*1.Florida International University*

*Knight Foundation School of Computing and Information Sciences*

Software Engineering Focus

Final Deliverable

Project Title:

STARS Tutor Chatbot

**Team Members:** Leandro Alvarez, Brian Otero, Kristian Vazquez, Miguel Garcia, Jesus Valdes

**Product Owner(s)**: Dr.Wells

**Mentor(s)**: Andres Lopez

**Instructor**: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) *2016 Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

***Abstract***

*This document presents the information necessary to gain a good understanding of the Stars Tutoring Chatbot project, explaining its goals, development process and design. The document begins with an explanation of the current system and what ways we are going to improve the product. It will then talk about our implemented and pending user stories. It will also include a project plan where it covers the hardware and software requirements to run our application and a detailed sprint breakdown. The system design is shown by our architectural patterns, system and subsystem decomposition, deployment diagrams, and design patterns. The document also explains how we validate our system. It concludes with a glossary, appendices which includes UML Diagrams, user interface designs, sprint review reports, and user manuals, along with references to give our users a complete guide of our product*

**Table of Contents**

**Introduction** …………………………………………………………………………………………….....5

Current System ……….…………………………………………………………………………………………………..5

Purpose of New System …………………………………………………..……………………………. 5

**User Stories**

Implemented User Stories ………………………………………….…………………………………..6

Pending User Stories …………………………………………………………………...………………………………..….. 8

**Project Plan**

Hardware and Software Resources ……………………………………………………………….… 8

Sprints Plan ……………………………………………………………..………………………………………………………. 9

*Sprint 1*  …………………………………………………………………………………………………………………………... 9

*Sprint 2*  …………………………………………………………………………………………………………………………... 10

*Sprint 3*  …………………………………………………………………………………………………………………………... 10

*Sprint 4*  …………………………………………………………………………………………………………………………... 10

*Sprint 5*  …………………………………………………………………………………………………………………………... 12

*Sprint 6*  …………………………………………………………………………………………………………………………... 13

*Sprint 7*  …………………………………………………………………………………………………………………………... 14

**System Design**

Architectural Patterns ………………………………………………………………………………………………….. 15

System and Subsystem Decomposition …………………………………………………………………………….… 16

Deployment Diagram …………………………………………………………………………………………………….…... 17

Design Patterns ………………………………………………………………………………………………………….….... 17

**System Validation**  …………………………………………………………………………………………………………………….19

**Glossary**  ………………………………………………………………………………………………………………………………….19

**Appendix**  ………………………………………………………………………………………………………………………………………………..21

Appendix A - UML Diagrams ………………………………………………………………………………………………………………….. 21

*Static UML Diagrams*  ………………………………………………………………………………………………………………………21

*Dynamic UML Diagrams*  ………………………………………………………………………………………………………………….22

Appendix B - User Interface Design …………………………………………………………………………….…................................ 22

Appendix C - Sprint Review Reports …………………………………………………………………………...……,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, 35

Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents …………………………………………………………………….………………………………. 42

**References** ……………………………………………………………………………………………………………………...………....................47

# 

# Introduction

STARS is an organization that was started by Dr. Patricia McDermott-Wells. She is also a professor at FIU and has over 25 years of experience as a software engineer. STARs Tutoring recognizes students who are struggling with their computer science classes and provides them with resources to help them succeed. We are currently assisting Dr. Wells in developing multiple chatbots that students can use to help them with their tutoring needs, as well as an admin page to simplify the process of getting students accepted to the program.

## 

## Current System

The current system that we inherited was very sparse visually and in features, tutorbot and codebot. Tutorbot being a general chatbot for all classes and codebot being a chatbot that helps you learn code or learn to prompt engineering. All of these chatbot chats disappear after you log off and back on, and the sidebar UI appears empty. This is only for the student side. The only documents we received were a project proposal from the technical lead for this project. This led us to question the existence of the codebot feature developed by the previous team, as it appeared to be only for prompt engineering and did not assist students in their classes. But as the team met up with the technical lead, we went right to work on the new system.

## Purpose of New System

The new system we created maintains the two features that was developed from the previous teams and expanded the use cases by adding chat history along with the ability to selected, adding and deleting chats. As well as adding class specific chatbots to get more specific details per class. For the tutors, we added a feature to edit the chat response from the chatbot if they did not like the way the bot answered the question. And for the admins, we made a new, easier way to look at a student’s available classes to tutor for within the app so this can be a one website experience.

# User Stories

The following section provides the detailed user stories that were implemented in this iteration of the Stars Tutor Chatbot project. These user stories served as the basis for the implementation of the project’s features. This section also shows the user stories that are to be considered for future development.

