BattleBuddy: An AI Machine Learning Companion

Connie Rodriguez, Beth Gallatin

Saint Martin's University

CSC 482 – Senior Capstone Project

Dr. Radana Dvorak

March 30, 2025

Note to the Evaluation Advisory Board (EAB)

This report represents the current state of progress for the Vets2Tech platform as of this submission. While the system architecture, core functionality, and design framework have been established, the project is still undergoing its testing and validation phases. With turning this project into the EAB board early, this report is not yet complete. We have until the beginning of May to complete this report.

Due to this early turn in we would like to respectfully note the following:

- Final testing is still in progress. As such, certain sections, particularly those related to evaluation and results, contain placeholders or projections based on anticipated outcomes.
- Section VI: *Results and Evaluation* will be completed upon the conclusion of the full testing process and user feedback collection.
- Any testing-related content included in this version of the report reflects planned methodologies and expected practices, but not yet finalized findings.

We appreciate your understanding as we continue to refine and validate our work, and we are committed to delivering a fully tested and evaluated version of the system by the final report deadline.

Thank you for your time, guidance, and support.

Sincerely,
The Vets2Tech Team

Executive Summary

The BattleBuddy platform is an AI-powered educational web platform developed to support veterans and non-traditional students pursuing careers in technology. The platform provides adaptive quizzes, an intelligent chatbot, and curated learning materials aligned with accelerated certification and professional development programs.

Designed to serve prospective, current, and graduated students, the platform features role-based access, topic-driven learning paths, and dynamic progress tracking. This project is being developed using Django and Python for the backend, with a modular architecture to support future scalability and cloud deployment. Testing and development are currently being conducted on a local server environment as part of the project's proof of concept phase.

WAV2T aims to address institutional goals by improving student onboarding, success, and certification readiness while enabling data-driven decision-making by administrators. The system's design prioritizes accessibility, security, and usability, ensuring it can grow alongside the needs of the program and its stakeholders.

I. Introduction

1. Problem Statement

Many veterans and non-traditional students face barriers when transitioning into academic programs and technology careers. These barriers include gaps in foundational knowledge, lack of preparation for entrance exams, and difficulty accessing personalized study resources. Additionally, program administrators face challenges in assessing student readiness, tracking progress, and providing targeted support.

There is a clear need for an integrated, adaptive, and user-friendly platform that guides students through academic preparation while offering actionable insights to instructors and staff.

2. Project Objectives

The WAV2T project seeks to develop a secure, accessible, and scalable web-based platform that addresses the following key objectives:

- Provide prospective students with entrance and skills assessment exams to determine readiness for program pathways.
- Offer adaptive practice exams and study materials that adjust to individual student performance.
- Support current and graduated students preparing for industry-recognized certifications.
- Deliver actionable insights and data analytics to administrators for tracking program outcomes and continuous improvement.
- Offer on-demand study support through a chatbot powered by OpenAI.
- Ensure compliance with data privacy, accessibility, and institutional IT standards.

3. Project Scope

The WAV2T platform includes the following core features:

- **Secure Login & Role-Based Access:** Supports prospective, current, and graduated students as well as administrators.
- **Entrance Exams:** Timed, randomized exams that determine placement into one of three learning tracks: Server and Cloud Applications (SCA), Cloud Application Development (CAD), or Cybersecurity Administration.

- **Adaptive Practice Exams:** Real-time difficulty adjustment based on student performance using backend logic tied to student progress.
- **Study Materials Repository:** Pathway-specific content including videos, documents, and practice exams.
- **AI-Powered Chatbot:** An intelligent assistant that answers study-related questions and helps users navigate the platform.
- **Progress Tracking & Reporting:** Allows students to monitor their improvement while giving admins insight into student performance trends.

The project is currently in a proof-of-concept phase with local hosting, using Django for the backend and static HTML templates for the front end. Full deployment, including cloud hosting, email integration, and advanced analytics, is planned for future phases.

4. Stakeholders

The following stakeholders are directly impacted by the WAV2T platform:

- **Prospective Students:** Individuals evaluating their readiness for entry into one of the WAV2T educational pathways.
- **Current Students:** Enrolled participants using the platform to access study materials and practice exams.
- **Graduated Students:** Alumni seeking to prepare for industry certifications through continued access to learning tools.
- **Administrators and Faculty:** Responsible for managing exams, monitoring student progress, and using analytics for program decisions.
- **Engineering Advisory Board (EAB):** Provides feedback and oversight on platform functionality and strategic alignment.
- **Development Team:** Student developers responsible for designing, building, and maintaining the platform during its initial phases.
- **Dr. Dvorak:** Faculty sponsor ensuring academic oversight and alignment with educational outcomes.

II. Background and Literature Review

1. Background

As the demand for skilled technology professionals grows, educational institutions are under increasing pressure to create accessible and effective pathways into technical careers. This challenge is especially critical for veterans and non-traditional students, who often bring valuable real-world experience but may lack recent academic exposure or familiarity with modern learning platforms.

Traditional educational systems frequently fail to provide personalized, adaptable resources to support these learners, leading to high attrition rates and underutilized talent. The Washington Vets 2 Tech (WAV2T) initiative was established to bridge this gap by offering a structured, technology-driven learning platform that caters specifically to the unique needs of transitioning service members and non-traditional learners.

WAV2T's goal is not only to support academic entry through entrance and skill assessment exams but to foster long-term career readiness by offering adaptive study tools and certification preparation. The platform is designed with flexibility in mind, enabling use by prospective, current, and graduated students, each with distinct learning needs and goals.

2. Literature Review

2.1 Adaptive Learning Systems

The concept of adaptive learning has gained significant traction in recent years. According to research by Brusilovsky and Millán (2007), adaptive learning systems personalize educational content by tracking user performance and adjusting instruction accordingly. These systems use learner models to tailor difficulty levels, content sequencing, and feedback in real time.

WAV2T incorporates this model by tracking student quiz performance using Django-based progress models. The backend dynamically adjusts the difficulty of subsequent questions, creating a learning experience that evolves with the student—much like an intelligent tutor. This personalized approach improves learner engagement and outcomes by meeting students where they are academically.

2.2 Technology for Veteran and Non-Traditional Learners

Multiple studies (e.g., Griffin & Gilbert, 2015; Vacchi & Berger, 2014) emphasize the unique needs of veterans in higher education. These include the need for flexible learning, technology-supported study aids, and systems that validate prior experience while building new skills. WAV2T is directly aligned with these findings, offering self-paced entrance assessments and adaptive materials designed to support knowledge refreshment and skill-building.

