Dhirubhai Ambani University

Classync - Virtual Classroom Platform

Sprint 4 Report

Group - 33

Course: Software Engineering
Professor: Prof. Saurabh Tiwari
Mentor: Shyam Patel

Team Members:

Role	Name (ID)
Team Leader	ATIK VOHRA - 202301447
Member	KANANI TANUJ - 202301474
Member	SANAGADHIYA RADHIKA - 202301184
Member	PATEL JAY - 202301430
Member	CHOVATIYA AYUSH - 202301461
Member	OM PATEL - 202301163
Member	BADRESIYA ANSH - 202301477
Member	PREM KUKADIYA - 202301452
Member	RAMOLIYA SHUBHAM - 202301442
Member	GAJERA MANTHAN - 202301488

Contents

1. Overview / Summary

1.1 Notification System

Deliver a unified notification system for classroom events such as assignments, submissions, enrollments, groups, stream posts, email verification, and general system messages. The goal of this sprint is to provide a consistent data model for notifications, user preference control, near real-time delivery (using initial polling with an optional Server-Sent Events (SSE) extension), and to achieve at least 80% test coverage for all new modules introduced in this feature set.

1.2 AI-Powered Learning Assistant

Deliver an AI-powered learning assistant that combines topic-based quiz generation with an AI tutor chatbot for academic question-answering. The aim of this sprint is to provide a consistent data model for quizzes and chat sessions, robust history management, predictable AI outputs, fast response times, and at least 80% test coverage for all new modules. The user experience focuses on clean flows for generating quizzes, practicing them, asking questions, and managing histories across both tools.

Duration

15th November, 2025 to 28th November, 2025

2. User Stories Covered - Notification System

2.1 Core Notifications

US-N01: Unread Notification Count Badge

Front Card

As a user, I want to see my unread notification count so I can gauge pending actions at a glance.

Back Card

Success

- Badge shows an accurate unread notification count that matches the backend state.
- Count decrements immediately when items are marked as read from any view (dropdown or full notification page).

Failure

- Count is desynchronized (shows incorrect value, becomes negative, or does not change on updates).
- Mark-read actions do not reduce the unread count on the badge.

US-N02: Mark All Notifications Read

Front Card

As a user, I want to mark all notifications as read so I can clear my queue quickly.

Back Card

Success

- Bulk "mark all as read" updates all notifications for the current user with `read=true`.
- Unread badge resets to zero immediately without requiring a manual page reload.

Failure

- Only a subset of notifications is updated, leaving some still unread in the backend.
- The UI still shows stale unread counts or badges after the operation completes.

US-N03: Assignment Published Notification

Front Card

As a student, I want a notification when a new assignment is published so I can start work promptly.

Back Card

Success

- A notification is created for all enrolled students when an assignment is published, containing the assignment title and due date.
- The unread badge increments appropriately and the new item appears in the notification dropdown feed and /notifications page.

Failure

- No notification is created for eligible students after an assignment is published.
- Badge count is not updated or remains stale, even though notifications exist in the backend.

US-N04: Assignment Grading Notification

Front Card

As a student, I want to receive a notification when my assignment is graded or when the instructor updates the grade of a submitted assignment so I can review feedback promptly.

Back Card

Success

- A notification is created immediately after the instructor assigns or updates a grade, including the assignment title and the grade/score.
- The notification links directly to the graded submission view where feedback comments are visible.

Failure

- No notification is created despite a grade being recorded or updated in the system.
- Notification refers to an incorrect assignment or displays an outdated grade value.

US-N05: Real-Time Updates (Polling / SSE)

Front Card

As a user, I want near real-time notification updates so I do not miss important events while active.

Back Card

Success

- Polling with an interval of at most 30 seconds fetches new notifications while the user session is active.
- When SSE is enabled, new notifications are pushed instantly to the client without a full page reload.

Failure

- Important events are missed or only appear after a manual refresh while the user is actively on the site.
- SSE connections are unstable (frequent disconnects, memory leaks, or repeated reconnections affecting performance).

US-N06: Assignment Submission Notification to Instructor

Front Card

As an instructor, I want a notification when a student submits an assignment so I can review it promptly.

Back Card

Success

- A single notification is created per submission, containing the student identifier and assignment identifier.
- The notification appears in the instructor notification feed and increments the unread count correctly.

Failure

- Multiple duplicate notifications are generated for the same submission event.
- Missing or incorrect metadata (such as wrong student ID or assignment ID) is attached to the notification.

