**SMSE Onboarding**

Team members: Shayna Bello, Zixiang Zhou, Long Pham

Sponsor: Michelle Sztupkay

Instructor: Jennifer Olsen

[**Project Summary 2**](#_7pk395s5ow5t)

[**Problem Statement 2**](#_oirwt4pwp9qy)

[**Stakeholders 2**](#_zgamt9cmggmq)

[Client/customer stakeholders: 2](#_rx89lg8lxjlo)

[End user (user roles) stakeholders: 3](#_a8nk1y6n2lta)

[**Requirements 3**](#_4izbbqj5tjk4)

[Functional Requirements: User Stories 3](#_vojomu2kofw4)

[Non-Functional Requirements 5](#_szye6dbxiq1n)

[Security 5](#_norn2slmtb9r)

[Documentation 5](#_geot18tjvrsr)

[Performance 5](#_kut2nbtdaufr)

[Other 5](#_1k0ue8t1z3lu)

[**Ethical and Accessibility Considerations 6**](#_f6mkiipkl1ir)

[Ethical Considerations 6](#_sn4z2boif93k)

[Accessibility Considerations 6](#_s7t9wnduek9i)

[**Design Plan 6**](#_8x03rsm3jspm)

[System Models 6](#_yagls19zv08h)

[Software Architecture 8](#_97402j69u370)

[Data Architecture 9](#_tqbohqogvjhu)

[External / Pre-Existing Data Sources 9](#_xl4lm5nkqf2n)

[Relational Database Info 10](#_f0xkrbpa3or)

[Database Tables 10](#_amf7ca5cbf33)

[Relations 12](#_8mp56fdzja3i)

[Non-Relational Database Info 12](#_jyr16fcofk2)

[Pre-Existing Databases 12](#_hebg57qg7tho)

[User Interaction Architecture 12](#_kygx9zxpfwzn)

[Proposed Development Environments 13](#_cbnbupjpw06m)

# Project Summary

The current new hire onboarding process at the Shiley Marcos School of Engineering (SMSE) is very inefficient. Currently, there is no central location where new hires and administrators can track onboarding progress. Both parties have to juggle email chains and write their own reminders to keep track of everything. This leads to delays, missing deadlines, and many more headaches for administrators and new hires alike.

# Problem Statement

The SMSE onboarding process for new hires is decentralized and inefficient. New hires are assigned multiple tasks to complete during the onboarding process such as get their USD account and get their office. Meanwhile, administrators have to manage multiple new hires along with what tasks they need to complete. Additionally, this process involves other departments such as Human Resources for processing onboarding documents and IT for setting up the new hire’s equipment.

New hires have to manage many tasks during the onboarding process. However, without a centralized location to view their tasks, new hires risk forgetting certain tasks and missing important deadlines. Additionally, administrators have to manage many new hires and their tasks. However, without a centralized location to view the new hires and their tasks, administrators may forget about certain new hires and their tasks. Administrators may also forget to send reminders to new hires about certain tasks. All of these factors lead to delays in the onboarding process and miscommunication between the administrators and new hires.

The SMSE Onboarding team will develop a web-based onboarding application that streamlines the onboarding process for new hires in the school of engineering. This solution will centralize all onboarding tasks, allowing new hires to efficiently track and complete assigned tasks. This solution also allows administrators to monitor new hires’ progress in real time, assign tasks, and send reminder notifications. By providing a clear and organized platform for both parties, the application will ensure tasks are completed on time, reduce administrative overhead, and foster a positive onboarding experience for new hires.

By addressing inefficiencies in the current system, the onboarding application will improve communication between administrators and new hires, mitigate delays, and enhance the overall efficiency of the onboarding process. This will help ensure that new hires can quickly integrate into USD and become productive members of their department while reducing the administrative workload on SMSE administrators.

# Stakeholders

## Client/customer stakeholders:

**SMSE Admin Team (Michelle Sztupkay, Paula Schmid, Ryan Smith, Hattie Morgan)**

**Description:** These are members of the SMSE admin team who, along with other tasks, manage aspects of the onboarding of new hires.