The system's structure supports asynchronous learning and personalized tracking, reducing stress and offering a smoother transition for users who may be re-entering formal education after a gap.

2.3 Chatbots and AI in Education

AI-powered chatbots have become increasingly prevalent in higher education. Studies like Winkler & Söllner (2018) show that well-designed educational chatbots can enhance learner satisfaction, increase help-seeking behavior, and reduce instructor burden. WAV2T integrates OpenAI's language model to create a chatbot study buddy that supports student navigation, answers course-related questions and improves engagement through immediate feedback and guidance.

This aligns with ongoing research that suggests conversational agents can significantly improve accessibility, especially for students who may hesitate to seek help from instructors.

2.4 Technical Foundation: Web-Based Educational Platforms

The technical foundation of WAV2T is grounded in modern web development best practices. Built using Django (a high-level Python framework), the platform adopts the Model-View-Template (MVT) pattern, which enhances maintainability, scalability, and security. Django's built-in authentication, ORM, and admin tools streamline backend development while ensuring proper data handling and role-based access.

For frontend responsiveness and accessibility, standard technologies such as HTML5, CSS3, JavaScript, and Bootstrap are employed. This ensures that WAV2T is fully compatible with a range of devices and meets accessibility standards such as WCAG 2.1.

Although currently hosted locally during its proof-of-concept stage, the platform's modular design enables seamless migration to scalable cloud solutions such as AWS, Azure, or institutional servers in the future.

3. Relevance and Contribution

WAV2T contributes to the growing field of educational technology by:

- Offering a veteran-centric, modular learning platform tailored to underserved student populations.
- Implementing a data-driven adaptive quiz engine based on proven models of personalized education.
- Integrating AI-driven support tools to improve usability and engagement.

- Aligning with industry certification standards, helping students transition directly into workforce opportunities.

Through the combination of educational theory, technical precision, and accessibility-focused design, WAV2T fills a critical gap in postsecondary and workforce education systems.

III. System Requirements

The WAV2T (Washington Vets 2 Tech) platform is an educational web application developed to support prospective, current, and graduated students in their academic and professional advancement. The system is built with a modular, secure architecture using Python and the Django framework, hosted locally for its proof-of-concept phase with future scalability in mind. Below is an overview of the system requirements, categorized into functional, non-functional, and design constraints.

Functional Requirements

At its core, the WAV2T platform delivers a set of critical features that directly support users along three primary educational pathways: Server and Cloud Applications (SCA), Cloud Application Development (CAD), and Cybersecurity Administration (Cyber). While we understand that the CAD program is not currently being offered, we wanted to showcase the toggle option. This option allows the admin to hide non-active courses allowing for greater flexibility to the dynamic needs of the program. The system allows users to register and authenticate securely using a role-based access model that differentiates between prospective students, enrolled students, graduates, and administrators.

Key functional capabilities include a vast exam management module, enabling administrators to create and administer randomized, timed entrance exams tailored to each learning path. A skills assessment feature further refines placement and study planning. Additionally, the platform offers an extensive study material repository with curated resources, such as videos, texts, and practice exams, and supports an adaptive learning engine. This engine uses machine learning to tailor exam content based on user performance, promoting more personalized and effective study experiences.

Students interact with a streamlined, multi-tabbed user interface that organizes features by educational track. Administrators, in turn, access a centralized dashboard for user management, content updates, exam grading, and performance analytics. Notifications, currently limited to internal push alerts, keep both students and staff informed of grading events, new resources, and progress milestones.

A key innovation within WAV2T is its AI-powered chatbot study assistant, which supports students by answering queries and directing them to appropriate study materials. Though currently focused on student interaction, plans for expanded administrator functionality are included in the system roadmap.

Non-Functional Requirements

To ensure a high-quality user experience, the WAV2T platform incorporates numerous non-functional considerations. The system's user interface is designed for accessibility and responsiveness, complying with Web Content Accessibility Guidelines (WCAG) and optimized for use across a range of devices and screen sizes. The platform prioritizes usability, with the goal that 90% of new users should be able to navigate its core functions without external guidance.

Performance benchmarks include a sub-500ms response time for page loads under typical operating conditions and the ability to support 500 concurrent users without notable degradation in performance. Real-time interactions, such as exam submissions and internal notifications, are processed within a two-second window, ensuring responsiveness critical to timed assessments and progress tracking.

Security is a central concern. Django's built-in authentication mechanisms, combined with role-based access control, password hashing, and session management, provide a strong foundation for user data protection. Future enhancements include two-factor authentication and encryption at rest using AES-256 once cloud deployment begins. The platform is designed to meet FERPA compliance standards and will undergo regular security audits and penetration testing when moved to production.

Maintainability and scalability are also integral to the system's design. Clear code documentation, adherence to PEP8 coding standards, and modular architecture support long-term development. The backend is structured to facilitate deployment via containers or cloud infrastructure such as AWS, with current use of SQLite allowing for rapid prototyping and future database migration paths already outlined.

Constraints

Several constraints have influenced the system's development. The platform is currently hosted locally, limiting real-time scalability and third-party integrations such as email notifications or institutional APIs. SQLite serves as the initial database engine, which, while effective for development, may not scale well with production-level traffic or data volumes. Budget constraints and institutional requirements have also delayed implementation of features such as full email integration, external identity provider support, and advanced reporting dashboards.

Technology choices, including Django as the backend framework, have shaped compatibility and development workflows, aligning the system with open-source ecosystems rather than proprietary tools like .NET. Additionally, accessibility, usability, and security standards impose strict guidelines for interface design, encryption protocols, and user role management, which must be carefully balanced against development timelines and resource limitations.

