Capstone Final Report

Computer Science Learning Center Tutor Portal

Cringe Coders Team

Austin Bailey
Mya Bell
Lindsey Langdon
Nolan Gregory

Semester Project Final Report for the
Computer Science Learning Center Website Project

Dr. Harvey Siy; College IS&T
University of Nebraska - Omaha

Capstone Course
December 6th, 2023

Report prepared in fulfillment of CSCI 4970: Computer Science Capstone Project

Contents

Chapter 1

Introduction

1.1 Motivation

The Computer Science Learning Center (CSLC), is a student service hosted through the College of Information Science & Technology (IS&T) for academic assistance and tutoring. The CSLC accepts tutoring by appointment or walk-ins Monday through Saturdays, and distance tutoring over Zoom. Student-tutors are paid for their services, and the University has administrators to oversee and manage CSLC staffing levels, work shifts, and payroll.

The existing CSLC portal has incomplete or missing features, functionality, and services. The current ticket submission system functions through a Google Form tied to a spreadsheet with the resulting student information.

To better meet the needs of the CSLC, the Cringe Coders Team proposes a full stack refresh of the existing technology stack. The ticketing system at present will be rebuilt to provide tutors and students with a more efficient way for exchanging essential information. This encompasses sharing specifics about the class assistance needed, articulating the present comprehension of the problem, identifying the respective course instructor, and the level of priority. The new portal will be able to facilitate this seamless sharing of this information between tutors and students.

1.2 Similar Applications

Last semester's capstone course had a group lay the foundations of the project work, including a basic website with some more advanced functionality like authentication. While a CSLC tutoring portal exists, the current solution does not have the full suite of functionality that the client desires. This includes a ticketing for students to request tutoring, dynamic web page elements to show new tickets without the need to refresh the page, UI/UX considerations, and report generation for tracking ticket trends.

Core elements of infrastructure were established with the prior group, but some supplemental features were given work-around solutions. Case and point is the ticketing and reporting functionality. The current implementation is to use Google Forms to submit tickets, which allows for a limited (alibeit incorrectly and partially formatted) report.

In light of the limitations of current solutions, the Cringe Coders Team takes this as a motivation to overhaul the UI/UX and add further back-end functionality, as is highlighted in our Motivation section via a proposed solution outlined in Chapter 3: Architecture and Design.

The Cringe Coders Team's current conception of such a project would take inspiration from ticket management solutions, such as ServiceNow. ServiceNow provides ticketing for requests, incidents, and miscellaneous processes that allows an organization a central way of tracking and administrating business processes. More in line with Cringe Coders Team's focus would be a solution like `https://www.go-redrock.com/solutions/tutortrac/`

1.3 Theoretical Foundations of the Project

The back-end database structure established by the prior group will likely need modification and updates, and thus necessitates additional modeling and design of structured tables and potentially databases (Connolly & Begg, 2014). Additionally, the data needs to be accessible via REST API, such that Django REST can make API calls to the database, for both read and write operations (Fielding, 2000).

Though the CSLC is labeled 'Computer Science', the CSLC services all IS&T majors and students. Consequently, security practices should be implemented in the development and deployment of a CSLC portal. This includes protecting against common security flaws, such as Cross-Site Scriting (XSS), and Cross-Site Request Forgery (CSRF) (OWASP, 2021). Much of this security will derive from proper implementation of front-end developments through proper project management of Django (Django Project, 2021).

1.4 Software Engineering Challenges

We anticipate only trivial computer science challenges will arise in the scope of work. The only challenge revolves around the synchronization of the front-end and back-end for real-time updates. Successfully establishing seamless communication between these elements, alongside the portal's relational sqllite database, is mentioned because the implementation is novel to the Cringe Coders Team.

This in not to suggest we anticipate no challenges. The success of this project hinges on everyone's familiarity with the technology stack in use. For some team members, and potentially he entire team, this means dedicating time to become acquainted with new frameworks and programming languages. The capacity to acquire new skills is essential in the world of software engineering. Also, given that not all team members have worked together before, getting an understanding on each other's work processes presents an additional challenge. In essence, the success of this project not only hinges on the team's technical expertise but also equally relies on the team's collaboration and communication.

1.5 System Context

The CSLC portal will occupy the intersection of faculty and student academic interests. The current and proposed solution are designed to facilitate tutoring by, and for, IS&T students. Below Cringe Coders Team outlines the various facets of the system's context.

1.5.1 Subject Facet

Kyle runs the CSLC, which tutors students via a ticketing system. Students request help in a certain subject or class, and the CSLC pairs them with an appropriate tutor. The CSLC is the official tutoring center for IS&T. Kyle uses ticket history to run a periodic report, detailing the classes that students struggle with most. Tutors at the center are paid employees and they use the ticket queue to find students they're qualified to help.

The Cringe Coders Team's scope of topics include, but are not limited to, UI/UX, Database design, software engineering, and version controling via Github. We anticipate that various MIS electives and the software engineering course to be a primary source of inspiration and guidance in addressing the broader scope.

1.5.2 Usage Facet

The primary stakeholder is Kyle Reestman, who is the director of the Computer Science Learning Center and therefore oversees all of the operations within. The users of the system will include the

students, student-tutors, teachers, and other staff members associated with the computer science department of the University of Nebraska at Omaha.

It is important to take note of the mental state of students who are using the portal. Most of the time that students are seeking help with school work, they are in a stressed and somewhat agitated state. This is another reason why having an easily usable interface for the tutoring portal is critical.

1.5.3 IT System Facet

Submitting a ticket on the CSLC Portal will be one of the first interactions a student will have with the CSLC. Therefore, it is crucial that this application will seamlessly fit into the already established technological environment. For context, the CSLC is tied to the University of Nebraska-Omaha. As a result, the reconstructed portal will have to conform with UNO's current authentication system and present a polished and professional user interface.

1.5.4 Development Facet

The main development concern with this portal is the familiarity with technology stack. To create a web application up to professional standards, everyone must be well-versed with the languages at hand. The team's overall proficiency with the technology stack will not only enhance the development process but also contribute to the long-term success and sustainability of the project.

1.5.5 Legal & Ethical Facet

We foresee no legal requirements, as the ingestion of data is self-reported by students, and only the tutors and Kyle can see it. Since this project will not integrate with other systems, basic security and architecture protections will suffice. The only possible issue we foresee is FERPA regulations limiting the projects integrating or sharing data with other systems.

Ethically, we believe any value-added aspects, such as 'About Us' or mission and vision statements have been addressed by the last capstone group. Our focus ethically will be around presenting a clean and professional interface for students. We believe that a haphazard or poorly designed interface will reflect poorly on the CSLC and serve to discredit the tutors by showing poor computer science principles on the CSLC website.

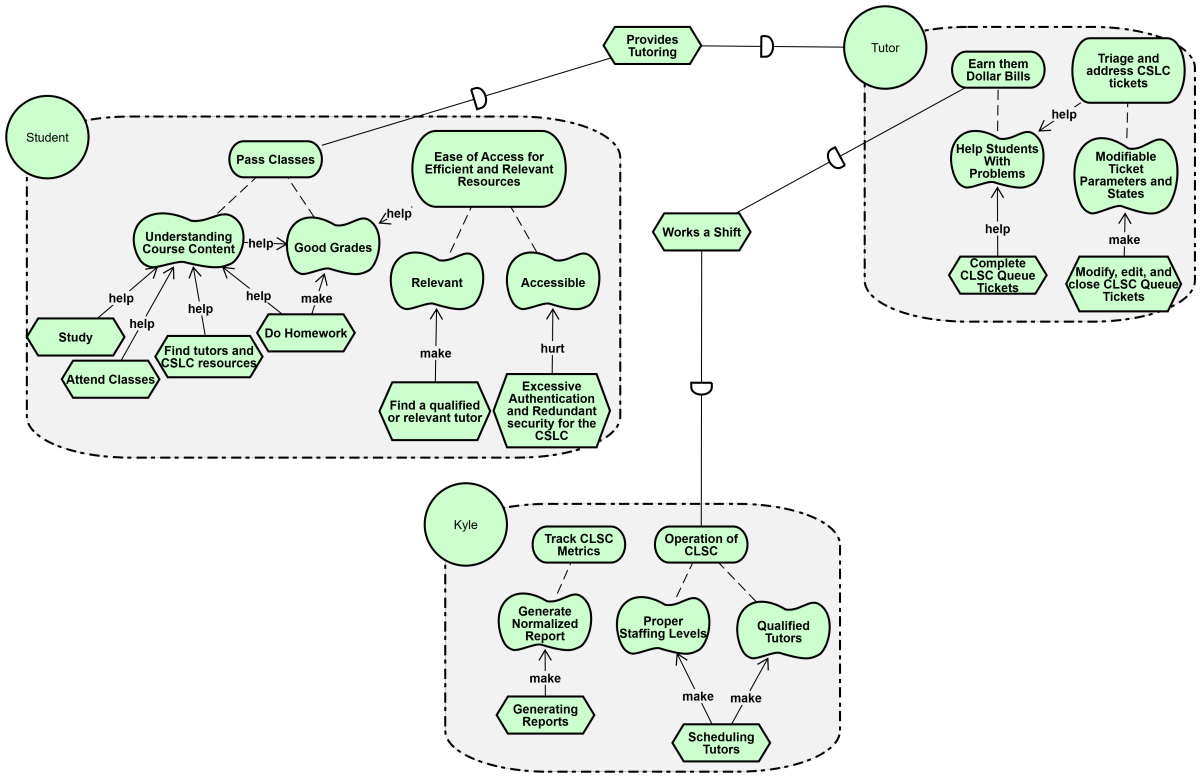
Cringe Coders Team will implement the ticketing system to holistically mirror the prior online version and the in-person experience. As such, ticketing will be on a first-come-first-serve basis, with small caveats for specialty topics or courses, that only specific tutors may be able to cover. In such circumstances, the new version will follow the principles of the old and in-person system, with advanced notice or scheduling requirements. This conforms to the clients specifications and produces an egalitarian system.

Chapter 2

Requirements

2.1 Goals

2.1.1 Goal Model



2.1.2 Goal Descriptions

The Student is one of the users of the system. Each student has an overarching goal of passing their classes, which the CSLC portal is designed to aid in. Students will use the portal as a tool that will connect them with people who will help them to understand the content they need help with.

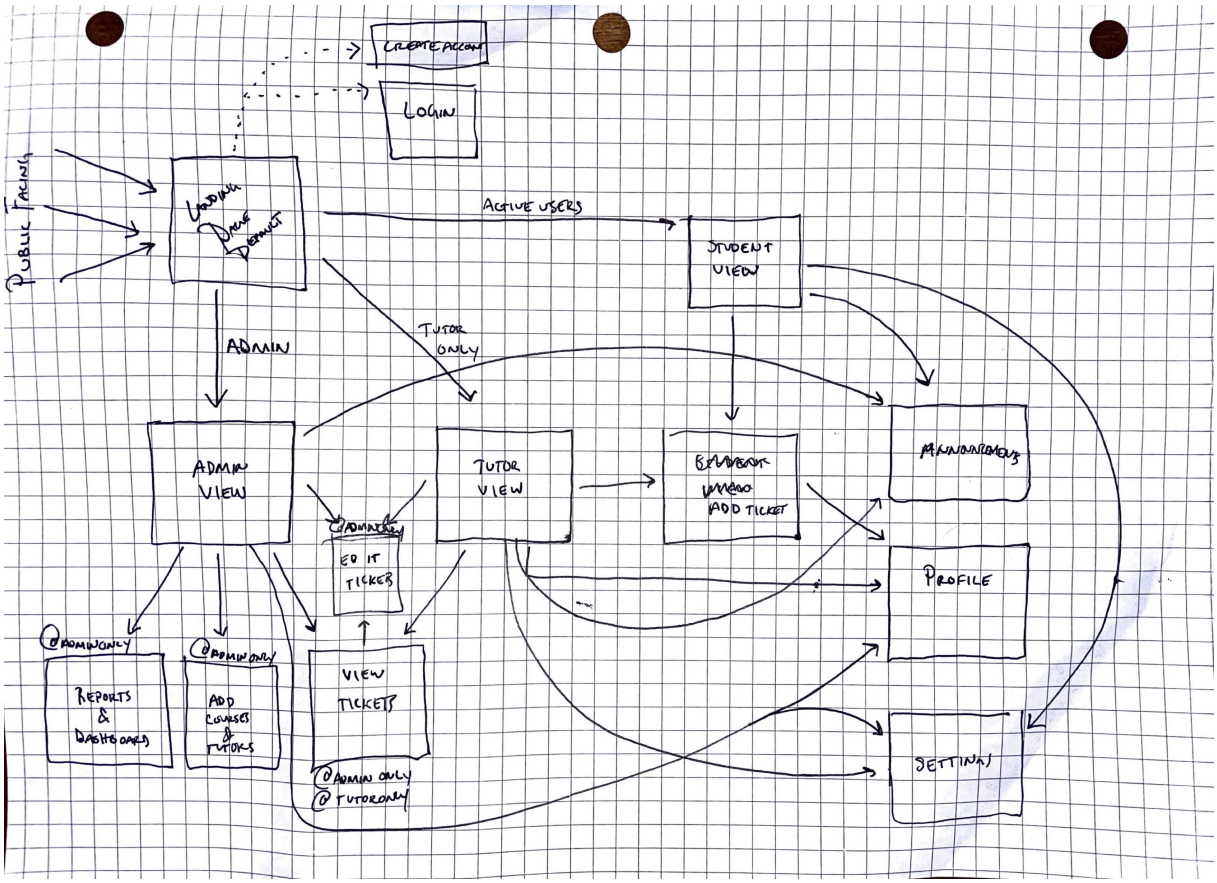
The Tutor is employed by the CSLC to tutor students. One of the main functionalities of the CSLC portal is the ticketing system. This function makes it easy for tutors to plan and organize what they need to do.

Kyle oversees the CSLC, so he has a goal of ensuring the smooth operation of the whole thing. The CSLC portal is designed to aid him in this goal by making it easy to schedule tutors and track what is actually going on.

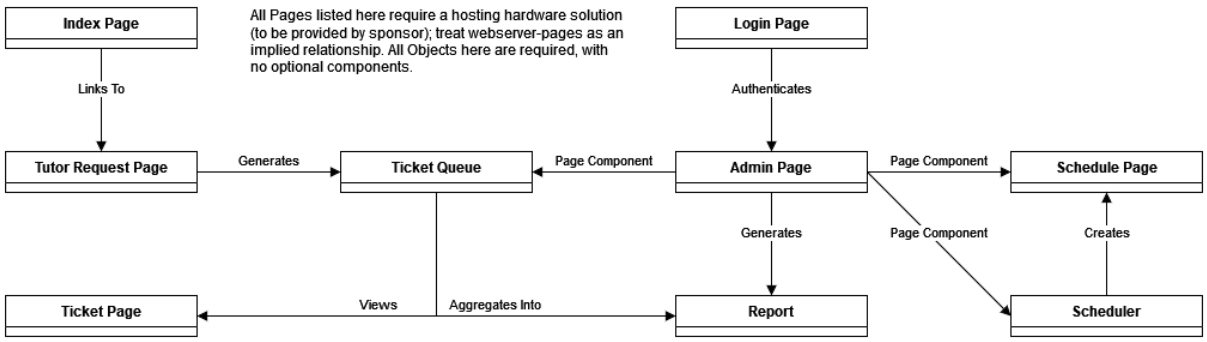
2.2 Functional Requirements

1. Must have. As the director of the Computer Science Learning Center, I want the CSLC portal to have efficient ticket management, so that tutoring requests can be efficiently handled, furthering students' goals of passing classes.
2. Must have. As the director of the Computer Science Learning Center, I want the CSLC portal to have the ability to pull down reports so that I can have accurate metrics concerning the operation of the CSLC.
3. Must have. As the director of the Computer Science Learning Center, I want to be able to easily view graphs and charts illustrating the current status of the CSLC so that I can have additional metrics concerning the CSLC's performance.
4. Must have. As a tutor at the Computer Science Learning Center, I want to be able to interact with students and administrators regarding tutoring requests as well as close and edit tickets, so that I can effectively provide help to students and contribute to the CSLC's operational efficiency while also reaching my financial goals.
5. Must have. As a tutor at the Computer Science Learning Center, I want the ability delegate tutoring requests to my colleagues in situations where I am unable to teach a class so that students can receive the necessary assistance to successfully pass their courses.
6. Must have. As a tutor at the Computer Science Learning Center, I want to see the status of tickets so that I can track and manage the progress of tutoring requests ensuring the smooth operation of the CSLC.
7. Must have. As a student I want to be able to make tutoring requests and see available tutors so that I can easily receive the help I need to pass my classes.
8. Should have. As an administrator, I want to be able to view and manage student requests, so that I can monitor progress and efficiency within the CSLC, thus aligning with the goals of both students and the director for improved learning support and operational effectiveness.
9. Could have. As a student, I want to be able to see how tutors are rated among other students, so that I can make an informed decision about choosing a tutor, contributing to the overall effectiveness and quality of tutoring services within the CSLC.
10. Won't have. As a user of the CSLC portal, I do not want to have to log in and authenticate redundantly, so that ticket submission is time efficient.

2.3 Wireframes



2.4 Object Model



2.5 Nonfunctional Requirements

1. Must have: Evaluating the CSLC Tutoring Portal application on ease of use from the point of view of the students in the context of submitting a ticket for help.
   - Q1: How easy is it to navigate the user interface?
     - M1. Task time
     - M2. Stroke count
     - M3. Problem count
   - Q2: How quickly can tasks be accomplished?
     - Reuse M1, M2, M3

Q3: Do users feel satisfied after using the portal?

M4. Conduct surveys to find the percentage of students reporting high satisfaction after using the portal

2. Should have: Assess the CSLC Tutoring Portal application on maintainability from the point of view of the developers in the context of ensuring that the application is up to date.
   Q1: How maintainable are the languages in the stack?
   M1. Check how often the languages are updated
   Q2: How many dependencies are there?
   M2. Count the dependencies

2.6 Analysis Requirements

Cringe Coders Teamdetermined that we have no analysis requirements.

2.7 External System Interfaces

Our application had to use the CSV provided by the registrar's office. This file was a key component for importing semester-related information into the system. The registrar CSV file required specific attention during the implementation phase. Despite the irregularities in its format, the use of the CSV file format allowed for the structured representation of tabular data, facilitating efficient parsing and manipulation. Pandas, a powerful data manipulation library, was employed to navigate and clean the CSV data, converting it into a format compatible with the application's database schema.

Attached in the appendix is a static copy of the Sphinx auto documentation that clarifies the full API interface, both internal and external. additionally, within the administrative interface, Kyle or another administrator can download a csv of the tickets table.

2.8 Requirements Not Implemented

Cringe Coders Team was able to quickly prototype, develop, and implement our capstone solution. Therefore, we have implemented all requirements, and were able ot address small stretch goals.

Chapter 3

Architecture and Design

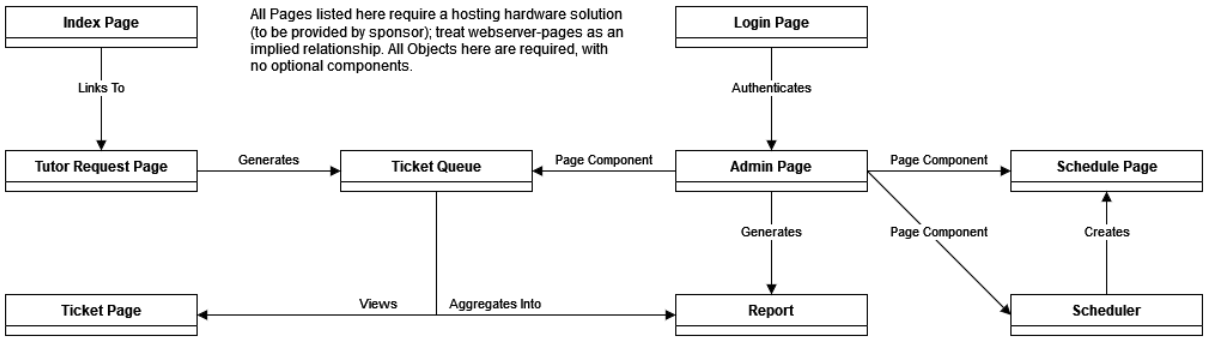
3.1 Overview

The CSLC portal will comprise a handful of web pages hosted on the `https://cslc.unomaha.edu/` subdomain, with integration of a back-end database. at a high-level technology-agnostic view, we have the following design requirements

1. A guest, index, or public page that links to the portals resources.
2. A ticket or tutoring request submission resources.
3. A tutor or employee view that allows access to the ticket queue and shift schedules of tutors.
4. An admin view that allows the client to export reports based on ticket history, and to configure work schedules.
5. A back-end database to facilitate the ticketing processes.

Subsequent sections will cover these designs through the various lens, culminating in a more concrete description of architecture.

3.2 Logical Decomposition View



Cringe Coders Teamprovides a simplified UML diagram of both the public and administrative functions of the CSLC portal. Note that the decomposition has merged the employee and admin pages, as the Cringe Coders Team's current intent is to implement RBAC roles associated with authenticated users that reveal components of the admin page. This will ideally minimize redundant pages and code.

For the sake of simplicity and ease of conveying design, the database subsystem has been omitted, as it will integrate will almost every other subsystem.

to see the Cringe Coders Team's accompanying functional requirements, see the Functional Requirements section.

3.3 Technology Stack

The Cringe Coders Team intends to replace the old CSLC portal with a full refresh of the technology stack. **Docker with nginx as a Proxy Server, a REST API Backend, and a React TypeScript Frontend Stack**

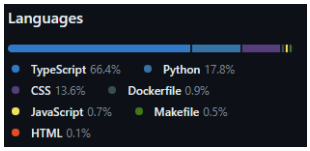
The CSLC Ticket Portal leverages Docker for containerization, allowing for a consistent development, testing, and deployment environment. Nginx is used as a proxy server, directing incoming requests to the appropriate backend services. The REST API serves as the backend, handling data storage, retrieval, and processing. The frontend of the portal is implemented using a React application written in TypeScript.

- **Docker**: The use of Docker allows for encapsulating each component of the application into containers, ensuring consistent environments across different stages of the software development lifecycle. This facilitates seamless deployment and scalability.
- **Nginx as a Proxy Server**: Nginx is employed as a reverse proxy server to handle client requests and route them to the appropriate backend services. This enables load balancing, caching, and SSL/TLS termination, enhancing security and performance.
- **REST API Backend**: The backend of the CSLC Ticket Portal is built as a RESTful API, enabling communication between the frontend and the database. It handles HTTP requests, processes data, and interacts with the database to perform CRUD (Create, Read, Update, Delete) operations.
- **React TypeScript Frontend**: The frontend of the portal is developed using React, a popular JavaScript library for building user interfaces. TypeScript, a superset of JavaScript, is utilized to add static typing to the codebase, thereby enhancing code quality, maintainability, and developer productivity.

Overall, this stack provides a robust, scalable, and maintainable architecture for the CSLC Ticket Portal, leveraging the strengths of containerization, proxy server functionality, RESTful communication, and a type-safe frontend development approach. This plan involves modernizing the conventional front-end, built of HTML and CSS, by transitioning to React. Moreover, the back-end architecture is undergoing a significant overhaul, featuring a custom REST API built with Django. Ultimately, the team aims to enhance the portal capabilities by integrating database functionalities, replacing the existing Google Form solution.

3.3.1 Languages

For this project, we used the following languages: Python, JavaScript, TypeScript, HTML, and CSS. We also had a marginal presence of the Makefile and Dockerfile languages to aid in deployment of the project. The language usage stems primarily from the Django, React, and Tailwind frameworks and libraries.



3.3.2 Libraries & Services

The front-end uses the React library and Tailwind CSS to enable better UI/UX. While our back-end is written using the Django ORM. We treated 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 used Postman to test our API, and we used sqllite as our database. Additionally, we can deploy our application using docker for containerization, Kubernetes for automating deployment/management of containerized applications, and we can host our application on Azure, pending any client instantiation specifications.

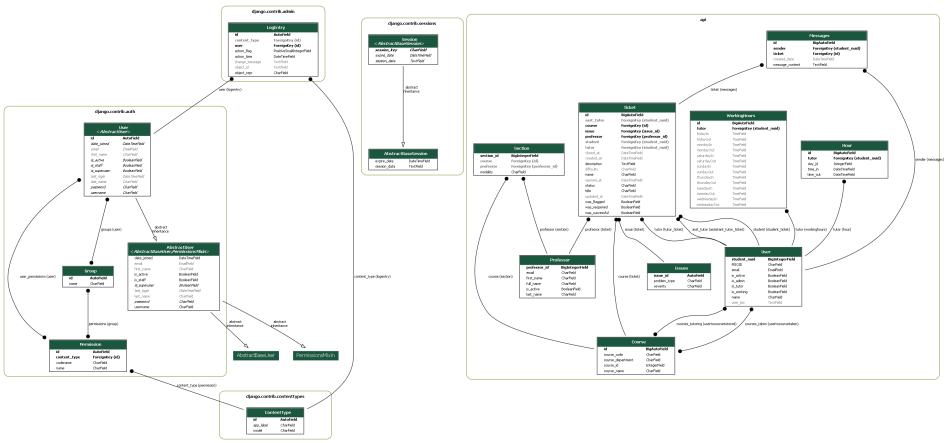


Figure 3.1: Simplified Deployment View Diagram for CSLC Ticket Portal

3.4 Deployment View

The deployment view of the CSLC Ticket Portal application involves the utilization of several components and technologies to ensure efficient deployment and operation.

Docker is used for containerization, providing a consistent environment for the application to run across different systems. This allows for easy deployment and scaling of the application components.

Nginx is employed as the web server to handle incoming HTTP requests and serve static content. It also acts as a reverse proxy to pass requests to the application server.

Ngrok is utilized for creating secure tunnels to localhost, enabling remote access to the application during the development and testing phases.

GitHub workflows are incorporated for automating the CI/CD (Continuous Integration/Continuous Deployment) process. This involves using GitHub Actions to define the steps for building, testing, and deploying the application whenever changes are pushed to the repository.

Continuous integration involves automatically building and testing the application code upon each commit, ensuring that new changes do not introduce errors. Continuous deployment automates the release process, allowing for rapid and reliable delivery of new features and updates to the production environment.