**Goals:**

* Stay informed on and track the progress of new hires’ onboarding.
* Have a centralized location to complete administrative tasks for SMSE new hires.

**Pain Points:**

* There is a lack of visibility of new hires' onboarding progress.
* Staying on top of deadlines is difficult.

## End user (user roles) stakeholders:

**SMSE Administrators (includes deans and department heads)**

**Description:** These are SMSE faculty members who have administrative privileges. They need to keep track of the new hire onboarding process and be able to update hiring information.

**Goals:**

* Have an overall checklist for each new hire onboarding process.
* Be able to send out messages automatically to new hires regarding their onboarding process.
* Have a better database that holds faculty information and is convenient for checking out the information of faculty members and new hires.

**Pain points**:

* Information is all over the place and is decentralized.
* It is hard to keep track of deadlines for tasks that need to be completed.
* It is difficult to keep track of the new hire onboarding process.

**New Hires**

**Description:** These are newly hired SMSE faculty members, with “new” being a relative term as the onboarding process takes a considerable amount of time. These include those being hired for full-time positions as well as adjunct positions.

**Goals:**

* Have better guidance on how to do each step in the onboarding process.
* Provide the correct links, locations, and contact information for the resources and people involved in the onboarding process.
* Have a centralized location to view all onboarding tasks as well as find resources and points of contact for onboarding tasks to be completed.

**Pain Points:**

* Onboarding tasks are unclear and confusing.
* Deadlines are hard to keep track of.

# Requirements

## Functional Requirements: User Stories

**User Role: SMSE Administrator**

1. As an SMSE administrator, I want to check where new hires are in their onboarding process so I can easily keep up-to-date with their progress and make sure they complete all their tasks on time.
2. As an SMSE administrator, I want to have easy access to new hires’ bios and CVs so I can quickly find certain information for administrative purposes.
3. As an SMSE administrator, I want to receive reminders for tasks that need to be completed for the onboarding process so I can avoid delays and make sure all tasks are completed efficiently even if I forget specific tasks.
4. As an SMSE administrator, I want to be able to automate adding new hires to mailing lists so I can streamline integrating them into communication channels, making it easier for them to receive updates from the very beginning.
5. As an SMSE administrator, I want to be able to automatically send a welcome gift to each new hire so that they feel appreciated and welcomed into the school of engineering which will promote a positive onboarding experience.
6. As an SMSE administrator, I need to send new hire information emails and welcome emails to new hires and the rest of the faculty so that the entire school of engineering can welcome the new hires.

**User Role: New Hire**

1. As a new hire, I need to automatically be signed up for mailing lists instead of manually doing it myself so that I can stay informed and connected to the SMSE community.
2. As a new hire, I need to be able to check and track all my onboarding tasks and progress from one location so that I can complete each onboarding task in a timely manner and avoid missing any important steps.
3. As a new hire, I want to be routed to all the locations and points of contact that I need for my onboarding process so that I can save time and avoid searching for the information on my own.
4. As a new hire, I want to be able to send and receive messages from the SMSE administrators about my onboarding process so that I can easily address any questions and issues and stay informed on each step of the process.
5. As a new hire, I want to be able to request a computer so that I can start working as soon as possible with the necessary equipment.
6. As a new hire, I want to be able to obtain key card access so that I can have access to my work facilities and workspaces that I need to complete my job responsibilities.
7. As a new hire, I want to be able to obtain a Zoom telephone number so that I can participate in and host Zoom meetings in order to collaborate with students and colleagues.
8. As a new hire, I want to be able to request furniture for my office so that I can have the necessary items to create a comfortable and productive work environment.
9. As a new hire, I want to obtain name tags and business cards so that I can easily represent myself at USD.
10. As a new hire, I need to obtain a mailbox so that I can receive work-related mail.
11. As a new hire, I need to obtain an office key so that I can secure my office and protect my personal belongings.
12. As a new hire, I need to attend the New Employee Orientation so that I can learn about all the resources available to me and integrate successfully into the USD community.

## Non-Functional Requirements

### Security

* The application should secure and protect users’ data so that the data is kept private and protected from any unauthorized access.

