AI Tutor Submission

Team 7
Spencer Thompson
Landon Towers

October 26, 2025

Contents

1 Team Organization

Team 7

- Spencer Thompson
- Landon Towers

Team Organization

- Spencer Thompson (Sprint Lead)
- Landon Towers
- **Meeting schedule:**
  - **Professor:** Thursdays at 11:15 AM
  - **Sponsor:** Mondays at 2:00 PM
  - **Team:** Ad-hoc, scheduled weekly.

2 Project Description

The AI Tutor is a generative AI-powered learning assistant designed to support students at Utah Valley University (UVU). It provides a personalized and interactive learning experience by integrating with the Canvas Learning Management System (LMS). The application consists of a web-based frontend and a robust backend, working together to deliver a seamless and intuitive user experience.

The frontend is a modern chat interface built with Svelte. It allows students to ask questions and receive answers from an AI tutor in a conversational manner. The interface supports rich text formatting, including markdown, code snippets with syntax highlighting, and mathematical equations using LaTeX. This ensures that the AI can provide clear and easy-to-understand explanations for a wide range of subjects.

The backend is the core of the AI Tutor, developed using Python and the FastAPI framework. It orchestrates the entire system, managing user authentication, data processing, and communication with external services. The backend uses a MongoDB database to store user data, chat history, and course information.

One of the key features of the AI Tutor is its "smart chat" functionality. By securely accessing a student's data from Canvas—including their enrolled courses, assignments, grades, and activity stream—the AI can provide context-aware and personalized assistance. For example, a student can ask for a summary of their upcoming assignments, clarification on a specific course topic, or help with a difficult concept from a lecture. The AI can also be customized by the user with a bio, to further personalize the experience.

To answer a wide range of questions, the AI Tutor is equipped with several tools. It can read the content of webpages, which is useful for providing explanations based on external resources. It can also execute Python code, allowing it to function as a calculator or to demonstrate programming concepts.

The AI Tutor is designed to be a safe and reliable learning environment. It includes a moderation service that filters out inappropriate content, and it uses a secure authentication system based on JSON Web Tokens (JWT) to protect user data. The application is also designed to be scalable and maintainable, with a containerized architecture using Docker.

In summary, the AI Tutor is a comprehensive and innovative learning platform that combines the power of generative AI with the rich data available in the Canvas LMS. It aims to provide UVU students with a powerful and accessible tool to enhance their learning experience and academic success.

3 Meeting Log & Contribution Form

Week Five Meeting Log

AI Tutor

Spencer Thompson | Landon Towers

9/21/2025

| What have you done since last team meeting?

- We have started adding some documentation for what we are working on as well as refactoring and fixing some pieces of the code that are problematic.

| What obstacles are you encountering?

- Determining which features to implement is difficult, hopefully we can nail down exactly what we want to and will be able to work on.

| What do you plan to accomplish by the next team meeting?

- Have a solid plan for features.

| Contributions

Team: 7	Sprint: 1	Date: 9/21/2025	Team Score: 100%
Name:	**Contribution (%):**	**Signature:**	**Individual Score:**
Spencer Thompson	50%	Spencer Thompson	100%
Landon Towers	50%	Landon Towers	100%

| Notes

- Getting back into the swing of school after the chaos of the last couple weeks was tough.

Week 5 Log

Week Six Meeting Log

AI Tutor

Spencer Thompson | Landon Towers

9/28/2025

| What have you done since last team meeting?

- We started working on features, particularly moving to a new and improved API for OpenAI as well as nailing down a staging environment.

| What obstacles are you encountering?

- Not really much, it has mostly been smooth sailing apart from our busy schedules.

| What do you plan to accomplish by the next team meeting?

- Finish out the migration to the new API and have a solid development and staging environment.

| Contributions

Team: 7	Sprint: 1	Date: 9/28/2025	Team Score: 100%
Name:	**Contribution (%):**	**Signature:**	**Individual Score:**
Spencer Thompson	50%	Spencer Thompson	100%
Landon Towers	50%	Landon Towers	100%

| Notes

- Getting back into the swing of school after the chaos of the last couple weeks was tough.

Week Seven Meeting Log

AI Tutor

Spencer Thompson | Landon Towers

10/5/2025

| What have you done since last team meeting?

- We were able to meet with the tech management department to find their concerns as well as nail down the features they want the tutor to have.

| What obstacles are you encountering?

- Given that this project is related to several different departments and people at UVU, it is difficult to do everything we need to keep everyone happy.

| What do you plan to accomplish by the next team meeting?

- Deploy some new code.

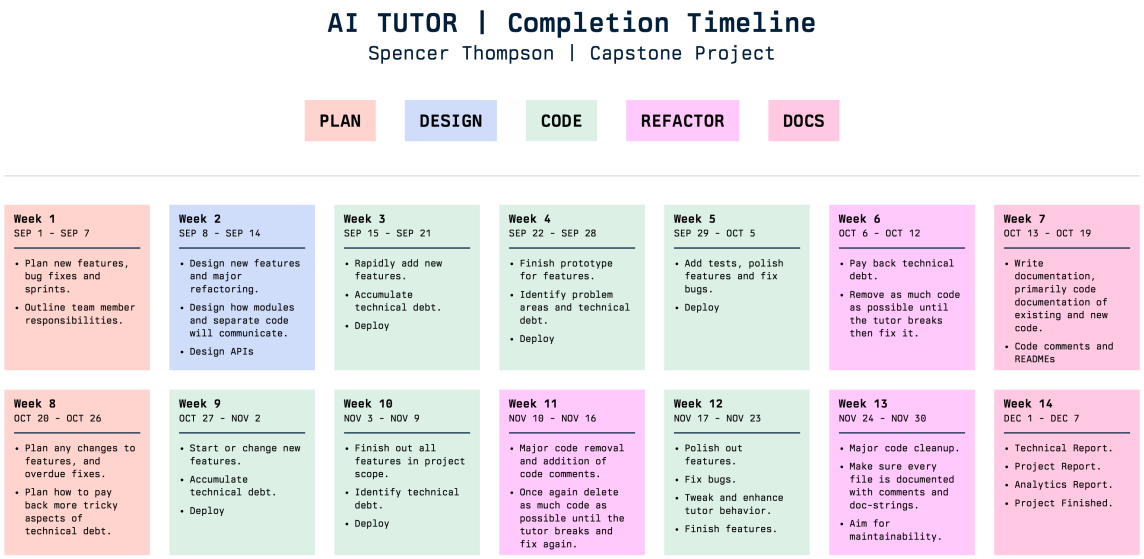
| Contributions

Team: 7	Sprint: 1	Date: 10/5/2025	Team Score: 100%
Name:	**Contribution (%):**	**Signature:**	**Individual Score:**
Spencer Thompson	50%	Spencer Thompson	100%
Landon Towers	50%	Landon Towers	100%

| Notes

- N/A

4 Backlogs and Sprints



Timeline, Description, and Backlog

Sprint 1 (Sep 1 - Sep 14)

- **Landon:** Plan new features, bug fixes and sprints.
- **Spencer:** Outline team member responsibilities.
- **Landon:** Design new features and major refactoring.
- **Spencer:** Design how modules and separate code will communicate.
- **Landon:** Design APIs.

Sprint 2 (Sep 15 - Sep 28)

- **Spencer:** Rapidly add new features.
- **Landon:** Accumulate technical debt.
- **Spencer:** Deploy.
- **Landon:** Finish prototype for features.
- **Spencer:** Identify problem areas and technical debt.
- **Landon:** Deploy.