Overall, the use of these technologies and processes facilitates a streamlined deployment pipeline for the CSLC Ticket Portal application, ensuring reliability and ease of maintenance.

Here is an example of a simplified deployment view diagram:

3.5 Development View

Development View

The development view of the CSLC Ticket Portal application encompasses the software and hardware components required to support the development, testing, and deployment of the system. It involves the architecture, development environment, version control, build tools, and testing tools used in the development process.

Architecture: The application follows a client-server architecture, with the front-end implemented using HTML, CSS, and JavaScript, and the back-end developed using Python and Django framework. The database management system used is PostgreSQL.

Development Environment: The development environment includes IDEs such as PyCharm and Visual Studio Code for coding, virtual environments using virtualenv for isolated Python environments, and package managers like pip for Python package installations.

Version Control: The system utilizes Git for version control, with a centralized repository hosted on

GitHub. Branching strategies such as feature branching and pull requests are used for collaborative development.

Build Tools: The application uses Django's built-in management commands for tasks like database migrations, static file collection, and server management. Additionally, tools like Vite were used for front-end asset compilation and bundling.

Django REST API Framework:

The Django REST framework is a powerful and flexible toolkit for building Web APIs. It works by providing a set of tools and libraries for building interactive and feature-rich APIs. It allows for the serialization of data and the creation of API endpoints, making it easier for the front end to communicate with the back end. In the CSLC Ticket Portal application, the Django REST framework was used to create endpoints for the different functionalities such as ticket creation, retrieval, and updating. These endpoints were then accessed by the front end to perform operations like creating new tickets, fetching ticket details, and updating ticket status.

- **Usage in the Application:** The application leveraged the Django REST framework to create API views which were linked to specific URL routes. These API views handled the incoming requests from the front end, processed the data using Django models, and returned the relevant data in a serialized format (e.g., JSON). The front end made REST API calls to these endpoints using standard HTTP methods such as GET, POST, PUT, and DELETE to perform CRUD (Create, Read, Update, Delete) operations on the data stored in the back end.
- **Django ORM:** Django's Object-Relational Mapping (ORM) was used to interact with the application's database. The ORM simplifies the interaction with the database by allowing developers to work with Python classes and objects instead of directly dealing with SQL queries. In the CSLC Ticket Portal application, the Django ORM was used to define models that represented the different entities such as tickets, users, and subjects. These models corresponded to database tables, and the ORM allowed for easy querying, insertion, and modification of data using Python syntax. Overall, the combination of Django REST framework for building APIs and the Django ORM for database access resulted in a robust and efficient architecture for the CSLC Ticket Portal application.

React Typescript in CSLC Ticket Portal

In the CSLC Ticket Portal application, React Typescript was used extensively to create reusable components that facilitate dynamic rendering of the frontend based on the data served by the backend.

- **Component Creation:** React Typescript was utilized to define components with explicit prop types and state types. This ensured clear interfaces for communication between components and allowed for better type checking during development.
- **Reusability:** By utilizing React Typescript, components were designed to be reusable across the application, promoting a consistent UI and reducing code redundancy.
- **Dynamic Rendering:** The use of React Typescript allowed for the creation of components that dynamically render based on the data received from the backend. This enabled the frontend to adapt seamlessly to changes in the backend data, providing a responsive user interface.
- **Type Safety:** With Typescript, the application benefited from static type checking, helping to catch potential errors during development and providing better long-term maintainability.
- **State Management:** React Typescript was used to manage component state, ensuring that the frontend remained in sync with the backend data and providing a seamless user experience.

Overall, the integration of React Typescript in the CSLC Ticket Portal application played a crucial role in building reusable components and enabling dynamic frontend rendering based on the information served by the backend.

Docker Containerization: To containerize the CSLC Ticket Portal application, Docker was utilized to encapsulate the application, its dependencies, and the necessary runtime environment into a portable container. This provided a consistent and predictable environment across different deployment environments.

- Docker's containerization technology allowed for easy packaging of the application, ensuring that it could run consistently on any infrastructure. This eliminated the "it works on my machine"

problem often encountered in software development and deployment.

- Additionally, the use of Docker enabled the creation of a reusable environment, facilitating easier deployment and scaling of the application as it could be run on any system supporting the Docker runtime. This simplifies the setup process, reduces environment-specific issues, and enhances the overall portability of the application.
- By utilizing Docker, the CSLC Ticket Portal application achieved a more efficient and streamlined deployment process, while also ensuring a consistent runtime environment across different deployment scenarios.

Testing Tools: To test the database and REST API endpoints in the CSLC Ticket Portal application, the Django pytest framework was utilized extensively. Through pytest, we were able to create and execute tests for the database models and API endpoints, ensuring their functionality and reliability. For database testing, we wrote pytest functions to validate the creation, modification, and deletion operations on the models. By using fixtures, we could set up the database in a specific state before running the tests, ensuring consistent test results. We employed assertions to verify the expected behavior of the database operations, such as checking for the creation of new records, the modification of existing data, and the proper handling of deletions. When testing the REST API endpoints, we used pytest to simulate HTTP requests and assess the responses from the API. By sending requests using the 'requests' library and inspecting the returned data, we verified that the endpoints were functioning as intended. We asserted the expected status codes, response bodies, and headers to ensure the API's correctness and robustness. Moving to the frontend, Jest was employed for testing and mocking React components. Jest allowed us to create unit tests for individual React components, which involved verifying their rendering, state changes, and interactions. We utilized Jest's mocking capabilities to simulate different scenarios and external dependencies, ensuring that the components behaved as expected under various conditions. By employing Jest's snapshot testing, we could capture the rendered output of components and compare it against previously saved snapshots, detecting unintended changes in UI rendering. This feature provided valuable insight into any unexpected modifications to the UI, promoting consistent and predictable component rendering. Overall, the combination of Django pytest for backend testing and Jest for frontend testing enabled thorough and comprehensive testing of the CSLC Ticket Portal application, ensuring the reliability and quality of both its database operations and user interface components.

Overall, the development view showcases the technological components and workflows utilized in the development of the CSLC Ticket Portal application.

Backend Visualization: The backend structure of the CSLC Ticket Portal involves the utilization of the Django REST framework, which is a powerful and flexible toolkit for building Web APIs. The backend serves as the intermediary between the frontend interface and the database, managing data and performing various operations.

[Frontend Interface] <—> [Django RESTful API Backend] <—> [Database]

The Frontend Interface communicates with the Django RESTful API Backend through HTTP requests and receives JSON responses. The backend processes these requests and interacts with the database to perform operations like retrieving, creating, updating, and deleting data.

The backend consists of several components, including:

1. URLs: Maps the URL patterns to the corresponding views within the application. 2. Views: Handles the request-response cycle, processing incoming requests and returning appropriate HTTP responses. 3. Serializers: Serialize and deserialize complex data types to JSON, allowing data to be converted between complex Python objects and JSON data. 4. Models: Define the data models and interact with the database using Object-Relational Mapping (ORM) provided by Django.

These components work together to handle requests from the frontend, retrieve/store data in the database, and return the necessary responses.

Frontend Visualization: The CSLC Ticket Portal frontend is built using React and consists of several components working together to provide the user interface for ticket management. The components include:

1. App Component: The top-level component responsible for rendering other components and managing application state.

2. TicketList Component: Responsible for displaying a list of available tickets, including details such as the tutor's name, subject, status, and the ability to select a ticket for further actions.

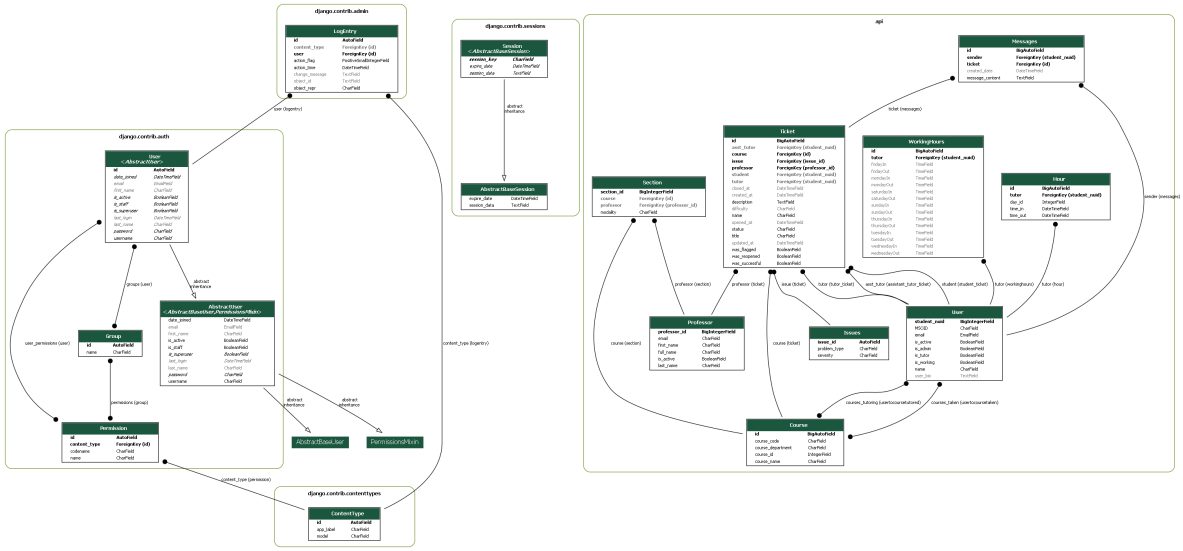
3. TicketDetails Component: Displays detailed information about a selected ticket, including the student's name, issue description, status, and the ability to interact with the ticket (e.g., claim, resolve, close).

4. TicketForm Component: Allows users to create a new ticket by providing information such as the subject, description, and any additional details.

5. Header and Footer Components: Provide navigation links, branding, and other static content for the application.

These components work together to provide a seamless user experience for tutors and students interacting with the ticketing system.

3.6 Data View



Our back-end relational database.

Fields:

- `id - AutoField`
- `action_time - DateTimeField`
- `user - ForeignKey`
- `content_type - ForeignKey`
- `object_id - TextField`
- `object_repr - CharField`
- `action_flag - PositiveSmallIntegerField`
- `change_message - TextField`

Methods (non-private/internal):

- `adelete()`
- `arefresh_from_db()`
- `asave()`
- `get_action_flag_display()`
- `get_admin_url()`

- get_change_message()
- get_constraints()
- get_edited_object()
- get_next_by_action_time()
- get_previous_by_action_time()
- is_addition()
- is_change()
- is_deletion()
- validate_constraints()

Fields:

- usertocoursetutored - ManyToManyRel
- usertocoursetaken - ManyToManyRel
- section - ManyToOneRel
- ticket - ManyToOneRel
- id - BigAutoField
- course_department - CharField
- course_name - CharField
- course_id - IntegerField
- course_code - CharField

Methods (non-private/internal):

- adelete()
- arefresh_from_db()
- asave()
- get_constraints()
- validate_constraints()

Fields:

- logentry - ManyToOneRel
- id - AutoField
- password - CharField
- last_login - DateTimeField
- is_superuser - BooleanField
- username - CharField
- first_name - CharField
- last_name - CharField
- email - EmailField
- is_staff - BooleanField
- is_active - BooleanField
- date_joined - DateTimeField
- groups - ManyToManyField
- user_permissions - ManyToManyField

Methods (non-private/internal):

- adelete()
- arefresh_from_db()
- asave()
- check_password()
- email_user()
- get_all_permissions()
- get_constraints()
- get_email_field_name()
- get_full_name()
- get_group_permissions()
- get_next_by_date_joined()
- get_previous_by_date_joined()
- get_session_auth_fallback_hash()
- get_session_auth_hash()
- get_short_name()
- get_user_permissions()
- get_username()
- has_module_perms()
- has_perm()
- has_perms()
- has_usable_password()
- natural_key()
- normalize_username()
- set_password()
- set_unusable_password()
- username_validator()
- validate_constraints()

contenttypes.ContentType

Fields:

- logentry - ManyToOneRel
- permission - ManyToOneRel
- id - AutoField
- app_label - CharField
- model - CharField

Methods (non-private/internal):

- adelete()
- arefresh_from_db()
- asave()
- get_all_objects_for_this_type()
- get_constraints()

- get_object_for_this_type()
- model_class()
- natural_key()
- validate_constraints()

Fields:

- session_key - CharField
- session_data - TextField
- expire_date - DateTimeField

Methods (non-private/internal):

- adelete()
- arefresh_from_db()
- asave()
- get_constraints()
- get_decoded()
- get_next_by_expire_date()
- get_previous_by_expire_date()
- get_session_store_class()
- validate_constraints()

Fields:

- usertocoursetutored - ManyToManyRel
- usertocoursetaken - ManyToManyRel
- section - ManyToOneRel
- ticket - ManyToOneRel
- id - BigAutoField
- course_department - CharField
- course_name - CharField
- course_id - IntegerField
- course_code - CharField

Methods (non-private/internal):

- adelete()
- arefresh_from_db()
- asave()
- get_constraints()
- validate_constraints()

api.Hour

Fields:

- id - BigAutoField
- tutor - ForeignKey
- day_id - IntegerField
- time_in - DateTimeField

- time_out - DateTimeField

Methods (non-private/internal):

- adelete()
- arefresh_from_db()
- asave()
- get_constraints()
- get_day_id_display()
- get_next_by_time_in()
- get_next_by_time_out()
- get_previous_by_time_in()
- get_previous_by_time_out()
- validate_constraints()

api.Issues

Fields:

- ticket - ManyToOneRel
- issue_id - AutoField
- problem_type - CharField
- severity - CharField

Methods (non-private/internal):

- adelete()
- arefresh_from_db()
- asave()
- get_constraints()
- get_severity_display()
- validate_constraints()

Fields:

- section - ManyToOneRel
- ticket - ManyToOneRel
- first_name - CharField
- last_name - CharField
- full_name - CharField
- email - CharField
- is_active - BooleanField
- professor_id - BigIntegerField

Methods (non-private/internal):

- adelete()
- arefresh_from_db()
- asave()
- get_constraints()
- validate_constraints()

Fields:

- modality - CharField
- professor - ForeignKey
- course - ForeignKey
- section_id - BigIntegerField

Methods (non-private/internal):

- adelete()
- arefresh_from_db()
- asave()
- get_constraints()
- validate_constraints()

Fields:

- messages - ManyToOneRel
- id - BigAutoField
- professor - ForeignKey
- course - ForeignKey
- issue - ForeignKey
- student - ForeignKey
- tutor - ForeignKey
- asst_tutor - ForeignKey
- name - CharField
- title - CharField
- description - TextField
- status - CharField
- created_at - DateTimeField
- opened_at - DateTimeField
- updated_at - DateTimeField
- closed_at - DateTimeField
- was_successful - BooleanField
- was_reopened - BooleanField
- was_flagged - BooleanField
- difficulty - CharField

Methods (non-private/internal):

- adelete()
- arefresh_from_db()
- asave()
- get_constraints()
- get_difficulty_display()
- get_next_by_created_at()
- get_next_by_updated_at()
- get_previous_by_created_at()

- get_previous_by_updated_at()
- get_status_display()
- validate_constraints()

Fields:

- hour - ManyToOneRel
- messages - ManyToOneRel
- student_ticket - ManyToOneRel
- tutor_ticket - ManyToOneRel
- assistant_tutor_ticket - ManyToOneRel
- workinghours - ManyToOneRel
- student_nuid - BigIntegerField
- name - CharField
- is_active - BooleanField
- is_working - BooleanField
- is_tutor - BooleanField
- is_admin - BooleanField
- user_bio - TextField
- email - EmailField
- MSOID - CharField
- courses_tutoring - ManyToManyField
- courses_taken - ManyToManyField

Methods (non-private/internal):

- adelete()
- arefresh_from_db()
- asave()
- get_constraints()
- validate_constraints()

Fields:

- id - BigAutoField
- tutor - ForeignKey
- mondayIn - TimeField
- mondayOut - TimeField
- tuesdayIn - TimeField
- tuesdayOut - TimeField
- wednesdayIn - TimeField
- wednesdayOut - TimeField
- thursdayIn - TimeField
- thursdayOut - TimeField
- fridayIn - TimeField
- fridayOut - TimeField
- saturdayIn - TimeField

- `saturdayOut - TimeField`
- `sundayIn - TimeField`
- `sundayOut - TimeField`

3.7 Concurrency View

Cringe Coders Team does not anticipate work that would involve this section. we include this section as a placeholder and an acknowledgement that future work may re-scope us into this area.

3.8 Execution Flow View

The execution flow for the CSLC Ticket Portal application can be described as follows:

1. Client makes a request to the frontend, which is built using React. This request could be for viewing available tutoring sessions, submitting a ticket, or any other functionality provided by the application.

2. React frontend communicates with the backend API, which is built using Django Rest Framework (DRF), to fetch or submit data related to the request. This communication is typically done using AJAX requests.

3. The requests from the React frontend are received by the DRF backend, which processes the requests, interacts with the database if necessary, and sends back the appropriate response to the frontend.

4. If the application is deployed using Docker, the requests may pass through a Docker container that hosts the Django backend, providing isolation and resource management for the backend application.

5. Nginx, a web server, may be used as a reverse proxy to handle and route incoming requests to the appropriate Docker container hosting the Django backend. Nginx can also serve static files and provide SSL termination.

6. The processed responses from the DRF backend are sent back to the React frontend, which then renders the data and presents it to the user.

This flow demonstrates how the client interacts with the frontend, which in turn communicates with the backend built using Django Rest Framework. Docker is used for containerization, and Nginx is employed as a reverse proxy for routing and handling incoming requests.

3.9 Screenshots

Below are several static captures of the website prior to deployment on UNO architecture. Some placeholder values and text are used.

3.10 Summary of Design Changes

Throughout the iterative process of crafting this application, the user interface has consistently adhered to a clean and minimalistic design philosophy. The commitment to simplicity and clarity in the interface has been a guiding principle, ensuring that users can navigate the platform intuitively and without unnecessary visual distractions. In addition to this commitment to minimalism, various features have been incorporated to enhance the overall aesthetic and usability of the application. Among these enhancements, the light and dark mode stands out as a significant design feature. Users now have the flexibility to choose between a light mode and a dark mode, allowing them to personalize their experience based on preferences or situational needs.

