

*Florida International University
Knight Foundation School of Computing and Information Sciences*

Make My Day

Final Deliverable
Project Title: Make My Day
Summer 2022

Team Members: Rey Jairus Marasigan, Syed Shavez Hussain, Christian Gonzalez Lopez, Kevin Lang Gallego, Ryan Shipman, Daniel Richard Carlson, Sehyun Cho, Daniel Alejandro Gonzalez Del Giovane, Jose Antonio Saleh, Mariela Torres Zuniga

Product Owner(s): Giri Narasimhan

Instructor: Masoud Sadjadi

The MIT License (MIT)
Copyright (c) 2016 *Florida International University*

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

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

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

Abstract

The Make My Day! Project is a collaborative effort to improve students' retention and comprehension of their classes. During the 2022 summer semester, the team was tasked with creating a full stack web application from the ground up that provides instructors and students a simple way of giving and receiving questions within curated times. By answering a question daily or weekly on the website notified via email, students can steadily improve their skills throughout a semester. This document details all the information concerning the website, the team's development process, and various artifacts and diagrams.

Table of Contents

Introduction	5
Current System	5
Implementation	5
User Functionality	7
User Stories	8
Implemented User Stories	8
Pending User Stories	11
Project Plan	12
Software Resources	12
Sprint Plannings	13
Sprint 1	13
Sprint 2	13
Sprint 3	14
Sprint 4	15
Sprint 5	17
System Design	18
Architectural Patterns	18
System and Subsystem Decomposition	18
Deployment Diagram	19
Design Patterns	19
System Validation	19
Glossary	21
Appendix	23
Appendix A - UML Diagrams	23
Appendix B - User Interface Design	28
Appendix C - Sprint Review Reports	33
Appendix D - User Manuals, Installation/Maintenance Documentation, Shortcomings/Wishlist Documentation	40
User Manual	40
Installation/Maintenance	44
Shortcomings	46
Wishlist	46

Introduction

The purpose of the SUMMER 2022 development team is to create a full stack web application that provides questions to students on specific curated times. The main goal of the website is to improve student retention levels of their classes. By providing a question daily or weekly, course material and information will be able to stick better for the long term.

Current System

Having started from scratch, the current system of the application is an initial build of the website. Given that there was only 10 weeks of development, our team focused heavily on planning and building a foundation that would help future developers. In conjunction to that, the team also implemented a list of requirements detailed to us by our Product Owner. This includes a database schema implemented, login and registration, question system, and notification system.

Implementation

Given our previous experience with **Python**, our best choice for the website was an industry standard web framework called **Django**. Django's easy and straightforward framework allowed us to develop many features quickly. Some of these features include the question system, various registration pages, and the email notification service.

Prior to developing the website itself, the team took about a week or so in planning the database. This was necessary in preventing data redundancy, and improving data integrity as much as possible for allowing the website to scale in the future. The database design is normalized at minimum 3rd Normal form. However, towards the end of the semester, we made some drastic changes to the database to better fit the Product Owners' requests.

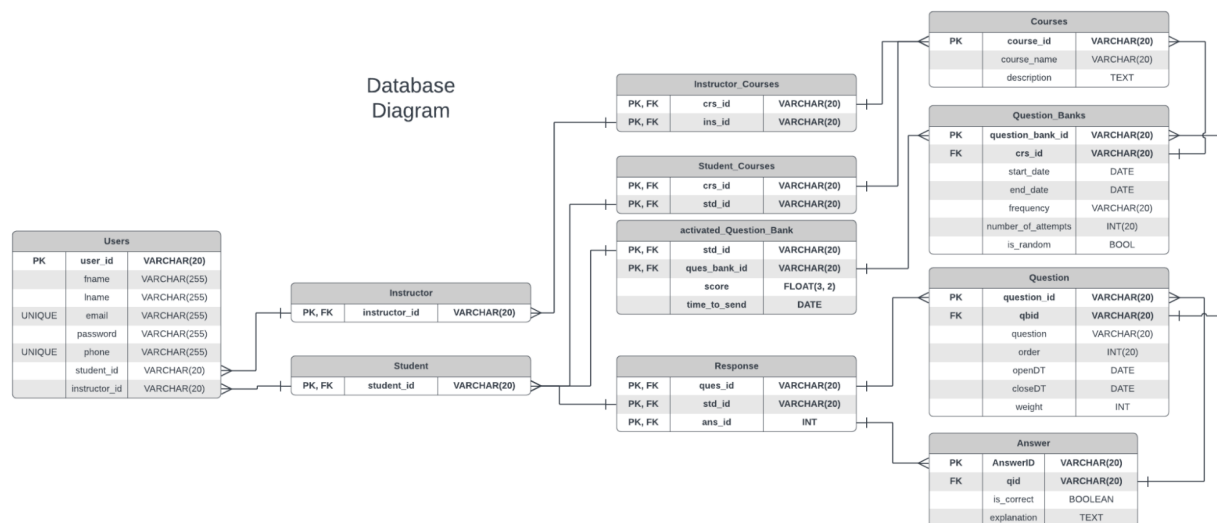


Fig 1. Initial Database Schema

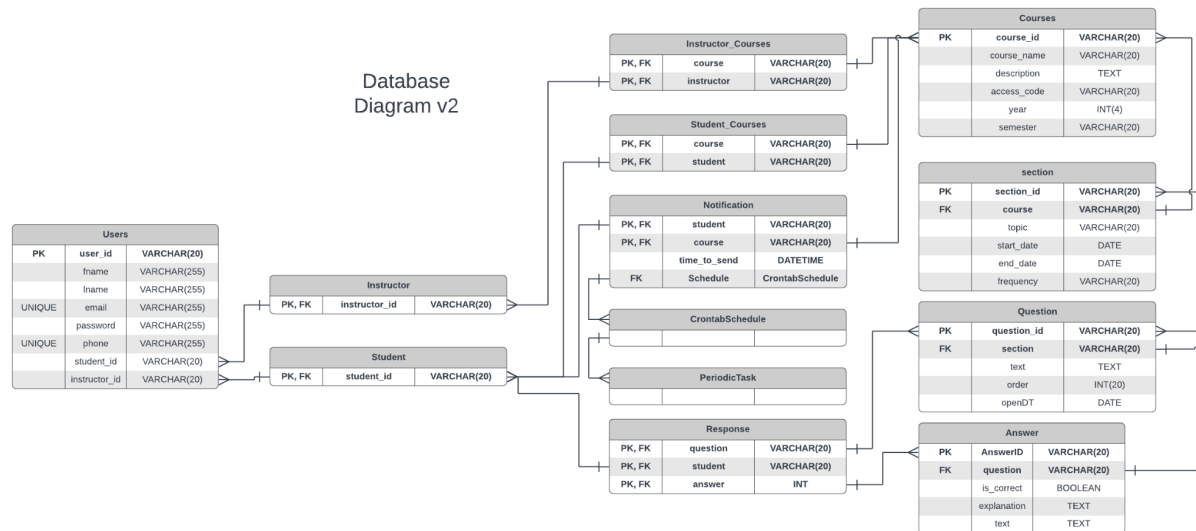


Fig 2. Final Database Schema

1* Website users inherit the built-in user functionality of Django.

2* The PeriodicTask contains the task for the website to send emails.

3* Each course has multiple sections for which instructors can sequentialize to their discretion. Each course is intended to only have one section active at a time (no overlapping start and end dates). This logic is not yet built in the backend.

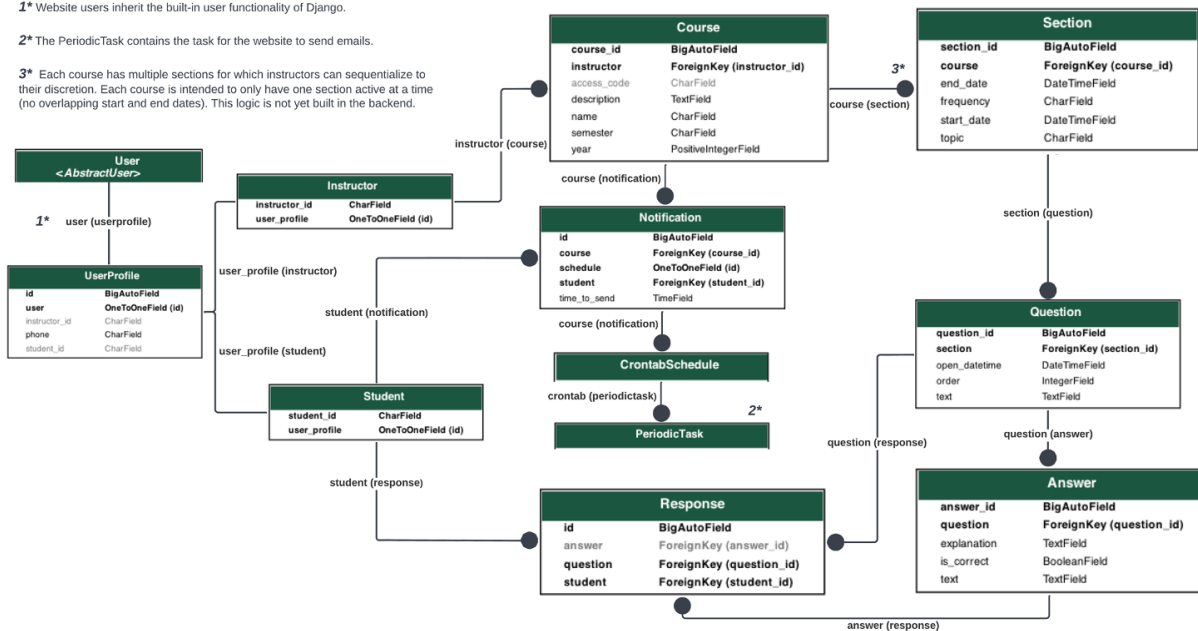


Fig 3. Django Database Schema

When implementing this database design, we chose **MySQL** as the database system. When scaling to thousands and possibly millions of instructors and students, the horizontal scalability of the website is crucial. During our research and planning, it was obvious to see that MySQL is great for such tasks. Currently, the website is running under a SQLite file for ease of use and testing. However, Django allows for an easy switch of databases once the website is near deployment.

After implementing the various tables as models (classes) in Django, we were able to

implement the registration and login for both students and teachers. Using Django's built in form system, this was easily accomplished. One register page and one login page is used for both students and instructors rather than two separate pages for each. This idea is often echoed throughout the website. For example, the homepage of the two different users are the same with different abilities.

