SmartClass

Design Document

Team Number: sddec25-10
Client/Advisor: Md Maruf Ahamed

Team Members:

Josh Dwight - Backend/Frontend
Logan Pfantz - Frontend/Backend
Matthew Gudenkauf - Frontend
Michael Becker - Frontend /Backend
Michael Geltz - Backend
Ryan Lin - Frontend/Backend

Executive Summary

SmartClass addresses an important issue in modern education: low student engagement in large classroom settings. Instructors often find it difficult to gauge student understanding or encourage participation, particularly in larger lecture halls. Students, especially those who are shy or seated far from the instructor, may feel intimidated to ask questions. This disengagement can lead to missed learning opportunities and a lack of real-time feedback. SmartClass seeks to solve this by providing a web-based platform that encourages interactive, anonymous, and real-time engagement during lectures.

Key design requirements for SmartClass include accessibility, ease of use, real-time communication, scalability, and role-based functionality for students, instructors, and teaching assistants. Functional features include anonymous Q&A, live chat, polling, short quizzes, archiving class activities, and viewing participation statistics. Accessibility requirements ensure that the platform works well on low-end hardware and supports assistive technologies for students who may need it.

The system is built using modern technology stack: React.js for a responsive fronted interface, WebSockets for real-time chat features, and a RESTful API with MySQL backend to manage user data, class content, and archived participation logs. Git is used for version control, while Google Docs is used to facilitate planning and documentation. Our team has adopted an Agile methodology to break the project down into manageable sprints, allowing for incremental development and weekly feedback from our client.

As of now, we have implemented core features such as user authentication, role-specific dashboards, class creation/join functionality, live chat, and the anonymous Q&A modules. Backend infrastructure is in place to support secure data handling and real-time updates, and the frontend components are responsive.

Initial usability testing has shown that the design is intuitive for our target users which are students and instructors. Features like anonymous participation directly address user needs gathered from our journey maps and personas. Weekly feedback sessions with our client confirm that the platform is on track and aligning with the project goals.

Next steps include completing the other modules such as the leaderboard and analytics features, refining UI/UX based on other user testing, and also conducting performance tests to ensure scalability in large classroom environments. Additional improvements will focus on accessibility, such as including customizable UI/UX. With continued iteration and testing, SmartClass is evolving to be a more reliable, inclusive, and scalable tool for increasing classroom engagement.

Learning Summary

Development Standards & Practices Used

For this project, the following standards were used:

- Secure Software Development: The OWASP secure development practices were followed to ensure the safety of the website
- Agile development: Weekly scrum meetings were used to check progress across the team
- Rigorous Testing: The code went through testing before being deployed

Summary of Requirements

- Role-based login and dashboards for students, TAs, and instructors
- Anonymous Q&A and live chat functionality
- Polling and quiz features with real-time updates
- Course management (creating courses, deleting courses)
- User interface with intuitive navigation
- Content management (uploading, downloading and previewing course materials)

Applicable Courses from Iowa State University Curriculum

- Com S 3090 - Software Development practices and project management
- Cybsc 4300 - Secure programming practices
- Com S 2270 & 2280 - Object-Oriented Programming and Data Structures
- Engl 3140 - Technical Writing
- Com S 3630 - Introduction to Database Management Systems
- S E 3290 - Software Project Management

New Skills/Knowledge acquired that was not taught in courses

- React Programming

Table of Contents

1. Introduction

1.1 Problem Statement

The SmartClass project aims to address the challenge of student engagement in large classrooms. Instructors often struggle to facilitate active participation, as students (especially the ones seated farther back) find it difficult to ask questions or contribute to class discussions. Traditional methods like verbal Q&A sessions or raising hands can be inefficient in large settings. To solve this problem, SmartClass will be a web/app-based interactive learning platform that allows students to ask questions (anonymously or with their names), participate in quizzes, and engage in class discussions. Instructors can use the platform to initiate polls, post questions, and provide real-time feedback. The system will also generate participation data that instructors can use for grading purposes. This platform aims to make learning more inclusive, accessible, and engaging for all students, regardless of their physical location in the classroom.

1.2 Intended Users

The intended users of this product are those involved in a classroom environment. Its most notable uses will be in a large lecture hall with many students, but there are use cases in smaller settings as well. The goal is to increase student participation and give teachers powerful tools to address the concerns of their classroom.

User Groups:

(1) Teachers

Characteristics:

- Leader in classrooms of over 100 students
- Passionate about teaching
- Lacks the time to address each student's questions
- Supportive of others who struggle to learn

Needs:

- A way to receive and address questions posed by students in a large classroom environment
- A centralized place to take attendance, annotate live lectures, post grades, save coursework and help students learn efficiently
- Methods to track student progress and attendance
- Tool to create polls and quizzes for students to participate in
- Way to organize the coursework in one place.

Benefit:

- SmartClass provides an easy way of tracking student participation and attendance in large classrooms
- The ability to upload documents provides a centralized location for all in class activity
- Checking student understanding of the lecture material is simple with quizzes and polls
- Ability to print and export data from SmartClass gives an easy to way transfer data

(2) Students

Characteristics:

- All types of personalities including: shy, bored, lazy, attentive, resilient
- Intelligent and ready to learn
- Young and inexperienced

Needs:

- A way to participate in class discussions
- Asks questions without fear
- Engage in learning activities beyond physical classroom constraints
- A way to access previous class materials

Benefit:

- Enables active participation
- Provides access to an archive of past questions and discussions
- Provides a more interactive learning environment

(3) Teaching Assistants

Characteristics:

- Wants to help students learn
- Eager to answer questions
- Analytical mind
- Struggles with having enough time for everything

Needs:

- Way to effectively analyze student participation
- Centralized system to answer student questions
- Recognize when a student might need support

Benefit:

- Ability to answer questions outside of class hours
- No longer restricted to office hours for interactions with students
- Answer more questions in a shorted time
- Ability to create resources and upload them for students to use
- Increase productivity in less time

2. Requirements, Constraints, And Standards

2.1 Requirements & Constraints

2.1.1 Functional:

- A website interface
- Register page
  - Create accounts for students and teachers
  - Create account through SSO login with globus
- Login page

    - Separate account types for students, teachers, and TAs
    - Login with Globus
- Teacher privileges:
    - Create classes and generate a join code
    - Create quizzes for classes
    - Enable and disable quiz access immediately with the push of a button
    - Upload notes, images, videos, etc. as class material for students to view / download
    - Start an online guide during a lecture which updates automatically as the professor goes forward in slides
    - Access a logs page with polling results, quiz results, and live chat logs.
    - Can assign “Teaching Assistant” role to students
- Student privileges:
    - Join classes using a class code
    - Submit questions in a forum
    - Answer student questions and respond to posts
    - Can participate in live chat and related quizzes/polls.
- Forum-style posts about class activities
    - Allow users to post anonymously or not
- Live Chat
    - Allows students to ask questions during lecture with option of the question being anonymous or not
- Quizzes
    - Question types
        - Multiple choice
        - True/False
        - Fill in the blank
    - Grades
        - Saved for future use
    - Access set by teacher
        - Can decide when to open and restrict access to students
- Polls
    - Polls are created by teachers and responded to by students
- Upload documents
    - Supported file types
        - pdf, png, jpeg, mp4, mov
    - Documents stored in database in BLOB format.
    - Documents can be uploaded in the forum pages and Images can be uploaded in quizzes.

- Security
    - User data should be encrypted and handled safely

2.1.2 Resource:

- Server to host the website
- Database to host user information - Keeps track of main functionality of the project
    - Tables for users, courses, courses enrolled, material, announcements, forum posts
- Globus to track SSO login information through universities

2.1.3 Aesthetic:

- Usable, quick interface
- Professional looking UI
- Interface to improve user efficiency

2.2 Engineering Standards

Engineering standards are fundamental to the field as they ensure that a product not only meets requirements but also is safe. Standards allow for teams of engineers to work together by providing a base level of capability and functionality to a piece of software and devices. These standards can also dictate how a device communicates with other devices, allowing integration. Safety standards are necessary to reduce the risks of using devices and software. This serves to protect the public and avoid spreading harm. An unsafe product is usually less helpful and less desirable than a safer product.

Top industries often adopt these standards to send out the highest-quality products. Following these standards will help us perform the best software practices and create a high-end, high-quality product that satisfies our customers.

These standards are very important when creating a software application. Many of these will already be incorporated due to the standard development process. Creating a roadmap to develop our program effectively and constantly checking with our client about the quality of the project are natural parts of building a piece of software.

Security measures will be taken to protect user data, such as names and sign-in information. Following the standards for security in software and cybersecurity will help keep user information safe.

Standards for architecture design will help ensure the product is maintainable and understandable by others working on this project in the future. The following standards will give others an idea of what to look for and how to find the information they are looking for.

