Comprehensive Exercise Report

Abdullah Arslan – 250ADB070

This project was completed individually. I received permission to work alone during Week 4. At the time, I was not informed about the requirement to include a team number or section ID. Please consider this when evaluating the submission.

Requirements / Analysis

The goal of this project was to create a web platform where students can anonymously submit their course projects and vote for other submissions. The need for anonymity and fairness in peer evaluation led to this project idea.

User Roles:

- Students: Submit projects, vote for others
- Instructor: Monitor results (optional)

Problem Solved:
In classroom settings, peer voting can be biased. An anonymous system encourages honest feedback and reduces favoritism.

Journal (W2)

In the second week, I finalized the problem statement and selected the SDLC methodology. I decided to follow the Incremental Model to develop and test the system in multiple working versions.

I divided the features into increments:

1. Project submission
2. Voting
3. Ranking and UI polish

Initial assumptions:

- Around 50 users
- Cookie-based vote limitation

Software Requirements

Functional Requirements

- Users can submit a project with a title and URL
- Projects are listed publicly
- Users can vote on projects
- Vote counts are displayed and updated

Non-Functional Requirements

- Application must be responsive (mobile and desktop)
- Backend: Node.js with Express
- Database: MongoDB
- Vote limitation handled via browser cookies
- Dockerized setup for easy deployment

Black-Box Testing

Black-box testing was applied after each increment. The system was treated as a “black box,” with inputs and expected outputs defined without knowledge of internal code logic.

Test Methods:

- Manual form submission tests
- Cookie-based vote tracking
- Testing different browsers
- Project list validation

Journal (W4)

In Week 4, I completed the voting module, allowing users to vote anonymously. The biggest challenge was preventing duplicate votes without user login.

Vote limits were enforced using cookies. However, this approach can be bypassed using private/incognito browser windows. I noted this limitation in documentation.

I also tested:

- Valid vs. invalid submissions
- Double voting attempts
- Vote count accuracy

Black-Box Test Cases

Test Case	**Input**	**Expected Output**	**Result**
Submit valid project	Title + URL	Project appears in list	✅
Submit empty project	No title or URL	Error message shown	✅
Vote on a project (1st time)	Click “Vote”	Vote count increases by 1	✅
Vote again (same browser)	Click “Vote” again	Vote not registered	✅
Vote from incognito browser	Open incognito, click “Vote”	Vote registered again (duplicate)	⚠
View project list	Access main page	Projects sorted by votes	✅

Design

The application follows a simple client-server model:

System Architecture:

- **Frontend:** HTML, CSS,
- **Backend:** Node.js + Express
- **Database:** MongoDB
- **Deployment:** Docker (local)

Key Components:

- Submission module
- Voting logic with cookie tracking
- Project listing and vote-based sorting
- Basic static frontend with responsive design

Journal (W6)

In Week 6, I built the **project listing and sorting functionality**. The frontend was improved to be mobile-friendly, and projects were sorted by vote count in descending order.

New improvements:

- Inline error messages for form validation
- Simplified CSS for responsiveness
- Sanitization of user input before storing

This completed the third increment.

Software Design

- Backend routes:
  - POST /submit – Submit a project
  - POST /vote/:id – Register a vote
  - GET /projects – Fetch all projects
- Vote logic: Vote count updated only if the user hasn't already voted (checked via cookie)
- Static frontend: Projects rendered with vote buttons

Implementation

Technologies Used:

- Node.js
- Express.js

- MongoDB + Mongoose
- Handlebarsd/CSS
- Docker / Docker Compose

Journal (W8)

Week 8 focused on finalizing the entire project pipeline. I completed:

- Error handling in backend routes
- Form validation
- Final CSS tweaks
- Dockerization and testing on a clean local environment

All components were integrated and working.

Implementation Details

The application was tested on:

- Google Chrome
- Firefox
- Edge

Features completed:

- Project submission
- Voting once per browser
- Live project list
- Deployment-ready Docker setup

Testing

Final Testing Results

Area	**Tested On**	**Result**
Form Validation	Chrome	✅
Vote Count Updates	Chrome	✅
Duplicate Voting	Regular mode	✅ Blocked
Duplicate Voting	Incognito mode	⚠ Passed
Docker Deployment	Localhost	✅

Journal (W10)

In the final week, I:

- Finalized the UI
- Recorded the demo video
- Prepared the presentation slides
- Validated all features
- Cleaned up the code and added README

Peers also tested the app and gave feedback.

Testing Details

The application was tested manually after each development increment using black-box methods.

Tests were conducted across:

- Different browsers (Chrome, Firefox)

- Normal vs incognito sessions

Focus areas:

- Form validation

- Cookie-based vote limitation

- Vote counting

- Page refresh handling

Two classmates also tested the system and verified core functionality.

Presentation

The project was presented with slides covering each SDLC stage:

- Motivation
- SDLC choice
- Requirements
- Design
- Incremental implementation
- Testing
- Challenges and Lessons

Video Link: https://www.youtube.com/watch?v=_pEmU8maGWQ

Preparation

Throughout the course, I documented progress weekly through journals. Before final submission, I compiled all required artifacts:

- Finalized and proofread the project report

- Created slide deck according to SDLC structure

- Practiced the presentation

- Uploaded all materials to a public GitHub repository

- Verified visibility, access, and content completeness

I ensured that each requirement was satisfied and aligned the submission timeline with the official deadline.

Grading Rubric Compliance

All deliverables outlined in the final project rubric were completed and submitted:

- ✓ Comprehensive Report with mandatory sections
- ✓ Weekly journals (W2, W4, W6, W8, W10)
- ✓ Black-box test cases
- ✓ Software Design diagrams and explanations
- ✓ Complete source code in GitHub
- ✓ Presentation slides covering all required topics
- ✓ Recorded video presentation
- ✓ Public GitHub repository with README, report, and slides