# Swift Exam Check-in/out

Team Should Be Fine:

Andrew, Edward, Karl, Zachary, Jason, Alejandro

Our project sponsor, Dan Biediger, tasked us with building a web app to facilitate swift check-in and check-out for exams using UH student IDs. For in person exams, the only existing method of taking a check-in/check-out log of attending students is by hand which has the prominent downside of being very time intensive and inefficient. This method also reduces time available to students to take the exam and the lines of students waiting to check-out are disruptive to students still taking the exam. Any long term solution needs to be seamless and fast and use an authentication method all students are guaranteed to already have (their UH ID). This attendance log for exams, while not required, can be very helpful when looking into issues of academic integrity. One such use case is for professors who want to run exams in Canvas without using Respondus Lockdown Browser where they need to be able to verify that students completed the Canvas exam (start stop times are recorded in canvas) only while in the classroom (not leaving the room part way through and completing the exam elsewhere without TA proctoring). Our app needs to help fill that need by allowing for student check-in and check-out in a practical, fast, and simple way.

Our app needed to have a variety of check-in methods and we chose to support card swipes with a UH student ID card, manual ID entry, or QR code scan from the UH app. Professors need to be able to create and edit courses, upload student rosters for courses, manually edit individual students (in case of drops or information updates), assign TAs to courses, create exams, and export and view exam attendance logs. All of this information needs to be stored in a MySQL database and the front end needs to be able to accessed from a browser (preferably from both laptop and tablet type devices). Role and user management is also important. Professors need to only be able to see their own courses and rosters and TAs need to only be able to see courses to which they are assigned and only have specific edit permissions.

We decided to build our web app in Django because it is a python framework focused on web app development. A few of its notable selling points is it being able to seamlessly link with MySQL databases and having user account support built in. We used GitHub for version control and collaboration; we made extensive use of branches to facilitate parallel development of features.

The most important feature of the app is the ability to quickly check students in and out with their student IDs or the UH Go app. The card scanner is very simple because it reads in as a keyboard input so the info is extremely fast and easy to parse. The QR code function using the built in camera was significantly harder. We tried several different approaches for detecting and using the camera and then deciding if a QR code was in frame and then decoding it. Our first attempts were very slow and required manual image taking which introduced a lot of potential for user error and took forever. Eventually Zach figured out a way for the app to frictionlessly identify a QR code in frame and then automatically snap a still frame and decode the student ID. With this, the QR code is actually the fastest way to check students in and out and we believe it will be super helpful.

Mid way through the project we decided to convert our visuals and layout structure to use Tailwind CSS. We liked Tailwind CSS because it has a lot of predefined classes built on top of vanilla CSS that cleaned up our in-page styling and made page-to-page consistency easier. This required a complete rewrite of all of our visual pages in parallel with ongoing feature development. However Tailwind CSS let us establish a base page template that all other pages inherit and common styling classes in a way that would have been more difficult or impossible with vanilla CSS. This greatly reduced the amount of duplicate code blocks in our HTML pages and made adding new pages to the app easy and error free.

Django can be set up with MySQL to take custom classes called Models and export them as a database schema to a running MySQL server. The database can then be dynamically edited and updated by making changes to the model classes and exporting to the MySQL server. This eliminates any errors that might result from having to maintain the database by interacting with it directly when changes to the schema need to be made. Another key feature we implemented via Django is site sessions and user-personalized views. We had to incorporate tables in the database to store and manage user authentication and track usage sessions. This enabled us to show only the data owned by the specific user such as a professor only seeing the courses they created and only being able to see students who are enrolled in their courses. This means that if a student is enrolled in multiple courses created by different professors, the respective professors will only see the student in their course (and not that the student is also enrolled in another course owned by another professor, even though the student object itself is shared across both courses behind the scenes). Implementing this segregation of information via user sessions was very important to making our web app follow data privacy rules and respect student privacy as well.

A huge challenge for all of us was the tools and frameworks we needed to use we had to first learn. The Django framework and its connection to MySQL, Tailwind CSS, email verification services, Jenkens CLI, etc were all new to us. We essentially learned these tools by building this app and we made a ton of mistakes and learned a ton about the quirks and specifics of these tools by research, trial, and error. This introduced a lot of friction into our early progress (at the start just Django and later on converting the project to use Tailwind CSS) but also made our learnings feel really valuable and rewarding.

A lot of our other challenges are mirrors of our coolest features. Having to research and try several different potential solutions before finding one that worked was usually the source of greatest friction for us throughout the semester. Most of us had fragmented at best knowledge of how to make a python web app so we were all learning on the go and every time someone tried something it was a learning experience for us all.