2.2.1 ISO Standards:

- ISO 12207 (Software Life Cycle Processes): This standard divides software processes into four main groups: agreement, organizational project-enabling, technical management, and technical processes. This is important in developing a software project as it acts as a roadmap for effective software development practices, leading to a higher quality product.
- ISO 8601 (Date and Time Format): This standard goes for consistency in date-time formatting. e.g., November 27, 2025, at 6:30 pm can be described as 2025-11-27 18:30:00:000. This will be an essential sddec25-10 standard for our project since keeping track of when something was posted will be a core component of our project.
- ISO 9000 (Quality Management): This standard checks for the quality of products and services and consistency to meet customer expectations. This is important because it provides a globally recognized standard to help deliver high-quality products and improve the process of delivering services consistently.

- ISO 9000:2015 (Quality Management Systems): This is the standard of fundamentals and vocabulary used in software products. It defines common terms like “customer,” “system,” and “policy.” By utilizing the terms used by the industry, outsiders looking at this project will be able to understand the goals and decisions behind the project. Common terminology reduces confusion among all involved and lets people convey more complex ideas succinctly.
- ISO 27000 (Information Security Management): This standard focuses on security and privacy protection. Following this standard is important because the customer’s data is important, and it should not easily be breached or stolen.
- ISO 9241 (Human-system Interaction): This standard focuses on the design of user interfaces for effectiveness, efficiency, and satisfaction. This standard allows SmartClass’s user experiences for both teachers and students to be improved throughout using SmartClass

2.2.2 IEEE Standards:

- IEEE 830 (Software Requirements Specification): This standard provides guidelines for writing clear, complete, and verifiable software requirements. For SmartClass, this ensures the functional requirements (such as quiz creation, live chat, and user authentication) and non-functional requirements (like performance, security, and usability) are clearly defined and maintained throughout development.
- IEEE 1016 (Software Design Description): This standard specifies the recommended structure and contents of software design documentation. This helps SmartClass organize development decisions, including the frontend interface, backend API design, and database design. This ensures future maintainers can easily understand and modify the system.
- IEEE 829 (Software Testing Documentation): This standard defines a set of documents for testing processes, including test plans, test designs, and test reports. Applying this standard helps ensure SmartClass is thoroughly tested to ensure every implementation works as intended.
- IEEE 1028 (Software Reviews and Audits): This standard outlines the review process for SmartClass’s feature implementations. This helps the SmartClass development team ensure quality throughout peer reviews before releasing an updated version of SmartClass.

3. Project Plan

3.1 Project Management/Tracking Procedures

The SmartClass team is following an Agile project management style because it allows us to iteratively build, test, and refine the SmartClass platform in collaboration with our client. Agile's emphasis on adaptability, team collaboration, and continuous feedback aligns well with the evolving needs of our classroom engagement solution.

To manage and track progress, the following tools will be used:

- Git: Git will be used for version control to manage and integrate code across all team members collaboratively. Each feature or bug fix is handled on a separate branch and merged into the main branch after review. We utilize Gitlab issues to delegate and break down work packages.
- Google Docs: Shared Google Docs is used to write our reports, maintain to-do lists, and record changes. This helps keep everyone up-to-date and provides a central location for documentation.
- Weekly Meetings: Weekly meetings with the client are held to discuss progress, receive feedback, and establish upcoming goals.

3.2 Task Decomposition

To build Smartclass, the overall problem was decomposed into key functional areas and corresponding subtasks. Each task represents a component of the final product and can be aligned with Agile sprints of iterative development.

1. Authentication and User Management
    - Create user registration and login functionality
    - Implement user roles (Student, Teacher, TA)
    - Secure user data (email & password hashing)
    - Enable profile management (password resets, etc)
    - Use Globus for institution logons.
2. Questions and Answers System
    - Build UI for students to post questions (text, image, etc)
    - Enable anonymous question posting
    - Allow replies/comments for each question
    - Implement media upload and playback support
    - Add categorization by context (lecture, homework, quiz, etc)
3. Polling and Quizzes Features
    - UI for instructors to create and launch polls/quizzes
    - Student response interface (real-time updates)
    - Store and track participation data
    - Export participation results for grading
4. Archiving and Retrieval
    - Store all Q&A and polls persistently in a database
    - Implement filtering/searching by category or date
    - Provide instructor and student views of the archive
5. Statistics and Leaderboard
    - Track the number of contributions per user
    - Display the leaderboard of the top 10 student contributors
    - Display TA contribution stats
    - Log quiz results and polling results, and live chat.
6. Backend Development
    - Design and implement a database schema (SQL)
    - Set up API endpoints
    - Ensure secure handling of media files
    - Implement data validation and error handling
7. Frontend Development
    - Set up a responsive UI framework (React)
    - Build navigation between views (auth, questions, polls, stats)
    - Integrate with backend APIs
    - Support mobile-friendly design
8. Testing
    - Unit tests for frontend and backend components
    - End-to-end system testing
    - Set up CI/CD pipeline

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

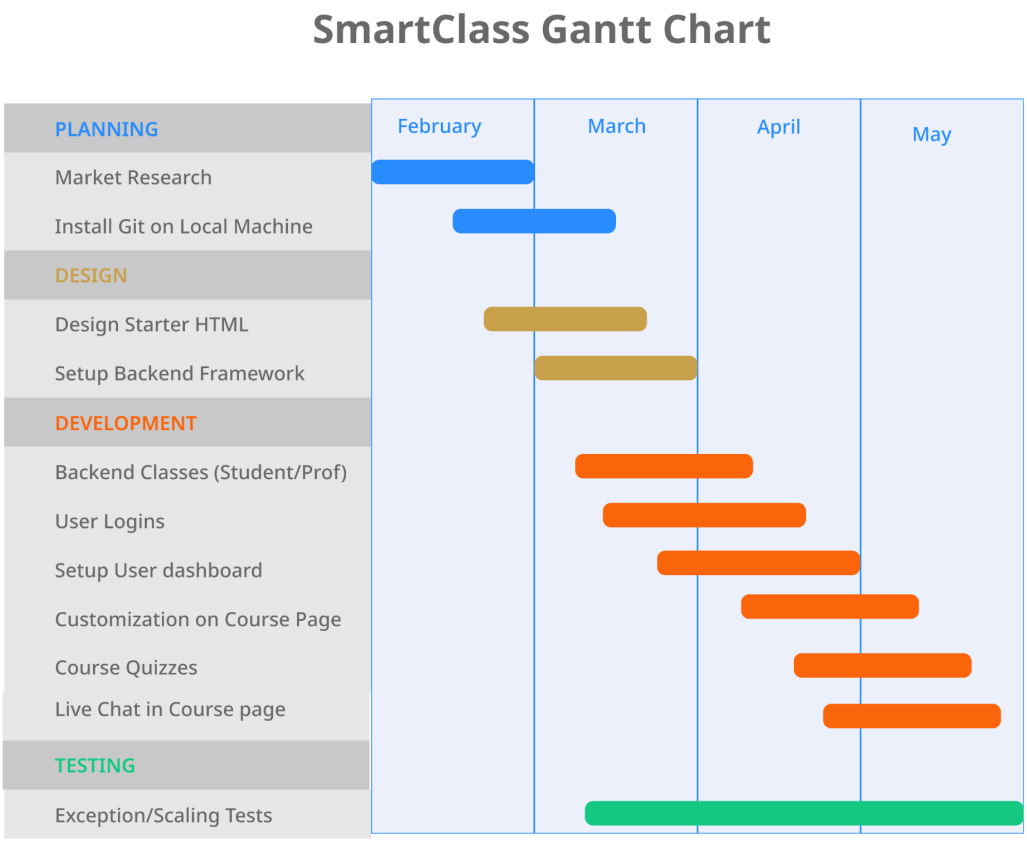
Key project milestones and corresponding metrics have been established to ensure timely and measurable progress on the SmartClass platform. These milestones are structured around the major components of the

application and aligned with our sprint schedule. Each milestone includes specific evaluation criteria to determine success.

1. Authentication & User Management
    - User login and registration are completed.
      Metric: Users can sign up and log in, and roles (Student, TA, Instructor) are correctly stored in the database.
    - Role-based access control implemented
      Metric: Students, TAs, and Instructors see different views and permissions.
2. Question & Answer System
    - Basic Q&A with text implemented
      Metric: Students can post and reply to text questions; data is stored and displayed correctly.
    - Media support enabled
      Metric: Students can upload and playback image, audio, and video content in posts or replies.
    - Anonymous posting and categorization are functional
      Metric: Students can choose anonymity; posts are sorted by type (e.g., lecture, lab, homework).
3. Polling and Quizzing
    - Poll creation and launch system completed.
      Metric: Instructors can create and launch polls using text/images.
    - Student responses and results displayed the implemented
      Metric: Real-time response tracking; instructors can download a CSV of participation.
4. Archiving and Retrieval
    - Archive system and basic search functional
      Metric: Q&A and polls are stored by date/category; can be retrieved and filtered by users.
5. Statistics and Leaderboard
    - Participation data tracking is complete
      Metric: Number of contributions logged per user.
    - Leaderboard for students and TAs implemented
      Metric: Top contributors displayed by rank and number of posts.
6. Frontend UI and UX
    - Responsive layout and navigation completed
      Metric: Pages render cleanly on different devices; navigation is smooth.
    - Client-approved UI/UX design polish
      Metric: Positive client feedback on usability and layout.
7. Testing and Deployment
    - The system passes unit and integration tests.
      Metric: All core features pass test cases without breaking existing functionality.