2.2 AI Learning Assistant

2.2.1 Quiz Generation

US-Q01: Topic-Based Quiz Generation

Front Card

As a student, I want to generate quizzes based on a topic so I can practice concepts more effectively.

Back Card

Success

- Quiz is generated using the selected topic, question type (MCQ/True-False), grade level, and the requested number of questions.
- Questions are relevant to the topic, clearly structured, and include correct answers and answer keys.
- Quiz appears instantly in the UI without requiring a full page reload.

Failure

- Generated quiz is irrelevant to the topic or poorly formatted.
- Answer keys are missing or incorrect.
- The "Generate" button becomes unresponsive or leads to UI errors.

US-Q02: Quiz Type Selection (MCQ / True-False)

Front Card

As a user, I want to choose the quiz type so I can practice in a format that suits me.

Back Card

Success

- The selected quiz type (MCQ or True-False) is correctly reflected in the generated questions.
- MCQ questions include viable multiple options, and True-False questions produce valid Boolean statements.
- The toggle or selection control clearly indicates the active quiz type to the user.

Failure

- The generated quiz ignores the selected type and produces mismatched question formats.
- Options are missing, malformed, or duplicated.
- The UI toggle state appears inconsistent with the actual quiz content.

US-Q03: Grade-Level Filtering

Front Card

As a student, I want to choose a grade level so the quiz difficulty matches my academic standard.

Back Card

Success

- Selected grade level influences quiz difficulty, vocabulary, and depth of questions.
- The UI clearly reflects the chosen grade level.
- Quiz content aligns with general curriculum expectations for the chosen grade.

Failure

- Grade selection has no noticeable effect on quiz difficulty or content.
- Dropdown fails to update, shows incorrect grade labels, or behaves inconsistently.
- Generated questions are too advanced or too basic for the selected grade level.

US-Q04: Select Number of Questions

Front Card

As a user, I want to select the number of quiz questions so I can control the quiz length.

Back Card

Success

- Generated quiz contains exactly the number of questions selected by the user.
- Question count selector (dropdown/slider) operates smoothly and reflects the correct value.

Failure

- Quiz generates more or fewer questions than requested.
- UI resets or ignores the selected question count when generating a new quiz.

US-Q05: Quiz History Management (View + Clear)

Front Card

As a user, I want to view and clear my quiz history so I can track learning progress or reset past attempts.

Back Card

Success

- "Show History" displays accurate records of previously generated quizzes, including topic, grade, type, and timestamp.
- "Clear History" removes all saved quiz attempts from storage.
- A newly generated quiz starts with a clean state after history is cleared.

Failure

- Quiz entries in history are missing, duplicated, or out of order.
- "Clear History" only partially deletes items or appears to work but leaves stale data.
- New quiz generation incorrectly reuses data from older quizzes.

2.2.2 AI Tutor Chatbot

US-C01: Ask a Tutor – Topic-Based AI Answering

Front Card

As a student, I want to ask the AI tutor questions by topic so I can get immediate academic help.

Back Card

Success

- User enters both a topic and a detailed question into the AI tutor interface.
- AI responds with accurate, structured, and topic-aligned explanations.
- Responses appear promptly and clearly in the chat UI without layout glitches.

Failure

- AI responses are irrelevant, incoherent, or off-topic.
- Long delays occur frequently or the UI shows empty/missing answers.
- The chat output fails to render properly in the interface.

US-C02: Detailed Question Input

Front Card

As a student, I want to describe my question in detail so the AI can generate precise, helpful responses.

Back Card

Success

- The input area supports long, detailed questions without breaking the layout.
- AI generates complete explanations with steps, context, and structured reasoning where appropriate.
- The layout manages long text gracefully without overflow or clipping issues.

Failure

- Large questions cause formatting issues or content truncation.
- AI responses are cut off, incomplete, or abruptly terminated.
- Input validation behaves inconsistently or incorrectly rejects valid questions.

US-C03: Chat History Management (View + Clear)

Front Card

As a user, I want to view and clear my chat history so I can revisit past conversations or reset for new ones.

Back Card

Success

- History view shows previously asked questions with corresponding AI answers, ordered by time.
- "Clear History" removes all saved chat records for the user.
- "New Chat" resets input fields and the active chat window to an empty state.

Failure

- Some chats are missing from history or appear in an incorrect order.
- "Clear History" does not fully reset stored data.
- "New Chat" still shows old messages or stale context.

