Northeastern University
Department of Electrical and Computer Engineering

EECE 4520 Software Engineering

Software Design Specification: University Classifier

Authors: Felix Chan, Marissa D'Alonzo, Sumer Malhotra, Jensen Rosemund, Stockton Sheehan

March 19, 2019

1. Introduction

1. Purpose of this Document

The purpose of this document is to give a more in depth understanding of the design and architecture of the “University Classifier” software.

2. Scope of the Development Project

“University Classifier” is a project that leverages Kaggle datasets to help potential graduate students determine which college is best for them. The student can enter the university they are interested in, and the application will tell them what scores and supplementary materials they need to boost their chances of getting in. Alternatively, the student can enter information about their current profile, including GRE scores, research experience, and GPA, and the application will suggest a list of schools that they have a chance of getting in.

3. Definitions, Acronyms, and Abbreviations

Term	**Definition**
User	Someone who interacts with the application
Admin/Administrator	System administrator who is given specific permission for managing and controlling the system
Stakeholder	Any person who has interaction with the system who is not a developer
Developer	Person responsible for writing the software
University	A school with graduate admissions information recorded in the database

4. References

[1] M. S. Acharya, “Graduate Admissions,” *RSNA Pneumonia Detection Challenge | Kaggle*, 28-Dec-2018. [Online]. Available: https://www.kaggle.com/mohansacharya/graduate-admissions. [Accessed: 12-Feb-2019].

[2] M. O'Neill, “World University Rankings,” *RSNA Pneumonia Detection Challenge | Kaggle*, 27-Sep-2016. [Online]. Available: https://www.kaggle.com/mylesoneill/world-university-rankings. [Accessed: 12-Feb-2019].

5. Major software requirements

The minimum specs you’ll need:

- Windows 7/8/10 64-bit + Mac OSX Sierra
- Core i3 2.4 GHz processor
- 4GB of system RAM
- Intel HD 4000 video card

6. Design constraints, limitations

For this software, the constraints and limitations are fairly lenient as it is not a particularly processing intensive software. As the server/database handles all the information, the user essentially only needs a standard computer with an internet connection.

1.6.1 Design Constraints

- This product can only be used on a PC. However, it can be used on a Mac with 3rd party software called MonoDevelop.
- The computer must have access to the web in order to pull data from the server.
- The app will not hold any user data on disc.
- All user inputs will be acknowledged within 3 seconds.
- A system crash should not result in data loss.

1.6.2 Limitations

The server has a storage limitation based on the hardware that is used to host it, and similarly may only be able to handle a certain amount of traffic at a given moment. Almost all of the limitations for the software will be based on the hardware of the server.

1.6.3 Design Goals

For our software, it should be, above all else, reliable and secure. Although ideally, the software to be the best, being as fast as possible, efficient as possible, while requiring little development time, it is just not feasible. Therefore, our priority for the software will be reliability and security for the database. As the software handles user information, we must keep the faith and loyalty of our clientele above all other goals. As for things such as reusability, the software is not expected to need to change a lot, nor have a lot of iterations, so this is mainly disregarded.

7. Changes to requirements

N/A, there have been no changes to any requirements.

8. Overview of Document

As we have just finished the introduction of the document, we will be moving onto the following sections, Data Design, System Architecture Description, Detailed description of components, and the Interface Design. In Data Design, we take a brief graphical look at the data structures of the objects and databases for this project. In System Architecture Description, we will be taking an in depth look at the architecture of the system, a fairly self-explanatory section. Next, we will describe all the components of the project, with a summary of their purpose, function, and implementation details. Lastly, we show a few images of what we plan for the interface to look like when the project is finished.

