

Northeastern University
Department of Electrical and Computer Engineering

EECE 4520 Software Engineering

Final Report:
University Classifier

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

April 15, 2019

Table of Contents

| | |
|--|----------|
| Problem Statement | 6 |
| Objective | 6 |
| Rationale | 6 |
| Existing Systems | 6 |
| Proposed Systems | 6 |
| Team Details and Plan of Action | 6 |
| Team Roles | 6 |
| Action Plan | 8 |
| Requirements Analysis - Updated SRS | 8 |
| Introduction | 8 |
| Purpose of this Document | 8 |
| Scope of the Development Project | 8 |
| Definitions, Acronyms, and Abbreviations | 9 |
| References | 9 |
| Overview of Document | 9 |
| General Description | 10 |
| User Characteristics | 10 |
| Product Perspective | 10 |
| Overview of Function Requirements | 11 |
| Overview of Data Requirements | 11 |
| General Constraints, Assumptions, Dependencies, Guidelines | 12 |
| User View of Product Use | 12 |
| Specific Requirements | 15 |
| External Interface Requirements | 15 |
| User Interface- N/A | 15 |
| Hardware Interface - N/A | 15 |
| Software Interfaces | 15 |
| Detailed Description of Functional Requirements | 16 |
| Template for describing functional requirements | 16 |
| Create an Account | 16 |
| Log In | 16 |
| Update Profile | 17 |
| Input Information | 17 |
| Search for Schools | 17 |

| | |
|--|-----------|
| Cancel | 17 |
| Error Message | 18 |
| Performance Requirements | 18 |
| Quality Attributes | 18 |
| System Design - Updated SDS | 19 |
| Introduction | 19 |
| Purpose of this Document | 19 |
| Scope of the Development Project | 19 |
| Definitions, Acronyms, and Abbreviations | 19 |
| References | 20 |
| Major software requirements | 20 |
| Design constraints, limitations | 20 |
| Design Constraints | 20 |
| Limitations | 21 |
| Design Goals | 21 |
| Changes to requirements | 21 |
| Overview of Document | 21 |
| Data Design | 21 |
| Data Objects and resultant data structures | 21 |
| File and database structures | 22 |
| System Architecture Description | 23 |
| Overview of Modules / Components | 23 |
| Structure and Relationships | 24 |
| Detailed description of components | 25 |
| Component template description | 25 |
| Presentation Layer | 26 |
| Application Layer | 27 |
| Data Layer | 29 |
| School Information Database | 30 |
| User Database | 31 |
| User Interface | 32 |
| School Profile | 32 |
| Admissions Profile | 33 |
| Student Account | 34 |
| Application profile | 35 |
| Datafile: List of universities | 36 |
| Interface Design | 37 |
| Test Document | 42 |

| | |
|---------------------------------|-----------|
| Algorithm Testing | 42 |
| Database Testing | 42 |
| Application Testing | 48 |
| User Manual/User Guide | 52 |
| Creating an account | 52 |
| Signing in: | 52 |
| Inputting Academic Information: | 52 |
| Search for School | 52 |
| Update Profile | 52 |
| Log Out | 52 |
| Glossary | 52 |
| References | 53 |
| Appendix | 53 |
| Use Cases | 54 |
| Class Diagram | 70 |
| Gantt Chart | 71 |
| Subsystem | 72 |
| State Machine | 74 |
| Sequence Diagram | 75 |
| Extra Wire Frames | 75 |

1. Problem Statement

a. Objective

The purpose of this project is to use existing 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. We also allow for students to search for a particular school and see their scores compared against the scores of an average student at that school.

b. Rationale

This project is useful for undergraduate students to determine their chances of getting into different graduate schools based on information such as their scores and the university rating. This application is needed by students who are unsure of their graduate school options. We provide this service as there is no service out there that works as simply for students looking towards a graduate school.

c. Existing Systems

Several programs already exist that match students with potential colleges. The most popular of these is the College Board College Search capability, where students can filter schools by test scores, location, program of study, and other factors. However, this only works for undergraduate applications.[3]The Princeton Review and Gradtrak have similar programs, but they do not take student's application profiles into account. [5][6] Gradschoolmatch.com can filter by test scores, but it is clumsy and hard to use - it takes fifteen minutes to set up a profile.[4]

d. Proposed Systems

The proposed system will allow the student will create an account where they can enter the aspects of their graduate school application, such as GRE score, TOEFL score, or if they conducted research. A neural network will take these parameters and return a list of schools along with the student's percent chance of admission to the school. In addition, the student can search for a particular school, and see their admissions criteria displayed next to the average criteria for students who have been admitted.

