

# **Software Design Document**

## **Form Buster: Web-Based Registration, Approval, and Tracking**

### **Team Members**

**Daniel Acosta** ([dacosta2022@my.fit.edu](mailto:dacosta2022@my.fit.edu))

**Christopher Demuro** ([cdemuro2022@my.fit.edu](mailto:cdemuro2022@my.fit.edu))

**Alex Merino** ([amerino2022@my.fit.edu](mailto:amerino2022@my.fit.edu))

**Luka Miodrag Starcevic** ([lstarcevic2022@my.fit.edu](mailto:lstarcevic2022@my.fit.edu))

### **Faculty Advisor**

**Phillip Bernhard** ([pbernhar@fit.edu](mailto:pbernhar@fit.edu))

**Version 1.0**

**Feb 24, 2025**

# **Table of Contents**

- 1. Introduction**
  - 1.1. Purpose**
  - 1.2. Scope**
  - 1.3. Objective**
  - 1.4. Key Interactions**
- 2. System Architecture**
  - 2.1. UML Class Diagram - Core User and Form Classes**
- 3. User Interface Design**
  - 3.1. Landing Page/Non User Page**
  - 3.2. Home Page/Dashboard**
  - 3.3. User Login**
  - 3.4. Student User Registration**
  - 3.5. Inbox**
  - 3.6. Form Selection**
  - 3.7. Form Completion**
  - 3.8. Settings/Account**
  - 3.9. Edit Form Selection**
  - 3.10. Form Layout Editor**
- 4. Database Design**
  - 4.1. Entity Relationship Diagram**
  - 4.2. User Collection**
  - 4.3. Active Form Collection**
  - 4.4. Form Layout Collection**

# **1. Introduction**

## **1.1. Purpose**

The purpose of this project is to create a web-based application designed to host the storing, completion, and distribution of registration forms at Florida Tech. This document aims to describe the design of the Form Buster system.

## **1.2. Scope**

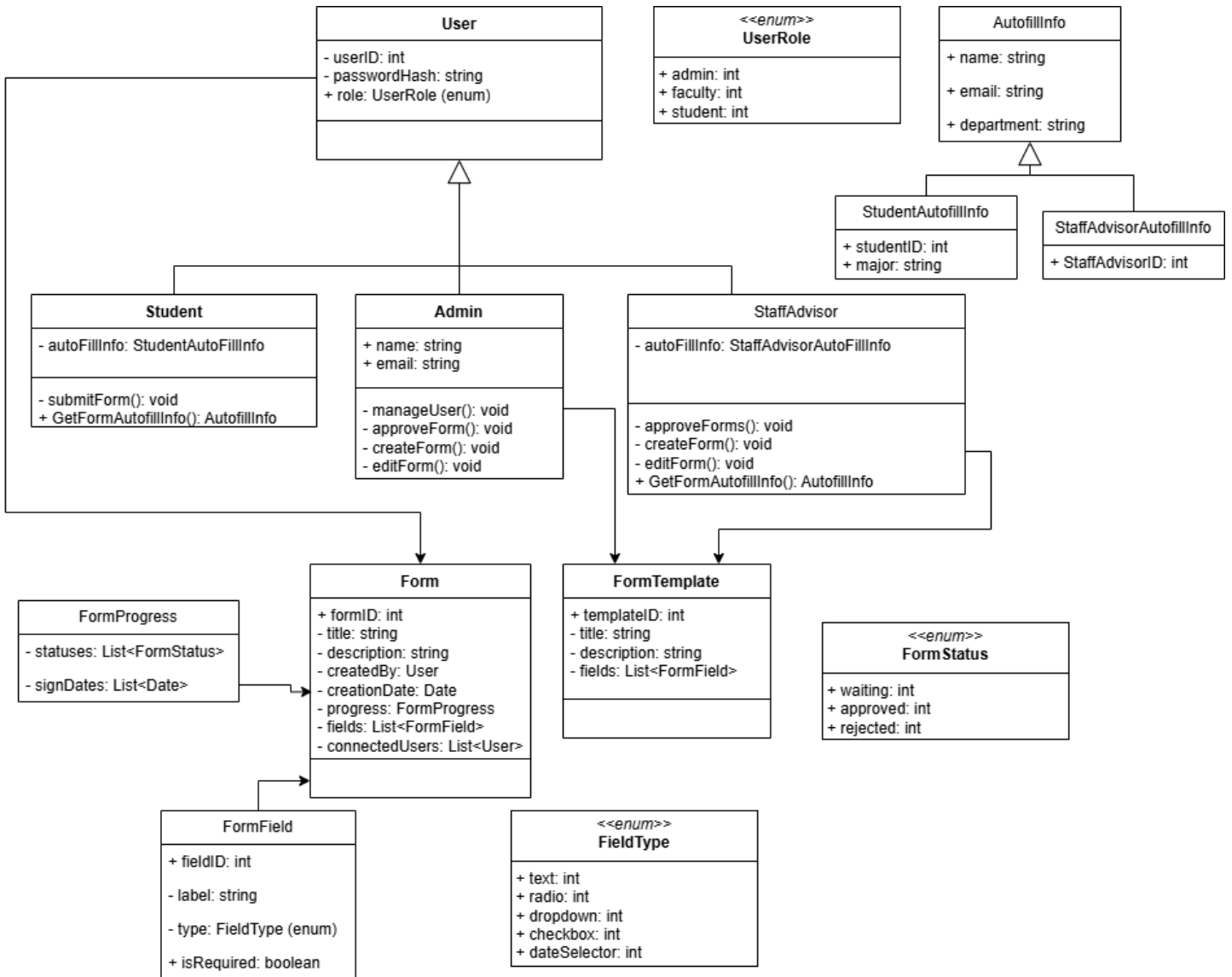
The scope of the Form Buster project is to replace the current registration process at Florida Tech. The idea is to create a registration portal similar to the Financial Aid portal that is in place at Florida Tech, which houses all registration forms and allows students, staff, and administrators to complete them.