2. Data Design

1. Data Objects and resultant data structures

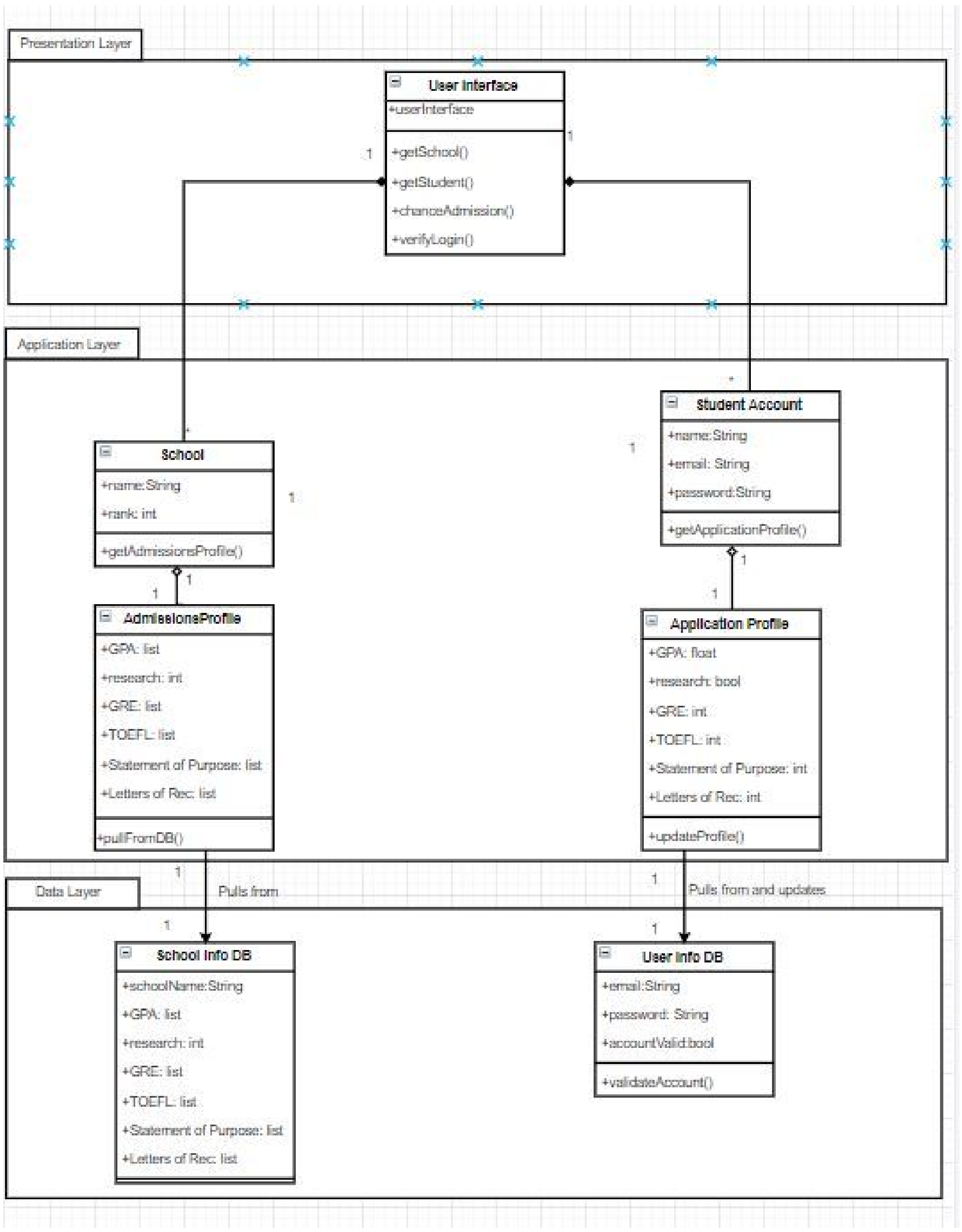


Figure 1 - The subsystem class diagram

2. File and database structures

The database has two tables with the following structures:

- School Information Database
    - School Name: String

    - Rank: Integer
    - GPA: List of integers
    - Research: Percentage
    - GRE: List of integers
    - TOEFL: List of integers
    - Statement of Purpose: list of integers
    - Letter of Recommendation: list of integers
- User Information Database
    - username: String
    - first name: String
    - last name: String
    - email address: String
    - password: String

3. System Architecture Description

1. Overview of Modules / Components

- Presentation Layer- This layer serves as the interface for the user to interact with the system.
- Application Layer- This layer processes all user input and requests, pulls elements from the data layer, and generates results using the machine learning algorithm.
- Data Layer- This layer is responsible for holding all of the data needed in order for the system to operate, which includes the School info DB and the User DB.
- School info DB- This database stores the graduate admission information of colleges.
- User DB- This database stores every user created account and all of their information.
- User interface- The user interface allows the user to directly interact with the system by requesting pages or inputting information.
- School profile- The school profile stores the graduate admission information for an individual school.
- Admissions profile- The admission profile allows a user to input their academic information.
- Student account- The student account stores the account information for an individual user, which includes the username, password, and email.
- Application profile- The admission profile stores an individual user's inputted academic information.

- Datafiles: List of Universities- The .csv file which includes graduate information of 500 schools; this data is compared with the user’s academic information.

Structure and relationships

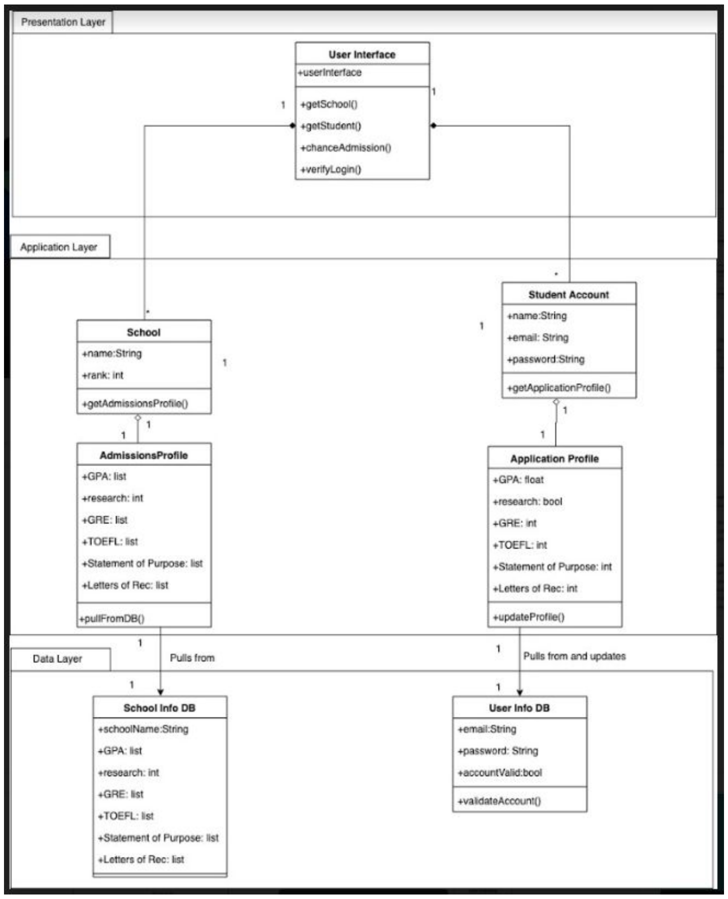


Figure 1 - The subsystem model of our system

The three layer architectural style is a good structure for our system since the class diagram can be decomposed into a presentation layer, application layer, and a data layer.

For this architectural style, the user only can interact with the system through the presentation layer. Therefore, the model’s presentation layer is a single class: the user interface, since this is the only class that the user can directly interact with. This layer interacts with the application layer by providing user input and requesting data and results to be displayed visually.

Unlike the presentation layer, the model’s application layer is a subsystem which includes two classes (and each one’s aggregate components): school and student account. Both of these classes perform functions in the system not directly involved with the user interface. The classes belong in this layer since they pull elements from the data layer, pull requests from the presentation layer, process the data and requests, and pass the final results to the presentation layer. Our structure does not include partitions since no two classes within the same layer need to access each other.

