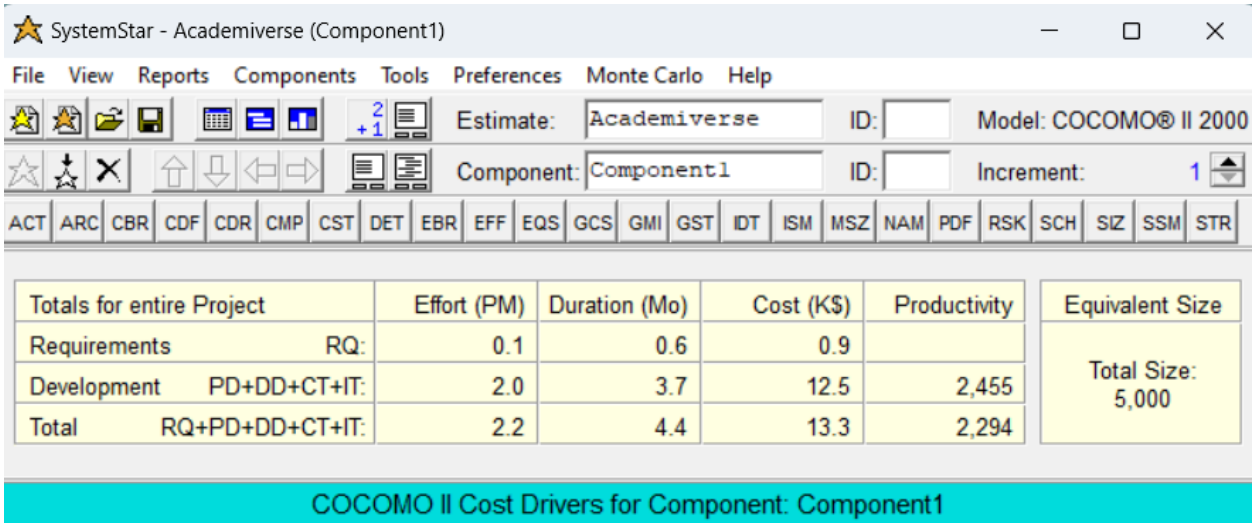
The project's scope covers the design, development, and implementation of a scalable LMS that caters to the needs of both professors and students at the university level. This platform will:

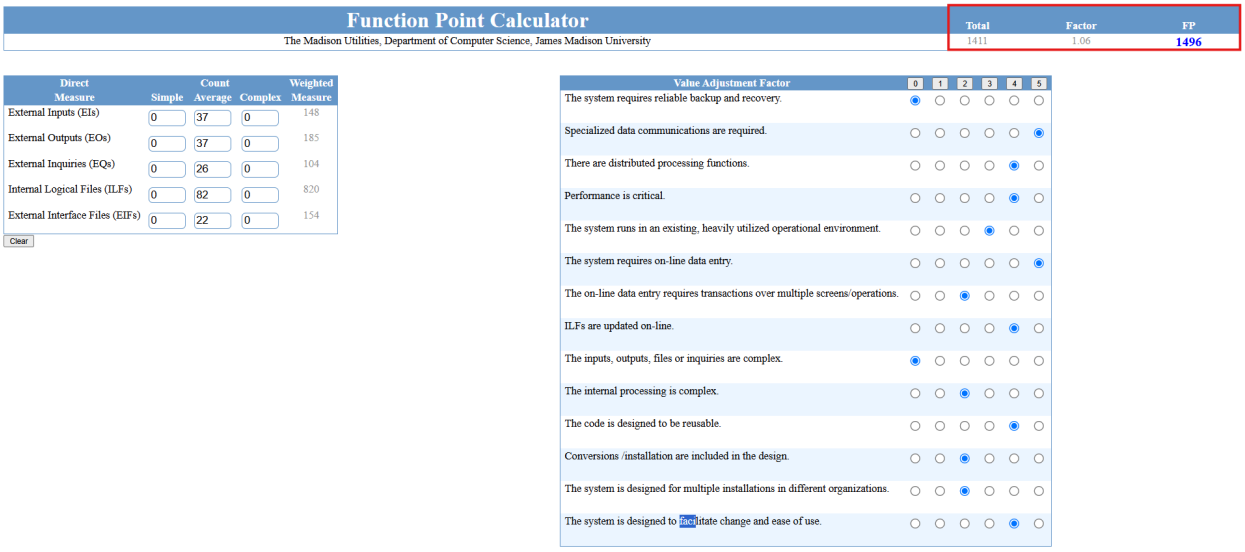
- Allow professors to manage courses, quizzes, and assignments.
- Provide students with seamless access to learning materials, assignments, and real-time grading.
- Facilitate communication and collaboration between students and professors.