## Implemented User Stories

1. Setup the GitHub Repository and Environment
   1. Get all team members to connect to the GitHub repository from our technical lead and our tech stack to work
2. Add Multi-Chat Memory
   1. Create a user object that can store chats into the database
3. Database Diagram
   1. Design the developer direction when creating the user object
4. Add New Chat
   1. Create a function where the user can add a new chat to make a new conversation with the chatbot
5. Selecting a Chat
   1. Create a select box to let the user choose what chat they want to talk to, sorted by the most recent
6. Delete Chat
   1. Create a function where the user can remove a chat from the selection box
7. AUTH Code
   1. Give access code to tutors to access fine tuning feature
8. Create and fine tune a class-specific (net-centric) chatbot
   1. Use the tutorbot as the basic embedding for this new chatbot.
9. Create Completions for the Net Cetric bot
   1. Use the few shot examples used in the Net Centric bot to create a more refined chatbot
10. Rework the UI for the website
    1. Rearrange the elements to comply with HCI principles
11. Add caches to the chats
    1. Uses caches to store chat history into the database
12. Add UI for Tutors
    1. Create UI for the tutors to fine tune the chats that the student has created.
13. Create an Operating Systems Chatbot
    1. Use the tutorbot as the basic embedding and knowledge of the class for this new chatbot
14. Create a Logic for Computer Science Chatbot
    1. Use the tutorbot as the basic embedding and knowledge of the class for this new chatbot
15. Create an Algorithm Techniques Chatbot
    1. Use the tutorbot as the basic embedding and knowledge of the class for this new chatbot
16. Create a Computer Architecture Chatbot
    1. Use the tutorbot as the basic embedding and knowledge of the class for this new chatbot
17. Improve Loading times
    1. Improve the loading times of the web app by reworking the code to not have no st.reruns
18. Student Enrollment Feature
    1. Have a way to improve the way the admins to approve new students instead doing it manually within the app
19. Hashing Passwords
    1. Hash the passwords so that the tutors or admins that have access to the students can’t see it
20. Create Remote Database
    1. Use a local server to connect to establish that this can run on a server
21. Create GUI for the Admin user
    1. Create UI for the admins to add and remove students
22. Allow to perform CRUD operations for Admin user
    1. Create tokens for tutors, reads user data from the database, approve/reject users, and remove chats from the website to an archive collection
23. Improve Admin GUI page
    1. Update the page where admins can access the list of available courses with the ability of adding/ removing classes to the list and archive chats
24. DB Optimizations guide
    1. Migrate the Local DB to the Remote DB
25. Deployment
    1. Launch our project to an official fiu.edu domain
26. Documentation
    1. Compile all the information that we used into the posters, PowerPoints and the project documentation for the showcase
27. Intro Video
    1. Create the Introduction video to our project by using the powerpoints slides we made
28. Demo Video
    1. Show off all the features we develop in this video
29. Install/Maintenance Video
    1. Show the users how to install all the software and json files we provide in the Future Dev Doc, as well as how to add new models to the web app
30. Shortcoming Video
    1. Create a video describing all the features we did not have time for
31. Future Dev Doc
    1. Create a doc to give to the PO for the next generation of developers explaining how everything works

## Pending User Stories

1. SSO
   1. Use FIU SSO to make the login more professional

# Project Plan

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

The user stories were primarily divided into groups, each of which created their own group branch and began researching for their specific user story and developing their features. Once one group has completed the feature, they create a pull request to see if there are any merge conflicts with the other group's work, ensuring that the rest of the team can access it and build on top of their own code, allowing them to quickly move on to the next feature.

## Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

- Operating System: Windows

-Streamlit

- GitHub

-Tailscale

- Visual Studio Code

-Python

-OpenAI API

-LangChain

-MongoDB Compass

-Groq

-Brevo

-Amazon Elastic Compute Cloud

## Sprints Plan

### Sprint 1

Total Hours: 50

1. Setup the GitHub Repository and Environment

-Assigned to: Leandro Alvarez, Brian Otero, Kristian Vazquez, Miguel Garcia, Jesus Valdes

-Description: Get all team members to connect to the GitHub repository from our technical lead and our tech stack to work

-Acceptance Criteria: Successfully got all the developers to connect with the repository

1. Add SSO to the Login Page

-Assigned to: Kristian Vazquez, Miguel Garcia

-Description: Develop a SSO login page that only accepts FIU students and staff.

-Acceptance Criteria: The page must direct the user to the FIU 2FA

1. Add Multi-Chat Memory

-Assigned to: Leandro Alvarez, Jesus Valdes, Brian Otero

-Description: Create a user object that can store chats into the database

-Acceptance Criteria: The user object should store chats to display it unto the web app

1. Database Diagram

-Assigned to: Kristian Vazquez, Jesus Valdes

-Description: Design the developer direction when creating the user object

-Acceptance Criteria: Successfully design a schema that allow the user to have chats in their object

### Sprint 2

Total Hours: 90

1. Add New Chat

-Assigned to: Jesus Valdes, Brian Otero

-Description: Create a function where the user can add a new chat to make a new conversation with the chatbot

-Acceptance Criteria: Chat should be saved in user object

1. Selecting a Chat

-Assigned to: Leandro Alvarez, Kristian Vazquez, Miguel Garcia

-Description: Create a select box to let the user choose what chat they want to talk to, sorted by the most recent

-Acceptance Criteria: Implement method to select chats between classes and bots

1. Delete Chat

-Assigned to: Jesus Valdes, Leandro Alvarez

-Description: Create a function where the user can remove a chat from the selection box

-Acceptance Criteria: Implement a way to delete it from the selection box but not from the user object

### Sprint 3

Total Hours: 150

1. AUTH Code

-Assigned to: Leandro Alvarez, Kristian Vazquez, Miguel Garcia

-Description: Give access code to tutors to access fine tuning feature

-Acceptance Criteria: The code must be only for tutors

1. Create and fine tune a class-specific chatbot

-Assigned to: Leandro Alvarez, Kristian Vazquez, Miguel Garcia, Jesus Valdes, Brian Otero

-Description: Use the tutor bot as the basic embedding for this new chatbot.