US-C04: Reliable Answer Generation

Front Card

As a user, I want the AI tutor to respond reliably so I can trust it during study sessions.

Back Card

Success

- Responses are generated consistently within an acceptable response time.
- Submit button is disabled while a request is in progress to prevent duplicates.
- The chat interface remains responsive without freezing or crashing during heavy use.

Failure

- Multiple submissions create duplicate or overlapping answers.
- Slow, intermittent failures or broken formatting degrade the learning experience.
- Chat becomes stuck or unresponsive, requiring a full page reload.

3. Proof of Concept (POC) - Notification System

3.1 Notification Views

This section provides a visual overview of how notifications appear to instructors and students within the platform.

- **1. Instructor Side Notification:** A mock or live view of the instructor notification dropdown and the full /notifications page, showing assignment submissions, grading events, and course-related updates.

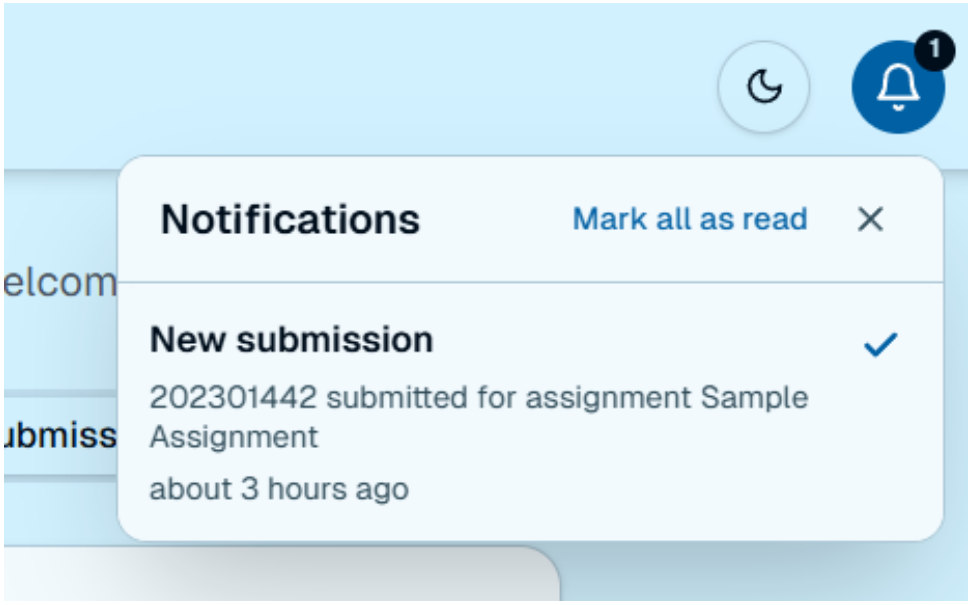


Figure 3.1: Notification for new submission on instructor side.

- **2. Student Side Notification:** A sample screen demonstrating student notifications for assignment publish events, grading updates, enrollments, and stream posts.