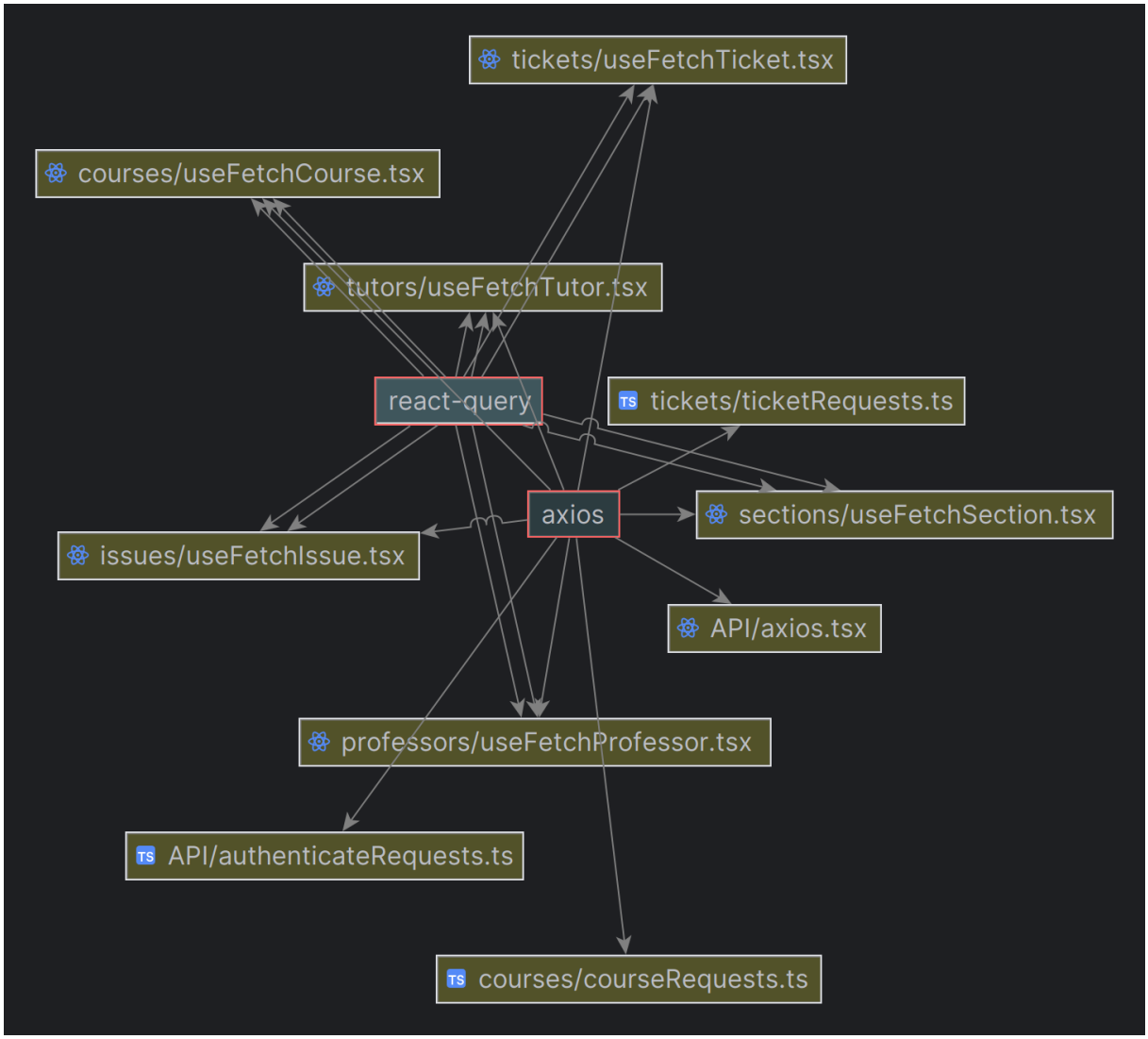


Figure 3.2: UML diagram of frontend API generated with IntelliJ

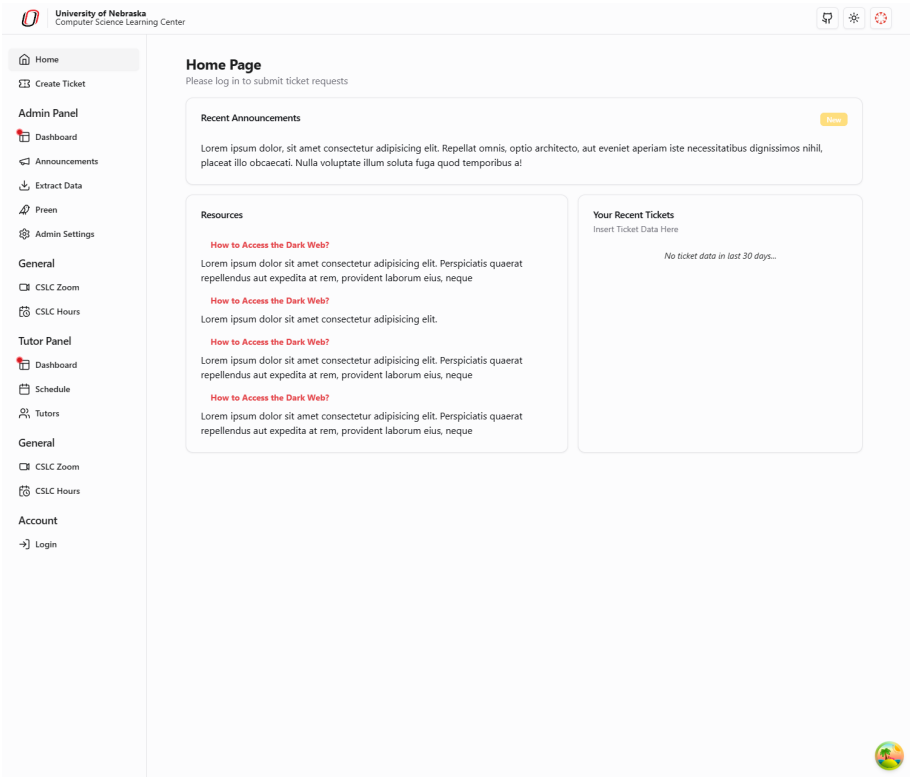


Figure 3.3: CSLC Home Page

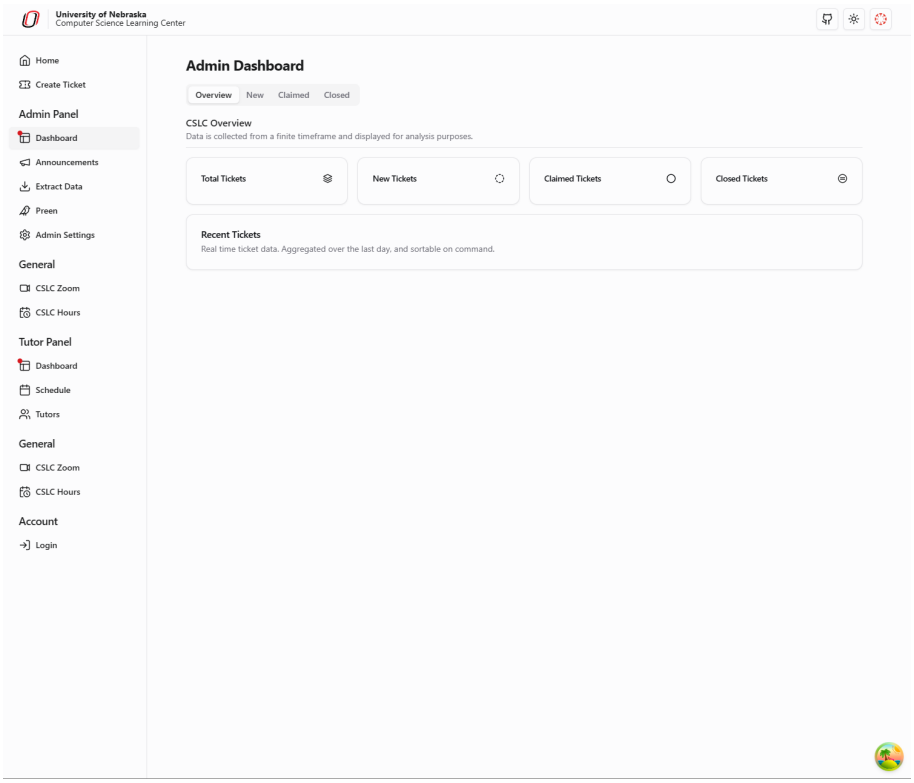


Figure 3.4: CSLC Dashboard

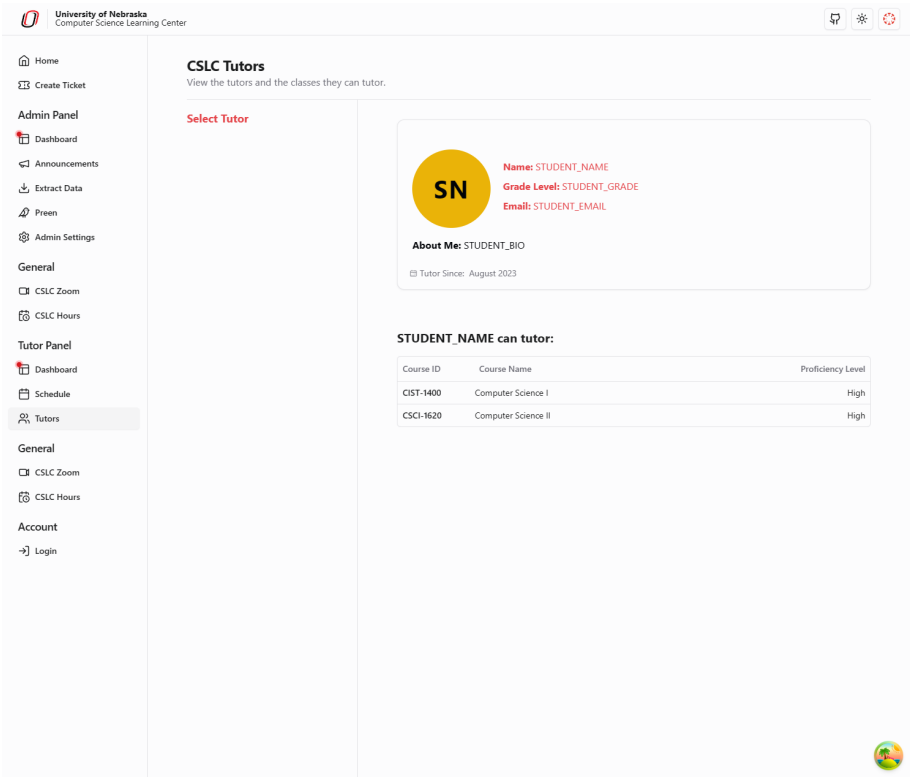


Figure 3.5: CSLC Tutor Page

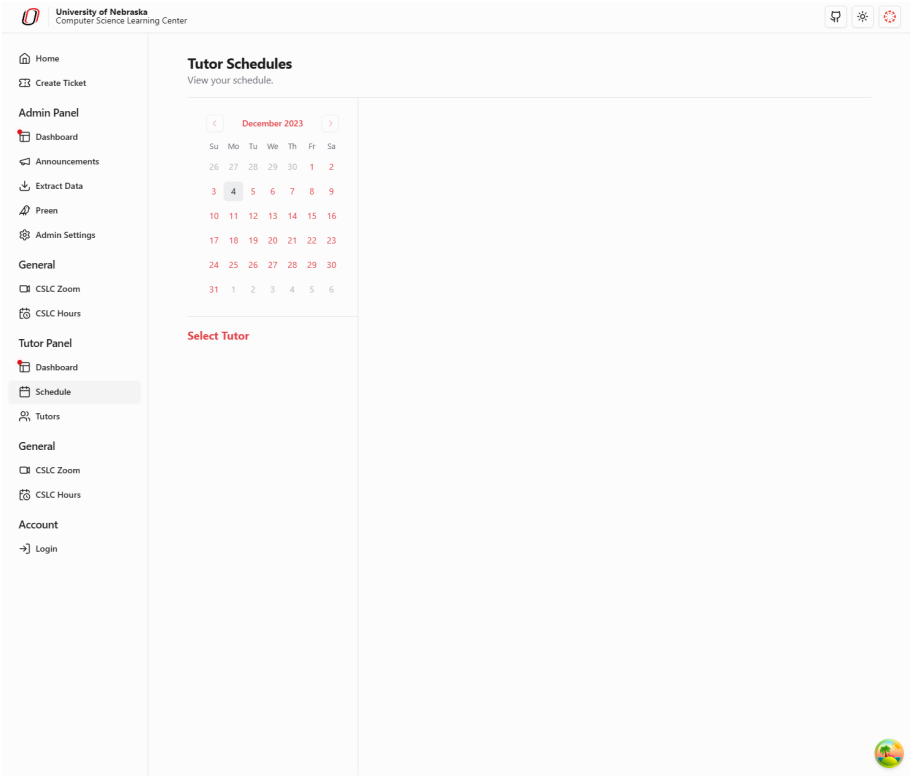


Figure 3.6: CSLC Schedule Page

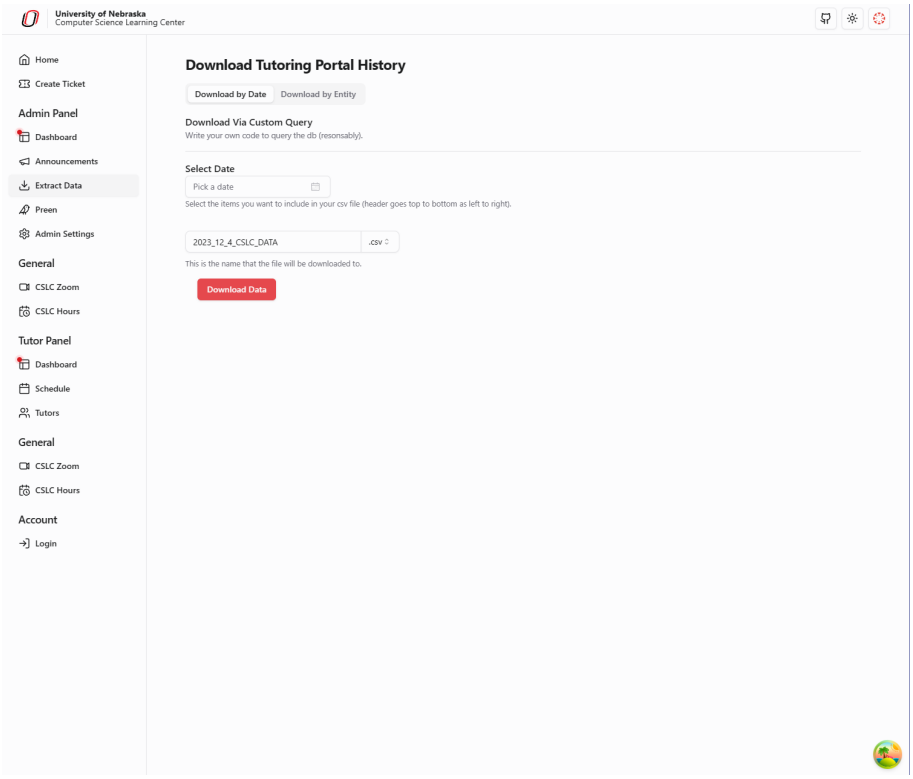


Figure 3.7: CSLC Download Page

3.11 Design Modularity and Extensibility

A stretch goal Cringe Coders Teamslated for work (which was accomplished within the scope of our work) was an integration with the registrar's office to automatically populate current IS&T courses. Both the client and the Cringe Coders Team discussed this in our client meetings. While IS&T has a table of courses, the upcoming changes in structure and shifting semester structure made this semester an inopportune time to tackle the issue.

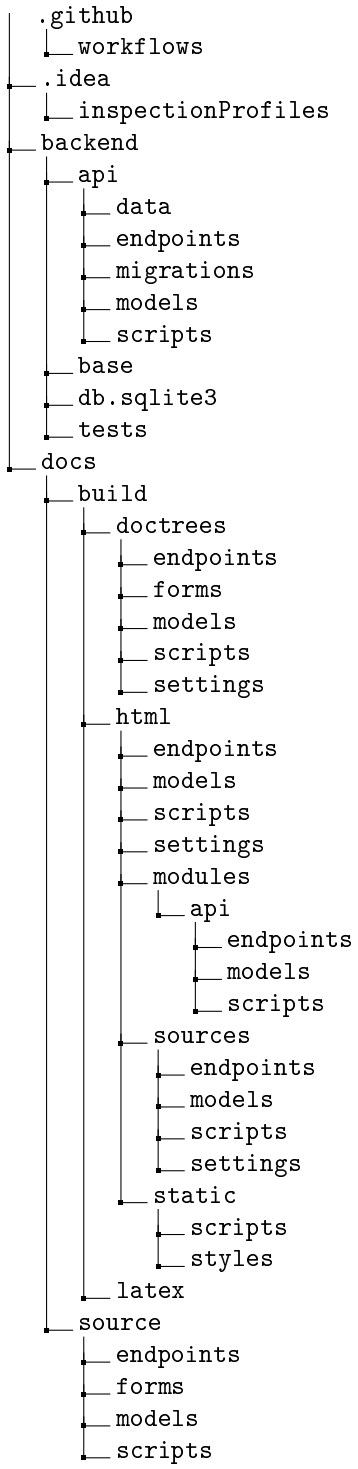
Future work would involve scoping and understanding the data's formatting requirements. A likely implementation would be an ingestion of a csv file through the admin portal. Via REST API, the csv data would populate a table in the back-end that would populate elements such as drop down menus for selecting a course on tutoring tickets.

Acknowledging that this potential work is a likely future endeavor, we have attempted to structure our database around semesters markers. However as Kyle clarified, changes to back-end nomenclature on semesters may make database refactoring a necessity, despite our best effort. Any future implementations would likely involve re-keying the database or potentially archiving historic data and repopulating with a new schema and data.

Chapter 4

Implementation

4.1 Directory Structure



```
    └── settings
├── frontend
│   ├── public
│   └── src
│       ├── API
│       │   ├── courses
│       │   ├── issues
│       │   ├── professors
│       │   ├── sections
│       │   ├── tickets
│       │   └── tutors
│       ├── components
│       │   ├── assets
│       │   ├── display
│       │   ├── fields
│       │   ├── forms
│       │   ├── navigation
│       │   ├── tables
│       │   │   ├── admin
│       │   │   └── data
│       │   ├── tickets
│       │   ├── typography
│       │   └── ui
│       ├── forms
│       ├── lib
│       ├── routes
│       ├── style
│       ├── tests
│       ├── types
│       └── views
│           ├── admin
│           ├── development
│           └── tutor
└── nginx
```

4.2 Technical Issues Encountered

The first technical issue emerged was when the codebase was being transitioned from Javascript to Typescript. Managing this transition proved to be a complex task, especially given the simultaneous learning curve associated with React and TailwindCSS. There were multiple errors in the TypeScript conversion process when defining the appropriate types for the components.

Challenges arose during the Hot Module Reload (HMR) integration due to the use of Docker, requiring the establishment of a linkage between the local directory and Docker to enable HMR functionality. This integration provided a lot of technical issues, but it was essential for real-time updates when saving changes.

In addition to the HMR challenge, there were technical issues in cleaning and processing data from the CSV file provided by the registrar's office. The CSV file was described as "malformed," leading to CSV lint warnings and necessitating data cleaning efforts. Despite the file's imperfections, Nolan successfully employed pandas to process and integrate the semester data into the database.

The back end architecture posed its own set of challenges. With an extensive application comprising over 200 files, integrating various components and ensuring seamless interaction between the back end and front end proved to be an obstacle. Each file and module had to work cohesively. As the number of files increased, maintaining clarity and consistency in the code base became crucial.

One of the last obstacles faced pertains to the hosting of the application on a different server.

The existing server, utilizing Apache, posed uncertainties regarding resource sufficiency for hosting the Docker container. The team was confronted with the task of assessing whether the server has adequate resources to accommodate the sizable Docker container and ensuring a seamless transition to a new server configured with nginx as a proxy.

4.3 User-Reported Bugs

Due to scheduling conflicts with UNO's hosting subject matter experts, Cringe Coders Team has been unable to deploy the service on UNO's architecture. Consequently, wide-scale user testing has not been conducted. In lieu of this, we have done extensive unit testing.

Chapter 5

Testing

5.1 Test Plan

This test plan outlines the high level testing strategy for the CSLC Tutoring Portal. The primary objective of this testing plan is to ensure the robustness, functionality, and performance of the web application. This plan includes both integration testing and end-to-end testing, enabling our team to deliver a high performing web application that will meet the demands of the end users.

In the integration testing scenario, we need to test the API integration and the component integration. With testing the API integration, we need to verify that the front-end properly communicates with the REST API back-end by testing API calls and responses for different endpoints. For component integration testing, Jest can be employed to ensure that the different web components of the application work together as expected. As Jest is primarily known for its unit testing, we plan to develop unit tests for our components to help identify potential bugs during the testing phase.

The end-to-end testing approach in this plan is designed to comprehensively evaluate the users' journey from start to finish. We plan on using Selenium to stimulate users actions such as clicks and inputs to asses the application's performance. Additionally, Selenium can be used for cross-browser compatibility testing to ensure that our application works across a variety of browsers. Similar to the integration testing, the end-to-end testing process is focused on identifying any potential bugs during the testing phase, ensuring the delivery of a dependable and user-friendly application.

5.2 Obtaining Realistic Test Data

In the test cases, we incorporated placeholder values for attributes such as course names, issue types, professor details, and other relevant data points. These values were chosen based off of typical scenarios that would be encountered. While the data may not reflect actual information from a live system, the tests provide a controlled and reproducible environment for testing, ensuring that the test cases are both realistic and capable of assessing the application's functionality under various circumstances.

5.3 Tests Conducted

The tests conducted used the Pytest framework to assess the functionality of key components. This framework was used in combination with Django 'Client' to stimulate HTTP requests and assess the application's behavior. For instance, certain test scenarios involved the generation and retrieval of objects associated with courses and issues. Furthermore, tests covered professor-related functionalities, such as validating the successful creation of a new professor and retrieving professor objects based on various query parameters. These tests systematically examine object creation, retrieval, and the impact of various query parameters, contributing to a comprehensive evaluation of the application's RESTful API. Additionally, the suite incorporates model-specific tests for the 'Course' and 'Professor' models, scrutinizing attributes, default values, and string representations. This testing approach ensures that essential features align with expectations.

5.4 Test Results

The absence of errors in the Pytest results suggests that the application successfully passed the automated tests. While this presentation indicates a positive outcome, it's essential to note that these results are simulated. In reality, the testing process involves thorough validation of various functionalities, corner cases, and integration scenarios. however, the ability to showcase simulated successful Pytest results is indicative that the team is on the right track.

5.5 Automated Test Outputs

Below is some of the Pytest results. Due to the number of tests conducted during the ongoing testing phase for production, only a subset of the outputs is displayed here. Note, the application is currently still in active testing.

```
============================================== test session starts ================
platform win32 -- Python 3.10.1, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\linds\University-Nebraska-Tutor-Portal\backend, configfile: pytest.ini
plugins: anyio-4.0.0
collected 8 items / 2 errors

==================================================== ERRORS ======================
_____________________________________ ERROR collecting tests/test_course.py ___________
ImportError while importing test module 'C:\Users\linds\University-Nebraska-Tutor-Portal\bacl
Hint: make sure your test modules/packages have valid Python names.
Traceback:
..\..\..\AppData\Local\Programs\Python\Python310\lib\importlib\__init__.py:126: in import_moc
    return _bootstrap._gcd_import(name[level:], package, level)
test_course.py:3: in <module>
    from api.models.course import Course
E   ModuleNotFoundError: No module named 'api'
_____________________________________ ERROR collecting tests/test_professor.py __________
ImportError while importing test module 'C:\Users\linds\University-Nebraska-Tutor-Portal\bacl
Hint: make sure your test modules/packages have valid Python names.
Traceback:
..\..\..\AppData\Local\Programs\Python\Python310\lib\importlib\__init__.py:126: in import_moc
    return _bootstrap._gcd_import(name[level:], package, level)
test_professor.py:3: in <module>
    from api.models.professor import Professor
E   ModuleNotFoundError: No module named 'api'
============================================== warnings summary ================
..\..\..\AppData\Local\Programs\Python\Python310\lib\site-packages\_pytest\config\__init__.py
  C:\Users\linds\AppData\Local\Programs\Python\Python310\lib\site-packages\_pytest\config\__

    self._warn_or_fail_if_strict(f"Unknown config option: {key}\n")

..\..\..\AppData\Local\Programs\Python\Python310\lib\site-packages\_pytest\config\__init__.py
  C:\Users\linds\AppData\Local\Programs\Python\Python310\lib\site-packages\_pytest\config\__

    self._warn_or_fail_if_strict(f"Unknown config option: {key}\n")

test_api_course.py:16
  C:\Users\linds\University-Nebraska-Tutor-Portal\backend\tests\test_api_course.py:16: Pytes
    @pytest.mark.django_db

test_api_issue.py:17
  C:\Users\linds\University-Nebraska-Tutor-Portal\backend\tests\test_api_issue.py:17: PytestU
```

```
    @pytest.mark.django_db

test_api_professor.py:18
  C:\Users\linds\University-Nebraska-Tutor-Portal\backend\tests\test_api_professor.py:18: Py
    @pytest.mark.django_db

test_api_professor.py:25
  C:\Users\linds\University-Nebraska-Tutor-Portal\backend\tests\test_api_professor.py:25: Py
    @pytest.mark.django_db

test_api_professor.py:39
  C:\Users\linds\University-Nebraska-Tutor-Portal\backend\tests\test_api_professor.py:39: Py
    @pytest.mark.django_db

test_api_professor.py:47
  C:\Users\linds\University-Nebraska-Tutor-Portal\backend\tests\test_api_professor.py:47: Py
    @pytest.mark.django_db

test_api_professor.py:55
  C:\Users\linds\University-Nebraska-Tutor-Portal\backend\tests\test_api_professor.py:55: Py
    @pytest.mark.django_db

test_api_professor.py:71
  C:\Users\linds\University-Nebraska-Tutor-Portal\backend\tests\test_api_professor.py:71: Py
    @pytest.mark.django_db

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
=============================================== short test summary info ================
Successfully executed 6 tests in 0.82s
```

5.6 Tests Not Conducted

Despite the comprehensive testing strategy outlined in the CSLC Tutoring Portal test plan, certain tests were unable to be executed. Firstly, the intended integration tests with a third-party API were hindered by temporary access restrictions, preventing the assessment of real-time data integration. Additionally, the cross-browser compatibility testing using Selenium was also not achievable. However, we did perform some user-testing with the tutors to see if it would be compatible on different browsers. Despite these challenges, the conducted tests using Pytest provided valuable insights into the application's functionality, serving as a foundation for addressing these constraints in future testing cycles.

Chapter 6

Summary

6.1 Summary of Project Organization

Cringe Coders Teamlargely upheld the initially outlined roles, as our record-keeping reflects.Throughout the course of Work, Nolan maintained his roles as the architect, technical lead, back-end developer and dev-ops and database developer. Nolan was continually the lead developer, working with Lindsey as the UI/UX and front-end & mobile lead to develop the code base. Mya and Austin worked as client liaison and quality assurance testers. Mya oversaw large portion of the requirements chapter as the requirements analyst.

6.2 Outcome of Risks

Our largest risk associated the the Cringe Coders Team's project was the quick uptake of technologies and frameworks. Both Lindsey and Nolan work as professional developers, and their agile and quick adoption of new frameworks mitigated the risk associated. Additionally, the usage and reliance on packages, frameworks, and modules allows for quick prototyping and consistent UI/UX for both end users and developers.

6.3 Milestone Summary

Milestone 1: For the first milestone, Nolan, working as the implementation lead, constructed the foundational elements for the front-end. Additionally, he established the connection between the front-end of the tutoring portal and the back-end along with integrating it with the database.

Milestone 2: As of Milestone 2, significant progress has been made in implementing key functionalities within the Computer Science Learning Center (CSLC) portal. Notably, the ticket submission process has been developed, allowing students to submit help requests, select specific professors, and choose relevant courses. The admin panel now had advanced features such as announcements and settings. Additionally, the introduction of a light and dark mode in the settings enhances user experience and customization options.

Milestone 3: Milestone 3 had significant progress in database management, data loading, and ticket creation. A notable implementation at this step was the script for loading semester data from a CSV file provided by the registrar's office, ensuring accurate information on the portal.

Milestone 4: In Milestone 4 of the CSLC ticketing portal project, significant enhancements were implemented to the application. Users, specifically students and tutors, can now log in to the system, submit tickets, and claim tickets for assistance. The ticketing system supports various statuses, including open, claimed, and closed, with real-time updates. Tutors had the ability to update ticket information and mark tickets as successful or unsuccessful, providing valuable feedback. The dashboard offers a concise overview of total and new tickets within the past 24 hours, with potential for further filtering by admins.

Milestone 5: In Milestone 5, enhancements were made to the front-end code, incorporating improvements such as drop-down menus, tool tips, and elements designed to be accessible for screen

readers. The tutoring forms, form rendering, and documentation also underwent improvements. Additionally, various formatting issues were addressed and resolved.

6.4 Lessons Learned

The largest lesson learned by the Cringe Coders Team is the complexity and thoroughness required for secure, compartmentalized code that is both easy to collaborate on, and secure by design. The many moving pieces across different team members necessitated de-conflicting of general expectations of work and behavior when designing components of the website. The front-end and back-end are distinct components, and both are nested inside a docker container. This layering and interchange of traffic required a collaborative approach. Consequently, Cringe Coders Teamnow has abetter understanding of at-scale collaboration within Computer Science projects, with an emphasis on developing securely.

6.5 Future Extensions

The future extensions of integrating an uptake method for a CSV file containing course and teacher information into an existing capstone project, the potential for streamlining administrative processes.

The inclusion of an uptake method introduces a dynamic layer to the capstone project, enabling administrators to effortlessly update and manage course and teacher information by feeding a formatted CSV file. This feature streamlines data input and modification, reducing the manual workload associated with maintaining such information. This not only enhances efficiency but also reduces the likelihood of errors associated with manual data entry.

However, integrating the uptake method into the existing capstone project may pose challenges. Handling various CSV formats, managing potential data conflicts, and ensuring the security of the upload functionality are paramount concerns. Proper error handling and feedback mechanisms should be implemented to guide administrators in case of issues during the upload process.

Chapter 7

Local and Global Impacts

The establishment of the IS&T's CSLC is a transformative initiative with far-reaching positive impacts, both locally and globally. This tutoring center, designed to harness the intellectual prowess of students within the university, not only contributes to the academic success of individuals but also serves as a catalyst for societal advancement.

At its core, the CSLC plays a pivotal role in building academic excellence within IS&T. By providing a dedicated space for students to receive personalized tutoring in various technology courses, the center becomes a nexus of knowledge and collaboration. The employment of student tutors ensures a relatable and accessible learning environment, fostering a sense of camaraderie and peer support. This not only enhances the academic performance of the tutored students but also cultivates a culture of knowledge-sharing and mentorship within the university.

One of the immediate local impacts is the empowerment of students to overcome academic challenges. As the CSLC aids in comprehension and mastery of complex computing concepts, it acts as a springboard for students to excel in their coursework. This success, in turn, contributes to the overall reputation of the college, attracting prospective students and faculty, thereby bolstering the academic community.

On a global scale, the CSLC contributes to the advancement of information science and technology. In an era where computing applications have become ubiquitous and integral to various aspects of society, the center becomes a hub for cultivating the next generation of tech leaders. Graduates of the College of IS&T, shaped by the support and guidance received at the CSLC, are poised to make meaningful contributions to client organizations, local communities, and business and industrial communities worldwide.

The positive impact of the CSLC extends beyond academic success. It serves as a cornerstone for the development of well-rounded individuals who not only excel in their chosen field but also understand the broader implications of their work on society. As computing applications continue to shape the fabric of our interconnected world, the CSLC emerges as a beacon of education, collaboration, and societal progress within the realm of information science and technology.

Chapter 8

Appendix

This Page is intentionally left blank

Project Proposal

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 31st, 2023

Project Proposal

Contents

CHAPTER 1

Client Information

Project Proposal

1.1 Organization

The Computer Science Learning Center (CSLC), is a student service hosted through the College of Information Science & Technology (IS&T) for academic assistance and tutoring. The CSLC accepts tutoring by appointment or walk-ins Monday through Saturdays, and distance tutoring over Zoom. Student-tutors are paid for their services, and the University has administrators to oversee and manage CSLC staffing levels, work shifts, and payroll.

1.2 Point of Contact

Kyle Reestman is the director of the Computer Science Learning Center. Apart from his directorial position, he also serves as an instructor for CIST1400, an introductory computer science course. This course is designed to provide students with a comprehensive overview of programming basics utilizing the Python programming language. Furthermore, as the director of the CSLC, Kyle oversees the orchestration and management of the tutoring center's operations. He coordinates the tutors schedules, supervises the tutors, and ensures that the center's tutors are well-equipped with the necessary teaching materials. Through his capacities as both an educator and a director, Kyle Reestman guarantees the smooth operation of the CSLC while fostering students' development in computer science.

For the duration of this capstone, Kyle Reestman will be the primary point of contact. Secondaries, subject matter expert, and other consultation contacts will be at the delegation of Kyle Reestman, to be determined as necessary throughout the course of the project.

Project Proposal

CHAPTER 2

Motivation

Project Proposal

2.1 Context

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 serves 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.

2.2 System Needs

The existing CSLC portal has incomplete or missing features, functionality, and services. The current ticket submission system functions through a Google Form tied to a spreadsheet with the resulting student information.

2.3 Proposed Solution

To better meet the needs of the CSLC, the Cringe Coders proposes a full stack refresh of the existing technology stack. The ticketing system at present will be rebuilt to provide tutors and students with a more efficient way for exchanging essential information. This encompasses sharing specifics about the class assistance needed, articulating the present comprehension of the problem, identifying the respective course instructor, and the level of priority. The new portal will be able to facilitate this seamless sharing of this information between tutors and students.

2.4 Client Role

The client holds the important role of qualifying goals, and acceptance criteria. Throughout this project the client will be the source of truth and final authority for product quality and completeness. This power requires that the client provides periodic input into goals and the scope of work. To prevent redundant or extraneous work, a scheduled periodic meeting and an ad hoc communication channel are required during the continuous delivery process.