## **1.3. Intended Audience**

The intended audience for the project is all students, staff advisors, and registrar employees/administrators and Florida Institute of Technology. A majority of the users will be the students so the project focuses on implementation specifically for those users. Another main functionality of the system is to allow for the administrators to modify forms, which is available to only administrators.

## 2. System Architecture

### 2.1. UML Class Diagram - Core User and Form Classes



#### Class Details:

- User:** This is a parent class for the 3 main user types of the application. It stores shared information such as the user's ID, their password hash, and the role assigned to the account. This class's primary purpose is to contain the data necessary to login, and allow its subclasses to specify additional information as needed per role.
- Student:** This is a subclass of the User class. The purpose of this class is to hold additional information relevant to students using the application, such as their student ID, name, email, department, and major. The student can submit forms after filling out a form template. The student's relevant information that can be autofilled during form completion

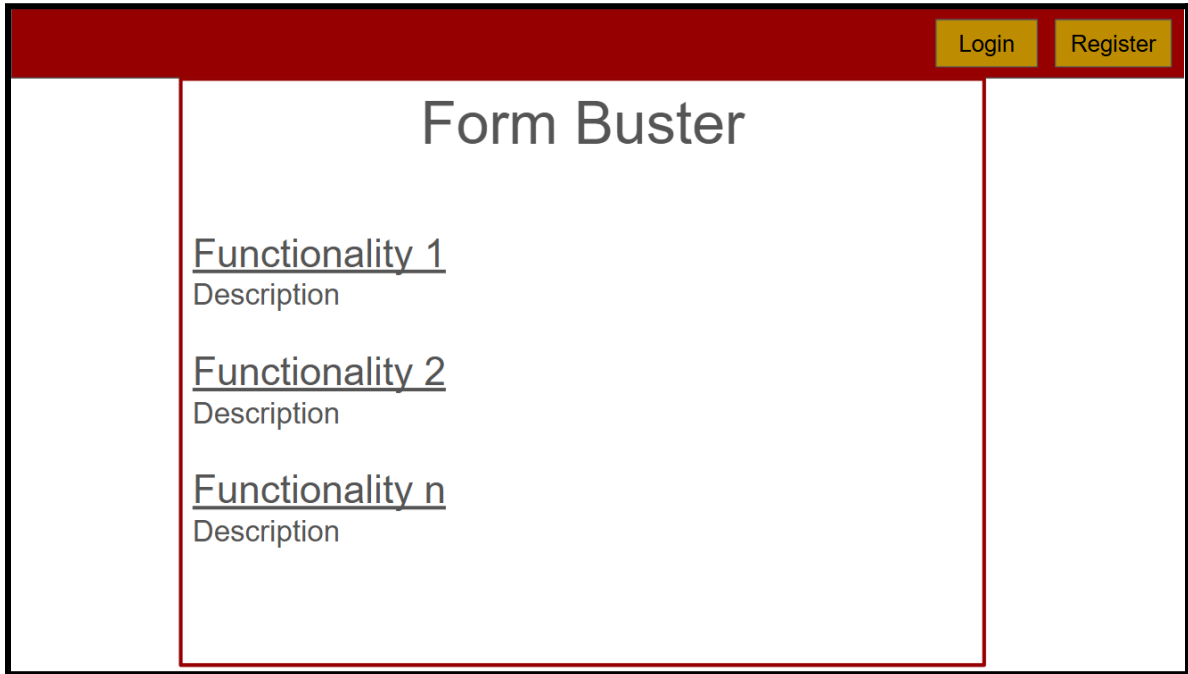
is stored in the StudentAutoFillInfo attribute, so that it is concentrated into one object that can be returned in its GetFormAutoFillInfo function as needed.