The front end of the website is built primarily using **HTML**, **CSS**, **Bootstrap** and **JavaScript**. HTML was primarily used for structuring all the webpages. We used CSS and Bootstrap for adding visual flair and JavaScript for handling any user requests directly such as submitting questions and enrolling into a course. Switching to a front end framework such as React can be easily done with Django.

To handle the email system, the website uses a combination of **RabbitMQ**, **Celery**, and **Celery Beat**. RabbitMQ is a messaging queue that allows the backend to breathe and redirect intensive processing to asynchronous workers. These asynchronous workers in the website are referred to as the Celery workers. When a student signs up for notifications, the database records the time as a task for when the student receives the emails. Celery Beat, an extension of django, can then query or look into these tasks in the database every 5 seconds for any outstanding emails to be sent out. Eventually these tasks will be read by Celery Beat and redirected to RabbitMQ's messaging queue. This messaging queue is then hooked up to a Celery worker that asynchronously sends the email. Celery Workers can be easily implemented/reduced to match the needs of the website. When possibly thousands or millions of users use the website, heavy processing needs to be redirected. This combination of RabbitMQ, Celery and Celery Beat is necessary to deal with such a task.

During development, many of our team members experienced difficulties in running the website from the command line due to a mismatch of requirements. To fix this, we decided to containerize the application using **Docker**. Using Docker allowed an easier start up of the website on any operating system. Instead of running five different terminals and installing various packages, running the Docker daemon and a simple *docker-compose up* command is only needed to start up the various dependencies of the website. Using Docker also streamlines the process in deploying the website.

User Functionality

For a student to start answering questions, they need to first sign up to a course using a specific access code provided by the instructor. The website will then prompt the student to enter a time for when the website will notify via email him/her daily on what questions to do. This notification is then stored in the database as an intermediate model between students and courses. On the home page, the student can then pick on the questions for the day. Each question answered will provide the student with information about the result and be recorded as a response in the database. To look at past questions, students can check on the specific course pages displayed on their homepage.

As for the instructors, their functionality is not yet thoroughly implemented due to the lack of time. As developers, we are easily able to create any instance of any model through Django's built in admin panel. However, the instructors' pages for creating courses, sections, questions, and answers are not yet ready for deployment. The pages are functional but they do not meet our acceptance criteria. The pages' visual design does not match the rest of the website, and some of its components are missing. The statistics page of each course also faces the same issues.

User Stories

The following section will provide a more detailed description of the user stories that were worked on and completed for the semester for the Make My Day! project. The user stories worked as a basis of progress and development for the team.

Implemented User Stories

Sprint 1:

As a developer, I want to:

- [US101] Know if we are building a website or an App
 - web development or app development
 - Choose the framework that best suits the team
 - Choose the primary language to be used.
- [US102] Know what database is being used to store the various information.
 - Must be scalable
 - Must be able to store questions, answers, student information, course information, and other data.
- [US104] Know my position in developing the product so that no redundant work is done.
 - Assign roles and responsibilities

Sprint 2:

As a developer, I want to:

- [US100] Know if the students receive the questions from email, SMS, whatsapp, or other.
 - Research and Pick one way students receiving the questions
- [US103] Know how the UI is going to look.
 - Create some mock ups or look at some other implementation
- [US105] Know how the database is initially going to be set up to minimize hurdles in eventual implementation and further understand the product owner's requirements.
 - Create a database schema
 - Present to PO and request feedback
- [US106] refine and normalize the database to avoid redundant data and increase data integrity
 - Make sure it covers the basic needed information
 - Normalize database to BCNF
- [US107] implements the database so that we have a place to store all the information.
 - Implement the tables and their attributes

- [US110] Create API endpoints in our back-end to return basic information such as questions, answers, courses, etc from the database.
 - Use Django views to create functions that will handle HTTP requests and return specific data from the database (views.py).
- [US111] Create Models for the database tables using ERD that was previously created (US105)
 - Create models to abstract each database table in Django (models.py).

Canceled:

- [US108] know how to keep the same instance of the database of MySQL across multiple computers.
 - Have the same information from one computer to another.

Sprint 3:

As a developer, I want to:

- [US112] learn how to send messages to students so that they can answer questions.
 - Know a way for the student to answer questions from a text message.
- [US113] set up the basic functionalities of a quiz for students to answer questions
 - Have a question be answered and then recorded on the database.

As an Instructor, I want to:

- [US202] Create a list of registered students to keep courses organized.
 - Have a web page that lists out a course's students.
 - Have instructors appear within their respective courses.

Sprint 4:

As a student, I want to be able to:

- [US301] register for questions for a course for a specific semester if s/he is on the student list set by the instructor to learn.
 - Have a way for students to be placed under specific courses. (access codes, specific links, instructors requesting for them to join courses, etc.)
- [US302] Respond to the given question and see whether it is correct or incorrect and an explanation as to why so that I can study it.
- [US303] request for questions to be sent at specific times of the day to accommodate my schedule.
- [US304] see my previous responses so that I can study them.
- [US305] receive emails for the questions that need to be answered so that I can keep up to date with the class. **IMPORTANT**

As an Instructor, I want to:

- [US201] register a course for a specific semester so that I can send questions to my students.
- [US202] Create a list of registered students to keep courses organized.
 - Have the ability to add registered users to the course.

Creating question banks/ questions

- [US203] create a new question bank with some broad parameters so that questions are sent and done under more specific conditions.
 - Create a forum/web page that creates question banks for courses (access point could be from the page of a course's students).
 - Have broad parameters.
- [US208] See the performance of my students to keep track of progress.

Sprint 5:

As a developer, I want to be able to:

- [US112] clean up the code and remove unnecessary comments so as to accommodate future development.
 - Indent properly
 - Have a consistent structure throughout the files
 - Comment appropriately

As a user, I want to be able to:

- [US401E] have a clean user interface so that navigating through the website is easy and simple.
 - Accommodate for all screen sizes.
 - Consistent design throughout all the webpages.

As a developer, I want to be able to:

- [US113] create documentation so that future developers can read and understand our code.
 - Create the final deliverable
 - Record the required videos
 - Create posters

Pending User Stories

Highest priority on top

As a developer, I want to be able to:

- [US114] host the site so that it can be accessible
 - Website environment is exactly the same as if it was being run from Docker.

As an Instructor, I want to:

- [US204] Delete, add, or edit questions or answers to questions so that my students can learn.
 - Create a forum/web page that creates questions and answers for question banks (could be in the same page as the question bank).
- [US205] reuse previous questions banks to avoid additional work.
- [US206] Provide explanations to each answer chosen by the student to improve comprehension.
- [US207] Send the questions in a specific order or randomized under specific topics/groups during a specific time interval so that questions are relevant to what is currently being taught.

As a user, I want to be able to:

- [US401E] have a clean user interface so that navigating through the website is easy and simple.
 - Accommodate for all screen sizes.
 - Consistent design throughout all the webpages.

Project Plan

This section will go more in depth into the planning that was done for the semester and the project. Using agile development techniques, the team worked in sprints that would help us keep track of our progress as development occurred. Below you will find sprint plannings that were done at the beginning of each sprint to organize them. Also described in this section will be some of the hardware and software that were used for this project.

Software Resources

The following is a list of the hardware and software that were used for this project in regards to development and installation:

- **Python:** an interpreted, object-oriented, high-level programming language with dynamic semantics.
- **Django:** a free and open-source, Python-based web framework that follows the model–template–views architectural pattern.
- **Celery:** a simple, flexible, and reliable distributed system to process vast amounts of messages, while providing operations with the tools required to maintain such a system.
- **Celery Flower:** a web based tool for monitoring and administrating Celery clusters
- **Javascript:** a scripting or programming language that allows implementation of complex features on web pages
- **JQuery:** a fast, small, and feature-rich JavaScript library. Functions include HTML document traversal and manipulation, event handling, animation, and Ajax.
- **HTML:** HyperText Markup Language is the standard markup language for documents designed to be displayed in a web browser.
- **CSS:** Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML.
- **Bootstrap:** An open-source CSS framework that is used for creating the designs and layouts for web pages.
- **RabbitMQ:** an widely popular open-source message-broker software. It provides applications a common platform to send and receive messages, and messages a safe place to live until received.
- **Docker:** a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers.
- **MySQL:** an open-source relational database management system.
- **Visual Studio Code:** a source-code editor developed by Microsoft for Windows
- **Github:** web-based version-control and collaboration platform for software developers

Sprint Plannings

Sprint 1

May 16, 2022 – May 27, 2022

User Stories/ Goals:

- As a product owner, I want to have everybody access to the resources needed so that we are ready to start to work on the project
 - Set up a meeting with the product owner.
 - Produce a product backlog
 - As a developer, I want to familiarize myself with the tools needed so that we can effectively implement what is asked of us.
 - Set up an environment (GitHub repository) for development.
 - As a team player, I want to understand the scrum process so that I can collaborate with my team and produce the best product possible.
 - Attend meetings and submit documents at their appropriate times.
 - As a team player, I want to know the best time to congregate with the rest of my team so that we can all speak to each other and avoid often rescheduling.
 - Set up a when2meet and have everyone input their available times.
 - As a developer, I want to know my position in developing the product so that no redundant work is done.
 - Assign roles and responsibilities
-

Sprint 2

May 31, 2022 – June 10, 2022

User Stories/ Goals:

As a developer, I want to:

- [US100] Know if the students receive the questions from email, SMS, whatsapp, or other.

- Research and Pick one way students receiving the questions
 - [US103] Know how the UI is going to look.
 - Create some mock ups or look at some other implementation
 - [US105] Know how the database is initially going to be set up to minimize hurdles in eventual implementation and further understand the product owner's requirements.
 - Create a database schema
 - Present to PO and request feedback
 - [US106] refine and normalize the database to avoid redundant data and increase data integrity
 - Make sure it covers the basic needed information
 - Normalize database to BCNF
 - [US107] implement the database so that we have a place to store all the information.
 - Implement the tables and their attributes
 - [US108] know how to keep the same instance of the database of MySQL across multiple computers.
 - Have the same information from one computer to another.
 - [US110] Create API endpoints in our back-end to return basic information such as questions, answers, courses, etc from the database.
 - Use Django views to create functions that will handle HTTP requests and return specific data from the database (views.py).
 - [US111] Create Models for the database tables using ERD that was previously created (US105)
 - Create models to abstract each database table in Django (models.py).
-

Sprint 3

June 13, 2022 – June 24, 2022

User Stories/ Goals:

As a developer, I want to:

- [US112] learn how to send messages to students so that they can answer questions.
 - Know a way for the student to answer questions from a text message.
- [US113] set up the basic functionalities of a quiz for students to answer questions

- Have a question be answered and then recorded on the database.