Sprint 3 (Sep 29 - Oct 12)

- **Spencer:** Add tests, polish features and fix bugs.
- **Landon:** Deploy.
- **Landon:** Pay back technical debt.
- **Spencer:** Remove as much code as possible until the tutor breaks then fix it.

Sprint 4 (Oct 13 - Oct 26)

- **Spencer:** Write documentation, primarily code documentation of existing and new code.
- **Landon:** Code comments and READMEs.
- **Landon:** Plan any changes to features, and overdue fixes.
- **Spencer:** Plan how to pay back more tricky aspects of technical debt.

Sprint 5 (Oct 27 - Nov 9)

- **Spencer:** Start or change new features.
- **Landon:** Accumulate technical debt.
- **Spencer:** Deploy.
- **Landon:** Finish out all features in project scope.
- **Spencer:** Identify technical debt.
- **Landon:** Deploy.

Sprint 6 (Nov 10 - Nov 23)

- **Spencer:** Major code removal and addition of code comments.
- **Landon:** Once again delete as much code as possible until the tutor breaks and fix again.
- **Landon:** Polish out features.
- **Spencer:** Fix bugs.
- **Landon:** Tweak and enhance tutor behavior.
- **Spencer:** Finish features.

Sprint 7 (Nov 24 - Dec 7)

- **Spencer:** Major code cleanup.
- **Landon:** Make sure every file is documented with comments and doc-strings.
- **Spencer:** Aim for maintainability.
- **Landon:** Technical Report.
- **Spencer:** Project Report.
- **Landon:** Analytics Report.
- **Spencer:** Project Finished.

5 Software Requirement Specification (SRS)

5.1 Functional Requirements

5.1.1 User Management

- **F1.1:** Users must be able to log in to the system using their UVU credentials via Canvas OAuth2.
- **F1.2:** The system must use JSON Web Tokens (JWT) for authenticating API requests after the initial login.
- **F1.3:** Users shall be able to set a custom bio in their profile. The content of this bio will be included in the system prompt sent to the AI to influence its responses.
- **F1.4:** The system shall provide an administrative dashboard that displays the following real-time analytics: total number of users, number of active users, and total messages sent.

5.1.2 Chat Interface

- **F2.1:** The system shall provide a web-based chat interface for users to interact with the AI Tutor.
- **F2.2:** The chat interface must display the conversation history, with user and AI messages clearly distinguished.
- **F2.3:** The system shall stream messages to the client in real-time using websockets or a similar technology.
- **F2.4:** The chat interface shall render AI responses in GitHub-flavored markdown, supporting tables, lists, bold, italics, and other standard formatting.
- **F2.5:** The system must render mathematical equations formatted using LaTeXsyntax.
- **F2.6:** The system must provide syntax highlighting for code snippets in Python, JavaScript, Java, C++, and SQL.

5.1.3 AI Tutor

- **F3.1:** The AI Tutor shall generate responses to user queries using a large language model (LLM) specified in the system configuration (e.g., GPT-4, Claude 3).
- **F3.3:** The system shall provide a ”smart chat” feature that injects context from the user's Canvas data into the LLM prompt to generate personalized responses.
- **F3.4:** The AI Tutor shall be able to retrieve the user's enrolled courses, upcoming assignments, and recent grades from the Canvas API.
- **F3.5:** The AI Tutor shall have the ability to use the following tools to perform specific tasks:
  - **F3.5.1:** A web scraper tool that can read the full text content of a webpage given a URL.
  - **F3.5.2:** A Python code interpreter that can execute sandboxed Python code to perform calculations or demonstrate programming concepts.

- **F3.6:** All user input and AI-generated responses must be passed through a content moderation filter. Any content flagged as inappropriate (e.g., hate speech, violence) shall be blocked and logged.

5.2 Non-functional Requirements

5.2.1 Security

- **NF1.1:** All network communication between the client and server must be encrypted using TLS 1.2 or higher.
- **NF1.2:** The system must be protected against the OWASP Top 10 web vulnerabilities, which includes Cross-Site Scripting (XSS) and NoSQL Injection.
- **NF1.3:** Access to the backend API must be restricted to authenticated users with valid JWTs. Direct access via API keys is prohibited.
- **NF1.4:** All user data, including Canvas information and chat history, must be encrypted at rest in the database using AES-256 encryption.

5.2.2 Performance

- **NF2.1:** The user interface shall have a response time of less than 200ms for all user interactions (e.g., button clicks, page loads).
- **NF2.2:** The first token of a streamed AI response shall be delivered to the client in under 2 seconds, on average.
- **NF2.3:** The system shall initially support 100 concurrent users with an average API response time of under 500ms.

5.2.3 Usability

- **NF3.1:** A new user must be able to successfully send a message in "smart chat" mode within 60 seconds of their first login without requiring documentation.
- **NF3.2:** All error messages displayed to the user must include a unique error code and a clear, human-readable explanation of the problem.
- **NF3.3:** The AI Tutor's responses shall achieve a Flesch-Kincaid grade level score between 8 and 12 to ensure they are understandable to a broad audience.

5.2.4 Scalability

- **NF4.1:** The system shall be horizontally scalable. An increase in container instances must result in a proportional increase in user capacity.
- **NF4.2:** The application shall be fully containerized using Docker, with all services defined in a `docker-compose.yml` file for automated deployment and scaling.

5.2.5 Reliability

- **NF5.1:** The system shall have a service availability of 99.9% (uptime).
- **NF5.2:** The system shall have a Mean Time To Recovery (MTTR) of less than 15 minutes.
- **NF5.3:** The content moderation filter shall have a false negative rate of less than 1% for clearly inappropriate content.

5.2.6 Maintainability

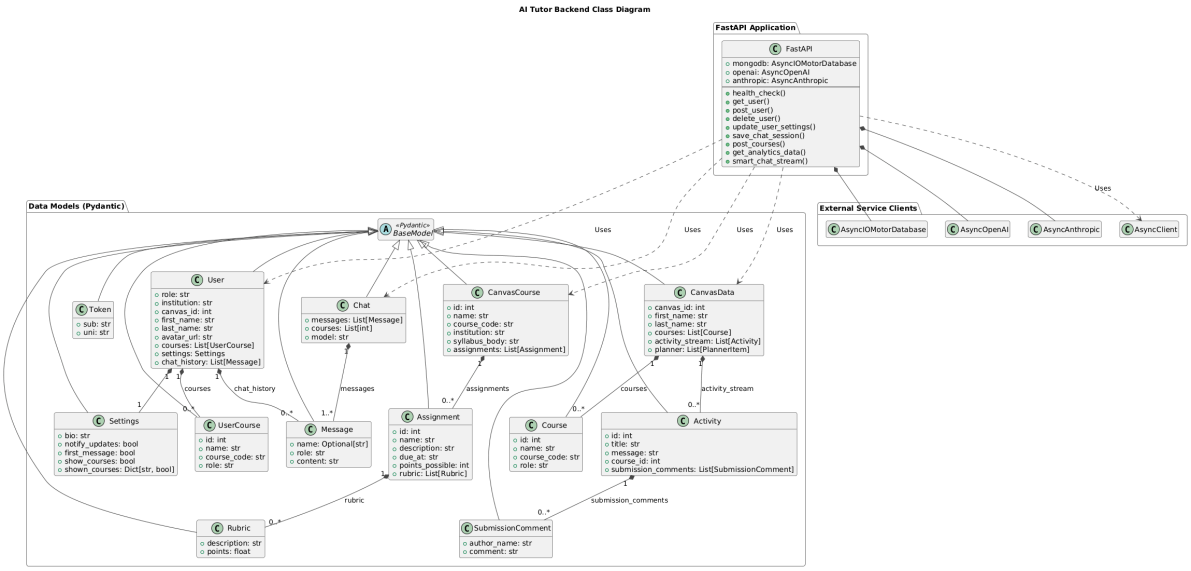
- **NF6.1:** All Python code must adhere to the PEP 8 style guide, enforced by an automated linter.
- **NF6.2:** The application shall have a modular design, with a clear separation of concerns between the data, business logic, and presentation layers.
- **NF6.3:** All new code committed to the main branch must have a minimum of 80% unit test coverage.