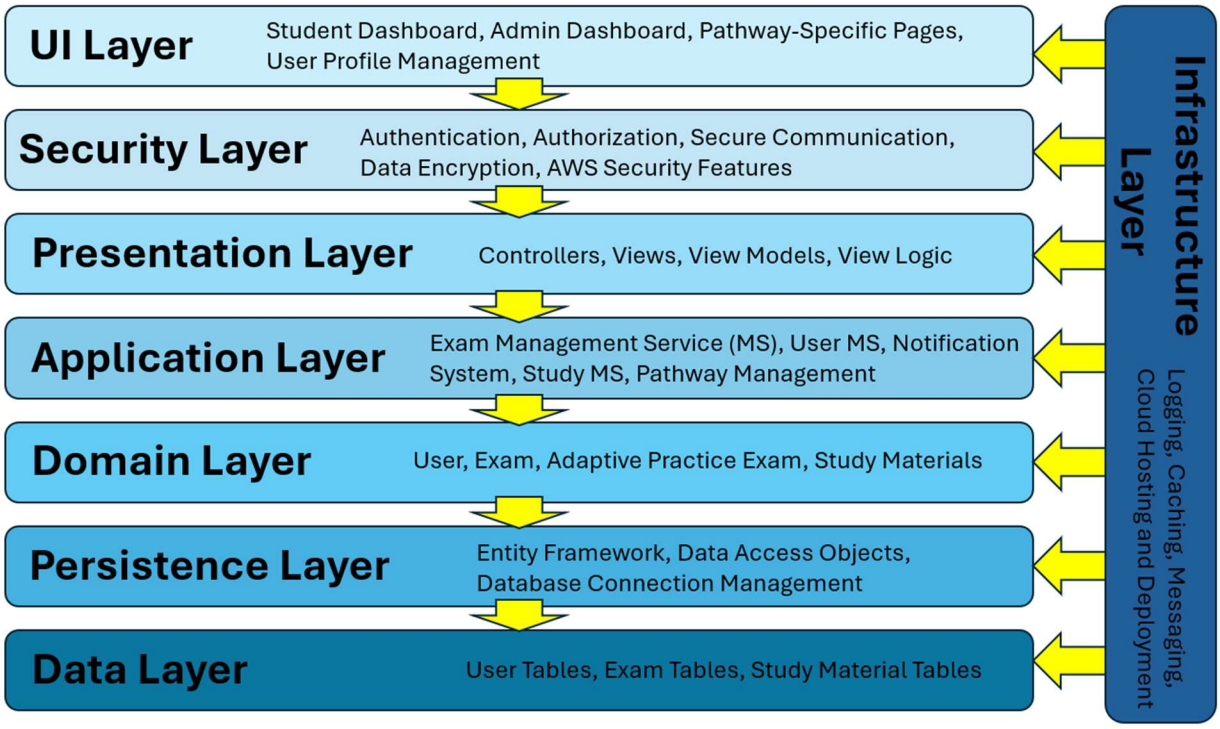
IV. System Design

The system design of the WAV2T platform encompasses the overall architecture, user interface layout, security structure, and major software components. The platform follows a layered, modular architecture that ensures scalability, maintainability, and security. The system is currently in a locally hosted proof-of-concept phase, with infrastructure and design choices that support a seamless transition to full-scale deployment.

1. System Architecture

The WAV2T platform uses a multi-layered architecture composed of the following primary layers:

- **User Interface (Presentation Layer):** Built using Django templates, HTML, CSS, and JavaScript, this layer handles all user interactions and provides an accessible and responsive front-end experience across devices.
- **Application Layer:** Manages the business logic and routes user input to appropriate services. This includes quiz logic, adaptive test mechanisms, chatbot interactions, and dashboard updates.
- **Domain Layer:** Defines the core models and entities (e.g., User, Exam, StudyMaterial) and enforces rules on how these interact.
- **Persistence Layer:** Interfaces with the SQLite database through Django's Object-Relational Mapping (ORM), managing CRUD operations and schema migrations.
- **Security Layer:** Embedded throughout the stack, responsible for authentication, access control, session management, and data protection.

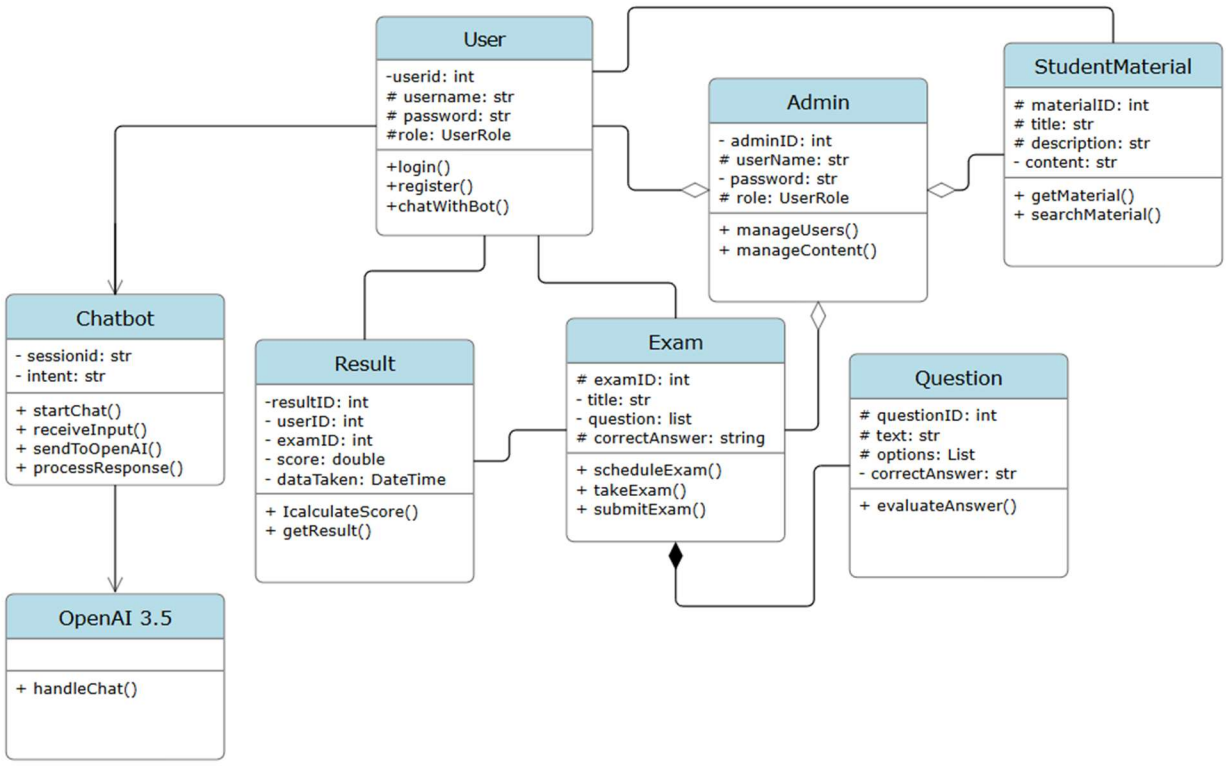


2. UML Design

The platform's main entities are structured using Django models and follow object-oriented design principles. The UML class diagram illustrates the relationships between core objects such as User, Exam, Result, and StudyMaterial.

Key design relationships include:

- One-to-many relationship between User to Exam, User to Result, Admin to User, Admin to Exam, Exam to Result, and Exam to Question.
- One-way relationship between User to Chatbot, and Chatbot to OpenAI.
- Many-to-many relationships between Admin to StudyMaterial.