-Acceptance Criteria: Successfully create the class-based chatbot onto the web app

### Sprint 4

Total Hours: 150

1. Rework the UI for the website

-Assigned to: Jesus Valdes, Brian Otero

-Description: Rearrange the elements to comply with HCI principles

-Acceptance Criteria: Design should be more intuitive

1. Add caches to the chats

-Assigned to: Leandro Alvarez, Brian Otero

-Description: Uses caches to store chat history into the database

-Acceptance Criteria: Successfully create the caches to the database to load the chats faster

1. Create Completions for the Net Centric bot

-Assigned to: Leandro Alvarez

-Description: Create a function where the user can add a new chat to make a new conversation with the chatbot

-Acceptance Criteria: Chat should be saved in user object

1. Add UI for Tutors

-Assigned to: Leandro Alvarez, Kristian Vazquez, Miguel Garcia

-Description: Create UI for the tutors to fine tune the chats that the student has created.

-Acceptance Criteria: Chat should be saved in user object

1. Create an Operating Systems Chatbot

-Assigned to: Brian Otero

-Description: Use the tutorbot as the basic embedding and knowledge of the class for this new chatbot

-Acceptance Criteria: Successfully create the class-based chatbot onto the web app

1. Create a Computer Architecture Chatbot

-Assigned to: Miguel Garcia

-Description: Use the tutorbot as the basic embedding and knowledge of the class for this new chatbot

-Acceptance Criteria: Successfully create the class-based chatbot onto the web app

1. Create an Algorithm Techniques Chatbot

-Assigned to: Kristian Vazquez

-Description: Use the tutorbot as the basic embedding and knowledge of the class for this new chatbot

-Acceptance Criteria: Successfully create the class-based chatbot onto the web app

### Sprint 5

Total Hours: 170

1. Create a Logic for Computer Science Chatbot

-Assigned to: Jesus Valdes

-Description: Use the tutorbot as the basic embedding and knowledge of the class for this new chatbot

-Acceptance Criteria: Successfully create the class-based chatbot onto the web app

1. Improve Loading times

-Assigned to: Leandro Alvarez, Jesus Valdes, Brian Otero

-Description: Improve the loading times of the web app by reworking the code to not have no st.reruns

-Acceptance Criteria: Successfully reduce loading times throughout the webapp

1. Student Enrollment Feature

-Assigned to: Leandro Alvarez, Kristian Vazquez, Miguel Garcia

-Description: Have a way to improve the way the admins to approve new students instead doing it manually within the app

-Acceptance Criteria: Successfully add the feature

1. Hashing Passwords

-Assigned to: Kristian Vazquez, Miguel Garcia

-Description: Hash the passwords so that the tutors or admins that have access to the students can’t see it

-Acceptance Criteria: Successfully hide the password

1. Create Remote Database

-Assigned to: Kristian Vazquez

-Description: Use a local server to connect to establish that this can run on a server

-Acceptance Criteria: Successfully add all users to the database

1. Create GUI for the Admin user

-Assigned to: Leandro Alvarez, Kristian Vazquez, Miguel Garcia

-Description: Create UI for the admins to add and remove students

-Acceptance Criteria: Make a easy to understand page for admins

1. Allow to perform CRUD operations for Admin user

-Assigned to: Leandro Alvarez, Kristian Vazquez, Miguel Garcia

-Description: Create tokens for tutors, reads user data from the database, approve/reject users, and remove chats from the website to a archive collections

-Acceptance Criteria: Successfully allow the admin to do all those operations

### Sprint 6

Total Hours: 170

1. Improve Admin GUI page

-Assigned to: Leandro Alvarez, Kristian Vazquez, Miguel Garcia

-Description: Update the page where admins can access the list of available courses with the ability of adding/ removing classes to the list and archive chats

-Acceptance Criteria: Allow the admin to archive chats and add/delete new course

1. Documentation

-Assigned to: Leandro Alvarez, Jesus Valdes, Brian Otero

-Description: Compile all the information that we used into the posters, PowerPoints and the project documentation for the showcase