3.4 Project Timeline/Schedule

3.4.1 Gantt Chart



3.4.2 Detailed Schedule Breakdown By Date

- February 15 - 28 – Market Research
- February 15 - March 15 –Obtaining and Initializing Server
- February 20 - March 20 – Design Starter HTML
- March 1 - March 31 – Setup Backend Framework
- March 10 - April 10 –Backend Classes (Student/Prof)
- March 15 - April 15 – User Logins
- March 20 - April 30 – Setup User Dashboard
- April 10 - May 5 – Customization on Course Page
- April 15 - May 15 – Course Quizzes
- April 20 - May 20 – Live Chat on Course Page
- March 10 - May 31 – Exception/Scaling Tests
- August 21-September 1 - Progression and refinement of core features
- September 1- October 15 - Forums page
- October 15 - November 1 - Teacher dashboard, Student Dashboard
- November 1 - November 15- Quizzes, Material Upload, Forums all at usable states
- November 15 - December 1-Personal testing, Bug fixing
- December 1- Present - UI Changes, Bug Fixes, In class testing

3.5 Risks and Risk Management/Mitigation

Potential risks associated with each major task have been identified. Each risk is assessed based on its probability (Low < 0.3, Medium ≈ 0.5, High > 0.7) and severity (Low, Medium, High). Mitigation plans for any risk with a probability greater than 0.5 or high severity.

1. Authentication & User Management
    - Risk: Security vulnerabilities in the login system (e.g., weak encryption, session hijacking) Probability: Medium (0.5)
      Severity: High
      Mitigation: Use industry-standard libraries (e.g., bcrypt for hashing), enforce secure login flows, and conduct security testing.
2. Question & Answer System
    - Risk: Media upload/storage issues (e.g., large file sizes, unsupported formats)
      Probability: High (0.7)
      Severity: Medium
      Mitigation: Limit upload sizes, validate file types, and use third-party cloud storage for scalability.
    - Risk: Poor UI/UX leads to underuse by students
      Probability: Medium (0.5)
      Severity: High
      Mitigation: Regular feedback from clients and students; perform usability tests and iterate on design early.
3. Polling and Quizzing Features
    - Risk: Real-time response delays or data inconsistency
      Probability: Medium (0.4)
      Severity: Medium
      Mitigation: Use efficient frameworks (e.g., socket.io or polling fallback), test under load, and simulate classroom-size users.
4. Archiving and Retrieval
    - Risk: Search or filter performance degrades with large datasets
      Probability: Medium (0.5)
      Severity: Medium
      Mitigation: Index database fields correctly; use pagination and caching to optimize frontend performance.
5. Statistics and Leaderboards
    - Risk: Inaccurate or unfair contribution tracking
      Probability: Medium (0.4)
      Severity: Medium
      Mitigation: Clearly define contribution rules (e.g., minimum content length), test metric logic, and allow instructor override if needed.
6. Testing and Deployment
    - Risk: Incomplete test coverage leads to bugs in production
      Probability: Medium (0.5)
      Severity: High
      Mitigation: Enforce testing as part of the development cycle (e.g., require tests for pull requests), and use automated test tools.
    - Risk: Deployment issues or downtime near the demo
      Probability: High (0.6)
      Severity: High

Mitigation: Deploy early on a staging server, conduct trial runs, and have a backup environment prepared.

7. Team-Level Risks
    - Risk: Uneven workload or member availability
      Probability: Medium (0.5)
      Severity: Medium
      Mitigation: Regular check-ins, shared documentation, rotating responsibilities, and pairing for critical tasks.
    - Risk: Delayed client feedback stalls progress
      Probability: Low (0.3)
      Severity: Medium
      Mitigation: Propose agenda in advance of meetings, and prioritize independent development items between feedback cycles.

3.6 Personnel Effort Requirements

Task	Subtask	Est. Hours
Authentication & User Management	User login/registration & role setup Profile management and access control Security implementation	10
Q&A System	Text-based Q&A functionality Media upload and playback Anonymous posting and categorization	15
Polling & Quizzes	Poll creation Student response interface and live updates Exporting participation data	10
Live Chat	Real-time messaging between users Typing indicators, message timestamps, and read receipts	15
Archiving & Retrieval	Data persistence and organization Search, filter, and view by category	10
Statistics & Leaderboard	User activity tracking Displaying leaderboards for students/TAs	10
Frontend UI/UX	Basic layout and responsive design Navigation and state management UI Polish and client feedback integration	25
Testing & Debugging	Unit and integration testing System testing and bug fixing	20
Deployment & Documentation	Hosting and deployment Final presentation and written documentation	30

3.7 Other Resource Requirements

In addition to time and effort from team members, the SmartClass project will require the following non-financial resources to support development:

1. Development tools & Platforms
    a. Code Editor/IDE: Visual Studio Code, IntelliJ
    b. Version Control: Git
2. Hosting & Backend Services
    a. Database: MySQL
3. Communication & Collaboration Tools
    a. Google Docs: For documentation and planning
    b. Email & meetings: Weekly client meetings for reports and feedback

4. Design

4.1 Design Context

4.1.1 Broader Context

SmartClass is designed to address student disengagement in large classrooms by providing an interactive, real-time learning platform. This problem exists at the intersection of education and technology, particularly in higher education institutions with students, instructors, and teaching assistants. However, educational institutions as a whole benefit through improved classroom dynamics, increased student performance, and better instructor insights into learning comprehension.

Area	Description	Examples
Public health, safety, and welfare	All users of the app experience piece of mind that questions can be asked	By offering students a way to ask questions in a large lecture hall, it helps alleviate stress of not knowing the answer to any questions they may have.
Global, cultural, and social	SmartClass reflects the aims of the groups that use it well as it will be used by students and professors to ask and answer questions.	If a professor's goal is to answer all questions that students have, SmartClass makes it easy to ensure that's possible
Environmental	SmartClass will have zero environmental impact, as it is a web application	N/A
Economic	SmartClass offers a way to communicate in real time with access to a question forum in a single application. That cannot be found cheap elsewhere.	Other apps do not have functionality for adding activities to forums and ability to download materials..

4.1.2 Prior Work/Solutions

Many education technology solutions attempt to improve classroom engagement, including TopHat, Piazza, Canvas, and Blackboard. However, these platforms often target narrow slices of classroom interaction. For example, Piazza focuses primarily on asynchronous Q&A, while TopHat emphasizes polling but lacks comprehensive class discussion features.

Pros and Cons Comparison:

Platform	Pros	Cons
TopHat	Engaging features, mobile-friendly	Expensive, limited adoption across all departments
Piazza	Anonymity support, widely adopted in STEM fields	Not designed for real-time lecture participation
Blackboard	Full LMS (Learning Management System), analytics	Poor UX, limited mobile support, expensive
SmartClass	Real-time chat, quizzes, polls, anonymity, stats	Still in development

4.1.3 Technical Complexity

The SmartClass project meets and exceeds expectations for technical complexity through the following:
Subsystems and Scientific Principles:

1. Frontend (React.js + WebSockets):
    a. Uses asynchronous event handling, state management, and responsive design principles.
    b. Real-time updates via WebSocket protocol require understanding, and responsive design principles.
    c. Uses session cookies for user session timeouts and persistence.
2. Backend (SpringBoot + Maven + MySQL):
    a. Relies on database normalization, secure authentication (bcrypt), and structured query logic.
    b. Engineering principles: encryption, input validation, and API design.
3. Live Features (Quizzes, Polls, Anonymous Chat):
    a. Combines database synchronization, UI/UX design, and real-time updates.
    b. Must be scalable and performant to handle classroom-size loads with low latency.
4. Statistics & Analytics:
    a. Tracks participation data and dynamically generates metrics like leaderboards.
    b. Relies on basic statistics, logging systems, and secure data aggregation.

Challenging Requirements:

- Must support hundreds of concurrent users per class.
- Needs to maintain user privacy while allowing instructors to grade based on participation.
- UI must remain intuitive across devices, accounting for accessibility.

These subsystems collectively utilize network protocols, software engineering standards (e.g. ISO 12207, ISO 27000), and design patterns to meet educational, functional, and ethical requirements.