Project Proposal

CHAPTER 3

Project Description

Project Proposal

3.1 Project Overview

The initial conception of the scope of work involves a complete overhaul of front-end and back-end frameworks. Feature augmentations will be added as part of the refresh.

3.2 Features and Services

Concurrent with the replacement the current portal with a full refresh of the CSLC portal, Cringe Coders will add database tie-ins to the portal to replace the current Google Form solution.

3.3 Languages

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

3.4 Technology Stack

The front-end will use the React library and Tailwind CSS to enable better UI/UX. While 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. Additionally, 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, pending any client objections.

3.5 Documentation

Lastly, 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).

3.6 Potential Features and Services

At present no further work is scoped. This will be revisited and addressed according to client needs.

CHAPTER 4

Implementation Strategy

Project Proposal

4.1 Website Refresh

This transformation starts with a redesign of the website's user interface. This shift requires moving away from the CSLC's current reliance on Google Forms. It involves constructing a new front-end using react, a widely recognized Javascript framework, and Tailwind CSS. This front-end will be intricately linked to a back-end using the Django REST framework and a database server.

4.2 Back-End Database & New Features

Later continuous delivery cycles will focus on creating a sqllite database and integrating it with the refreshed portal website.

4.3 Security Controls

At all stages of delivery, security controls will be implemented at an architectural level, and periodically audited for validity and completeness.

Project Proposal

CHAPTER 5

Computer Science Challenges

Project Proposal

5.1 Challenges

We anticipate only trivial computer science challenges will arise in the scope of work. The only challenge revolves around the synchronization of the front-end and back-end for real-time updates. Successfully establishing seamless communication between these elements, alongside the portal's relational sqllite database, is mentioned because the implementation is novel to the Cringe Coders team.

Project Proposal

CHAPTER 6

Computer Engineering Challenges

Project Proposal

6.1 New Frameworks & Technologies

The success of this project hinges on everyone's familiarity with the technology stack in use. For some team members, and potentially the entire team, this means dedicating time to become acquainted with new frameworks and programming languages. The capacity to acquire new skills is essential in the world of software engineering. Also, given that not all team members have worked together before, getting an understanding on each other's work processes presents an additional challenge. In essence, the success of this project not only hinges on the team's technical expertise but also equally relies on the team's collaboration and communication.

CHAPTER 7

Team Members

Project Proposal

7.1 Background on the Cringe Coders

The Cringe Coders consists of four members with a range of hard skills; soft skills; professional computer science experience; and, experiences in adjacent fields of information assurance and mathematics. Based on internal deliberation among team members, we have aligned ourselves to the roles most apt for individual skill sets.

7.2 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.

7.3 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.

7.4 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.

Project Proposal

7.5 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.

Project Plan

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

Project Plan

Contents

Project Plan

Project Plan

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.

Project Plan

CHAPTER 2

Project Organization

4

Project Plan

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.

Project Plan

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

Project Plan

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.

Project Plan

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

Project Plan

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

Project Plan

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

Project Plan

- 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

Project Plan

5.12 Project Closure

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

Project Plan

CHAPTER 6

Project Schedule

Project Plan

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

Project Plan

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

Project Plan

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

Project Plan

CHAPTER 7

Monitoring and Reporting Mechanisms

20

Project Plan

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

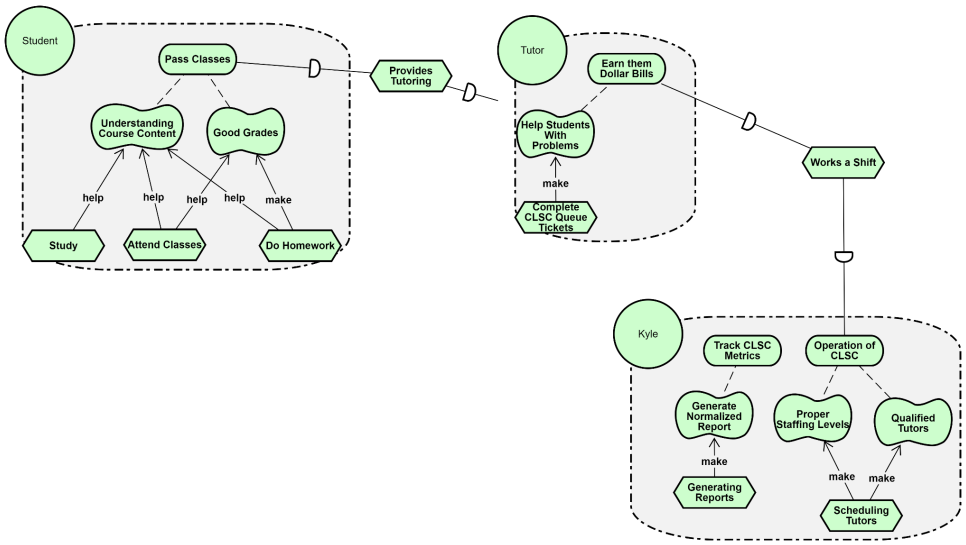
We will use Google Drive and Confluence to share project-related documents, reports, and documentation. This will ensure that everyone has access to the latest information.

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.

Initial Requirements

Goals

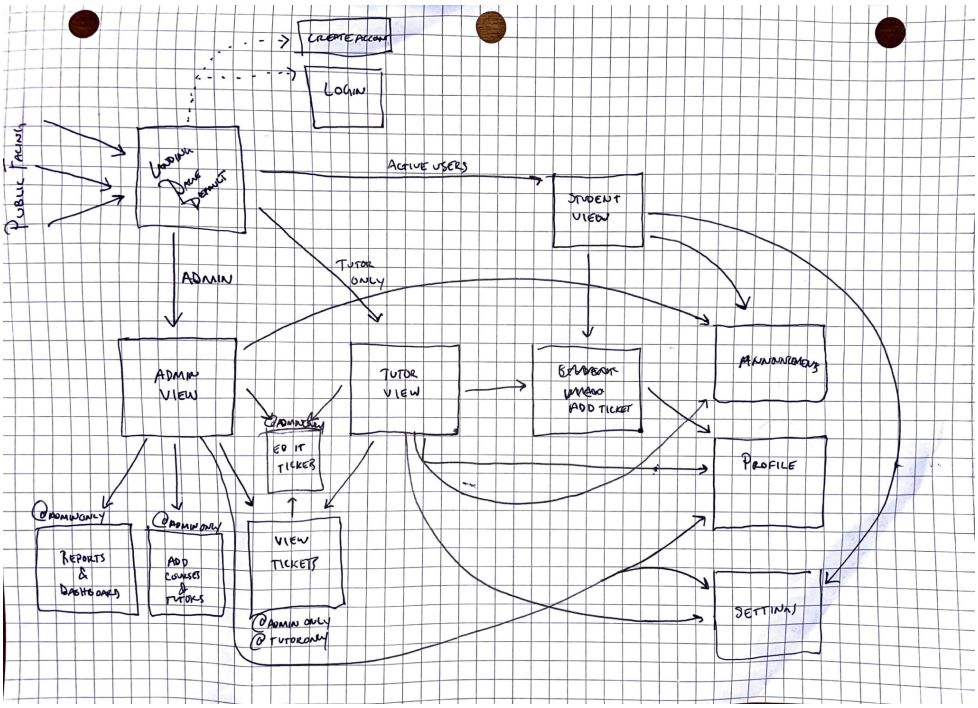


Functional Requirements

1. Must have. As the director of the Computer Science Learning Center, I want the CSLC portal to have efficient ticket management, so that tutoring requests can be managed.
2. Must have. As a tutor at the Computer Science Learning Center, I want to be able to interact with students and administrators regarding tutoring requests and progress as well as close and edit tickets, so that I can effectively provide help to students.
3. Must have. As a student I want to be able to make tutoring requests and see available tutors so that I can receive the help I need.
4. Should have. As an administrator, I want to be able to view and manage student requests, so that I can monitor progress and efficiency within the CSLC.
5. Could have. As a student, I want to be able to see how tutors are rated among other students, so that I can make an informed decision about choosing a tutor.
6. Won't have. As a user of the CSLC portal, I do not want to have to log in and authenticate redundantly, so that ticket submission is time efficient.

Initial Requirements

Wireframes



Context Document

Context Document

Cringe Coders

Austin Bailey
Mya Bell
Lindsey Langdon
Nolan "OG" Gregory

Context Document for the
Computer Science Learning Center Website Project

Dr. Harvey Siy; College IS&T
University of Nebraska - Omaha
Capstone Course
September 7th, 2023

Context Document

Contents

Context Document

Subject Facet

Kyle runs the CLSC, which tutors students via a ticketing system. Students request help in a certain subject or class, and the CLSC pairs them with an appropriate tutor. The CLSC is the official tutoring center for IS&T. Kyle uses ticket history to run a periodic report, detailing the classes that students struggle with most. Tutors at the center are paid employees and they use the ticket queue to find students they're qualified to help.

Usage Facet

The primary stakeholder is Kyle Reestman, who is the director of the Computer Science Learning Center and therefore oversees all of the operations within. The users of the system will include the students, student-tutors, teachers, and other staff members associated with the computer science department of the University of Nebraska at Omaha.

IT System Facet

Submitting a ticket on the CSLC Portal will be one of the first interactions a student will have with the CSLC. Therefore, it is crucial that this application will seamlessly fit into the already established technological environment. For context, the CSLC is tied to the University of Nebraska-Omaha. As a result, the reconstructed portal will have to conform with UNO's current authentication system and present a polished and professional user interface.

Development Facet

The main development concern with this portal is the familiarity with technology stack. To create a web application up to professional standards, everyone must be well-versed with the languages at hand. The team's overall proficiency with the technology stack will not only enhance the development process but also contribute to the long-term success and sustainability of the project.

Legal & Ethical Facet

We foresee no legal requirements, as the ingestion of data is self-reported by students, and only the tutors and Kyle can see it. Since this project will not integrate with other systems, basic security and architecture protections will suffice. The only possible issue we foresee is FERPA regulations limiting the projects integrating or sharing data with other systems.

Ethically, we believe any value-added aspects, such as 'About Us' or mission and vision statements have been addressed by the last capstone group. Our focus ethically will be around presenting a clean and professional interface for students. We believe that a haphazard or poorly designed interface will reflect poorly on the CLSC and serve to discredit the tutors by showing poor computer science principles on the CLSC website.

Mid Semester Project Report

Mid-Semester Project Report

Computer Science Learning Center Website Project

Cringe Coders Team

Austin Bailey
Mya Bell
Lindsey Langdon
Nolan "OG" Gregory

Mid-Semester Project Report for the
Computer Science Learning Center Website Project

Dr. Harvey Siy; College IS&T
University of Nebraska - Omaha
Capstone Course
October 12th, 2023

Report prepared in partial fulfillment of CSCI 4970: Computer Science Capstone Project

Mid Semester Project Report

Contents

Mid Semester Project Report

Mid Semester Project Report

Chapter 1

Introduction

1.1 Motivation

The Computer Science Learning Center (CSLC), is a student service hosted through the College of Information Science & Technology (IS&T) for academic assistance and tutoring. The CSLC accepts tutoring by appointment or walk-ins Monday through Saturdays, and distance tutoring over Zoom. Student-tutors are paid for their services, and the University has administrators to oversee and manage CSLC staffing levels, work shifts, and payroll.

The existing CSLC portal has incomplete or missing features, functionality, and services. The current ticket submission system functions through a Google Form tied to a spreadsheet with the resulting student information.

To better meet the needs of the CSLC, the Cringe Coders Team proposes a full stack refresh of the existing technology stack. The ticketing system at present will be rebuilt to provide tutors and students with a more efficient way for exchanging essential information. This encompasses sharing specifics about the class assistance needed, articulating the present comprehension of the problem, identifying the respective course instructor, and the level of priority. The new portal will be able to facilitate this seamless sharing of this information between tutors and students.

1.2 Similar Applications

Last semester's capstone course had a group lay the foundations of the project work, including a basic website with some more advanced functionality like authentication. While a CSLC tutoring portal exists, the current solution does not have the full suite of functionality that the client desires. This includes a ticketing for students to request tutoring, dynamic web page elements to show new tickets without the need to refresh the page, UI/UX considerations, and report generation for tracking ticket trends.

Core elements of infrastructure were established with the prior group, but some supplemental features were given work-around solutions. Case and point is the ticketing and reporting functionality. The current implementation is to use Google Forms to submit tickets, which allows for a limited (alibeit incorrectly and partially formatted) report.

In light of the limitations of current solutions, the Cringe Coders Team takes this as a motivation to overhaul the UI/UX and add further back-end functionality, as is highlighted in our Motivation section via a proposed solution outlined in Chapter 3: Architecture and Design.

1.3 Theoretical Foundations of the Project

The back-end database structure established by the prior group will likely need modification and updates, and thus necessitates additional modeling and design of structured tables and potentially databases (Connolly & Begg, 2014). Additionally, the data needs to be accessible via REST API, such that Django REST can make API calls to the database, for both read and write operations (Fielding, 2000).

Mid Semester Project Report

Though the CSLC is labeled 'Computer Science', the CSLC services all IS&T majors and students. Consequently, security practices should be implemented in the development and deployment of a CSLC portal. This includes protecting against common security flaws, such as Cross-Site Scriting (XSS), and Cross-Site Request Forgery (CSRF) (OWASP, 2021). Much of this security will derive from proper implementation of front-end developments through proper project management of Django (Django Project, 2021).

1.4 Software Engineering Challenges

We anticipate only trivial computer science challenges will arise in the scope of work. The only challenge revolves around the synchronization of the front-end and back-end for real-time updates. Successfully establishing seamless communication between these elements, alongside the portal's relational sqllite database, is mentioned because the implementation is novel to the Cringe Coders Team.

This in not to suggest we anticipate no cahllenges. The success of this project hinges on everyone's familiarity with the technology stack in use. For some team members, and potentially he entire team, this means dedicating time to become acquainted with new frameworks and programming languages. The capacity to acquire new skills is essential in the world of software engineering. Also, given that not all team members have worked together before, getting an understanding on each other's work processes presents an additional challenge. In essence, the success of this project not only hinges on the team's technical expertise but also equally relies on the team's collaboration and communication.

1.5 System Context

The CLSC portal will occupy the intersection of faculty and student academic interests. The current and proposed solution are designed to facilitate tutoring by, and for, IS&T students. Below Cringe Coders Team outlines the various facets of the system's context.

1.5.1 Subject Facet

Kyle runs the CLSC, which tutors students via a ticketing system. Students request help in a certain subject or class, and the CLSC pairs them with an appropriate tutor. The CLSC is the official tutoring center for IS&T. Kyle uses ticket history to run a periodic report, detailing the classes that students struggle with most. Tutors at the center are paid employees and they use the ticket queue to find students they're qualified to help.

1.5.2 Usage Facet

The primary stakeholder is Kyle Reestman, who is the director of the Computer Science Learning Center and therefore oversees all of the operations within. The users of the system will include the students, student-tutors, teachers, and other staff members associated with the computer science department of the University of Nebraska at Omaha.

1.5.3 IT System Facet

Submitting a ticket on the CSLC Portal will be one of the first interactions a student will have with the CSLC. Therefore, it is crucial that this application will seamlessly fit into the already established technological environment. For context, the CSLC is tied to the University of Nebraska-Omaha. As a result, the reconstructed portal will have to conform with UNO's current authentication system and present a polished and professional user interface.

Mid Semester Project Report

1.5.4 Development Facet

The main development concern with this portal is the familiarity with technology stack. To create a web application up to professional standards, everyone must be well-versed with the languages at hand. The team's overall proficiency with the technology stack will not only enhance the development process but also contribute to the long-term success and sustainability of the project.

1.5.5 Legal & Ethical Facet

We foresee no legal requirements, as the ingestion of data is self-reported by students, and only the tutors and Kyle can see it. Since this project will not integrate with other systems, basic security and architecture protections will suffice. The only possible issue we foresee is FERPA regulations limiting the projects integrating or sharing data with other systems.

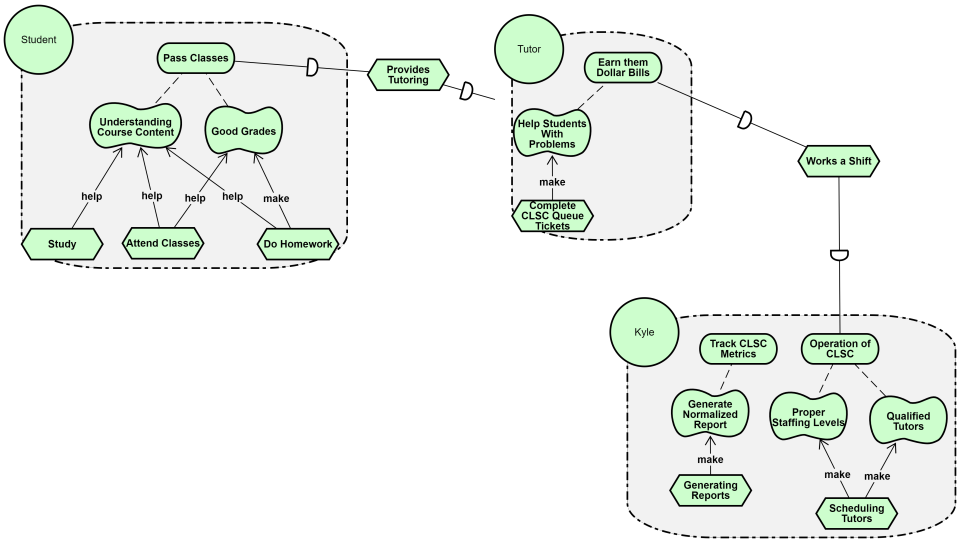
Ethically, we believe any value-added aspects, such as 'About Us' or mission and vision statements have been addressed by the last capstone group. Our focus ethically will be around presenting a clean and professional interface for students. We believe that a haphazard or poorly designed interface will reflect poorly on the CLSC and serve to discredit the tutors by showing poor computer science principles on the CLSC website.

CHAPTER 2

Requirements

2.1 Goals

2.1.1 Goal Model



2.1.2 Goal Descriptions

The Student is one of the users of the system. Each student has an overarching goal of passing their classes, which the CSLC portal is designed to aid in. Students will use the portal as a tool that will connect them with people who will help them to understand the content they need help with.

The Tutor is employed by the CSLC to tutor students. One of the main functionalities of the CSLC portal is the ticketing system. This function makes it easy for tutors to plan and organize what they need to do.

Kyle oversees the CSLC, so he has a goal of ensuring the smooth operation of the whole thing. The CSLC portal is designed to aid him in this goal by making it easy to schedule tutors and track what is actually going on.

2.2 Functional Requirements

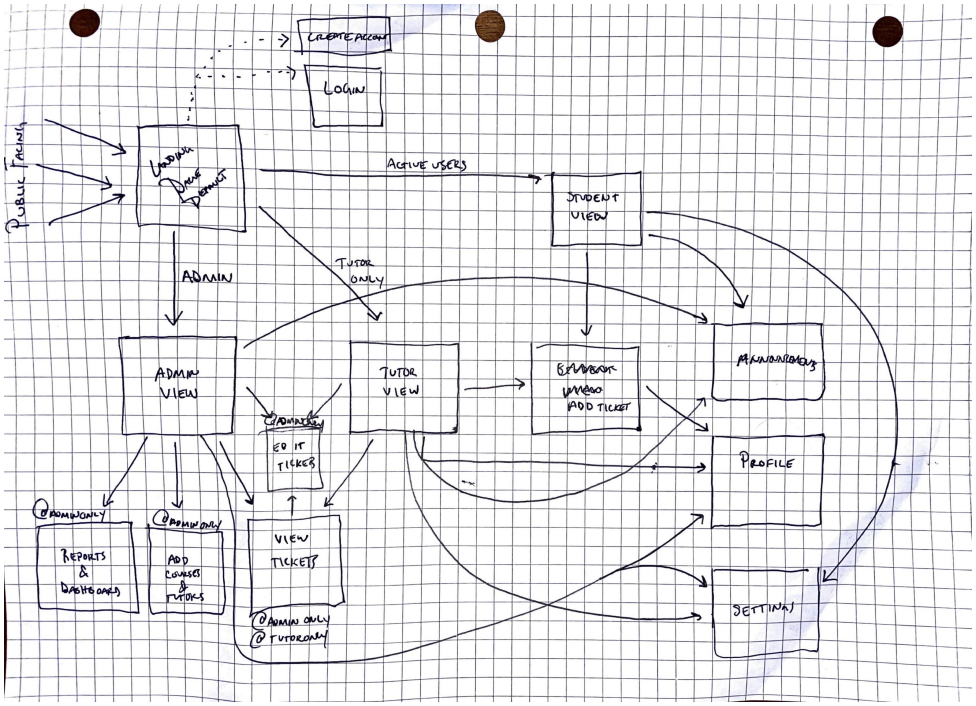
1. Must have. As the director of the Computer Science Learning Center, I want the CSLC portal to have efficient ticket management, so that tutoring requests can be managed.
2. Must have. As a tutor at the Computer Science Learning Center, I want to be able to interact

Mid Semester Project Report

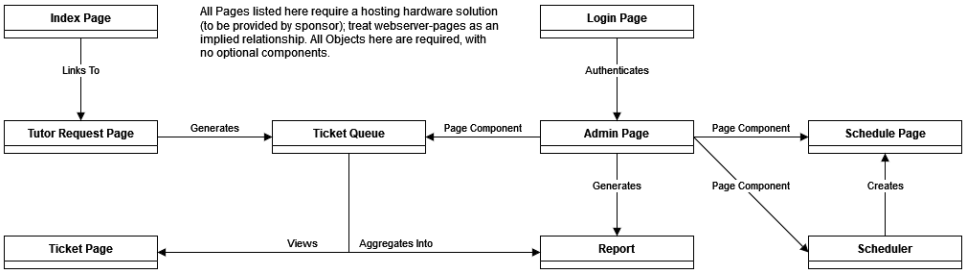
with students and administrators regarding tutoring requests and progress as well as close and edit tickets, so that I can effectively provide help to students.

3. Must have. As a student I want to be able to make tutoring requests and see available tutors so that I can receive the help I need.
4. Should have. As an administrator, I want to be able to view and manage student requests, so that I can monitor progress and efficiency within the CSLC.
5. Could have. As a student, I want to be able to see how tutors are rated among other students, so that I can make an informed decision about choosing a tutor.
6. Won't have. As a user of the CSLC portal, I do not want to have to log in and authenticate redundantly, so that ticket submission is time efficient.

2.3 Wireframes



2.4 Object Model



Mid Semester Project Report

2.5 Nonfunctional Requirements

1. Must have: Evaluating the CSLC Tutoring Portal application on ease of use from the point of view of the students in the context of submitting a ticket for help.
    - Q1: How easy is it to navigate the user interface?
        - M1. Task time
        - M2. Stroke count
        - M3. Problem count
    - Q2: How quickly can tasks be accomplished?
        - Reuse M1, M2, M3
    - Q3: Do users feel satisfied after using the portal?
        - M4. Conduct surveys to find the percentage of students reporting high satisfaction after using the portal
2. Should have: Assess the CSLC Tutoring Portal application on maintainability from the point of view of the developers in the context of ensuring that the application is up to date.
    - Q1: How maintainable are the languages in the stack?
        - M1. Check how often the languages are updated
    - Q2: How many dependencies are there?
        - M2. Count the dependencies

2.6 Analysis Requirements

No Analysis requirements have been determined as of yet. (TBD in Final Draft)

2.7 External System Interfaces

- API specifications of existing systems that interact with your product, e.g., the name of the application being accessed, services called, parameters passed, outputs returned, set up instructions (e.g., API keys)
- file formats your product is required to use, e.g., if you had to input from, or output to a file with a certain csv format

(add content as needed)

2.8 Requirements Not Implemented

(TBD in Final Draft)

Mid Semester Project Report

CHAPTER 3

Architecture and Design

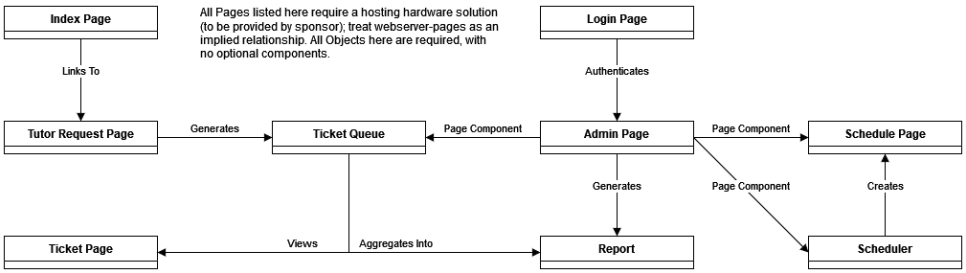
3.1 Overview

The CSLC portal will comprise a handful of web pages hosted on the `https://cslc.unomaha.edu/` subdomain, with integration of a back-end database. at a high-level technology-agnostic view, we have the following design requirements

1. A guest, index, or public page that links to the portals resources.
2. A ticket or tutoring request submission resources.
3. A tutor or employee view that allows access to the ticket queue and shift schedules of tutors.
4. An admin view that allows the client to export reports based on ticket history, and to configure work schedules.
5. A back-end database to facilitate the ticketing processes.

Subsequent sections will cover these designs through the various lens, culminating in a more concrete description of architecture.

3.2 Logical Decomposition View



Cringe Coders Teamprovides a simplified UML diagram of both the public and administrative functions of the CLSC portal. Note that the decomposition has merged the employee and admin pages, as the Cringe Coders Team's current intent is to implement RBAC roles associated with authenticated users that reveal components of the admin page. This will ideally minimize redundant pages and code.

For the sake of simplicity and ease of conveying design, the database subsystem has been omitted, as it will integrate will almost every other subsystem.

to see the Cringe Coders Team's accompanying functional requirements, see the Functional Requirements section.

Mid Semester Project Report

3.3 Technology Stack

The Cringe Coders Team intends to replace the current portal with a full refresh of the technology stack. Additionally, will add database tie-ins to the portal to replace the current Google Form solution.