-Acceptance Criteria: Successfully reduce loading times throughout the webapp

1. Deployment

-Assigned to: Leandro Alvarez, Kristian Vazquez, Miguel Garcia

-Description: Launch our project to an official fiu.edu domain

-Acceptance Criteria: Successfully deploy our project

1. DB Optimizations guide

-Assigned to: Kristian Vazquez

-Description: Migrate the Local DB to the Remote DB

-Acceptance Criteria: Get the raspberry pi to run on our computer

### Sprint 7

Total Hours: 170

1. Intro Video

-Assigned to: Leandro Alvarez, Kristian Vazquez, Miguel Garcia, Jesus Valdes, Brian Otero

-Description: Create the Introduction video to our project by using the powerpoints slides we made

-Acceptance Criteria: Show off all the individual contributions we did

1. Demo Video

-Assigned to: Leandro Alvarez, Kristian Vazquez, Miguel Garcia, Jesus Valdes, Brian Otero

-Description: Show off all the features we develop in this video

-Acceptance Criteria: Successfully showcased on all the features we worked on

1. Install/Maintenance Video

-Assigned to: Brian Otero

-Description: Show the users how to install all the software and json files we provide in the Future Dev Doc, as well as how to add new models to the web app

-Acceptance Criteria: Show the step by step process on how to install and maintain the project

1. Shortcoming Video

-Assigned to: Leandro Alvarez

-Description: Create a video describing all the features we did not have time for

-Acceptance Criteria: Accurately describe the features we want the future developers to do

1. Future Dev Doc

-Assigned to: Leandro Alvarez

-Description: Create a doc to give to the PO for the next generation of developers explaining how everything works

-Acceptance Criteria: Successfully describe all the links we provide to the PO

# System Design

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

## Architectural Patterns

Our system is designed with a multi-tier modular architecture, with separate responsibilities for the UI, logic, and persistence layers. We also adopted a modular monolith approach for the chatbot components, which included reusable, encapsulated logic. We also use event-driven programming for features such as automated email notifications prompted by administrative actions. Each architectural pattern used is described in detail below, along with how it was implemented in our project.

1. **Multi-Tier Architecture**

The system is structured into three logical layers:

* Presentation Tier: Built using Streamlit, it handles user interactions for students, tutors, and admins. This includes login/signup, chat interface, and dashboards.
* Application Tier: Core backend logic written in Python, including chatbot control flow, fine-tuning management, and state/session handling.
* Data Tier: Uses MongoDB for persistent storage of user accounts, chat history, courses, and system state.

This layered approach simplifies development and testing by separating user interface, logic, and data concerns.

1. **Modular Monolith Architecture**

The system follows a modular structure that mimics service separation:

* Chatbots like tutorbot and codebot are implemented as modules with isolated responsibilities.
* Components such as authentication, fine-tuning, and course management are separated into self-contained files (e.g., chatbot.py, fine\_tuning.py, admin.py).

This modular design makes the codebase easier to maintain and scale, even within a single-deployment architecture.

1. **Event-Driven Programming (Code-Level)**

Some interactions follow an event-driven logic model:

* When an admin approves or rejects a user, it triggers an email event using the transactional email system (Brevo).
* Although the app does not use a message broker or event queue, this pattern allows for **decoupling between approval logic and notifications**.

This enhances responsiveness and sets the stage for future asynchronous features.

## System and Subsystem Decomposition

1. Login screen
   1. User has two option to choose, login or sign up
   2. For the sign-up form we ask the user for their name, panther id, valid FIU email and password
   3. If they are a tutor, they can click the “Tutor?” button and will ask for their key
   4. Tutors will have their own key provided to them from the admin
   5. If a student signs up, the sign-up form will ask for their pdf of their schedule
   6. We needed to import all the classes that are available to take for Stars tutoring so that the students only have computer science classes that STARS provided tutored for
   7. Once submitted, the information will be sent to the database instantly
   8. If the newly registered students try to log on, they will be directed to a notification saying that the student has not been approved yet.
   9. Only once the admin approves the student, the student can access the website.
2. Main Page
   1. You will have access to the two general bots- Tutor bot and Code bot to interactive with
   2. The users will also get their class specific chatbot from their schedule
   3. All these bots will generate a response based on the questions and inputs that the user provides
   4. To access the other bots, you click on the dropdown box called “Select Bot” and then select the bots that you want to interact with
   5. Users can create and delete chats from the selected bot
   6. Users can also select a certain chat to continue the conversations
   7. FOR TUTORS ONLY- They have an option to edit a chat response from any of the classes that the students did chats in. Asking a couple of questions on the accuracy of the response.
3. Admin Page
   1. Admin has the privilege to add or remove course that are available to tutor
   2. Once the semester is over, admins have the ability to archive the chats of students and tutors
   3. Once a student signs up, the admin can choose to accept or reject them based on the students’ classes
   4. Once enrolled, the admin still has an option to reject the student or revise the student if they picked up another class instead, and download all the chats that they currently have
   5. Tutors can be given access to the chatbots by the code that the admins provided within the app
   6. Once a tutor applies, admin has the option to approve or reject them
   7. When the tutor is enrolled, the admin has the option to remove them, revise their course selection, or to download their chats

## Deployment Diagram

A close-up of a card

AI-generated content may be incorrect.