### Documentation

* The application should be easy to update and maintain in the years to come.
* The application should follow the guidelines for accessibility and be accessible to all users regardless of their disabilities.

### Performance

* The application should have good performance and reliability. The application should load the correct information in under 1 second with a stable internet connection.
* The application should be easily scalable so that it can easily manage the growth of the engineering department along with the arrival of new faculty members and administrators.

### Other

* The application’s interface should be user-friendly and not too complex. It should be easy for users to navigate the interface and find the necessary information.
* The application should be compatible with most web browsers such as Firefox and Safari. The user should also be able to access the application from their mobile device.

# Ethical and Accessibility Considerations

## Ethical Considerations

* Since our project will contain a database, an ethical consideration we have concerns data privacy. We are concerned with how to keep users’ data secure and private in the database. We will ensure the transparency of the data we collect so that users will be well-informed and know what data we are collecting. We will also consider allowing the user to choose what information they want stored in the database.
* Another ethical consideration we have concerns getting users’ consent. We want to make sure that users can consent to having their data stored in our application’s database. We plan for our application to have a pop-up message when the user first opens up the application. The pop-up message will ask for the user’s consent to have their data stored in the application. An edge case for this consideration would be if the user does not consent. In that case, some components of the application will be disabled and the user will have the option to complete paperwork instead.
* Another ethical consideration we have is ensuring that our application promotes inclusivity by being as accessible as possible and removing as many obstacles as possible. By doing this, we can make sure that everyone, including people with disabilities, can easily use our application.

## Accessibility Considerations

* Our onboarding application will be as accessible and user-friendly as possible for all people, including people with disabilities. We plan to use Django and will work to apply 3rd-party libraries with built-in accessibility features like text-to-speech, semantic HTML, ARIA labels, and appropriate colors. These modalities are useful for considering visual, mobility, hearing, & cognitive impairments.
* Our application will follow standard accessibility guidelines as directed by The World Wide Web Consortium. This will ensure that our application can be accessible for everyone including those with disabilities and will be critical to making the user experience as fluid as possible.