- **Admin:** This is a subclass of the User class. The purpose of this class is to hold additional information relevant to administrators of the application, such as their name and email. Admins do not need to autofill their information in any forms, so the name and email are stored individually in the class as opposed to being stored in an AutoFillInfo subclass. Admin users can manage users (update data, remove users, reset passwords, etc), approve forms, create new forms, and edit existing forms.
- **StaffAdvisor:** This is a subclass of the User class. The purpose of this class is to hold additional information relevant to staff advisors using the application, such as their staffAdvisorID, name, email, and department. The staff advisor can approve forms, create new forms, and edit existing forms. The staff advisor's relevant information that can be autofilled during form approval is stored in the StaffAutofill attribute, so that it is concentrated into one object that can be returned in its GetFormAutoFillInfo function as needed.
- **Form:** This is a class for completed forms that have been submitted by a user and are awaiting approval. The form has a form ID to help identify it, and the following attributes to describe the form's contents: title, description, and fields (a list of all of the fields in the form). The form stores the date of its creation as well as the creator of the form. The progress of the form's completion as well as the users that are connected to the form's approval process are also stored.
- **Form Template:** This is a class for templates of forms that will be submitted by various users. It contains the template's ID, the title of the template, its description, as well as a list of the fields that should be filled out when submitting an instance of the template.
- **FormProgress:** This class primarily serves as a data type to relate the statuses of form approval signatures to the dates on which the signatures were signed. This data type is vital to tracking the progress of the form through its approval process.
- **FormField:** This is a class for fields that will appear on forms. The field's ID, label and FieldType are stored in this class for when a form needs to be generated onto the user's screen. Additionally, the isRequired boolean is stored for flexibility in forms if some fields are optional.

### 3. User Interface Design

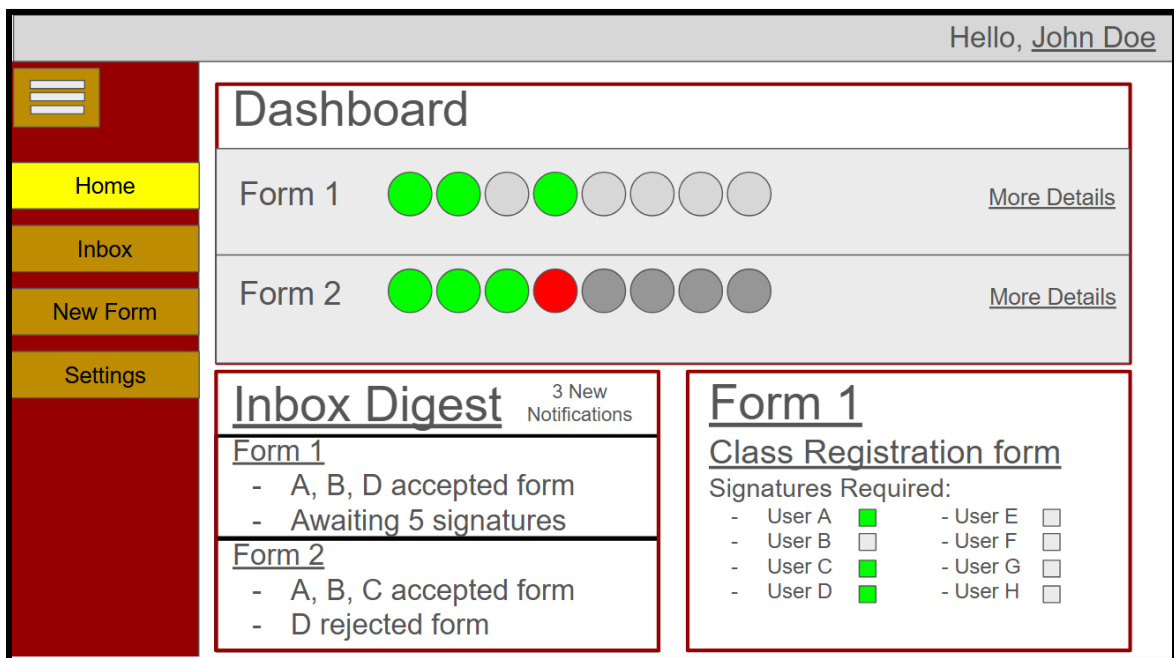
#### 3.1. Landing Page/Non-User Page

Landing page for the web application, mainly for logged-out users.