As a course instructor, I want to be able to:

- [US201] register a course for a specific semester so that I can send questions to my students.
 - Ability for the instructor to create courses
- [US202] Create a list of registered students to keep courses organized.
 - Have a web page that lists out a course's students.
 - Have instructors appear within their respective courses.
 - Have the ability to add registered users to the course.
- [US203] create a new question bank with some broad parameters so that questions are sent and done under more specific conditions.
 - Create a forum/web page that creates question banks for courses (access point could be from the page of a course's students).
 - Have broad parameters.
- [US204] Delete, add, or edit questions or answers to questions so that my students can learn.
 - Create a forum/web page that creates questions and answers for question banks (could be in the same page as the question bank).

As a student, I want to be able to:

- [US301] register for questions for a course for a specific semester if s/he is on the student list set by the instructor to learn.
 - Have a way for students to be placed under specific courses. (access codes, specific links, instructors requesting for them to join courses, etc.)

As a user, I want to be able to:

- [US401E] have a clean user interface so that navigating through the website is easy and simple.
 - Accommodate for all screen sizes.
 - Consistent design throughout all the webpages.

Sprint 4

June 27, 2022 – July 8, 2022

User Stories/ Goals:

As an Instructor, I want to:

- [US201] register a course for a specific semester so that I can send questions to my students.
- [US202] Create a list of registered students to keep courses organized.
 - Have the ability to add registered users to the course.
- [US203] create a new question bank with some broad parameters so that questions are sent and done under more specific conditions.
 - Create a forum/web page that creates question banks for courses (access point could be from the page of a course's students).
 - Have broad parameters.
- [US204] Delete, add, or edit questions or answers to questions so that my students can learn.
 - Create a forum/web page that creates questions and answers for question banks (could be in the same page as the question bank).
- [US205] reuse previous questions banks to avoid additional work.
- [US206] Provide explanations to each answer chosen by the student to improve comprehension.
- [US207] Send the questions in a specific order or randomized under specific topics/groups during a specific time interval so that questions are relevant to what is currently being taught.
- [US208] See the performance of my students to keep track of progress.

As a student, I want to be able to:

- [US301] register for questions for a course for a specific semester if s/he is on the student list set by the instructor to learn.
 - Have a way for students to be placed under specific courses. (access codes, specific links, instructors requesting for them to join courses, etc.)
- [US302] Respond to the given question, and see whether it is correct or incorrect and an explanation as to why so that I can study it.
- [US303] request for questions to be sent at specific times of the day to accommodate my schedule.
- [US304] see my previous responses so that I can study them.
- [US305] receive emails for the questions that need to be answered so that I can keep up to date with the class.

As a user, I want to be able to:

- [US401E] have a clean user interface so that navigating through the website is easy and simple.
 - Accommodate for all screen sizes.
 - Consistent design throughout all the webpages.

Sprint 5

July 11, 2022 – July 22, 2022

User Stories/ Goals:

As a developer, I want to be able to:

- [US112] clean up the code and remove unnecessary comments so as to accommodate future development.
 - Indent properly
 - Have a consistent structure throughout the files
 - Comment appropriately
- [US113] create documentation so that future developers can read and understand our code.
 - Create the final deliverable
 - Record the required videos
 - Create posters
- [US114] host the site so that it can be accessible
 - Website environment is exactly the same as if it was being run from the CLI.

As a user, I want to be able to:

- [US401E] have a clean user interface so that navigating through the website is easy and simple.
 - Accommodate for all screen sizes.
 - Consistent design throughout all the webpages.
-

System Design

In this section we will cover some of the design choices and implementations of different features of our application. Diagrams will also be provided to give a quick overview of how we are currently deploying our application.

Architectural Patterns

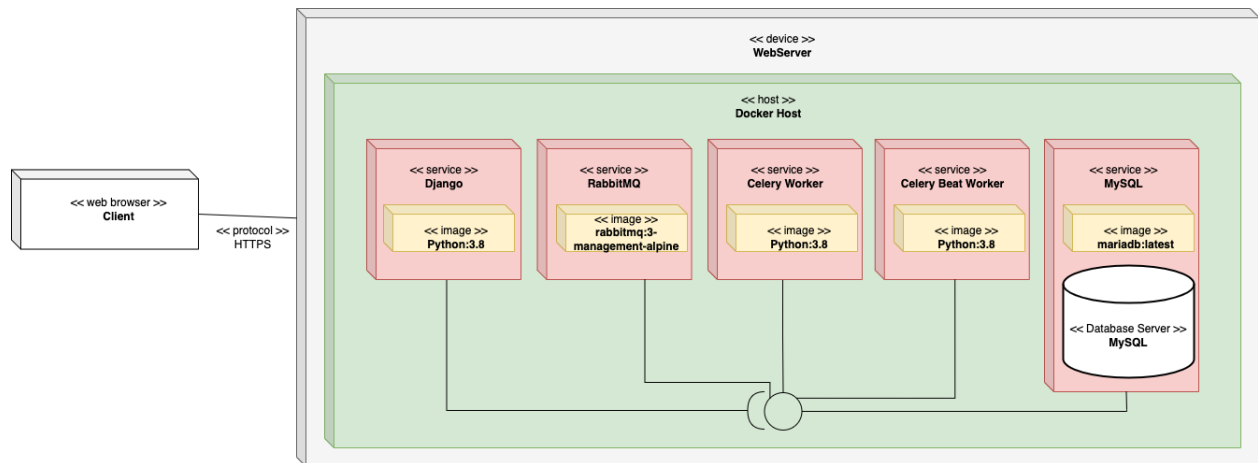
The Make My Day! Project was largely developed using the Django framework in Python. Being a Django application, our project implements the Model-View-Controller pattern. In MVC, the model represents the data logic and data structure that the application leverages to serve its users. In the case of our app, this would be the SQLite database along with the Python objects that represent entities from the database schema. The view handles how the information from the model is displayed to the user. In Django, the view has two parts. First we specify a “view function” that determines which data from the model should be served to the user. Lastly, we use template files that accept data from the view function and display the data in a uniform way. The controller represents the process of matching a user request with the appropriate action on the model or view. In the case of Django, the controller can be thought of as the framework itself. The actual functionality that routes user requests to the correct destination happens behind the scenes. However, as developers, we are able to provide the framework a simple mapping of URL routes to view functions.

System and Subsystem Decomposition

The Make My Day! Website is set up to be run within Docker containers. In the current iteration, the application and its services are run across 4 containers simultaneously. One container runs the Django application, another runs the RabbitMQ messaging service we use for sending emails to users, and the last two containers are dedicated to the Celery workers that are responsible for querying the database for notifications and passing those notifications to the message queue.

As explained in the previous section, Django takes advantage of the Model-View-Controller architecture. We also use the RabbitMQ and Celery frameworks to simplify and streamline the handling of email notifications.

Deployment Diagram



Design Patterns

- **Facade:** A structural design pattern that provides a simple interface for various classes.
- **Iterator:** A way to access the elements sequentially without exposing the underlying representation.
- **Mediator:** an object that defines how a set of objects will interact with each other.
- **Observer:** a design that is used when there is a one to many relationship between objects such as if one object is changed, the dependent objects will be notified automatically

System Validation

We made test cases for the features that our team worked on this semester. This is the first semester that the project is worked on so there were no prior features to test.

T001:

- **Purpose:** Verify that all routes are reachable.
- **Preconditions:** Logged in as a student user and then log in as an instructor user.
- **Expected Results:** Every page should load and be reachable without error.

T002:

- **Purpose:** Check to see that student users can enroll in a course.
- **Preconditions:** user must be logged in as a student.
- **Expected Results:** When a student user enters a valid access code for an existing course the student would be enrolled into that course as a student.

T003:

- **Purpose:** Check to see that an instructor user can create a course.
- **Preconditions:** user must be logged in as an instructor.
- **Expected Results:** When an instructor fills the create course form a course is created in the database.

T004:

- **Purpose:** Check to see that student users can see course cards for courses they are enrolled in.
- **Preconditions:** User must be logged in as a student and be enrolled in at least one course.
- **Expected Result:** A card is rendered on the home view for each of the courses the student user is currently enrolled in.

T005:

- **Purpose:** User must be able to register as an instructor/student.
- **Preconditions:** None.
- **Expected Result:** User of selected type (instructor or student) should be created on the database when filling out and submitting the register page form with valid information.

T006:

- **Purpose:** User must not be able to register as an instructor/student with invalid information.
- **Preconditions:** None.
- **Expected Result:** User of selected type (instructor or student) should not be created when filling out and submitting the register page form with invalid information.

T007:

- **Purpose:** User must be able to login as an instructor/student with valid information.
- **Preconditions:** User already registered an account with valid information.
- **Expected Result:** User of selected type (instructor or student) should be logged in with the dashboard of the selected user type when filling out and submitting the login page form with valid information.

T008:

- **Purpose:** User must not be able to login as an instructor/student with invalid information.
- **Preconditions:** User already registered an account with valid information.
- **Expected Result:** User of selected type (instructor or student) should not be logged in with the dashboard of the selected user type when filling out and submitting the login page form with invalid information and instead promoted with error message.

T009:

- **Purpose:** Verify that student user course questions show up on the to-do component.
- **Prerequisites:** Must be logged in as a student user, enrolled in a course, and have a question due.
- **Expected Result:** questions that are due should render on the to-do list on the student home page.

T010:

- **Purpose:** Check to see that an instructor user can edit a course.
- **Preconditions:** user must be logged in as an instructor and have a course created.
- **Expected Results:** When an instructor clicks the button to edit a course from the course information page and changes the information for a course it should show the change in the database.

T011:

- **Purpose:** Check to see that an instructor user can delete a course.
- **Preconditions:** user must be logged in as an instructor and have a course created.
- **Expected Results:** When an instructor clicks the button to delete a course from the course information page the course should be deleted from the database.

T012:

- **Purpose:** Verify that users who are not logged in are prevented from viewing restricted pages and that they are able to see them after logging in.
- **Prerequisites:** Be logged out.
- **Expected Result:** User should be prompted with a message stating they are not allowed to see the page due to verification issues.

Glossary

Home page: This is the page where users will be prompted once they login to the website. The student home page will give them the option to register for a course and look at the questions they have due that day. An instructor home page will show the courses the instructor currently teaches on the home page.

Register Form: This page allows users to register for the web page. If a student is creating a profile, the student would fill all the required text boxes except the instructor. If a instructor is creating a profile, the instructor would fill all the required text boxes except the student. Each user has specific functionality correlated with them.

