CSE 360

Team Project Phase 3 Report

Team 5

Team Member Names:

1. Jad Khayyati
2. Joshua Wright
3. Kyle Ferolito
4. Steven Grisham
5. Jarod Wagner

1. The List of User Stories to be Implemented

TEAM MEMBER	USER STORIES TO IMPLEMENT
Jad Khayyati	- As a reviewer, I can create, view, update, and delete reviews for individual answers. - As a reviewer, I can create reviews with ratings 1-5 to show quality of answers. - As a reviewer, I can edit my review content and rating post upload. - As a reviewer, I can delete my reviews.
Joshua Wright	- As a student or reviewer, I want to send and receive private messages with other students and reviewers. - As a user, I want to give and receive feedback on reviews or answers. - As a reviewer, I want to view and manage a list of all my reviews. - As a reviewer, I want to create and edit my profile with my name, expertise, and other details so that students can see my qualifications when reading my reviews.
Kyle Ferolito	- As a student, I can request to become a reviewer. - As an instructor, I can view and approve or reject requests from students to become reviewers. - As an admin, I can promote users to the instructor role. - As an admin, I can see a list of all users and their corresponding role - As a user, I can see a review page for a given answer
Steven Grisham	- As a student, I can assign a weight to each reviewer, which will alter the order in which reviews are shown to me.
Jarod Wagner	- As a student, I can establish and maintain a list of trusted reviewers. - As a student, I can search exclusively for reviews from my trusted reviewers.

2. Implementation Plan

2.1. 3.Allocated Work

2.1.1. Joshua Wright

- Private messaging; 11/6; 11/7
- Feedback to reviewers and students; 11/6; 11/7
- Review list and management page; 11/7; 11/8
- Reviewer profile creation and update feature; 11/7; 11/8

2.1.2. Steven Grisham

- Reviewer weighting for students; 11/6; 11/7
- Code formatting & documentation; 11/7; 11/8
- Javadoc generation; 11/8; 11/9
- Review search sorted by weighted reviewers; 11/8; 11/9

2.1.3. Kyle Ferolito

- Admins can promote users;11/6; 11/7
- Admin page showing users and roles; 11/6; 11/7
- Student requesting reviewer role; 11/7; 11/8
- Instructors can approve or reject reviewer requests; 11/7; 11/8
- Review page; 11/7; 11/8

2.1.4. Jarod Wagner

- UML diagrams for new classes; 11/8; 11/9
- Write/update README file; 11/5; 11/6
- Reviewer curation for students; 11/6; 11/7
- Search by curated reviewers; 11/7; 11/8

2.1.5. Jad Khayyati

- Create Review.java and ReviewManager.java; 11/4; 11/5
- TBD; 11/7; 11/8
- Create scripts for standup meetings; 11/6; 11/9
- JUnit tests for review validation; 11/8; 11/9

2.2. Schedule of Standup Meetings and Notes

2.2.1. 11/2 – 11/9

- **11/5; 4 PM PST**
  – Joshua; Reviewed TP3, helped team clarify backend structure of reviews
  – Steven; Merged Jad’s backend work into main branch
  – Kyle; Went over requirements for TP3
  – Jarod; Volunteered to create UML diagrams
  – Jad; Provided initial iteration of review backend

- **11/6; 4 PM PST**
  - Joshua Wright; Completed private messaging backend and started feedback system UI
  - Steven Grisham; Implemented reviewer weighting system for students
  - Kyle Ferolito; Finished student reviewer request feature, preparing approval system
  - Jarod Wagner; Updated README and continued reviewer curation development
  - Jad Khayyati; Completed ReviewTest.java with all 12 tests passing
- **11/8; 3 PM PST**
  - Joshua Wright; Completed review list management, began reviewer profile creation
  - Steven Grisham; Finished code formatting and started Javadoc generation
  - Kyle Ferolito; Completed instructor approval system and began admin promotion features
  - Jarod Wagner; Finished reviewer curation, began search by curated reviewers
  - Jad Khayyati; Refined review implementation and finalized test documentation
- **11/9; 2 PM PST**
  - Joshua Wright; Completed reviewer profile creation and final testing for integration
  - Steven Grisham; Completed Javadoc generation and review sorting by weight
  - Kyle Ferolito; Completed admin page and final testing of review page
  - Jarod Wagner; Finalized UML diagrams and reviewed documentation for screencast
  - Jad Khayyati; Prepared JUnit test demonstration and finalized standup scripts

3. JUnit Tests to be Implemented

JUnit Test 1: testValidQuestionReview; Assigned to: Jad Khayyati
Verifies that a valid review can be created for a question with proper content, rating (1-5), and that the review correctly identifies itself as a question review.