6 Demo Code

Not available

7 Design

- Class Diagram (High Cohesion and Loose Coupling)

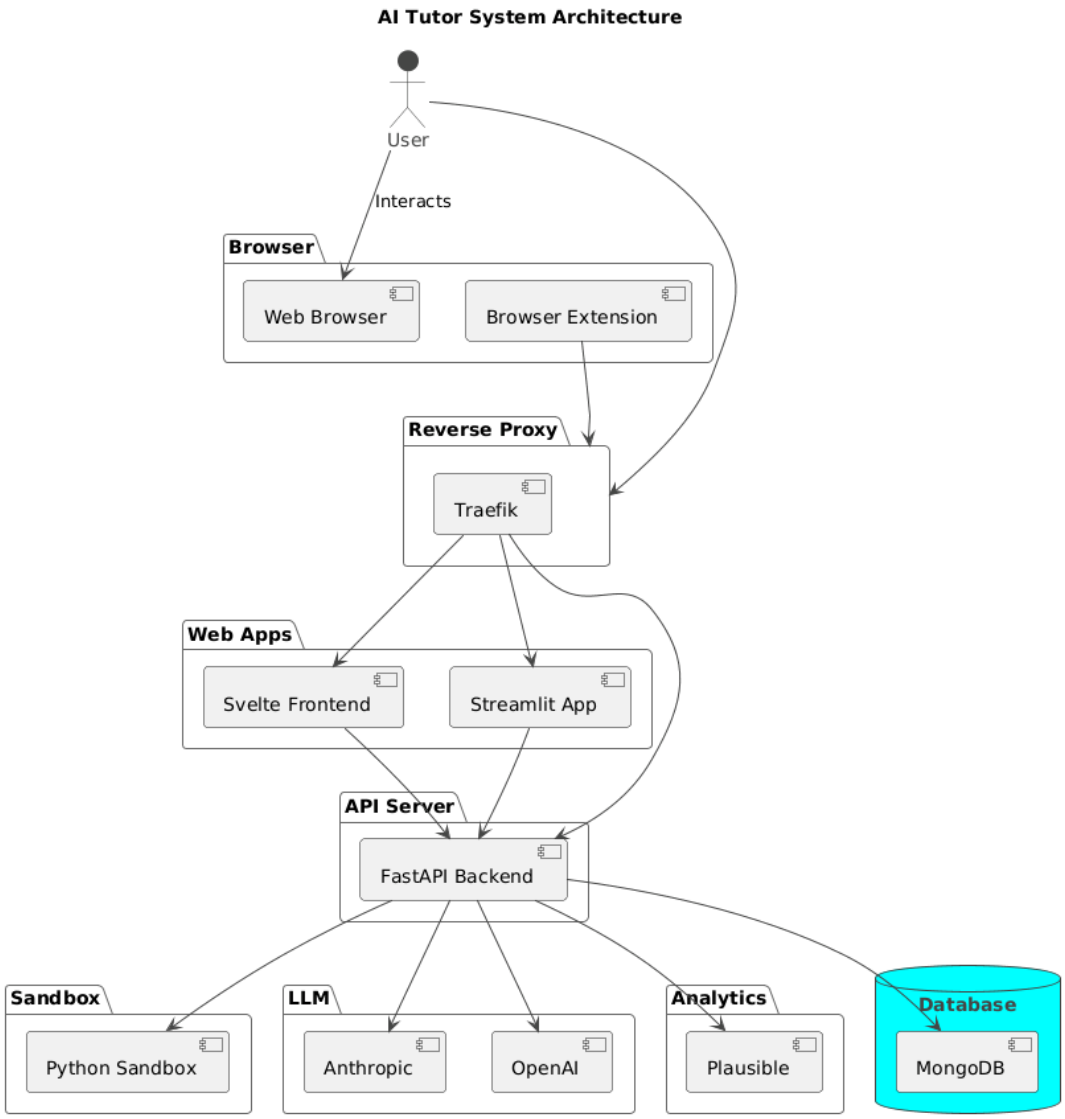


Class Diagram

- Design patterns
  - **Singleton Pattern:** The FastAPI application instance (`app`) and the MongoDB client (`app.mongodb_client`) are initialized once and used throughout the application's lifecycle. This is managed by the `db_lifpan` async context manager, ensuring a single, shared connection to the database.
  - **Strategy Pattern:** The backend can use different large language models (e.g., OpenAI, Anthropic) to generate responses. The `openai_iter_response` and `anthropic_iter_response` functions in `ai.py` represent different strategies for the same task. The application can select the strategy based on the user's choice of model.
  - **Decorator Pattern:** The FastAPI framework uses decorators extensively to define API endpoints (e.g., `@app.get`, `@app.post`). These decorators add routing and other functionality to the endpoint functions without modifying their core logic, which is a clear example of the Decorator pattern.
- Architecture diagram
- ERD diagram
- Use case diagram