Course Form: This page pertains to the instructor. This allows the instructor to create a course and create questions associated with those courses. An instructor is required to fill all required text boxes pertaining to these forms. An instructor will also create an access code that would allow students that know the access code to gain access to the questions.

Section Form: This page pertains to the instructor. This allows the instructor to create a section of questions that the instructor will be allowed to open at a specific time.

Quiz Form: This page pertains to the instructor. This allows the instructor to create a question under a section.

Student Dashboard: This is essentially the home page once a student logs in. This page will give the student the option to register for a course (if they have an access code) and look at the questions that are due to that day.

Instructor Dashboard: This is essentially the home page once an instructor logs in. This page will give instructors the option to create a course as well as view the current courses they are teaching. If they are interested in seeing more information regarding the course, they could click the course and see some of the questions they have created.

Course Dashboard: This dashboard is shown to instructors and students. An instructor will be able to see students registered for the section and make any modification such as deleting a course or updating a course. This dashboard will also give instructors access to create a section or add questions to a section. A student will be shown a general course page with a description, time reminder, and sections that are open to the student.

Admin Dashboard: This page allows admin users to see information saved in the database as well as the structure of the website. Through this page, admins can make changes to any information in the database in case a user entered incorrect information or if a student/instructor needs to make a modification.

MVT: Model-view-template is the software design pattern for Make My Day database. The model holds persistent data that is represented by the database. The view queries the data from the database, sets up a specific template, and returns that information to the user. The template is a presentation layer that is essentially the HTML, CSS code that renders the data. It is useful for implementing web interfaces.

Appendix

Appendix A - UML Diagrams

Sehyun Cho (Appendix A - UML Diagrams)

Fig 1. Use Case Diagram

[US 305] Use Case Diagram

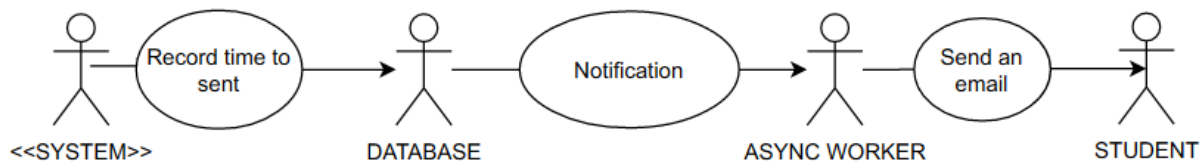


Fig 2. Class Diagram

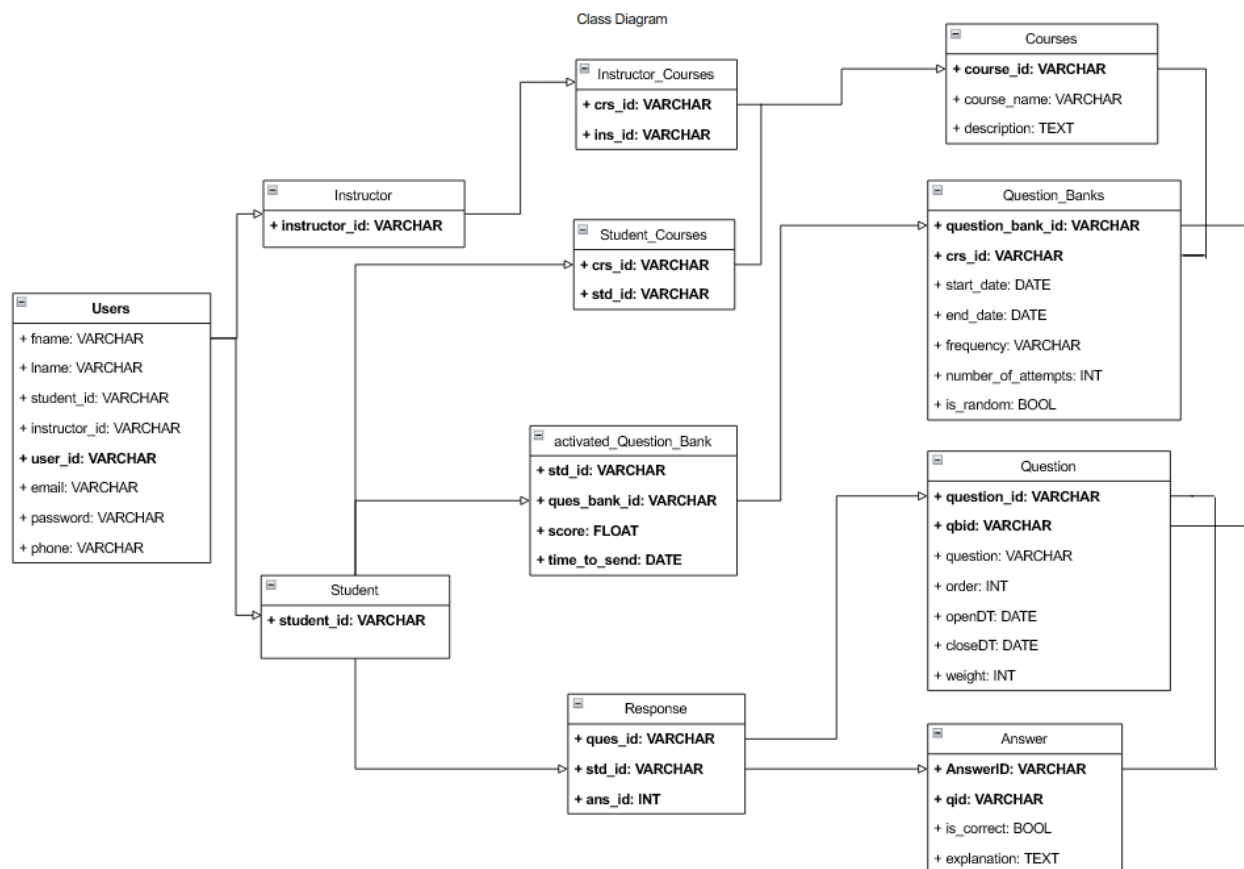
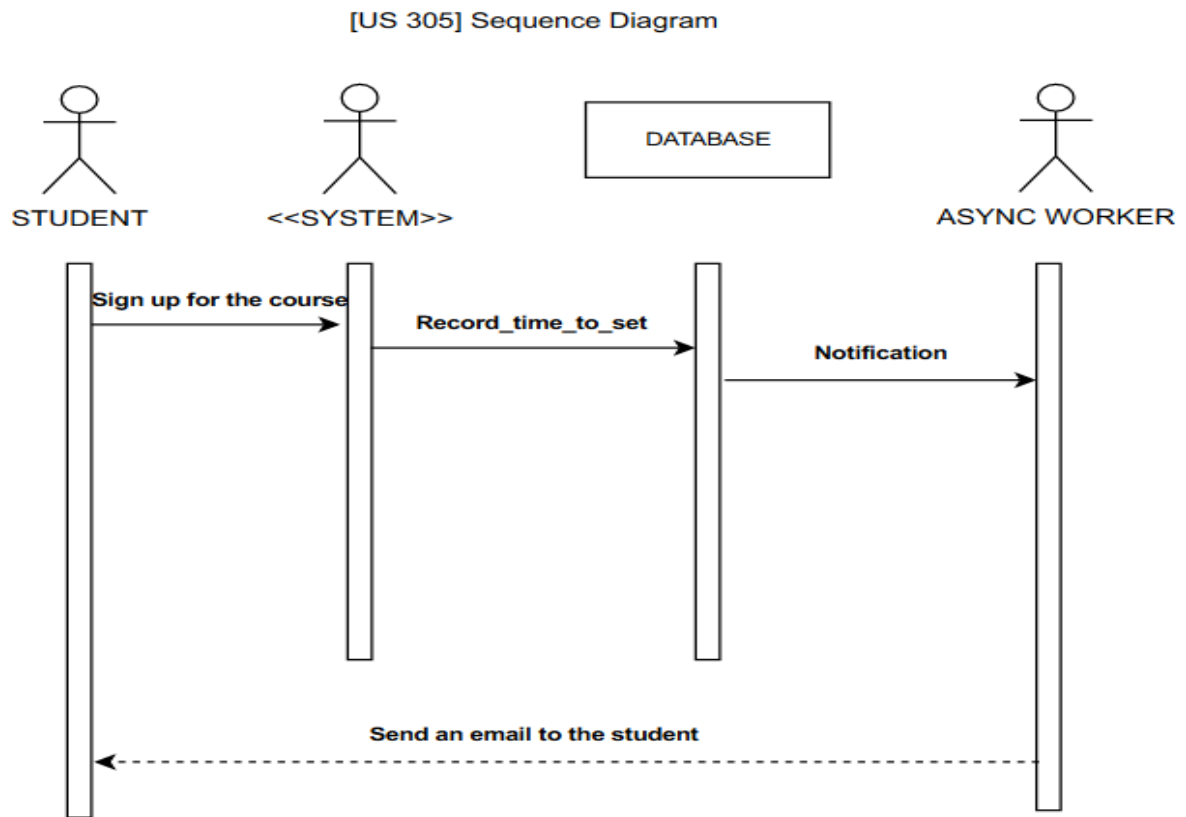