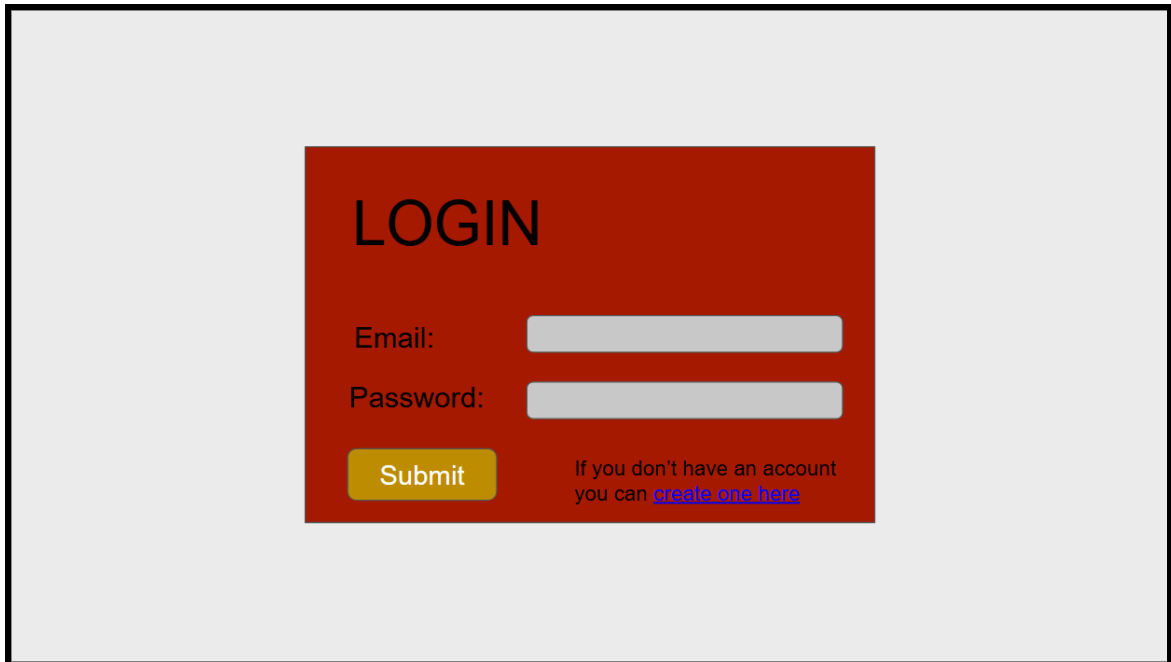
#### 3.2. Home Page/Dashboard

Home page for signed-in users with hamburger menu open.



### 3.3. User Login

A sample login page with email and password as inputs, and the grey boxes are editable inputs



**LOGIN**

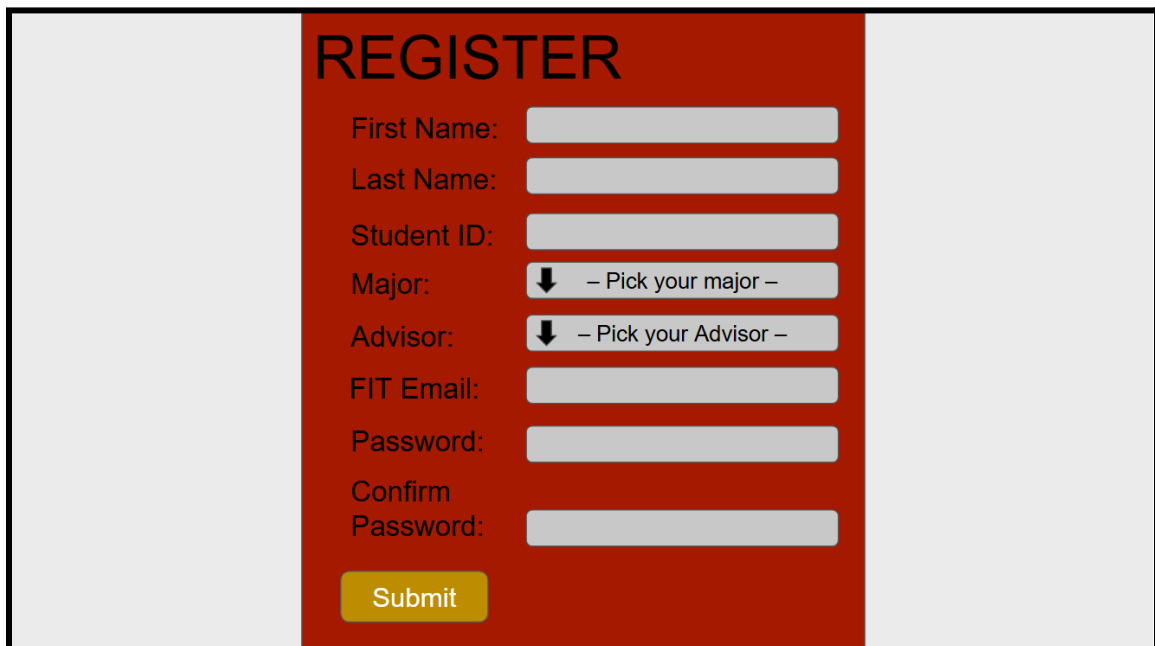
Email:

Password:

If you don't have an account you can [create one here](#)

### 3.4. Student User Registration

A sample student registration page, which asks for multiple inputs



**REGISTER**

First Name:

Last Name:

Student ID:

Major:

Advisor:

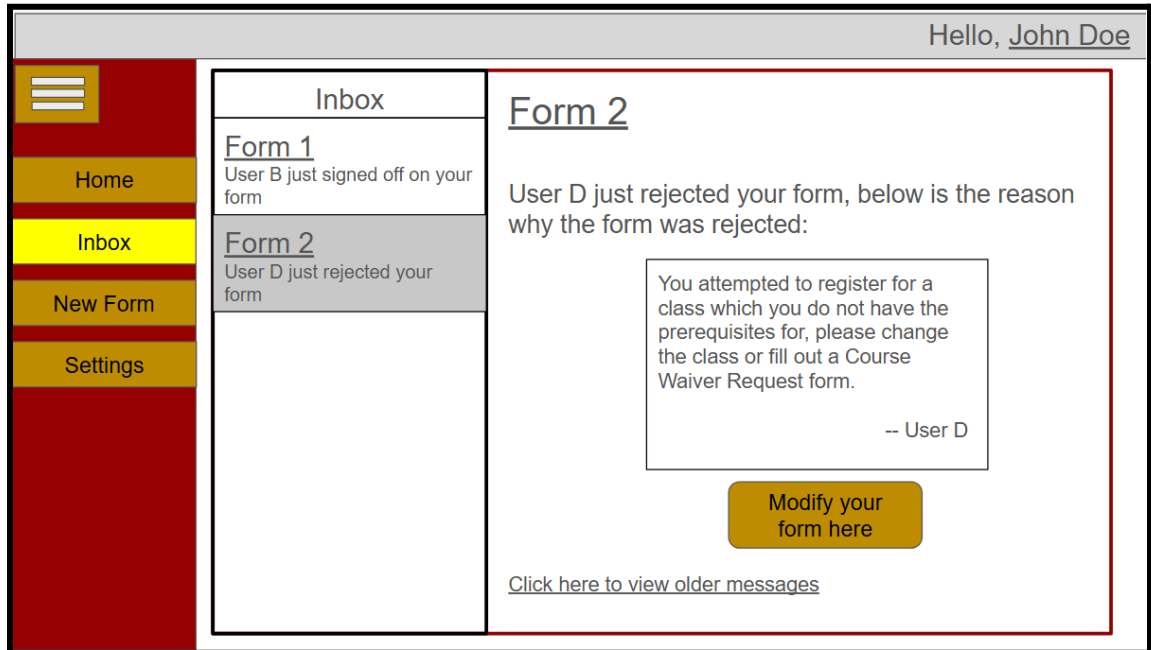
FIT Email:

Password:

Confirm Password:

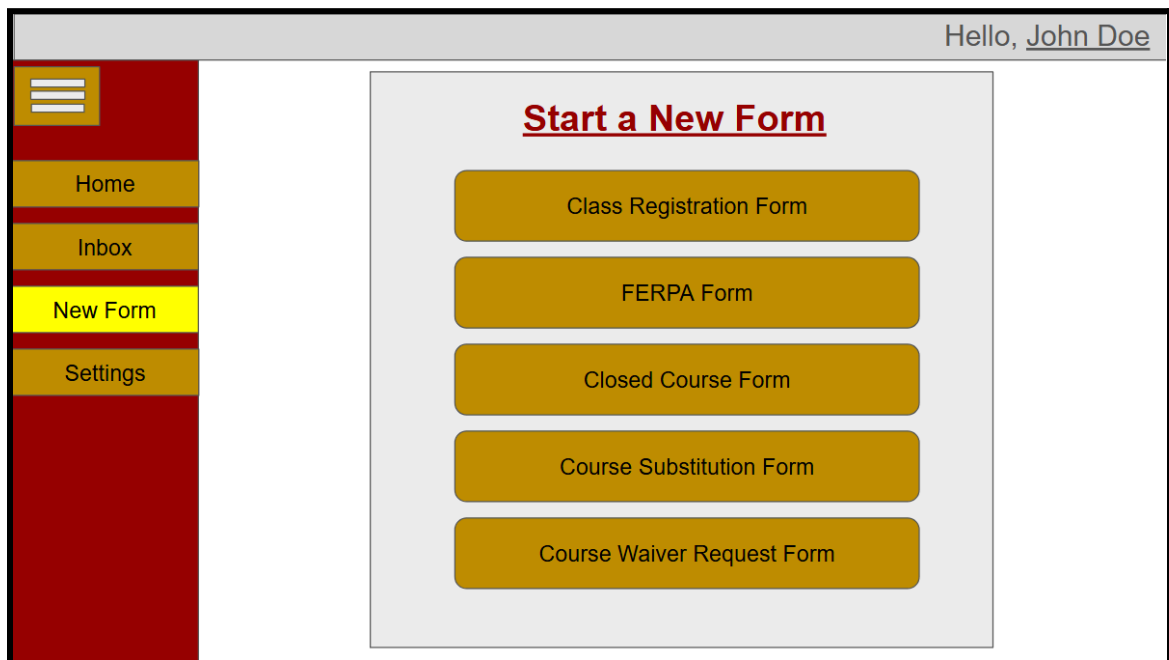
### 3.5. Inbox

A sample inbox page, similar to any email inbox, where users can view recent notifications on their forms and comments on these forms.



### 3.6. Form Selection

A sample page that holds the list of forms that a user can start. The forms shown are forms that will be added, but there will be more when implemented as well.





### 3.7. Form Completion

The page for a Class Registration Form contains inputs for all the information that a class registration form needs. The dark grey inputs denote that the inputs come from the User's account information and are auto-filled when the form is started.

Hello, [John Doe](#)

### Class Registration Form

First Name:

Last Name:

Email:

Student ID:

Major Code:

Advisor:

Add Classes:

Course Registration Number (CRN):

Add another class

### 3.8. Settings/Account

A sample Settings/Account page that allows the user to view their account information and settings they can modify.

Hello, [John Doe](#)

