Valliant Certification Tracker Design

JIA 2112

Andrey Voytko, Rushi Shah, Zachery Davis, Saimanas Enduri, Akhil Potturi, Cole Shea

Client: Valliant Public School District

https://github.com/RS257/JIA-2112-Project

Table of Contents

Commented [WPSG1]: Why is the title at the bottom?

Table of Figures

Commented [WPSG2]: Table of Figures needs a title and should start on it's own page

Terminology

API: Application Programming Interface (Contract of service between two applications; standard that describes how to build or use such a connection or interface)

Backend: Parts of application that are fundamental to application operation but are not accessible by user

Bootstrap: Free front-end framework that includes HTML and CSS based design templates (forms, buttons, tables, navigation, etc.)

Certification: In our project, this refers to an official document that satisfies a requirement laid out by the school district (VPSD)

CSS (Cascading Style Sheets): Rule based language that specifies groups of styles that should be applied to elements or groups of elements on web page

Django Database: Used to provide structure to application and store users' information (Supports MySQL)

Frontend: Focuses on coding and creation of elements and features of website seen by user

HTML (Hypertext Markup Language): Markup language for the web that defines structure of web pages

Key-Value Store: Database that stores data as collection of key-value pairs where keys are the unique identifiers for the values

Local File Storage: Storing data (images) locally on user computer

Server: Piece of hardware or software that provides a service to another computer program and its user (client)

User Interface: The point of human-computer interaction in an application

Valid: A certificate being valid refers to it still being good to use. This means the certificate hasn't expired yet.

Commented [LP3]: Maybe consider adding certification here to it is easier to understand what a certification entails, whether it be some kind of educational skill proof or some proof of doing some kind of training

Introduction

Background

Until now, the Valliant Public School district has been using a paper ledger to keep track of a whole district's employees' certifications. It proved too tedious to track down certifications of certain employees, and it has been very difficult to track everyone's' progress and expiration dates without making a single mistake for our client. That's where we come in with our project. Our project is to design a certification tracker website application (based on HTML bootstrap with Django Database) for the Valliant Public School district. There are two types of users on this application: Admin and Faculty. The admin can add/remove certifications and roles for faculty. When the faculty accesses our application, they will be greeted with a dashboard that will contain all the certifications to be completed and organized by the due date. The backend for this application is based on a key-value store database that associates each faculty member to their role, a list of their certifications, and pdf files that correspond to the certifications. The front end is themed after the VPSD colors and provides the users a way to interact with the application to complete their certifications.

Commented [PS4]: this is the intro for the static system architecture but we need an intro page that has like all the different sections and blurbs for them

Commented [ES5R4]: ah ok

Commented [LP6]: Your intro is very strong, the only thing I really see is try to keep key-value and key value consistent throughout (probably best to match it with how the key terms are done).

Commented [WPSG7R6]: 👍

Commented [WPSG8]: Why you don't need to sell me your project give me a sentence or two about why you are working on this project. What are you trying to accomplish?

Commented [WPSG9R8]: What kind of application are you developing?

Commented [CA10]: The title has 2 'L's in Valliant, but only one L here. I'm not sure which one is correct

Commented [WPSG11R10]: 👍

Commented [WPSG12]: Consider: "completed and organized by…"

Document Summary

This report documents the Valliant Certification tracker in four main parts.

The *System Architecture* section describes the static and dynamic elements of our system's structural components and the interactions within the system (demonstrated in figures 1 and 2). We will be using a Django backend to hold our key-value storage of users and their certification information. This combined with our HTML/CSS frontend interface will provide the structure to our application.

The *Data Storage Design* section describes the database structure of our system, which consists of the MySQL relational database in the Django system which stores the user profiles along with the corresponding certification documents. In the section, we include an entity relationship diagram for the Django relational database with the three major entities of the system (admin, faculty, and certifications), a description on the file format and storage, as well as an understanding of the data exchange between the database and the backend and frontend integrations.

The *Component Design Detail* section provides details of the components of our certification tracker application. There is the web view and data portion of the component design. This section provides the static and dynamic detailed views of our system, including the intercomponent relationships and how they interact with the database.

The *UI Design* section showcases the major screens that the user will interact with in the application. The UI is shown from two perspectives – the faculty and admin.

Commented [CA13]: For readability, consider bumping this section to a new page so you don't introduce a new section and immediately have a page break

Commented [WPSG14]: What diagrams?

Commented [WPSG15]: Hum… This doesn't sound right for what you were suppose to present in this section. You should have static and dynamic detailed views of your system. The UI is addressed in its own section.

