Project Plan Documentation

Cringe Coders

Austin Bailey
Mya Bell
Nolan Gregory
Lindsey Langdon

Comprehensive Project Overview
for the CSLC Portal

Dr. Harvey Siy; College IS&T
University of Nebraska - Omaha
Capstone Course
August 25th, 2023

Contents

CHAPTER 1

Introduction

The Computer Science Tutoring Portal is a comprehensive web application designed to streamline and enhance the tutoring experience within the Computer Science Learning Center. This portal will serve as a centralized platform for students, tutors, and administrators to manage tutoring requests, track progress, and facilitate efficient communication. The application's key features include ticket management, tutor and course administration, and seamless interaction between students, tutors, and administrators.

CHAPTER 2

Project Organization

2.1 Austin Bailey

Client Liaison
Information Security Analyst
Quality Assurance Tester

Austin Bailey is responsible for thoroughly testing the application to identify vulnerabilities, usability issues, and performance bottlenecks. Austin will create and execute test cases, provide feedback to the development team, and ensure a high-quality final product. Likewise, Austin will simulate real-world cyberattacks to identify vulnerabilities and weaknesses. Austin will help our team improve our security posture and protect sensitive information by uncovering any potential security flaws.

2.2 Mya Bell

Requirements Analyst
Quality Assurance Tester
Client Liaison

Mya Bell will oversee the development process, and ensure that the project stays on track, objectives are met, and communication flows smoothly between team members. Likewise, Mya will facilitate communication and collaboration between leadership and team players to ensure a successful outcome. Lastly, Mya will create and execute test cases to ensure a high-quality final product.

2.3 Nolan Gregory

Architect
Technical Lead
Back-End Developer
Dev-Ops and Database

Nolan Gregory is responsible for making technical decisions and guiding the development process. Nolan will provide technical expertise, review code, and ensure that the architecture and technologies chosen align with the project's goals. Nolan will handle all server-side logic, database management, and API calls. Nolan will build the core functionalities of the application, including ticket management, user authentication, and communication features. Nolan will design and maintain the database structure, ensuring efficient data storage, retrieval, and integrity. Lastly, Nolan will set up the deployment infrastructure, manage continuous integration/continuous deployment (CI/CD) pipelines, and ensures smooth deployment and scaling of the ticketing portal.

2.4 Lindsey Langdon

UI/UX Lead
Front-End Developer
Mobile Design Lead

Lindsey Langdon is responsible for creating an intuitive and visually appealing user interface. Lindsey will design wireframes, mockups, and prototypes, ensuring a user-friendly experience and consistent branding. Lindsey will be responsible for implementing the user interface and ensuring that the application is responsive and visually engaging. They work closely with the UI/UX Designer to translate design concepts into functional front-end components. Lindsey will ensure that the front-end can run on mobile or tablet devices, as well as on desktop systems, with a focus on mobile-first design.

CHAPTER 3

Risk Analysis

3.1 Data Breach or Unauthorized Access

Threat Level: MODERATE to HIGH

Our team will implement encryption algorithms for sensitive data in transit. We will use a multi-factor authentication for user accounts (*OAuth*). Our team will conduct security reports and vulnerability assessments. Lastly, we will employ access control mechanisms to limit access based on user roles and user permissions.

3.2 Application Vulnerabilities

Threat Level: HIGH

Our team will follow secure coding practices such as: conducting regular security code reviews, implementing input validation and output encoding to prevent common vulnerabilities like SQL injection and cross-site scripting (XSS), and keep libraries and frameworks up to date to address known vulnerabilities.

3.3 Denial of Service (DoS) Attacks

Threat Level: MODERATE

Our team will implement rate limiting and CAPTCHA mechanisms to mitigate automated attacks and monitor server and network performance for unusual spikes in traffic.

3.4 Inadequate Authentication and Authorization

Threat Level: HIGH

Our team will implement a strong authentication mechanism, namely OAuth, and enforce role-based access control.

3.5 Poorly Managed Sessions and Cookies

Threat Level: MODERATE

Our team will use secure session management techniques, such as session timeouts and unique session identifiers.

3.6 Data Loss or Corruption

Threat Level: MODERATE

Our team will implement regular data backups and ensure data integrity. Likewise, we will use redundancy for pivotal components of the system.