### Doe, John

### Settings

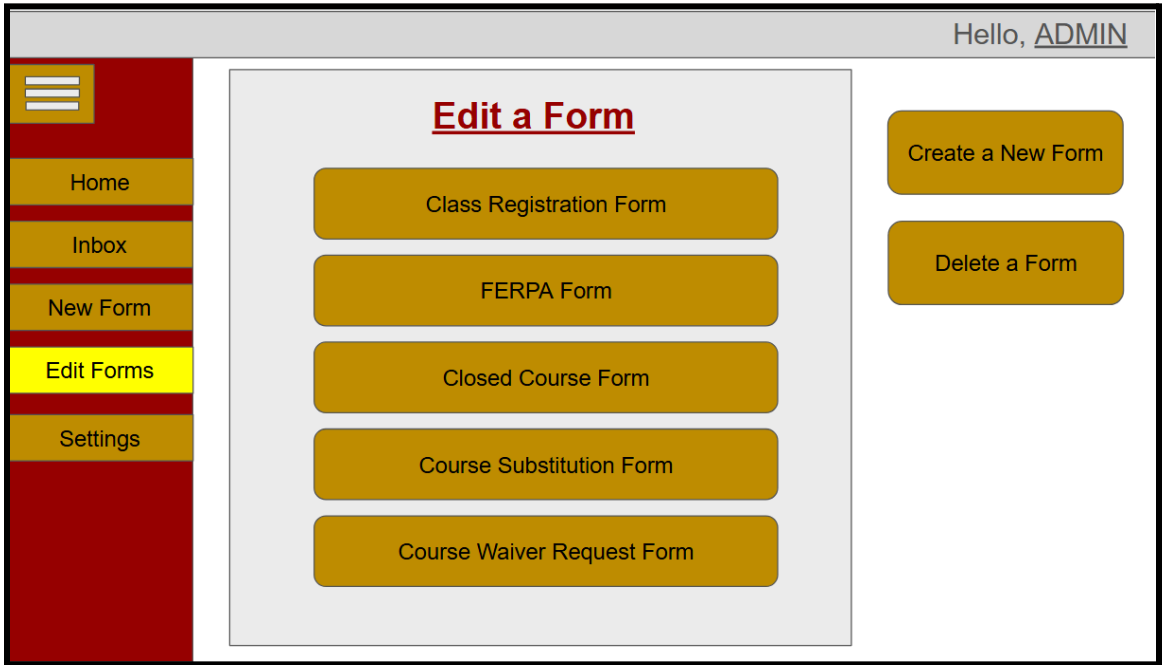
Email Notifications are enabled

### Account Information

Email: jdoe2025@my.fit.edu  
First Name: John  
Last Name: Doe  
Student ID: 900000000  
Password: \*\*\*\*\*  
- [View Password](#)  
- [Change Password here](#)  
Major: Computer Science  
Advisor: Dr. Chan

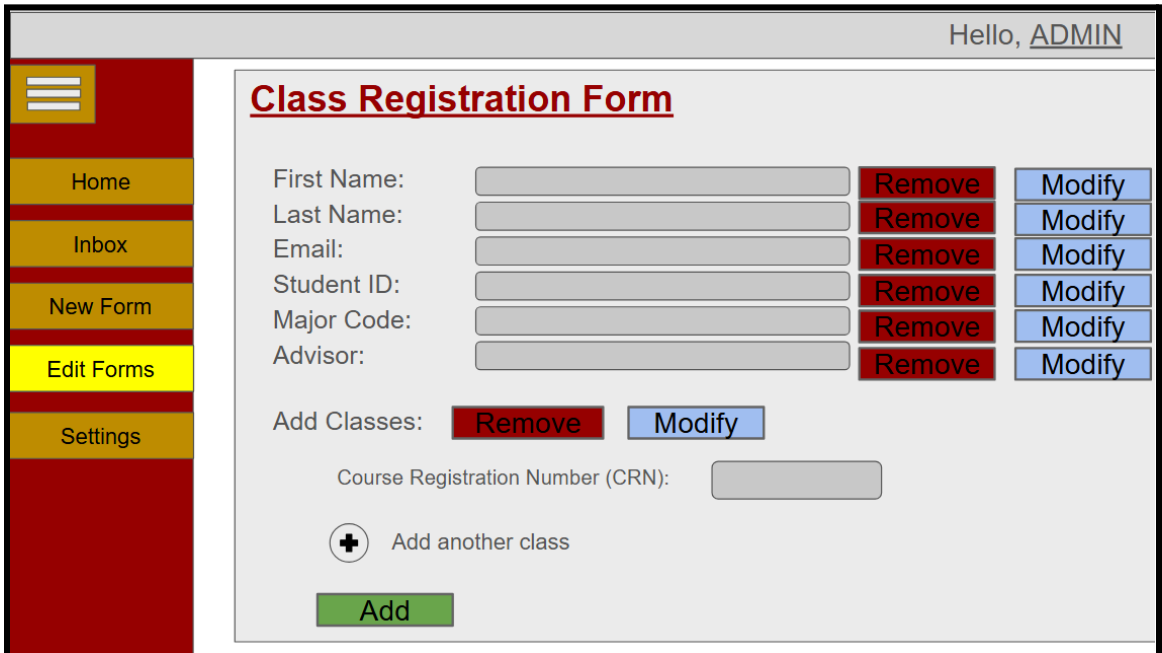
### 3.9. Edit Form Selection

A sample page for the Administrators that allows them to choose a form to edit. Note that in the hamburger menu, Administrators have an extra tab for the form editor since the Student and Staff Advisor users can not edit form layouts.