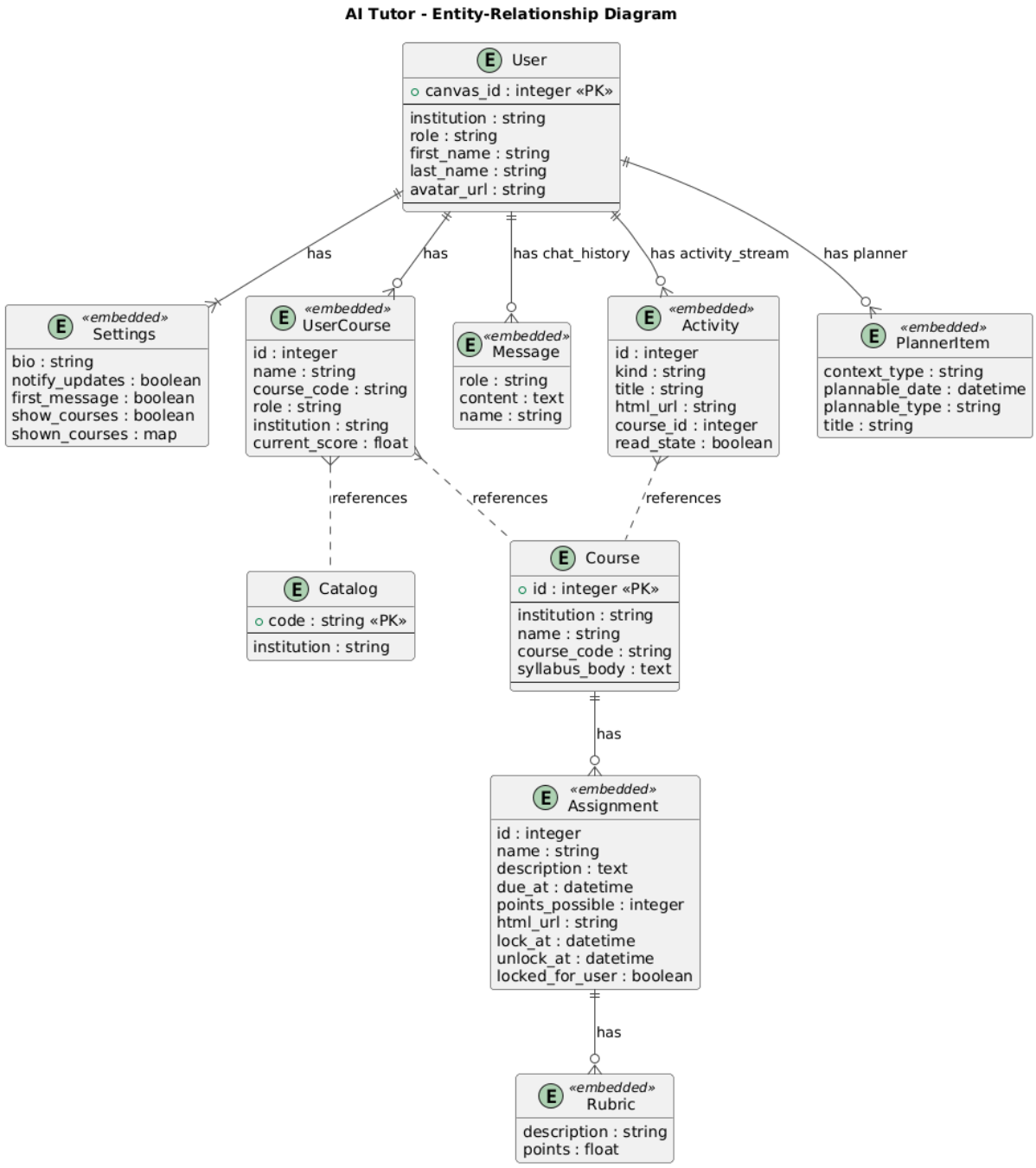
7.1 Use Case Descriptions

- **Authenticate with UVU ID:** The user logs into the system. The system initiates an OAuth2 flow with the external Canvas LMS, which handles the actual authentication using the user's UVU credentials.
- **Fetch User Profile:** The user retrieves their current profile information from the system, such as their name and custom bio, which is then displayed in the application's settings interface.



Architecture Diagram

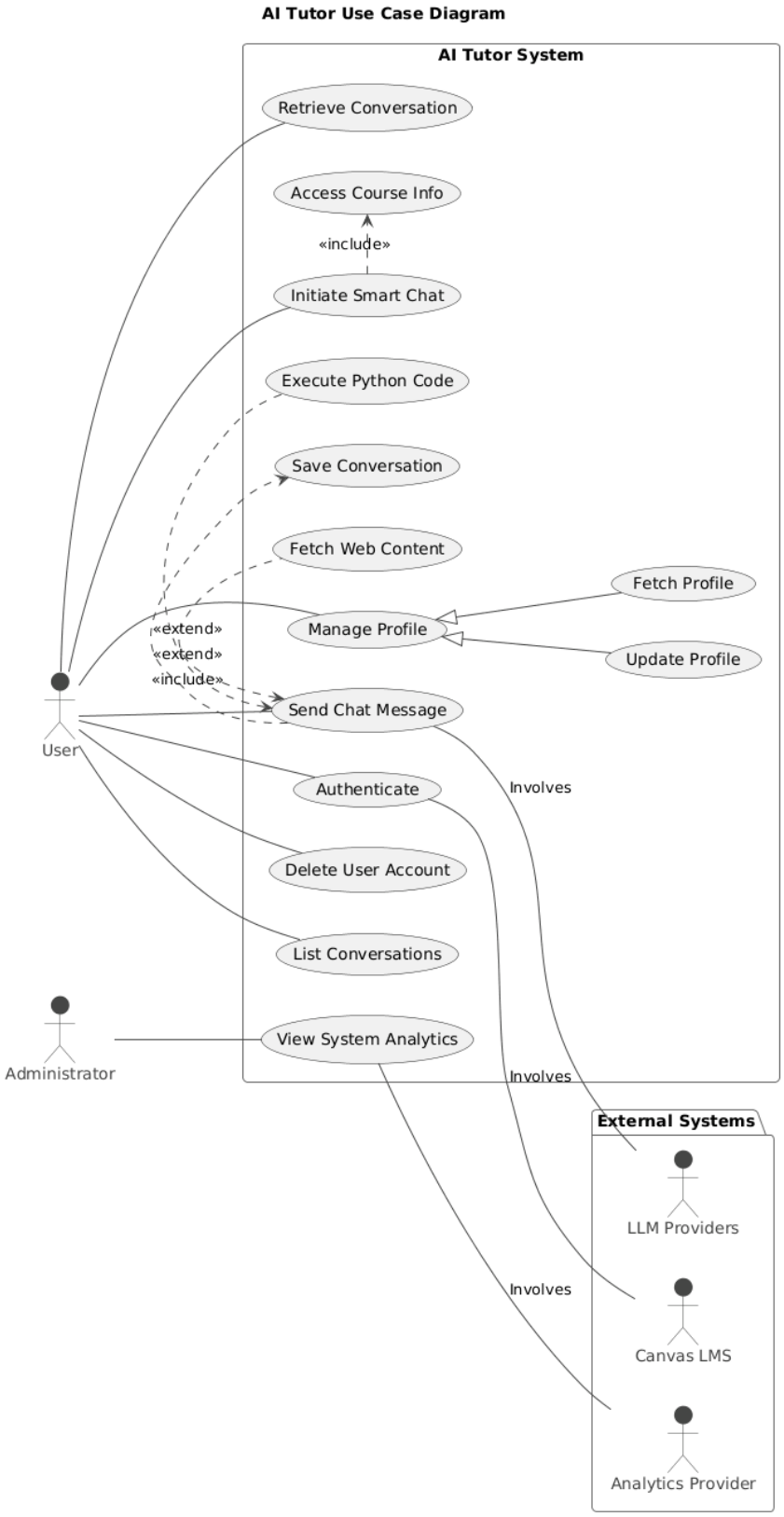
- **Update User Profile:** The user modifies and saves their profile information (specifically their bio) to personalize the AI's tone and responses.
- **List Conversations:** The user's client application requests and displays a list of all their past chat sessions, allowing them to select a previous conversation for review.
- **Retrieve Conversation:** After a user selects a conversation from the list, the client application retrieves the full message history for that specific chat session from the server.
- **Initiate Smart Chat:** The user starts a new conversation in "smart chat" mode. This is a distinct action that tells the backend to inject context from the user's Canvas data into the AI's prompt.
- **Send Chat Message:** The user sends a query to the AI Tutor. This is the most common action. This use case can be extended with specialized tools if the query requires them.
- **Access Course Info:** When a "Smart Chat" is active, the system automatically fetches the user's course information, assignments, and other relevant data from the Canvas API to provide context for the AI's response. This is a system action triggered by the user's



ERD Diagram

chat message.

- **Fetch Web Content:** As an extension of "Send Chat Message," if the user's query contains a URL, the AI can use this tool to access the content of that webpage to inform its answer.
- **Execute Python Code:** As an extension of "Send Chat Message," the AI can use this tool to write and run Python code in a secure, sandboxed environment to perform calculations, run algorithms, or demonstrate programming concepts.
- **View System Analytics:** The Administrator accesses a separate analytics dashboard (Plausible) to view aggregated, anonymized usage data and system performance metrics.



Use Case Diagram

8 Testing

- Testing Plan

This testing plan outlines the strategy and procedures to ensure the AI Tutor application is reliable, functional, and secure. The plan covers multiple levels of testing, from individual components to the system as a whole.

8.1 1. Scope

This plan applies to all components of the AI Tutor project, including:

- **Backend:** The Python FastAPI application, including all API endpoints, business logic, and interactions with the database and external services (Canvas, LLMs).
- **Frontend:** The Svelte-based web interface, including UI components, state management, and user interactions.
- **Infrastructure:** The Docker-based containerization and deployment configuration.

8.2 2. Testing Strategy

A multi-layered testing approach will be used to validate the application at different levels.