2. Team Details and Plan of Action

a. Team Roles

- i. Felix Chan - "Team Support/Flex Member"

- Helped with documentation
 - Tested the application
 - Lent assistance to areas that needed some extra hands
- ii. Marissa D'Alonzo- "Database Engineer/Project Manager"
- Preprocess the university datasets. This included joining three datasets with different rankings for schools, eliminating duplicates, and calculating an average ranking for each school.
 - Create and maintain Amazon Web Services remote Microsoft SQL database.
 - Create and populate tables in database.
 - Coordinate meetings.
 - Edit presentations and paper.
 - Create and update relevant documentation, including use case diagrams, Software Requirements Specification, and Gantt chart
- iii. Sumer Malhotra
- iv. Jensen Rosemund - "Application Engineer"
- Wrote functions in C# to create profile, log in, search for schools, and update profile
 - Connect C# application to Microsoft SQL database
 - Connect C# application to python script
 - Test C# application
 - Assist with documentation
- v. Stockton Sheehan - "Machine Learning Engineer/Python coder"
- Wrote python script using Tensorflow and Keras to produce and save an algorithm which takes student info as an input and returns admission chance for a list of schools based on their ranking; this program isn't used directly in our system but produces a file (the algorithm) which is used
 - Trained algorithm using dataset of student information and admission chances
 - Tweaked parameters of algorithm in order to generate results which were accurate as possible
 - Tested algorithm to ensure that each parameter correctly influenced the output (ex. Test with an input; then slightly lower just one of the test scores and retest to check if admission chance slightly goes down)
 - Created python script with a function that takes in all necessary inputs, loads the saved algorithm, runs it on the data, and returns

an string which includes a list of schools with admission chances;
this program is directly called in our system

- Produced python script to generate average academic scores of students with a high admission chance based on school each ranking

b. Action Plan

Our group worked on diagrams collaboratively, but we divided the tasks of the system implementation based on each member's skills and software knowledge. Dividing up these tasks required lots of communication between team members to ensure that each one of our components would work properly together. We believed that using this method would allow our group to work as efficiently as possible. For example, there would not have been much of an advantage to ask a group member who has never worked with C#/python/sql to work on a component developed in that software. Once coded, each component was tested individually by the team member who worked on it. Afterwards, the entire team worked together on testing different groups of components as well as the system as a whole.

3. Requirements Analysis - Updated SRS

a. Introduction

This section gives a scope description and overview of everything included in this SRS document. Also the purpose of this document is described and a list of abbreviations and definitions is provided.

i. Purpose of this Document

The purpose of this document is to give a detailed description of the requirements for the "University Classifier" software. It will illustrate the purpose and complete declaration for the development of the system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team.

ii. 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. If we were to develop this further, we would also filter out schools based off the desired major of the student.

iii. Definitions, Acronyms, and Abbreviations

iv.

| 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 |

v. 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].

vi. Overview of Document

The remainder of this document includes three chapters and appendices. The second one provides a general description of the system functionality and system interaction with other systems. This chapter also introduces different types of stakeholders and their interaction with the system. Furthermore, the chapter also mentions the system constraints and assumptions about the project.

The third chapter provides the requirements specification in detailed terms and a description of the different system interfaces. Different specification techniques are used in order to specify the requirements more precisely for different audiences.

The fourth chapter deals with the any other/additional requirements that may be deemed necessary in order for the completion and smooth execution of the project.

The Appendices in the end of the document include all the results of the requirements including use cases, class diagrams and extra wire frames.

b. General Description

i. User Characteristics

The main purpose of this system is to help prospective graduate students search for and select graduate schools that are determined to fit the needs of the student. Finding the perfect graduate program can be a daunting task for someone who's finishing up their undergraduate studies, or for someone who's looking to go back to school while dealing with their daily lives. Therefore, this system needs to be intuitive, easy to use, and most importantly, effective. If the system does a satisfactory job of supplying

users with programs that are the perfect fit, then this system has completed its purpose. The following human characteristics will be essential for the system to meet the users' needs.