3.7 Lack of Input Validation

Threat Level: HIGH

Our team will implement strict input validation for all user inputs. We will validate and sanitize user inputs to prevent malicious input from causing security vulnerabilities.

3.8 Third-Party Dependencies

Threat Level: HIGH

Our team will carefully review third-party libraries and components for security vulnerabilities, as these libraries can sometimes be the target for attacks.

CHAPTER 4

Requirements

4.1 Software Requirements

Listed below are the software requirements for the CSLC Tutoring. These requirements are necessary in order to correctly and reliably implement the application. Likewise, these requirements are needed to appropriately develop the codebase.

4.1.1 Integrated Development Environments

For this application, we will use several Integrated Development Environments (IDE's). These include Visual Studio Code for full stack development, Jetbrains WebStorm for front-end development, and Jetbrains Pycharm for back-end development.

4.1.2 Version Control System

For this application, we will use Git as our version control system. Similarly, we will use GitHub to track changes and branches across members. We have chosen to use this Version Control System (VCS) as it is simple to use and easy to keep tabs on.

4.1.3 Languages

For this application, we will use the following languages: Python, JavaScript, HTML, and CSS. We will also use the following frameworks and libraries: Django, React, and Tailwind.

4.1.4 Front-end and UI/UX

Our front-end will be written using the React library. Likewise, we will use Tailwind CSS to develop elegant user interfaces and components.

4.1.5 Back-end and Database

Our back-end will be written using the Django ORM. We will treat it as a REST back-end by using the Django REST framework. This will allow us to populate the webpage with real time database information via API calls. We will use Postman to test our API, and we will use sqllite as our database.

4.1.6 Deployment

We will deploy our application using docker for containerization, Kubernetes for automating deployment/management of containerized applications, and we will host our application on Azure.

4.1.7 Documentation

We will use a combination of Sphinx and Autodoc for our technical documentation. We will also use LaTeX to write our documentation (such as this document you are reading).

4.1.8 Management

We will use Jira and Trello for managing tasks and tracking project progress. We will communicate through a designated discord server, and use Google as our file distribution network.

CHAPTER 5

Work Breakdown Structure

5.1 Role Assignment

Role Assignment	
Ticket Management System	Nolan Gregory
User Roles and Authentication	Nolan Gregory & Lindsey Langdon
Tutor and Course Management	Nolan Gregory
Communication and Feedback	Lindsey Langdon & Mya Bell
Analytics and Database Architecture	Nolan Gregory
UI/UX Design	Lindsey Langdon
Front-End Implementation	Lindsey Langdon
Back-End Implementation	Nolan Gregory
Testing and Quality Assurance	Mya Bell & Austin Bailey
AGILE Management	Mya Bell
Cybersecurity Analyst	Austin Bailey
Documentation	Austin Bailey
Deployment and Infrastructure	Nolan Gregory

5.2 Project Initiation

- Define project scope and objectives
- Identify stakeholders and establish communication plan
- Create project plan and timeline

5.3 Requirements Gathering and Analysis

- Define user stories and use cases
- Gather functional and non-functional requirements
- Conduct interviews with stakeholders to gather detailed requirements

5.4 System Design

- High-level architecture design
- Database schema design
- UI/UX wireframing and design
- Design API endpoints and data flow

5.5 Front-End Development

- Set up project structure and version control
- Develop user registration and login components
- Create user dashboard for students, tutors, and admins

- Implement ticket creation and management UI
- Develop UI for communication features (messaging, feedback)

5.6 Back-End Development

- Set up server environment and framework
- Develop authentication and authorization logic
- Create API endpoints for user management
- Implement ticket management and linking with tutors, courses, and students
- Develop tutor and course management functionalities
- Integrate messaging and communication features

5.7 Database Development

- Set up database server
- Create tables for users, tickets, courses, messages, etc.
- Implement relationships between database entities
- Establish data integrity constraints and indexes

5.8 Testing and Quality Assurance

- Develop unit tests for individual components
- Conduct integration testing of front-end and back-end
- Perform user acceptance testing (UAT)
- Identify and fix bugs and issues