The system will be built using agile methodologies (Scrum) with iterative development cycles, ensuring continuous feedback and timely delivery of the project within a 2.5-month period.

Cost and time estimation:

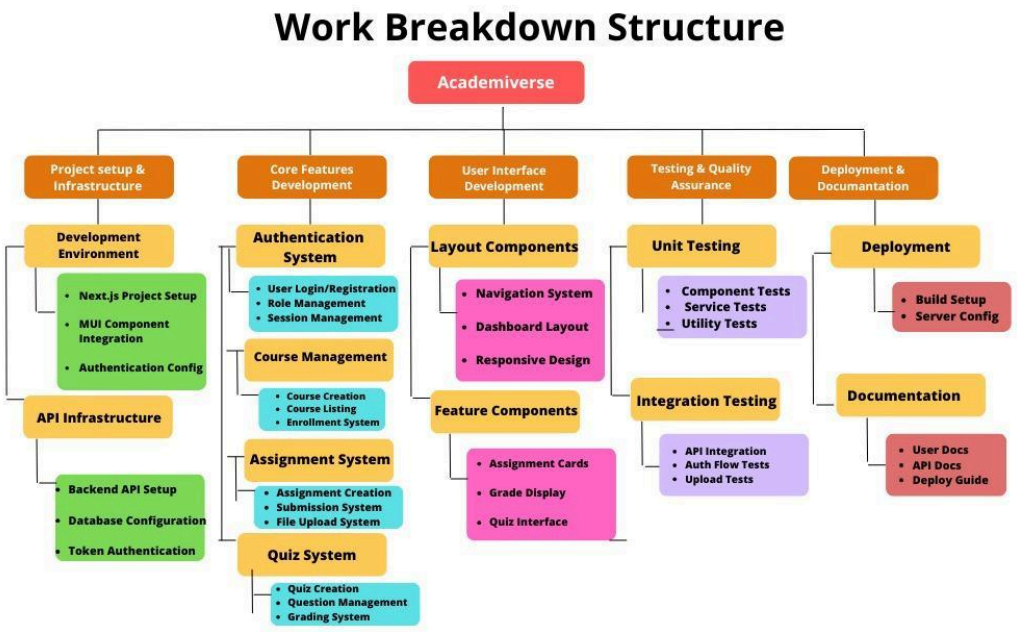
Estimates Size: 5000 LOC



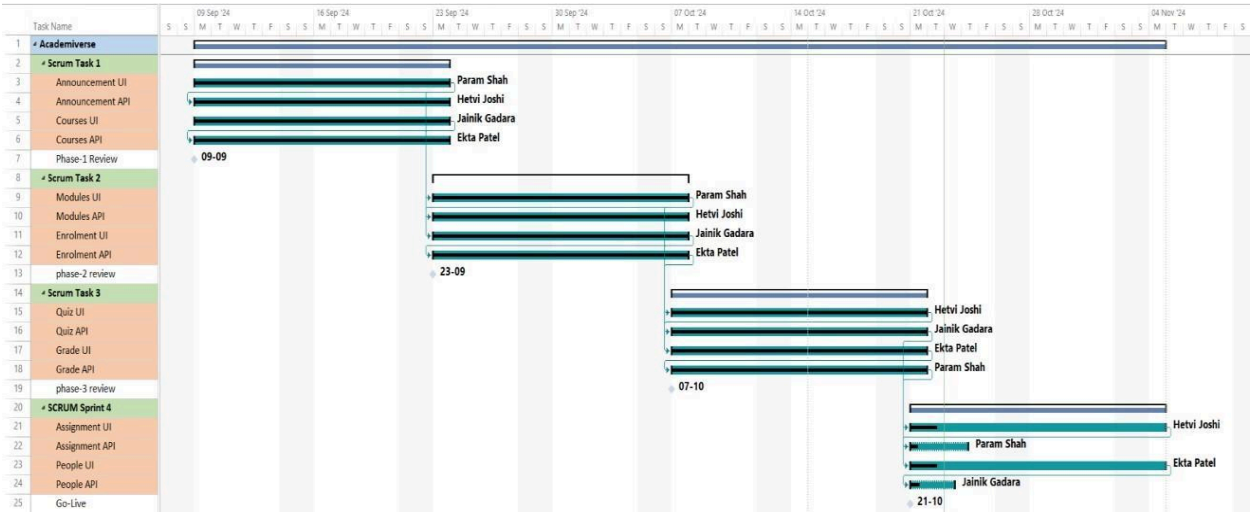


Part 2: Project Planning

Work breakdown structure:



Project Schedule:



Part 3: Risk Management

Risk Identification:

Technical Risks:

1. Integration issues between frontend and backend
2. Performance bottlenecks with multiple concurrent users
3. Database scalability challenges

Organizational Risks:

1. Team member availability
2. Communication gaps
3. Scope creep

External Risks:

1. Third-party API changes
2. Cloud service disruptions
3. Security vulnerabilities

Risk Analysis:

Impact (1-5) × Probability (1-5):

Risk

Risk	Impact	Probability	Score
Integration issues	4	3	12
Performance bottlenecks	5	3	15
Database scalability	4	2	8
Team availability	3	4	12
Scope creep	4	4	16

Mitigation Plan:

1. Scope Creep (High Priority):
    a. Strict adherence to sprint planning
    b. Regular stakeholder reviews
    c. Change control process
2. Performance Issues:
    a. Regular performance testing
    b. Load testing before deployment
    c. Optimization sprints