The data layer contains the static databases which contain information that the application layer needs to use. The classes in this layer perform no functions, so therefore do not need to know anything about the layers above them. The only purpose of this layer is to be accessed by the application layer for extracting elements.

4. Detailed description of components

1. Component template description

A description of the template we will be using in sections 4.2-4.12 can be found below.

- **Identification**

  The unique name for the component and the location of the component in the system. (You may want to include a reference to which part of the architecture diagram this component belongs.)
- **Type**

  A module, a subprogram, a data file, a control procedure, a class, etc.
- **Purpose**

  Function and performance requirements implemented by the design component, including derived requirements. Derived requirements are not explicitly stated in the SRS, but are implied or adjunct to formally stated SDS requirements. This is where you explain the connection of this component to some requirement(s) as specified in the SRS.

- **Function**

  What the component does, the transformation process, the specific inputs that are processed, the algorithms that are used, the outputs that are produced, where the data items are stored, and which data items are modified.

- **Subordinates/ Modules used**

  The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part.

- **Dependencies**

  How the component's function and performance relate to other components. How this component is used by other components. Include the other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components.

- **Resources**

  A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services.

- **Processing**

  The full description of the functions presented in the Function subsection. Pseudo-code may be used to document algorithms, equations, and logic.

- **Data**

  For the data internal to the component, describes the representation method, initial values, use, semantics, and format. This information will probably be recorded in the data dictionary.

2. Presentation Layer

- **Identification:** Presentation Layer. Located in the subsystem architecture as the top layer.
- **Type**: Module
- **Purpose:** The purpose of this layer is to act as the client interface to the system and communicate with the application layer. This is the only part of the system the client can interact with, and is required to fulfill all the functional requirements specified in the SRS.

- **Function:** This layer received all user input and forwards it to the application layer, which performs the necessary computations and returns a result, which this layer displays to the user. Examples of this could include the processes of logging in or searching for a particular school. It does not store any information, but the student account and admissions profile can be modified through this layer.
- **Subordinates/ Modules used**: This module includes only the User Interface class. This module is required to fulfill all functional requirements because all the functional requirements specified in the SRS require user input.
- **Dependencies:** This module can only communicate with the application layer. This module allows user to access the results of the functional requirements:
    - create a user account
    - log into a user account
    - search for a school
    - input academic information
    - request a list of schools with percent chance of admission
    - cancel any action

  with all input acknowledged within 3 seconds as stated in the nonfunctional requirements of the SRS.
- **Resources:** A computer with a functioning screen, mouse, and keyboard is required for the user to see and interact with this layer. An Internet connection is required to for the user to connect to the application and for the application to connect to the remote database.
- **Processing:** The presentation layer functions in the following manner:
    - Receive input from the user in the form of mouse clicks or text input.
    - Forward input to application layer.
    - Wait for application layer to send back information, then present to the user.
- **Data:** There is no internal data required for this layer.

3. Application Layer

- **Identification:** Application Layer. Located in the subsystem architecture as the middle layer.
- **Type:** Module
- **Purpose:** The purpose of this layer is to perform all the logical functions and algorithms of the software. This includes functional requirements:
    - create a user account
    - log into a user account
    - search for a school

    - input academic information
    - request a list of schools with percent chance of admission given admissions profile.
- **Function:** This layer takes user input from the presentation layer and performs the necessary logic or algorithms to produce the desired output. This can include CRUD operations to the user database, searching the school database, and running the Tensorflow algorithm to calculate the list of schools with percent chance of admission given the user's academic profile.
- **Subordinates/ Modules used:**
    - School Class: Stores the given information about a school
    - Admissions Profile Class: Stores the application information about the students previously admitted to the school, used for the Tensorflow algorithm
    - Student Account Class: allows user to log into account and stores login information
    - Application Profile Class: contains the user's application information used to determine which schools they can get into
- **Dependencies:**
    - Presentation layer: inputs user information to determine which operation to perform. After performing the desired operation, the application layer then returns the result to display to the user.
    - Data layer: pulls data from the databases to find a school, log into/create an account, and run the Tensorflow algorithm
- **Resources:** To execute the functional requirements that involve the databases, such as logging into an account or searching for a school, an Internet connection is required to access the remote database. To run the Tensorflow algorithm, significant CPU execution time for computation is needed.
- **Processing**
    - Create an account: Receive user input. Connect to the database. Check that there is not already an account with the same information. If there is not, create the account and store in database. If there is, prompt user to try again.
    - Log into account: Receive user input. Connect to the database. Verify the login information is correct. If it is not, prompt user to try again. If it is, bring user to profile screen.
    - Search for school: Receive user input. Connect to database. Search school database for input with same name. Return the information matching that name or an error saying that school does not exist.

    - Update profile: Receive user input. Connect to database. Update appropriate rows.
    - Algorithm to find list of schools with percent chance of admission: Uses machine learning to match user's application profile with the schools with the most similar admissions profile.
- **Data**
    - School Class: This class cannot be modified by the user. All creations of this class will be done before releasing the software.
    - Admissions Profile Class: This class cannot be modified by the user. All creations of this class will be done before releasing the software.
    - Student Account Class: This class can be created and updated by the user. The default values for everything is null.
    - Application Profile Class: This class can be created and updated by the user. The default values for everything is null.