Figure 1: Our Deployment Diagram for Tailscale and MongoDB

**Design Patterns**

To ensure a scalable and flexible system, we used several design patterns while developing our web app. These patterns improve the structural integrity of our codebase, making it easier to understand and debug. Each design pattern used is described in detail below, along with how it was implemented in our project.

1. **Data Access Object (DAO)**

* We utilized MongoDB to persist data including user information, chat logs, course details, and tutoring session information.
* To manage the interaction with the database, we implemented the Data Access Object (DAO) pattern.
* The DAO pattern abstracts the underlying database system, providing a clean interface for data access. This encapsulates the data persistence logic and separates it from the rest of the application.
* For instance, the system stores chat histories sorted by timestamp and employs caching mechanisms, which are managed by the DAO, to optimize performance.

1. **Model-View-Controller (MVC)**

* The architecture of our system strongly reflects the Model-View-Controller (MVC) pattern or a close variant:
  + **Model:** The backend logic, developed in Python and using Langchain, represents the data and business logic of the application.
  + **View:** The frontend is built with Streamlit, which handles the presentation layer and user interactions.
  + **Controller:** While not a traditional controller, MongoDB acts as the data store, and the DAO helps mediate between the Model and the View.
* This separation allows for a modular design where changes in the UI (View) do not directly impact the data handling (Model) or the data storage (MongoDB).

1. **Observer Pattern**

* The enrollment process incorporates the Observer pattern to manage notifications.
* When an administrator approves or rejects a student's enrollment, the system automatically sends an email to the student.
* In this case, the student is the "Observer," and the enrollment status is the "Subject." When the Subject's state changes, the Observer is notified.

1. **Strategy Pattern**

* Our chatbot system employs different chatbot models tailored for specific classes, such as "Tutorbot" and "Codebot".
* This design utilizes the Strategy pattern, where each chatbot model represents a different strategy for handling tutoring inquiries.
* The system can dynamically select the appropriate chatbot (strategy) based on the user's selected class.

1. **Singleton Pattern**
   * This design optimizes the web app's efficiency by only initializing the LangChain and prompt template once each session.
   * Used across chatbot components to ensure uniform chaining behavior.

# System Validation

We used several different tests throughout the process of improving the webapp to ensure that it would not receive an error message. This includes unit testing like deleting a non-existent chat, not displaying a new chat when no other chats are available, and the blue button not displaying the current chat on the screen, among other things. We would first test our features on a separate branch to iron out any initial bugs. Following that, we request that the other team members run the branch on their environment to see if we missed any bugs that were not detected on the original branch's environment. As a result, the code works in any environment. We'd then run the integration test to see if the newly added code was working on the main branch. If there are any problems with merging to main, another branch will be created to fix the bugs in main. Another testing method we used was security testing, which involved looking for any loopholes that users could exploit to gain access to the app and fixing them, as well as hashing the passwords so that administrators could not see what passwords everyone had.

**Glossary**

**Backend:** The server-side logic of a software application.

**Frontend:** The user interface of a software application.

**Brevo:** An email marketing and transactional email platform used for automated emails.

**Chatbot:** A computer program designed to simulate conversation with human users, especially over the Internet.

**Completion Data:** Data used to train AI models to generate appropriate responses.

**EC2:** Amazon Elastic Compute Cloud, a service that provides resizable compute capacity in the cloud.

**Few-Shot Examples :** Question-Pair examples used to improve the few-shot prompting from the model.

**Few-Shot Learning:** A machine learning approach where the model learns from a small number of examples.

**Fine Tuning:** The process of further training a pre-trained language model on a specific dataset to improve its performance on a particular task.

**Groq:** A hardware accelerator designed for machine learning workloads.

**LangChain:** A framework to assist in the development of applications powered by language models.

**LLM:** Large Language Model

**MongoDB:** A NoSQL database used to store data.

**OpenAI:** An AI research and deployment company.

**Raspberry Pi:** A series of small single-board computers used for various applications, including deploying web applications and databases.

**SHA-256:** A cryptographic hash function used for password hashing.

**Streamlit:** An open-source Python framework for building web applications for machine learning and data science.

**Tailscale:** A VPN service used for secure network connections.

**VPN**: Virtual Private Network.

## Appendix A - UML Diagrams

A diagram of a process

AI-generated content may be incorrect.

A diagram of a process

AI-generated content may be incorrect.