- Intuitive - The interface must be simple enough to understand without a user manual
- Efficient - Generating a report shouldn't take an excessive amount of time
- Attractive - The UI must be appealing and engaging in order to keep the users returning.
- Responsive - Navigating through the system should be smooth to prevent user irritation.

ii. Product Perspective

This system in of itself, is a stand-alone product, however, it will be built based upon data gathered from external sources. This data will consist of values used to represent the likelihood of the user being accepted to a particular graduate school using the data about the user and the university in question. The user data will include GPA, GRE score, TOEFL score, and other parameters. The university data will be gathered once upon the system's initial release, with possible periodic updates to the data.

While this product is a stand-alone product, it can be used in tandem with other graduate program search tools. No tool is perfect, so a user may wish to use this product alongside others that account for factors not considered in our application before making a final decision. Also, this product has no external interfaces or hardware requirements. The server that will hold all data is considered to be apart of the software system, so there's no external interface requirement. And since the system can be ran from a computer, there is no hardware requirement besides the single computer.

iii. Overview of Function Requirements

The functional requirements of the system are as follows:

1. The user must be able to create an account.
2. The user must be able to log into their account after its creation.
3. The user must be able to update their profile.
4. The user must be able to input their information and be given a list of schools with their chance of admission as a percentage.
5. 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.

6. The user must be able to cancel any action
7. The application must display an error page if the information given is incorrect

iv. Overview of Data Requirements

This system is entirely data-driven. The user profile will contain the following information:

1. Name
2. Password
3. GRE score
4. TOEFL score
5. Research conducted
6. Letter of Recommendation
7. Statement of Purpose
8. GPA

The university profile will contain the university name, as well as data points 3 through 8 in the user profile, represented as the range of scores achieved by admitted students. This data will be collected from pre-existing Kaggle datasets.

After the data has been collected from the user, it will be fed into an algorithm that generates the report. If the user is searching for a specific school, the report will output a list of all the parameters accompanied by the range of scores needed to maximize the chance of admission. If the user is inputting their own information, it will return a list of schools with the chance of admission to each.

v. General Constraints, Assumptions, Dependencies, Guidelines

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

vi. User View of Product Use

The following figures show a rough outline of the look of each page of the application.



Figure 1 - The home screen of the application

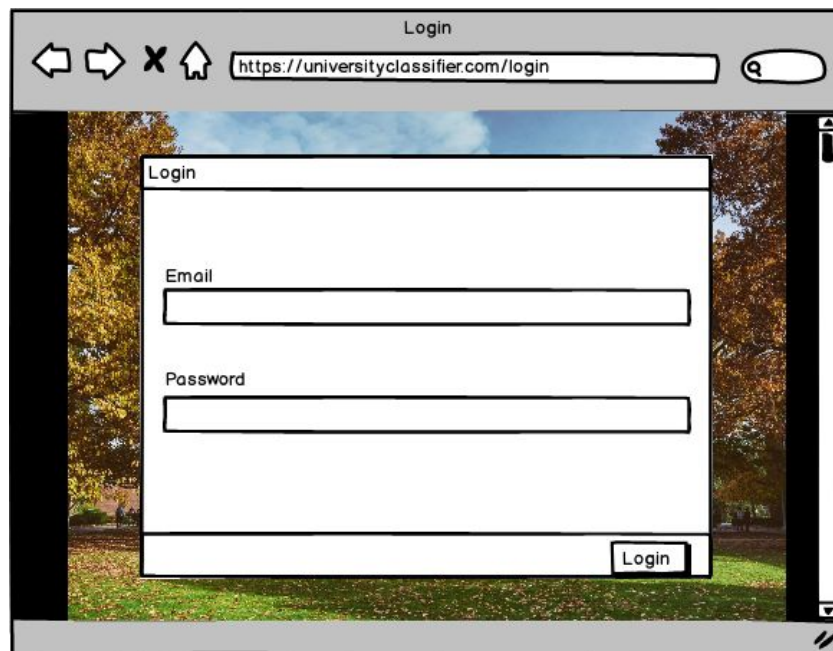


Figure 2 - The Login page

The Register and Edit Profile options look very similar to Figure 2, and can be found in Appendix 5.3, along with an example error message.