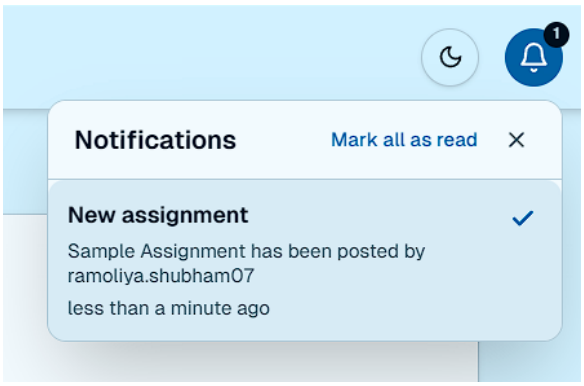


Figure 3.2: New Assignment Notification

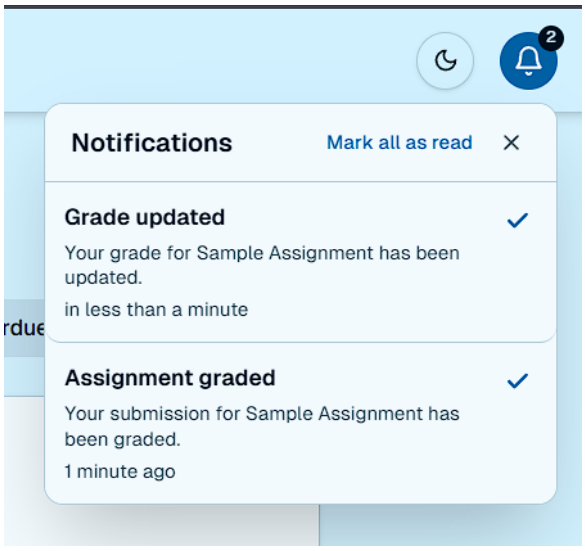


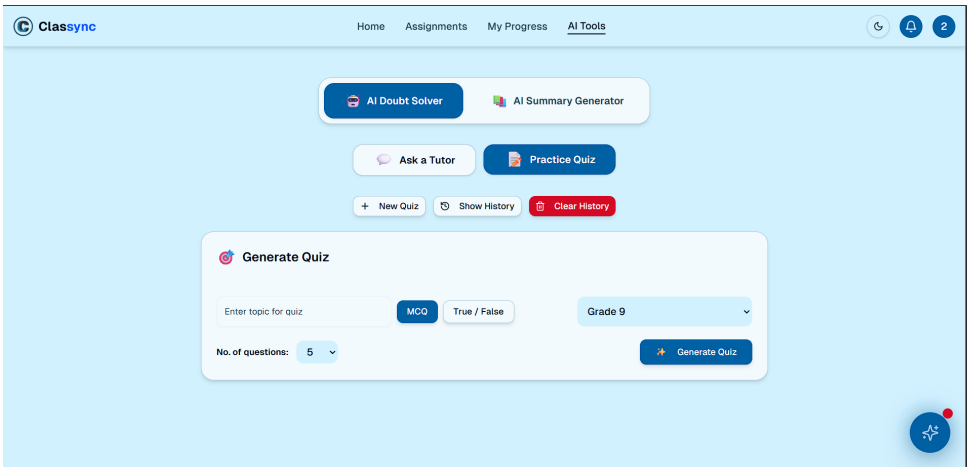
Figure 3.3: Assignment grading and grade updating notification

3.2 AI Learning Assistant

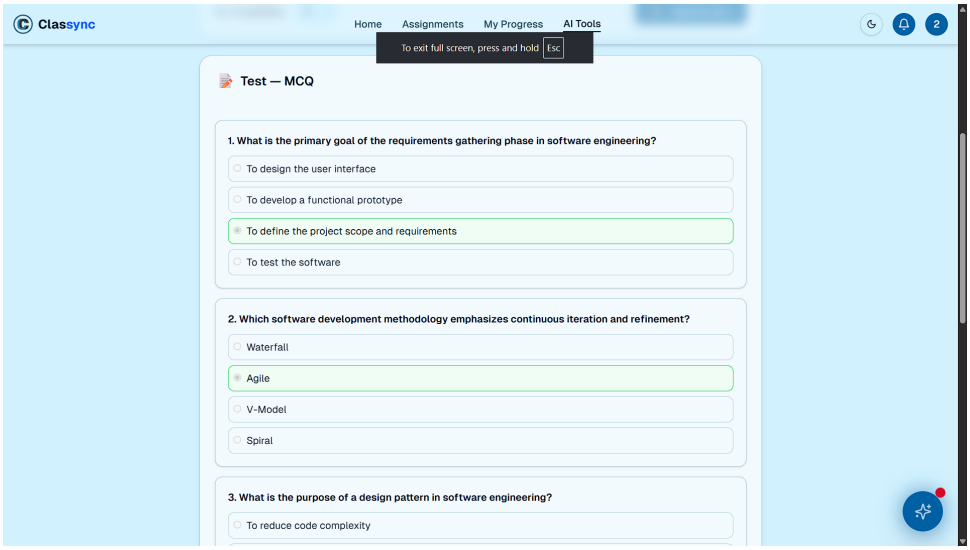
3.2.1 Quiz Flow

This section provides a visual overview of how quiz generation and practice flows work in the application.