3. User Interface (UI) Design

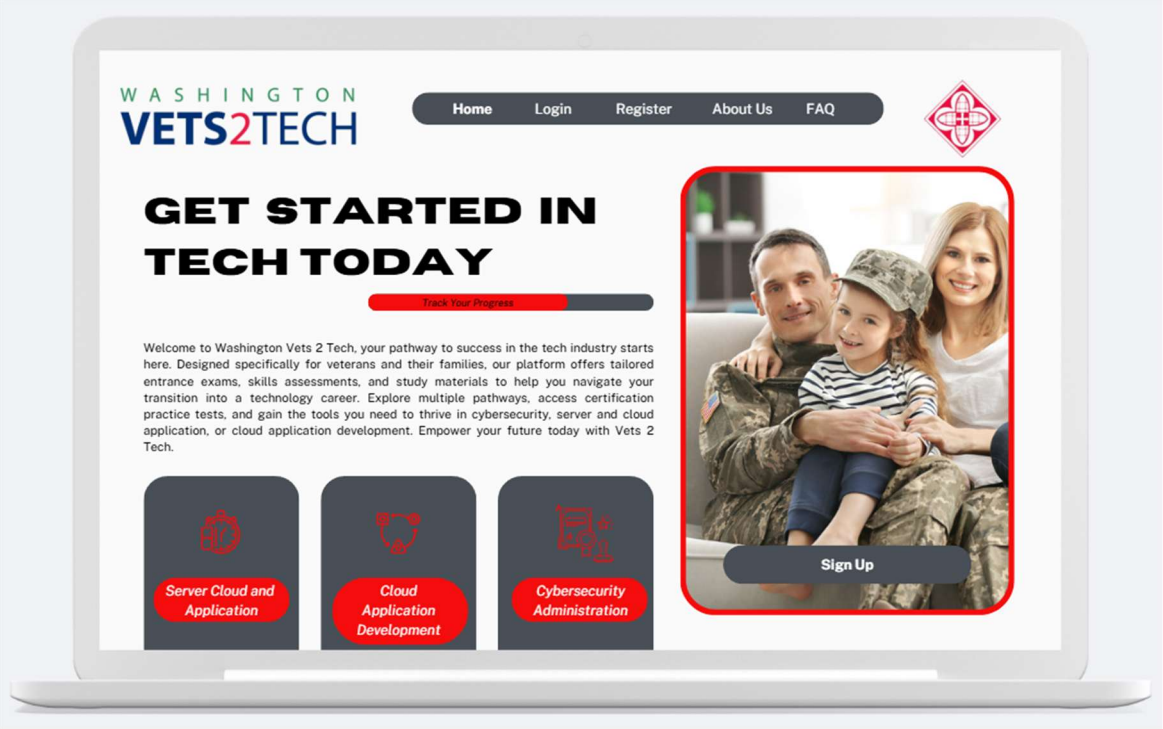
The UI was designed with simplicity and accessibility in mind, ensuring that all users, from prospective students to administrators, can intuitively navigate the platform. The multi-tabbed layout provides clear pathways for SCA, CAD, and Cyber users, while admin users have access to a separate dashboard for content and account management.

Core interface components include:

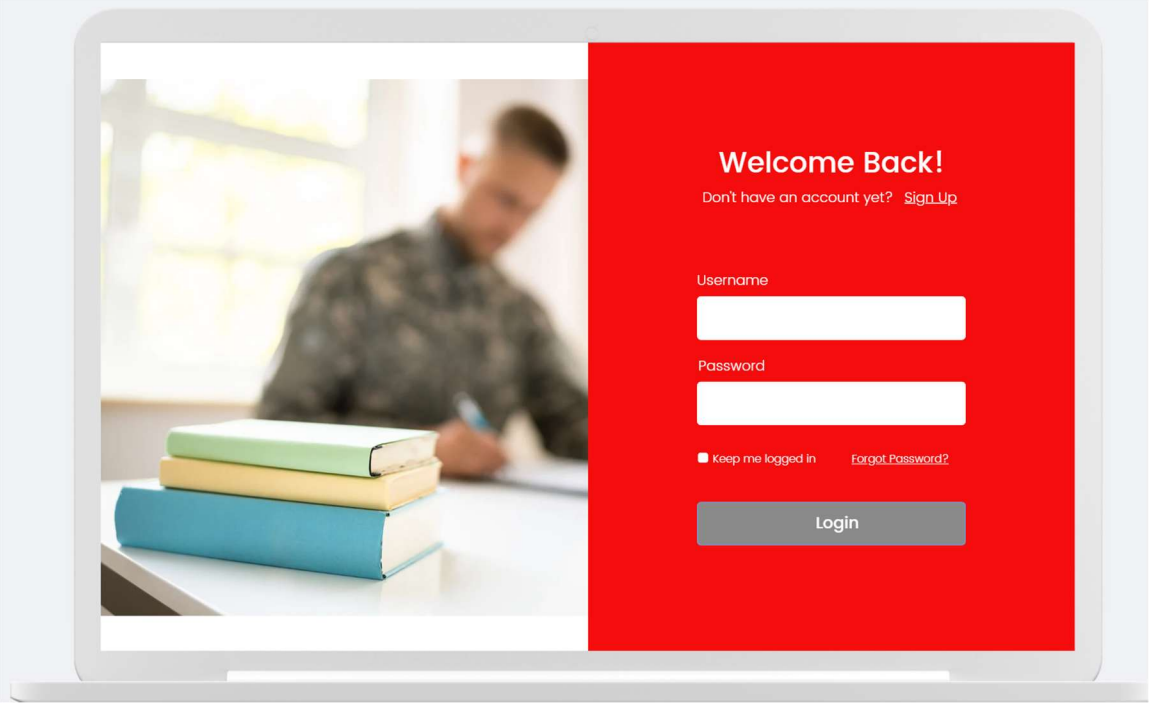
- **Login/Registration Page:** Secure access point with CSRF protection and future-ready two factor authentication (2FA) prompts
- **Student Dashboard:** Personalized with pathway-specific tabs, progress tracking, and study resources
- **Admin Dashboard:** Central hub for exam management, notifications, and analytics
- **Chatbot Interface:** Integrated into all pages, providing contextual support and study guidance

Wireframes:

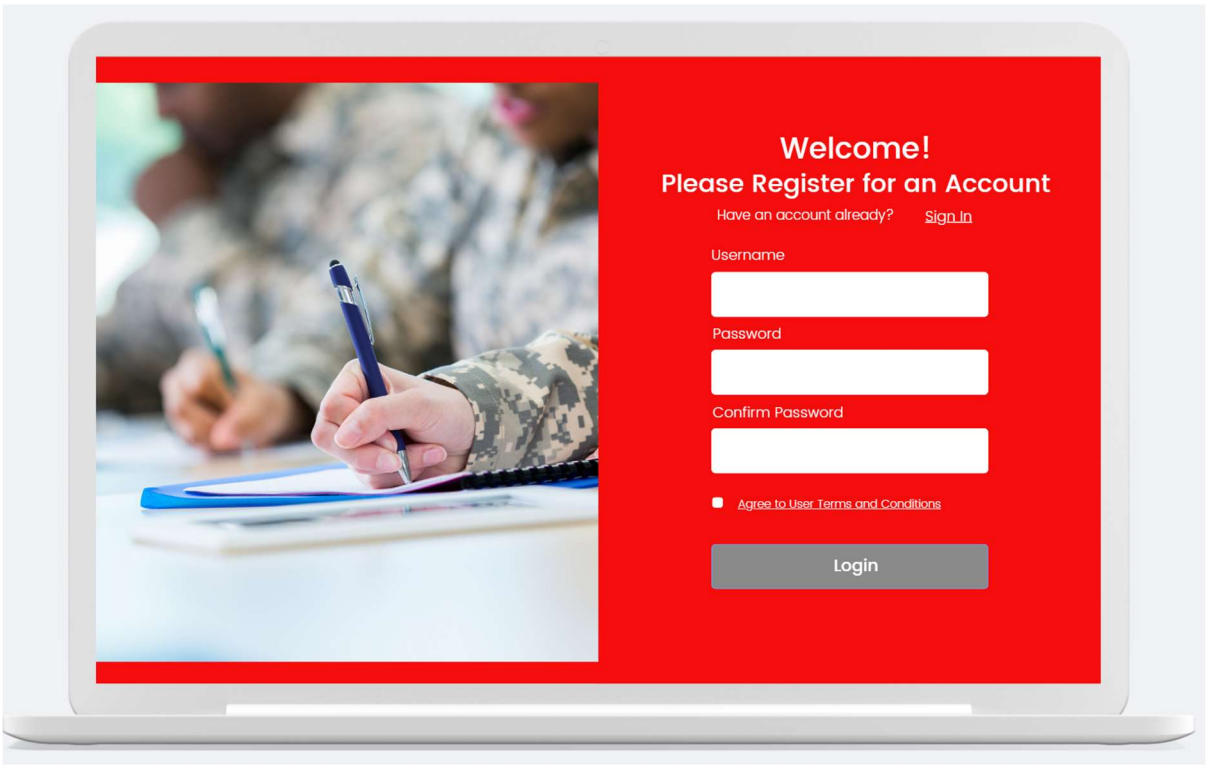
Home Page:



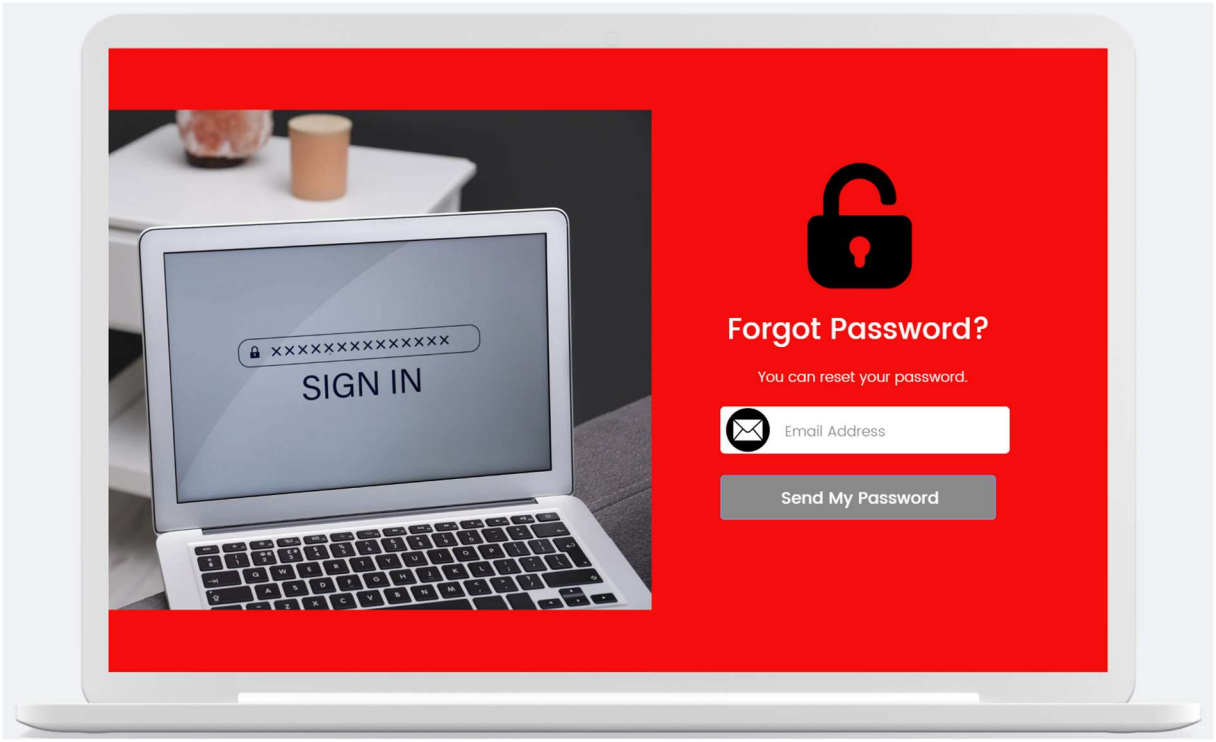
Login:



Register:



Forgot Password:



4. Security Design

Security is embedded at every level of the WAV2T platform. The system currently uses Django Allauth for authentication, and role-based access control is enforced through Django's permission groups and middleware.

Key security features include:

- **Authentication:** Users log in securely via Django's built-in auth system. All passwords are hashed using PBKDF2.
- **Authorization:** Role-based access ensures only authorized users can access exams, results, or admin functions.
- **Data Encryption:** Sensitive data is encrypted in transit using HTTPS and hashed at rest.
- **Session Management:** Sessions expire after inactivity, and login attempt limits help mitigate brute-force attacks (Django-Axes).
- **Audit Logging:** Administrative actions and login attempts are logged for future monitoring and compliance.

Planned enhancements include multi-factor authentication (MFA), email-based alerts, and SSO integration for production environments.