8.2.1 a. Unit Testing

- **Objective:** To verify that individual functions and components work correctly in isolation.
- **Backend (Python):** Each function in the FastAPI application will have corresponding unit tests. Mocking will be used to isolate components from external dependencies like the database or the Canvas API.
  - **Tools:** `pytest`
- **Frontend (Svelte):** Individual Svelte components and utility functions will be tested to ensure they render and behave as expected given specific props and user events.
  - **Tools:** `Vitest`, `Svelte Testing Library`

8.2.2 b. Integration Testing

- **Objective:** To verify that different parts of the application work together as intended.
- **Backend:** Tests will focus on the interaction between API endpoints and the MongoDB database. For example, ensuring that creating a user via an endpoint correctly stores the user in the database.
  - **Tools:** `pytest`, `TestClient` for FastAPI
- **Frontend-Backend:** Tests will verify that the frontend can successfully make API calls to the backend and handle the responses correctly. This will involve running both the frontend and backend servers in a test environment.
  - **Tools:** `Playwright` or `Cypress`

8.2.3 c. End-to-End (E2E) Testing

- **Objective:** To simulate real user scenarios from start to finish, ensuring the entire application flow is working correctly.
- **Scenarios:** Key user journeys will be tested, such as:
  1. User logs in via Canvas, asks a question in Smart Chat, and receives a context-aware answer.
  2. User updates their bio, starts a new chat, and verifies the AI's tone has changed.
  3. User asks a question that requires executing Python code.
- **Tools:** `Playwright`

8.2.4 d. Security Testing

- **Objective:** To identify and mitigate potential security vulnerabilities.
- **Methods:**
  - Regular dependency scanning to find known vulnerabilities in third-party packages.
  - Manual penetration testing of key endpoints, focusing on authentication (JWT handling) and input validation to prevent injection attacks.
  - Ensuring all sensitive data is handled securely and API keys are not exposed.

8.3 3. Roles and Responsibilities

- **All Team Members:** Responsible for writing unit tests for the code they develop.
- **Sprint Lead:** Responsible for overseeing the testing process, running integration and E2E tests before a release, and managing bug reports.

8.4 4. Metrics and Quality Assurance

- **Code Coverage:** Aim for a minimum of 80% unit test coverage for the backend Python code, measured using `pytest-cov`.
- **Bug Tracking:** All bugs will be logged as GitHub Issues. Each bug report will include a description, steps to reproduce, priority, and severity.
- **Pass/Fail Criteria:** For a sprint to be considered complete, all unit and integration tests must pass, and there should be no outstanding critical or high-priority bugs.
- Test Cases

This section provides a sample of test cases derived from the testing plan. The `Test Date` is set to the creation date and `Pass/Fail` status is pending execution.

8.4.1 Unit Test Cases (Backend)

Table 1: Unit Test Cases (Backend)

Test Case #	Description	Test Date	Inputs	Expected Outputs	Pass/Fail
UT-BE-01	**User Model Validation:** Tests that the `User` Pydantic model in `models.py` successfully validates a correct user object.	2025-10-05	`{"sub": "123", "name": "Test User"}`	The `User` model is instantiated without errors.	Pending
UT-BE-02	**Invalid User Model:** Tests that the `User` model raises a `ValidationError` if required fields are missing.	2025-10-05	`{"sub": "123"}` (missing `name`)	A `pydantic.ValidationError` is raised.	Pending
UT-BE-03	**AI Response Formatting:** Tests a utility function in `ai.py` that formats a raw LLM response into the correct JSON structure for the frontend.	2025-10-05	Raw text string from a mocked LLM.	A valid JSON object with `type` and `content` fields is returned.	Pending

8.4.2 Unit Test Cases (Frontend)

Table 2: Unit Test Cases (Frontend)

Test Case #	Description	Test Date	Inputs	Expected Outputs	Pass/Fail
UT-FE-01	**Message Component Rendering:** Verifies that the `Message.svelte` component correctly displays the message text and author.	2025-10-05	`props = { author: "AI", text: "Hello!" }`	The component renders a `div` containing the text "Hello!" and indicates it is from the "AI".	Pending
UT-FE-02	**Send Button Event:** Verifies that clicking the "Send" button in the `ChatInput.svelte` component dispatches a `send` event.	2025-10-05	User clicks the send button.	A `send` event is dispatched with the input field's text content as the payload.	Pending

8.4.3 Integration Test Cases

Table 3: Integration Test Cases

Test Case #	Description	Test Date	Inputs	Expected Outputs	Pass/Fail
IT-BE-01	**Get User Profile:** Verifies the `/api/users/me` endpoint.	2025-10-05	`GET` request to `/api/users/me` with a valid JWT for a user stored in the test database.	A `200 OK` response is returned with a JSON body containing the correct user's profile data.	Pending
IT-BE-02	**Create Conversation:** Verifies the `/api/conversations/` endpoint.	2025-10-05	`POST` request to `/api/conversations/` with a valid JWT and a title.	A `201 Created` response is returned, and a new conversation document is created in the database for that user.	Pending

8.4.4 End-to-End Test Cases

Table 4: End-to-End Test Cases

Test Case #	Description	Test Date	Inputs	Expected Outputs	Pass/Fail
E2E-01	**Full Login and Chat Flow:** Simulates a user logging in, sending a message, and receiving a response.	2025-10-05	1. Navigate to home page. 2. Click "Login". 3. Complete mock Canvas login. 4. Type "Hello" into chat. 5. Click "Send".	1. User is redirected to Canvas. 2. User is redirected back to the app. 3. The message "Hello" appears in the chat window. 4. An AI response is streamed into the chat window.	Pending

8.4.5 Security Test Cases

Table 5: Security Test Cases

Test Case #	Description	Test Date	Inputs	Expected Outputs	Pass/Fail
ST-01	**Unauthorized API Access:** Attempt to access an authenticated endpoint without proper credentials.	2025-10-05	`GET` request to `/api/users/me` with no `Authorization` header.	The API returns a `401 Unauthorized` or `403 Forbidden` status code.	Pending
ST-02	**XSS in Chat Input:** Attempt to inject a script into the chat.	2025-10-05	User types `<script>alert('XSS')</script>` into the chat input and sends.	The message is displayed as plain text in the chat window, and no alert dialog appears. The script is sanitized.	Pending

- Bug Report

This section will be populated with bug reports as they are identified during the testing process. At this time, formal testing has not yet begun.

Table 6: Bug Report

Date	Tester	Test Case Number	Priority	Severity Level	Assigned Developer	Fix Date

- Assessment Report

This section will contain an analysis of the testing results, including metrics trends and an overall assessment of application quality. This report will be generated after the initial testing cycles are complete.

8.5 5. Preventive QA

- **Static Code Analysis:** Automated tools will be used to analyze the source code without executing it. This helps to identify potential vulnerabilities, bugs, and code smells early in the development process.
  - **Tools:** `ruff`, `bandit`
- **Code Reviews:** All new code will be reviewed by at least one other team member before it is merged into the main branch. This helps to ensure code quality, consistency, and knowledge sharing among the team.
- **Coding Standards:** The team will adhere to a consistent set of coding standards (e.g., PEP 8 for Python) to ensure that the codebase is readable and maintainable.

8.6 6. Design and Maintenance Metrics

8.6.1 a. Design Metrics

- **Cyclomatic Complexity:** This metric measures the complexity of a function's decision-making logic. A high cyclomatic complexity can indicate that a function is difficult to test and maintain. The target is to keep the cyclomatic complexity of all functions below 10.
- **Coupling:** This metric measures the degree of interdependence between modules. Low coupling is desirable, as it makes it easier to modify one module without affecting others.
- **Cohesion:** This metric measures how closely related the responsibilities of a single module are. High cohesion is desirable, as it indicates that a module has a well-defined purpose.