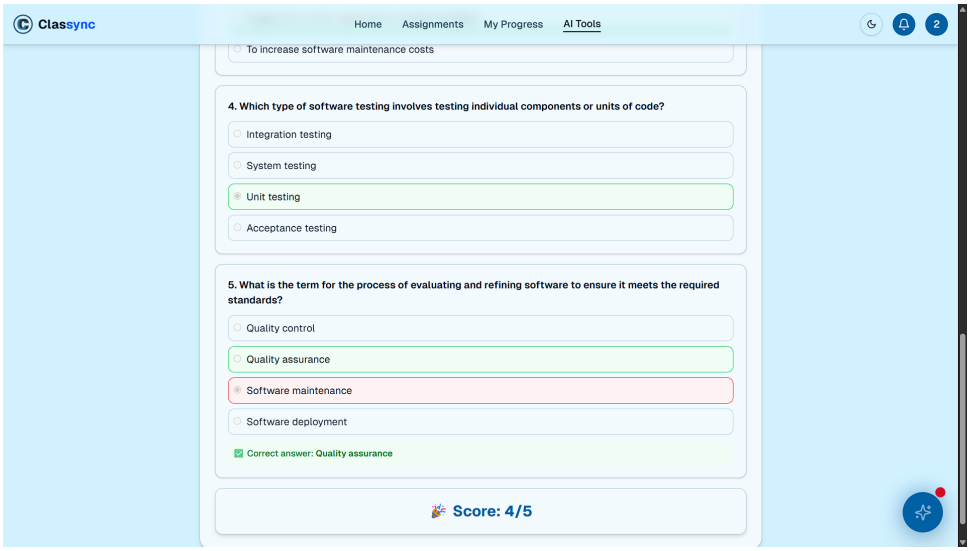
- **1. Quiz Generator Panel:** Topic input, question type selector (MCQ/True-False), grade dropdown, and question count selector allowing users to configure their quiz.



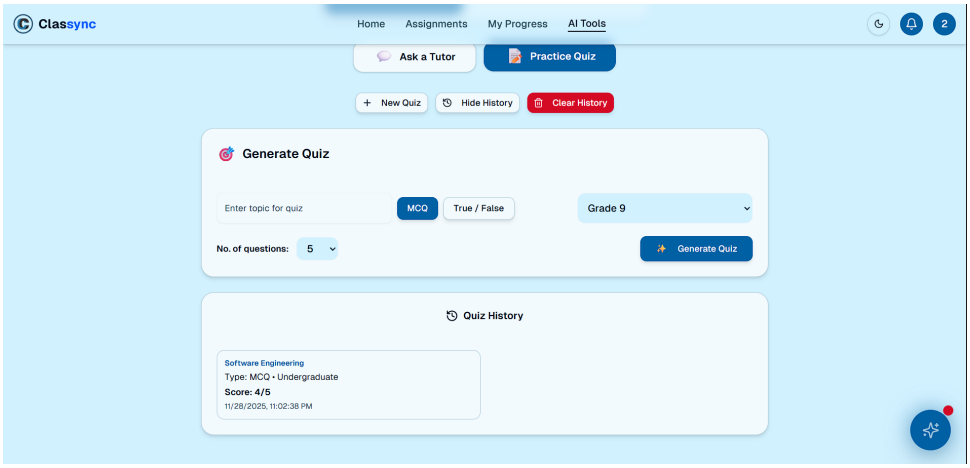
- **2. Generated Quiz Output:** AI-generated questions rendered in a clean card layout, with options and answer keys clearly visible.



- **3. Practice Quiz View:** Students interactively attempt the generated quiz by selecting answers for each question.

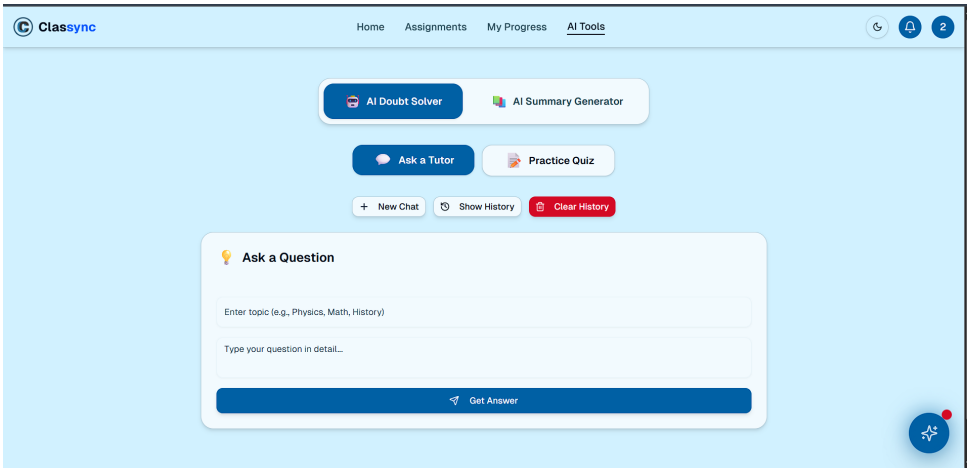


- **4. Quiz History Page:** List of previously generated quizzes, each showing timestamp, topic, question type, and grade level.