Input Information

https://universityclassifier.com/inputinformation

Edit Profile Input Information Log Out

search

Input Academic Information

GPA

GRE Score

TOEFL Score

Statement of Purpose Score

Letters of Rec Score

Research Conducted

☐ Yes ☐ No

Cancel Submit

Figure 3 - This is the page where the user enters their academic information

Results

https://universityclassifier.com/results

Edit Profile Input Information Log Out

search

| 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 4 - An example results page generated after the user presses "Submit"

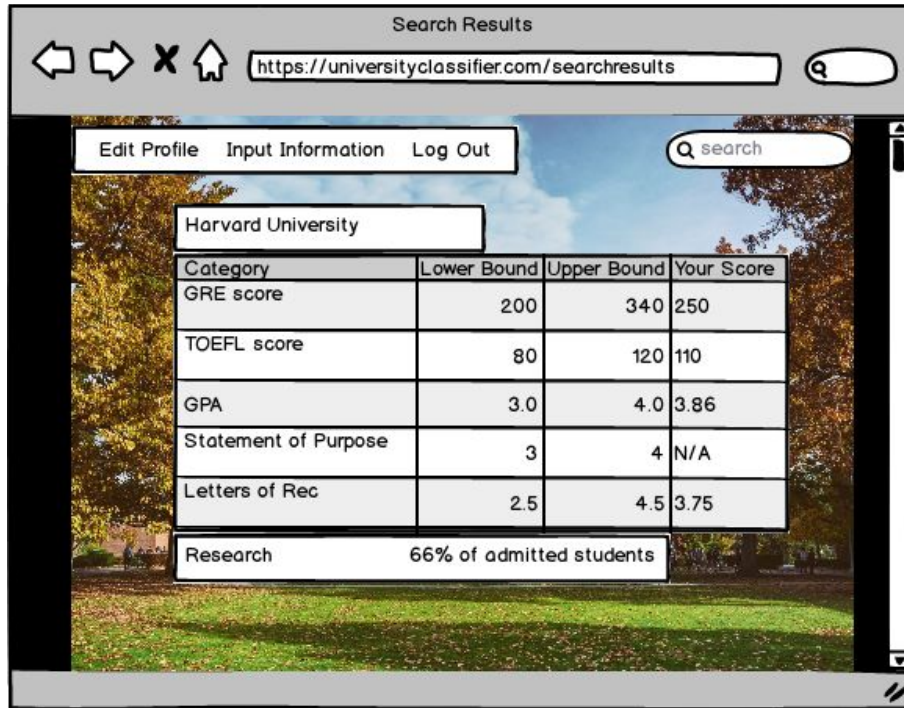


Figure 5 - The results of a university search

c. Specific Requirements

i. External Interface Requirements

- User Interface- N/A
- Hardware Interface - N/A
- Software Interfaces

When the application is opened, the user will be shown a page with the following two buttons:

1. Register - Leads to a page where the user will be prompted to enter information such as desired username and password.
2. Login - Lead to a page prompting the user to enter username and password. The interface will display if a password is incorrect and prompt the user to retry entering the details.

The user will reach the main application interface after successfully logging in. This page contains the following options:

1. Edit Profile - Allows the user to enter or modify information to be displayed on his or her account.

2. Enter Information - Prompts the user to enter academic information.
3. Search Schools - Prompts the user to select a school from the list of options.
4. Logout - Lead the user back to the initial page with 'Register' and 'Log in'

Both 'Enter Information' and 'Search Schools' include a 'Generate Results' button which will display the results if all of the necessary information is entered; otherwise, the user will be prompted to fill out the missing information. Clicking on that button will lead the user to the results page where they will be able to see the results generated from the algorithm, which would include either a list of schools that the user can likely get into or a percent chance of admission to the chosen school. This page has a 'return' button in order to return to the main screen.

Our system does not require an interface between hardware and software since no hardware components are used. The system will include an interface between the application and the algorithm. The application will send the user results to the algorithm, the algorithm will compute results based on the user inputs, and the generated results will be sent back to the application to be displayed. The application will also need an interface between itself and the dataset of schools in order for the user to select an option. An interface between the system and the dataset for scores and admission percent chances will not need to be included since the algorithm will already be trained using this data in order to generate its results.