Figure 2: UML of Fine-Tuning Figure 3: UML of Student Enrollment

## A diagram of a process AI-generated content may be incorrect.

## Figure 5: Diagram of Database

## Figure 4: UML for Tutor Enrollment

## Appendix B - User Interface Design

A screenshot of a chatbot

Description automatically generated

Figure 6: Login page

A screenshot of a chatbot

Description automatically generated

Figure 7: Part 1 of the sign up form for students/tutors asking for their name, PID and email

A screenshot of a computer

Description automatically generated

Figure 8: Part 2 of the sign-up form where it asks the user to upload their schedule

A screenshot of a computer

Description automatically generated

Figure 9: Displaying what course are available for the student to take

A screenshot of a chatbot

Description automatically generated

Figure 10: This will be shown to the pending user if they have not been approved by the admins yet

A screenshot of a computer

Description automatically generated

Figure 11: Admin page, currently showing all the current classes that are available that students can take

A screenshot of a computer

Description automatically generated

Figure 12: Option to archive the chats

A screenshot of a computer

Description automatically generated

Figure 13: UI to add a new class to the list of tutorable classes

A screenshot of a computer

Description automatically generated

Figure 14: Inserting a course id to remove the certain class from the list

A screenshot of a computer

Description automatically generated

Figure 15: Page that shows all pending and enrolled students

A screenshot of a computer

Description automatically generated

Figure 16: Option to approve certain student

A screenshot of a computer

Description automatically generated

Figure 17: Option to reject certain student

A screenshot of a computer screen

Description automatically generated

Figure 18: Displaying options to remove, revise or download all the chats of certain student users

A screenshot of a computer

Description automatically generated

Figure 19: Page that sets up tutor code and shows all pending and enrolled tutors

A screenshot of a computer

Description automatically generated

Figure 20: Sign up page once you click “Tutor?” button

A screenshot of a chatbot

Description automatically generated

Figure 21: Sign up page for tutors only

A screenshot of a computer

Description automatically generated

Figure 22: Option to approve or reject pending tutors

A screenshot of a computer

Description automatically generated

Figure 23: Option to remove, revise, or download all chats from certain enrolled tutors

A screenshot of a chatbot

Description automatically generated

Figure 24: Main page, Class-specific chatbot

A screenshot of a chatbot

Description automatically generated

Figure 25: Page for Student revising their courses

A screenshot of a computer

Description automatically generated

Figure 26: Complete response when user scrolling down

A screenshot of a computer

Description automatically generated

Figure 27: Selection of Chatbots

A screenshot of a chatbot

Description automatically generated

Figure 28: Showing new chat and blue button feature

A screenshot of a chat box

Description automatically generated

Figure 29: A confirmation popup to make sure that you want to delete the current chat

A screenshot of a computer

Description automatically generated

Figure 30: TUTORS ONLY- View completions of the selected chat that the tutor did

A screenshot of a computer

Description automatically generated

Figure 31: TUTOR ONLY- Fine-tuning the chat via few-shot prompting

A screenshot of a computer

Description automatically generated  
Figure 32: Page for Tutors revising their courses

## Appendix C - Sprint Review Reports

**Sprint Review Meeting Minutes (Sprint 1)**

Project: STARS Tutoring Chatbot

Attendees: Leandro Alvarez, Brian Otero, Kristian Vazquez, Miguel Garcia, Jesus Valdes

Start time: 2:00

End time: 3:00

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

* User Story 1 Leandro Alvarez: 1 on 1 Meeting with technical lead (past team leader) and team meeting with PO.
* User Story 2 Jesus Valdes: Prepare Coding Environments and Repository
* User Story 3 Jesus Valdes: Study and Review the Project Stack
* User Story 4 Kristian Vazquez: Read and understand documentation for project tech stack (repository, project cycle, frameworks), along with studying concepts about the project proposal for boost in productivity.
* User Story 5 Miguel Garcia: Familiarize myself with the tech stack and current repository for chatbot, and start some early research on general topics such as few shot learning and semantic search.
* User Story 6 Brian Otero: Refamiliarize with Streamlit and MongoDB and start learning how to use OpenAI and LangChain. After that, start designing the database for chat history

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

All of our user stories were completed on time.

**Sprint Review Meeting Minutes (Sprint 2)**

Project: STARS Tutoring Chatbot

Attendees: Leandro Alvarez, Brian Otero, Kristian Vazquez, Miguel Garcia, Jesus Valdes

Start time: 2:00

End time: 3:00

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

* User Story 1 Leandro Alvarez – Improve and optimize multichat feature and then start work on class based bots, learn langchain embeddings
* User Story 2 Jesus Valdes - Continue to work on and completing the multi-chat feature and helping the other user stories
* User Story 3 Kristian Vazquez- Continue working on the user auth with FIU SSO login and multichat feature to select the chat from chat history, along with the chat histories being sorted from newest to oldest
* User Story 4 Miguel Garcia- Looking into Langchain embeddings and implement new embeddings to bots, and have FIU SSO login implemented for user credentials.
* User Story 5 Brian Otero- Finish up on the Multi-chat feature. start creating the class specific chatbot and help out on other user stories.

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

All of our user stories were completed on time.

**Sprint Review Meeting Minutes (Sprint 3)**

Project: STARS Tutoring Chatbot

Attendees: Leandro Alvarez, Brian Otero, Kristian Vazquez, Miguel Garcia, Jesus Valdes

Start time: 2:00

End time: 3:00

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

* User Story 1 Leandro Alvarez – Submit SSO form, talk with Andres regarding future implementations. Class specific bots.
* User Story 2 Jesus Valdes - Work on the class specific chatbot and finetuning
* User Story 3 Kristian Vazquez- Continue to work on class specific chatbot, finetuning the chatbots, and work on SSO login once team lead receives access.
* User Story 4 Miguel Garcia- Work on SSO authentication, and work on finetuning class specific bot
* User Story 5 Brian Otero- Continue to work on the class specific chatbot and add restrictions to the chatbot to ensure topic relevance

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

All of our user stories were completed on time.