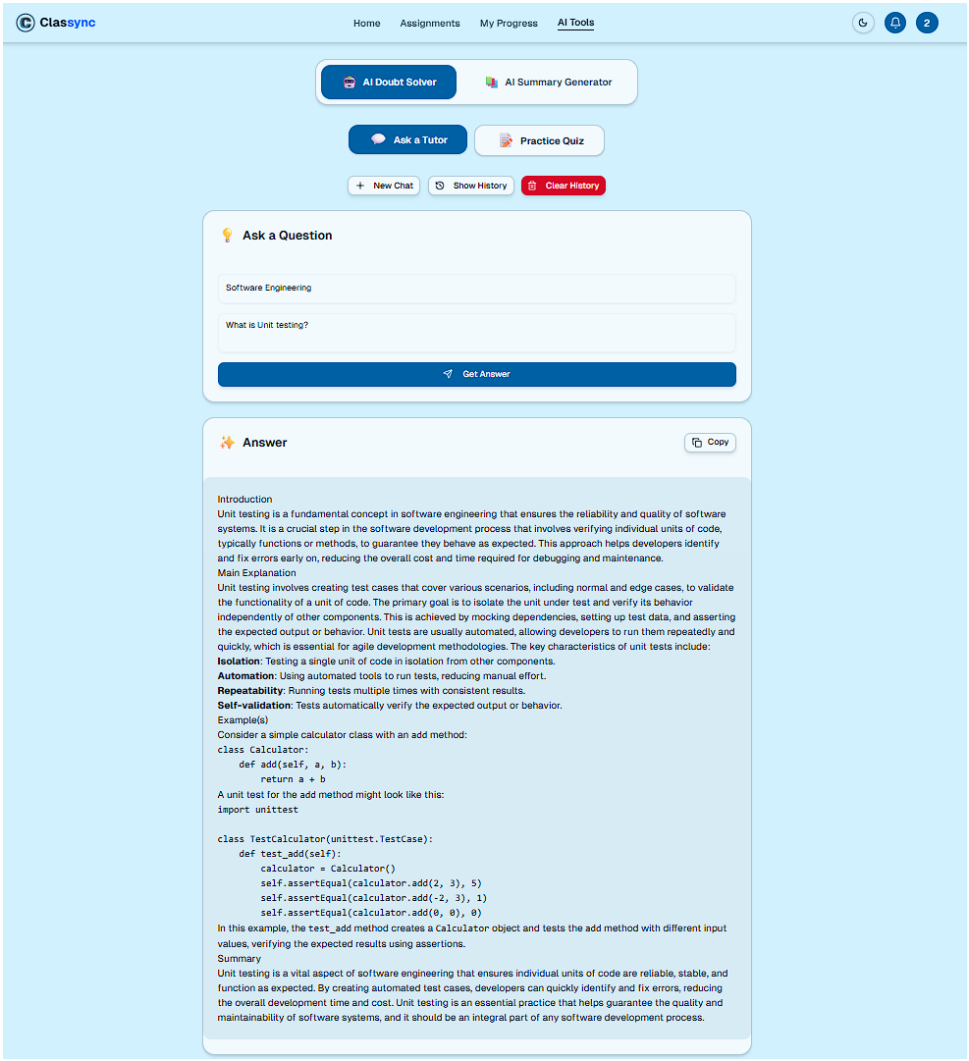


3.2.2 AI Chatbot Flow

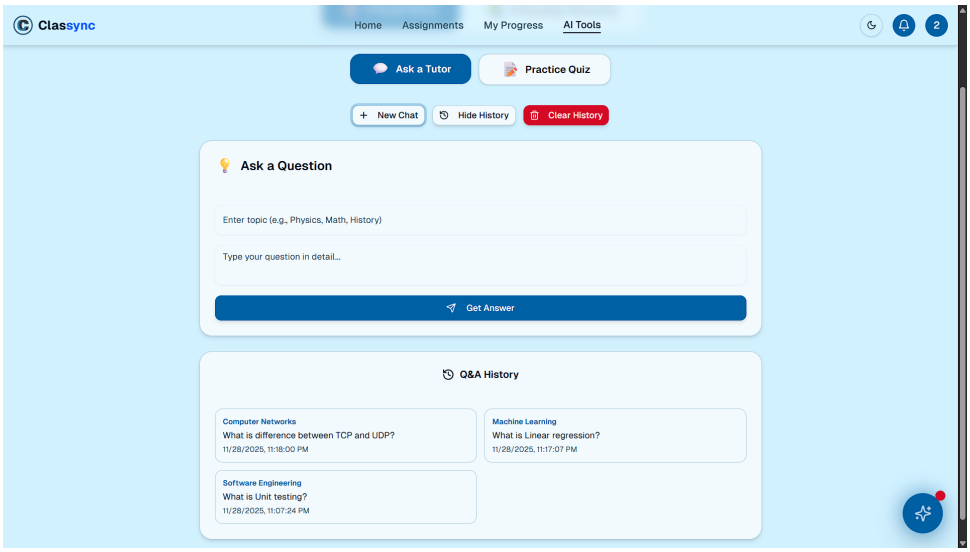
- **1. AI Chat Window:** Chat interface that opens when the user selects the AI Assistant, with inputs for topic and detailed question.



- **2. AI Tutor Response:** Structured, educational answers rendered in the chat stream, tailored to the user's topic and query.

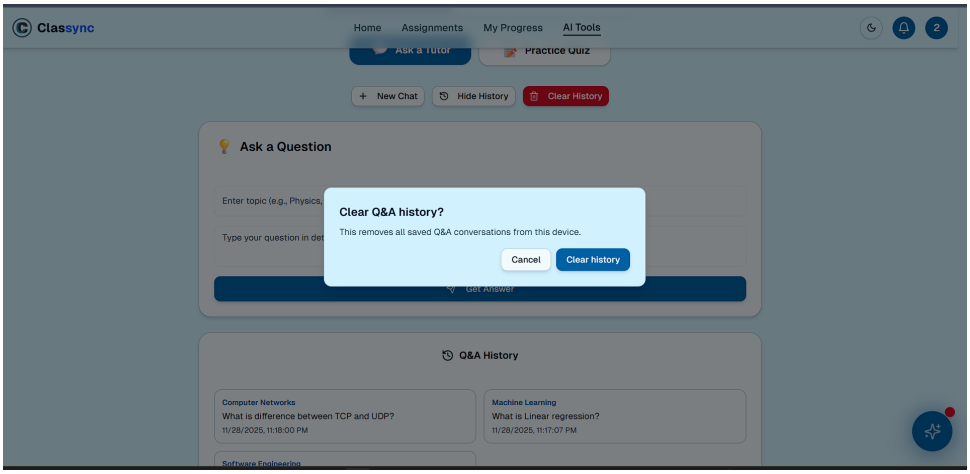


- **3. Chat History Page:** Dedicated history view listing previous Q&A interactions saved for later reference.



- **4. Clear Chat History:** Confirmation dialog for clearing chat history and demon-

stration of the UI resetting to a fresh state.



4. Sprint Review

4.1 Notification System

4.1.1 What We Achieved

Data Model and Backend

- Designed and implemented a unified `notifications` model with fields such as `userId`, `type`, `title`, `body`, `read`, `createdAt`, and a flexible `meta` JSON object for contextual data.
- Added a dedicated `notificationPreferences` model to capture per-user preferences for notification types and delivery channels (in-app, email).
- Created indexed queries for `userId` and `read` fields to ensure fast retrieval of unread notifications even at scale.

APIs and Event Integration

- Implemented REST endpoints for listing notifications with filters (unread, type, pagination), bulk marking as read, and updating notification preferences.
- Added a stretch endpoint for SSE-based streaming of notifications, allowing the frontend to listen for real-time events without heavy polling.
- Hooked domain events (assignment publish, assignment submission, course enroll/unenroll, group updates, stream posts, email verification) into the notification emitter layer so that each event results in a consistent notification payload.

UI and User Experience

- Enhanced the existing Notification Bell component to show an unread badge, a rich dropdown with the latest 10 notifications, and a "Mark all as read" action.
- Built a full `/notifications` page to browse historical notifications with pagination and filters.
- Implemented a preferences modal that allows users to enable/disable notification types and toggle email delivery, with preferences enforced before persistence.

Reliability and Testing

- Added an in-memory queue abstraction and basic retry mechanism for transient notification delivery failures.
- Achieved at least 80% test coverage for notification-related models, emitters, API routes, and React components.
- Wrote integration tests to validate end-to-end flows, such as "assignment published" → event emitted → notification created → badge updated on UI.