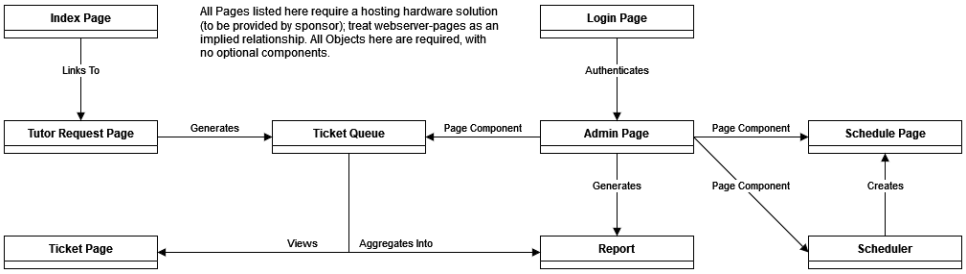
3.3.1 Languages

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

3.3.2 Libraries & Services

The front-end will use the React library and Tailwind CSS to enable better UI/UX. While 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. Additionally, 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, pending any client objections

3.4 Deployment View



(add content)

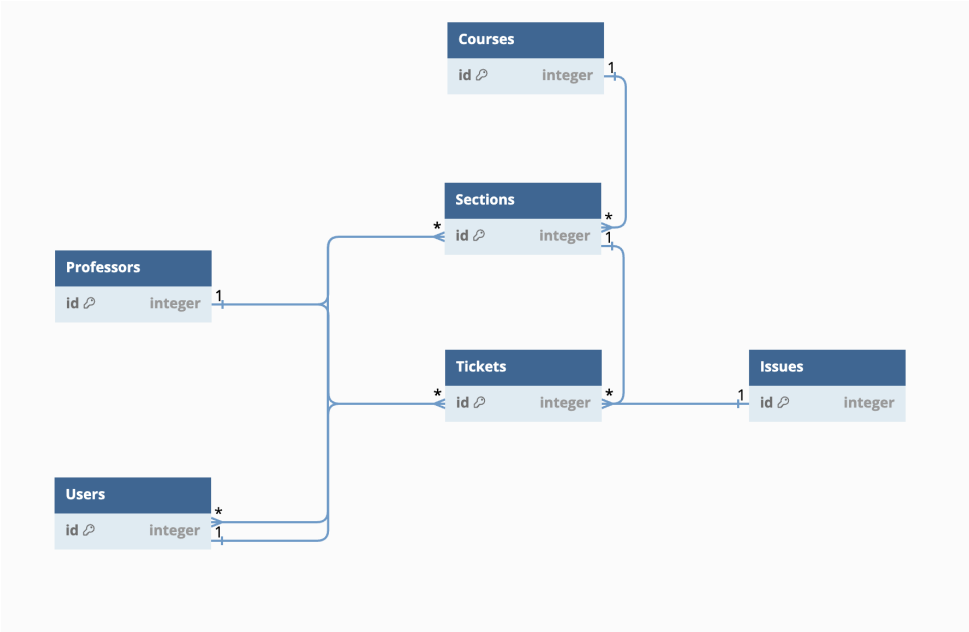
3.5 Development View

- discuss how the code is organized and how this organization is related to the previously identified subsystems
- discuss the role of the framework(s) used and how your code interacts with them
- identify which code modules or components were reused (e.g., open source libraries, existing code from another project, etc.)
- describe API calls to external services and third-party libraries
- corresponds to C4
- Links to an external site. Levels 3 and 4 diagrams
- 1 or more UML class diagrams are required
  - if there are 15 or less modules/classes, show interconnections between classes
  - if there are more than 15, use a UML package diagram to show the hierarchy

Mid Semester Project Report

(add content)

3.6 Data View



Our current back-end relational database uses six tables. A More detailed version, and additional details will be made available in the final report.

3.7 Concurrency View

Cringe Coders Team does not anticipate work that would involve this section. we include this section as a placeholder and an acknowledgement that future work may re-scope us into this area.

3.8 Execution Flow View

- discuss illustrative scenarios that show data and control flow through the application modules to deliver a particular service
- 1 or more UML sequence diagrams are required

(add content)

3.9 Screenshots

(TBD in Final Draft)

Mid Semester Project Report

3.10 Summary of Design Changes

- summarize as applicable how the design has evolved from the original concept to the final version and explain the reason for the changes

  (TBD in Final Draft)

3.11 Design Modularity and Extensibility

A stretch goal currently slated for work (but unlikely to be accomplished within the scope of our work) would be an integration with the registrar's office to automatically populate current IS&T courses. Both the client and the Cringe Coders Team is unaware of an existing structured data source.

Future work would involve scoping and understanding the data's formatting requirements. A likely implementation would be an ingestion of a csv file through the admin portal.Via REST API, the csv data would populate a table in the back-end that would populate elements such as drop down menus for selecting a course on tutoring tickets.

Chapter 4

Implementation

4.1 Directory Structure

(TBD in Final Draft)

4.2 Technical Issues Encountered

(TBD in Final Draft)

4.3 User-Reported Bugs

(TBD in Final Draft)

CHAPTER 5

Testing

5.1 Test Plan

This test plan outlines the high level testing strategy for the CSLC Tutoring Portal. The primary objective of this testing plan is to ensure the robustness, functionality, and performance of the web application. This plan includes both integration testing and end-to-end testing, enabling our team to deliver a high performing web application that will meet the demands of the end users.

In the integration testing scenario, we need to test the API integration and the component integration. With testing the API integration, we need to verify that the front-end properly communicates with the REST API back-end by testing API calls and responses for different endpoints. For component integration testing, Jest can be employed to ensure that the different web components of the application work together as expected. As Jest is primarily known for its unit testing, we plan to develop unit tests for our components to help identify potential bugs during the testing phase.

The end-to-end testing approach in this plan is designed to comprehensively evaluate the users' journey from start to finish. We plan on using Selenium to stimulate users actions such as clicks and inputs to asses the application's performance. Additionally, Selenium can be used for cross-browser compatibility testing to ensure that our application works across a variety of browsers. Similar to the integration testing, the end-to-end testing process is focused on identifying any potential bugs during the testing phase, ensuring the delivery of a dependable and user-friendly application.

5.2 Obtaining Realistic Test Data

(TBD in Final Draft)

5.3 Tests Conducted

(TBD in Final Draft)

5.4 Test Results

(TBD in Final Draft)

5.5 Automated Test Outputs

(TBD in Final Draft)

5.6 Tests Not Conducted

(TBD in Final Draft)

CHAPTER 6

Summary

6.1 Summary of Project Organization

(TBD in Final Draft)

6.2 Outcome of Risks

(TBD in Final Draft)

6.3 Milestone Summary

(TBD in Final Draft)

6.4 Lessons Learned

(TBD in Final Draft)

6.5 Future Extensions

(TBD in Final Draft)

Mid Semester Project Report

Chapter 7

Local and Global Impacts

(TBD in Final Draft)

Chapter 8

Appendix

This Page is intentionally left blank

8.6 Setup & Deployment Instructions

The most detailed and up to date instructions for deploying the project are available through the GitHub repository: `https://github.com/nulzo/University-Nebraska-Tutor-Portal`. The as-is rendering the markdown instructions is given below for the sake of completeness.

Github README.md Instructions *(As of 26-Nov-23)*

University of Nebraska CSLC Tutoring Portal

This application is designed to facilitate communication between students and tutors by allowing students to create, manage, and track tickets for tutoring assistance. Tutors can view, modify, claim, and close these tickets, ensuring a smooth and efficient tutoring experience. Additionally, an admin panel is available for database management and data retrieval.

Explore the docs »

View Demo . Report Bug . Request Feature

Project

Contributors

Table Of Contents

Github README.md Instructions *(As of 26-Nov-23)*

- License
- Authors
- Acknowledgements

About The Project

There are many great README templates available on GitHub, however, I didn't find one that really suit my needs so I created this enhanced one. I want to create a README template so amazing that it'll be the last one you ever need.

Here's why:

- Your time should be focused on creating something amazing. A project that solves a problem and helps others
- You shouldn't be doing the same tasks over and over like creating a README from scratch
- You should element DRY principles to the rest of your life :smile:

Of course, no one template will serve all projects since your needs may be different. So I'll be adding more in the near future. You may also suggest changes by forking this repo and creating a pull request or opening an issue.

A list of commonly used resources that I find helpful are listed in the acknowledgements.

Built With

- **Backend:** Django REST framework
- **Frontend:** React and TypeScript
- **Proxy Server:** Nginx
- **CSS Framework:** Tailwind CSS

Getting Started

This is an example of how you may give instructions on setting up your project locally. To get a local copy up and running follow these simple example steps.

Prerequisites

You will need the following packages and libraries in order to successfully run the application:

1. **npm**

   ```
   npm install npm@latest -g
   ```

Github README.md Instructions *(As of 26-Nov-23)*

2. **Poetry**

```
curl -sSL https://install.python-poetry.org | python3 -
```

3. **Docker**

```
sudo apt-get update
sudo apt-get install docker.io
```

4. **Python**
   - Python is often pre-installed on many systems, or you can install here: Python Images

5. **TypeScript**

```
npm install -g typescript
```

Installation

1. Clone the repo

```
git clone https://github.com/nulzo/University-Nebraska-Tutor-Portal
```

2. Run Docker Compose

```
docker compose up --build -d
```

Usage

Use this space to show useful examples of how a project can be used. Additional screenshots, code examples and demos work well in this space. You may also link to more resources.

For more examples, please refer to the Documentation

Roadmap

See the open issues for a list of proposed features (and known issues).

License

Distributed under the MIT License. See LICENSE for more information.

Github README.md Instructions *(As of 26-Nov-23)*

Authors

- **Nolan Gregory** - *Lead Developer* - Nolan Gregory - *Created the application*

Acknowledgements

- ShaanCoding
- Othneil Drew
- ImgShields

8.7 Team Meeting Notes

Sans the initial client meeting, the Cringe Coders Team has felt no sense of urgency or disconnect between us and our client that would necessitate extra-ordinary meetings outside of class time. Consequently, we have held no meetings outside of class times, and our meetings during class time typically span about five minutes. Therefore, we have neglected to keep a record of internal team meeting notes.

We do have a record and transcription of our initial client meeting attached in the appendix separate from our team meetings, and we do have the below interaction with out client Kyle. Much, if not all of the intended content, such as assignment of work and completion of work are readily implied by the Cringe Coders Team's individual journals.

October 3rd Mid-Semester Meeting Scheduling

Nolan Hey [Kyle]! Hope everything has been going well these past few weeks. We have made some pretty solid progress on the portal so far. During our first meeting we talked about meeting up sometime around mid-October for another, so I was just reaching out to get a date figured out. I think fall break is in two weeks, so would you have availability to meet up sometime next week? I think the best time for our group is Tuesday/Thursday from 1:00-1:30 or sometime around 2:00ish - it's hard to tell how long Harvey will lecture for. Just let us know if any of those times work for you, and we can figure something else out if you're booked during those times. thanks!

Kyle Well, I don't know if any of you are going out of town for Fall Break, but I could do the Tuesday during it (the 17th) in the afternoon and then you wouldn't have to worry about when the lecture gets out. If not, we could do that Thursday (the 19th).

Also I forgot to give the order I was wanting the reports to be in. Here is that order: URL, Student Email, Student First Name, Student Last Name, Assignment, Question, Problem Type, Status, Time Created, Time Closed, Was Successful, Primary Tutor, Assistant Tutor, Semester, Course Number, Section Number, Professor, ANYTHING NEW

And the url it should use: `https://cslc.unomaha.edu/`

Nolan Hey Kyle, I have spoken with the group and I believe Thursday will work best as some team members have plans for the midterm break. I'd be happy to meet you in your office or at the CSLC or something and pick out a time that works!

Also, we have some exciting news to share: Harvey looks to have gotten some VMs for us to use, so hopefully we can get that figured out so you can have an easy test environment to mess around with instead of pulling down and running docker containers :) I'll keep you posted with that, still need to figure out exactly how he has them set up!

Looking forward to seeing you soon, and let us know if you have any questions in the meantime :) have a good weekend!

8.8 Individual Journals

8.8.1 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.

Journal
8/31 - built project context with lindsey
9/05 - met with client; record and will transcribe
9/07 - met helped build context document
9/12 - record context document with Mya
9/14 - Attended class; Built a goal model; provided pen + paper for wire framing
9/19 - Attended class; consolidated documents for submitting to canvas with team; scoped work of UML diagram to be done by EOD thursday (9/21)
9/21 - attended class
9/26 - Weekly Meeting with Siy; attended class
9/28 - Attended class
10/03 - Coordinated with Nolan for follow-up meeting with client; attended class
10/05 - no changes to project; team agreed no changes were necessary. Attended class
10/10 - context document formalization; project report work; attended class
10/12 - Mid-semester Project report finalization
10/17 - Fall Break 10/19 - Client meeting; recorded meeting; attended class
10/24 - Attended Class; Studying for MFT; Formatting transcript
10/26 - Attended Class; Professor Check-In; Studying for MFT; Formatting transcript
10/31 - Attended Class; Formatting transcript
11/02 - Attended Class; Professor Check-In; Addressed any IP concerns (N/A)
11/07 - Attended Class; Formatting transcript
11/09 - Attended Class; Professor Check-In
11/14 - Attended Class
11/16 - Attended Class; Professor Check-In
11/21 - Finalizing transcript
11/23 - Thanksgiving
11/28 - Attended Class; Review mid-semester report comments; Begin drafting final report
11/30 - Attended Class; Professor Check-In; drafting final report

8.8.2 Mya Bell

Client Liaison
Requirements Analyst
Quality Assurance Tester

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.

Journal
August 31 - Discussed and finalized individual roles, project proposal, and project plan
September 5 - First meeting with client

September 7 - Wrote and recorded context document
September 12 - Wrote initial requirements document
September 14 - Finished requirements document
September 19 - Started nonfunctional requirements
September 21 - Discussed object model and finished nonfunctional requirements and updated requirements
September 26 - Went over progress on code milestone 1
September 28 - Check in with Mr. Siy
October 3 - Updated client on progress and scheduled next meeting with client
October 5 - Check in with Mr. Siy and reviewed current project plan
October 10 - Automated testing tools demo
October 12 - Check in with Mr. Siy and worked on midsemester project report

8.8.3 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.

Journal
See github commit history

8/22 - Made the GitHub repository.

8/23 - Added Django and React to the repository.

8/23 - Added Github workflows to the project.

9/4 - Work on getting the API connection set up and added a linter/formatter.

9/5 - Added Makefile to automate tasks.

9/11 - Converted React frontend to use Typescript.

9/15 - Added authentication routing.

9/18 - Added form logic to frontend.

9/30 - Development on state management across project.

10/3 - Added a new backend API.

10/5 - Wrote tests for API calls.

10/8 - Update database schema.

10/10 - Create an API layer.

10/12 - Improve REST API.

10/13 - Added websockets to the application.

10/13 - Refine directory structure.

10/13 - Build README.

10/17 - Added darkmode, new UI components, and improved loadtimes.

10/17 - Added admin settings.

10/19 - Enhancements to the Github workflow pipeline.

10/20 - Added popout sidebar and improved responsiveness.

10/20 - Added admin announcement and download feature.

10/23 - Improve API layer access and create script that seeds database.

10/25 - Added security workflow and containerization within docker.

10/27 - Added new pipeline stages and enhanced seeding data.

10/27 - Added new testcases.

10/29 - Changes to login status and submission requirements.

10/29 - Added nginx as a proxy server.

10/29 - Added HMR and data synchronization between local environment and docker.

10/31 - Added new admin secions and improvements to form resolution.

11/2 - Made application responsive on a mobile device.

11/5 - Changed the way tickets can be viewed in the dashboard.

11/7 - Updated DB schema and added information to the README.

11/10 - Added query string sanitization.

11/10 - Updated nginx routing configuration.

11/11 - Improvements to the layout of the directories.

11/17 - Updated documentation.

11/17 - Finalized backend schema and build architecture.

11/20 - Finalized docker architecture and design.

11/22 - Finalized nginx architecture and design.

11/24 - Improvements to the abstraction of API and frontend.

11/25 - Added fuzzy search algorithm to the search on tables.

11/27 - Genericize form fields such that they are reusable.

11/30 - Finalized frontend build and architecture.

11/31 - Improved formatting to achieve perfect linting score on all facets of app.

12/1 to 12/7 - Add and update documentation.

8.8.4 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.

Journal
8/31 - built project context with Austin and worked on Project Proposal
9/5 - went over client meeting notes
9/7 - worked on wireframing exercises
9/12 - assisted with writing context documentation
9/14 - recorded Wireframes and Goal Model Requirements presentations
9/19 - worked on GQM Modeling
9/21 - attended class
9/26 - weekly Meeting with Siy
9/28 - attended class
10/3 - attended class
10/5 - team agreement that there were no changes to project plan
10/10 - recorded Testing Tool Demo presentation
10/12 - worked on Mid-semester Project report
10/17 - attended class and helped work on milestone two

8.9 Transcripts of Client Meetings

Transcript of Initial Client Meeting

Transcript of Initial Client Meeting

Computer Science Learning Center Website Project

Cringe Coders Team

Austin Bailey
Mya Bell
Lindsey Langdon
Nolan "OG" Gregory

Transcript of Initial Client Meeting for the
Computer Science Learning Center Website Project

Dr. Harvey Siy; College IS&T
University of Nebraska - Omaha
Capstone Course
September 7th, 2023

Transcript of Initial Client Meeting

Transcript of Initial Client Meeting

Nolan	So this is Austin.
Austin	Hi.
Kyle	Nice to meet you
Nolan	And Maya.
Kyle	Okay, Nice meet you.
Nolan	I think they both you took cs1 in Java, right? Okay.
Kyle	Okay.
Austin	Yeah. Also I'm old guard.
Kyle	Did you take cs2 and grab with them as well?
Austin	Yeah, I started in 2018.
Kyle	Okay.
Austin	So i i've been here a minute
Kyle	may or may not have had me for CS2 then depending on
Austin	Um, i don't, no, i took a cs2 in the summer so i had So the guy teaches theory of Comp.
Austin, Kyle	Solheim.
Austin	Yeah, I had him because I decided to do calc 3 and java 2 in the summer compressed.
Kyle	Yeah. But, Yeah, because i yeah, i was doing cs2 the online section of cs2 for A while when i was in java.
Austin	Was that during COVID or?
Kyle	Yes. During COVID as well.
Austin	Okay, Because I I was here 2018 to 19 and then 19 to 20 activity gap year, a lot and then I came back to COVID but I had already finished up pretty much all the The low level cores.

Transcript of Initial Client Meeting

Kyle	Yeah, I don't know. Do you want it closed or not? I don't know if you're recording, if it's going to screw with sound or something
Austin	it shouldn't And it doesn't have to be perfect
Nolan	Leave it cracked because I don't know how people, you know, everyone else reacts to sound traveling. Don't want to step on any of your professor's toes.
Kyle	So yeah, awesome. Um I know that i think they got in contact with you the previous group, but yeah, what the code base looks like
Nolan	Yep. So they used flask and pretty much just flask for everything. Um, just fine. So i looked at a probably going to reuse all their model, like their database diagrams that they have But we're going to use a different ORM. We're gonna use Django instead Just so that can plug in with any I guess the database structure that you choose. Uh, whether that be like mysql or postgres or anything like that, And then, We're going to use react also, so that like whenever we have a component change or something like that, like if a student submit a ticket we don't have to refresh the page. So it's all like Loaded dynamically.
Kyle	Okay
Nolan	Just because that's kind of annoying. Whatever. I was working with the tutoring center. You always had. Like, I remember I downloaded a chrome extension. That's like an auto refresher like every five seconds or something. Just get the students.
Kyle	Oh Okay
Nolan	They're like submitted a ticket but it didn't pop up or something like that. Um, and then because that was something that Clim and Clayton, Clayton? Yeah, yeah. That was what they wanted to do that but they ran out of time because they said that, they Something happened where they weren't able to start development until the end of march.
Kyle	Yeah, so there was an issue with originally we had we didn't have a hosting solution
Nolan	okay?
Kyle	So our first thought was trying to host it on the UNO community page. Which doesn't allow python for some reason. You can do it. The thing like, is set up to do it. But like UNO, specifically doesn't allow python for it.
Nolan	What do they prefer
Kyle	PHP.
Nolan	Seriously? In 2023?

Transcript of Initial Client Meeting

Kyle So so they were they were like looking into that as a potential option to do like a lamp stack and pHP. Um, but then decided that that was just called That was just a bit too much. So we pivoted at the last minute and tried to find other hosting solutions, I do think O have a hosting solution now.

Nolan Cool.

Kyle Which is the good news, I haven't super logged on to it much. But I was just looking at that information.

Nolan Uhm

Kyle So, there is A new domain here. I'm just compiling kind of a list of notes of things. Um, That'll be cslc.unmaha.edu, so like there should be a domain created and everything

Nolan okay

Kyle I will have access to it. So you had mentioned like late november. Part of this will be to help deploy it. So I have access to that so we can like meet and set up a time to actually deploy.

Austin It's just a working session for an hour or two?

Kyle Essentially. Yeah, just to get it up and running and then from there, you can only do your testing on it. If things need to change, then it should be easier for me to do redeploys the second time after i, like go through the first time. So

Nolan cool. Yeah we can help set up like a integration or something just so that it's easier to deploy if something happens or something needs to change the code base or something. Its just a quick little thing push it and it just automatically.

Kyle Yeah. I like the script essentially again. Cool, that sounds great. Um, so yeah, i think that that will be part of the the goals i guess is because i know i know you had mentioned like late, november?

Nolan that's, Yeah. it's like ideal

Kyle if that's the if that works. Um, And then they have some documentation. I don't know how much more documentation we'll need to be added, of course, with this, which to django, there will be somethings Um, Because i, i, All of the set up documentation as well. Should be updated then. To be django.

Nolan Yeah. Like to build stuff.

Austin It'll mostly just be translating it from

Kyle basically. Yeah

Transcript of Initial Client Meeting

Nolan that's a big part of this is basically just making it jump over from flask to django to Django. So that's easier.

Kyle Um yeah there is one thing i don't i didn't get to look too much at their reporting piece because that was one thing that i had talked to them about and asked about and they they had showed it off a little bit but didn't fully look into it.There is a format that i would prefer because i have a program on my end, that does an analysis. Of the report file and like, spits out things for me.

Nolan You want it as a CSV or a JSON or something?

Kyle Yeah, i take it in as a CSV,

Nolan okay?

Kyle And I have like, the specific. Fields. That i'm kind of looking for like we, we can add more fields, that's perfectly fine. Um –

Nolan but it's like a base

Kyle but that'd be a base of like what i'm looking for in that report. Um, and then if we add anything else, probably best at the end, just i'm thinking backwards compatibility-wise for my analysis program. So don't have to change too much myself.

Nolan Um, so like an advent panel sort of thing.

Kyle Yeah, yeah. So there should be an admin panel that set up in there and then there should be a way to generate a report. Um, i don't remember if they set up like Over certain times that you can generate it or for certain classes, i'm not certain.

Nolan It could be pretty. I mean doing it for a certain things. Like you could like filter it but things that would be pretty easy to do. That was one of the things that I worked on previously, actually. So,

Kyle So so yeah, so that's kind of the That's I guess the first goal and I don't know if we wanted to to meet up. Maybe sometime around fall break. Just to kind of see where you were at, as well.

Nolan Yeah, when it's fall break is that

Austin it's like October 18th or something?

Kyle yeah, like 16

Austin october.

Nolan Yeah, i'm trying to remember when the First milestone is okay. So i think it's around that time now, october something,

Transcript of Initial Client Meeting

Austin	I mean, in all honesty, I'm not too concerned with the capstone's milestones. I'm more so concerned with Incremental progression the end state which i don't think will have an issue with scopes.
Kyle	Yeah. And that would be six weeks as well. So that's that's a decent amount of time. Just to at least, you know, Start work on it but also like have an idea of it so that we can talk more about it on like what exact expectations will be
Nolan	okay.
Kyle	But um, Yeah, definitely. Like if you want to switch over to Django, get react up. That'd be great.
Nolan	Yeah. Is there like, i mean, if that seems too com – not complex, but anything that's like out of the, ordinary.
Kyle	I I think that should work because there is already a code base, it's just shifting the framework, and then adding a little bit on top. I think that'll be fine. Get the reports working and, like, to to the set standard because I don't know if they quite match the standard with their report that I would prefer, okay? And then, Uh getting the deploy set up.
Nolan	Okay?
Nolan	So that that that really are the main takeaways that i'd like from this.
Nolan	Yeah, that's um, easy, right? Easy. But that'll be, yeah. Fun to do.
Kyle	And then so like if we want to meet around like that week after fall break or something, Like so that'd be six weeks. Um, so we could chat.
Nolan	Okay, let me throw it out little thing here. So don't forget so that week after fall break
Kyle	or the week of fall but would whichever whichever Sometime around that ever. Yeah, sometime mid october. Um, And then maybe. Is it, is it due? The eighth is that when Like due date is for the capstone.
Nolan	Oh, the final report? Yeah, it's sometime around the eighth of July, something like that.
Austin	Yeah, the cyber capstone is the eighth or at least that's when they do the presentation,
Nolan	i think. Mya i think, oh, yeah, the eighth, Mya i put it on the check at the Jira board here. So yeah
Austin	I know for the cyber one. They they actually start winding down about the 24th of november because then you go into Um,

Transcript of Initial Client Meeting

Kyle thanksgiving.

Austin Yeah, thanksgiving and then you have dead week.

Nolan Yeah. That i would want it to be live like the last week of november at the latest just so that we could test it and see if there's any bugs.

Austin If we can get it live before thanksgiving, then that just gives us kind of three weeks. Of –

Kyle yeah

Austin bug fixing UI/UX

Kyle that was kind of my thoughts. So either either that week before, thanksgiving, or the week of like the first few days of that, um, i think would work. And then, yeah, because always, after thanksgiving, there's a week of class dead week, finals week, and that dead week is when the The final milestones are due, so, If we can get it done before thanksgiving, then you can have that last week of like class after Thanksgiving to Actually, do testing and, and milestone checking and all that correct.

Nolan And then, if i set up like a little test, like a production environment, like, do you know what type of database this we'll use or does it?

Kyle I'm guessing it's my sequel. I don't quite remember, i'm pretty sure they have it in the documentation which one they used as well.

Nolan Okay, oh yeah.

Kyle But i would, i would guess

Nolan I think that they used mySQL. Also.