3. Team Availability:
    a. Backup resource planning
    b. Clear communication channels
    c. Regular status updates

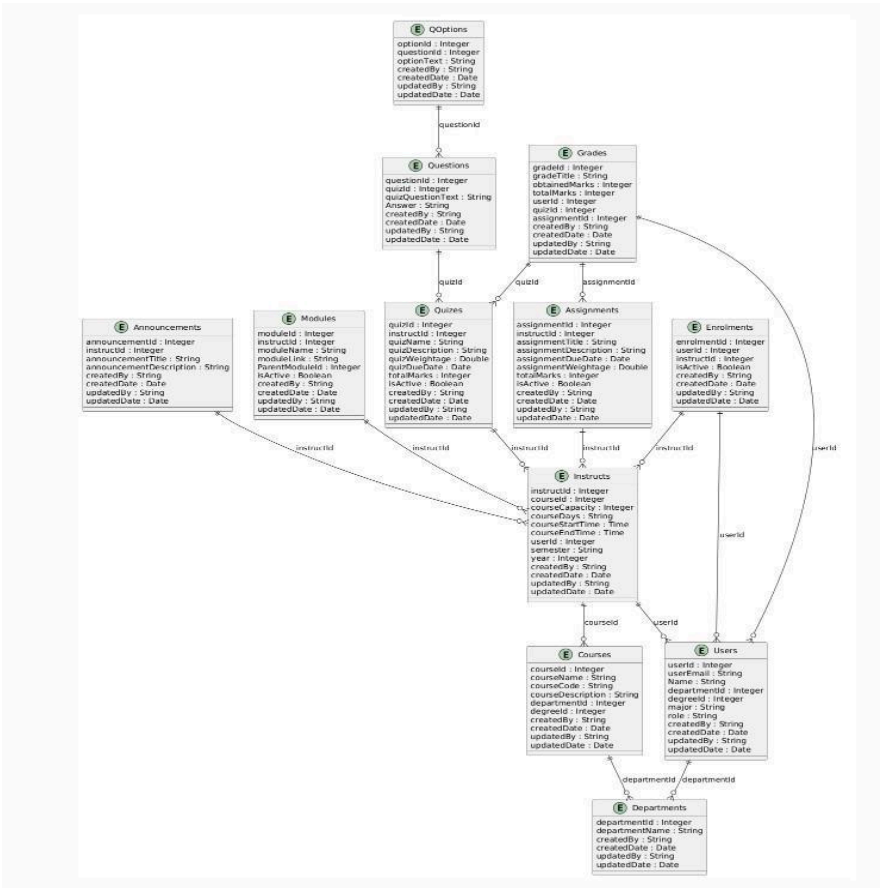
Part 4: Project Quality Assurance

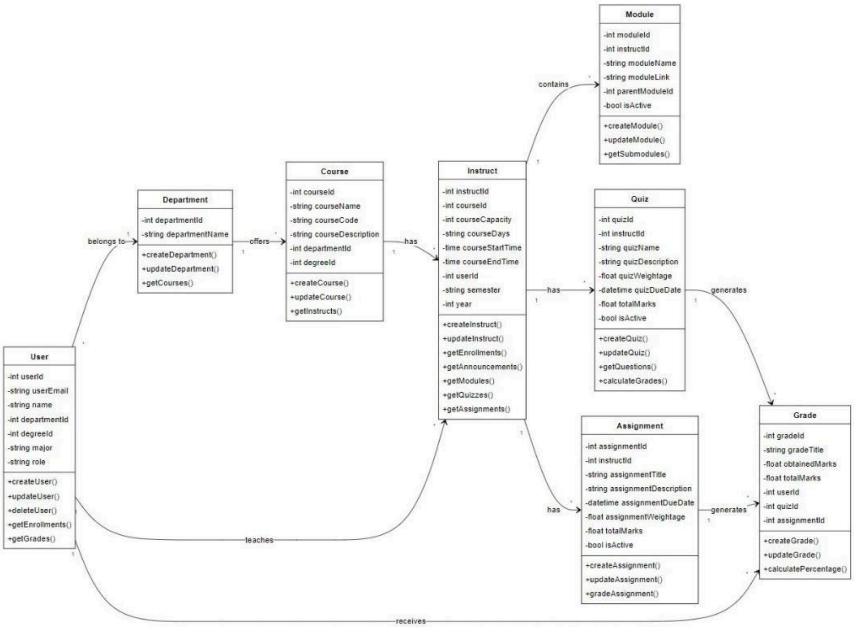
1. Quality Management Plan:
    - Code review process
    - Testing protocols
    - Documentation standards

2. Quality Metrics:
    - Code coverage (>80%)
    - Response time (<2 seconds)
    - Bug resolution time
    - User satisfaction scores

3. Quality Control Measures:
    - Automated testing
    - Peer reviews
    - Performance monitoring
    - Security audits

Part 4: Design





Part 5: Implementation

Tools:

- Visual Studio Code, IntelliJ IDEA: IDEs for frontend and backend development.
- GitHub: Version control.
- AWS: Cloud hosting (RDS for PostgreSQL and EBS for frontend & backend deployment).
- Jira: Task management for Scrum sprints.

Frameworks:

- Spring Boot: Backend development with APIs, JPA, and security.
- Next.js: Frontend with server-side rendering for better performance.

Libraries:

- Material UI: Responsive UI design.
- React Router: Client-side routing.
- Axios: HTTP request handling.
- JUnit & Jest: Testing backend and frontend.
- OAuth & JWT: Secure authentication and session management.