Fig 3. Sequence Diagram

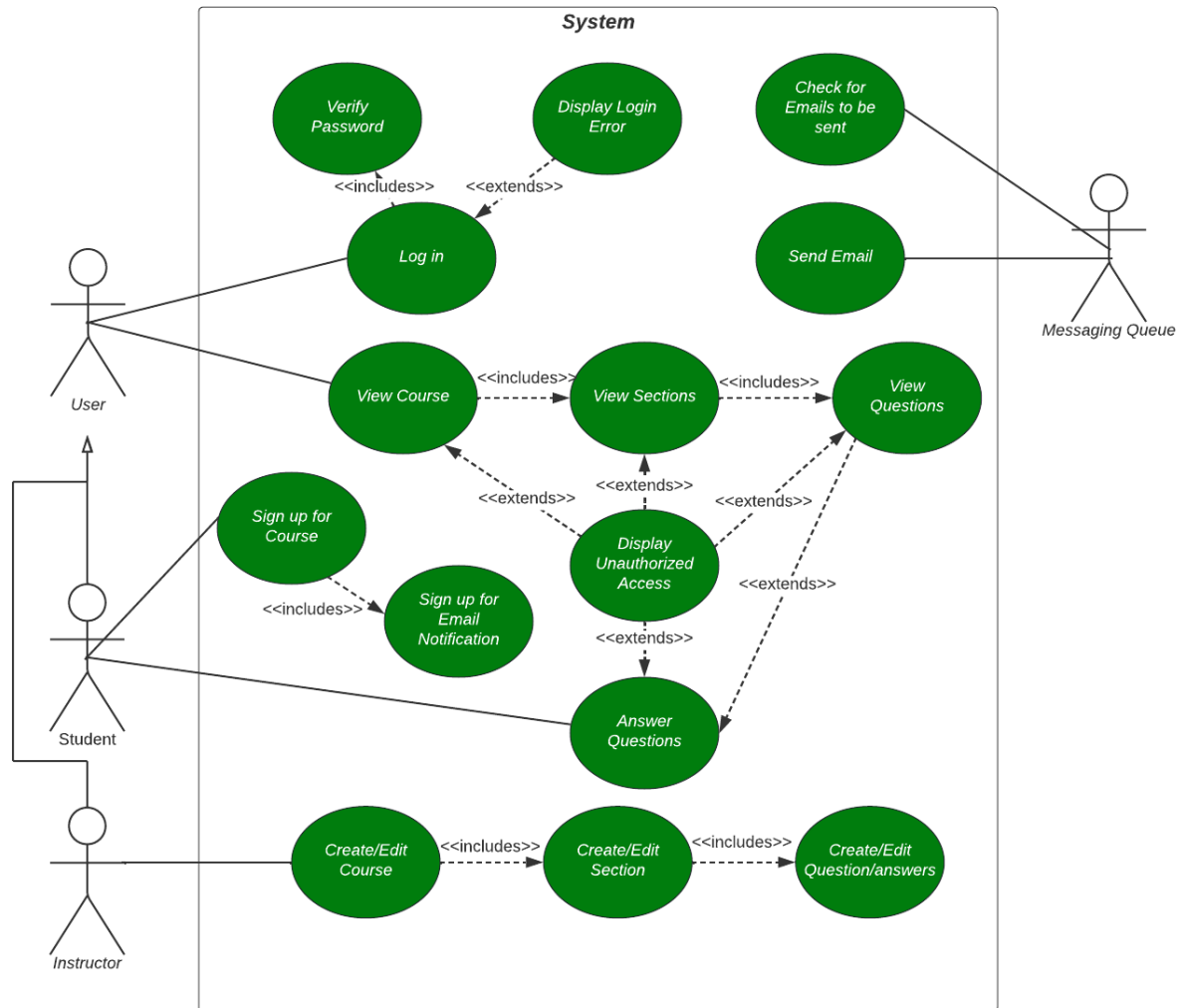
Rey Jairus Marasigan (Appendix A - UML Diagrams)**Fig 1. Use Case Diagram**

Fig 2. Class Diagram

1* Website users inherit the built-in user functionality of Django.

2* The PeriodicTask contains the task for the website to send emails.

3* Each course has multiple sections for which instructors can sequentialize to their discretion. Each course is intended to only have one section active at a time (no overlapping start and end dates). This logic is not yet built in the backend.

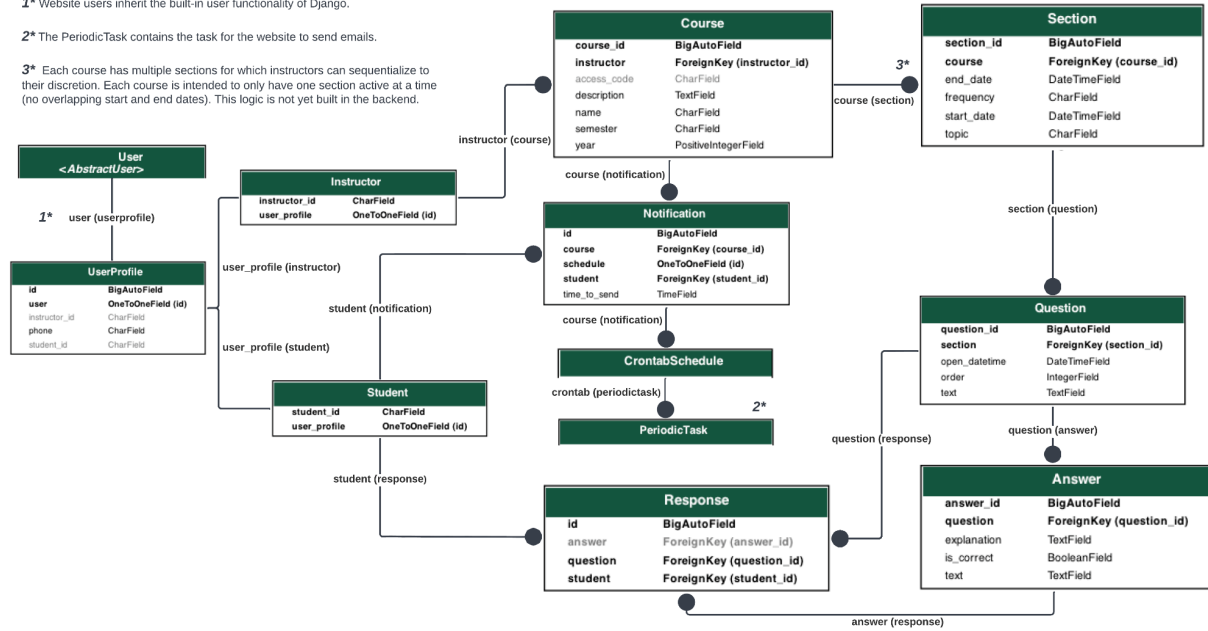


Fig 3. Sequence Diagram

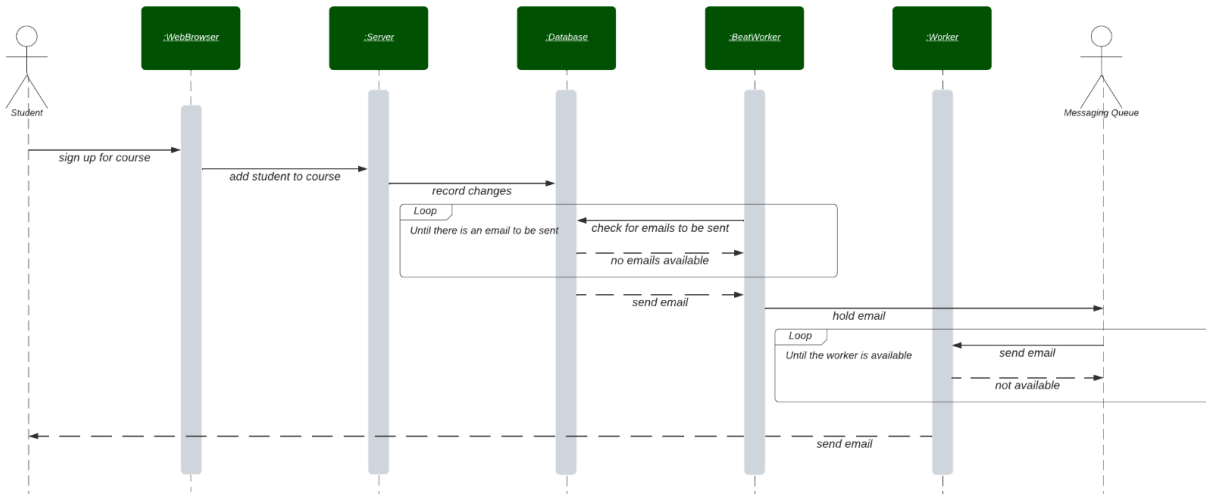


Fig 4. Initial Database Schema

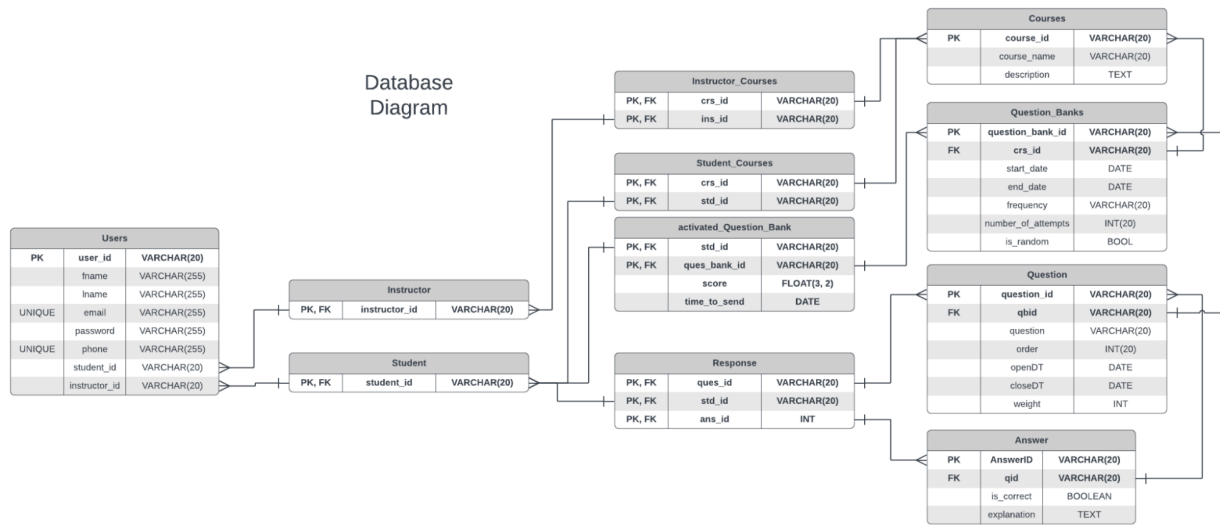
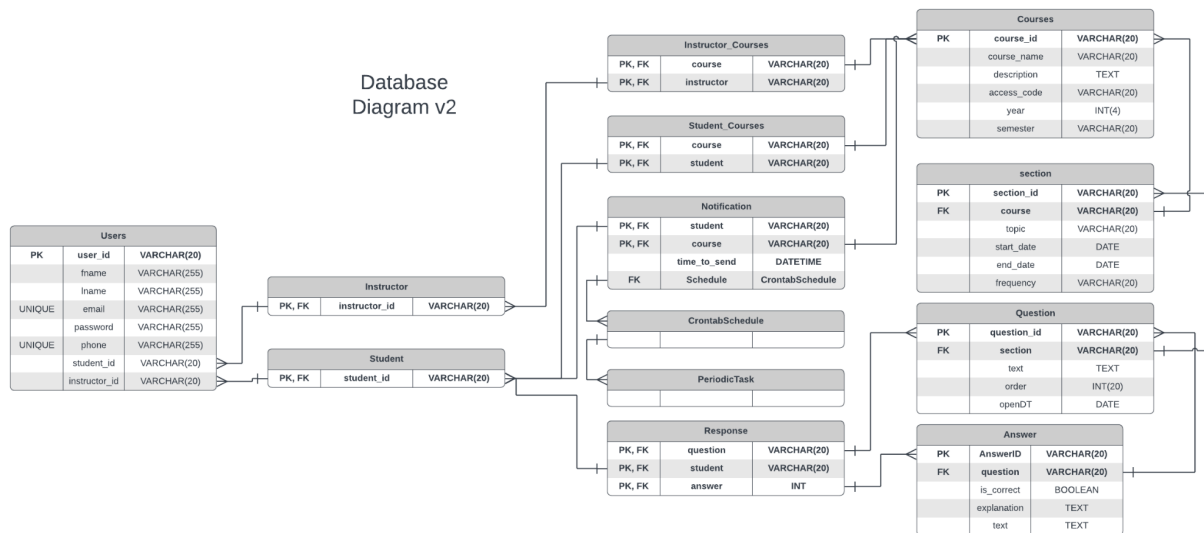