V. System Implementation

The implementation phase of the WAV2T platform focused on building a secure, modular, and user-centric web application using industry best practices. This section outlines the methodology employed, the organization of the project codebase, implementation strategies, and the testing processes used to validate functionality and performance.

1. Development Methodology

The WAV2T project adopted an Agile development methodology to facilitate continuous improvement, flexibility, and rapid iteration. Development was organized into sprints, with regular check-ins and milestone reviews. Key elements of the methodology included:

- **Iterative Development:** Functional components such as user authentication, exams, and study materials were developed in phases, allowing for early testing and stakeholder feedback.
- **Collaboration:** The team used version control (Git) and GitHub for source code collaboration, ensuring transparency and coordination between developers.

- **User-Centered Design:** Regular usability reviews were conducted to refine interfaces and improve accessibility, informed by the needs of students and administrators.

This approach enabled the team to adjust priorities based on ongoing feedback and technical feasibility while delivering stable, functional builds throughout development.

2. Code Implementation

The system is implemented using Python and the Django web framework. Django's built-in components, such as the ORM, templating engine, and admin interface, streamlined backend development while maintaining a secure and scalable architecture.

Key technologies and features include:

- **Frontend:** Django templates (HTML, CSS, JavaScript, Bootstrap) for responsive, accessible UI components.
- **Backend Logic:** Django views and models process data, enforce business logic, and route requests. Custom logic handles adaptive quiz behavior, exam grading workflows, and chatbot integration.
- **Database:** SQLite is used during development for lightweight data persistence. Models define schema for users, exams, results, and study materials.
- **Chatbot Integration:** The OpenAI GPT-3.5 API powers an intelligent chatbot embedded within the student interface to assist with study-related queries.
- **Security:** Django Allauth handles authentication and role-based access, with middleware enforcing secure sessions and user permissions.

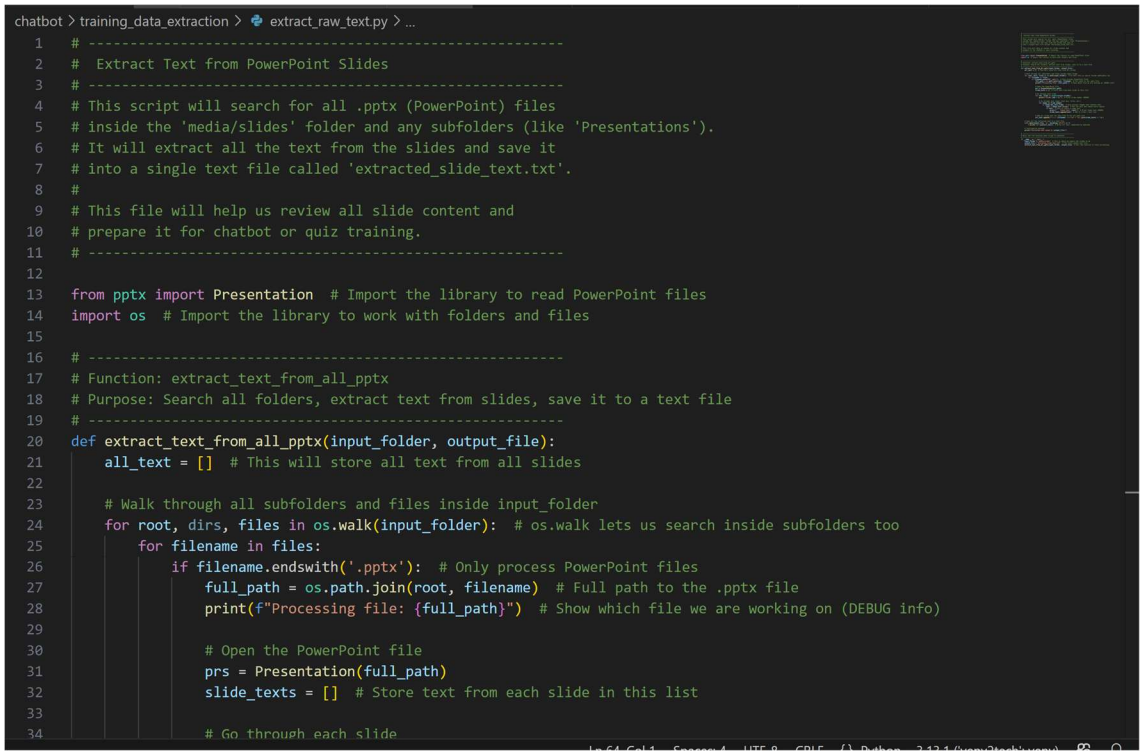
Code Snippet (Adaptive Practice Exam Logic):

```python
from django.db import models
from django.contrib.auth.models import User  # Links quiz progress to users

class Question(models.Model):
    """
    Stores a single quiz question with multiple-choice answers.
    Each question is tagged with a topic and difficulty level to support adaptive learning.
    """
    # Difficulty levels used to dynamically adjust question selection based on user performance
    DIFFICULTY_LEVELS = [
        ('easy', 'Easy'),
        ('medium', 'Medium'),
        ('hard', 'Hard'),
    ]

    # Predefined topics to categorize questions and track user strengths/weaknesses
    TOPICS = [
        ('cybersecurity', 'Cybersecurity'),
        ('networking', 'Networking'),
        ('cloud', 'Cloud'),
        ('computer_basics', 'Computer Basics'),
    ]
```