Part 6: Testing

Unit-testing with Jest:

The component has been thoroughly tested using Jest and React Testing Library.

The test suite includes the following implemented test cases:

1. Basic Rendering
    a. Verifies that the announcement page renders with the correct title
    b. Confirms the presence of "Course Announcements" text
2. Role-Based Access Control
    a. Validates that professors can see the "New Announcement" button
    b. Confirms that students cannot see the "New Announcement" button
3. Data Display
    a. Verifies announcements are properly displayed when data is available
    b. Tests handling of empty announcement data
    c. Validates course information display (course code, name)
4. Component Integration
    a. Properly integrates with Material UI ThemeProvider
    b. Correctly uses mock data for announcements service
    c. Handles navigation and routing through mocked hooks
    d. Integrates with next-auth for session management

5. Mock Implementation
    a. Uses mock announcement data with full course and user details
    b. Implements mock services for announcement fetching and saving
    c. Mocks next-auth useSession hook for authentication testing
    d. Mocks next/navigation hooks for routing functionality

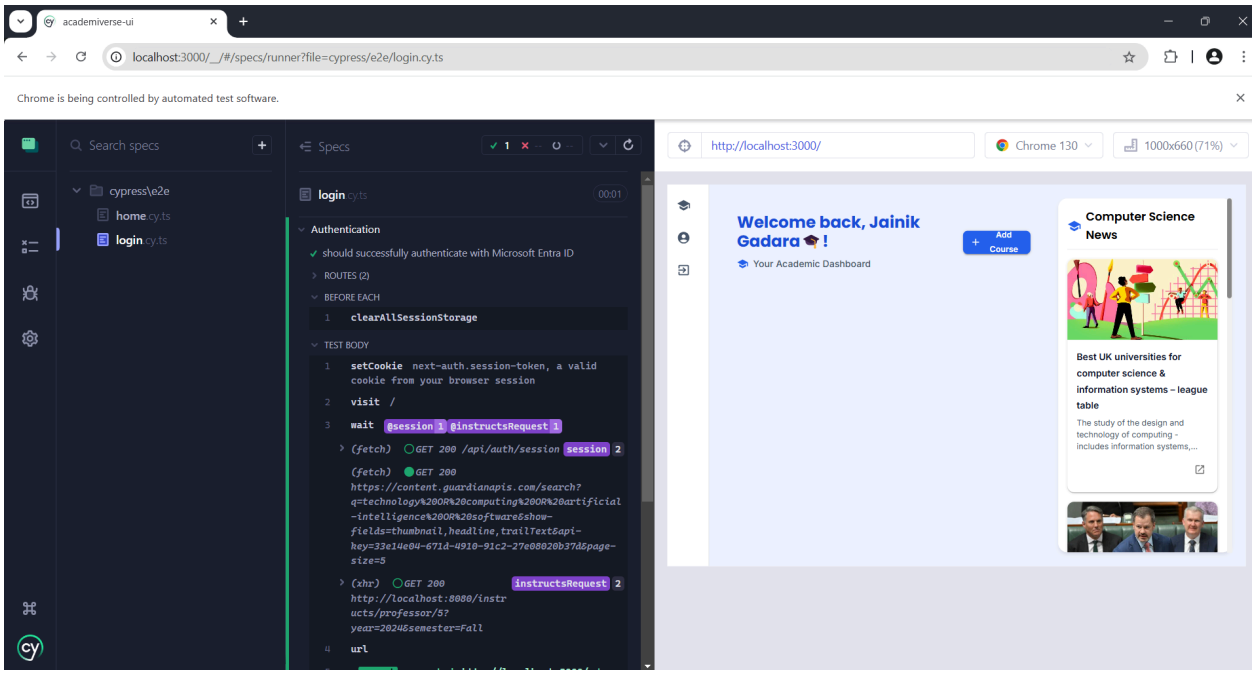
```
> academiverse-ui@0.1.0 test
> jest

PASS src/unit-test/announcement.test.js (34.892 s)
  AnnouncementPage
    √ renders the announcement page with title (254 ms)
    √ shows new announcement button for professors (63 ms)
    √ hides new announcement button for students (66 ms)
    √ displays announcements when data is available (56 ms)
    √ handles empty announcement data (43 ms)

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        37.258 s
Ran all test suites.
```