Fig 5. Final Database Schema



Appendix B - User Interface Design

[US103]

Fig 1. Home Page - Not Logged In

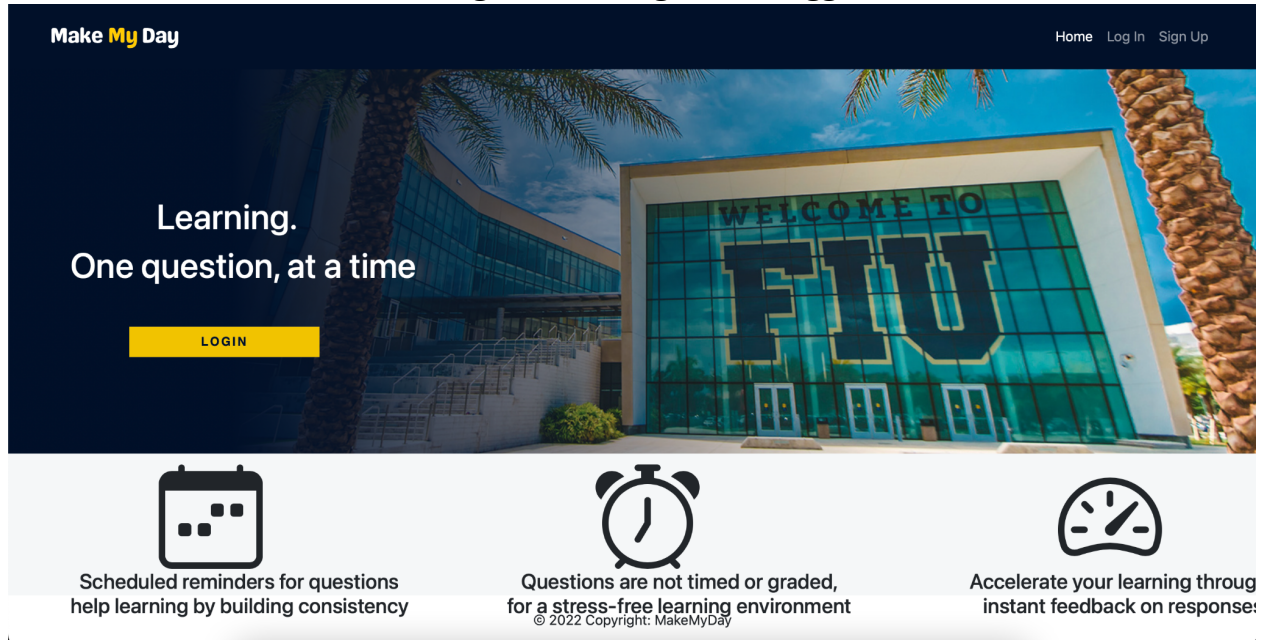


Fig 2. Log In Form

Please login to your account

Username*

Password*

Login

[Forgot password?](#)

Don't have an account? [Sign Up](#)

Fig 3. Sign Up Form

Username: Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email:

First name:

Last name:

Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation: Enter the same password as before, for verification.

Phone:

Student id:

Instructor id:

[US401E]

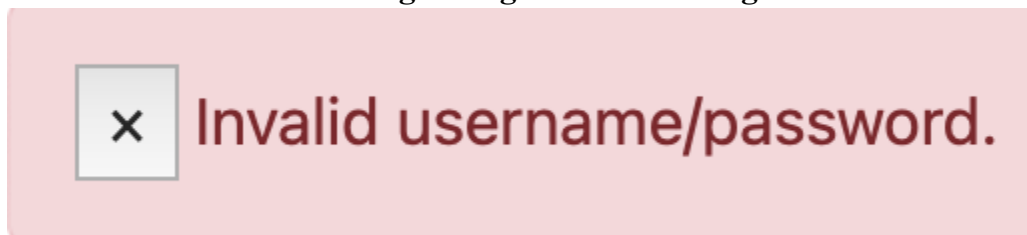
Fig 1. Login Error Message

Fig 2. Login success message

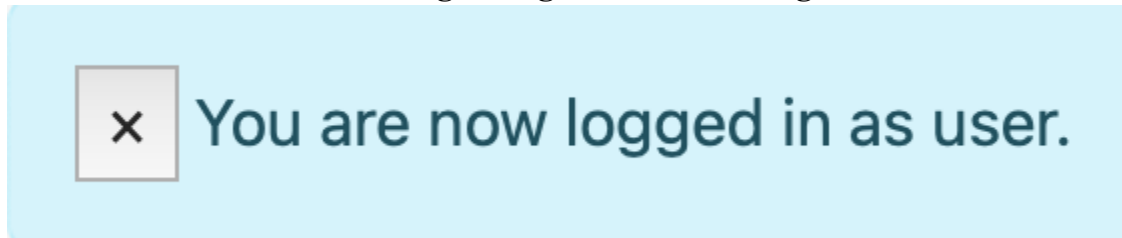
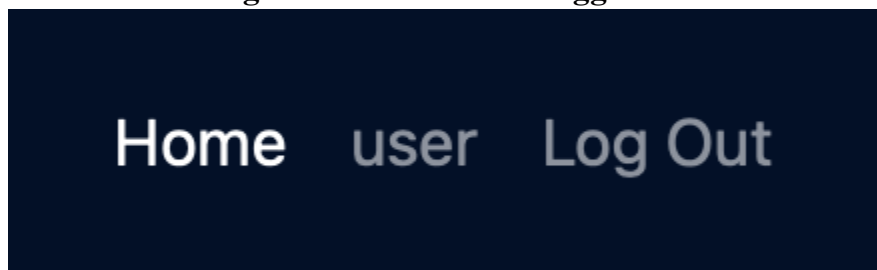


Fig 3. Nav Bar When Logged In



[US302]

Fig 1. Question Page

Home Click to go forward, hold to see history

Exit Quiz

Which address is used in an internet employing the TCP/IP protocols?

☐ physical address and logical address
☐ port address
☐ specific address
☐ all of the mentioned

Submit

© 2022 Copyright: MakeMyDay

[US301] [US303]

Fig 1. Signing up for course and notifications

Make My Day Home rmara020 Log Out

Data Structures 1

Netcentric Core 2

Confirm

Are you sure you want to begin Data Structures?

- Course ID: 1
- Instructor: Sadjadi, Masoud
- Year: 2022
- Semester: SUMMER

All things arrays, heaps, and more!

Please enter a time to receive the questions DAILY.

09:00 AM

07 58 PM
08 59 AM
09 00
10 01
11 02
12 03
01 04

Yes

New Questions

Today
July 26, 2022

© 2022 Copyright: MakeMyDay

Appendix C - Sprint Review Reports

Sprint Review/Retrospective - Sprint 1

Date: 5/27/22

Start Time: 2:00PM

End Time: 3:00PM

Attendees: Christian Gonzalez Lopez, Syed Hussain, Kevin Lang Gallego, Rey Jairus Marasigan, Ryan Shipman, Daniel Richard Carlson, Sehyun Cho, Daniel Alejandro Gonzalez Del Giovane, Jose Antonio Saleh, Mariela Torres Zuniga

Completed/Approved User Stories:

As a developer, I want to:

- [US101] Know if we are building a website or an App
 - web development or app development
 - Choose the framework that best suits the team
 - Choose the primary language to be used.
- [US102] Know what database is being used to store the various information.
 - Must be scalable
 - Must be able to store questions, answers, student information, course information, and other data.
- [US104] Know my position in developing the product so that no redundant work is done.
 - Assign roles and responsibilities

User Stories moved to Backlog:

As a developer, I want to:

- [US100] Know if the students receive the questions from email, SMS, whatsApp, or other.
 - Research and Pick one way students receiving the questions
- [US103] Know how the UI is going to look.
 - Create some mock ups or look at some other implementation

What the team should start doing in future sprints:

- Turn in documents by the end of meetings.
- Increase communication with the PO.

What the team should stop doing in future sprints:

- The team should decrease the frequency of being absent in meetings.

What the team should continue to do for future sprints:

- Continue trying to be comfortable with the current tech stack.

Sprint Review/Retrospective - Sprint 2

Date: 6/10/22

Start Time: 2:00PM

End Time: 3:00PM

Attendees: Christian Gonzalez Lopez, Syed Hussain, Kevin Lang Gallego, Rey Jairus Marasigan, Ryan Shipman, Daniel Richard Carlson, Sehyune Cho, Daniel Alejandro Gonzalez Del Giovane, Jose Antonio Saleh, Mariela Torres Zuniga

Completed/Approved User Stories:

As a Developer, I want to:

- [US100] Know if the students receive the questions from email, SMS, whatsApp, or other.
 - Research and Pick one way students receiving the questions

As a Developer, I want to:

- [US103] Know how the UI is going to look.
 - Create some mock ups or look at some other implementation
- [US105] Know how the database is initially going to be set up to minimize hurdles in eventual implementation and further understand the product owner's requirements.
 - Create a database schema
 - Present to PO and request feedback
- [US106] refine and normalize the database to avoid redundant data and increase data integrity
 - Make sure it covers the basic needed information
 - Normalize database to BCNF
- [US107] implement the database so that we have a place to store all the information.
 - Implement the tables and their attributes
- [US110] Create API endpoints in our back-end to return basic information such as questions, answers, courses, etc from the database.
 - Use Django views to create functions that will handle HTTP requests and return specific data from the database (views.py).
- [US111] Create Models for the database tables using ERD that was previously created (US105)
 - Create models to abstract each database table in Django (models.py).

User Stories moved to Backlog:

As a Developer, I want to:

- [US108] know how to keep the same instance of the database of MySQL across multiple computers.
 - Have the same information from one computer to another.