4.2 Design Exploration

4.2.1 Design Decisions

1. Page Layout
    a. Since this project is a website, the user will only be interacting with the website. It is critical that the user can navigate through features and access the information they are looking for quickly. The layout of the page includes what information is shown on which page, what elements need to be accessible from the navbar, what needs to be displayed on each page, and many other visual elements. Without an easy-to-use interface, the software will not stand a chance of being used by anyone.
2. Design Flow
    a. The users need to be able to accomplish tasks quickly and easily. Avoiding overly complicated processes to create an account should be avoided. Creating a task decomposition allows for minimizing unnecessary steps that get between the user and their goal.
3. Data to store for users
    a. The amount of data needed for each user should be kept to a minimum. At least a username and password should be stored to allow users to sign in and use their accounts. Beyond that, any extra information asked for is another barrier to obtaining new users. The goal should be to reduce unnecessary data to give users ease of mind as to what personal data they are giving out.
    b. Data stored for students:
        i. Name
        ii. Email address
        iii. School name
    c. Data stored for professors:
        i. Name
        ii. Email address
        iii. School name
        iv. Area of study
4. Connection methods
    a. This project requires passing some amount of data between the frontend and the backend. The methods chosen determine what protocols are available. For personal information, higher levels of security should be used when passing user data back and forth between the server and the website.
    b. This project uses http, https, and websockets for connections. These protocols allow for information to be passed between front and back ends securely and in real-time.
5. Frameworks to use
    a. The frameworks used dictate the methods and functions available for programming the website. Certain frameworks provide libraries that make certain functions quicker and easier to create.
    b. React is the framework used for the frontend. This provides the ability to use JavaScript to give the website interactivity.
    c. Springboot is the framework used for the backend. This makes creating and connecting to a database easy.

6. Feature Selection
    a. Key features from existing LMS platforms have been strategically chosen and integrated while addressing their issues and shortcomings. The following elements have been selected to be incorporated into SmartClass:
    b. Anonymous questioning capabilities from Piazza
    c. Interactive classroom activities from Top Hat
    d. Student analytic features from Blackboard, but focusing on a better user interface.
    e. Overall, the centralization and organization are implementing a better user experience.

4.2.2 Ideation

Page layout was the design decision with the most variation and choice for how to implement. Other decisions were based on the skill and previous experience of the team developing the project, while deciding what to put on a page was much broader. When designing the page layout, many factors must be considered, such as readability of text, useful links in the toolbar, the order of information that makes sense to the user, and many more visual aspects.

Market research revealed several options for page layouts. Seeing what competitors decided was important for their layout provided a base for the project. This served as a starting point for discussions about what should be included. Discussions about why these things are included and how they could be improved defined many of the design decisions made.

Adding a navigation bar at the top of the page was decided upon for quick access to all the significant features of the website. There are links for jumping to the Home, Live, Courses, Chat, and Profile pages in the navigation bar. These will be the most frequently used pages, and providing a way to always jump to those pages was deemed necessary.

Another decision had to be made on the general content layout of the pages. This had many options for different circumstances. The webpage could have a two column design where one is used for text and the other for pictures or videos. There could be “blocks” of information where formatting the text to highlight headers and section titles similar to chapters breaking up a book. Multiple choices for colors can create different moods in users when using the website. Bright colors are more cheerful and playful, while darker black and white pages make the text the center point of the screen.

The final decision was made to use layers in designing the web pages. This approach is common in many websites, where content is organized into rows, and each row functions as a section centered on a specific topic. There can be any number of columns within each section and varying this was a good way to make the webpage feel more lively. Some sections have a picture on the left and text on the right, others are swapped. No matter what the configuration of the layer is, all information within it is related.

4.2.3 Decision-Making and Trade-Off

The decision-making process was kept quite simple to allow for quick and agile development. Anytime a question arose for a design aspect of the project, the team would vote for what they deemed the best option. A list of pros and cons was discussed, allowing all perspectives to be heard. Team members were allowed to give their opinions and suggest alternatives. Having a team that is open to criticism allows this strategy to be effective. We also utilized Gitlab issues to break down existing issues and features that still needed to be implemented into workable packages, suitable for individual development and seamless integration.

4.3 Design Problems

4.3.1 Overview

SmartClass is a web-based platform designed to help students and instructors interact more easily in large lecture halls. In larger classes, it is often more challenging for students to ask questions or for instructors to know who is having trouble understanding the lecture. SmartClass fixes that by giving everyone simple tools to participate in real-time.

Students can:

- Answer live quizzes or polls during class
- Ask questions anytime (even anonymously) using text, images, audio, or video
- Reply to each other and instructors in an open discussion space

Instructors and TAs can:

- Post quizzes or discussion prompts to check understanding
- Respond to student questions
- See participation statistics to help with grading and course planning
- View logs page detailing polling results and quiz results.

All conversations are saved and organized by topic (like homework or lectures), so students can go back and review them anytime. In the future, SmartClass can also be turned into a mobile app, so it is easy to use on the go.

The goal is to make learning more interactive and inclusive, giving every student a voice, no matter where they are sitting.

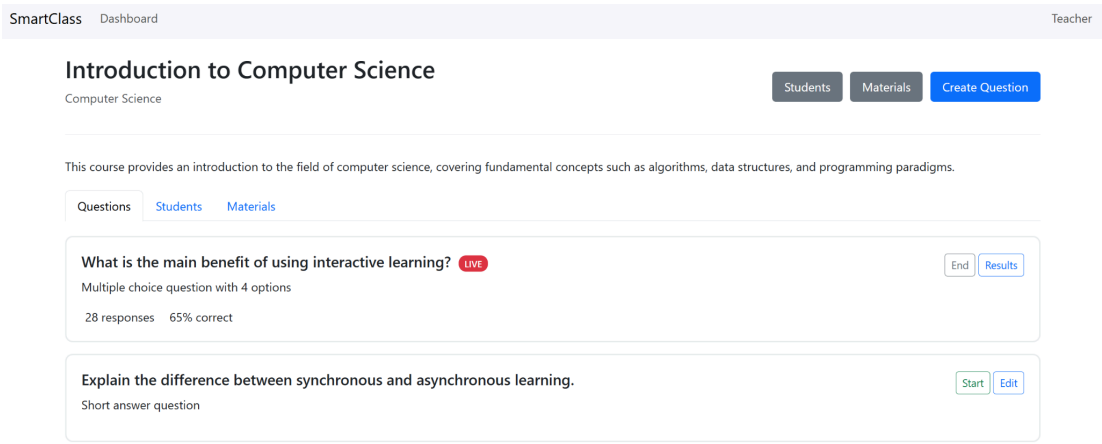
4.3.2 Detailed Designs and Visuals

SmartClass is a full-stack, role-based interactive classroom web platform designed to facilitate student engagement, especially in large lecture environments. It enables real-time Q&A, live quizzes, anonymous participation, media-rich discussions, and statistical tracking for performance and engagement.

The system consists of the following major components:

- Frontend (React.js + Websockets)
    - Role-based navigation for Student, Teacher, and TA accounts
    - Pages: Home, Class Dashboard, Forum, Activities, Archive
    - Dynamic updates for polls/questions using WebSockets
    - Potential Mobile-friendly accessible design in the future
- Backend (Spring Boot + Maven)
    - Serves as middleware for database access and logic control
    - Manages authentication, quiz logic, question handling, and user roles
    - Handles media file uploads and secure data storage
- Database (MySQL)
    - Stores user credentials, class rosters, Q&A threads, quiz results, participation logs, and uploaded files
    - Indexed for filtering/search performance
- WebSocket Server

- Enables real-time communication for:
  - Live poll updates
  - Forum interaction
  - Class-wide announcements



4.3.3 Functionality

SmartClass is designed to enhance the classroom experience for the students and the teachers. Here is how the system would function during a typical school day:

Before Class:

1. The teacher logs into SmartClass and creates their course (or if created access it again)
2. They prepare their lecture materials, including quizzes, slideshows, and other interactive elements.
3. Students then access the course page to review previous materials and prepare for class

During Class:

1. The teacher starts the lecture session and activates attendance tracking
2. Students log in and a displayed code is given to them in class and they input the code to be marked present.
3. As the lecture progresses the teacher can:
   a. Launch quick polls to gauge understanding
   b. Post quiz questions to get immediate feedback and understanding
4. Students can:
   a. View Lecture materials on their devices
   b. Respond to the polls and quizzes in class
   c. Submit questions to group/class discussion or to the teacher themselves.
   d. Take notes and comment on lectures directly on the application

After Class:

1. All lecture materials, this includes annotations and interactions are automatically saved.
2. Students can review materials and continue discussions.
3. Teachers can analyze the logs page for the lecture to see quiz results, polling results, and a log of the live chat.

4. More resources can be posted based on identified knowledge gaps.

4.3.4 Areas of Concern and Development

The current designs envisioned for SmartClass are strongly aligned with both user needs and system requirements. The proposed features directly target the core problems laced in large classroom settings, particularly the difficulty of inclusive participation, real-time feedback, and accessible learning tools.

Key areas where the design meets user needs:

- Inclusive Interaction: The anonymous Q&A, Forum, and live chat tools address the needs of shy or hesitant students who want to engage without speaking up
- Real-Time Feedback: Live polls and quizzes allow instructors to quickly assess class comprehension, supporting their need to adapt teaching on the fly.
- Centralized Resources: The design includes archived questions, lecture materials, and discussion threads, helping students revisit material even if they miss class.
- Role-Based Structure: Custom views and permissions for teachers, students, and TAs help streamline workflows and responsibilities.
- Future Scalability and Accessibility: The team is considering mobile-friendly design and accessibility features such as TTS and minimal hardware requirements, making the platform usable by a wide range of students.