- ii. Detailed Description of Functional Requirements
- iii. Template for describing functional requirements

The template used to describe functional requirements is as follows:

- *Purpose:* What is accomplished by this functional requirement
- *Inputs to the Component:* What the user must input to the computer
- *Processing:* What the computer does to meet the requirement
- *Outputs:* What is returned by the system to the user

iv. Create an Account

- *Purpose:* This function allows the user to access all other features of the application. Without an account, a user will not be able to successfully log in, so they cannot reach the main page of the application.
- *Inputs to the Component:* User will type in the following inputs: Desired username, password, and email address
- *Processing:* The system will check that the username is not already taken and that the password meets the standard requirements
- *Outputs:* With valid inputs, the user's account information will be stored so they will be able to login; with invalid inputs, the user will be prompted to reenter information

v. Log In

- *Purpose:* This function allows a user to view their specific results generated by the application without having to input their information every time they access it.
- *Inputs to the Component:* User will type in the following inputs: Chosen username, and password
- *Processing:* The system will check that the username and password are both registered within the system, as well as verifying that they match each other.
- *Outputs:* With valid inputs, the user may access their account information and utilize the application's otherwise functions; with invalid inputs, the user will be given an error notification.

vi. Update Profile

- *Purpose:* This function allows a user to add extra information to their profile, allowing for more accurate searches and results when using the application.
- *Inputs to the Component:* User will input data, such as GPA, names of classes, desired field of study, and extracurricular activities.
- *Processing:* The system will take all inputs, adjusting the data if it conflicts with previous inputs. This will happen in cases such as updating the GPA. If there is no conflict, it will add the data to the profile.
- *Outputs:* The user's profile will include the new data inputted.

vii. Input Information

- *Purpose:* This function allows the user to input their information the first time, with all of their credentials and merits. After doing so, this allows the system to provide a list of schools to show
- *Inputs to the Component:* User will input data, which includes GPA, extracurricular activities, and desired field of study.
- *Processing:* The system will take the inputs and save them under their relevant fields. For example, as GPA is a common input, it will be saved under the GPA label.
- *Outputs:* The user will be presented with a list of schools, along with an approximation of their chance to be admitted.

viii. Search for Schools

- *Purpose:* This function is to allow for users to see if they are in good standings to enter the school of their choice, and if not, they can see where they may be lacking.
- *Inputs to the Component:* The user will input the name of the school.
- *Processing:* The system will run through the database for a school name that matches the input. If no school matches the input, the output will reflect this.
- *Outputs:* Either a single school, or a list of schools if the input is not specific enough. it will show all schools with similar names., with the scores of tests and grades in order to enter. If there is no school in the database with the inputted name, it will show that there are no results.

ix. Cancel

- *Purpose:* This function will allow the user to stop what they are doing in the case that there is an error, or they wish to stop using the function.
- *Inputs to the Component:* The user will click on a cancel button that will be present in all functions.
- *Processing:* The system will act once the cancel button is pressed at any function.
- *Outputs:* The user will be presented the page prior to when they cancelled.

x. Error Message

- *Purpose:* This function ensures that all inputs will comply with what the application is ready to handle.

- *Inputs to the Component:* The user can input anything into any function.
- *Processing:* The system checks if the input from the user does not match the specified requirements of the function they are using.
- *Outputs:* The system will output an error message under the invalid input on the page that they are currently viewing.

xi. Performance Requirements

- Shorten processing time of the application
 - The time between generating and displaying results should be within 20 seconds
 - This will most likely be done with a python script
- Be able to handle requests from multiple users at once
 - Not a huge priority, with the only expected drawback being extended loading times
- User switching between pages should be within 5 seconds

xii. Quality Attributes

- Security is a main factor
 - Users should never have their data stolen by third parties from the system
 - 3 attempts allowed per account before a security email is sent
- Application should be available almost all the time
 - System should be going down only for maintenance and updates

4. System Design - Updated SDS

a. Introduction

i. 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.

ii. 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.

iii. 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 |

iv. 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].

v. 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

vi. 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.

● 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.