JUnit Test 2: testValidAnswerReview; Assigned to: Jad Khayyati
Verifies that a valid review can be created for an answer with proper content, rating (1-5), and that the review correctly identifies itself as an answer review.

JUnit Test 3: testInvalidRatingTooLow; Assigned to: Jad Khayyati
Ensures that creating a review with a rating below 1 throws an IllegalArgumentException, validating the lower bound of the rating constraint.

JUnit Test 4: testInvalidRatingTooHigh; Assigned to: Jad Khayyati
Ensures that creating a review with a rating above 5 throws an IllegalArgumentException, validating the upper bound of the rating constraint.

JUnit Test 5: testEmptyContent; Assigned to: Jad Khayyati
Verifies that creating a review with empty string content throws an IllegalArgumentException, ensuring reviews must have meaningful content.

JUnit Test 6: testNullContent; Assigned to: Jad Khayyati
Verifies that creating a review with null content throws an IllegalArgumentException, preventing null pointer exceptions.

JUnit Test 7: testSetContent; Assigned to: Jad Khayyati
Tests the setContent() method to ensure review content can be successfully updated after creation.

JUnit Test 8: testSetInvalidContent; Assigned to: Jad Khayyati
Ensures that updating a review's content to an empty string throws an IllegalArgumentException, maintaining data integrity.

JUnit Test 9: testSetRating; Assigned to: Jad Khayyati
Tests the setRating() method to ensure review ratings can be successfully updated to valid values (1-5).

JUnit Test 10: testSetInvalidRating; Assigned to: Jad Khayyati
Ensures that updating a review's rating to an invalid value throws an IllegalArgumentException, maintaining rating constraints.

JUnit Test 11: testGetReviewedQuestion; Assigned to: Jad Khayyati
Verifies that a question review correctly returns the associated question object and returns null for the answer object.

JUnit Test 12: testGetReviewedAnswer; Assigned to: Jad Khayyati
Verifies that an answer review correctly returns the associated answer object and returns null for the question object.

JUnit Test 13: testSendPrivateMessage; Assigned to: Joshua Wright
Verifies that a user can successfully send a private message and that it is stored correctly in the database.

JUnit Test 14: testReceivePrivateMessage; Assigned to: Joshua Wright
Ensures that a recipient can retrieve received messages and that message content matches what was sent.

JUnit Test 15: testFeedbackSubmission; Assigned to: Joshua Wright
Checks that a reviewer can submit feedback on a review and that it links to the correct review ID.

JUnit Test 16: testReviewerProfileUpdate; Assigned to: Joshua Wright
Validates that reviewers can update their profile information and that changes persist in the database.

JUnit Test 17: testCreateReviewerRequest; Assigned to: Kyle Ferolito
Verifies that students can submit reviewer requests and they appear in the pending approval list.

JUnit Test 18: testInstructorApproval; Assigned to: Kyle Ferolito
Ensures instructors can approve or reject reviewer requests and the system updates correctly.

JUnit Test 19: testAdminPromotion; Assigned to: Kyle Ferolito
Checks that admins can promote users and their permissions update properly.

JUnit Test 20: testReviewerWeightCalculation; Assigned to: Steven Grisham
Checks that reviewer weight is calculated correctly.

JUnit Test 21: testReviewSortingByWeight; Assigned to: Steven Grisham
Checks that reviews are sorted by reviewer weight.

JUnit Test 22: testReviewSearchByKeyword; Assigned to: Steven Grisham
Checks that search by keyword returns matching reviews.

JUnit Test 23: testTrustedReviewerCuration; Assigned to: Jarod Wagner
Checks that a trusted reviewer can be added and retrieved.

JUnit Test 24: testSearchByCuratedReviewer; Assigned to: Jarod Wagner
Checks that reviews can be filtered by trusted reviewers.

JUnit Test 25: testUMLDiagramValidation; Assigned to: Jarod Wagner
Checks that UML diagrams match the class structure.

JUnit Test 26: testReviewCreation; Assigned to: Jad Khayyati
Checks that a review can be created.

JUnit Test 27: testReviewEditing; Assigned to: Jad Khayyati
Checks that a review can be updated.

JUnit Test 28: testReviewDeletion; Assigned to: Jad Khayyati
Checks that a review can be deleted.

JUnit Test 29: testReviewAverageScore; Assigned to: Jad Khayyati
Checks that average review scores are calculated.

JUnit Test 30: testReviewIntegration; Assigned to: Jad Khayyati
Checks that review features work together.