**Sprint Review Meeting Minutes (Sprint 4)**

Project: STARS Tutoring Chatbot

Attendees: Leandro Alvarez, Brian Otero, Kristian Vazquez, Miguel Garcia, Jesus Valdes

Start time: 2:00

End time: 3:00

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

* User Story 1 Leandro Alvarez –
  + Work on examples for DB management class
  + Research power automate triggers
  + Work on tutor completions and few shot prompt
* User Story 2 Jesus Valdes -
  + Continue Working on User UI
  + Work on examples for the Logic for CS bot
  + Help with the reducing system prompt and Tutors side UI
* User Story 3 Kristian Vazquez-
  + User’s chats will be cached
  + caching functions will be tested for metrics
  + User will have a fine-tuned class for Algorithm Techniques.
* User Story 4 Miguel Garcia-
  + Have functioning UI for tutors to implement examples for chatbot
  + Work on getting examples for computer Architecture to have a 4th class related bot.
* User Story 5 Brian Otero-
  + Help out Miguel with the UI of the tutors
  + Continue to make the website more presentable following HCI principles

Start working on the Operating System class bot

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

All of our user stories were completed on time.

**Sprint Review Meeting Minutes (Sprint 5)**

Project: STARS Tutoring Chatbot

Attendees: Leandro Alvarez, Brian Otero, Kristian Vazquez, Miguel Garcia, Jesus Valdes

Start time: 2:00

End time: 3:00

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

* User Story 1 Leandro Alvarez –
  + - Work on creating functions for student enrollment
    - Look into tidio and have meeting with professor gilal over how student enrollment currently works
    - Automate student enrollment after tidio form upload.
* User Story 2 Jesus Valdes –
  + Work on the improvement off loading times , Documentation and Class bot
* User Story 3 Kristian Vazquez-
  + Admin user should be able to perform CRUD operations on courses
  + Create MongoDB remote cluster
  + Hash passwords for all users
* User Story 4 Miguel Garcia-
  + Finish Asmin GUI to accept or decline student users based on enrollments
  + Look into SSO authentication for student enrollment
  + Finish CRUD operations for class table
* User Story 5 Brian Otero-
  + Improve loading times
  + Start working on the documentation

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

All of our user stories were completed on time.

**Sprint Review Meeting Minutes (Sprint 6)**

Project: STARS Tutoring Chatbot

Attendees: Leandro Alvarez, Brian Otero, Kristian Vazquez, Miguel Garcia, Jesus Valdes

Start time: 2:00

End time: 3:00

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

* User Story 1 Leandro Alvarez –
  + Admin page, tutor sign in, email api
  + Move examples/ completions to db
  + Make fine tuned models using current completions
  + Integrate SSO
* User Story 2 Jesus Valdes -
  + Continue to work on the documentation and any coding the groups needs
* User Story 3 Kristian Vazquez-
  + Finish db optimization guide
  + Work on posters
  + Looking into hosting options
  + Query Params
* User Story 4 Miguel Garcia-
  + Work on posters
  + Continue work on Admin page functions
  + Finish Computer Architecture examples for optimization
* User Story 5 Brian Otero-
  + Start work on the posters
  + Continue to work on the documentation
  + Continue to bug fix

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

All of our user stories were completed on time.

**Sprint Review Meeting Minutes (Sprint 7)**

Project: STARS Tutoring Chatbot

Attendees: Leandro Alvarez, Brian Otero, Kristian Vazquez, Miguel Garcia, Jesus Valdes

Start time: 2:00

End time: 3:00

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

* User Story 1 Leandro Alvarez –
  + Worked on poster
  + Future Dev docs
  + Final development implemenations
* User Story 2 Jesus Valdes -
  + Finish Presentation
  + Finish Documentation
* User Story 3 Kristian Vazquez-
  + Finish Video Presentation
  + Make sure final deliverable is ready
  + Finish hosting the application using AWS
* User Story 4 Miguel Garcia-
  + Finish presentation slide
  + Finish video presentation
* User Story 5 Brian Otero-
  + Finish up on the presentation slides
  + Finish the videos
  + Have everything organized for the final deliverable

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

All of our user stories were completed on time.

## Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents

STAR TUTORING CHATBOT USER MANUAL

**1. Introduction**

Welcome to the STARS Tutor Chatbot application! This tool supports FIU students in their CS courses using class-specific AI chatbots. Tutors and administrators are also provided with dedicated dashboards to enhance and oversee the system.

**2. Getting Started**

To launch the app:

1. Install dependencies listed in requirements.txt.
2. Set environment variables in .streamlit/secrets.toml.
3. Start MongoDB locally or configure remote access.
4. Run the application:

streamlit run main.py

**3. User Authentication**

**3.1 Sign Up**

* Click **Sign Up**.
* Provide: FIU Email, Name, Panther ID, Password.
* Students upload PDF schedule.
* Tutors use an admin-provided code to access tutor options and select courses.
* Submit form and await admin approval.