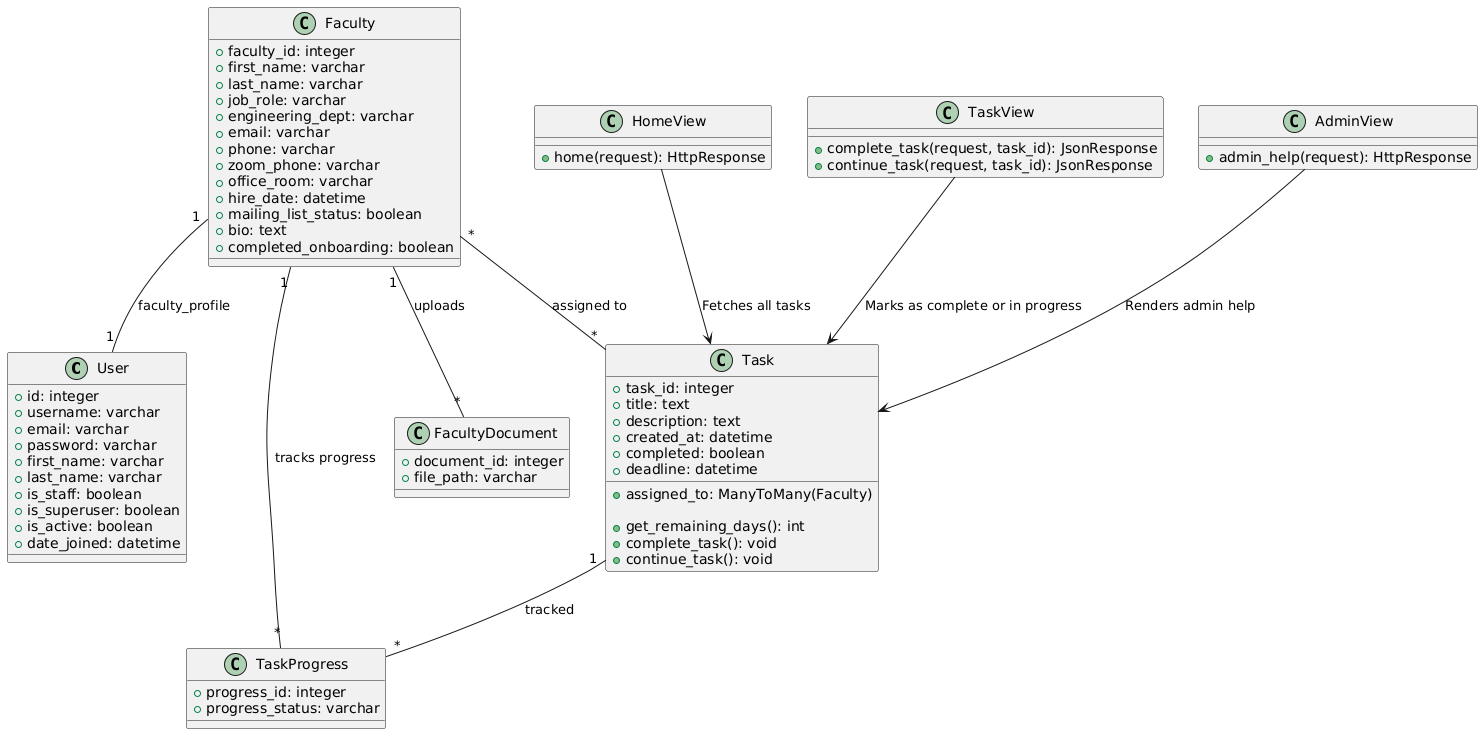
# Design Plan

## System Models

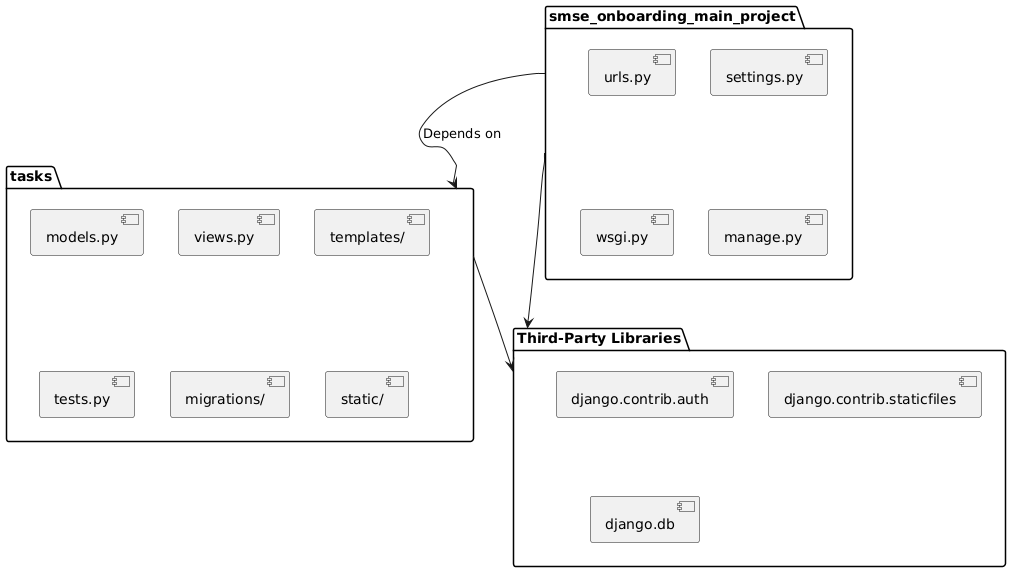
1. Static Diagrams - Class Diagrams and Package Diagrams