What the team should start doing in future sprints:

- Focus on user stories under the *Instructor* and *Student* subsections.

What the team should stop doing in future sprints:

N/A

What the team should continue to do for future sprints:

The same work ethic that was done this sprint to keep a consistent flow of progress.

Sprint Review/Retrospective - Sprint 3

Date: 6/24/22

Start Time: 2:00PM

End Time: 3:00PM

Attendees: Christian Gonzalez Lopez, Syed Hussain, Kevin Lang Gallego, Rey Jairus Marasigan, Ryan Shipman, Daniel Richard Carlson, Sehyun Cho, Daniel Alejandro Gonzalez Del Giovane, Jose Antonio Saleh, Mariela Torres Zuniga

Completed/Approved User Stories:

As a developer, I want to:

- [US112] learn how to send messages to students so that they can answer questions.
 - Know a way for the student to answer questions from a text message.
- [US113] set up the basic functionalities of a quiz for students to answer questions
 - Have a question be answered and then recorded on the database.

User Stories moved to Backlog:

As a course instructor, I want to be able to:

- [US201] register a course for a specific semester so that I can send questions to my students.
 - Ability for the instructor to create courses
- [US202] Create a list of registered students to keep courses organized.
 - Have a web page that lists out a course's students.
 - Have instructors appear within their respective courses.

- Have the ability to add registered users to the course.
- [US203] create a new question bank with some broad parameters so that questions are sent and done under more specific conditions.
 - Create a forum/web page that creates question banks for courses (access point could be from the page of a course's students).
 - Have broad parameters.
- [US204] Delete, add, or edit questions or answers to questions so that my students can learn.
 - Create a forum/web page that creates questions and answers for question banks (could be in the same page as the question bank).

As a student, I want to be able to:

- [US301] register for questions for a course for a specific semester if s/he is on the student list set by the instructor to learn.
 - Have a way for students to be placed under specific courses. (access codes, specific links, instructors requesting for them to join courses, etc.)

As a user, I want to be able to:

- [US401E] have a clean user interface so that navigating through the website is easy and simple.
 - Accommodate for all screen sizes.
 - Consistent design throughout all the webpages.

What the team should start doing in future sprints:

- Focus on creating more forums for creating questions, answers to those questions, and question banks.
- Focus on Front end design

What the team should stop doing in future sprints:

N/A

What the team should continue to do for future sprints:

- The same work ethic to keep a consistent flow of progress.

Sprint Review/Retrospective - Sprint 4

Date: 7/8/22

Start Time: 2:00PM

End Time: 3:00PM

Attendees: Christian Gonzalez Lopez, Syed Hussain, Kevin Lang Gallego, Rey Jairus Marasigan, Ryan Shipman, Daniel Richard Carlson, Sehyune Cho, Daniel Alejandro Gonzalez Del Giovane, Jose Antonio Saleh, Mariela Torres Zuniga

Completed/Approved User Stories:

As a student, I want to be able to:

- [US301] register for questions for a course for a specific semester if s/he is on the student list set by the instructor to learn.
 - Have a way for students to be placed under specific courses. (access codes, specific links, instructors requesting for them to join courses, etc.)
- [US302] Respond to the given question, and see whether it is correct or incorrect and an explanation as to why so that I can study it.
- [US303] request for questions to be sent at specific times of the day to accommodate my schedule.
- [US304] see my previous responses so that I can study them.
- [US305] receive emails for the questions that need to be answered so that I can keep up to date with the class. **IMPORTANT**

As an Instructor, I want to:

- [US201] register a course for a specific semester so that I can send questions to my students.
- [US202] Create a list of registered students to keep courses organized.
 - Have the ability to add registered users to the course.
- Creating question banks/ questions
- [US203] create a new question bank with some broad parameters so that questions are sent and done under more specific conditions.
 - Create a forum/web page that creates question banks for courses (access point could be from the page of a course's students).
 - Have broad parameters.
- [US208] See the performance of my students to keep track of progress.

User Stories moved to Backlog:

As an Instructor, I want to:

Creating question banks/ questions

- [US204] Delete, add, or edit questions or answers to questions so that my students can learn.
 - Create a forum/web page that creates questions and answers for question banks (could be in the same page as the question bank).
- [US206] Provide explanations to each answer chosen by the student to improve comprehension.
- [US207] Send the questions in a specific order or randomized under specific topics/groups during a specific time interval so that questions are relevant to what is currently being taught.

What the team should start doing in future sprints:

Documentation, cleaning up the code

What the team should stop doing in future sprints:

N/A

What the team should continue to do for future sprints:

Improving the look of the website

Sprint Review/Retrospective - Sprint 5

Date: 7/22/22

Start Time: 2:00PM

End Time: 3:00PM

Attendees: Christian Gonzalez Lopez, Syed Hussain, Kevin Lang Gallego, Rey Jairus Marasigan, Ryan Shipman, Daniel Richard Carlson, Sehyune Cho, Daniel Alejandro Gonzalez Del Giovane, Jose Antonio Saleh, Mariela Torres Zuniga

Completed/Approved User Stories:

As a user, I want to be able to:

- [US401E] have a clean user interface so that navigating through the website is easy and simple.

- Accommodate for all screen sizes.
- Consistent design throughout all the webpages.

As a developer, I want to be able to:

- [US112] clean up the code and remove unnecessary comments so as to accommodate future development.
 - Indent properly
 - Have a consistent structure throughout the files
 - Comment appropriately
- [US113] create documentation so that future developers can read and understand our code.
 - Create the final deliverable
 - Record the required videos
 - Create posters

User Stories moved to Backlog:

N/A

What the team should start doing in future sprints:

N/A

What the team should stop doing in future sprints:

N/A

What the team should continue to do for future sprints:

N/A

Appendix D - User Manuals, Installation/Maintenance Documentation, Shortcomings/Wishlist Documentation

User Manual

Using the Navbar

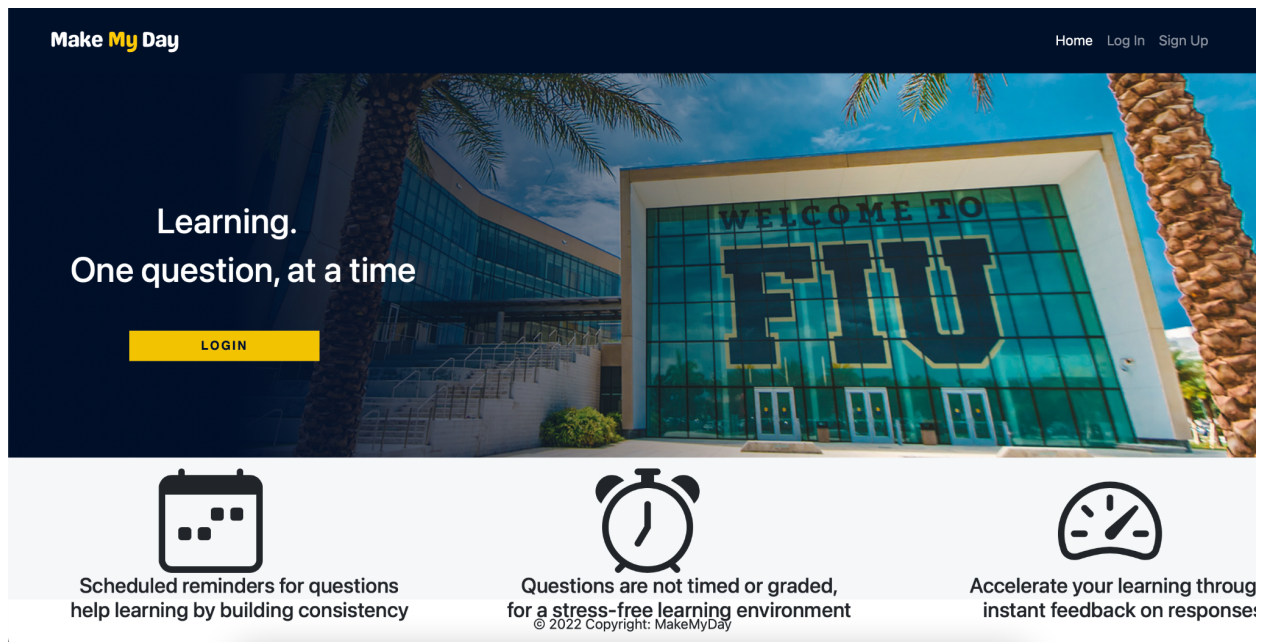


MMD's website navigation bar is located at the top of many of our pages. This navbar features a logo, home, login, and sign up links. The goal of this navigation bar is for users to always have a way of traveling back to the homepage, help with signing in, and also displaying who is signed in.

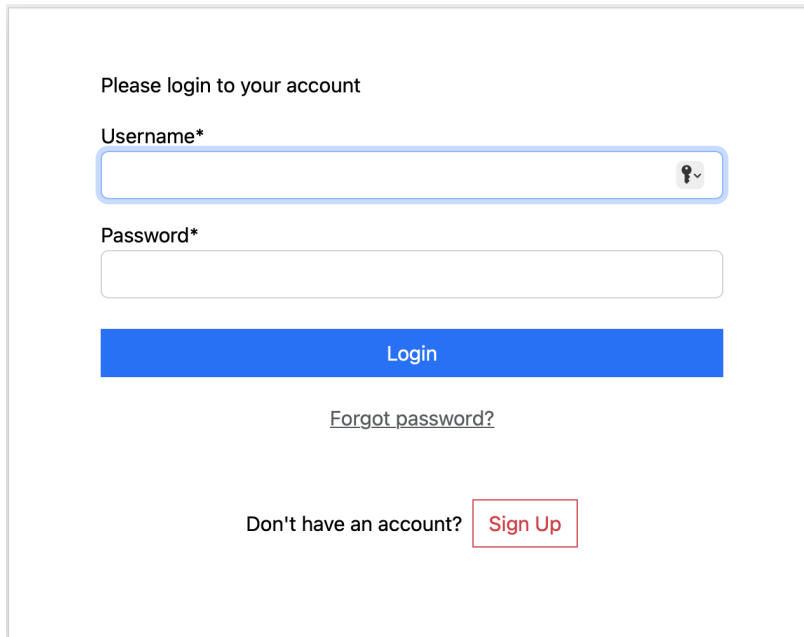
The logo on the top-left of the navbar is clickable and redirects you back to the homepage. As well as the Home link, these two have the same function. The Log in link redirects you to the sign in page of our application and the Sign up link to the register page.