System Architecture

Introduction

In order to understand the structure of our application we have provided designs of architecture of our system in Figures 1 and 2. These diagrams model a high-level description of the structural components of the application and relations between them. The application is defined by three core functional components: the Key-Value store, where the user information/certifications are stored, the third party file hosting where all certifications are downloaded, and the app that the user interfaces with to initiate the app functionality.

Static System Architecture

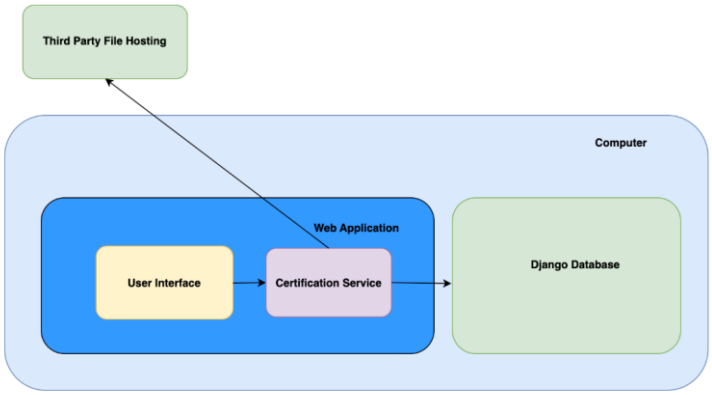


Figure 1 Static System Architecture

The static architectural design shown here provides a visualization of the structure of the system. In this diagram we show how the components are interrelated and their relation to one another. There are two major components that our application needs to account for: web application and data storage. Our web application is comprised of a user interface that depends on a certification service. This certification service (Python API) depends on the data (Django Database and Third-Party File Hosting) to provide the interface with the necessary info. This is the general idea behind our whole application.

As an example, which we will later see in the dynamic section in more detail, an admin user will interact with the interface to indicate they want to remove a user. Then, a call is made on the certification service, which will then work with our Django Backend to remove this user.

To cover all our use cases, we need a backend database, third party file hosting, and a user interface on top of the web application. The interrelations between these components to bring our website to life are illustrated in figure 1 above.

Commented [CA16]: Remember to include figure names and descriptions! For readability, consider centering your figures on the page

Commented [WPSG17R16]: Also consider rearanging your boxes to have less "elbows" and more straight lines.

Commented [RN18]: Design: Please make sure to label each image with a short description of what it is presenting, i.e. Figure 1: Static System Architecture...

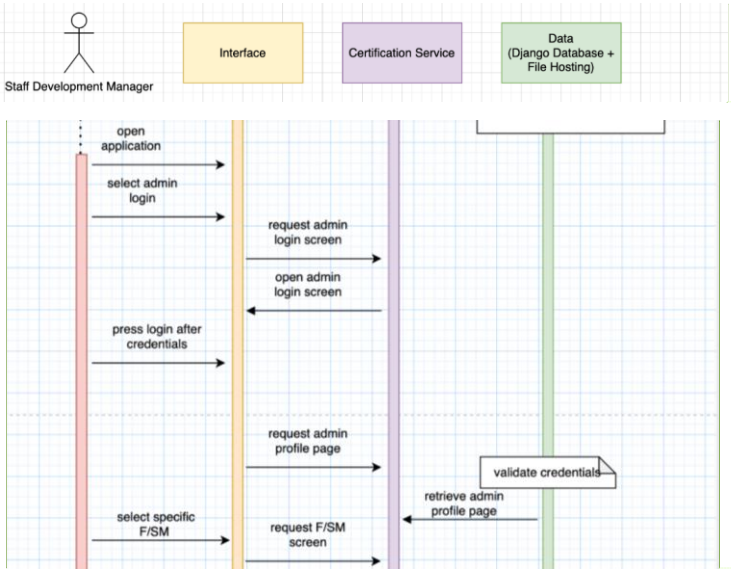
Commented [WPSG19R18]:

Commented [RN20]: Design: Center this image for better appeal to the reader.

Commented [WPSG21]: The static view of the system should show the structure and NOT the runtime functionality of the system. Think about what the arrows in your diagram are showing. They should not be showing flow, they should be showing dependency. think about which components "know about" which other components; that would be a dependency.

Commented [WPSG22]: what diagram? Include a reference once you have the captions for your figures.