Automated testing with Cypress

1. Session Management
    a. Using clearAllSessionStorage() before each test ensures a clean state
    b. This prevents previous login states from affecting new tests
2. Authentication Flow
    a. The code uses a custom cy.login() command (presumably handling Microsoft Entra ID authentication)
    b. Tests the complete authentication flow from visit to successful login
3. API Mocking
    a. Uses cy.intercept() to mock API responses
    b. Particularly useful for handling dependent services (like the instructs endpoint)
    c. Helps create consistent test conditions
4. Assertions
    a. Verifies successful login through multiple checks:
        i. URL validation (cy.url().should())
        ii. UI element presence (cy.get('[data-testid="welcome-message"]'))

b. Uses data-testid attributes for reliable element selection

5. Wait Handling
    a. Implements cy.wait() to ensure asynchronous operations complete
    b. Waits for specific intercepted requests (@session, @instructsRequest)



User Acceptance Testing

User acceptance Testing				
Test no.	**Test cases**	**Expected results**	**Actual results**	**Status**
1	Grade Entry Workflow	Professor can enter and save student grades for assignments	Professor successfully enters grades and system saves them	Pass
2	Quiz Taking Experience	Student can take quiz, answer questions and submit	Student completes quiz and receives confirmation	Pass
3	Assignment Submission	Student can upload and submit assignment files	Assignment submission successful with file upload	Pass
4	Grade Viewing	Students can view their grades and overall performance	Grades displayed with percentage calculations	Pass
5	Quiz Creation	Professor can create quiz with questions and options	Quiz created and saved successfully	Pass

Integration Testing

	Integration Testing			
Test no.	**Test cases**	**Expected results**	**Actual results**	**Status**
1	Assignment Service Integration	Assignment submission API endpoints work correctly with frontend	Assignment submission and retrieval working	Pass
2	Grade Calculation Integration	Grade calculations sync between frontend and backend	Grade calculations accurate and consistent	Pass
3	Quiz Submission Flow	Quiz answers are properly recorded and scored	Quiz submission flow complete	Pass
4	File Upload Integration	File upload service properly handles assignment submissions	File upload and storage working	Pass
5	Grade Update Sync	Grade updates reflect immediately across all views	Grade updates sync properly	Pass

System Testing

	System Testing			
Test no.	**Test cases**	**Expected results**	**Actual results**	**Status**
1	Performance Testing	System handles multiple simultaneous quiz submissions	Smooth performance under load	Pass
2	Security Testing	Only authorized users can access grade management	Role-based access control working	Pass
3	Data Integrity	Grade calculations remain accurate across all views	Consistent grade calculations	Pass
4	Error Handling	System properly handles and displays error messages	Error handling implemented across components	Pass
5	Session Management	User sessions maintained properly across operations	Session management working with token handling	Pass

Part 7: Maintenance

Future Improvements:

1. Mobile application development
2. Advanced analytics dashboard
3. Integration with video conferencing
4. Machine learning for personalized learning
5. Expanded API capabilities
6. Enhanced reporting features

Part 8: Roles and Responsibilities

1. Ekta Patel: Scrum Master - Oversees project progress, manages daily scrum meetings, resolves team blockers.

2. Param Shah: Frontend Developer - Responsible for developing the user interface (UI) and ensuring responsive design.

3. Hetvi Joshi: Backend Developer - Implements core functionalities including course management, quiz features, and grading system.

4. Jainik Gadara: Database Architect - Designs and manages the Postgres database structure and handles data integrity.