When logged in, the Navbar changes in the way it operates. Instead of a Log in link, that will be replaced with a Log out link that signs the user with a click of the link. It will also display the username of the user signed in, instead of the sign up button.

Homepage (Logged out)



MMD's homepage when not logged in is simple, the usual navbar at the top, and the homepage displays a single login button so it is impossible to miss the login.

Log in

Please login to your account

Username*

Password*

Login

[Forgot password?](#)

Don't have an account? [Sign Up](#)

This is the page you will be redirected to in order to log in. It contains a username and password field. “Forgot password?” to aid in the recovery process of the account and a link to the register page using the “Sign Up” button. Press the blue Login button once the Username and Password fields have been set to log in a user or an instructor.

Register

Username: Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email:

First name:

Last name:

Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation: Enter the same password as before, for verification.

Phone:

Student id:

Instructor id:

This will be the page a user that is not logged in, will be redirected to if they click the “Sign Up” link in the navbar. To create an account, all the fields provided must be filled in and the password must be up to standards set by the register page. Once all the fields have been filled, the blue button that says “Register” can be used to register the account.

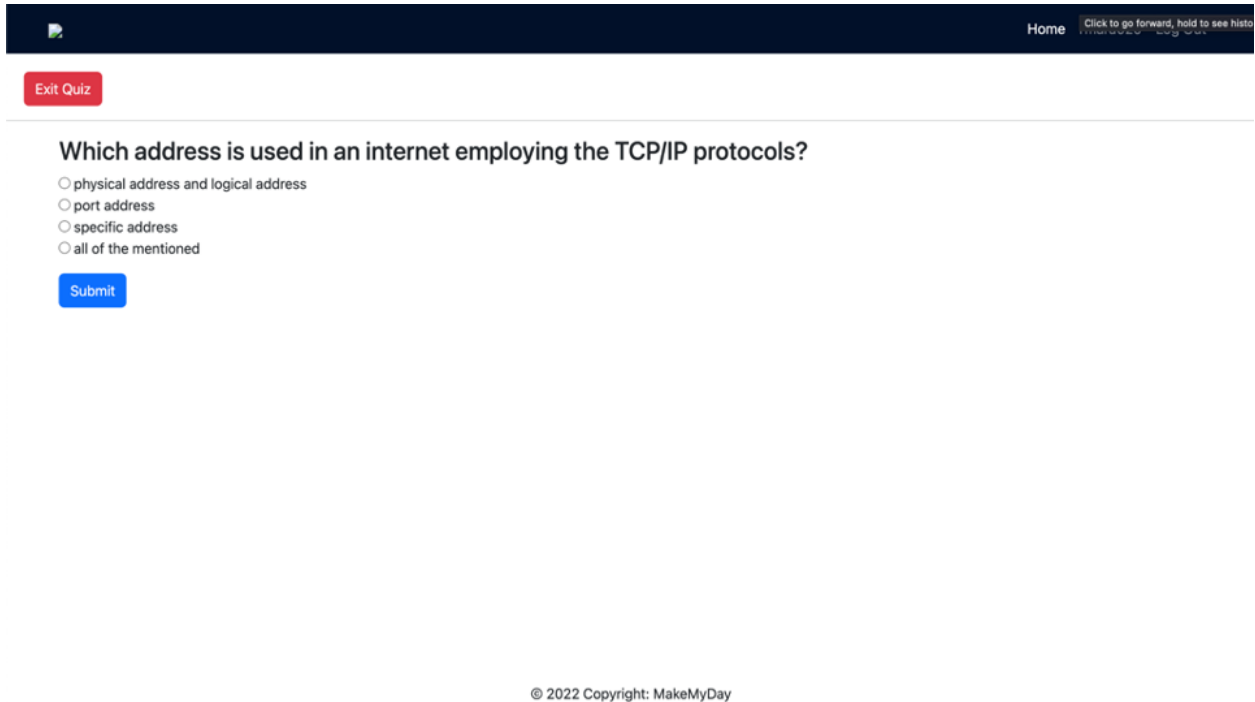
Homepage (Logged in)

While logged in, the homepage should display cards of all the classes that the user is enrolled in, clicking on any of these courses will redirect the user to the view of that course. There will also be a card for students to enter an access code to enroll to another course. For professors, the cards display the classes they have created and the extra card is used to create an extra course of their liking.

This page contains a To-do list on the right-hand side of the page which summarizes the questions due today for the student that is logged in. The professor does not have a To-do list.

Course Page

Quiz Page



Which address is used in an internet employing the TCP/IP protocols?

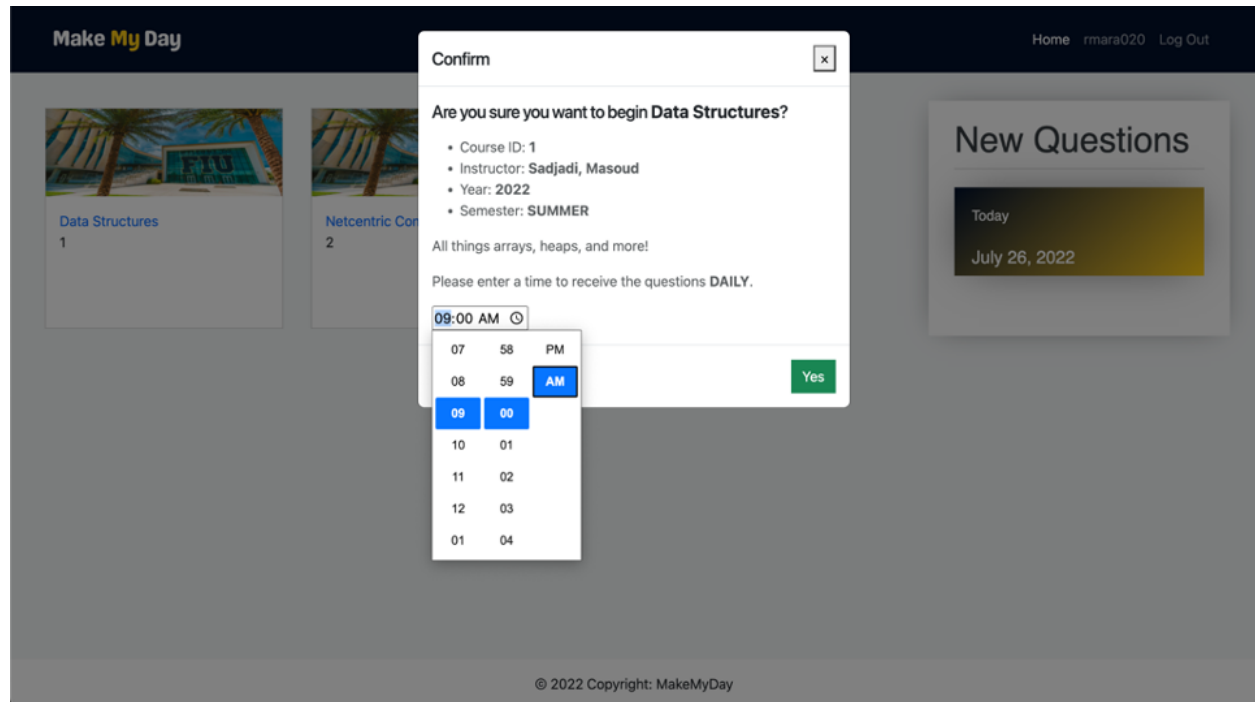
- ☐ physical address and logical address
- ☐ port address
- ☐ specific address
- ☐ all of the mentioned

Submit

© 2022 Copyright: MakeMyDay

Assuming that a student is logged in and a question is readily available for them to take. The quiz can be accessed either through the course page or in the to-do list on the right of the homepage while signed in by clicking on the question itself. You will be redirected to the page above where the question is displayed. Select the answer by clicking on it and press “Submit” when ready. If you would like to exit this quiz, the “Exit Quiz” button is on display at the top left. After submission, the result of this question will be displayed along with an explanation of why it’s correct.

Registering for a course



As a student, to register for a course you must have received an access code from the professor teaching the course you want to enroll in. After entering this access code, you will be displayed this screen confirming the details of the course. If everything looks good, select the field with time and select the time to receive questions for this class daily. The time selected be used to send you a question and a reminder at the same time daily. Once the time has been selected press “Yes” to save and confirm your desired response.

Installation/Maintenance

Running With Docker:

Make sure you have docker installed:

<https://docs.docker.com/get-docker/>

In the command line, make your way to the *makemyday* directory (the outer one, not the app) and run:

```
docker-compose up
```

The website should be on <http://127.0.0.1:8000>

Docker Setup With Windows:

Proceed to website and install Docker for Windows:

<https://docs.docker.com/get-docker/>

Go to Windows Features on your computer and enable Hyper-V and Containers.

If a message stating WSL 2 installation is incomplete pops up, go to this link <https://aka.ms/wsl2kernel> and download the latest package.

In the command line, make your way to the *makemyday* directory (the outer one, not the app) and run:

```
docker-compose up
```

If the above method does not work, below is a more detailed alternative way.

Alternative Installation:

Make sure you have python installed: <https://www.python.org/downloads/>

1. Create a folder and place the repository inside
2. Run *python3 -m venv /the/folder/where/repository/lies*
3. activate virtual environment from the folder: *source bin/activate*
4. Inside virtual environment run *pip install -r requirements.txt*
5. Install RabbitMQ

On Mac:

<https://www.rabbitmq.com/install-homebrew.html>

<https://code2care.org/pages/permanently-set-path-variable-in-mac-zsh-shell>

On Linux:

<https://www.youtube.com/watch?v=fBfzE0yk97k>

On Windows:

<https://www.youtube.com/watch?v=8lnybIaDz2M>

Running The Website:

The steps below detail the commands needed to run the website for Mac. They should be somewhat the same for other operating systems.

Steps **1-4** are only for the email/notification system.

Steps **2-5** make sure you are in the makemyday directory (the outer one, not the app itself).

terminal 1: *rabbitmq-server start*

This starts the messaging queue for the asynchronous sending of Email.

terminal 2: *celery -A makemyday worker -l info*

This creates the celery worker for handling emails to be sent out.

terminal 3: *celery flower -A makemyday --port=5555*

For debugging purposes. URL: <http://localhost:5555/>

terminal 4: *celery -A makemyday beat -l info --scheduler
django_celery_beat.schedulers:DatabaseScheduler*

This is for checking the database for any scheduled emails to be sent out.

terminal 5: *python manage.py runserver*

Running the website itself.

Shortcomings

See [Pending Use Cases](#) above

Wishlist

- Deployment
- Use of MySQL
- Cleaner website UI