Austin I mean, that's what they teach. So

Kyle It's it's what should be on all the things? Um, and then, i don't know if you want to set up like a discord group for, if you have another way that you're communicating,

Nolan yeah, we have one Clim was saying that they had one with you last semester. Um, if you want, we could just do that one. Also, if you already have that or

Kyle oh, if you want to join that.

Nolan Yeah. Because he was saying, like, it's just so that you don't have a million different chats all the time. Other things. Doesn't matter to us and we have a discord chat. We could add you to it. Or

Transcript of Initial Client Meeting

Kyle	whichever one, you're more comfortable with because if you still want to have one without me, that's that's fine. Okay.
Nolan	Like we could
Kyle	I encourage you to not always have to bring me into the conversation.
Nolan	Yeah, we could do it. Like the way that tutoring one's set up where it's just like you have the one channel, like an advisor channel or something like,
Kyle	oh, you have like a, you have like a server?
Nolan	Oh yeah, we have a server.
Kyle	Oh, yeah. Yeah. If you want to do it that way. That's easy. Okay. So sorry they had a group chat.
Nolan	Yeah, i actually a group, would be better because i want to clog out your left side screen with another server because that happens about this – every class.
Austin	Don't you group yours at all?
Nolan	No, i just
Kyle	oh, i do some grouping. I have like a
Nolan	i don't even know that
Kyle	you know, everything folder
Austin	like i have. Like my UNO ones. I like my friends, social ones and gaming and like one tab and yeah, i'll just like minimize them and then i have like three things over there.
Nolan	I didn't know that i'll have to do that because
Austin	you're looking all cluttered over there bud.
Nolan	Yeah, this isn't even my this is just my like one of my personal discords
Kyle	I just have the one that i use for everything.
Nolan	I got scared, i didn't want to like have my personal one, be linked to school stuff and then accidentally send like a meme and like a school server or something. I don't know.
Austin	Are you saying your memes are too spicy?

Transcript of Initial Client Meeting

Nolan	Maybe, i don't know.
Austin	I don't believe you.
Nolan	Uh, okay, so cool. That's a That's good, like starting.
Kyle	Well and then and then i can send you so so once once you, you end up setting that up,
Nolan	okay? Okays
Kyle	Um, like the the url to use, because i know that that's something that Clim was asking for toward the end of the last semester was that he needed an actually, a URL. So, i can do that and, as well as the report Uh, Fields.
Nolan	Oh yeah, that would be perfect. Just database like relationships and stuff. But yeah, voice for voice pulling stuff down. It should be pretty easy with like the relationality of the database just to like filter things to grab like Oh i want all of these professors and stuff like that, that would be pretty easy. But there was like some things i was thinking. And i don't know, like how Not ethical, but like Uh, rating system for like tutors, for example, like a student could leave, like a thing, that's kind of like In concept. It's good. But i feel like i'm practice really bad idea.
Kyle	Yeah, what we've used and i don't know. I haven't checked it in a little while. But we had like a A feedback form.
Nolan	Okay
Kyle	not many students would go and do it. It's only if they had like a particularly bad experience is when they would do.
Nolan	Yeah that seems
Kyle	but um we did have a link to a feedback form.So, i mean, i could I could, we could look to that and include that. Otherwise,
Nolan	it's up to you. Yeah. I was just kind of thinking of things that could be fit up at me.
Kyle	Yeah, there's a few other things. Um, One that i would like that is not currently a field.But i would like to be a field is whether it was an online ticket or not. And some way of marking it as an online ticket. I think i think they did do that, but i definitely want that to be included. Um, And then i was also thinking of
Nolan	We could make it like a required one too, so, it's not always like, uh, you forget them?
Kyle	Yeah.

Transcript of Initial Client Meeting

Nolan Or something like that. That would be really good.

Kyle Because then the other one that i was thinking of and this would be A lot more work. So this is very much not a requirement. Um, but if there was an easy way, a script that could run That could. Look through the corse search or maybe i know that there was a group, a capstone group that did like the course scheduler for the university.

Nolan Oh, and like auto populate

Kyle auto populate for the semester because currently what happens is i go into and did it by hand. Yeah i would like look through the course search populated by hand. Okay. So if there was some way of Automating that if not like i can look into that myself in the future but

Nolan yeah we definitely could because i've written scripts before that will go through and create the like database fields and stuff but I don't know what that looks like.

Kyle Exactly.

Austin So does UNO publish it. Outside of mavlink.

Nolan Probably not. I don't think that it's going to be public data. I think it probably would have to get it from the university, maybe.

Kyle Yeah, that's, that's the question. I don't quite know.

Nolan Okay.

Kyle So that's why it's, it's not a requirement it's something to maybe think about.Would it be possible? But it's definitely not a request.

Nolan Yeah, sometimes i don't know, once we start getting the main stuff done, if we have like an extra bit of time, i can go to the registrar and ask or if you want to. You can just

Kyle yeah, i could talk to and i could also talk to Dr. C there there was a group i know. Last semester. That was doing like a computer science registrar system.

Nolan Yeah, yeah, i've seen that. I think that's what ethan did. Also, he did the schedule. Yeah, he did it with Con but they scrapped theirs. I'm guessing because they were rendering all their back end like aligning and padding and i like no, no other front end stuff was being calculated in the back end. It was very, it was very interesting setup. It was all c, sharp. So, That's okay to do, but I guess i didn't follow up with it. They had another takeover but That's a tough. That's a tough one, because they were having to write, like, genetic algorithms to, like, determine if they were, like, if this room was taken by the next best fit for the room, and i remember at the end of the semester, Ethan was almost like, i'm just, we just can't do this. Like, this is just too much for, you know, four months of work.

Transcript of Initial Client Meeting

Nolan	Yeah. So, i I don't know how possible it is, but Yeah, thatd be something id like to add making a little bit easier for you so you don't have to go through manually. Add every single professor update things, if things need to be updated.
Kyle	But again, if if you don't do it, no worries. I can always try and figure something out of myself.
Nolan	Okay. Um, we have to go because the class start.
Kyle	Yeah, i'm sorry. Taking you right after the time.
Nolan	Yeah. I'll make a chat and then, yeah.
Kyle	And then i can Get back to you on in there.
Nolan	Cool.
Kyle	And we'll keep in touch.
Nolan	Yeah
Austin	perfect.
Nolan	And let's schedule some time in mid october.
Kyle	Awesome
Nolan	Cool
Kyle	Have a good rest of the day
Nolan	Yeah, see you later. Yeah.

Transcript of 19 October Client Check-In

Transcript of 19th October Client Check-In Meeting

Computer Science Learning Center Website Project

Cringe Coders Team

Austin Bailey
Mya Bell
Lindsey Langdon
Nolan "OG" Gregory

Transcript of 19 October Client Check-In Meeting for the Computer Science Learning Center Website Project

Dr. Harvey Siy; College IS&T
University of Nebraska - Omaha
Capstone Course
October 20th, 2023

Transcript of 19 October Client Check-In

Austin	The recording is live last time. It was 11 pages of transcript. Let's see if we can
Nolan	So we talked about using the rest framework yes, and I built that out so we have like the back end is all restful
Nolan	And just use this it's pretty cool. I've never done that before so it's pretty fun to actually write the API stuff is it
Kyle	AWS or is it so with hosting it?
Nolan	Okay, but the Django rest oh, yes
Nolan	Yeah, so the you can like I've written all the endpoints for it so all the URLs work
Nolan	I can query stuff perfect
Nolan	Which is pretty pretty cool to write yeah, and then we did all the react front end stuff
Nolan	I'm still working on that just because
Nolan	I've never done react before so I've had to learn react and it's like I
Nolan	Don't like JavaScript to begin with and it is like JavaScript, but
Nolan	Worse, it's very there's like things that just don't make any sense in it. Have you ever used it before I?
Kyle	Don't remember if I've used react. I mean I've used like no JS. Okay a little bit
Kyle	So I mean I have familiar
Kyle	Familiarity with certain types of JavaScript, but I don't know
Nolan	React necessarily okay.
Nolan	Yeah, there's just things that don't make any sense like there's a there's like state like each component has like
Nolan	Components just like any aspect of a screen essentially like you could have anything
Nolan	Component and they could have state so like whether they're like on or off
Nolan	But state isn't

Transcript of 19 October Client Check-In

Nolan	saved through page refreshes
Nolan	Which I guess would make sense
Nolan	And then there's also
Nolan	Yeah, so there's a couple different ways that you could do that, but state also does not get
Nolan	Instantly updated so if you call the update function on the state what gets shown on the screen is the previous state
Nolan	That was not updated so I'm trying so the thing I was facing is you know when you're typing a ticket
Nolan	You can only submit X amount of characters
Nolan	I was sent to like a thousand twenty four right now
Nolan	And while they're typing you know like Twitter will have the thing with a like countdown
Nolan	Yeah, he's to at least I haven't used Twitter in a while
Nolan	I was adding that and it was showing the previous count so
Nolan	I would type something it would still say
Nolan	1024 and then you type another
Austin	It was updating the last one yeah, it would show the last one which was
Nolan	Pretty tough to figure out and react does not make that very easy
Austin	I feel like that would almost be better to have a JavaScript
Austin	Running in the page on the client side rather than have server communication on that
Nolan	Yeah for some small things like yeah, yeah, and there's we use feet for it instead of woodpeck
Nolan	So there's some things that get rendered out that way
Austin	I have dealt with some front-end rendering because there was a remote code execution

Transcript of 19 October Client Check-In

Austin	I've been working on for a while and the way they do that is they have an object. It's called double underscore view state and
Austin	You you hold that basically cookie on your machine
Austin	And then anytime you send a get or a post or anything to that host
Austin	You serialize the data and then encrypt it with a public key
Austin	which
Austin	That's practice should be instantiated when you can create a session it creates a new private public key
Austin	You encrypt it with public key and on the server can decrypt it you don't do it properly you get remote code execution because you're
Austin	deserializing actual data and it
Austin	Becomes a very big issue very quickly, but if if we do it right, that's what I've seen done, and that's by Oracle
Nolan	Mm-hmm that was actually a lot of people didn't use the rest framework at the Django one at first
Nolan	That's what that was a problem that they had was that exact thing
Nolan	But they fixed it now obviously
Austin	They'll come in and fix all the issue you've run into that is if you have dynamic keys determined by session
Austin	You have a cluster of servers and there's inter server communication
Austin	they can't have dynamic keys and so then you have to have a static key and
Austin	A lot of times what ends up happening is software companies push out
Austin	Software and they populate it with a predefined key and then people build up these instances and then don't
Austin	Rotate the keys with the session and if they leave it at the default, then you can deserialize it and payload it
Nolan	Yeah, the other group did they had session keys and I'll probably just use what they had they had

Transcript of 19 October Client Check-In

Nolan	They had 30 minute session keys, which I feel like is pretty I think that's what the old
Nolan	Portal had was it might have I don't remember there was something about the old portal
Nolan	I've been trying to remember and I cannot remember like certain aspects of it
Nolan	But
Nolan	Yeah, we got the Django stuff built out and it's been a that's been a hurdle just good said
Nolan	Yeah, that's something I've never done before and I didn't like JavaScript to begin with so
Nolan	Yeah, yeah
Nolan	callbacks
Austin	Only through Linux. Yeah. Yeah, there's
Nolan	Basically everything with the state is it I mean the way to solve that one the thing I was talking about was
Nolan	Callback functions because all the state like whenever you saying like set the state
Nolan	Asynchronous call to set it. So yeah, like a
Nolan	Fetch or promise or whatever where it's just saying oh this is gonna happen at some point
Nolan	But you don't know when so you can't count on the state being
Nolan	updated which is just that was another thing with getting the
Nolan	The API calls and stuff. That was also tough because it's all promise-based and I had no
Kyle	It's all it's all more more more event based in general. Yeah. Yeah
Nolan	So that was something I had to learn was promises and the concept of you know
Nolan	It's not actually there. It's just a promise that it will happen
Nolan	So I was like, why is this not updating and I had I asked clip because clement experience with it

Transcript of 19 October Client Check-In

Nolan	I think then why is it's not updating and he's like, oh cuz it's just a promise that it will happen
Nolan	You're never like actually not calling that promise. But like yeah, I guess calling that promise. I'm like
Nolan	But we got it we got it working with all the promises and stuff and then we have a
Nolan	The
Nolan	Download thing is working, but it's using a library that was last updated in 2022. So I'm not like super
Nolan	stoked about that just because with this like
Nolan	Another thing I mean, I'm just gonna bash on react the whole time when it changes so often that you know
Nolan	A year feels like forever. Uh-huh. Usually I feel like a year shouldn't make that big of a difference
Nolan	But I feel like it makes a huge difference with this
Austin	It depends on what kind of open-source project notepad plus plus
Austin	Once every three months that is fine with me, but like all the the web things like angular react
Nolan	They're all and they all instantly change and they change it. Yeah. Yeah. Sorry CI CD. Yeah
Nolan	yeah, there was a there was a
Nolan	One thing I was using for this that hadn't been updated for two years and I didn't realize that I built it all into the
Nolan	Application with like a rich text editor in the ticketing system so that people could put in code blocks or whatever. Mm-hmm
Nolan	I was like, okay, let me see
Nolan	You know, there's an issue. So I was gonna post it on github and I was like there are
Nolan	1020 issues on this github page. I probably should not use this. So I got rid of it

Transcript of 19 October Client Check-In

Austin	I got hit by the lottery bus. Yeah, they got hit by they didn't want to do open-source anymore probably because
Nolan	There was too much work before
Nolan	Oh, so the current courses I was thinking of a way that we could I haven't reached out to the
Nolan	Department downstairs yet. Okay registrar's but
Nolan	Is there like any data that you want like seeded into the database by default?
Kyle	We could do it that way, okay because we could essentially just take
Kyle	The list of courses that we see the database to start with
Kyle	Because we could come up with that
Kyle	Like I could even you know
Kyle	Or one of you could just pull like a list of the courses. Okay in general because that's what I've been doing
Nolan	It worked now just in like development just to test things. I've just been using like faker which is like a JavaScript thing
Nolan	Just fake data
Nolan	But that's obviously not really good for production because you don't want random professor names and stuff
Austin	I still have that listed as like a stretch goal
Austin	But I already started pre populating the report saying this is future work
Austin	And we haven't really scoped it yet because I don't know that the register has any structured data for it
Austin	So I mean at least not public I know that the downstairs
Kyle	Because I think it was last semester there. There was a team capstone team that was working on a scheduler
Nolan	Oh, that was yeah, Ethan did that wasn't Ethan. Yeah, he did that and there was another group
Nolan	They did it last semester, but they didn't finish it. Oh, okay

Transcript of 19 October Client Check-In

Austin	It's a reoccurring topic to try to write genetic algorithms to book rooms for classes and to that's the toughest do all of that stuff
Austin	Yeah, every three years they come back
Austin	We need a new genetic algorithm and they take a non bioinformatics major and they say why don't you do it?
Nolan	Yeah, I don't know how to do that. Best best match stuff like that. That's what I was trying to do
Austin	It's genuinely higher-level
Austin	bioinformatics because all of their algorithms are heuristic because if you try and do
Austin	Computational algorithms that are well definite then get into computational issues very quickly. I mean, that's that's a machine learning issue
Kyle	I mean as well and so it doesn't always have to be bio
Austin	Most of those algorithms aren't even machine learning they're
Austin	They're holistic and heuristic algorithms
Austin	but
Nolan	Like that's a I don't know
Kyle	I remember in what I I don't know if the AI class now still does it but I know when I took it
Kyle	We talked about like heuristic algorithms and work with others. Yeah. Yeah
Nolan	Yes, I don't know I think so the way that they got it
Nolan	I think it's something that was a big CSV that they had or a JSON. Okay, either which I mean
Nolan	They're both easy to go through
Kyle	Yeah, I mean and we could honestly I could just go on I mean after here
Kyle	I could just go downstairs and try and talk to Carly
Kyle	And see if like what she uses for it if she just has like a big file

Transcript of 19 October Client Check-In

Kyle	Okay, that like she could just even just send me every year and then yeah every semester
Nolan	I'll be honest to have so I could test and just test loading data in from that and it and if there's a
Kyle	possibility then we could make it so that like
Kyle	Upload a CSV when you want to create a new semester and then you could just go through that data
Nolan	Yeah through the data be perfectly the only thing I was pivoted to other tables probably. Yeah, I was thinking and
Nolan	Yeah, you could do that based off of this but like if there's professors, you know
Nolan	We load all the professors into the database and some professors let's say like I know Santos isn't teaching C
Nolan	I don't know if he's teaching anymore. But right now I don't think he is
Nolan	Yeah, so like if or if a professor were to leave I
Nolan	Don't know if you would want those deleted from the database entirely because I don't know
Nolan	I mean, I think it's fine to leave them in especially because then you you can keep legacy data in yeah
Nolan	I was wondering I was worried about losing like relationships like
Nolan	So it's fine to leave it in okay, probably
Austin	What you'll do is it'll be class
Austin	section teacher
Austin	and then you'll have
Austin	Fall or spring and then the calendar what it and then you'll filter if you're gonna populate a form with a ticket
Austin	It's gonna be current semester. But if you're gonna pull historic data, you're gonna pull
Kyle	The way that I think the the old database was set up was that it was all based on

Transcript of 19 October Client Check-In

Kyle	Semester and so like you would have to recreate a class every semester because it didn't have that information
Kyle	So it depends on how you want to do it if you want to base it on semester like that
Kyle	Or if you want to base it on class and then add a new semester with a new professor
Nolan	Okay up to you on how you yeah right now. I have it as like sections a section
Nolan	We'll store a course ID professor stuff like that and it has a semester in it as well
Nolan	just as like an enumeration class, but
Nolan	We could do the other way that's the way that they did it before that's the way they've worked
Nolan	So I'll probably just stick with that and I don't know too much about database design
Nolan	So that's all I never took like a class. I didn't help you. There's something like this. It's
Austin	Not terribly difficult and there's there's no real concern. You're not dealing with such large data
Austin	Yeah, it matters. But if you want to get into the relational algebra just to look cool. I can teach you that as well
Nolan	Yeah, I just don't want to lose
Nolan	Like have something get deleted and then that propagate and delete everything and the database like everything's like an on delete
Austin	So just just make a backup of the database and then we can quote unquote normalize
Nolan	Just right through to another database and then you know right through to three other databases just for data redundancy
Nolan	So, yeah, we got that and then I guess with the
Nolan	Courses, I'll be really cool to have just so that we could test it. Yeah, I can I can talk to currently about that here
Nolan	Okay, cool. Yeah, just write like some sort of script that would go through and
Nolan	Do that

Transcript of 19 October Client Check-In

Nolan	Loaded into the database and stuff and I get if it is based on semester previously
Nolan	I can just rewrite what we have now to do that except my makes more sense. Anyway, I
Nolan	Don't know the way the last group did it. I was going through all their models and stuff. It looks like they I
Nolan	Think they did it based off semester two from what I could tell them because they
Nolan	how everything was linked to a semester but is that because I think
Kyle	Because I because I know that Clem was really familiar with the system
Kyle	So I wouldn't be surprised I know Clem talked to like Jean as well. Okay, she had been working on stuff. So yeah
Nolan	yeah, there's there when it comes to like the semester was it like
Kyle	Was it just spring and fall or some spring fall summer, okay
Nolan	And then do you want to do like a J term or no, I
Kyle	Don't know how long if that's not like I mean it sounds like it might be sick. Oh god, there's a whole thing
Kyle	We might be going down to 14 weeks
Kyle	Semesters really?
Kyle	We're technically at 15 and then the 16 is the finals week
Kyle	but we might be losing a week in each of our semesters and just because they want people to be more stressed or
Kyle	Because we're going down to try and keep the J term and also expanding the summer and right now the schedule for I think
Kyle	2026 as the current schedule stands doesn't work. So they're like we need to make some sort of change and
Nolan	Term that's the change
Kyle	See apparently it's been good for you. I know
Kyle	Really Lincoln and Carney hate it, but it's been good for you

Transcript of 19 October Client Check-In

Austin	But I like this so my critique is there's nothing you can teach me in three weeks that I will retain
Austin	Three weeks after that. Yeah, they were teaching a calculus class during J term
Nolan	They were saying this is not enough to learn calculus
Austin	It is because when I was in my freshman year, I did calc 3 in 28 days and that includes weekends
Austin	So it was 20 days at least count three was
Nolan	More fun than how to play it is cuz like I know they teach oats here or zaps intact. Oh, yeah
Kyle	I think that's about the only thing we can teach other than like
Kyle	Hey, we're gonna teach you like a random programming language that maybe you're interested in
Nolan	Yeah, it could be cool for stuff like like fundamental or like yeah, like stuff like
Austin	I can't honestly tell you the last time I a have done calc 3 but be how do we even do it anymore?
Nolan	They should do I guess they already do have summer terms. Yeah. Yeah, it's just like glorified summer
Kyle	Well, it will essentially what the idea with summer is then if that summer will be the same length as fall and spring
Nolan	Oh, so they're not gonna do I like sections anymore. There's gonna do well
Kyle	You could so we have trimester could still do sections in summer
Kyle	But the idea would be that it could be a full end of the plans. I'm just sorry
Kyle	I don't we don't need to have this on them. Oh
Austin	Sorry, sorry. Sorry. Sorry. No, it's all good. It transcribes automatically. So I just touch it up
Nolan	Okay, that's interesting. Yeah, I think the first time I was here was like 16 weeks or something
Kyle	It was it was originally a 13 week summer
Kyle	with

Transcript of 19 October Client Check-In

Transcript of 19 October Client Check-In Meeting

Kyle	As far as summer courses 13 weeks summer with ace with a one week break in the middle and then to six weeks on
Kyle	other side and now it's just 12 week which with the to six weeks no break and
Kyle	Now with J term we've lost the break before between spring and summer. We just didn't have a break week this year
Kyle	Which is fun. That's insane. Yeah for the professors
Nolan	I mean, it seems like from what I saw like in the discord
Nolan	It was like it was like no turnaround like it just like relax a little bit after grading
Nolan	It was like the week after things like guys. It's literally no break. That is fun
Kyle	But yeah, so I I don't know about J term I mean
Kyle	Honestly, they don't even consider J term a separate semester. Currently they consider it spring semester J term
Kyle	Okay, so so we could just throw it in spring semester. Okay, and so we don't think you need a separate
Nolan	Okay, cool
Nolan	And if something did need to change I guess it won't be hard to change the scheme I just to support like
Nolan	Spring
Nolan	So, let's see I've got I
Nolan	Guess I'm going to show you what we've got so far
Nolan	So we have
Nolan	This is just a default like landing page and then this is like this is all
Nolan	Not everyone would see this right? Yes, you were an admin you would see that
Nolan	And if you were a student you would have a second one is just called
Nolan	Like general I mean they have a place to submit a ticket
Nolan	And then like look at the zoom hours and stuff like that. All the elements are are back

Transcript of 19 October Client Check-In

Nolan	Yeah, and then so on like the admin panel, you know, you have your dashboard where you would see everything
Nolan	So this is just dummy data right now, obviously, but you'd be able to select up here like a time frame
Nolan	So you'd say okay. Well from this length to this length
Nolan	I want to see the amount of tickets closed how many were open success and then you know a breakdown of that
Nolan	And then I guess here would be the same thing
Nolan	I've really touched this part too much just because I want to get feedback a little bit. What's the what's the tutors tab?
Kyle	What would that miss?
Nolan	So what I was thinking is that would show active tutors like ones that are right now because down here
Nolan	You would also have a tutors tab, but then my owners my thinking here would be this would be like where you would assign tutors
Nolan	Like give tutors their
Kyle	Tutor privilege. Yes. Yes, because because right now because the way that it used to work
Kyle	Is that there was like a place to Oh enter in an email address or whatever and then give them okay tutor privileges
Nolan	Yeah, see that's I was thinking like we could do that here
Nolan	Sure, and then here you would be able to set the hours for the CSLC if you want
Nolan	Sure, and then these are just two placeholders because I haven't I thought about what else to put here. Yeah, they're
Kyle	So if you go back here to home
Kyle	Okay, so that announcements that would be something that I could set as it I haven't been as well yeah
Nolan	Yeah, so right here you'd be able to set your announcement so you could say like oh I want to put my like that the

Transcript of 19 October Client Check-In

Kyle	Title the well because what what I used to do is I would set an announcement every time that we were
Kyle	Closed for a break or something. Okay
Kyle	We haven't been able to do that because it's just a bit weird now on the website
Kyle	But I just set them all at the start of the semester. But okay
Nolan	Yeah, so you'd be able to say like, you know fall break and then closed for that. We
Nolan	I was working on this at like 3 last night. So don't know if this is still gonna work. This was okay
Nolan	Yeah, that's just my default like page right now. Please don't burn yourself. Yeah. Oh, no, don't worry
Nolan	but yeah, this is the one that I was trying to
Nolan	Fix last night where I'm typing up you can see it's not working at all anymore. Oh
Nolan	I see. Uh-huh. When I change the state I change the state of this. So this is it
Nolan	Oh, so when I select one of these, you know, you have general informative alert warning. Okay, so if I
Nolan	Click on it now, it's not gonna work
Nolan	It's probably gonna say that I'm trying to access a state that doesn't exist
Nolan	I'm trying to change the state or update it to something
Nolan	Sure, and then you have your date range to it. You could say like I want it to be displayed
Kyle	Yep, only between these dates. Yeah, great. And then you could say here if you want those dates to be shown actually on the announcement
Kyle	Okay, cool
Kyle	You know if that that ever needed to happen like we're gonna be closed. I'm sorry. I just I just I just recognized
Nolan	So what's that? Oh, it's like I like the feature of get help where you can see who did something

Transcript of 19 October Client Check-In

Nolan	I was like thinking in my head. Oh, well, what if someone came in and they're like, oh I had this ticket on this day
Nolan	But I don't remember who it was, but it was horrible and you're like, oh, okay
Nolan	Let me go right now. There's nothing there
Austin	You made a blame. Yeah, it's like it's a blame panel
Nolan	I mean, we don't have to keep that. I just I love that feature on get because it's so it's like it doesn't lie
Kyle	It doesn't lie. We we maybe want to rename it
Kyle	But but I I'd like the idea. But yeah
Nolan	Yeah, so yeah, we can we can change the name. I just kept it like that cuz I
Nolan	Just love you like I think it's awesome and then this is the download thing. So you can see here
Nolan	By default, it's like everything I want and that's the way that you had it. Also like the URL student email first and last name
Kyle	Oh, yeah, can we had shoot? I don't remember if I included it
Kyle	Can we add online to this because I forgot that was one of the things that I wanted to also yeah
Nolan	Totally this is super easy to change around to and then also one of my favorite is this one
Nolan	So custom lets you select anything
Nolan	So let's say you only wanted a certain professor or a certain section
Kyle	I mean just the section the section and course thing is something that happens all the time
Kyle	Like I'll have a professor asked me. Hey, how many students come in for this course?
Nolan	So so and then you could select I guess this would maybe make more sense also to have one
Nolan	I was like a certain semester. Yeah, I only want it from the 2nd of October to you know, the 28th of October