8.6.2 b. Maintenance Metrics

- **Mean Time To Repair (MTTR):** This metric measures the average time it takes to fix a bug, from the time it is reported to the time a fix is deployed. The target MTTR for critical bugs is less than 24 hours.
- **Mean Time Between Failures (MTBF):** This metric measures the average time between system failures. A high MTBF indicates a reliable system. The target MTBF is greater than 1000 hours.
- **Code Churn:** This metric measures the number of times a file is modified. A high code churn can indicate that a file is a "hotspot" in the codebase and may be a candidate for refactoring.

9 Feedback Analysis and Action Plan

9.1 Analysis of Feedback

The user feedback is generally positive, with users appreciating the AI Tutor's integration with Canvas for accessing assignments and grades. The ability to answer specific questions and provide summaries is also highly valued. However, several key areas for improvement have been identified:

Key Issues & Bugs:

- **Accuracy:** There are instances of the AI providing incorrect information, particularly regarding class schedules and assignments with distant due dates.
- **Stability:** Several users reported crashes and error messages, especially when asking certain types of questions.
- **Complex Queries:** The AI struggles with complex or multi-part questions and does not handle "cafeteria-style" grading structures well.
- **Scope:** The AI's knowledge is limited to the information it can access, leading to incorrect assumptions (e.g., all assignments are required).

Feature Requests & Suggestions:

- **Syllabus Integration:** Direct access to the course syllabus would significantly improve accuracy, especially for grading policies and assignment requirements.
- **Expanded Scope:** Users want the AI to provide information on school events, announcements, and course locations (including a map).
- **Improved Task Management:** Suggestions include adding timeframes to assignments and providing reminders for due dates.
- **Onboarding:** Some users were confused about how to use the tool or that it required a browser extension.

9.2 Plan to Address Feedback

Based on the analysis, the following actions are proposed:

1. **Improve Accuracy and Stability:**
    - **Bug Fixing:** Prioritize fixing the identified bugs, including the causes of crashes and incorrect information.
    - **Enhanced Canvas Integration:** Improve the data retrieval from Canvas to get more accurate and complete information, including assignments with distant due dates.
    - **Contextual Understanding:** Improve the AI's ability to understand the context of questions, especially for complex and multi-part queries.
2. **Implement High-Impact Features:**
    - **Syllabus Integration:** Develop a feature to allow the AI to access and understand the course syllabus. This will be the highest priority new feature.
    - **Improved Onboarding:** Create a better onboarding experience for new users, explaining how to use the tool and its limitations.

- **Task Management Features:** Investigate the feasibility of adding timeframes to assignments and a reminder system.

3. **Expand the AI's Knowledge Base:**
   - **School-wide Information:** Explore the possibility of integrating with other school systems to provide information on events and announcements.
   - **Location Services:** Investigate adding a map feature for course locations.
4. **Continuous Improvement:**
   - **Feedback Mechanism:** Implement a more direct way for users to provide feedback within the application.
   - **Monitoring:** Continue to monitor user feedback and system performance to identify and address issues proactively.

9.3 User Feedback

The AI tutor was very helpful. I was surprised at how well it was able to access the information in Canvas. I was able to ask it some very specific questions about grades, and it had the answers. I also had a question about accessing my textbooks through Wolverine Access and it was able to answer than and help me find what I was looking for. I likely wouldn't have been able to find it on my own very easily, and I found it useful to have the AI Tutor to help me.

After I downloaded the Ai, the first question I asked was, "What are my missing assignments?" It gave me a list of upcoming assignments due this week. Basically I asked it questions about assignments, due dates and which are most urgent. The feedback was on point. I will continue to use the Ai to check on my assignments and grades as well. I feel it lacks in the accuracy of providing overall school events, announcements and personalization of course times and location; I 'm feel a map for location to each course would really help, especially in the first couple weeks of class.

The AI tutor was interesting. I didn't appreciate that I had to download an extension in order to use it, but I digress. I enjoyed the AI tutor. I asked it to help me understand technology that goes with human genome editing. We started off with what it is, how the editing works, key players, editing techniques, an overview, ethics, and then it asked me to ask more questions so we could refine my understanding and move deeper into our understanding of this topic. I will probably use this tutor for math, because that is what I struggle with the most.

I asked it to create a homework schedule where I take 3 hours per class per week. Which is honestly on par with what I do typically. The feature I appreciate most is it seems to be oriented to serving UVU students which is nice, but will become a pitfall if a BYU student tried to use this program. I won't mention improvements just yet since I expect to experience some things as I use this system I rather not give feedback just yet. I'm not sure how I will use this program.

After downloading the AI tutor I am unsure of where to implement this, I noticed that the tutor is still in development. Hopefully I will be able to use this in canvas to assess what my standings are in my courses.

I asked the AI tutor about the Delphinium program since I hadn't used it before. I first asked if points show up automatically after completing assignments or if you have to wait for grading, and it answered my question. However, I found the

AI less helpful in a cafeteria-style format. When I asked if TechTip assignments were required, it said all assignments were required, which could be improved with syllabus access. I plan to keep using the AI tutor for questions, as it often provides answers before I need to ask my teacher.

I found the AI tutor helpful in explaining assignments, sometimes it helps just to see it reworded. This semester I have two cafeteria style classes, including this one. I've been told multiple times that canvas doesn't understand cafeteria style, so I was curious if the AI understood it. When I asked it to explain the grading system for those classes, it told me that there were categories weighted differently that would contribute to a final grade. I tried being more specific, and asked it to explain the system based off the syllabus, but it gave me the same answer. I do think I will use it to explain confusing assignments, but definitely double check everything it says.

AI Tutor is a lot cooler than I had expected going into this and am glad I did. The connectivity it has with all my school info is extremally helpful for wanting to know when my next test might be. I put it through some stress tests to see if it would falter, but it gave me what I was looking for every time. I plan to actually use this in the future for all school related needs.

My experience with the AI tutor was mixed. With my first question I asked if I had any upcoming assignments in my aviation class. I did not mention the title of this class as this class was called AVSC 1010 but it understood that this class was the class I was talking about. However it could not pull up an assignment that was due later this month in that class. My second question was probably the most successful question. I asked what assignments were due this week and it pulled up all the assignments that were due this week from all five of my courses. They were all correct and this made me very happy. It does not take multiple questions at once, it struggled with midterms but defined other things extremely well. In the future, I will probably just ask it simple questions regarding the syllabus or due dates which it accomplishes very well. Overall the AI tutor has potential.

I asked the Ai assistant to tell me the things I need to get done for the week. I asked the Ai than to assist me on prioritizing my time for the assignments this week. I found out that it gives you links quickly to the assignments due. I would recommend adding timeframes for assigments to better organize my time and attention. I plan to use it to help me organize and prioritize my assignments in the week and month throughout the semester.

I got a message that said the AI Tutor is currently in development, I do not know if that is why I could not really use it.

I thought that the AI tutor was really cool, I first asked it when my next assignment was due and it gave me a list in each course and even the next due date! I think that it is super helpful that it connects to my canvas courses. I think that the AI tutor will be really helpful in understanding complicated subjects without needing to head to google.

With AI tutor I asked it many questions about questions I was being asked to complete assignments and it gave me ideas on how I could answer the question. I thought it was cool how it synced up to my canvas and could specifically ask questions about the courses I am taking. I think I can use Tutor AI to ask questions and help me brainstorm for my classes.