Dynamic System Architecture



Commented [WPSG23]: Make sure you use the correct syntax for an SSD (i.e. use the actor figure). The boxes should be the same exact components that are shown in the static view of your system.
I suggest that you simplify your diagram and focus on the point of this view: To show the runtime (dynamic) interaction of the different components/layers of your system. It may be more beneficial to show a complete interaction between all the layers/pieces (i.e. expand horizontally) rather than more functionality (i.e. vertically). You may also want to break the diagram into 2 separate diagrams since you have different actors. You can always include the 'Ref' fragment to reference another diagram/use case.

Commented [LP24]: This may be a user specific thing, but the dynamic system architecture is hard to read, it might be worth making each screenshot a little bigger

Commented [LP25]: I think it may be worth saying something like remove cert from to be done user list or something more specific. It can be confusing because from my understanding you guys would have 2 lists, one with the done certifications and one with to be done certifications

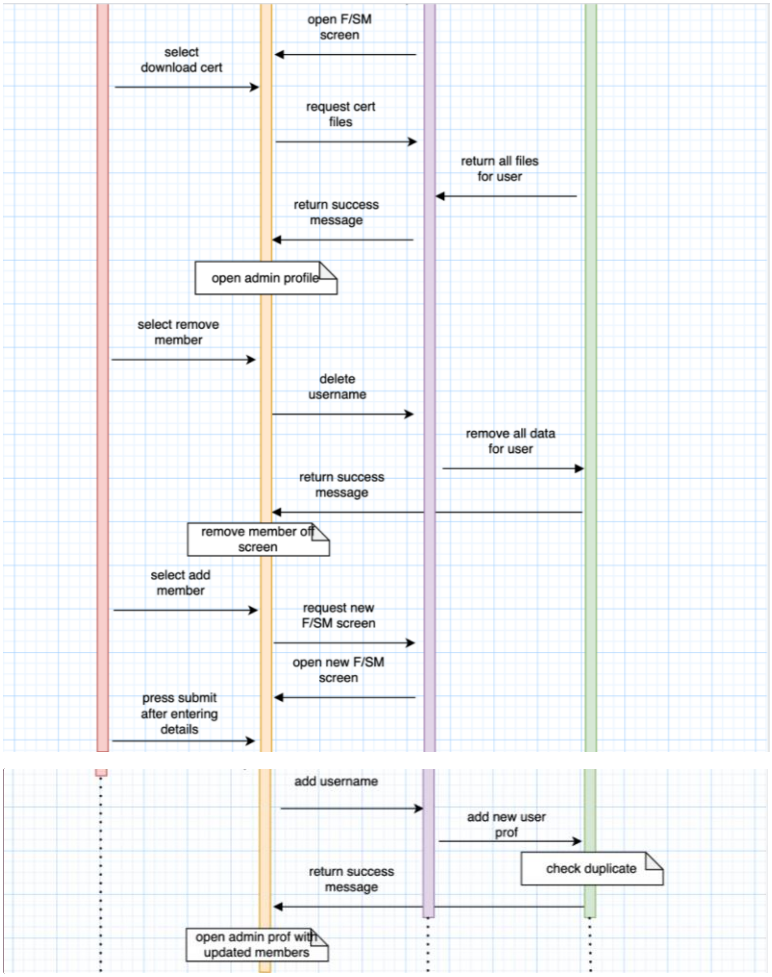


Figure 2 Dynamic System Architecture

The dynamic architectural design shown in figure 2 above conveys the control flow within the system and more specifically how the various components are interacting and passing and exchanging information. The sequence diagram above demonstrates the scenario that occurs as an admin navigates through the system and makes two actions (download certifications, remove user). The actor or user, admin, is shown to be going through the flow of downloading the certifications and removing a user from the system. These actions cover all the layers included in

Commented [RN26]: Design: Once again, make sure to include a short description to help the reader understand what the image is about.

Commented [WPSG27R26]:

Commented [RN28]: Design: Make sure to center these images for a better appeal to the reader.

Commented [WPSG29R28]:

Commented [WPSG30]: You may want to include a reference to the figure number instead of positioning information … i.e. the design shown in figure 1.x

the static system architecture, so we chose to use these scenarios. The color of each component corresponds to its associated component in the static system architecture.

Data Storage Design

Introduction

The Entity Relationship Diagram below represents the data storage design of the relational database used in the application. The diagram below conveys how the data is stored for each entity and which elements belong to each of the entities as well. There are three main entities being the user – divided into either admin or non-admin faculty and the certifications. All users have the attributes shown below and the unique identifier is their username. For faculty, there is also a role assigned based on their position. Certifications on the other hand have a unique identifier as an ID as well as a file in the PDF format and completion mark to display whether a certification is completed. The admin ultimately assigns and approves certifications and the faculty upload their certifications.