4. Data Layer

- **Identification:** Data Layer. This layer is the bottom layer of the subsystem architecture.
- **Type**: Module
- **Purpose:** This layer is responsible for storing all the persistent data needed to fulfill the functional requirements, including the information about the universities and their admissions data and the student accounts.
- **Function:** This layer simply stores information. When the application layer queries the data layer, the data layer will return the desired information or an error saying it was not found. Information can be updated through commands from the application layer. The data is stored in a AWS remote database with one schema and several tables. There is a table for user accounts, one for school information.
- **Subordinates/ Modules used:**
    - School Information Database: Stores information about the schools, including name, rank, and ranges of scores for admitted students
    - User Information Database: Stores information about the user, including username, password, and application information.
- **Dependencies:** The application layer uses this layer to access the necessary information needed to carry out the functional requirements, such as creating an account or searching for a

school. The application layer is responsible for creating, modifying, and deleting instances in the data layer. The data layer responds to the application layer in under 3 seconds.

- **Resources**: An Amazon Web Services remote database is used to host this layer. MySQL is used to manually view the database, while the C# application layer accesses the database for the system.
- **Processing**: The application layer controls all functions of the data layer.
- **Data:**
  - School Information Database: The software engineers will populate this database before the software is released. The user cannot modify instances of this database.
  - User Information Database: The users will create and update the information stored in this database.

5. School Information Database

- **Identification:** School Information Database. This is a class located in the data layer.
- **Type:** Class
- **Purpose**: The purpose of this class is to store all the instances of the school class for easy access.
- **Function:** The function of this class is to store all the instances of the school class. This information is used to determine the percent chance of admission to all schools and return all information about a given school when it is searched for. The information can be modified through the application layer.
- **Subordinates/ Modules used:** This class helps to satisfy the functional requirements of:
  - The user must be able to input their information and be given a list of schools with their chance of admission as a percentage.
  - The user must be able to search for a specific school and be given a list of the scores needed to maximize their chance of admission.

  This class consists of the following attributes:
  - School Name: String
  - Rank: Integer
  - GPA: List of integers
  - Research: Percentage
  - GRE: List of integers
  - TOEFL: List of integers
  - Statement of Purpose: list of integers
  - Letter of Recommendation: list of integers

- **Dependencies**: This class composed of instances the admissions profile and school classes. It is used by the Tensorflow algorithm and the search function to return information about schools. Entries in this class are created and updated by the software engineers and cannot be modified within the application.
- **Resources:** This class is stored on an Amazon Web Services remote database. An Internet connection is required for the application to access the information. It should return information to the application in under 3 seconds.
- **Processing:** This class stores information on a remote database. Classes in the application layer can query it, and it will return or update the desired information. All data is preprocessed before being entered into the database, so no algorithms are needed.

- **Data:** This class stores information about schools with the following attributes:
    - School Name: String
    - Rank: Integer
    - GPA: List of integers
    - Research: Percentage
    - GRE: List of integers
    - TOEFL: List of integers
    - Statement of Purpose: list of integers
    - Letter of Recommendation: list of integer

6. User Database

- **Identification:** User Information Database. This is a class located in the data layer.
- **Type:** Class
- **Purpose**: The purpose of this class is to store all user information.
- **Function:** This class can store the attributes of each user. This includes first and last name, username, email address and password.
- **Subordinates/ Modules used:** This class helps to satisfy the functional requirements of:
    - The user must be able to create an account.
    - The user must be able to log into their account after its creation.

  This class consists of the following attributes:
    - username: String
    - first name: String
    - last name: String
    - email address: String
    - password: String

- **Dependencies**: New entries can be made by application when a new account is made. Otherwise, the application can only pull data for account verification purposes.
- **Resources:** This class is stored on an Amazon Web Services remote database. An Internet connection is required for the application to access the information. It should return information to the application in under 3 seconds.
- **Processing:** This class stores information on a remote database. Classes in the application layer can query it. All data is preprocessed before being entered into the database, so no algorithms are needed.
- **Data:** This class stores information about schools with the following attributes:
    - username: String
    - first name: String
    - last name: String
    - email address: String
    - password: String

7. User Interface

- **Identification:** User Interface. This is what the user interacts with when using the application.
- **Type:** UI
- **Purpose**: The purpose of the UI is to give the user a visual method of controlling the application.
- **Function:** The UI consists of multiple pages to accomplish the goals of our application as seen in section 5.
- **Subordinates/ Modules used:** This class helps to satisfy all of the functional requirements.
- **Dependencies**: All aspects depend on the user interface since it serves as the bridge between the user and the application itself.
- **Resources:** This application is an executable file with a file extension type .exe. It needs to be run on a Windows machine.
- **Processing:** N/A
- **Data:** N/A

8. School Profile

- **Identification:** School Profile. This is a class in the application layer.
- **Type:** Class
- **Purpose**: The purpose of this class is to store information related to an individual school.

- **Function:** This class serves as the bridge between the application and the database for the school information. This class stores the basic attributes and is used when getting the full admissions profile.
- **Subordinates/ Modules used:** This class helps to satisfy the functional requirements of:
    - The user must be able to search for a specific school and be given a list of the scores needed to maximize their chance of admission.

  This class consists of the following attributes:
    - name: String
    - rank: int
- **Dependencies**: This class depends on the user interface to know which school is being searched for.
- **Resources:** This class will query information from the school database. An Internet connection is required for the application to access the information. It should return information to the application in under 3 seconds.
- **Processing:** This class in the application layer can query data. All data is preprocessed before being entered into the database, so no algorithms are needed.
- **Data:** This class stores information about schools with the following attributes:
    - name: String
    - rank: int

9. Admissions Profile

- **Identification:** Admissions Profile. This is a class in the application layer.
- **Type**: Class
- **Purpose:** The purpose of this class is to store the admissions profile for a specific student. It is linked to their account and helps to fulfill the functional requirement:
    - The user must be able to input their information and be given a list of schools with their chance of admission as a percentage.
- **Function:** This class receives information about the user's admissions profile from the user interface class/ presentation layer. It creates an instance of the class with the specified information and stores it in the user database. This class is also used as the parameters of the Tensorflow algorithm.
- **Subordinates/ Modules used:** This class fulfills the functional requirement: The user must be able to input their information and

be given a list of schools with their chance of admission as a percentage. The class consists of the following attributes:
  - GPA
  - Research
  - GRE
  - TOEFL
  - Statement of Purpose
  - Letters of Recommendation
- **Dependencies:** This class depends on the user interface class to fill out the attributes of an instance of the class. The Tensorflow algorithm depends on this class to provide the necessary parameters. Instances of this class are stored in the user information database.
- **Resources:** This class requires memory in the remote database to store instances.
- **Processing:** N/A
- **Data**: The attributes of this class are:
  - GPA: int
  - Research: bool
  - GRE: int
  - TOEFL: int
  - Statement of Purpose: int
  - Letters of Recommendation:int