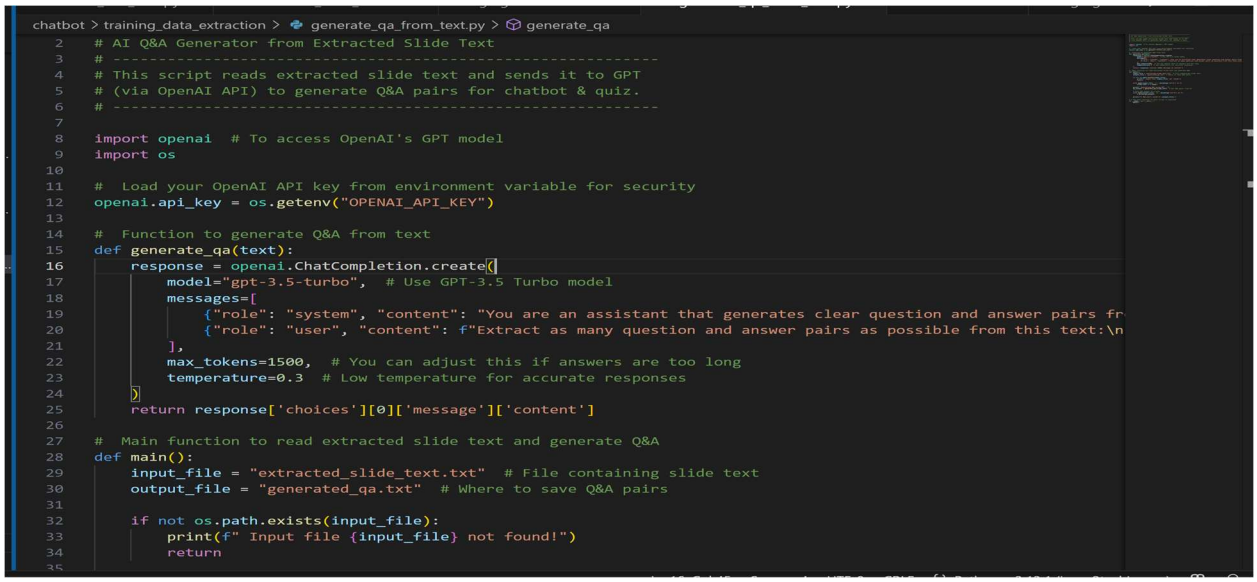
```python
    # Core question and answer fields
    question_text = models.TextField()  # Stores the question itself
    option_1 = models.CharField(max_length=255)
    option_2 = models.CharField(max_length=255)
    option_3 = models.CharField(max_length=255)
    option_4 = models.CharField(max_length=255)
    correct_answer = models.CharField(max_length=255)  # Correct answer for validation
    difficulty = models.CharField(max_length=10, choices=DIFFICULTY_LEVELS)
    topic = models.CharField(max_length=20, choices=TOPICS)

    #user will see topic and difficulty level when they are taking the quiz
    def __str__(self):
        return f"{self.topic} - {self.difficulty}: {self.question_text}"
```

```python
class UserQuizProgress(models.Model):
    """
    Tracks an individual user's performance in each topic and difficulty level.
    Enables adaptive question delivery and personalized feedback by storing ongoing results.
    """
    # User the progress data belongs to
    user = models.ForeignKey(User, on_delete=models.CASCADE)  # Links progress to a user

    # Topic and difficulty being tracked for targeted improvement
    topic = models.CharField(max_length=20, choices=Question.TOPICS)  # Topic being tracked
    difficulty = models.CharField(max_length=10, choices=Question.DIFFICULTY_LEVELS, default="easy")

    # Performance tracking fields
    correct_count = models.IntegerField(default=0)  # Number of correct answers
    incorrect_count = models.IntegerField(default=0)  # Number of incorrect answers

    #shows user the subject and their current level
    def __str__(self):
        return f"{self.user.username} - {self.topic} ({self.difficulty})"
```

Code Snippet (Load Training Data):



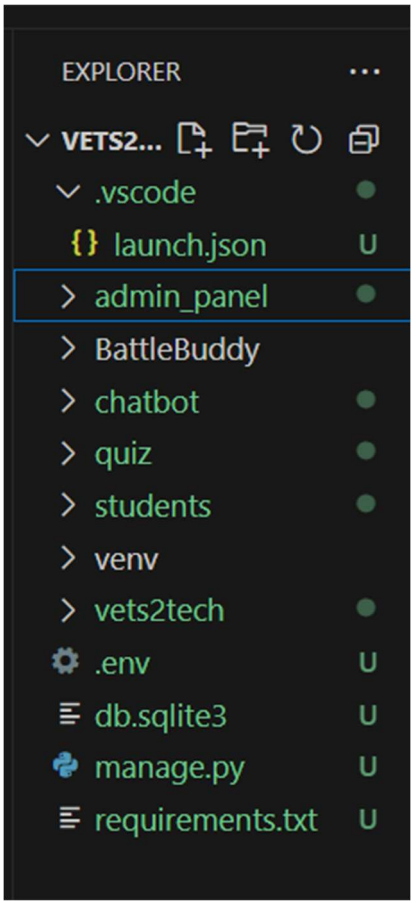
```
chatbot > training_data_extraction > extract_raw_text.py > ...
# ------------------------------------------------------------
#  Extract Text from PowerPoint Slides
# ------------------------------------------------------------
# This script will search for all .pptx (PowerPoint) files
# inside the 'media/slides' folder and any subfolders (like 'Presentations').
# It will extract all the text from the slides and save it
# into a single text file called 'extracted_slide_text.txt'.
#
# This file will help us review all slide content and
# prepare it for chatbot or quiz training.
# ------------------------------------------------------------

from pptx import Presentation  # Import the library to read PowerPoint files
import os  # Import the library to work with folders and files

# ------------------------------------------------------------
# Function: extract_text_from_all_pptx
# Purpose: Search all folders, extract text from slides, save it to a text file
# ------------------------------------------------------------
def extract_text_from_all_pptx(input_folder, output_file):
    all_text = []  # This will store all text from all slides

    # Walk through all subfolders and files inside input_folder
    for root, dirs, files in os.walk(input_folder):  # os.walk lets us search inside subfolders too
        for filename in files:
            if filename.endswith('.pptx'):  # Only process PowerPoint files
                full_path = os.path.join(root, filename)  # Full path to the .pptx file
                print(f"Processing file: {full_path}")  # Show which file we are working on (DEBUG info)

                # Open the PowerPoint file
                prs = Presentation(full_path)
                slide_texts = []  # Store text from each slide in this list

                # Go through each slide
```



```
chatbot > training_data_extraction > generate_qa_from_text.py > generate_qa
# AI Q&A Generator from Extracted Slide Text
# ------------------------------------------------------------
# This script reads extracted slide text and sends it to GPT
# (via OpenAI API) to generate Q&A pairs for chatbot & quiz.
# ------------------------------------------------------------

import openai  # To access OpenAI's GPT model
import os

#  Load your OpenAI API key from environment variable for security
openai.api_key = os.getenv("OPENAI_API_KEY")

#  Function to generate Q&A from text
def generate_qa(text):
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",  # Use GPT-3.5 Turbo model
        messages=[
            {"role": "system", "content": "You are an assistant that generates clear question and answer pairs fr
            {"role": "user", "content": f"Extract as many question and answer pairs as possible from this text:\n
        ],
        max_tokens=1500,  # You can adjust this if answers are too long
        temperature=0.3  # Low temperature for accurate responses
    )
    return response['choices'][0]['message']['content']

#  Main function to read extracted slide text and generate Q&A
def main():
    input_file = "extracted_slide_text.txt"  # File containing slide text
    output_file = "generated_qa.txt"  # Where to save Q&A pairs

    if not os.path.exists(input_file):
        print(f" Input file {input_file} not found!")
        return
```

3. Project Organization

The project followed a clear modular structure within the Django framework, using a separation-of-concerns model to keep logic clean and maintainable:



Each Django app is self-contained and includes its own models, views, templates, and URLs, which promotes scalability and easy onboarding for new contributors.