Database Use

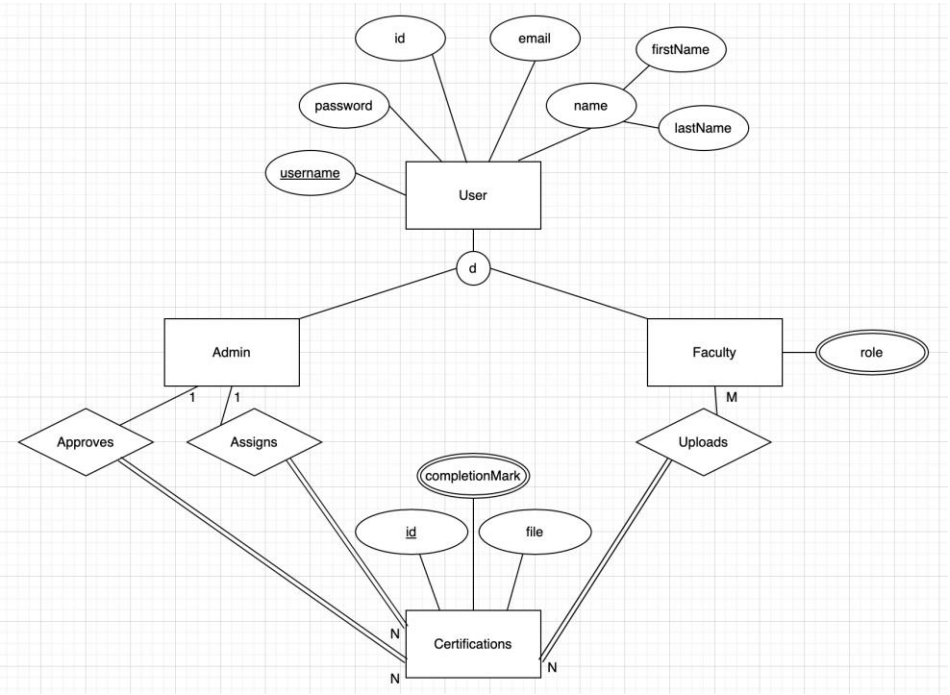


Figure 3 Database

Commented [WPSG31]: Your diagram is a bit difficult to read. Consider rearranging the entities and attributes so that it is easier to follow. I'm not very familiar with the circle notation that you are using. Is there 1 DB table for all users and the Faculty records have an additional attribute of Role?

Commented [RN32]: Design: Description and center image just like comments above.

File Use

The files are stored as pictures or as PDF in the media folder located in the third-party media files hosting. The client is free to choose what hosting should be used for the files' storage. One of the examples might be Amazon S3 or Amazon Simple Storage Service, which is a service offered by Amazon Web Services that provides object storage through a web service interface. These types of web services provide high reliability and security for their clients.

Data Exchange

Only the backend server will be able to delete/add/edit profiles created as well as any corresponding data in the Django database. The communication between the frontend and backend will happen throughout an API. The user can upload a certification from the frontend and the Django API will serve this upload by saving the certification as a file in a third-party files' storage hosting, the API will also create all related fields in the database system. Additionally, the API will serve the user's download request of the file. Data that is stored will be secure in the database such that it will be accessible by no one other than the backend server.

Commented [WPSG33]: The Data Exchange section is where you should give the user the most information. Give an example of how the communication happens between the front and back end through the API. This section should have a lot more. You also want a separate section or paragraph to discuss security concerns. How is the DB secured? Storing teachers' certification information would seem rather sensitive and you would want safeguards.

Component Design Detail

Static Component Design

Commented [RN34]: Design: Add a space here for consistency between other title - image relationships

Commented [RN35]: Design: Adjust image width for consistency and once again add short description and center it.