5.9 Security and Performance Optimization

- Implement encryption for sensitive data
- Conduct security testing and vulnerability assessment
- Optimize database queries for performance
- Implement caching mechanisms for improved responsiveness

5.10 Documentation

- Create user documentation for students, tutors, and admins
- Document API endpoints and usage
- Prepare deployment and setup guides
- Compile project documentation for future reference

5.11 Deployment and Launch

- Prepare production environment
- Deploy application to hosting server or cloud platform
- Configure domain and SSL certificates
- Perform final testing in the production environment

5.12 Project Closure

- Perform final project review
- Document lessons learned and best practices
- Hand over project to maintenance and support team

CHAPTER 6

Project Schedule

6.1 Milestone One

Time to complete: *Four Weeks*

1. **Project Initiation**
   - Define project scope and objectives
   - Identify stakeholders and establish communication plan
   - Design and develop project plan and timeline
2. **Project Planning**
   - Define user stories and use cases
   - Gather detailed functional and non-functional requirements
   - Conduct product owner meetings for clarification
3. **Project Setup**
   - Setup VCS
   - Establish environments

6.2 Milestone Two

Time to complete: *Three Weeks*

1. **Architecture Design**
   - Develop high-level architecture design
   - Create database schema design
   - Define API endpoints and data flow
   - Create API endpoints for user management
2. **Develop Database Schema**
   - Develop database model diagram
   - Write Database models
   - Develop authentication and authorization logic
3. **User-Interface Design and Setup**
   - Design UI/UX wireframes and layouts
   - Develop UI for communication features
   - Implement ticket creation and management UI
   - Develop user registration and login components
   - Create user dashboard for students, tutors, and admins
4. **Application Logic Design**
   - Develop authentication and authorization logic
   - Implement ticket management and linking with tutors, courses, and students
   - Develop tutor and course management functionalities
   - Integrate messaging and communication features

6.3 Milestone Three

Time to Complete: *Three Weeks*

1. **Testing**
   - Develop unit tests for components
   - Conduct integration testing of front-end and back-end
   - Perform user acceptance testing
   - Create API endpoints for user management
2. **Rudimentary Optimization**
   - Identify and fix bugs and issues
   - Optimize database queries for performance [potential]
   - Implement caching mechanisms for responsiveness
3. **Security Perspectives**
   - Conduct security testing and vulnerability assessment
   - Implement encryption for sensitive data

6.4 Milestone Four

Time to complete: *Two Weeks*

1. **Wrap-up Documentation**
   - Finalize user documentation for students, tutors, and admins
   - Finalize documentation of API endpoints and usage
   - Design deployment and setup guides
   - Perform final testing in the production environment
2. **Continuous Delivery and Integration**
   - Finalize the CI/CD pipeline
   - Prepare production environment
   - Configure domain and SSL certificates
   - Deploy application to hosting server or cloud platform
3. **Security Perspectives**
   - Conduct security testing and vulnerability assessment
   - Implement encryption for sensitive data

6.5 Wrapping Up

Time to complete: < *1 Week*

1. **KT on Prod**
   - Conduct training sessions for tutors
   - Conduct training sessions for product owner
   - Conduct training sessions for application maintainers
2. **Final Loose Ends**
   - Document lessons learned and best practices
   - Hand over project to owner

CHAPTER 7

Monitoring and Reporting Mechanisms

7.1 Project Management Tools

We will use Jira to create tasks, assign responsibilities, set due dates, and track progress. By using Jira, we will be able to manage tasks, prioritize work, and monitor the project status.

7.2 Version Control and Collaboration

We will use a version control system (Git) with GitHub to manage code changes, collaborate on code, and track contributions.

7.3 Meetings and Client Communication

We will schedule regular team meetings to discuss progress, challenges, and upcoming tasks. Likewise, we will conduct daily stand-up meetings for brief updates and address any "blockers".

7.4 Document Sharing

We will use Google Drive and Confluence to share project-related documents, reports, and documentation. This will ensure that everyone has access to the latest information.

7.5 Code Reviews, Testing, and QA

We will implement a code review process using GitHub's pull request feature. We will also set up a testing environment and conduct regular testing and quality assurance activities. By setting up our GitHub integration testing, we can ensure each push is valid and will not break the production environment.