Version control was maintained using Git, with branches dedicated to features, bug fixes, and releases. GitHub Projects and Issues were used to track progress, assign tasks, and manage development cycles.

4. Testing and Validation

Testing was an essential part of the development process, ensuring the platform functioned reliably and securely across a range of user scenarios.

- **Unit Testing:** Individual models, forms, and views were tested using Django's built-in testing framework to validate logic and data handling.
- **Manual Testing:** Critical workflows (e.g., login, exam submission, chatbot queries) were manually tested across different browsers and devices to ensure proper rendering and responsiveness.

- **Accessibility Checks:** Keyboard navigation, color contrast, and screen reader compatibility were validated against WCAG guidelines.
- **Security Testing:** Password hashing, role-based access control, and input validation were tested to prevent unauthorized access and injection attacks.

VI. Results and Evaluation

Will be written once we are completed with the evaluation of our system.

VII. Project Management

Project Timeline

January 2025

- Transitioned from ASP.NET with C# to Django with Python.
- Rebuilt platform structure to support AI integration and adaptive learning.
- Established new database and user authentication system.
- Set up development environment and configured version control.

February 2025

- Implemented core features: quizzes, user authentication, and study material uploads.
- Began testing and refining the adaptive quiz system.
- Addressed cross-platform compatibility with responsive design.
- Resolved proprietary content issues.
- Presented project poster at the Engineering Banquet.

March 2025

- Conducted system-wide testing for performance, functionality, and security.
- Fixed bugs and optimized user experience.
- Finalized role-based access and admin panel controls.
- Created long-term maintenance documentation.

April 2025

- Delivered stakeholder presentation to the EAB board.
- Verified system component readiness and functionality.
- Conducted final testing and refinement at 85% milestone.
- Began preparations for Scholars Day and future enhancements.

Team Roles & Responsibilities

- **Beth Gallatin – Project Manager**
    - Oversees project timeline, coordination, and stakeholder communication.
    - Contributes to backend development and system integration.
- **Connie Rodriguez – Developer**
    - Leads both backend and frontend development efforts.
    - Implements authentication, quiz logic, and system functionality.
- **John McDurmon – UI/UX Designer**
    - Designs the user interface and ensures an intuitive user experience.
    - Implements accessibility and responsive design standards.

Communication Plan

- Weekly team stand-ups to assess progress and adjust tasks.
- Monthly client/stakeholder meetings for milestone alignment and feedback.
- GitHub used for version control and code collaboration.
- Trello board used to track sprint tasks and development flow.
- Shared documentation hub for technical reference and team onboarding.

Risk Management

- **Scope Creep**
  → Controlled with regular check-ins, milestone tracking, and change request reviews.
- **Data Loss**
  → Mitigated through version control (GitHub) and local backups.

- **Delayed Integration**
  → Addressed by building features modularly and testing early/often.
- **AI/Chatbot Failures**
  → Separated chatbot logic for isolated testing, using OpenAI 3.5 for proven reliability.
- **Accessibility or Usability Gaps**
  → Solved via adherence to WCAG standards, bilingual testing, and user feedback loops.

Future Development (Post-April 2025)

- Finalize adaptive quiz and chatbot systems.
- Add bilingual (English/Spanish) interface support.
- Polish accessibility and frontend UI.
- Enable data collection and analytics export (for R).
- Continue code cleanup and technical documentation.
- Begin work on infrastructure, deployment, and scalability for production use.

VIII. Future Work

As the Vets2Tech platform matures, several opportunities for continued development and improvement have been identified. These future efforts aim to enhance system reliability, user experience, and long-term sustainability.

Insightful Recommendations

- **Enhanced AI Personalization**: Continue to improve the AI chatbot companion by integrating user learning history to offer personalized study paths, contextual suggestions, and adaptive conversational support.
- **Mobile App Development**: Extend platform functionality to a mobile app to improve accessibility and usability for students on the go, especially veterans balancing work and education.
- **Advanced Analytics Dashboard**: Create a comprehensive admin dashboard for tracking student performance, quiz trends, and engagement metrics. Enable data export and visualization for continuous improvement.

- **Multi-language & Accessibility Expansion**: Build on bilingual support by integrating more languages and further improving accessibility for users with visual or cognitive impairments.
- **Gamification Features**: Introduce badges, progress bars, or leaderboards to improve student motivation and engagement without compromising the academic integrity of the platform.

Scalability Considerations

- **Infrastructure Optimization**: Transition from local hosting to cloud-based infrastructure (e.g., AWS EC2 with auto-scaling groups and RDS for managed databases) to accommodate increasing user loads efficiently.
- **Modular Architecture**: Refactor the backend using modular, service-oriented architecture to allow easier feature updates, team collaboration, and future microservice migration.
- **Database Indexing and Optimization**: As the platform grows, implement indexing strategies, database sharding (if needed), and optimized query patterns to maintain performance.
- **Load Testing and Auto-scaling**: Perform simulated load testing to identify breaking points and configure auto-scaling rules based on system metrics (CPU, memory usage).
- **Security & Compliance**: Plan for future implementation of advanced user authentication (MFA), audit trails, and potential compliance with standards such as FERPA or SOC 2.

References:

Peter Brusilovsky. School of Computing and Information | University of Pittsburgh. (2024, September 10). https://www.sci.pitt.edu/people/peter-brusilovsky

Better Transitions for Troops: An Application of Schlossberg's Transition Framework to Analyses of Barriers and Institutional Support Structures for Student Veterans: The Journal of Higher Education: Vol 86, No 1, www.tandfonline.com/doi/abs/10.1080/00221546.2015.11777357. Accessed 30 Mar. 2025.

(PDF) Student Veterans and the Transition to Higher Education: Integrating Existing Literatures, www.researchgate.net/publication/320191005_Student_Veterans_and_the_Transition_to_Higher_Education_Integrating_Existing_Literatures. Accessed 31 Mar. 2025.

(PDF) Student Veterans and the Transition to Higher Education: Integrating Existing Literatures, www.researchgate.net/publication/320191005_Student_Veterans_and_the_Transition_to_Higher_Education_Integrating_Existing_Literatures. Accessed 31 Mar. 2025.

Appendices:

To be added