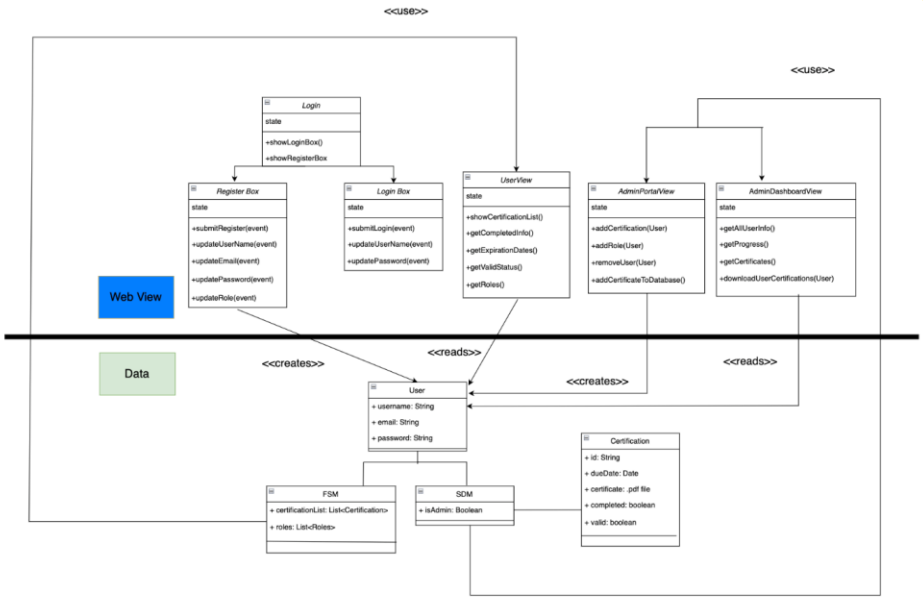


Figure 4 Static Design

The class diagram in Figure 4 represents the static detailed components of our system, in the static form. Our diagram reflects the specific classes we have implemented color coded to reflect our MVC system architecture. There are two main sections: the web application and the data as seen in the MVC system architecture. Blue represents the web application view and green represents the data (file storage, Django database). The certification service dependencies are shown using the <<creates>>, <<reads>>, and <<uses>>.

The diagram shows the relationships between the components and the functions they use to read/write to the database.

Dynamic Component Design

Commented [RN36]: Design: Same as above with description and image centering

The dynamic component design can be seen below through our robustness diagram. It shows how the users interact with the system and more specifically how each component interacts with each other and the users. A user can be a standard user or an admin. Both will be prompted to login or register, which will authenticate whether the login or registration information is valid. Upon login, the user or admin will be brought to their respective home page, and from there will have access to various other pages such as: Certification Directory Page for users and admins respectively, and a “users” Page. These pages would have functions to assign roles to users, view, edit, and download certifications, and assign certifications, all of which would interact with the Certification API and Users. The diagram shows how each page will interact with

controllers and data.

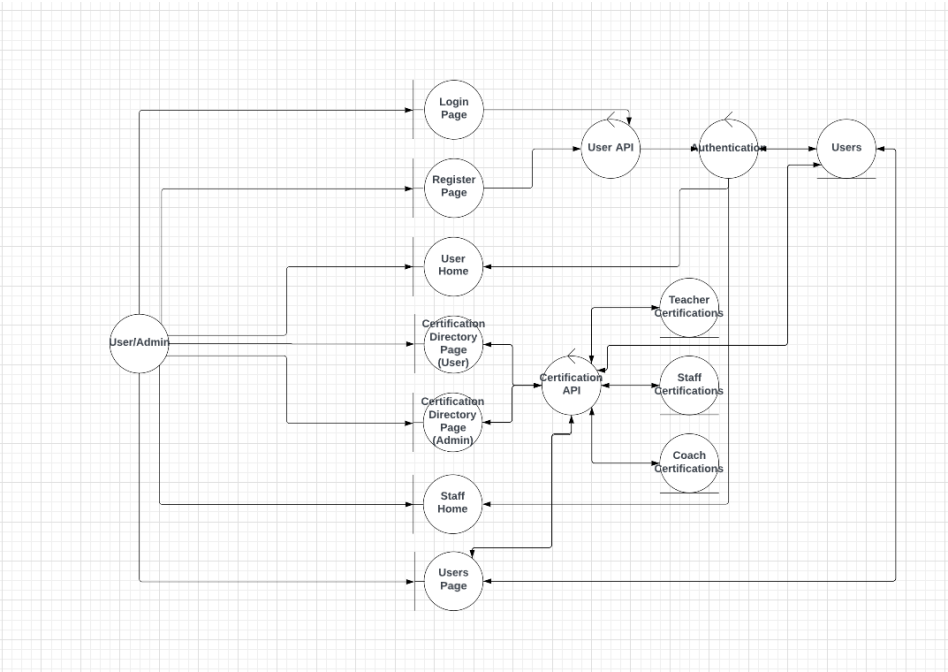


Figure 5 Dynamic Design

UI Design

In this section, we present the major views of our application. Overall, we decided to stick with the official color scheme of the Valliant Public Schools, for their main website also uses a similar color scheme.