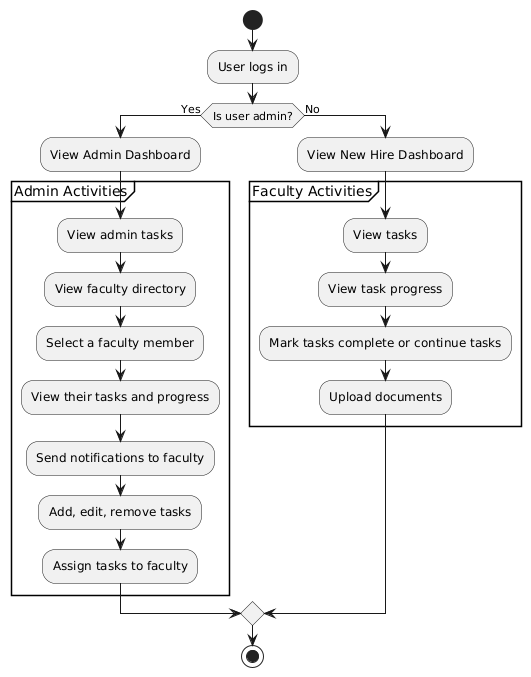
**SMSE Class Diagram [SPRINT 2 UPDATE]:**

Django has a built-in auth.User class that our Faculty class and Admin class can inherit from.



**SMSE Package (app) Diagram :**



1. Dynamic Diagrams - Sequence Diagrams and Activity Diagrams

## Software Architecture

We have chosen to use the MVC architecture for this project. We chose this architecture because our project is a web application where the user makes requests to the application. Our application needs to process and handle different types of user requests. Our application also needs a database to maintain information about the users (phone numbers, emails, etc.), tasks, and relevant documents. The application will be using Django for both the frontend and backend which takes over the roles of both the controller and view in a traditional MVC architecture.

* Model
  + PostgreSQL: PostgreSQL will be used to produce our database which handles structured data like user information, tasks, onboarding progress, and documents.
* View
  + Django and Bootstrap: We’ll be using Django templates and Bootstrap to create the user interface, styling, layout, and responsiveness.
* Controller
  + Django: We’ll be using Django to handle all the backend logic including CRUD operations. The controller accepts HTTP requests from the user and then returns HTTP responses to a Django template with the retrieved data from the database model. Django will act as a bridge between the Model and View.