**3.2 Log In**

* Click Sign In.
* Enter registered email and password.

**4. Using the Chatbot**

After logging in:

* Choose a bot: TutorBot, CodeBot, or course-specific.
* Interact with the selected bot using natural language questions.
* Responses are generated using LangChain with fine-tuned completions.
* Bot answers prioritize concept guidance over giving direct answers.

**5. Switching and Managing Chats**

* Start New Chat: Creates new chat object for the selected bot
* Switch Chats: Use the sidebar to view summaries and pick previous chats.
* Delete Chat: Remove the current chat (saved in archive).

**6. Fine-Tuning for Tutors**

If logged in as a tutor:

* View message history.
* Select any question-answer pair to fine-tune.
* Fill out follow-up details and submit improved answer.
* System validates and stores it as a few-shot training example.

**7. Admin Dashboard**

Accessible to users with admin role after login.

**7.1 Courses Management**

* Add/Remove Courses using input forms.
* Archive all chats at the end of a semester.

**7.2 Students Management**

* View pending/enrolled students.
* Approve, reject, or request resubmission.
* Download student chat history as JSON.

**7.3 Tutors Management**

* View pending/enrolled tutors.
* Set or generate tokens for tutor registration.
* Approve/reject tutors.

**8. Troubleshooting**

* Can't Log In: Make sure you're using a valid @fiu.edu email.
* No Courses Detected: Ensure uploaded PDF contains "Schedule of Classes".
* Chat Not Saving: Confirm MongoDB is running and secrets.toml is configured.
* No Admin Privilege: Make sure to change the user\_type to “admin” in MongoDB

Installation/Maintenance Document

**Create Virtual Environment:**

Run the following command:

* python -m venv venv

**Activate Virtual Environment:**

Mac/Linux:

* source venv/bin/activate

Windows:

* venv\Scripts\activate

**Install Dependencies:**

Run the following command:

* pip install -r requirements.txt

To install all the requirements for our program , those being:

* ﻿langchain
* langchain\_community
* langchain\_openai
* chromadb
* pymongo
* streamlit
* cachetools
* jsonlines
* groq
* streamlit-modal
* pypdf
* pypdf[crypto]
* sib-api-v3-sdk

**Set Up Environment Variables:**

Create a .streamlit folder in the root directory, followed by a secrets.toml inside the folder.

Add the following variables:

* OPENAI\_API\_KEY=your OpenAI API key
* GROQ\_API\_KEY=your Groq API key
* BREVO\_API\_KEY= your Brevo API key

If you are connecting to an external database you will also include

DB\_USER = your user

DB\_PWD = your password for the DB

DB\_IP =the DB IP

If you are not connecting to a external database , you should install your own local mongo db compass and change the user and the password as needed.

MongoDB Setup

**Download MongoDB Compass (GUI):**

[Download MongoDB Compass](https://www.mongodb.com/try/download/compass)

**Create New Connection in Compass:**

Under the new connection, create the following databases:

* user\_data
  + Inside user\_data, create the collections users and tokens .
* courses
  + Inside courses, create the collection course\_list.
* chat\_app
  + Inside chat\_app, create the collection chats .

**Feature Branch Workflow**

**Create a New Branch:**

Run the following command:

* git checkout -b feature-branch-name

**Make Changes to the Code:**

* Edit your code as needed.

**Commit the Changes:**

Run the following command:

* git commit -m "commit message"

**Push the Changes:**

Run the following command:

* git push origin feature-branch-name

Create a Pull Request:

* Open your repository on GitHub and create a pull request for your feature branch.

**Run The Program**

Run the following command:

* streamlit run main.py

**Shortcomings/Wishlist**

**Shortcoming**

* Unable to integrate FIU SSO due to lack of time. We requested early January and it was received less than 2 weeks ago.

**Wish**

* SSO needs to pass back email attribute, check if inside db then authenticate. If not in DB then redirect to signup

**Shortcoming**

* Automated emails do not send to FIU domain this is because we need DNS records from IT department but ran out of time.

**Wish**

* Contact IT Department and ask for DNS records. Look at brevo api for specific records needed.

**Shortcoming**

* Fine Tuning: continuous model training would be too expensive, therefore storing training data and triggering it later is the only option.​

**Wish**

* Create a developer app for creating new training data and triggering the fine tuning of models

**Shortcoming**

* We were restricted from deploying on AWS, told to leave it for next semester so we deployed on a raspberry pi.

**Wish**

* Deploy database and web app on AWS. Use AWS private link as a secure private endpoint

# References

* [Streamlit Documentation](https://docs.streamlit.io/)
* [MongoDB Python Driver](https://pymongo.readthedocs.io)
* [OpenAI API](https://platform.openai.com/docs)
* [LangChain](https://www.langchain.com/)
* [Brevo](https://developers.brevo.com/)
* [Groq](https://console.groq.com/docs/overview)