Transcript of 19 October Client Check-In

Transcript of 19 October Client Check-In Meeting

Kyle	Yeah, I mean, I I think that day range could work
Kyle	But I think having an option to have it be a semester would be nice
Nolan	Well, that would be super easy to add and then this, you know, you can export as a CSV in Excel or XML
Nolan	And this is just by default the name that it's just the date the words
Nolan	It just updates each day. Obviously
Nolan	And then this one this will be cool. I haven't added it yet, but this will essentially allow you to
Nolan	say
Nolan	Instead of I might switch these two because custom this seems more like an entity based than custom and then this one
Nolan	Because what I was thinking is you could say you could type your own or not type your own but access your own a like API
Nolan	link because basically with the API you can like obviously do like the question mark and then say professor equals my professor name and then
Nolan	Some section and then sure pull all that data just by the way that is querying it
Nolan	These are the two that
Nolan	Important and if I have time, okay, I'll do that. Sure
Nolan	So yeah, then under the settings, I guess where you went over that and so for the tutor panel, you have the dashboard
Nolan	So it's very similar. I right now they're not padding
Nolan	I was working on this because I'm probably gonna make this only I just called it pulse
Nolan	Another github thing because I just like that sure and I was thinking this could be 24 hours
Nolan	So it'll show like the total tickets over the past 20 and they can't change this because I don't want them to be able to
Nolan	See, you know two years worth of data. Yeah, you know to it or trying to crash the database every two seconds

16

Transcript of 19 October Client Check-In

Kyle	Yeah, I think 24 hours is good
Nolan	Yeah, so it'll show them their total tickets close and just in that amount of time and then here
Nolan	I'll probably get rid of this total ticket overview. I might keep recent stuff, but yeah, but I I like the idea of the recents
Kyle	especially because you know, sometimes we'll have someone who like
Kyle	Comes back in or something and you're like wait. Wait, what happened there? Yeah
Nolan	But yeah, I'm just trying to think so right now what to put here, but here this is cool
Nolan	So I have populated database. You can see these are just be a brand them. Yeah. Yeah, so these are actually names
Nolan	It's just these are the ID IDs of the news. Yeah, so I just need to when I'm serializing the data
Nolan	I found that last night that when I'm posting
Nolan	Serializing is different than what I'm getting so I have to write the serializers
Nolan	But again the post because I was like why this post was working last night then I serialized it now
Austin	It's not working. Is there not an innate serialization function that you can use writing your own there. They have one
Nolan	That's called like, you know model serializer or something
Nolan	But whenever you're trying to serialize the strings because these are all foreign keys, so they're like relationships
Nolan	So it's just getting the ID associated with those
Nolan	So that's what it's showing on the page
Nolan	It's when I'm trying to make that post request
Nolan	It's trying to take in instead of the ID which is what I need to send in
Nolan	It's trying to take in the actual string of the professor, which I don't want to send
Austin	So you're not serializing you're sanitizing in the no no and from getting it

Transcript of 19 October Client Check-In

Nolan	Yeah, I'm serializing from the database to
Nolan	From database to JSON sending it and then from this way it's going JSON back to
Nolan	Whatever is going into the database. So with the post, I don't know
Nolan	That's the one I have to fix because that's the string issue that's happening right now
Nolan	And then yes is just paper data. So you can see there's an error right here. Cuz
Nolan	This will never happen
Nolan	And that's why you see like you shouldn't be seeing this because this once you make a ticket actually in the order
Nolan	But there's a section to make the ticket and this gets auto populated
Nolan	So these are right now these are just like
Nolan	Empty and they don't okay functionality, but I'm thinking
Nolan	If I'm trying this is what I was trying to think about with the old system
Nolan	I don't remember the process of claiming a ticket. I think yes opened up
Kyle	So it opened up like the edit screen of the ticket like you could do this differently if you want to but what it did
Kyle	Is it opened the edit screen of the ticket and then you would have to go down and then like find your name in?
Kyle	The list of tutors and put your name in for in the claim box
Nolan	Would it be cool if I just said like here if I just the person pressed it and pressed it
Nolan	Oh and said like you are gonna claim the ticket as then the person saying that. Yes. Yes
Kyle	The one thing you might want to account for though is multiple people claiming a ticket because if you know multiple people end up helping
Kyle	That might be something to consider as well and then
Nolan	Yeah, and then someone can close the ticket without having claimed it as well because I know there's situations. I'm sure you're familiar where

Transcript of 19 October Client Check-In

Kyle	Students like help someone they finish up but then they're like they're leaving and they forget to close the ticket
Kyle	But then that person's gone
Kyle	So someone else can go in and like close the ticket for them. Okay. Yes kind of the idea
Nolan	I guess once they're in the section where it's open or you're going to close the ticket
Nolan	Maybe it opens up a more comprehensive thing
Nolan	It says like, you know, then you can slap the primary tutor or the assistant to the secondary whatever it's called
Kyle	Yeah
Kyle	I mean I and I'm even fine with having not just primary and secondary like if you want to have more even more people on
Kyle	I think that's fine. You can even just have that as like a list and then you like list the tutors on it
Kyle	Okay, instead of having separate
Kyle	Pieces of the database the separate columns. You can just have like a single column then just like yeah populate with that whoever totally
Nolan	Yeah, we can do that. That way they could just select any arbitrary. Yeah people that helped
Nolan	And yeah, I guess you could then trace back those tickets to the
Nolan	So yeah this so I guess on this first page this would just be like a claim and then instantly
Nolan	Claim it or show some modal. I sure it would be more comprehensive if you're trying to close it
Kyle	Yeah, especially because then close you want to account for the potentially I mean potentially the time taken
Kyle	I mean that could be automatic though when you close it
Kyle	But then you do want to figure out that successful on successful. Mm-hmm. Yeah. Yeah, and actually if I

Transcript of 19 October Client Check-In

Transcript of 19 October Client Check-In Meeting

Nolan	So, let me let me go here real fast and go to the
Nolan	It's gonna go to the admin page
Nolan	Or this is like the Django admin default screen so if I go here
Nolan	and then if I go to tickets and I say that this one was
Nolan	completed started and was successful and I
Nolan	Say today
Nolan	Now today now say I go back here
Nolan	Refresh it should show right? Okay here. So
Kyle	What's it going closed yeah, oh yeah so close right now open and close should be no
Nolan	But yeah, it will show and I did add a feature here I go back
Nolan	Oh my gosh, there's so much
Nolan	That thing at the react projects are just they have so much code cheese, so I go
Nolan	Should be something here. I had it. So if it was successful, I guess I could just tell you that's a new screenshot
Nolan	If it was successful, so it'll say, you know, these will get populated down so it might look that's something like this
Nolan	Kind of looks weird all the way the top
Nolan	So I don't think it might start at the bottom and they get pushed to the top
Nolan	Progressively because this gets filled down all the way. So this will say claimed by and that's like their idea
Nolan	Yeah, and then it'll say closed at and then a time time and then it'll have either a little check mark or an X
Nolan	Oh, sure. I'm sure just saying if it was successful or not. Cool and then
Nolan	Under
Nolan	Schedule that's there's nothing there yet

20

Transcript of 19 October Client Check-In

Nolan	We actually I do have this implemented like a table that shows everyone's times that they work
Nolan	But I just don't have it here because I had it as a different view and then change that
Nolan	And then this is the zoom thing. This is pretty cool. I think that this could work Wow
Nolan	It's working right now if I were to say that I'm an active tutor and I'm working this would show up instead of red
Nolan	It would be green and say one tutor. Oh, yeah
Nolan	And then this is just a zoom like obviously I wanted this to integrate like actually open zoom, but if you go here
Nolan	It'll just like sign in sign ins. Yeah
Nolan	Yeah, I'm not too there's like an SDK you could use to like integrate zoom into the browser and I was like that is too
Kyle	Yeah, that's too much. And then ours this also
Nolan	It's it's working. It's just if you were to look it's probably gonna say that there's a API error because yeah, I
Nolan	Changed the API endpoint for this one. So it's it still works
Nolan	This is just saved in the database as a random table
Nolan	Which I don't know
Nolan	If that's the best way to do it with the hierarchical database because it kind of just sits there without any
Nolan	Relations, it's a flat table. So I don't know if that's the best for what it's just the hours
Nolan	So like if you wanted the hours to change or something
Nolan	You would go in and change them in the admin settings
Nolan	Which then would update them in that database and then this queries them from the database
Austin	Are you talking like the weekly hours?

Transcript of 19 October Client Check-In

Austin	having that
Nolan	Alone is perfectly fine because it's really small. It just doesn't have like any relations like
Austin	potentially something like that you would also tie into
Austin	Scheduling for tutoring hours
Austin	Okay, so you might have some limitations when you go to schedule that you can't have them before open and after close. Okay
Nolan	Yeah, so oh, yeah, then you have all my favorite part right here I added
Nolan	But here you can see now we have
Nolan	It's pretty pretty cool
Kyle	So here's what's the other account pieces? What's oh the message?
Nolan	Yeah
Nolan	So profile is just I'm thinking that this is just going to be like I feel like everything needs to have like something to see
Nolan	So just like the
Nolan	basic stuff that you would like their name maybe if people wanted to go like
Nolan	If they wanted to add like are their pronouns to their page or if they wanted to say, you know
Nolan	This is my because I don't I don't want them to be able to change their you and oh student
Nolan	Yeah, say I want to go buy a different name like show a different name on my tickets
Nolan	They would be able to update it here. Uh-huh, but you'd still be able to query it as
Nolan	You know
Nolan	And then messages this is kind of crazy I am I actually added web sockets into this so it does support real-time messaging
Nolan	But the whole point of that was if I'm on at my ticket page and I make a post request

Transcript of 19 October Client Check-In

Transcript of 19 October Client Check-In Meeting

Nolan	What that then does is through web sockets will send from my client to the server something saying I just made a post request
Nolan	issue back to all the clients that have stateful components that they need to re-render those components because
Nolan	State from the database is not changed. So we're gonna call that and
Nolan	So basically if I'm here and I'm you know, maybe I'm a tutor and I'm sitting here talking to my this is empty
Nolan	If someone makes a post request, it's this page is gonna update one of these components will update adding that ticket right here
Nolan	Okay, without a refresh being required. Okay. So the whole point is I did not like the refresh system
Nolan	Portal. Oh, yeah. Yeah, I think you'd mess much of that. Yeah. Yeah, so I was messing around with it and the
Nolan	I was thinking oh I could just ping the database every so often
Nolan	I was like, well you have 50 people on their computers to just have this passively open
Nolan	I don't want them to all be hitting the database with the API calls
Nolan	Yeah, like every ten minutes or five minutes or two minutes or something
Nolan	So this I feel like it's a better way to only does it when someone actually does something sure
Austin	It's just one time. We'll see if I can make a dose out of it. I doubt it
Nolan	It seems pretty I like looked into the package that I use. It seems pretty good
Nolan	So this is just how I like what I'm using right now to populate all the stuff in the database
Nolan	And this is just gonna be a little test field where I can test out the API call endpoints
Nolan	Okay, mostly just for postman stuff and then this will be the same thing like this testings
Nolan	But I haven't added these yet just because I was working on actually gonna be API working

Transcript of 19 October Client Check-In

Kyle	What is the student view look like then? Yeah, so if I go here and I go to my dashboard
Nolan	Oh
Nolan	Yeah, the I didn't realize that there were gonna be so many files in here it's been like
Nolan	kind of overwhelming
Austin	Front-end development does that? Yeah
Nolan	The back end looks so small and dinky in comparison, there's just so much stuff here
Nolan	Yeah, so this would be the student views
Nolan	Okay, actually you wouldn't have the development portal that one's just like sure present there right now. So they would have it. Oh
Nolan	Yeah, and then search I'll probably get rid of search. I don't think that there's not really I'm not gonna be used to more
Nolan	What is the search that would be like if they typed in like search or ticket
Nolan	Okay, like, you know submit a ticket or something, but I feel like it's such a small application doesn't really
Nolan	Shooters, oh, yeah, this this is a this is the old this was my very first mock-up here
Nolan	It's funny that it still works. I'm very surprised that this is still working
Nolan	Because yeah, this is just saying these are showing tickets actually not even yeah
Nolan	So this is the very first mock-up I had you can see how much everything's yeah
Nolan	But yeah, this is the first mock-up. That's funny. That's though because because I'm curious I do wonder
Kyle	I mean we used to have a thing
Kyle	And I'm
Kyle	I'm torn on whether I liked it or not, but
Kyle	One thing we used to have was on the home page if you remember it would show

Transcript of 19 October Client Check-In

Transcript of 19 October Client Check-In Meeting

Kyle	Certain set of classes and how many tutors are available for yeah class
Nolan	The problem was people want to update update and update what classes they could help with eyes
Nolan	Yes, that is one of the big things that's actually I made that a field and tutors
Nolan	So tutors can add you know courses that they've taken and courses that they want to help with and then I assume that that's something
Nolan	That they can't change or that you would change and generally it was something that they would change. Okay? Yeah
Nolan	So I have that added. I don't know if it's something. Yeah, you want to implement
Kyle	Cuz because that was one of my only thoughts for like if there is a page like this
Kyle	That is talking about tutors open. That's what I think something that could happen on that. Okay, whether whether
Kyle	We want to implement that or not. It depends because like I said it, you know
Kyle	It was it was sometimes
Nolan	Really work in practice. Yeah, I think it's a girl. Yes. I would like to see it work
Nolan	But yeah, I don't know if it was
Nolan	Always working the best. I was trying to figure out a way, you know how like on discord or on Microsoft Teams or something
Nolan	it'll show like
Nolan	if you're
Nolan	Active like oh, yes. I really wanted to implement something like that. I just don't know how to do it
Nolan	I've been like looking into it and I have no idea how that yeah
Nolan	Yeah, well I was thinking in the terms of sewing people who like tutors that are like
Nolan	Currently working because yeah
Kyle	the old way of doing that was literally like you would have to go into your own profile and

25

Transcript of 19 October Client Check-In

Kyle	Click hey, I'm working and I'm in turn it off very easy to forget
Nolan	Yeah, so an automated way if you're logged in show you something like that
Austin	Probably be to query the shift schedule
Nolan	Yeah
Kyle	You just go off the shift schedule
Austin	You just got a look at the hours. I have some arithmetic
Nolan	Yes, I could work. I might just make a mock-up of that then but yeah this I don't think I'm gonna add this just for
Nolan	the fact that
Nolan	We've had things in the past from people knowing when tutors work. Yeah, that's that's that's one of the things that
Kyle	That's why I I wouldn't want names
Nolan	Yeah
Nolan	That's here
Nolan	That's what I I had this and the thought of oh it'll show all the people who are working on what teach it
Nolan	Of course, they can teach but but then I know that there's been things. Yeah, I definitely don't want the names to be shown
Kyle	But I would be fine with like courses and number of people working those courses
Kyle	People just like they're like certain tutors and like if they know that that's fine, but
Kyle	Sometimes it ends up being I like this tutor because I know they will give me more
Kyle	Than others will so they like they keep
Austin	They single you out because they know they're gonna get something from you
Kyle	Yes
Kyle	when it's like we've had that situation before we're like as

Transcript of 19 October Client Check-In

Kyle	Students like I feel like I'm doing so much like research to try and help them and it's like that doesn't feel like no
Kyle	That's not part of it. So I'm like trying to set those boundaries
Nolan	See yeah
Nolan	So I don't think yeah, so this is this page works right now what it's trying to do is query
Nolan	But this is that's where you create yeah
Nolan	And it does a drop-down of all the professors in the database and the courses
Nolan	Just so people don't have to enter it themselves. Just nice which is so so nice
Nolan	So they're not just like random course random course names super easy to query things get actual data from it
Nolan	What I was trying to do is like once they press, you know
Nolan	Oh, I'm doing this course then it would update the professor's and say okay
Nolan	Well only show professors that are teaching that course. Yeah or vice versa
Nolan	And I have like a prototype of that, but it's just a lot. It's like very chunky
Nolan	So I don't yeah know how well that would actually work
Kyle	I mean, I think it will be better once you actually like if you're taking the data from it
Kyle	So like once the course is set, yeah, you can kind of limit it from there
Kyle	I mean, there's there's some ways of doing that. Yeah, it just depends and then yeah view tickets
Nolan	What I was thinking is that this would only show the tickets
Nolan	I don't know why these are here, but this would show the tickets for the students for them. Yes
Kyle	Yeah, I don't think they need to edit after they
Kyle	Made it the only thing that I would like is
Kyle	And I don't know the best way to do this either so it might not be possible

Transcript of 19 October Client Check-In

Transcript of 19 October Client Check-In Meeting

Kyle	but the thing that I would be interested in is
Kyle	so remember when students would enter in their like
Kyle	Problem because they would have to enter in a problem and they'd enter in something random. Yeah, because they had to enter it
Kyle	Yeah, I don't know if there's a way that we could make it be more
Kyle	Actually useful information. Yeah, I actually random
Kyle	if it's
Austin	What?
Nolan	database here
Kyle	So there there would normally be like a problem field where it's like hey
Kyle	So tell us a little bit about what problem you're having and a lot of students would just be like
Kyle	Random letters just to fit it and then they'd be like, oh cuz I'm gonna talk to them about it myself
Kyle	But it's like well that doesn't help our records. Yeah
Nolan	I had I had so like one it's a drop-down field and it's in the database called issues
Nolan	You can make issues and then you know, you'd say what type of issue it was and then give some severity
Nolan	So if you wanted to I mean, this is just like
Austin	Something like that
Austin	I've dealt with because ServiceNow tickets are the same way you make it a required field and then people throw in crap
Austin	And they refuse to work with you and then you just get frustrated
Austin	Yeah drop downs and if you have a
Austin	Semblance of a taxonomy of we people come to me for this sort of problem and you can make three or four categories
Austin	Drop down and then give a fill-in so we have a taxonomy as far as like

Transcript of 19 October Client Check-In

Kyle	I don't know if you remember it like I'm struggling with this part of the process
Kyle	Mm-hmm taxonomy, but I feel like students don't really use it. Well to describe it
Nolan	Yeah, I was thinking like when I populate I have something that seeds the issue types and I just have like a homework
Nolan	Exam help or study help or stuff like that like very basic one or two word
Kyle	Just because one of the things that I really do want to get track of is I want to know which assignment they're working on
Kyle	Because that's something that people ask all the time
Kyle	They're like we which assignments do that have trouble with and then I have to go and parse through their problem statements
Kyle	Sometimes nothing
Kyle	so like getting a sense of like what us what the assignment actually is would be very helpful and
Kyle	Then getting some sort of issue they're having with it. Okay, just because again
Kyle	and
Kyle	Sometimes we would try and ask the tutors to help fill in and as well like if it if it's maybe not quite
Kyle	Great what they put in can you go and fill it in but that's always hard to do
Kyle	Especially if you're helping out people so I don't know what the best solution is there
Kyle	But I that's something that I think we can think about a little bit. Yeah
Nolan	I mean we could make it based off of like
Nolan	See, that's you just have to get the assignments from every single
Kyle	That's so hard
Austin	You have to make it a free form
Austin	Unfortunately, there's no way of doing it. I feel like for certain things like if you were saying maybe they selected data structures

Transcript of 19 October Client Check-In

Nolan	We know what data structures are going to be very specific things
Kyle	So I think that's something that we could build into in the future
Kyle	This is one of the problems
Kyle	Is that like if I can get better data on it and I can analyze that I can build it into the future
Kyle	But I feel like I don't know if I have that right now in a way that we could
Kyle	Individualize it per per course. Yeah
Austin	I can work with you on UX UI
Austin	It like I said, I've done some work with us and service now and stuff
Austin	Yeah, people are stupid and if you give them too much freedom, they give you bad tickets. Yeah. Yeah
Kyle	They take the easy way out right because it's like hey, I know that I just want to help and I'm coming in here
Kyle	I'm already sitting here in the center and you're just like hey fill out a ticket. It's like, okay
Kyle	I just fill out whatever I need to do so I can get help
Austin	You do the bare minimum to get through and so yeah, I'll think think about that
Nolan	Way to try to do that
Nolan	Yeah, right
Austin	Essentially you have to hold their hand through the process and give them a branching option
Austin	that pigeonholes them in a way that a forces them to think about it and if they're incapable of thought which
Austin	Fortunately, there are a handful of people in this world that get by by that
Austin	Then you at least have the the tree structure to go off of and it's
Austin	bucketed in like one of 16 ways and then
Austin	You can do further analysis from there and kind of fill in the gaps. Okay

Transcript of 19 October Client Check-In

Transcript of 19 October Client Check-In Meeting

Kyle	Yeah, so I can I can get you that old tax on me. I'll go find it and get you that
Kyle	Yeah, so so at least there's like a taxonomy to use but then yeah to kind of think about
Kyle	What what else to do there because that's
Kyle	Just an issue that's an issue. So it's universal. Yeah
Nolan	Okay, I think that's mostly it for now just crack what what we have
Nolan	Is there anything else that you want that you'd like to see here?
Kyle	I mean just some of the stuff we had talked about so like this
Kyle	Or at least showing showing like what classes are currently
Nolan	Yeah, something like that works
Nolan	And then I guess I could pull back up these other ones as we can see
Nolan	And then I've been so claims group did all of the
Nolan	What do you call that like SSL or whatever?
Nolan	Certifications and oh, yeah signing which is insane. So I was looking at that and
Nolan	That is tough and I was like how'd you do it and he just laughs he's like that is
Nolan	One of the hardest things I've ever done. Oh
Austin	Trying to do certain work
Nolan	Yeah. Yeah a little bit. I don't see I haven't delved too much into it yet. I can help you. Okay, perfect
Austin	Yeah, because there's two people that do it who sit right next to me Todd
Austin	Oh perfect, Josh, but I can help you with SSL certs because if you use 1.1 or 1.0
Austin	I will beat you to a pole. Okay 1.2. I'm sorry. Yeah. These are just yeah dates. Yeah
Kyle	one thing that that might be nice here is
Kyle	Breaking down my class as well because that's something that I do on my own anyway. Oh

Transcript of 19 October Client Check-In

Transcript of 19 October Client Check-In Meeting

Kyle	So like I I mean I create it and see it in my own visual form
Kyle	But if it's available on here, it makes a bit easier. Okay, so I mean if you if you yeah
Kyle	And I you want to otherwise, it's fine. Okay, I do it anyway, but
Kyle	Announcements good downloads great. Oh, yeah. Sorry, not that one
Kyle	Admin settings. Yeah, kind of see where those go
Kyle	Yeah, otherwise, I think it's I think it's good. I mean that schedule
Nolan	Yeah, yeah, this is like every single view so this would be like
Nolan	Okay, awesome, so
Nolan	And I'll change this
Nolan	No
Kyle	Cuz cuz what was what's it doing exactly again?
Nolan	It's looking for it would be saying like maybe you had a tutor who said they had someone that wasn't good
Nolan	You would basically find that it would show you you'd let you query based off specific things and I'll show you
Nolan	This is a very like get blame ask away
Kyle	Like an advanced search or something. Yeah, basically
Nolan	Yeah
Nolan	And then
Nolan	Yeah, I don't know what else would have to be changed through here
Kyle	I'm guessing that there's probably at least something but we could get yeah, we could talk more about them later
Nolan	Okay, so I'll work on that and then I want to have some my my idea of what kind of trying to crank this out
Nolan	Right now is because I want to have it done by the by the very very very beginning of November like the actual

Transcript of 19 October Client Check-In

Nolan	Okay, all of this done so that we can focus on getting it hosted sure cuz that'll probably take
Nolan	That'll take a little bit. Yeah, so I want to make sure we have enough time for that. So I'm I
Nolan	Think that this will be done by now. I'm not certain that it will if not sooner. So that gives me ten days
Austin	I feel like and I'm not terribly concerned because after
Austin	Probably the fifth or 12th of November my schedule should clear up a little more so I can help you with
Austin	Implementation. Yeah. Yeah, hopefully
Nolan	We have it up and getting tested by that just so that we could see it if people like if the tutors
Nolan	Suggest anything up there like oh we have this bug
Nolan	Yeah, but so if you want to meet some time around yeah, yeah if we want to try and meet
Kyle	Really maybe we make it maybe we make it the second week just to be a bit more clear there
Kyle	Is it is it still like Thursdays are the best day for you all or is it Tuesday Thursday?
Kyle	Before or after class, I mean it says Election Day, but we don't have elections this year
Kyle	So I think Tuesday the 7th would work for me
Austin	I'll still come in with
Kyle	Pretty sure we don't I got I got
Kyle	Summoned after but I got drafted to be an election worker because apparently they can draft you to be an election worker
Kyle	I thought you were just gonna say I got drafted. Oh, I didn't know they brought that back yet. I
Kyle	Would have gotten a notice of sourcing before you get drafted
Nolan	That's I got selected for jury duty, but I got out of it

Transcript of 19 October Client Check-In

Nolan	So it's important and I sent a message to the judge and I was like, hey, I'm like not there
Nolan	So, please don't make me come because I would be very upset with you not it's at
Nolan	That's your simple responsibility. I hope that someday in the future you
Nolan	Will become a juror just to you know, fulfill your duty
Kyle	I mean, I I would have loved to be on jury duty
Kyle	I had the time that I got called I was in the middle of my thesis though
Kyle	So I was like, hey, can I get it deferred because I'm in my thesis?
Kyle	Pushed back and then the day that I would have gone was a massive snow day
Kyle	So they just didn't call me in
Austin	They pay you $13 a day and they still make you pay for the goddamn parking come on
Nolan	I've got to be more than that a day an hour
Austin	I want to say I got maybe 50 bucks for a day and a half a day or something
Austin	It was some pittance, but they still also make you pay for parking. Yeah, they made you come here. They give you lunch, right? No
Nolan	Oh
Austin	They're like, there's a cafe downstairs
Austin	Yeah, anyway, I'm wrap on the transcript there good

8.10 Rasterized Documentation of Module Interfaces

The documentation for the CSLC portal was created via Sphinx. Here, we use the python package `weasyprint` to crudely cast the html documentation into a pdf format for the sake of appending a static copy to the final report. This is not meant as a replacement to Sphinx's html documentation; instead, the pdf renderings provide proof of work. We converted the html files to pdf by running the following bash commands:

```
    find -name "*.html" | while read -r file; do
        weasyprint "\$file" "./\${file}.pdf";
    done

```

University of Nebraska - Computer Science Tutoring Portal

Release 0.0.1

Nolan Gregory

Dec 11, 2023

CSLC PORTAL BACKEND

CHAPTER

ONE

ABOUT

CHAPTER

TWO

API ENDPOINTS

2.1 Course

class api.endpoints.course.**APICourseList**(***kwargs*)

Bases: APIView

get(*request: Request*) → Response

get_querystring(*request: Request*) → QueryDict | Any

post(*request: Request*) → Response

renderer_classes = (<class 'rest_framework.renderers.BrowsableAPIRenderer'>, <class 'rest_framework.renderers.JSONRenderer'>, <class 'rest_framework.renderers.HTMLFormRenderer'>)

sanitize(*querystring: str*) → str

serializer_class

alias of CourseSerializer

2.2 Issues

class api.endpoints.issue.**APIIssueDetail**(***kwargs*)

Bases: APIView

get(*request: Request*, *pk: str | Any = Ellipsis*) → Response

put(*request: Request*, *pk: str | Any = Ellipsis*) → Response

query_obj(*pk: str*) → QuerySet | Any

class api.endpoints.issue.**APIIssueView**(***kwargs*)

Bases: APIView

get(*request: Request*) → Response

post(*request: Request*) → Response

renderer_classes = (<class 'rest_framework.renderers.BrowsableAPIRenderer'>, <class 'rest_framework.renderers.JSONRenderer'>, <class 'rest_framework.renderers.HTMLFormRenderer'>)

serializer_class
alias of IssueSerializer

2.3 Professor

class api.endpoints.professor.**APIProfessorDetail**(***kwargs*)

Bases: APIView

get(*request: Request*, *professor_pk: str*) → Response

put(*request: Request*, *professor_pk: str*) → Response

query_obj(*pk: str*) → QuerySet
Add more fish or shrimp to the tank.