Primary Concerns Moving Forward:

1. Scope Creep vs Core Focus
    a. The team has many ideas but needs to prioritize what's the most essential.
    b. Concern: Trying to implement too much at once may slow progress or dilute core values.
2. User Adoption and Usability
    a. Since no prototype has been tested yet, it is unclear how intuitive the UI will be.
    b. Concern: Will students and instructors understand and embrace the platform during live lectures without training or confusion?
3. Real-Time System Complexity
    a. Real-time features like WebSocket-based quizzes and chats are conceptually solid, but complex to implement.
    b. Concern: Achieving low latency and stable performance in a large class environment will require careful design and testing.

Immediate Plans to Address These Concerns

- Develop Wireframes and Mock UIs: Begin with static designs for all major views (login, forum, quiz view, etc.) to guide frontend development and gather feedback before coding.
- Focus on Important Core: Finalize which 2–3 features are essential for initial testing (e.g., Q&A forum, anonymous questions, and quizzes), and limit scope to just those for now.
- Build a Clickable Frontend Prototype: Use tools like Figma or early-stage React components to simulate workflows and run usability tests with students and instructors.
- Backend Planning: Continue building REST API endpoints and defining the database schema to support core data types like users, classes, questions, and responses.
- Early Testing Strategy: Plan small usability and feedback tests even on unfinished components, to refine UX early. Use past examples like ColorWorks' diverse user testing as inspiration .

Open Questions for Clients, TAs, and Faculty Advisors

- For Instructors (Clients):
  - Which 2–3 features would you most like to test live in class?
  - Would you prefer to moderate questions live, after class, or not at all?
  - Are any institutional integrations (Canvas, grade export) important now or only later?
- For TAs:
  - What tools would help you manage student questions more easily?
  - Would you use the system if it replaced current email or office hour Q&A?
- For Faculty Advisors:
  - Do you foresee any concerns with our role-based permissions or anonymous posting?
  - Should accessibility testing (e.g., screen reader support) be prioritized earlier?

4.4 Technology Considerations

The SmartClass system design leverages a modern full-stack web development approach. Each chosen technology reflects a trade-off between scalability, ease of use, performance, and team familiarity. Below is an overview of the technologies being used, why they were selected, and what alternatives were considered.

Frontend (React.js)

- Strengths:
  - Highly modular and component-based, making it easy to build reusable UI elements.
  - Strong environment with support for state management (e.g., React Context, Redux), form validation, routing, and testing libraries.
  - Community support and rapid development cycle.
- Weaknesses:
  - Steeper learning curve for new developers compared to traditional HTML/CSS.
  - Requires setup with bundlers (Webpack, Vite) and tooling to be production-ready.
  - Can be overkill for very simple views.
- Trade-offs:
  - Chosen for flexibility and scalability, even though setup and initial development require more overhead than static HTML/CSS.
- Alternatives Considered:
  - Static HTML/CSS with JS: Simpler but not scalable for real-time, dynamic interactions needed in SmartClass.

Backend (String Boot)

- Strengths:
  - Robust and well-documented framework for RESTful APIs.
  - Strong support for secure authentication, form validation, and database access via JPA.
  - Scalable for enterprise-grade applications.
- Weaknesses:
  - Verbose syntax compared to more modern frameworks like Node.js/Express or Python's FastAPI.
  - Slower startup time during development.
  - More complex deployment pipeline for beginners.
- Trade-offs:
  - Chosen for its maturity and built-in features for security, scalability, and structure, despite being more heavyweight.

- Alternatives Considered:
    - Node.js with Express: Lighter and faster to prototype, but less structured and potentially messier at scale.

Database (MySQL)

- Strengths:
    - A well-supported relational database that integrates smoothly with Spring Boot via JPA and Hibernate.
    - Ideal for structured data like user roles, quizzes, and posts.
- Weaknesses:
    - Schema changes require migrations, which can slow iteration during early prototyping.
    - Doesn't handle unstructured data (e.g. media metadata) as flexibly as NoSQL solutions.
- Trade-offs:
    - Chosen for structure and ACID compliance, which is important for handling user data, class records, and participation logs.
- Alternatives Considered:
    - MongoDB (NoSQL): More flexible, easier to work with media or JSON-like documents, but lacks relational integrity.
    - PostgreSQL: Considered, but MySQL was chosen due to easier hosting and broader familiarity among developers.

WebSockets

- Strengths:
    - Enables real-time push-based communication (ideal for polls, chats, and live Q&A).
    - Better performance than frequent REST polling.
- Weaknesses:
    - Harder to debug and test than traditional HTTP endpoints.
    - Introduces complexity in connection handling and user session tracking.
- Trade-offs:
    - Chosen for interactivity; however, fallback mechanisms like long-polling may be implemented if needed for poor network conditions.
- Alternatives Considered:
    - Polling via REST: Simpler but less efficient.
    - SignalR (Microsoft): Considered but not compatible with Java backend.

Testing Tools (Jest, React Testing Library, Mockito, Postman, CI/CD)

- Jest (Frontend Unit Testing)
    - Strengths: Fast and popular JavaScript testing framework; integrates well with React Testing Library
    - Weaknesses: Limited to frontend/unit scope (doesn't cover integration or user flows)
    - Use Case: Testing UI logic, component rendering, props behavior, and simulated user actions
- React Testing Library Strengths:
    - Encourages testing from the user's perspective
    - Weaknesses: More verbose than traditional shallow testing
    - Use Case: Testing component accessibility and realistic interactions

- Mockito (Backend Unit Testing)
    - Strengths: Allows mocking of service layers and testing business logic in isolation

    - Weaknesses: Requires setup of mock data and test scaffolding
- Postman
    - Strengths: Easy to manually test REST endpoints, debug headers, and payloads
    - Weaknesses: Manual testing unless combined with Newman (for automation)
- CI/CD Pipeline
    - Tools Considered: GitHub Actions or GitLab CI
    - Purpose: Automate linting, unit testing, and deployment processes
    - Benefit: Prevent regressions and ensure production-readiness by catching errors earlier

4.5 Design Analysis

As of now, the SmartClass team has made solid progress on both the frontend and backend, laying the foundation for key system functionality. While the full product isn't yet operational, several important components are partially implemented and aligned with the original design goals outlined in section 4.3.

What Has Been Built So Far:

- A website built with React which has a home, about us, and sign-up for account pages.
- A database for storing user login and account information

5. Testing

The tests are split into two major categories: frontend and backend. This allows each team to focus on their area of expertise and integrate new features quicker. The frontend tests focus more on user interactivity while the backend tests work on checking connections and communications between clients and the server. Tests that require both the frontend and backend are conducted by both teams to ensure problems are effectively communicated and the solutions are understood by both parties..

Tests are continuously run throughout the development process to guide the project in the right direction. Anytime functionality is lost, the tests notify where the problem is occurring allowing for quick fixes. Testing is done on each individual branch before merging into main to mitigate issues going to the live website. Should a bug be merged into main, the tests are run again giving further defense against bugs.

Writing tests using available APIs and public libraries is sufficient for this project. The tests are tailored to this application and focus on the continued functionality of the website. Each time the project is updated, automated tests are run using CI/CD controls to ensure no functionality gets broken by the update and that the website cont. These tests are performed by Docker. Other APIs used for testing include JUnit, Mockito, and SpringBootTests.

Frontend

- Unit Tests (Simulate user workflows)
- Usability tests (Testing buttons)
- Security Testing
- UX Testing

Backend

- Use Mockito for testing REST APIs in the backend to create accounts and retrieve user information

- Integration Tests w/ frontend (Ensure the backend is functioning correctly with frontend)

5.1 Unit Testing

Unit testing will be done on an individual method basis. Each method must be tested to check for functionality and proper implementation. Unit testing consists of providing an input to a method and comparing the output with an expected output which is calculated beforehand.

JUnit is used for conducting unit tests. JUnit is a library that provides many of the features wanted while testing. One useful set of methods includes assertions that allow for many types of comparisons, including equality, less than, greater than, and inequality. JUnit also allows automatic testing by allowing inputs to be read from an input file to test many cases at the same time without requiring knowledge of coding.