  The default values for all these attributes are null. Any value can be null, but it will not be included in the Tensorflow algorithm

10. Student Account

- **Identification:** Student Account. This is a class in the application layer.
- **Type:** Class
- **Purpose**: The purpose of this class is to store information related to an individual user.
- **Function:** This class serves as the bridge between the application and the database for user information. This class stores the basic attributes and is used when getting the user's full application profile.
- **Subordinates/ Modules used:** This class helps to satisfy the functional requirements of:
  - The user must be able to log into their account after its creation.
  - The user must be able to update their profile.

    - The user must be able to input their information and be given a list of schools with their chance of admission as a percentage.

  This class consists of the following attributes:
    - username: String
    - email: String
    - password: String
- **Dependencies**: This class depends on the user interface as the user must use their credentials for the attributes to be known.
- **Resources:** This class will query information from the user database. An Internet connection is required for the application to access the information. It should return information to the application in under 3 seconds.
- **Processing:** This class in the application layer can query data. All data is preprocessed before being entered into the database, so no algorithms are needed.
- **Data:** This class stores information about users with the following attributes:
    - username: String
    - email: String
    - password: String

11. Application profile

- **Identification:** Application Profile. This is a class in the application layer.
- **Type:** Class
- **Purpose**: The purpose of this class is to store the user's application information.
- **Function:** This class is used to determine the user's list of potential programs. The data stored in the application profile is fed into the algorithm.
- **Subordinates/ Modules used:** This class helps to satisfy the functional requirements of:
    - The user must be able to update their profile.
    - The user must be able to input their information and be given a list of schools with their chance of admission as a percentage.

  This class consists of the following attributes:
    - GPA: float
    - research: boolean
    - GRE: int
    - TOEFL: int
    - Statement of Purpose: int
    - Letters of Rec: int

- **Dependencies**: This class will only be accessed if a valid user is logged in. When a user populates their profile or performs a search, the data from this class is accessed.
- **Resources:** This class will query information from the user database. An Internet connection is required for the application to access the information. It should return information to the application in under 3 seconds.
- **Processing:** This class in the application layer can query data. All data is preprocessed before being entered into the database, so no algorithms are needed.
- **Data:** This class stores information about users with the following attributes:
    - GPA: float
    - research: boolean
    - GRE: int
    - TOEFL: int
    - Statement of Purpose: int
    - Letters of Rec: int

12.Datafile: List of universities

- **Identification:** University Data. This is a .csv file of data found on the internet.
- **Type:** Dataset
- **Purpose**: To serve as a reference for determining user eligibility.
- **Function:** The data is compared to the user's score to determine how much of a chance the user has to make it into the program.
- **Subordinates/ Modules used:** This class helps to satisfy the functional requirements of:
    - The user must be able to search for a specific school and be given a list of the scores needed to maximize their chance of admission.

  This class consists of the following attributes:
    - School Name: String
    - Rank: Integer
    - GPA: List of integers
    - Research: Percentage
    - GRE: List of integers
    - TOEFL: List of integers
    - Statement of Purpose: list of integers
    - Letter of Recommendation: list of integer

- **Dependencies**: This dataset is only used to populate the school information database.

- **Resources:** N/A
- **Processing:** There wasn't any serious processing done on the data. It was automatically read into the database.
- **Data:** This class stores information about users with the following attributes:
    - School Name: String
    - Rank: Integer
    - GPA: List of integers
    - Research: Percentage
    - GRE: List of integers
    - TOEFL: List of integers
    - Statement of Purpose: list of integers
    - Letter of Recommendation: list of integer

5. Interface Design

There has been little change to the outline of the application since the SRS. The following figures depict a BALSAMIQ rendering of each page.