● 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.

vii. 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.

viii. Changes to requirements

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

ix. 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.

b. Data Design

i. Data Objects and resultant data structures

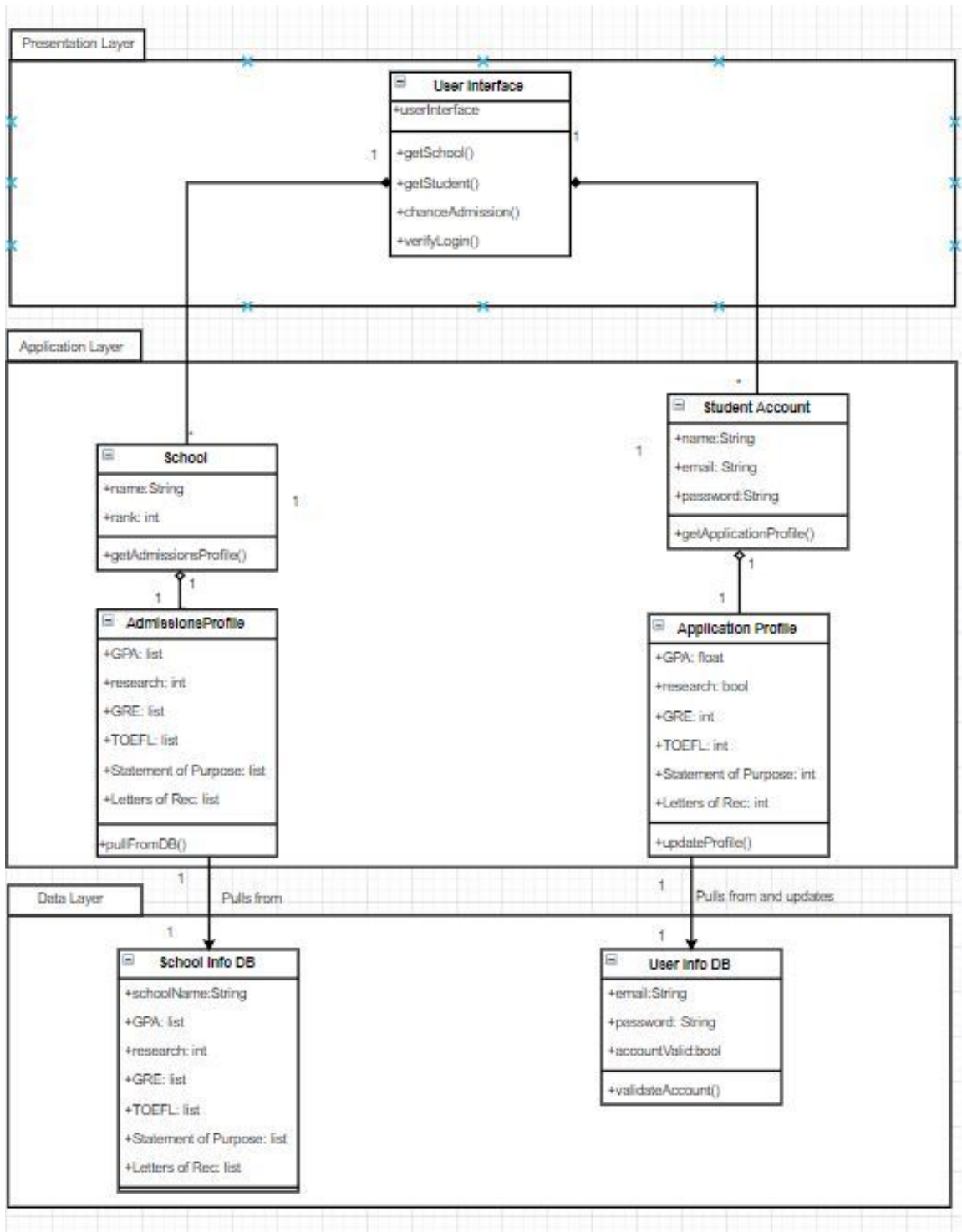


Figure 6 - The subsystem class diagram

ii. 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

c. System Architecture Description

i. 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.