Key Interfaces:

- JUnit integration with Eclipse IDE
    - Allow for a single click to perform all tests
    - Shows detailed analysis of passing and failed tests

Testing Approach:

- Write multiple tests for each method
    - Test for edge cases
    - Test a variety of common values with randomly generated input
    - Test boundaries
- Use input files for test cases
    - Keep a structured file for writing inputs and expected outputs

Testing Tools:

- Decorations:
    - @Test : Indicates test methods
    - @BeforeEach : Performs this operation before every test
    - @AfterEach : Performs this operation after every test
    - @BeforeAll : Performs this operation once before any tests
    - @AfterAll : Performs this operation once after all tests are complete
- Methods:
    - AssertEquals(expected, actual) : Compares for equality between expected and actual
    - AssertNotEqual(expected, actual) : Compares for inequality between expected and actual
    - AssertTrue(condition) : Checks condition for true

Testing Process:

1) Write tests for methods as they are created
2) Run tests
3) Evaluate test results and fix errors
4) Save tests to rerun in the future for regression testing

5.2 Interface Testing

Key Interfaces:

- Frontend to Backend
    - REST API endpoints for user authentication (ie. login and registration for different roles)
    - WebSocket connections (polls, live chat, and question posting)
    - File upload of lecture materials
- Backend to Database
    - Data persistence for accounts, attendance and course data
    - Storage of quiz responses and participation statistics
- Role-Based Access
    - Student, Teacher, TA permissions
- Frontend UI Components
    - Navigation between course pages, lecture content, quizzes, etc.
    - Quiz display and submission
    - Live lecture information
        - Chat
        - File uploads
        - Attendence
        - Anonymous mode

Testing Approach:

- API Interface testing
    - Test API endpoints for correct functionality
    - Test frontend UI for connection to database
    - Test anonymity features
- Websocket Testing
    - Test real-time chat function
    - Test storage of chat logs
- Files & downloads
    - Test lecture material uploads
    - Verification of file types
    - Test download of files and materials

Testing Tools:

- API testing
    - Postman for endpoints
    - Automated API tests in CI/CD pipeline
- Fronted Interface
    - React testing library and Jest for testing the components
- Backend Interface Testing
    - Spring Boot
    - JUnit for testing backend service interfaces

Testing Process:

1. Test every interface
2. Integrated tests

3. Test role-based permissions
    a. Teacher permissions
    b. Student permissions
4. Error handling
5. Real-time features maintain consistent and synced data
    a. Forums
    b. Attendence
    c. Live chat

5.3 Integration Testing

Critical Integration Paths:

- User Authentication and Authorization
    - Student, TA, and Teacher registration
    - Role-based access
    - Class creation and joining a class

  Justification: This path is important because it controls access to the overall use and functionality of SmartClass. Different permissions are given to different levels of users. Users who have access to sensitive data such as quiz results must be authorized.

- Live Interaction
    - Real-time chat function
    - In-class quiz delivery and response
    - Submissions and answering
    - Participation statistics

  Justification: This path is important because the main feature and use for SmartClass is to enhance student participation. Tracking key data points is required to validate the success of SmartClass's design.

- Content
    - Lecture uploads and downloads
    - Course organization
    - Previous lecture and quiz access
    - Attendance tracking
    - Performance statistics

Justification: Storage and retrieval of materials is important for the platform both inside and outside of the classroom. Student participation is also tracked to be used in statistics for the teacher. Quiz questions and answers are stored in the database for teachers to upload class quizzes and view the student responses.

Testing Tools:

- Automated Tests
    - Supertest for API

    - CI/CD
- API testing
    - Postman for API workthrough
- Real-Time Testing
    - Websocket libraries to test functionality of chat and quiz

Testing Process:

1. Integration tests run automatically in the CI/CD pipeline for changes in code
2. Tests will simulate interaction of different user types
3. Each integration path will be tested independently
4. Failed tests will block a merge to the main branch

Testing Approach:

- Full Test Scenario:
    - Full simulation of the web application being used, with multiple user types
    - Creating classes, joining, posting, and responding
- Components
    - Testing interactions between different components
    - Verify that data and responses are tracked correctly with the frontend and backend
    - Test the websockets
- Data Integrity Testing
    - Verify that the user inputs are correctly stored
    - Test that the statistics and analytics work
    - Quiz results

5.4 System Testing

System testing ensures the independent components are working together to provide all the services promised by the website.

Testing Strategy:

The system will combine multiple testing methods to make sure that SmartClass meets the functional and non-functional requirements outlined in section 2.1.

Key Test Scenarios:

- Full Classroom Test:
    - Scenario: Log in multiple users of different roles, this being a teacher, students, TA
    - Actions: Teacher starts a lecture session, uploads materials, creates polls and quizzes
    - Student Actions: Attendance, answering polls, submitting anonymous questions
    - Tools: Selenium for UI testing
- Course Management
    - Scenario: Teacher creates a course, generates join codes, and students join
    - Actions: Manage attendance
    - Verification: Role-based permissions are enforced, and class materials are available
    - Tools: Automated Scripts

End-to-end Process:

- Live lecture activities

- The teacher starts a lecture and takes attendance
- Real-time quiz
- Student question submission
- Teacher/TA response
- Verification all real-time features function

Performance:

1. Load Testing
    a. Testing response times
2. UI Load Testing
    a. Determine the number of users that can be supported
    b. Website remains fast and responsive

Tools:

- Selenium
- Test scripts
- CI/CD pipeline

5.5 Regression Testing

To ensure that new feature implementations do not disrupt or cause existing functionality to fail, a combination of testing strategies and tools is employed:

- Automated Testing
    - Unit tests for both the frontend and backend are being implemented. Background logic is tested using tools such as Mockito.
    - The tests are expected to grow over time whenever new features are introduced.
- Integration Testing
    - Integration tests between the frontend and backend are used to verify that APIs behave as expected with UI components.
- Usability Testing
    - Manual and observational testing are part of the ongoing development, where button layouts and flow are tested to make sure other features such as quizzes and Q&A continue to work

Critical Features to Maintain

- Authentication and Role-based Views
    - Login and dashboard logic for students, TAs, and teachers must not break as they control feature access.
- Anonymous Q&A and Discussion
    - These core engagement features directly align with the project's goal to support shy or remote students and must remain functional.
- Quizzes and Polling System
    - Any failure in these systems would disrupt classroom interactivity and grading processes.
- Document Uploading and Archiving
    - The teacher's ability to upload notes and students' access to past class content are important.

Is it Requirements-Driven?

Yes, the testing is driven heavily by:

- User Needs & Functional Requirements
    - Anonymous interaction, quiz handling, participation tracking, and accessibility support are outlined as needs and requirements.
- Engineering Standards
    - Compliance with ISO 9000 and 12207 makes sure of software quality and adherence to a good development lifecycle, including testing phases.

Tools Involved

- Mockito for backend REST API testing
- React Testing Library
- Manual usability testing

5.6 Acceptance Testing

To demonstrate that both functional and non-functional requirements are met, a structured acceptance testing process will be followed:

Functional Requirements Validation

- Scenario-based testing: Each core feature will be tested with use cases based on the requirements outlined in our documentation
- Use-case tests: An example can be a teacher creating a class, adding a quiz, students joining and responding, teacher viewing participation results. This demonstrates that multiple components are functioning together properly

Non-Functional Requirements Validation

- Usability: Conduct usability testing with diverse users to ensure the UI is intuitive and responsive
- Performance: Stimulate high user loads to validate the scalability of chats, quiz submissions, and database queries
- Accessibility: Ensure features like text-to-speech, keyboard navigation, and clean design work for users with different needs
- Security: Validate secure login, data encryption, and access control through test accounts and role-switching

Involving Clients in Acceptance Testing

- Client Walkthroughs: Demonstrate new features and prototypes to a client to effectively confirm each party agrees on where the project is headed. Involving the client in the workflow helps ensure the product created is the one that meets their needs. These demonstrations feature both a live demo portion in a controlled environment. At the end, the client is allowed to go hands on with the demo to give more feedback on what works and what doesn't.
- Feedback-Driven Iterations: Weekly client meetings allow for iterative feedback and verification of components as they are developed. Several new features have been suggested and added during these feedback sessions. Modifications to existing features have also been incorporated into the design.

5.7 Security Testing

Security testing is important for the SmartClass learning platform due to the nature of the data shared. To be more specific it holds user credentials, education data and resources, and an overall goal to maintain academic integrity. Due to SmartClass being an interactive learning platform that connects students, teachers, and TAs, SmartClass requires in-depth security testing to protect the privacy of users and prevent unauthorized access.

Security Testing Areas:

- Authentication mechanisms
- Protect personal information
- Session management
- Integrity within the anonymous posting features
- Protection against common web vulnerabilities:
    - XSS
    - SQL injection
    - CSRF
- Security within file upload

Testing Methods:

1. Burp Suite
    a. Web vulnerability scanning
    b. Intercepting proxy to analyze the requests and responses
    c. Active scanning for web vulnerabilities (referenced above)
    d. Session analysis for weaknesses
    e. Testing the anonymous posting features for integrity.
2. Manual Review
    a. Reviewing code for the authentication methods
    b. Review of password storage
    c. Check guidelines in comparison with ISO 27000 (commonly used for web applications)
    d. Testing privilege escalation
3. Data Protection Testing
    a. Verification of data encryption (HTTPS)

Testing Plan:

- Initial scan before “completion”
- After major refinements in code base there will be security scans
- Light “pen-testing” before release, hands on approach.

Documentation:

- Findings documented alongside severity ratings
- Vulnerabilities tracked and resolved
- Security testing results logged to be reviewed before release.

As mentioned in section 2.2 the ISO 27000 is centered around security and privacy protection and will be used to ensure that SmartClass provides a secure environment and interface for users. The goal is to protect user data using good cybersecurity practices.

5.8 Results

The current results of testing are light. Unit tests are unimplemented, so all testing has been done by hand up to this point. This tedious process consists of writing code, restarting the server to test, fix code, restart the server to test again, and repeat. The goal is to begin automating tests using the strategies mentioned in this section to alleviate some of the workload.

However, the results of the manual testing are good. Bugs have been fixed by trial and error. Updates to the webpage are seen with each refresh allowing for quicker testing on the frontend. The user experience isn't anything fancy as of yet on account of the focus being on adding features to the website. That being said, the navbar is functional and accessible at all times from all pages. All currently in-progress and finished pages are accessible from the navbar for usability.