There are two types of users an Admin (SDM) or a Faculty/ Staff Member (F/SM). Their uses are different, but they both will start off on the initial home screen as seen in Figure 6. If a user clicks on the “Login” button they will be redirected to the Login screen as shown in Figure 7. If a user clicks on “Register”, they will be redirected to the Registration screen as shown in Figure 8. If a user clicks on "Back” button of either the Login or Registration screen, they will be redirected to the Home Screen.

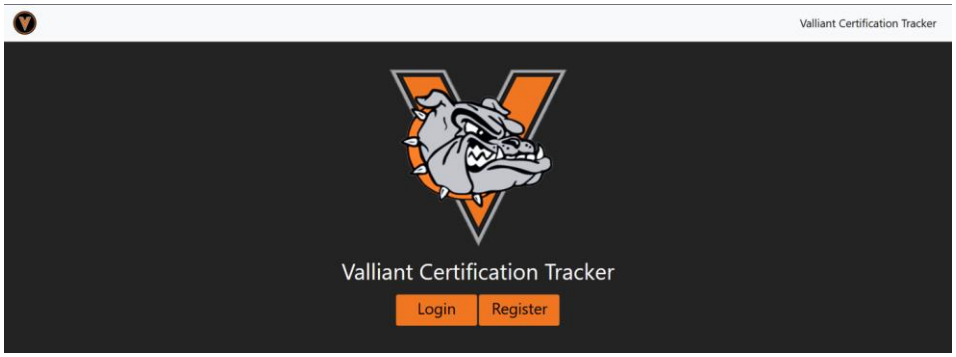


Figure 6 Initial Launch Page

Commented [HJ37]: Components: The draft includes all required components (except UI design which has not been assigned yet). Component detail design section is missing an introduction to provide context to the reader and the components of the ER diagram could be explained a little more. The introduction and system architecture sections are very strong.

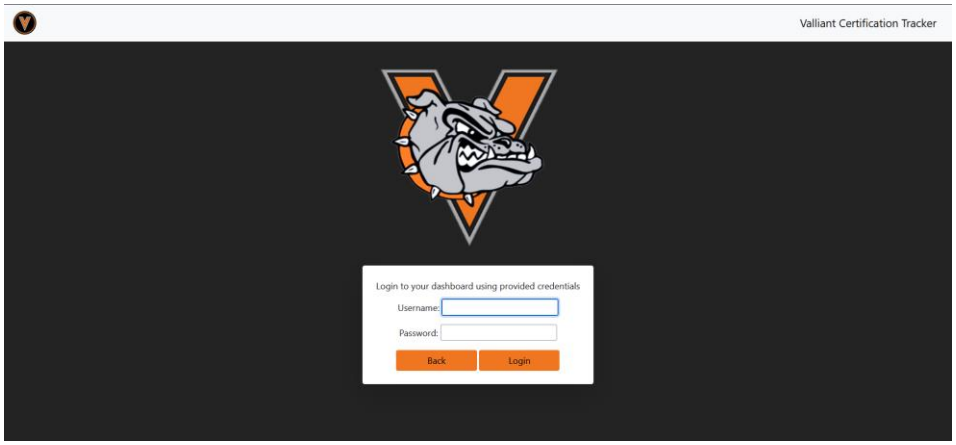


Figure 7 Login Screen

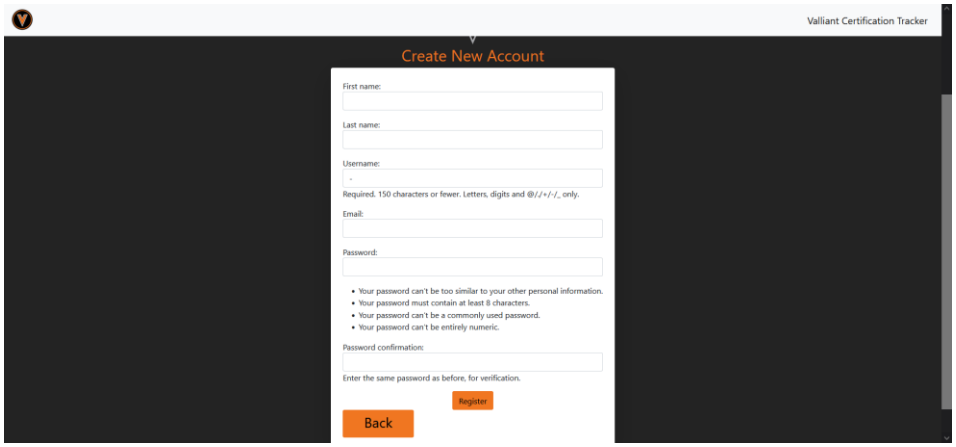


Figure 8 Register Page

When a user logs or registers into the system, their dashboard will change based on if they are a F/SM (Figure 9) or a SDM (Figure 11). Since both screens have a "Logout" Button, we will discuss them here. Clicking on it redirects to the Logout Screen (Figure 13). Clicking on the "Back" button on the Logout screen redirects to the respective Dashboard screen of the user type. Clicking on "Yes" redirects to the Login Screen (Figure 7).

