

Computer Science — Capstone Project Documentation

ClassConnect

Knight Foundation School of Computing and Information Sciences (KFSCIS) — Florida International University

Instructor: Masoud Sadjadi | Version: 2026 Spring

Poster: 36"×48" horizontal • Videos: Intro & Demo • Presentation: 10–15 min • Scrum: disciplined, evidence-based.

This template is ACM-aligned and maps to the Capstone grading rubric (Project Documentation = 10%).

ACM Student Outcomes (O1–O6)

Outcome	Description
O1. Problem Analysis & Requirements	Elicit and analyze user/stakeholder needs; define constraints and success criteria.
O2. System Design & Implementation	Design solutions using appropriate abstractions, patterns, and tools; implement maintainable code.
O3. Experimentation, Testing & Evaluation	Devise evaluation methods; collect evidence; interpret results to improve the system.
O4. Communication & Collaboration	Communicate effectively with technical and non-technical stakeholders; collaborate in teams.
O5. Professional, Ethical, Legal & Societal	Recognize responsibilities; address ethics, security, privacy, accessibility, and sustainability.
O6. Algorithmic Thinking & Complexity	Discipline-specific depth outcome; addressed in dedicated sections below.

Outcome & Rubric Crosswalk (Checklist)

Use this to ensure your documentation and artifacts demonstrate each outcome and satisfy the rubric.

Mark each ☐ when complete.

Outcome	Where It Appears in This Document	External Evidence (Links/Appendices)	Status (☐ / ☒)
O1	§2 Problem & Requirements; §3 System Overview	Appendix A (User Research), Backlog, User Stories	☒
O2	§3 System Overview & Architecture; §5 Implementation Notes	Repo structure, Code, CI status badge	☒
O3	§6 Testing & Evaluation	Test reports, coverage, performance dashboards	☒
O4	§1 Executive Summary; §9 Limitations & Future Work	Poster, Slides, Videos	☒

O5	§7 Security/Privacy/Compliance	Threat model / Model card / SBOM	☐
O6	§4 Discipline-Specific Depth	Artifacts noted in §4	☒

1. Executive Summary

ClassConnect solves the challenge of connecting FIU students and instructors with each other and their classmates both before and after class using existing tools and technology. Traditional communication methods available to FIU including Canvas Discussion Boards, Email, and WhatsApp do not allow students and instructors to engage in real-time communication, have a clear way of knowing when or if each question from a student has been asked, remain anonymous, and see all of the questions that have been submitted. Because of this, students often do not ask their questions for fear of looking dumb or asking too many questions and instructors lose sight of the growing confusion or issues students may have.

ClassConnect is a centralized, real-time assigned question board specifically created for classroom use at FIU. Students can post questions anonymously or otherwise, collect votes on submitted questions, develop a threaded conversation around particular questions, and receive notification from instructors when a question has been answered. Instructors also benefit from a streamlined central panel dashboard that displays active questions that need answering; what category a question falls into; and what level of priority a question is.

ClassConnect was developed as a Canvas-integrated application to be used with a dedicated Web Portal and enhances inside and outside the classroom instruction, instructor awareness, and assists in creating a Panther culture of interactive learning. The team has developed a fully working product that demonstrates how students and instructors will interact with each other through the platform's dashboard.

2. Problem & Requirements (O1)

Context and stakeholders:

- Primary Personas: FIU students, FIU instructors
- Secondary Stakeholders: FIU academic departments, instructional designers, student support staff

Personas:

- Student: Wants to ask questions (anonymously in some cases), receive quick answers, and avoid response board clutter.
- Instructor: Needs a singular platform to view all student questions, with the ability to mark status and respond efficiently.