No user testing has been done yet, so it's to be determined as to how others would interact with the website. More testing also needs to be done with color and accessibility for people of different capabilities. Decisions on font styles and sizes are reliant on this testing getting done.

The plan is to begin testing as soon as this week since the website is finally in a usable state. Enough features have been implemented to allow for user, system, and acceptance testing to begin. Soon, unit and regression tests will also be integrated using JUnit.

6. Implementation

As of this stage in development, there have been several key components of SmartClass that have been implemented:

Frontend Implementation:

- Login and signup forms built using React.
- Student and Instructor dashboards with role-based views.
- Anonymous question board and live-chat functionality.
- Basic UI features for class management for teachers.
- Basic UI features for course viewing for students.
- Ability to join courses.
- UI elements ready to handle viewing materials.

Backend Implementation:

- Implemented using Spring Boot, designed with a RESTful architecture.
- APIs have been implemented for:
    - User registration.
    - User login.
    - Posting and retrieving anonymous questions.
    - Creating and submitting polls.
    - Logging student interactions.
    - Mockito has been used to develop unit tests.
    - Integration testing is done through Postman to validate requests and responses.

Database Implementation:

- MySQL has been designed and implemented with tables for:
    - Users

- Roles and permissions
- Class sessions
- Messages

Real-Time Features Implementation:

- Real-time communication (chat, polling, live Q&A) is enabled through WebSockets.

7. Ethics and Professional Responsibility

7.1 Areas of Responsibility

In the context of SmartClass, engineering ethics refers to prioritizing the well-being, rights, and safety of all users– including students, instructors, institutions– through responsible software development and coding practices. Professional responsibility revolves around maintaining integrity in decision making and being accountable for outcomes. SmartClass is centered around the ethical philosophy of transparency and accountability. The design process incorporates both ensuring functionality, security and accessibility (technical responsibility) and social responsibility (ensuring fairness and transparency in classroom engagement.

Areas of Professional Responsibility/Codes of Ethics

This discussion is with respect to the paper by J. McCormack and colleagues titled "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", International Journal of Engineering Education Vol. 28, No. 2, pp. 416–424, 2012

Area of Responsibility	Definition	Relevant IEEE Code of Ethics Clause	How Is It Applied?
Work Competence	Performing and creating high-quality work with integrity.	#6:Maintain and Improve our technical competence	Followed Agile workflows, performed peer reviews, and sought out feedback to unsure a technically sound application.
Financial Responsibility	Create features without unnecessary costs and wasting money.	#9: Avoid Conflicts of interest	Designed the system to be free of charge for students.
Communication Honesty	Talk with teammates and clients in a respectful truthful manner with clear concise plans.	#3:Be honest and realistic in stating claims or estimates	Shared weekly progress reports with the client and maintained truthful progress and failures with transparency
Health,Safety, Well-Being	Avoid harming users, clients, anyone but especially promoting a safe environment.	#1: "Hold paramount the safety, health, and welfare of the public."	Implemented an anonymous Q&A feature that would create a safe place for students to share

			thoughts
Property Ownership	Respect intellectual property as well as the data and information for users and institutions	*#5*: “Avoid injuring others, their property, reputation, or employment.”	Used hashed passwords, store very little personal data.
Sustainability	Consider the impact that the technology could have on the environment	#10: “Support sustainable development.”	Selected efficient deployment options
Social Responsibility	Make sure users are treated with fairness in all aspects. This includes but is not limited to inclusivity	*#8*: “Treat fairly all persons regardless of race, religion, gender…”	Designed a UI that will give all teachers the same roles and the students equal access and the same roles.

Strength: Communication Honesty

SmartClass excelled and currently excels at keeping communication transparent within the group and with the client. Weekly meetings included honest discussion and Git issues stated clear progress.

Improvement Area: Work Competence

The basis of Agile workflow was great and it became muddy when the group was knocked off track with sometimes there being miscommunication.

7.2 Four Principles

	Beneficence	Nonmaleficence	Respect for autonomy	Justice
Public health, safety, and welfare	Design systems that maximise well‑being (e.g. safe water, clean air)	Identify and eliminate potential hazards (e.g. fail‑safe designs)	Provide clear information on risks/benefits (e.g. product labeling)	Ensure equal access to safe technologies (e.g. affordable vaccines)
Global, cultural, and social	Promote technology transfer that uplifts developing regions	Avoid imposing solutions that disrupt local ways of life	Engage local communities in decision‑making	Prevent “technology colonialism”—i.e. fair partnerships
Environmental	Develop sustainable,	Minimize pollution, resource	Involve affected communities in	Distribute environmental

	regenerative technologies (e.g. green energy)	depletion, habitat destruction	environmental impact assessments	benefits and burdens evenly
Economic	Create products/processes that boost livelihoods and GDP	Prevent economic harm (e.g. avoid obsoleting workers without retraining)	Provide transparent cost/benefit data so users can choose	Ensure fair wages, avoid exploitation of labor

Public health, safety, and welfare paired with beneficence is the pair that is most important to SmartClass. Offering students a way to ask questions easily ensures their mental wellbeing. Giving teachers a way to answer questions easily ensures they are teaching to the best of their ability, ensuring their mental wellbeing. SmartClass can also engage local communities in decision-making. The communities involved would be students and professors.

SmartClass is lacking in most environmental pairs, as it is doing nothing for the environment. This negative is overcome by the positives as there is no reason for a web application to strive for environmental change.

7.3 Virtues

Throughout the development of SmartClass, many core virtues have shaped how we collaborate, solve problems, and make ethical decisions.

1. Accountability
    a. Taking responsibility for assigned tasks and the outcome of those tasks regardless of success or failure.
    b. The team used weekly sprint check-ins and task assignments to maintain transparency and create accountability. Missed tasks were followed by action as opposed to blame.
2. Integrity
    a. Being honest in communication, quality of work, knowledge, skill level, and decisions. Doing the right thing and making the right decisions no matter what.
    b. Team members raised concerns when features didn't fully meet requirements and offered outside knowledge and support to keep the SmartClass mission and requirements moving.
3. Collaboration
    a. Working effectively in a team setting, sharing knowledge, and supporting the groups contributions
    b. In practice regular working sessions were held to debug frontend and backend issues and it encouraged peer feedback on design decisions.

4. Team Members Comments

- Logan Pfantz
    - Perseverance has played a major role in my journey, especially in times when I wasn't sure how to move forward. One example was learning how to implement WebSockets to build a live chat feature for our website—something I had no prior experience with.

Instead of giving up, I kept researching, testing, and improving until it finally worked. I believe perseverance teaches us what we're truly capable of. I've tried many different paths in life, and after pushing through challenges and uncertainty, I've found a sense of purpose through my Software Engineering degree. I wouldn't be here without the determination to keep going, even when things felt overwhelming.
    - One virtue I haven't demonstrated well during my senior design project is self-discipline. I've struggled with managing my time, often waiting until deadlines to complete tasks—and sometimes even letting them slip into the following week. I know that self-discipline is essential, not just for my own growth, but because it allows my team to count on me. This semester, I haven't always been someone they could rely on, and that's something I want to change. Moving forward, I plan to engage more actively with my team and set clear, achievable goals for myself each week. I've already overcome challenges I never thought I could, and building self-discipline is the next step I want to take in becoming a better teammate and engineer.
- Matthew Gudenkauf
    - One virtue that I've demonstrated the most thus far is collaboration. Collaboration has been my way to update everyone including myself on each other's work and what is in progress. Without collaboration we wouldn't have been able to reach deadlines on our tasks. Collaboration has helped me multiple times by moving to a new task that hasn't been assigned to anyone, increasing our development time and eliminating any wasted time there would have been if multiple people were doing the same task.
    - One virtue that I don't think I've demonstrated as much is integrity. In specific, there have been times of procrastination where I waited until the last few days of the week to finish the task I was supposed to finish. This resulted in rushing to finish the work, which resulted in problems a few times. To demonstrate integrity moving forward I plan to find a few hours during the beginning days of each sprint to plan and create ideas. This will help by being able to spend some time developing a plan to implement a new feature little by little during the sprint cycle..
- Josh Dwight
    - One virtue that I've demonstrated is collaboration. Throughout this semester, I have collaborated with my teammates to create this product. On top of that, I have discussed with our team the best practices for collaboration and have proposed that we use a better method for keeping track of work next semester.
    - One virtue that I feel I have lacked at times this semester is efficiency. In the beginning of the semester, I spent a very long time learning how to work on the backend, when I should've spent more time just going in and working on it and learning by doing instead of learning by reading about it online and watching videos on it. I will be better about being efficient in my work next semester.