Parameters

- **inhabitant** – The type of inhabitant, either shrimp of fish
- **quantity** – The number of fish or shrimp to be added

Raises
TankIsFullError – if the tank is already full

renderer_classes = (<class 'rest_framework.renderers.BrowsableAPIRenderer'>, <class 'rest_framework.renderers.JSONRenderer'>, <class 'rest_framework.renderers.HTMLFormRenderer'>)

serializer_class
alias of ProfessorSerializer

class api.endpoints.professor.**APIProfessorView**(***kwargs*)

Bases: APIView

get(*request: Request*) → Response

get_querystring(*request: Request*) → QueryDict | Any

post(*request: Request*, *search: str | None = None*) → Response

renderer_classes = (<class 'rest_framework.renderers.BrowsableAPIRenderer'>, <class 'rest_framework.renderers.JSONRenderer'>, <class 'rest_framework.renderers.HTMLFormRenderer'>)

sanitize(*querystring: str*) → str

serializer_class
alias of ProfessorSerializer

2.4 Section

class api.endpoints.section.**APISectionDetail**(***kwargs*)

Bases: APIView

get(*request: Request*, *section_id: str*) → Response

put(*request: Request*, *section_id: str*) → Response

query_obj(*pk: str*) → QuerySet | None

renderer_classes = (<class 'rest_framework.renderers.BrowsableAPIRenderer'>, <class 'rest_framework.renderers.JSONRenderer'>, <class 'rest_framework.renderers.HTMLFormRenderer'>)

serializer_class

alias of SectionSerializer

class api.endpoints.section.**APISectionView**(***kwargs*)

Bases: APIView

get(*request: Request*) → Response

get_querystring(*request: Request*) → QueryDict | Any

post(*request: Request*) → Response

renderer_classes = (<class 'rest_framework.renderers.BrowsableAPIRenderer'>, <class 'rest_framework.renderers.JSONRenderer'>, <class 'rest_framework.renderers.HTMLFormRenderer'>)

sanitize(*querystring: str*) → str

serializer_class

alias of SectionSerializer

2.5 Ticket

class api.endpoints.ticket.**APITicketDetail**(***kwargs*)

Bases: APIView

get(*request: Request*, *ticket_id: int*) → Response

patch(*request: Request*, *ticket_id: int*) → Response

renderer_classes = (<class 'rest_framework.renderers.BrowsableAPIRenderer'>, <class 'rest_framework.renderers.JSONRenderer'>, <class 'rest_framework.renderers.HTMLFormRenderer'>)

serializer_class

alias of TicketGetSerializer

class api.endpoints.ticket.**APITicketView**(***kwargs*)

Bases: APIView

get(*request: Request*) → Response

get_querystring(*request: Request*) → Any

post(*request: Request*) → Response

renderer_classes = (<class 'rest_framework.renderers.BrowsableAPIRenderer'>, <class 'rest_framework.renderers.JSONRenderer'>, <class 'rest_framework.renderers.HTMLFormRenderer'>)

sanitize(*querystring: str*) → str

serializer_class

alias of `TicketSerializer`

2.6 User

class api.endpoints.user.**APIUserDetail**(***kwargs*)

Bases: `APIView`

get(*request: Request*, *user_id: str*) → Response

put(*request: Request*, *user_id: str*) → Response

query_obj(*pk: str*) → QueryDict | Any

renderer_classes = (<class 'rest_framework.renderers.BrowsableAPIRenderer'>, <class 'rest_framework.renderers.JSONRenderer'>, <class 'rest_framework.renderers.HTMLFormRenderer'>)

serializer_class

alias of `UserSerializer`

class api.endpoints.user.**APIUserView**(***kwargs*)

Bases: `APIView`

get(*request: Request*) → Response

get_querystring(*request: Request*) → QueryDict | Any

post(*request: Request*) → Response

renderer_classes = (<class 'rest_framework.renderers.BrowsableAPIRenderer'>, <class 'rest_framework.renderers.JSONRenderer'>, <class 'rest_framework.renderers.HTMLFormRenderer'>)

sanitize(*querystring: str*) → str

serializer_class

alias of `UserSerializer`

CHAPTER

THREE

DATABASE MODELS

3.1 Course

class api.models.course.**Course**(**args*, ***kwargs*)

Bases: Model

The course table holds all information about a specific course.

Important: A course can have many sections, while a section can only be attributed to a single course. The course represents what you would see in a catalog (e.g., CIST-1400 & CSCI-1620), and sections are specific instances of the course (online, in person, etc.). Unlike sections, a course does not have a directly assigned professor; this data is stored in sections.

---

course_department

The department code of the course (e.g., "CSCI").

Type
CharField

course_name

The name of the course as it appears

Type
CharField

in the catalog

Type
e.g., "Operating Systems"

course_id

The unique identifier for the course.

Type
IntegerField

course_code

The code associated with the course (e.g., "4500").

Type
CharField

Manager:
generic (Manager): The default manager for the *Course* model.

Example

Examples can be given using either the `Example` or `Examples` sections. Sections support any reStructuredText formatting, including literal blocks:

```
$ python example_google.py
```

Todo:

- For module TODOs
- You have to also use `sphinx.ext.todo` extension

---

exception DoesNotExist

Bases: `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `MultipleObjectsReturned`

course_code

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

course_department

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

course_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

course_name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

generic: Manager = <django.db.models.manager.Manager object>

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

section_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

ticket_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

usertocoursetaken

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

usertocoursetutored

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

3.2 Issues

class api.models.issue.**IssueManager**(**args*, ***kwargs*)

Bases: `Manager`

get_issues() → QuerySet

class api.models.issue.**Issues**(**args*, ***kwargs*)

Bases: `Model`

Issues are a field in the ticket that a user will choose to describe the issue they are having. This allows for ticket issues to be comprehensively studied, and provides cleaner data as opposed to users entering their issue type manually. With this style of ticket issue, data analysis can be done more rapidly and with a higher confidence than previous methods allowed. The fields in the table are the problem type (i.e. homework), the severity (used for data collection purposes). The admin is able to quickly add and modify the issues with no consequence to the attributed tickets.

exception DoesNotExist

Bases: `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `MultipleObjectsReturned`

generic: Manager = <django.db.models.manager.Manager object>

get_severity_display(*, *field=<django.db.models.fields.CharField: severity>*)

issue_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

problem_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

severity

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

ticket_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

3.3 Professor

class api.models.professor.**Professor**(**args*, ***kwargs*)

Bases: `Model`

The professor table holds all details attributed to a professor. This table contains a professors first name, their last name, their full name, their email address, whether they are currently active (see "semester" for more detail on that), and their prof. ID. Since the professor ID is guranteed to be unique, we have opted to use this as the primary key for the table. As such, professors are searchable by their unique ID, and the serializer will likewise return the str representation as well.

exception DoesNotExist

Bases: `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `MultipleObjectsReturned`

email

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

first_name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

full_name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

generic = <django.db.models.manager.Manager object>

is_active

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

last_name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

professor = <api.models.professor.ProfessorManager object>

professor_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

section_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

ticket_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

class api.models.professor.**ProfessorManager**(**args*, ***kwargs*)

Bases: `Manager`

get_professor(*professor: str*) → QuerySet

get_professors() → QuerySet

3.4 Section

3.5 Ticket

class api.models.ticket.**Ticket**(**args*, ***kwargs*)

Bases: Model

The base class for tickets in the database. This is where the meat of the application purpose lies, as this table will hold all fields associated with a specific ticket. These fields are the professor, the section, the semester, the issue, the student who submitted the ticket, the tutor who primarily helped with the ticket, the tutor(s) who assisted the primary tutor, the name of the student, whether it was a successful ticket, the time the ticket was created, the date the ticket was created, the time the ticket was claimed (different than created), the time the ticket was closed, additional tutor comments, and if the ticket was reopened after it was closed.

CLOSED = 'CLOSED'

DIFFICULTY_CHOICES = [('EASY', 'Easy'), ('MEDIUM', 'Medium'), ('HARD', 'Hard')]

exception DoesNotExist

Bases: ObjectDoesNotExist

exception MultipleObjectsReturned

Bases: MultipleObjectsReturned

NEW = 'NEW'

OPENED = 'OPENED'

STATUS_CHOICES = [('NEW', 'New'), ('OPENED', 'Opened'), ('CLOSED', 'Closed')]

asst_tutor

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

asst_tutor_id

closed_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

course

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

course_id

created_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

description

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

difficulty

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

generic = <django.db.models.manager.Manager object>

get_difficulty_display(*, *field=<django.db.models.fields.CharField: difficulty>*)

get_next_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=True*, ***kwargs*)

get_next_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=True*, ***kwargs*)

get_previous_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=False*, ***kwargs*)

get_previous_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=False*, ***kwargs*)

get_status_display(*, *field=<django.db.models.fields.CharField: status>*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

issue

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

issue_id

messages_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

opened_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

professor

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

professor_id

status

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

student

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

student_id

ticket = <api.models.ticket.TicketManager object>

title

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

tutor

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

tutor_id

updated_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

was_flagged

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

was_reopened

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

was_successful

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class api.models.ticket.**TicketManager**(**args*, ***kwargs*)

Bases: Manager

get_active() → QuerySet

get_all() → QuerySet

get_completed() → QuerySet

get_course(*course: str*) → QuerySet

get_professor(*professor: str*) → QuerySet

get_section(*section: str*) → QuerySet

get_student(*student: str*) → QuerySet

get_successful() → QuerySet

get_tutor(*tutor: str*) → QuerySet

get_unclaimed() → QuerySet

3.6 User

class api.models.user.**AdminManager**(**args*, ***kwargs*)

Bases: Manager

get_admins() → QuerySet

class api.models.user.**StudentManager**(**args*, ***kwargs*)

Bases: Manager

get_student(*name: str*) → QuerySet

get_students() → QuerySet

class api.models.user.**TutorManager**(**args*, ***kwargs*)

Bases: Manager

get_tutor(*name: str*) → QuerySet

get_tutors() → QuerySet

class api.models.user.**User**(**args*, ***kwargs*)

Bases: Model

Generic user model for all active users of the application. Flags are used to indicate whether a user is an admin, a tutor, or a regular student. This currently will hold all data, such as tutor and admin specific data, but this might lead to empty fields within many of the objects in the table. This will be updated in the future to allow specific roles to be relations in the database, thus preventing empty fields (i.e. empty cells).

exception DoesNotExist

Bases: ObjectDoesNotExist

MSOID

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

exception MultipleObjectsReturned

Bases: MultipleObjectsReturned

admin = <api.models.user.AdminManager object>

courses_taken

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

Pizza.toppings and Topping.pizzas are ManyToManyDescriptor instances.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

courses_tutoring

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

Pizza.toppings and Topping.pizzas are ManyToManyDescriptor instances.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

email

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

generic = <django.db.models.manager.Manager object>

hour_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

is_active

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_admin

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_tutor

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_working

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

messages_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

student = <api.models.user.StudentManager object>

student_nuid

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

student_ticket

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

tutor = <api.models.user.TutorManager object>

tutor_ticket

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

user_bio

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

workinghours_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

CHAPTER

FOUR

API SCRIPTS

CHAPTER

FIVE

API CONFIG

5.1 Settings

5.2 base.asgi module

ASGI config for University-Nebraska-Tutor-Portal project.

It exposes the ASGI callable as a module-level variable named `application`.

For more information on this file, see https://docs.djangoproject.com/en/4.2/howto/deployment/asgi/

5.3 base.settings module

Settings for University-Nebraska-Tutor-Portal project.

5.4 base.urls module

URL configuration for University-Nebraska-Tutor-Portal project.

The *urlpatterns* list routes URLs to views. For more information please see:
https://docs.djangoproject.com/en/4.2/topics/http/urls/

Examples: Function views

1. Add an import: from my_app import views
2. Add a URL to urlpatterns: path(‘’, views.home, name=’home’)

Class-based views

1. Add an import: from other_app.views import Home
2. Add a URL to urlpatterns: path(‘’, Home.as_view(), name=’home’)

Including another URLconf

1. Import the include() function: from django.urls import include, path
2. Add a URL to urlpatterns: path(‘blog/’, include(‘blog.urls’))

5.5 base.wsgi module

WSGI config for University-Nebraska-Tutor-Portal project.

It exposes the WSGI callable as a module-level variable named `application`.

For more information on this file, see https://docs.djangoproject.com/en/4.2/howto/deployment/wsgi/

5.6 Module contents

CHAPTER

SIX

FORMS

6.1 TicketForm TypeScript Module Documentation

The *TicketForm* TypeScript module defines a React component responsible for creating support tickets. This documentation provides an overview of the module's structure, key components, and their responsibilities.

6.1.1 Overview

The *TicketForm* component facilitates the creation of support tickets by gathering information from users, such as their name, selected professor, course, issue type, and a detailed description of the problem. It leverages various utility functions, hooks, and external APIs to enhance its functionality.

6.1.2 Key Components

FormSchema

The *FormSchema* is a Zod schema defining the shape and validation rules for the form data. It ensures that the user provides valid input for fields such as name, title, professor, course, issue, and description.

TicketLabel Component

This component renders a formatted label for certain form fields, such as the full name.

TicketDescription Component

Similar to *TicketLabel*, this component renders a description for specific form fields, providing additional guidance or context.

TicketForm Component

The main *TicketForm* component orchestrates the entire form. It uses the *react-hook-form* library for form management and validation. The component includes form fields for user information, selects for professors, courses, and issue types, and a text area for the ticket description. It also handles form submission, triggering the creation of a support ticket through the *createTicket* API.

useFetchProfessor, useFetchCourse, useFetchIssue Hooks

These custom hooks fetch data from external APIs related to professors, courses, and issue types, respectively. They ensure that the form has up-to-date information for dropdowns and selects.

LoadingSelect Component

This component provides a visual indication (e.g., loading spinner) when fetching data for selects.

useToast Hook

The *useToast* hook manages the display of toast messages, providing user feedback on form submission.

useNavigate Hook

The *useNavigate* hook from *react-router-dom* facilitates navigation upon successful ticket submission.

6.2 TicketForm React Component Documentation

The *TicketForm* component is a React form used to create a new ticket in a ticketing system. It allows users to provide information about their name, the professor, course, issue type, a brief summary, and a detailed description of their ticket. The form incorporates various UI components, including buttons, inputs, and popovers.

6.2.1 Overview

The *TicketForm* component comprises several key sections and components:

- **Form Initialization**: The form is initialized using the *useForm* hook from *react-hook-form*. It uses the *FormSchema* provided by the *zod* library for validation.
- **Form Submission**: The form submission is handled by the *onSubmit* function, which triggers the *createTicket* mutation and displays a success toast message.
- **LoadingSelect**: A loading state is displayed for select dropdowns while fetching data.
- **Popovers and Commands**: Popovers are used for professor, course, and issue type selection, incorporating a search functionality (*Command*). Each selection dropdown is triggered by a *Button* and displays a list of items with a search input.
- **Form Fields and Validation**: Various form fields are utilized, including *Input*, *Textarea*, and custom *FormField* components. The form includes validation for name, title, professor, course, issue type, and description.
- **TicketLabel and TicketDescription**: Custom components (*TicketLabel* and *TicketDescription*) are used for styling labels and descriptions consistently.
- **Button Component**: The form submission is triggered by a *Button* component with dynamic disabling based on the mutation's pending state.

6.2.2 Usage

To use the *TicketForm* component, integrate it into the desired page or component. The form handles the creation of new tickets, including input validation and mutation handling.

Example:

```jsx import TicketForm from "@/path/to/TicketForm";

**function NewTicketPage() {**

    **return (**

        **<div>**
            <h1>Create a New Ticket</h1> <TicketForm />

        </div>

    );

}
```

Additional Notes

- The *max_ticket_length* constant defines the maximum character limit for the ticket description.
- The form utilizes the *zodResolver* from *@hookform/resolvers/zod* for Zod schema-based validation.
- The *useMutation* hook from *@tanstack/react-query* manages the asynchronous ticket creation process.

6.2.3 Usage

To use the *TicketForm* component, integrate it into a parent component or page within a React application. Make sure to include the necessary dependencies and handle form submission accordingly.

Example:

```jsx import TicketForm from "@/path/to/TicketForm";

**function SupportPage() {**

    **return (**

        **<div>**
            <h1>Submit a Support Ticket</h1> <TicketForm />

        </div>

    );

}
```

6.3 AnnouncementForm

This module contains the *AnnouncementForm* component, which is used to create and submit announcements.

6.3.1 Form Schema

The *FormSchema* object defines the schema for the announcement form, including the title, body, variant, start date, end date, and show date. It also specifies the validation rules and error messages for each field.

6.3.2 Submitting the Form

When the form is submitted, the *onSubmit* function is called, which triggers a toast notification displaying the submitted values in a formatted JSON style.

6.3.3 Rendering the Form

The *AnnouncementForm* component renders a form using the *useForm* hook from *react-hook-form*. It includes input fields for the announcement title, body, variant, start date, end date, and show date. Each field is rendered with appropriate validation and error messages.

6.3.4 UI Components

The form utilizes various UI components such as *Textarea*, *Select*, *Input*, *Button*, *Switch*, *Popover*, and *CalendarIcon* for user interaction and input.

6.3.5 Character Limits

The form also enforces character limits for the body field, with a maximum length of 255 characters. The character count is dynamically displayed as the user types in the *Textarea*, providing real-time feedback to the user.

6.3.6 Form Submission

Upon completion, the user can submit the form by clicking the "Schedule" button.

This RST documentation provides an overview of the *AnnouncementForm* component and its functionality within the ticketing portal.

6.4 ThemeProvider React Component Documentation

The *ThemeProvider* component is a React context provider that manages the theme state for a ticketing portal. It allows users to customize the theme (dark, light, or system default) and persists the selected theme in local storage.

6.4.1 Overview

The *ThemeProvider* component is responsible for providing a theme context to its children components. It offers functionality to set and retrieve the current theme, and it automatically syncs the theme with the user's system preference when the "system" theme is selected.

6.4.2 Key Components

createContext, useContext

These functions are used from the *react* library to create and access the theme context, respectively.

useState

The *useState* hook is used to manage the local state of the selected theme.

useEffect

The *useEffect* hook is used to update the document's root element with the selected theme class. It also listens for changes in the theme and adjusts the document accordingly.

ThemeProviderProps

This type defines the props expected by the *ThemeProvider* component, including *children*, *defaultTheme*, and *storageKey*.

ThemeProviderState

This type represents the state of the *ThemeProvider* component, including the current theme and the *setTheme* function.

initialState

This constant represents the initial state of the *ThemeProvider* component, with the default theme set to "system."

ThemeProviderContext

The context created using *createContext* to provide theme-related values to child components.

ThemeProvider Component

The main *ThemeProvider* component that manages the theme state, sets the theme class on the document root, and provides the theme context to its children.

useTheme Hook

The *useTheme* hook is a custom hook that retrieves the current theme and *setTheme* function from the context. It throws an error if used outside a *ThemeProvider*.

6.4.3 Usage

To use the *ThemeProvider* component, wrap it around the components that need access to the theme context within a React application. Customize the theme by providing the desired values for *defaultTheme* and *storageKey*.

Example:

```jsx
import ThemeProvider, { useTheme } from "@/path/to/ThemeProvider";

function App() {
  return (
    <ThemeProvider defaultTheme="dark">
      <MainContent />
    </ThemeProvider>
  );
}

function MainContent() {
  const { theme, setTheme } = useTheme();

  // Implement theme-specific rendering or functionality based on the theme context.

  return (
    <div>
      <h1>Theme is {theme}</h1> <button onClick={() => setTheme("light")}>Switch to Light Theme</button>
    </div>
  );
}
```

6.5 ModeToggle React Component Documentation

The *ModeToggle* component is a React component that provides a mode toggle button for changing the theme of a ticketing portal. It utilizes icons for the sun and moon to represent light and dark themes, respectively. The component is implemented using the *lucide-react* library for icon components and leverages the *DropdownMenu* component for the theme selection options.

6.5.1 Overview

The *ModeToggle* component offers a user-friendly way to switch between light, dark, and system (default) themes in the ticketing portal. Users can click on the button to reveal a dropdown menu with theme options.

6.5.2 Key Components

Button Component

The *Button* component from *@/components/ui/button* is used to create the clickable button for toggling themes. It includes icons for the sun and moon, providing a visual representation of the available themes.

DropdownMenu Components

The *DropdownMenu*, *DropdownMenuTrigger*, *DropdownMenuContent*, and *DropdownMenuItem* components from *@/components/ui/dropdown-menu* are utilized to create a dropdown menu for selecting different themes. The menu is triggered by clicking on the mode toggle button.

useTheme Hook

The *useTheme* hook from *@/forms/ThemeProvider* is used to access the *setTheme* function, enabling the component to update the selected theme based on user input.

6.5.3 Usage

To use the *ModeToggle* component, integrate it into a parent component or page within a React application. Ensure that the necessary dependencies are imported, and handle theme changes accordingly.

Example:

```jsx import ModeToggle from "@/path/to/ModeToggle";

**function ThemeSettingsPage() {**

    **return (**

        **<div>**
            <h1>Theme Settings</h1> <ModeToggle />

        </div>

    );

}
```

6.6 TutorTicketForm React Component Documentation

The *TutorTicketForm* component is a React form used in a ticketing portal to display and update ticket details. It includes various form fields for ticket information and provides functionality to update the ticket using the *updateTicket* API. The form is built using the *react-hook-form* library and incorporates components from the project's UI library.

6.6.1 Overview

The *TutorTicketForm* component consists of several key sections and components:

- **Alert Dialog and Trigger**: The form is displayed within an *AlertDialog*, triggered by a button with three dots.
- **Form Initialization**: The form is initialized using the *useForm* hook from *react-hook-form*. It uses a *FormSchema* provided by the *zod* library for validation.
- **Form Submission**: The form submission is handled by the *onSubmit* function, which triggers the *updateTicket* mutation and displays a success toast message.
- **Ticket Details Display**: Ticket details, including the title, ID, status, and actions, are displayed at the top of the form.
- **Dropdown Menus**: The form includes two dropdown menus with various actions, such as copying ticket details or flagging the ticket.
- **CheckDropField**: A custom checkbox field (*CheckDropField*) is used to toggle the *was_successful* property.
- **TextareaField**: A custom textarea field (*TextareaField*) is used for entering and displaying the ticket description.
- **DropdownField**: A custom dropdown field (*DropdownField*) is used for selecting the ticket status.
- **SearchFilterField and DropField**: Custom fields are used for selecting tutors and difficulty levels, respectively.
- **DetailLink Component**: The *DetailLink* component displays labeled details, such as student, professor, and course information.
- **Button Components**: Various buttons are used for actions like updating the form, discarding changes, or copying ticket details.

6.6.2 Usage

To use the *TutorTicketForm* component, pass the ticket object as a prop. The form handles the display and editing of ticket details. It relies on the *react-hook-form* library for form management and incorporates components from the project's UI library for consistent styling.

Example:

```jsx import TutorTicketForm from "@/path/to/TutorTicketForm";

**function TicketDetailsPage({ ticket }) {**

    **return (**

        **<div>**
            <h1>Ticket Details</h1> <TutorTicketForm ticket={ticket} />

        </div>

    );

}
```

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

b

INDEX

G

H

I

L

M

N

O

P

Q

R

S

T

U

W

Chapter 9

References

Connolly, T. M., & Begg, C. E. (2014). *Database Systems: A Practical Approach to Design, Implementation, and Management.* Pearson.

Django Project. (2021). *Django Documentation.* Django. `https://docs.djangoproject.com/en/3.2/`

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* Doctoral dissertation, University of California, Irvine.

Open Web Application Security Project (OWASP). (2021). OWASP Top Ten. `https://owasp.org/www-project-top-ten/`

OpenAI. (2023, November 17). *OpenAI/Whisper: Robust Speech Recognition via Large-Scale Weak Supervision.* GitHub. `https://github.com/openai/whisper/tree/main`