At first I thought that having another AI but this one is just for Canvas felt redundant, since I can use one to ask questions that I may have about the assignment, but the moment I asked it about the upcoming assignments, not only did it show all

the assignment and their due dates, it also gave a brief sentence on what to expect on it, which is super nice considering that it takes a lot of time reading, and then understanding. This AI does help a lot in the realm of focusing the tasks, and since it's tailored towards my assignments, I will find myself using to ask specific questions instead of asking the professor and waiting for a response, although, it's important to remember that if I have a question to always ask the professor first.

I asked the AI Tutor about an assignment that's due on wednesday, and it brought up the assignment requirements, details, dates, and basically anything important. I like how you can select specific classes, as well as how it has access to the course so it can give you specific information about assignments, due dates, requirements, etc. I also like how it gives you links for the specific information it is giving you, that could help very much if you can't find a specific detail, date, or really anything.. It could be very useful. I will probably use it to summarize assignments and simplify my workload.

The Artificial Intelligence Tutor tool helps me stay on top of my assignments. It can help me create a schedule that will allow me to stay on top of my assignments, without falling behind. Often as a student, it can be difficult to manage large workloads for multiple different classes. With the Artificial Intelligence Tutor, students can finally have aid in completing their assignments and creating ways to get ahead of them. As a Utah Valley University student, I will use the Artificial Intelligence Tutor to aid me with any questions regarding how to prepare best to complete my assignments and complete them before the assigned due date.

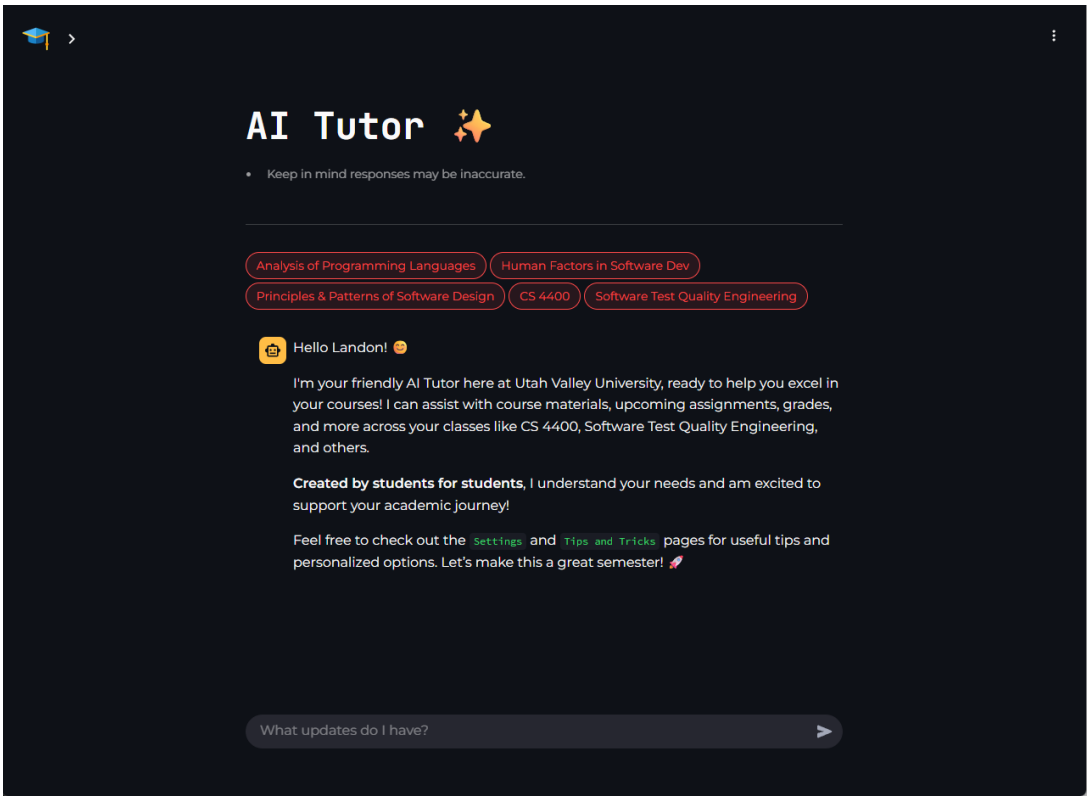
I found this AI Tutor to not be the best. It helped me with very simple tasks such as "When is my next assignment due?", however when I asked another question that I thought was as simple as the one I just asked, "When do I have these classes?" half of the time it crashed on me showing me an error sign, the other 50% of the time it mixed up my classes, telling me that the classes I have on Monday and Wednesday, according to the AI, I have it on Tuesday and Thursday which is incorrect. I am incredibly hesitant about using this AI feature this semester as I encountered many bugs in most of my experiences with it. Once I see this application more ironed out, I may come to use it.

I used AI Tutor to give me some background information about the play Antigone, by Sophocles. I asked about this because I have a discussion next week in my philosophy class so I wanted to get some brief background knowledge before I start reading the play. It helped go over the play and even added key themes and cultural context about the play. In all it helped me get a better understanding about what I am about to get into. I may use AI Tutor to help me with small things or to ask it questions to help guide me in the right direction of the topic.

I asked the AI tutor to help me create a schedule to keep up with my History class and its readings. Not that I really need it but I just wanted to see what it could do. It did create a schedule but only for this week and for the two classes I asked it not to do, haha. That's all that I asked it to do because I didn't have a ton of time to explore it. I would use it to send me reminders or something for assignments a day before it is due. I do pretty good about remembering to turn stuff in but occasionally I forget smaller assignments.

10 Readme

This is the repository for the Generative AI Tutor pioneered at UVU.



AI Tutor Homepage

10.1 Introduction

The AI Tutor project has been under active development for quite some time, started back in early 2024. In discussions with the excellent faculty in the Tech Management Department at Utah Valley University, we hypothesized that using the new and exciting technology of large language models, we could provide excellent, *personalized* tutoring to students *whenever they needed it.*

So, we set out to provide a novel way to accomplish this goal. The original AI Tutor was indeed a success, quickly being adopted into a several courses at Utah Valley University, and gaining attention among multiple departments and more than a handful of students.

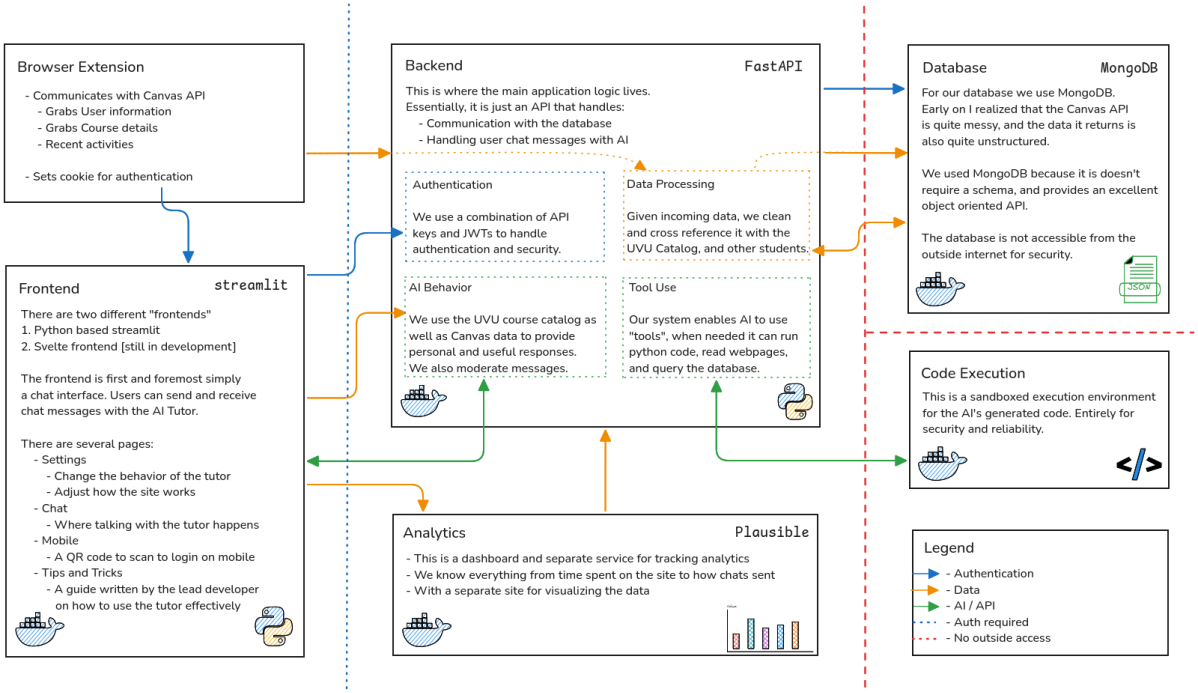
This repository is where that code lives.