4.1.2 Challenges

Real-Time Delivery and Performance

Challenge 1: Balancing Polling vs. SSE Issue: Pure SSE could introduce complexity and instability under unreliable network conditions, while polling alone might feel sluggish.

Solution: Adopted a hybrid approach where all clients default to 30-second polling, and SSE can be turned on for supported browsers. SSE includes reconnection backoff and heartbeat checks.

Learning: Understood trade-offs between real-time technologies and how to design a progressive enhancement strategy rather than a hard dependency.

Challenge 2: Avoiding Duplicate Events Issue: Some domain events (like resubmitting assignments or repeated saves) risked generating duplicate notifications.

Solution: Introduced idempotency keys at the emitter layer and added unique constraints over a tuple (userId, type, meta-hash) within a time window.

Learning: Learned how to design idempotent event-driven flows and prevent noisy notification streams.

User Preference Enforcement

Challenge 3: Preference-Aware Delivery Issue: Notification events are global, but delivery must respect per-user preferences (e.g., some students may disable email notifications).

Solution: Centralized preference checks inside the notification service before persisting or sending any notification and cached preferences per user for performance.

Learning: Importance of a single decision point for policy enforcement and separation of "event happened" from "should this user see it?".

4.2 AI Learning Assistant

4.2.1 What We Achieved

Data Model and Storage

- Defined `quizAttempts` with fields: `topic`, `grade`, `type`, `questions[]` (including answers), and `createdAt`.
- Defined `chatSessions` with fields: `topic`, `question`, `answer`, and `createdAt`.
- Implemented history storage either locally (browser storage) or via backend persistence with timestamps, depending on deployment constraints.

APIs and Core Logic

- Implemented `/quiz/generate` to create a quiz from topic, type, grade level, and question count using the AI backend.
- Implemented `/tutor/ask` to generate AI answers for detailed academic questions.
- Implemented `/history/list` and `/history/clear` for fetching and clearing quiz/chat history, abstracting over local vs. remote storage.

Event Flow and UI

- "Generate Quiz" triggers AI generation, displays structured questions, and then persists the quiz attempt into history.
- "Ask Tutor" sends the detailed question, receives explanation, and persistently stores the Q&A pair.
- "New Quiz" and "New Chat" actions reset UI state cleanly for fresh sessions, while "Show History" loads entries with the latest first.

Reliability and Testing

- Added input validation to all user-facing forms (topic, grade, type, question count, and detailed question text).
- Implemented loading states and disabled buttons during AI calls to avoid duplicate requests.
- Added fallback messages for network errors, timeouts, or AI failures so the UI fails gracefully.

- Wrote unit tests for AI formatting functions, validation logic, and history helpers, plus integration tests for end-to-end quiz and chat flows.
- Achieved at least 80% coverage across newly introduced modules in this sprint.

4.2.2 Challenges

Predictability of AI Output

Challenge 1: Consistent Quiz Structure Issue: Raw AI responses were sometimes inconsistent in format, making it hard to render reliable quiz UIs.

Solution: Implemented a strict output schema and post-processing layer to validate and normalize AI output before it reached the UI.

Learning: Learned the importance of schema validation and deterministic formatting when integrating generative AI in user-facing workflows.

History Management

Challenge 2: History Size and Performance Issue: Large histories for quizzes and chat sessions risked slowing down retrieval or cluttering the UI.

Solution: Introduced pagination and limits for history views, as well as pruning strategies (e.g., maximum number of saved entries per user).

Learning: Understood the trade-offs between rich history and long-term performance, and how to set sensible limits for educational tools.

5. Conclusion

5.1 Notification System

By the end of this sprint, users receive timely, configurable notifications that significantly improve engagement and awareness of key classroom events. The architecture supports incremental real-time improvements and analytics without heavy refactoring. Success for this sprint is measured by the functional POC, working integration of emitters for key events (assignments, enrollments, stream posts), a smooth UI experience for badges and feeds, and achieving the targeted test coverage threshold.

5.2 AI Learning Assistant

By the end of this sprint, students can quickly generate topic-based quizzes and ask detailed academic questions with accurate AI responses. The system delivers stable performance, clear UI behavior, and reliable history tracking across quizzes and chat interactions. The architecture is ready for future enhancements such as difficulty scaling, exporting quizzes as PDF, or richer tutoring interactions, all while maintaining strong test coverage and code quality.