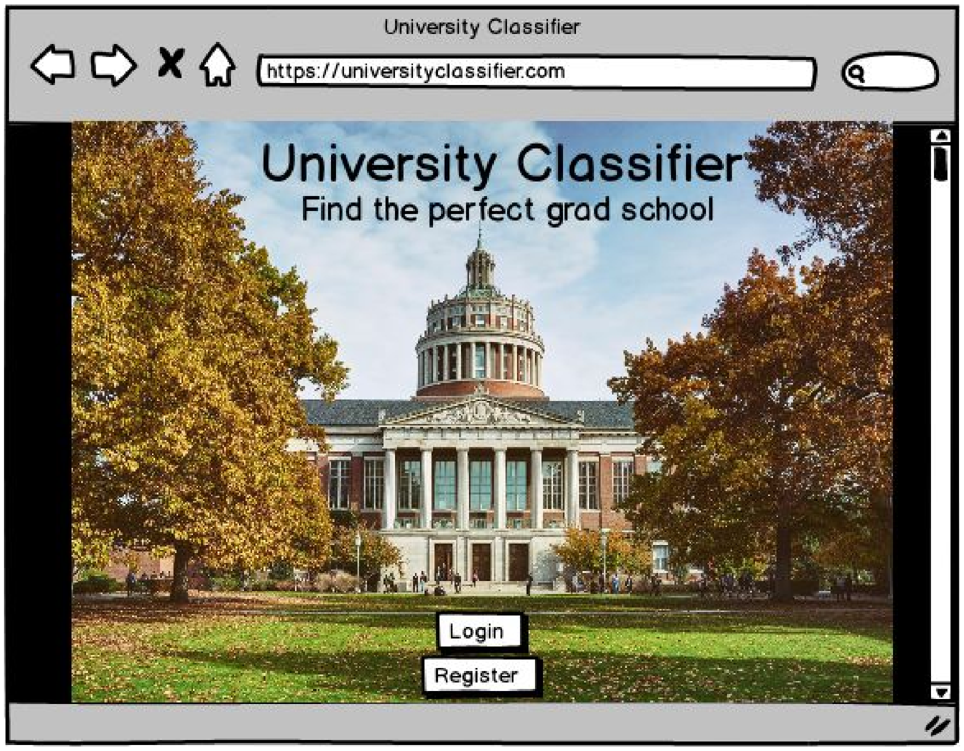

University Classifier
https://universityclassifier.com
University Classifier
Find the perfect grad school
Login
Register

Figure 2 - The home screen of the application

Figure 2 depicts the home screen of the application. From here, the user can only choose to login or register.

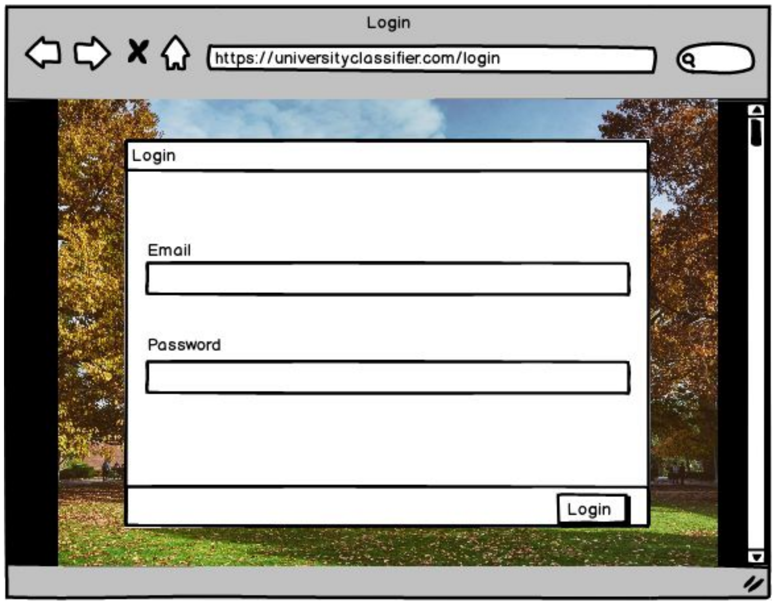


Figure 3 - The Login page

Figure 3 depicts the login page. The user enters their username and password and presses “Submit” to login. If the credentials are correct, the user is brought to the profile page. If not, the error message is shown.

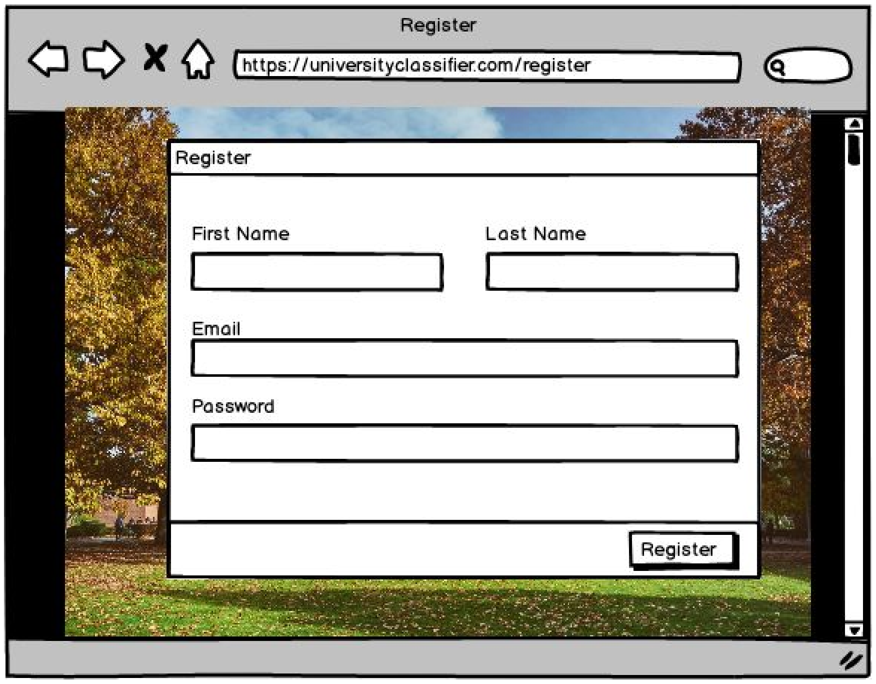
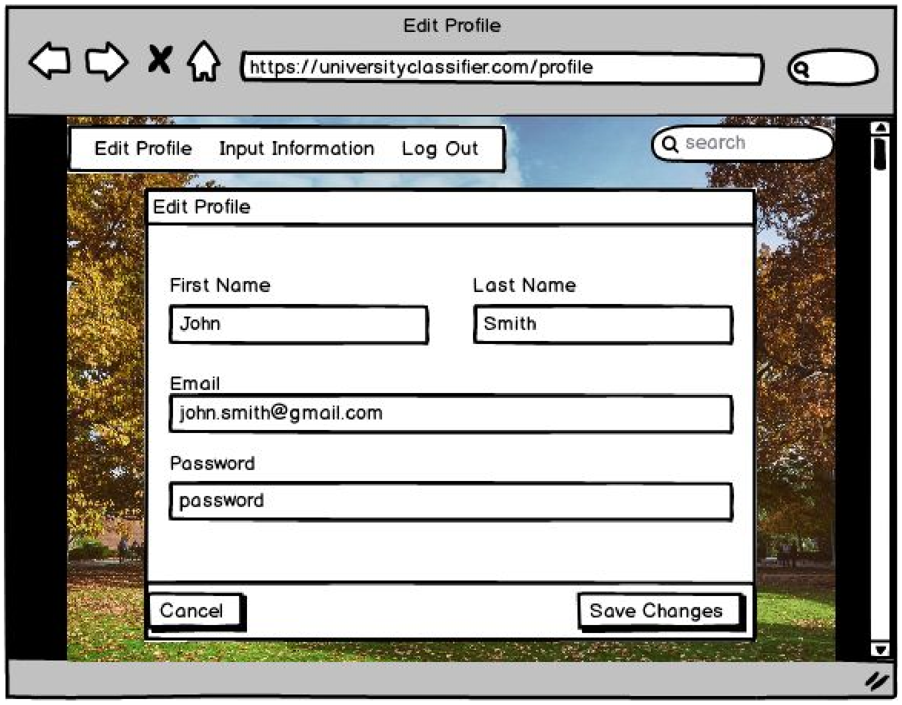