Functional Requirements:

1. Post questions (named or anonymous)

2. Upvote questions
3. View responses and threaded replies
4. Instructor marking: active, resolving, and/or answered
5. Real-time updates
6. Category filters (technical, conceptual, assignment, general, project help)
7. Distinct student and instructor views
8. Canvas integration (icon/module link)

Non-Functional Requirements:

- Usability and clarity for in-class use
- FIU brand-consistent UI
- Fast response times
- Privacy for anonymous users

Constraints:

- Integration with canvas through LTI link
- Anonymity support
- Proper support for desktop

Success Criteria:

- Students ask more questions vs. current tools
- Instructors can answer questions faster
- Instructors don't get the same questions repeated multiple times over the course
- Reduced confusion during and after class
- Clear, organized question board with minimal clutter

Include a prioritized backlog:

1. Posting both anonymous and non-anonymous questions
2. Dashboard for instructors with status marking
3. Upvote mechanism
4. Discussion in threads
5. Categories of questions
6. Integration of Canvas
7. System for searching and filtering
8. Tools for instructor moderation
9. Flow of user onboarding
10. Improvements in accessibility

3. System Overview & Architecture (O2)

Architecture Description:

The Canvas-integrated web extension known as ClassConnect allows users to access the ClassConnect portal by clicking on the "ClassConnect" button. ClassConnect will allow:

- Students to ask questions and upvoting other students' questions as well as view the threaded replies.
- Instructors to moderate students' posts, categorise and set the status of the posts to open/closed/answered.

Key technologies:

React

PRISMA

[Node.js](#)

PostgreSQL

4. Discipline-Specific Depth (O6)

This project demonstrates core software engineering principles through the design of a real-time classroom question platform. We applied component-based architecture using React to create modular and reusable interfaces for both students and instructors. The system includes fully implemented CRUD operations, validation, error handling, and relational querying. These operations form the backbone of the platform's dynamic and interactive functionality. State management and data flow were carefully structured to support real-time updates and ensure consistency across components. The system was also designed with scalability and integration in mind.

5. Implementation Notes (O2)

Prototype allowed for efficient implementation:

- Frames
- Components
- Reusable UI elements
- Interactions
- Prototyping behaviors (links, transitions, overlays)

6. Testing & Evaluation (O3)

Overview of Testing:

The system was tested using API testing tools such as Thunder Client. Each endpoint was validated using both successful and failure scenarios, including invalid inputs, missing records, and duplicate actions. CRUD operations were verified for all major entities, and relational queries were tested to ensure correct

data retrieval. The backend was confirmed to handle edge cases and return appropriate HTTP status codes.

Usability Testing:

With a number of people, we ran basic usability testing. While using the prototype, each participant was required to do essential activities.

Tasks Examined:

- Switching between question categories
- Finishing a complete user flow, such as making a question, choosing a choice, and submitting a form
- Recognizing interactive elements and buttons
- Going back to earlier screens

Important Findings:

- Users rapidly grasped the whole flow.
- Certain screens required more precise button positioning or labeling.
- A small number of participants clicked on non-interactive sections, exposing portions with ambiguous indicators.

Modifications:

- Added more readable text labels beneath symbols
- Increased button size and spacing
- For readability, the color contrast was changed.
- Transitions in the prototype were smoothed out.

Peer Feedback:

- Increased alignment consistency
- Improved screen-to-screen navigation
- Enhancing button and header hierarchy

7. Security, Privacy, Accessibility & Compliance (O5)

The solution promotes a more equitable and inclusive classroom by giving all students, especially those who may feel hesitant a safe way to participate through optional anonymity. It improves fairness by ensuring that answers are shared class-wide, reducing information gaps and repeated questions.

From a privacy and security standpoint, the platform limits personal data exposure, supports anonymous posting, and would follow secure authentication practices through Canvas integration. Data handling would prioritize protecting student identity and preventing misuse.

Accessibility is addressed through a clean, intuitive interface designed for ease of use in fast-paced classroom settings, with plans to support responsive layouts and inclusive design standards.

8. Deployment & Operations

The frontend is built using React and communicates with a Node.js/Express backend through RESTful APIs. The backend uses Prisma ORM to interact with a PostgreSQL database. Core entities include Users, Courses, Questions, Replies, and Votes. The system is designed to support modular expansion and scalable data handling, with clearly defined routes and structured data relationships.

link to detailed guides in the appendices.

9. Limitations & Future Work

- Canvas link integration

Future Work:

Moving forward, our primary goal is to develop this platform as a direct extension within Canvas, allowing students and instructors to access it seamlessly within their existing workflow. By integrating more deeply into the tools already used in the classroom, we aim to create a more streamlined and efficient experience.