4. Implementation

4.1. URL and access path to the source code and the architecture and design documents

- GitHub Repository: https://github.com/JoshWright22/CSE360Group5
- Design documents are stored in the root of the repository. Direct links are as follows:
    - UML Diagram for ReviewManager.java: https://github.com/JoshWright22/CSE360Group5/blob/main/ReviewManagerUML.PNG
    - UML Diagram for Review.java: https://github.com/JoshWright22/CSE360Group5/blob/main/ReviewUML.png
    - UML Diagram for ReviewerProfileManager.java: https://github.com/JoshWright22/CSE360Group5/blob/main/ReviewerProfileManagerUML.PNG
    - UML Diagram for ReviewerProfile.java: https://github.com/JoshWright22/CSE360Group5/blob/main/ReviewerProfileUML.PNG
    - UML Diagram for Message.java: https://github.com/JoshWright22/CSE360Group5/blob/main/message_UML.PNG

4.2. URL and access path to a folder of Javadoc HTML Files

- All Javadoc-related files are in `<repo root>/doc`.
- Path to Javadoc index: https://github.com/JoshWright22/CSE360Group5/blob/main/doc/index.html

5. GitHub Repository

5.1. The URL to access the GitHub Repository with a ReadMe file

- GitHub Repository: https://github.com/JoshWright22/CSE360Group5

5.2. Access to the Team's Solution for TP3

- GitHub Repository: https://github.com/JoshWright22/CSE360Group5

5.3. Access to Screencasts

5.3.1. Screencast for how the new code works, how it was implemented, how it was validated using JUnit testing, and how it works from the various role-players that would use the application

- All relevant screencasts are stored in `<repo root>/screencasts/TP3`.
- Direct link to Code Walkthrough screencast: https://github.com/JoshWright22/CSE360Group5/blob/main/screencasts/TP3/screencast1_code_walkthrough_final.mp4

5.3.2. Screencast showing and explaining the alignment from the stakeholders' needs as described in the product vision through the architecture and detailed design to the validation of the code using JUnit.

- All relevant screencasts are stored in `<repo root>/screencasts/TP3`.
- Direct link to Program Demo screencast: https://github.com/JoshWright22/CSE360Group5/blob/main/screencasts/TP3/screencast2_program_demo_final.mp4

5.3.3. Screencasts from each Standup Meeting

- All relevant screencasts are stored in `<repo root>/screencasts/TP3`.
- Direct link to Standup Meeting 1: https://github.com/JoshWright22/CSE360Group5/blob/main/screencasts/TP3/meeting1_final.mp4
- Direct link to Standup Meeting 2: https://github.com/JoshWright22/CSE360Group5/blob/main/screencasts/TP3/meeting2_final.mp4
- Direct link to Standup Meeting 3: https://github.com/JoshWright22/CSE360Group5/blob/main/screencasts/TP3/meeting3_final.mp4
- Direct link to Standup Meeting 4: https://github.com/JoshWright22/CSE360Group5/blob/main/screencasts/TP3/meeting4_final.mp4

6. Appendix A: Credit Sheet

This appendix lists the members of the team and includes a description of the contribution that team member has made to this submission. Each team member must provide text for their contribution at a team meeting just prior to the submission of this deliverable and the entire team must agree. If a team member fails to provide this information and/or does not participate in the agreement process to fill out this table, that member will receive no credit for the submission of this deliverable.

Team Member Name	Contributions
Jad Khayyati	Created Review.java Created ReviewManager.java Created ReviewTests.java (12 Junit tests) Provided documentation for all the Junit tests Integrated ReviewManager into StartCSE360 Created scripts/notes for our standup meeting
Joshua Wright	Implemented private messaging system Developed feedback feature for reviews Created reviewer profile management Built review list and management page Tested and debugged new user features Assisted team with backend integration
Kyle Ferolito	Implemented student reviewer request system Developed instructor approval/rejection features Created admin promotion and role management page Designed and tested review display page Integrated admin and instructor functionality Assisted with system testing and screencast preparation
Steven Grisham	Implemented reviewer weighting system for students Added sorting of reviews by assigned weight Performed code formatting and documentation cleanup Generated Javadoc for the full project Assisted with review search and filtering logic Helped finalize documentation for submission
Jarod Wagner	Created and updated UML diagrams in Astah Implemented reviewer curation and trusted reviewer lists Added search by curated reviewers Updated README with TP3 features and documentation Coordinated final review of design alignment Assisted in preparing and organizing screencasts

7. Appendix B: Team Norms

No changes have been made to Team Norms.