In Figure 9, a F/SM can see what certifications they need to upload. Clicking on the "Upload" button will bring up a pop up in which the F/SM can upload a certificate for completion.

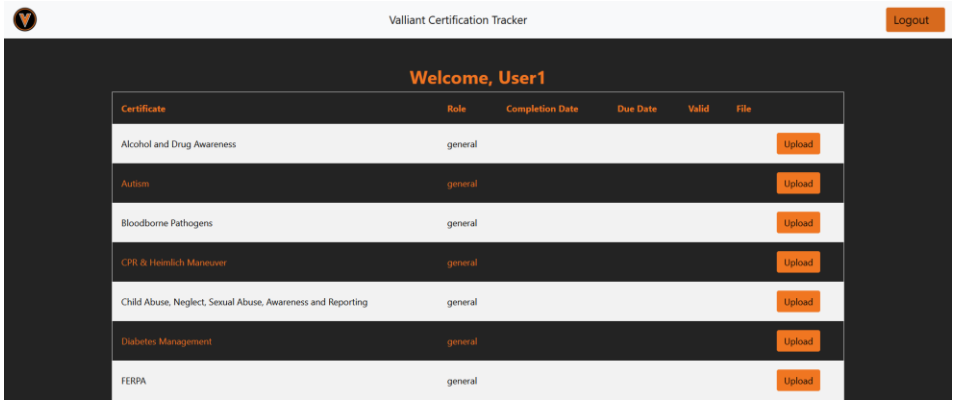


Figure 9 Dashboard View of F/SM

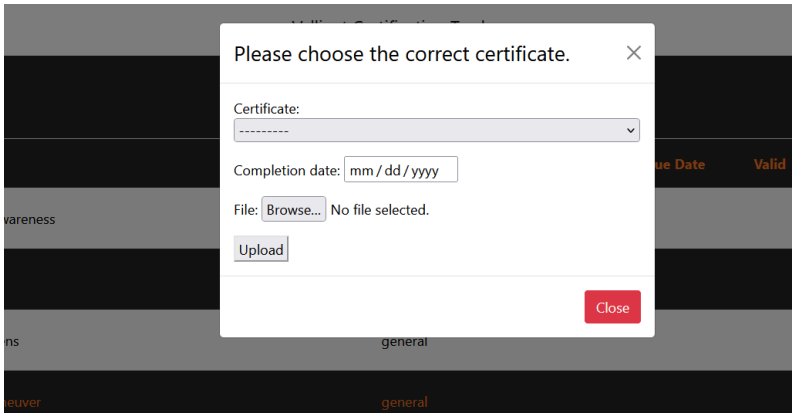


Figure 10 Upload Certification Form

In figure 11, the Admin can have a quick view of certifications that have been completed by F/SMs. This table is collapsible, so if one clicks on a name then it expands to show more information as seen in Figure 12. We decided this will be the best way to see 100+ employees while also seeing the numerous amounts of certifications each one has completed or needs to be complete.