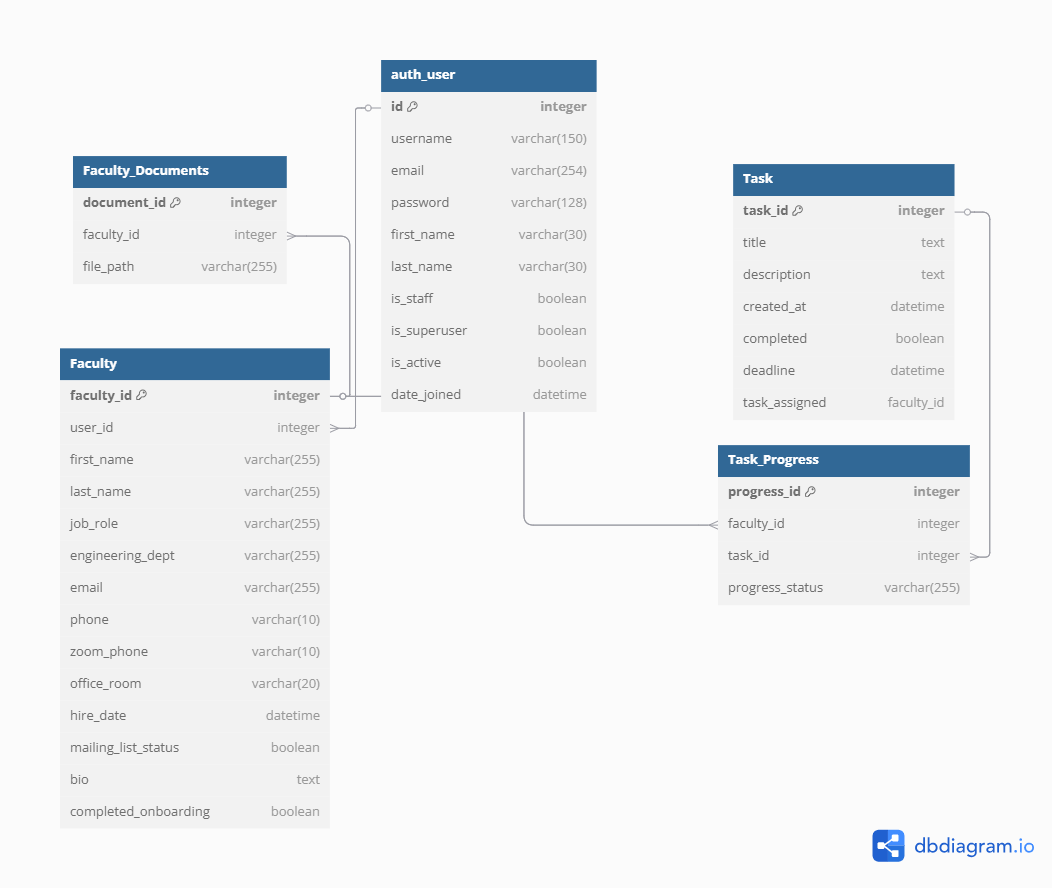
## Data Architecture

### External / Pre-Existing Data Sources

* **Data Source 1:**
  + **Description:** This is a spreadsheet containing the SMSE faculty directory. It is up-to-date and contains information about faculty members, including their names, phone numbers, office locations, and emails. The directory also separates the rows based on departments (computer science, electrical engineering, mechanical engineering, etc.).
  + **Type:** Structured
* **Data Source 2:**
  + **Description:** This is a spreadsheet containing the SMSE new hire onboarding tasks. It is up-to-date and contains information about new hire onboarding tasks, including their names, timelines, involved departments, and who initiates the tasks.
  + **Type:** Structured

### Relational Database Info [SPRINT 2 UPDATE]

#### Database Tables



1. Faculty
   * Primary Key Name: faculty\_id
   * Columns:
     1. faculty\_id: integer
     2. user: User
     3. first\_name: varchar(255)
     4. last\_name: varchar(255)
     5. job\_role: varchar(255)
     6. engineering\_dept: varchar(255)
     7. email: varchar(255)
     8. phone: varchar(10)
     9. zoom\_phone: varchar(10)
     10. office\_room: varchar(20)
     11. hire\_date: date/time
     12. mailing\_list\_status: boolean
     13. bio: text
     14. completed\_onboarding: boolean
2. Task
   * Primary Key Name: task\_id
   * Columns:
     1. task\_id: integer(10)
     2. title: text
     3. deadline: date/time
     4. created\_date: date/time
     5. completed: boolean
     6. assigned\_to: varchar(255)
     7. description: text
3. Task Progress (join table)
   * Primary Key Name: progress\_id
   * Columns:
     1. progress\_id: integer(10)
     2. faculty: integer(10)
     3. task: integer(10)
     4. progress\_status: varchar(255)
4. Faculty Documents
   * Primary Key Name: document\_id
   * Columns:
     1. document\_id: integer(10)
     2. faculty: integer(10)
     3. file\_path: varchar(255)

#### Relations

* One-to-many relationships:
  + Faculty → Task Progress
  + Faculty → Faculty Documents
* Many-to-many relationships:
  + Faculty → Task:
    - The Task Progress table will serve as a join table, linking the Faculty and Task tables in a many-to-many relationship. Each faculty member can be assigned multiple tasks and each task can be assigned to multiple faculty members.
* One-to-one relationships:
  + N/A

### Non-Relational Database Info

Non-relational databases were considered but deemed unnecessary due to the structured nature of our data (faculty, tasks, onboarding progress) which fits well into relational databases.

### Pre-Existing Databases

According to the SMSE administrators, there is a website and existing database that store the resume and bio information for faculty members. Also, the task sheet and SMSE directory have already been created.

## User Interaction Architecture

Users will interact with the system through a web UI which will be built using Django templates and styled using Bootstrap. There will be 3 main pages:

**Login Page:** Every user has to log in to the application and they will be identified as either an administrator with special permissions or a new hire faculty member. If they are an administrator, they can have access to both the faculty page and the admin page. If they are a new hire faculty member, they will be directed to only the faculty page.

The login page will be basic and contain two text fields for the user to enter their USD username and password. Below those text fields will be the login button for the user to click after entering their username and password.