- Michael Becker
    - I would say I thought I have demonstrated the virtue of collaboration. When I am unsure of doing something the first people I ask are my group members. I have at the end of the semester tried to be better at explaining what exactly I am doing in the discord chat and

during meetings. It is important to me because in a team setting collaboration and being able to work together cohesively plays a major role in a teams success.
    - One virtue I have not displayed is initiative. I think that I should have taken a more active role in helping and offering help as opposed to being asked for it. It is important to me because I feel as if I could have offered my help more. To demonstrate this virtue I will take more initiative in the project next semester by bringing more ideas to the table and becoming a "louder" voice during group discussions.
- Michael Geltz
    - One virtue I've demonstrated so far is collaboration. I have been consistent in putting what I need into words and also communicating what I have done or need my team members to do. Collaboration is essential for this project due to the deeply entwined nature of web development. Working on the backend has required me to develop things on an as needed basis a lot of times and being able to understand what my team members need is critical.
    - One virtue that I need to be better at applying next semester is accountability. I have not held myself accountable for deadlines a lot this semester, and this has led to sometimes not being able to deliver something when it is needed. To better demonstrate this virtue next semester I will more clearly ask for what needs to be done and how I can best accomplish it.
- Ryan Lin
    - One virtue that I've demonstrated is collaboration. If I am available and if someone has a question or needs help, I try to help them whether that is troubleshooting issues together or explaining concepts in a way that moves the project forward. I believe that shared understanding leads to stronger solutions, and I try to make an effort to contribute positively to team discussions and group tasks.
    - One virtue that I can improve at is my communication. I tend to heavily focus on completing my work independently and sometimes delay notifying the team until after it is done or if I know I am going to miss a meeting. I am working on being more proactive in updating the team on my progress, timelines, and availability so that we can stay aligned more effectively. Better communication will help the team plan around dependencies and reduce uncertainty.

8 Closing Material

8.1 Conclusion

The SmartClass team has made significant progress in the development of a user-friendly classroom engagement platform that allows for real-time interaction, anonymous Q&A, and attendance tracking.

Original goals:

- Enabling anonymous student participation
- Supporting real-time chat and polling
- Role-based dashboard for instructors and students
- Logging participation and quiz results for instructors to view

To achieve these goals, a modern full-stack architecture was developed and implemented. React for the frontend, Spring Boot for the backend, and MySQL for the database. The solution also implemented real-time chat features built on WebSockets.

Constraints:

- Learning curves related to React and Spring Boot security
- Web Application security
- Scheduling limitations reduced opportunities for more in person meetings

Future considerations:

The future of SmartClass is bright and involves refining the current model and setting up usability testing to ensure smoother transitions when SmartClass fully launches.

8.2 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE). See link:
https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf

[1] IEEE, "IEEE Code of Ethics," [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: May 4, 2025].

[2] ReactJS, "React – A JavaScript library for building user interfaces," Meta, [Online]. Available: https://reactjs.org/. [Accessed: May 4, 2025].

[3] Spring Boot, "Spring Boot Documentation," The Spring Team, [Online]. Available: https://spring.io/projects/spring-boot. [Accessed: May 4, 2025].

[4] MySQL, "MySQL 8.0 Reference Manual," Oracle, [Online]. Available: https://dev.mysql.com/doc/. [Accessed: May 4, 2025].

[5] J. McCormack, et al., “Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment,” *Int. J. Eng. Educ.*, vol. 28, no. 2, pp. 416–424, 2012.

[6] Piazza, “Anonymous Q&A Platform for Higher Education,” [Online]. Available: https://piazza.com. [Accessed: May 4, 2025].

[7] Top Hat, “Interactive Teaching Platform,” [Online]. Available: https://tophat.com. [Accessed: May 4, 2025].

[8] Postman, “API Platform for Building and Testing,” [Online]. Available: https://www.postman.com/. [Accessed: May 4, 2025].

8.3 Appendices

N/A

9 Team

9.1 Team Members

1) Logan Pfantz
2) Matthew Gudenkauf
3) Josh Dwight
4) Michael Becker
5) Michael Geltz
6) Ryan Lin

9.2 Required Skill Sets for Your Project

These are broad categories of essential skill sets required to build SmartClass.

Programming	Development of the core project functionality.
Experience in Git	Code repository management software
Testing	Ensuring software quality and reliability
Time Management	Ensuring completion of tasks

9.3 Skill Sets covered by the Team

The team collectively has all of the required skill sets for the project. While some members have a better understanding in certain areas than others

9.4 Project Management Style Adopted by the team

Typically, Waterfall or Agile for project management.

9.5 Initial Project Management Roles

Organization - Michael Becker

Client Interaction - Logan Pfantz

Individual Component Design - Michael Geltz, Josh Dwight

Testing - Matt Gudenkauf, Ryan Lin

9.6 Team Contract

Team Procedures

Day, time, and location (face-to-face or virtual) for regular team meetings:

Mondays : 12-1pm through discord

Additionally communicating work that is being pushed/worked on

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):

- Discord - Used for our main source of communication
- GitLab - Using issue boards to give a clear picture of what everyone is doing
- Email - Communicating with our client

3. Decision-making policy (e.g., consensus, majority vote):

- Coming to a consensus on all decisions.

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

- A shared google document has been created to track our minutes following our clients template

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

- Everyone should attend group meetings. Let the group know if you can't make it
- Participate, share new ideas, and plan ahead for the week

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

- Everyone is responsible for putting effort into completing the project by the deadline
- There may be cases where a team member can't work on the project on certain days. That's okay, but it shouldn't be a weekly occurrence
- Each team member is expected to work about the same amount on the project by the time it's due
    - We should all put in equal effort and time into the project

3. Expected level of communication with other team members:

- Updating the team using Discord on any progress
- Asking the group questions if anyone is struggling with any implementations
- Commenting code clearly

4. Expected level of commitment to team decisions and tasks:

- Everyone is expected to participate in weekly meetings to discuss new and current ideas to help other team members come to a consensus on the best way to develop the project
- Tasks assigned to a team member will come with an expected completion date, which we will hold the person responsible for meeting.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

- Organization - Michael Becker

- Client Interaction - Logan Pfantz
- Individual Component Design - Michael Geltz, Josh Dwight
- Testing - Matt Gudenkauf, Ryan Lin

2. Strategies for supporting and guiding the work of all team members:

- Assigning tasks that fit people's skillset
- Setting small, manageable deadlines for tasks to be completed
- Communicating through Discord if any problems arise while working on the project

3. Strategies for recognizing the contributions of all team members:

- Highlighting their contributions in meetings
- Keeping records of who worked on what and when they completed it
- A credits section in the presentation at the end of the semester

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the

Team.

- Matthew G - Taken classes with Java, JS, HTML, CSS, Databases, and more.
- Logan P - I've worked with C, C++, Java, SQL, some HTML, and JS using React. I have front-end development experience from programming and designing an Android app from COMS309.
- Michael G - Java, C, Python, SQL, JS, HTML. Experienced in backend development.
- Michael B - Java, Python, C, C++, HTML, JS. Like Ryan mostly worked with the frontend.
- Ryan L - Taken classes with Java, Python, JS, HTML, CSS, C, C++, etc. Currently in an internship learning ASP.NET and TS. Experienced mostly with frontend work.
- Josh D - I work at John Deere and mostly do stuff with Python and AWS there.

Other than that, I have experience with C, Java, Go, React, JS, Linux, and some

cyber tools.

2. Strategies for encouraging and support contributions and ideas from all team members:

- Listening to all ideas and opinions. Critiquing any ideas in a friendly manner, explaining how it could be good/bad.
- Communicating and coming to a consensus on which would best fit the project.

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will

a team member inform the team that the team environment is obstructing their

opportunity or ability to contribute?)

- Our main goal is to resolve any issues one-on-one with the other person.
- Using Discord to communicate to the team if anything is getting in the way of contributing to the project.
    - Resolving the issue in a friendly manner ASAP.
- We will allow all group members to voice their opinions in team meetings. Even if we disagree, we don't want to shut out differing views in case they may result in a better project.
- If we see someone who wants to say something, we'll ask them what they think

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

- Learn to work in a team by establishing timelines, communicating issues, and cooperating to get work done.
- Create a fun and valuable tool that can be used in a real-world setting.
- Create connections with new students for personal growth and professional development.
- Create a solid base and put ourselves in a great position to finish the project by the end of next semester.

2. Strategies for planning and assigning individual and team work:

- We plan to use issues on GitLab for features we want to implement and are working on. We can keep track of individual efforts via commits.

3. Strategies for keeping on task:

- During our weekly meetings, we will review anything we want done within the next week. If anybody fails to meet all their goals, we will turn to the consequences section of this contract for proper punishment.
- We will also have milestones for certain points in development where we can have specific features implemented and working.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

- We will reach out to the offending individual and attempt to resolve any issues preventing them from meeting the requirements of this project.

2. What will your team do if the infractions continue?

- We will reach out to the instructors if issues happen continuously. We plan to record all individual efforts and missing assignments within the group to keep track of infractions. This should hold everyone accountable.

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) Matthew Gudenkauf_____ DATE 02/11/2025

2) Michael Geltz_____ DATE 02/11/2025

3) Logan Pfantz_____ DATE 02/11/2025

4) Michael Becker_____ DATE 02/11/2025

5) Ryan Lin_____ DATE 02/11/2025

6) Josh Dwight_____ DATE 02/11/2025