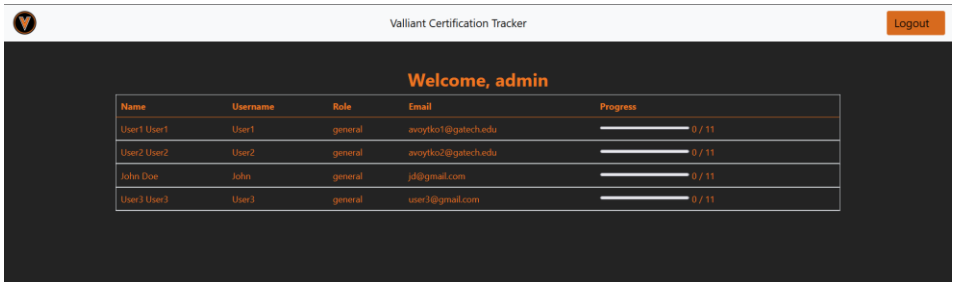


Figure 11 Dashboard Admin View

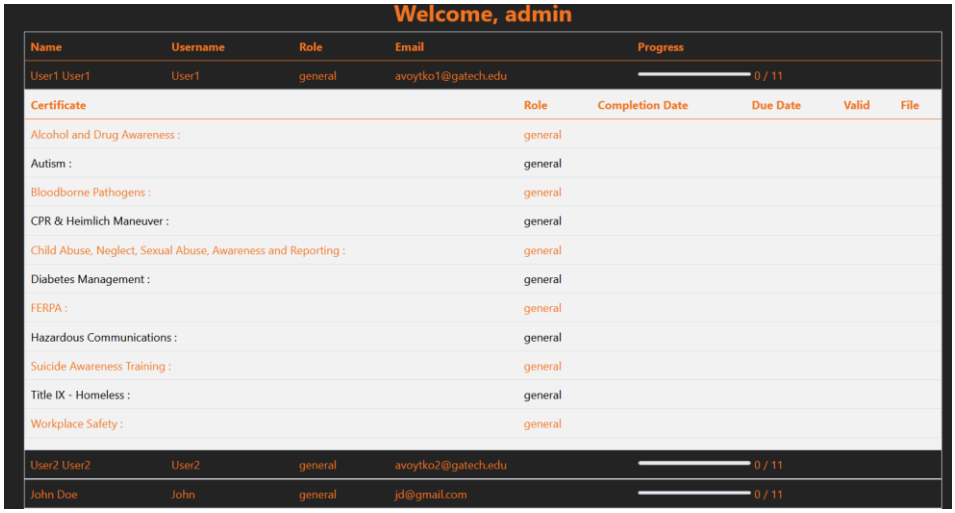


Figure 12 Collapsing Table on Admin Screen

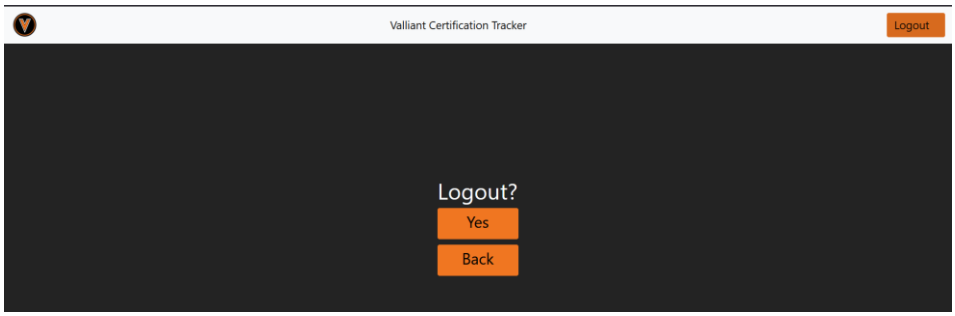


Figure 13 Logout Screen

There is also a hidden screen for admins by typing "/admin" the home screen URL. This screen below allows Admins to manipulate data however they see fit. For example, it can used to add or remover certifications from the system or assign different roles of F/SM a specific certification. We kept the color scheme of this screen separate as to help differentiate between the quick table dashboard view of Figure 11 and the bulk working sections of Figure 14 that a SDM will use.

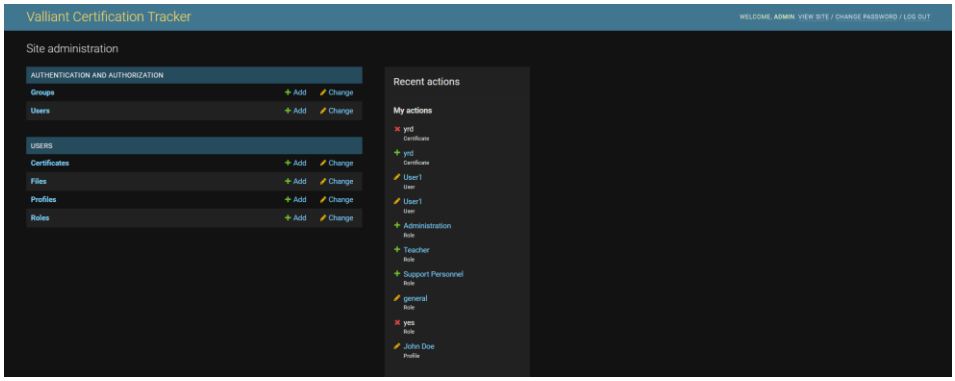


Figure 14 Admin Control Panel