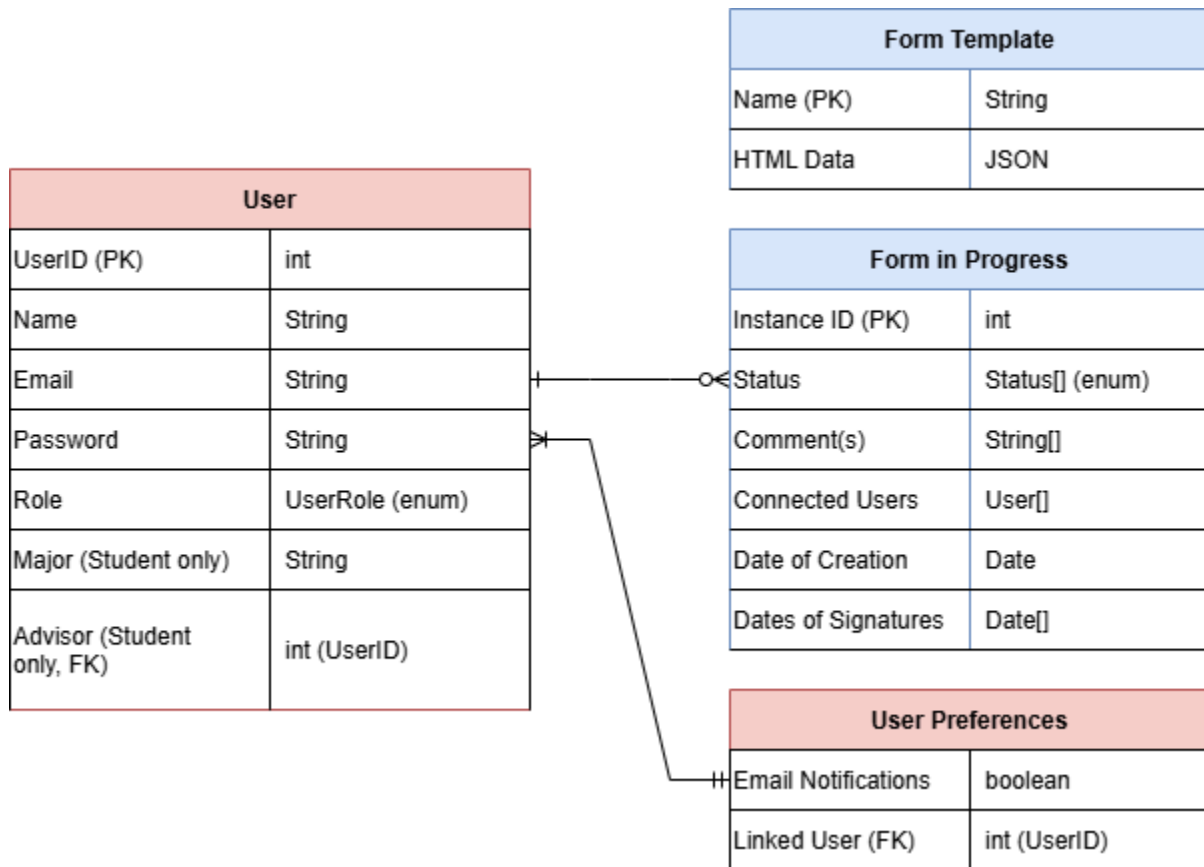
### 3.10. Form Layout Editor

A sample GUI of what the Form Layout Editor may look like, where Administrators can add, remove, or modify inputs into a form.



## 4. Database Design

### 4.1. Entity Relationship Diagram



### 4.2. User Collection

#### 4.2.1. User

- UserID (Primary Key): A unique identifier
- Name: First and last name of the user
- Email: FIT email of the user
- Password: Password of the user
- Role: Role deciding the user's level of access (student, staff advisor, admin)
- Major: Major only stored for students
- Advisor: Advisor only stored for students

#### 4.2.2. User Description

This collection stores all the data related to a user. The email and password are used for logging in, the ID is used for distinguishing between users, the name is used for auto-filling forms, and the role decides what the user can

access on the website. Students have extra data stored for their major and advisor.

### **4.3. User Preferences Collection**

#### **4.3.1. User Preferences**

- Email Notifications: Decides whether the user receives email notifications on top of the inbox notifications
- UserID: References back to the user

#### **4.3.2. User Preferences Description**

This collection stores the user's preferences regarding the website. Currently this only stores their preference regarding email notifications, but more may be added in the future (dark mode, etc.).

### **4.4. Form Template Collection**

#### **4.4.1. Form Template**

- Name: The unique name of the form (Used as identifier since the names are unique)
- HTML Data: All the information on the form, including the text, its formatting, the required fields and their names and attributes etc.

#### **4.4.2. Form Template Description**

The form template is the empty form that is to be parsed and displayed for the user to see, fill out, and/or edit. Most of the data is stored as a JSON file that will make parsing each part of the form convenient, as the HTML code could be directly stored as objects and parsed back when called easily.

### **4.5. Form in Progress Collection**

#### **4.5.1. Form in Progress**

- Instance ID: ID of this specific form, which will be prefaced with the ID of the associated student
- Status: List of statuses for each party associated, which will be used to display the overall status of the form
- Comment(s): Comments left by anyone who has approved or rejected the form
- Connected Users: List of the users connected to the form
- Date of Creation: The date the form was created
- Dates of Signatures: The date of approval/rejection for progress tracking

#### **4.5.2. Form in Progress Description**

The forms in progress are the forms that have been submitted by a student and are currently either waiting for approval, have been approved, or have been rejected. These forms store the necessary status, user, date, and comment information to allow the display of the status tracking. The ID is

used to identify each from, and it is dependent on the student ID of the associated student.