Figure 4 - The Register page

Figure 4 depicts the Register page. The user needs to enter their full name, email, and password in order to register. After pressing “Register”, the database verifies that a valid email has been entered and that there are no duplicate accounts, then creates an account and brings the user to the profile page. If there is an error, the error message is displayed.



Edit Profile

First Name	Last Name
John	Smith

Email: john.smith@gmail.com

Password: password

Figure 5 - The edit profile page

The user is brought to Figure 5, the profile page, after a successful login or account creation. This page can also be accessed by clicking “Edit Profile” in the top left corner. Here, the user can update any necessary profile information.

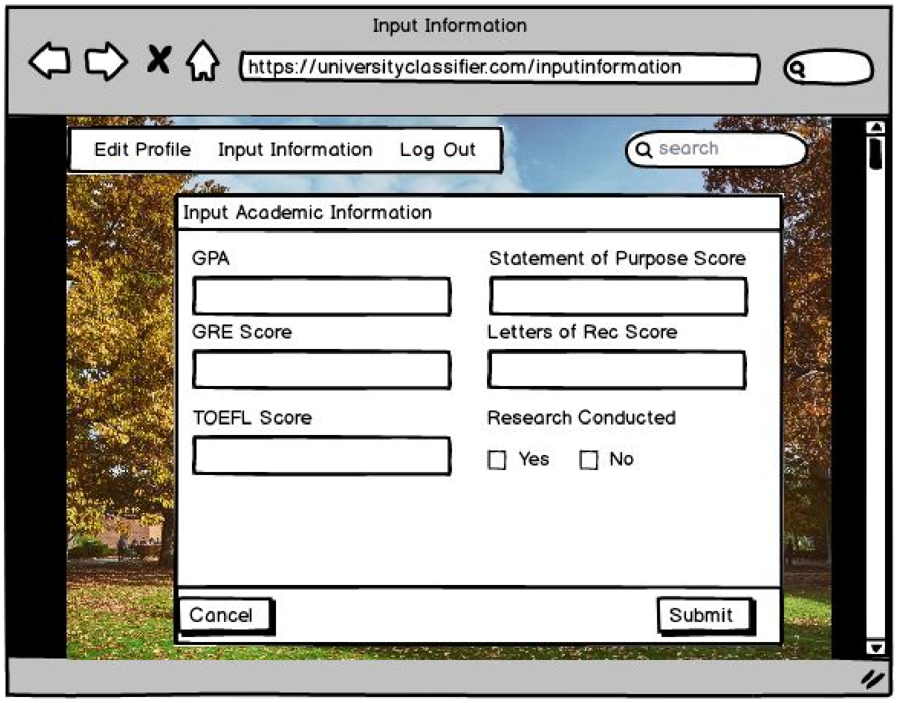
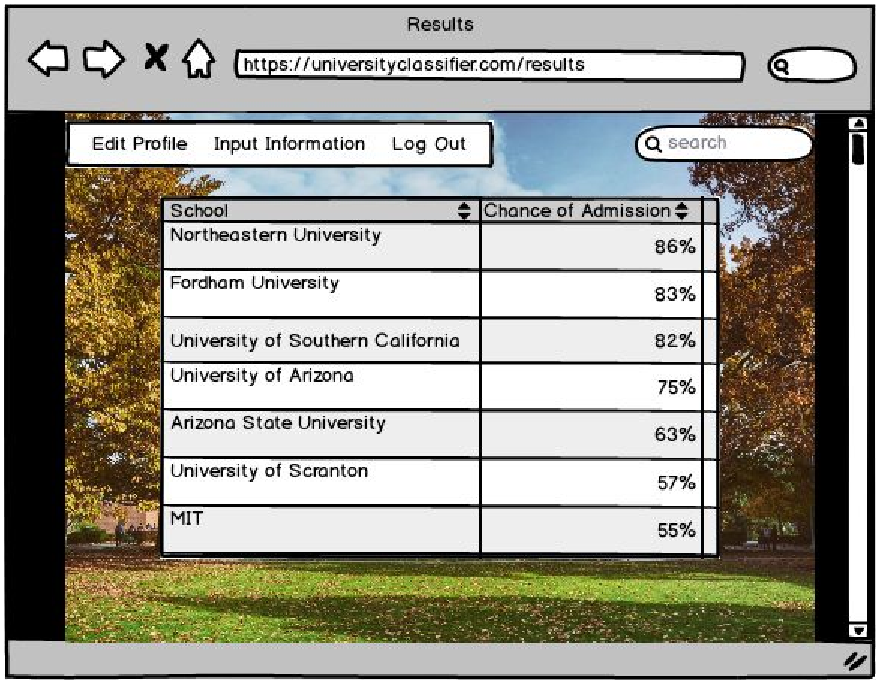