**Faculty Page:** The faculty page will have a new hire dashboard and will allow new hires to see their assigned tasks, track their overall progress, and check things like whether or not they are part of the mailing list. They can also mark their own tasks as complete and upload documents like bios and resumes. Checking off tasks and uploading documents trigger the backend and database to update the new hire’s progress.

**Admin Page:** The admin page will have an admin dashboard where the admin user will be able to see a more detailed list of all the faculty members including new hires. The admin user will also be able to view faculty members’ tasks and progress, add tasks, delete tasks, mark tasks as complete, download bios and resumes, check whether or not someone is on the mailing list, and send reminder emails for specific tasks for any faculty member. Bootstrap will be used to make any tables and dropdown components needed for this page. There will also be a button for the admin users to assign new tasks to faculty members which will trigger the backend to update their progress and send email notifications.

Django has a built in auth.User class the faculty and admin classes can inherit from. Also, Django already has a way to set permissions and distinguish between admin users, non-admin users, and other users that have admin-like privileges. Additionally, an other\_employee database table may be necessary to track other staff members who interact with new hire users.

## Proposed Development Environments

**Frameworks + Libraries**

* Django
  + We will be using Django for both the frontend and backend parts of our application. Django has a templating engine that allows it to handle frontend requirements. The reason we chose Django is because it is a full stack framework that uses Python, a language that the entire team is knowledgeable in. Also, Python is a very easy-to-read and easy-to-use language.
  + Django is also known for being a great framework for backend development. It already has a lot of built-in functionality for handling things like email notifications, form submissions, doing CRUD operations on databases, and more.
  + We won't have to use that many 3rd-party tools to fulfill our project requirements. This makes it a lot easier to maintain the application without having to keep track of updating multiple 3rd-party tools or figuring out how to integrate them. Django already has an email framework built in so we wouldn't have to find a 3rd-party tool for this. Python also has an admin interface built in which should prove helpful for us.
  + Django also has built-in features such as authentication and email notifications which reduce development overhead and align with our project requirements for the onboarding process.
  + Django’s ORM will manage the interaction between the application and the PostgreSQL database, allowing us to handle most data operations without writing raw SQL. This ensures that the development team can focus on business logic while maintaining a robust and secure data layer.
* Bootstrap
  + Some of our team members are already pretty familiar with Bootstrap. It is a highly popular and well-documented UI library that has pre-built components that adhere to standard accessibility guidelines. Some of the components it has are things like tables, forms, buttons, progress bars, and checklists. All of these are things that we require in our application. Also, it is a good way to maintain consistency throughout the entire application if we follow the look of one library. Bootstrap also has built-in responsiveness that should work for web browsers on a computer, tablet, or mobile phone which would be useful for users.
* PostgreSQL:
  + We’ll be using PostgreSQL because it is highly documented and works well with Django. It is easy to use Django to set up a PostgreSQL database.
  + It is efficient, lightweight, and easy to set up and use for local testing. It is able to handle large volumes of data which helps to support scalability and contain more faculty members' and administrators’ data over time.
  + PostgreSQL was chosen for its widespread use, scalability, and ability to handle the structured data required for this application.
  + We are using PostgreSQL as the relational database management system for both the development and production environments. This decision ensures consistency across environments, making it easier to manage the data structure and migrations during the transition from development to production.

**Languages**

* Python:
  + We will be using Python since Django is a Python-based framework.
  + We will be using this programming language because it is required when using Django. Also, we believe there are a lot of similarities between Django and HTML so Django should be easy to pick up alongside Python. It is also very easy to read and understand all the UI components when written using Python and Django.
* CSS/HTML:
  + CSS is the required web application styling language and it combined with Bootstrap will present a well-structured and professional look and feel for our web application. HTML is also a required language for web applications and pairs well with CSS.
  + CSS and HTML will give out simple but direct pages and will be used on the login page, task dashboard, document upload screen, and more. HTML will provide the foundation for our website’s frontend and allow us to create basic layouts for each webpage. CSS through Bootstrap will handle the styling and responsiveness of each webpage.