10.2 Design

10.3 Development

In order to run the project in development mode:

1. Ensure Docker and Docker Compose are installed.
2. In the project root directory, run the command: `docker compose -f develop.yaml build`
3. Then, still in the project root, run: `docker compose -f develop.yaml up --watch`



An overview of the system design of the AI Tutor

4. Everything should be up and running :)

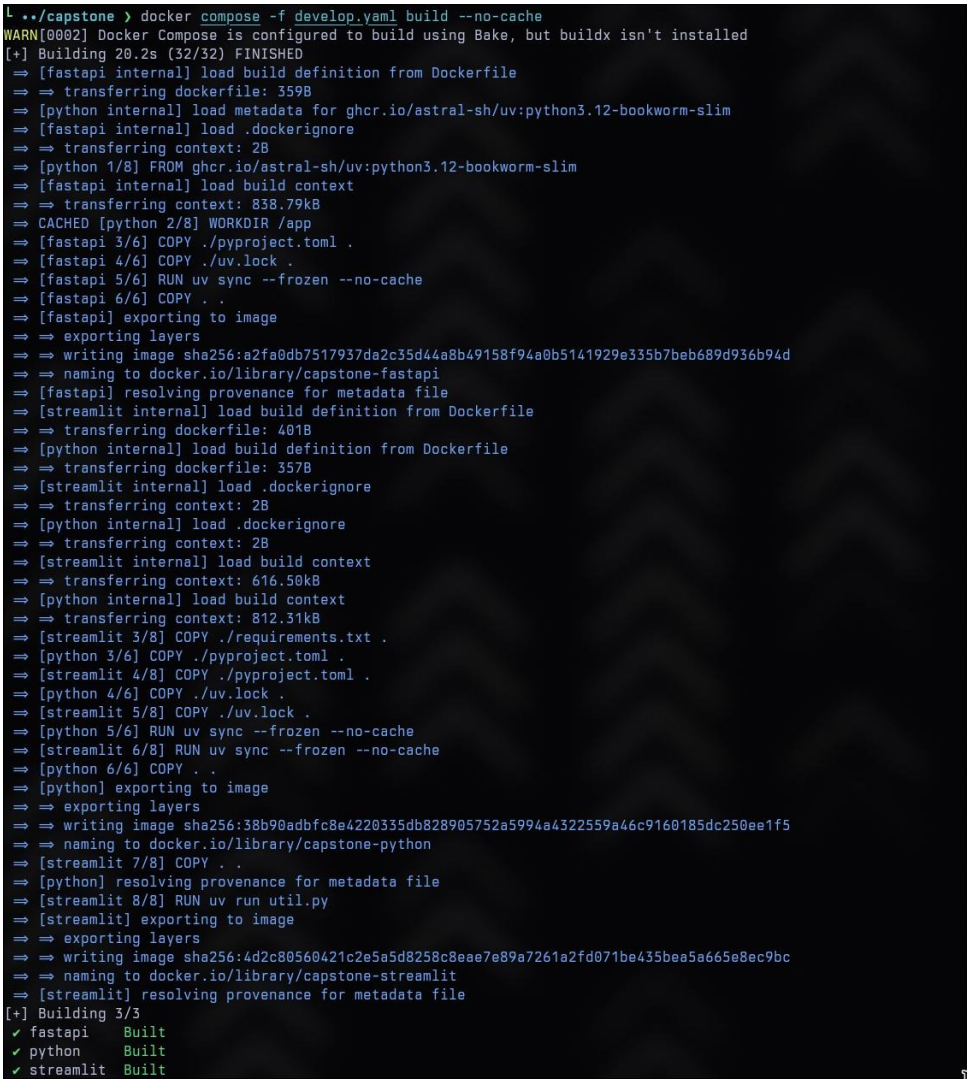
10.4 Deployment

- Our deployments are hosted on a Hetzner virtual private server.

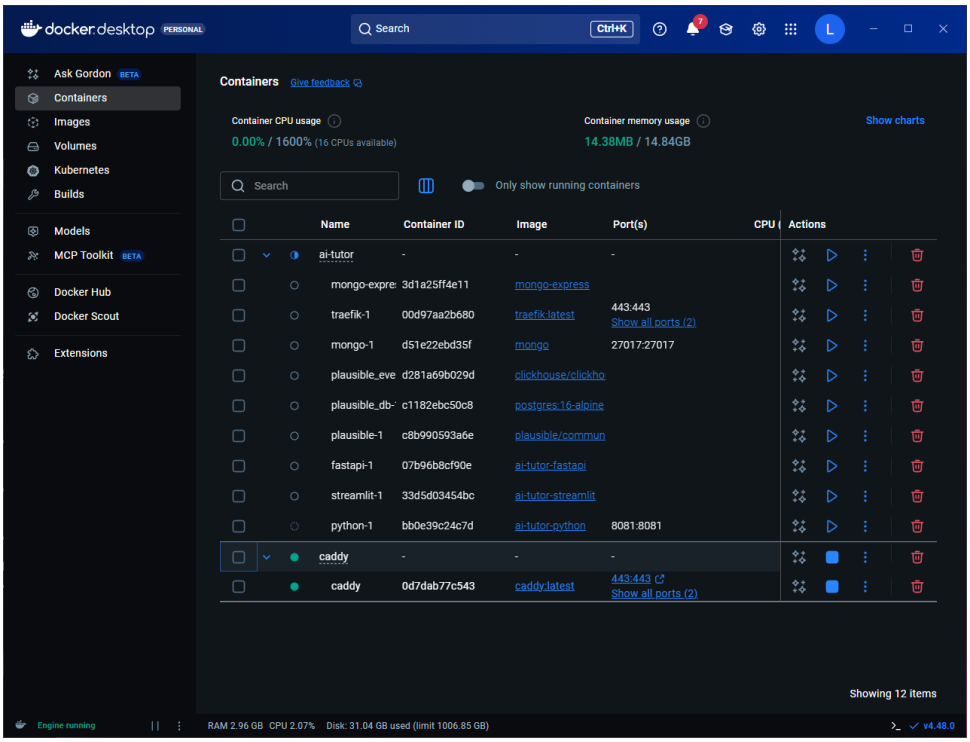
1. We use just to bundle everything needed to deploy into one command `just deploy`
2. This essentially just uses `rsync` and `ssh` to send the files up, build the docker containers, and run them.

10.5 Acknowledgments

- **Dr. Ahmed Alsharif**: For making everything possible and supporting this project so wholeheartedly.
- **Dr. Armen Ilikchyan**: For paving the way, and providing invaluable guidance.



Docker Building



Docker Running