Figure 6 - This is the page where the user enters their academic information

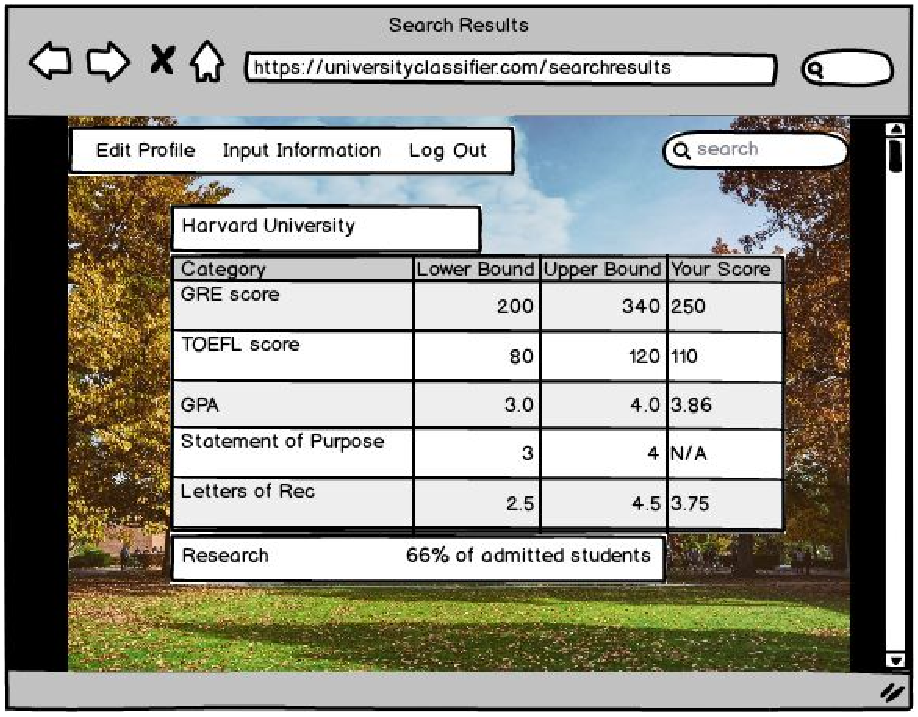
Figure 6 shows the “Input Information” page, which can be accessed by clicking the “Input Information” box in the top left corner. Here, the user can enter in any relevant information about their graduate school application. When the user presses Submit, the software will use the information provided to display a list of schools with the student’s percent chance of admission.



School	Chance of Admission
Northeastern University	86%
Fordham University	83%
University of Southern California	82%
University of Arizona	75%
Arizona State University	63%
University of Scranton	57%
MIT	55%

Figure 7 - An example results page generated after the user presses “Submit”

Figure 7 shows an example results page after the user presses “Submit” on the Input Information page.



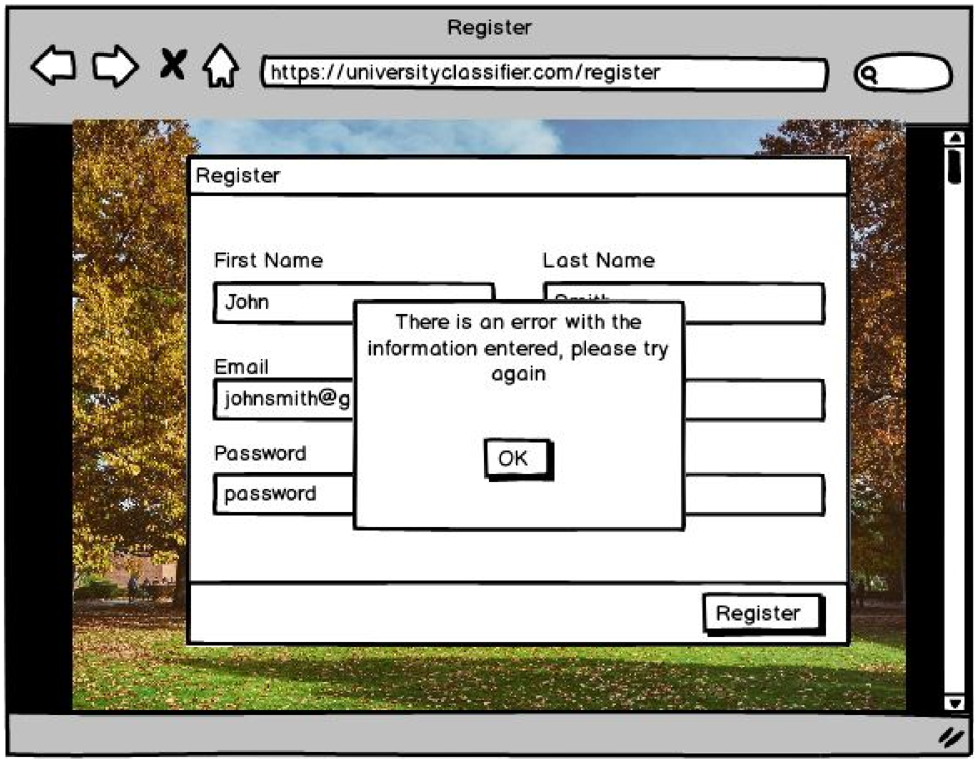
Harvard University

Category	Lower Bound	Upper Bound	Your Score
GRE score	200	340	250
TOEFL score	80	120	110
GPA	3.0	4.0	3.86
Statement of Purpose	3	4	N/A
Letters of Rec	2.5	4.5	3.75

Research 66% of admitted students

Figure 8 - The results of a university search

Figure 8 shows the results of entering "Harvard" in the search bar in the top left corner. For each facet of the graduate school application, the software displays the upper and lower bound of what students admitted to the given school have scored, as well as the user's score.



There is an error with the information entered, please try again

Figure 9 - The error message

Figure 9 shows an example of an error message. The error message can be displayed anywhere in the system when the user enters invalid information.