ii. Structure and Relationships

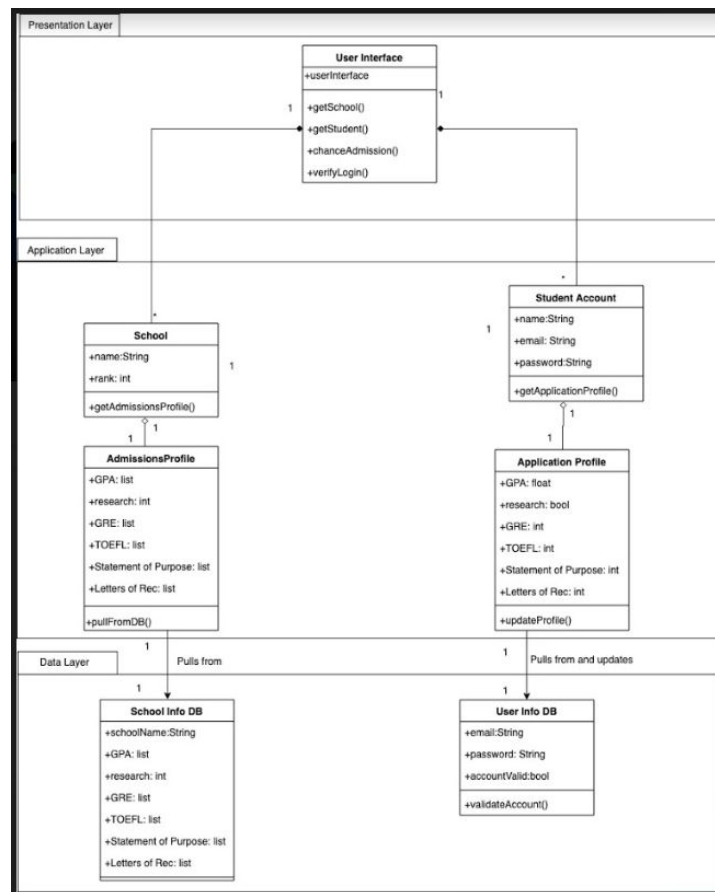


Figure 7 - The subsystem model of our system

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.

d. Detailed description of components

i. 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.

ii. 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.

iii. 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.

iv. 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.

v. 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

vi. 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

vii. 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

viii. 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

ix. 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

x. 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

xi. 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

xii. 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

e. 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.

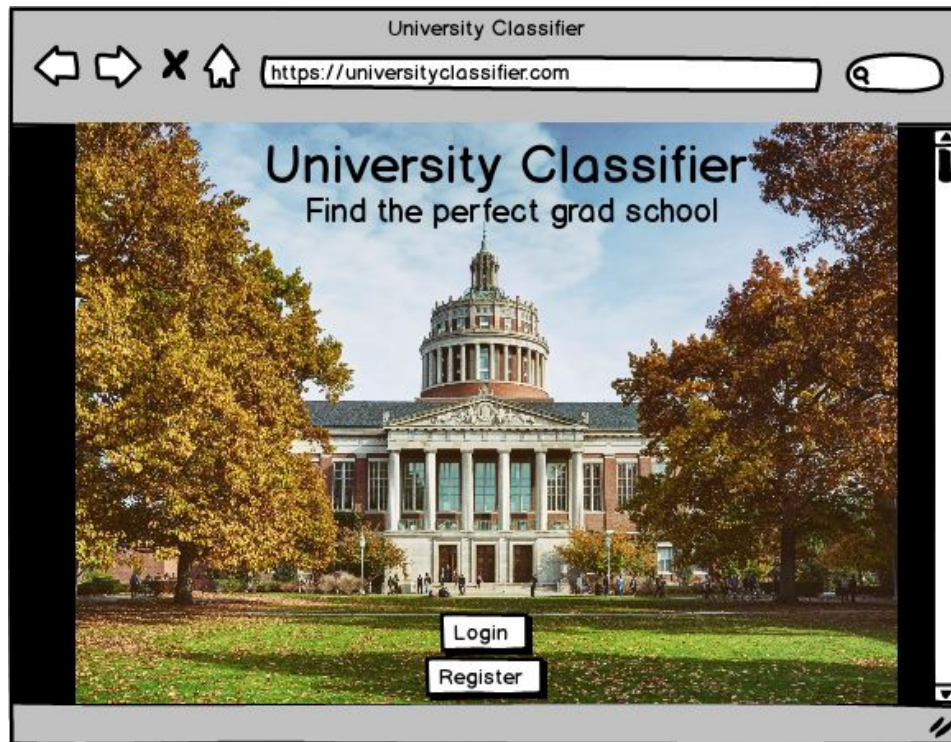


Figure 8 - The home screen of the application

Figure 8 depicts the home screen of the application. From here, the user can only choose to login or register.

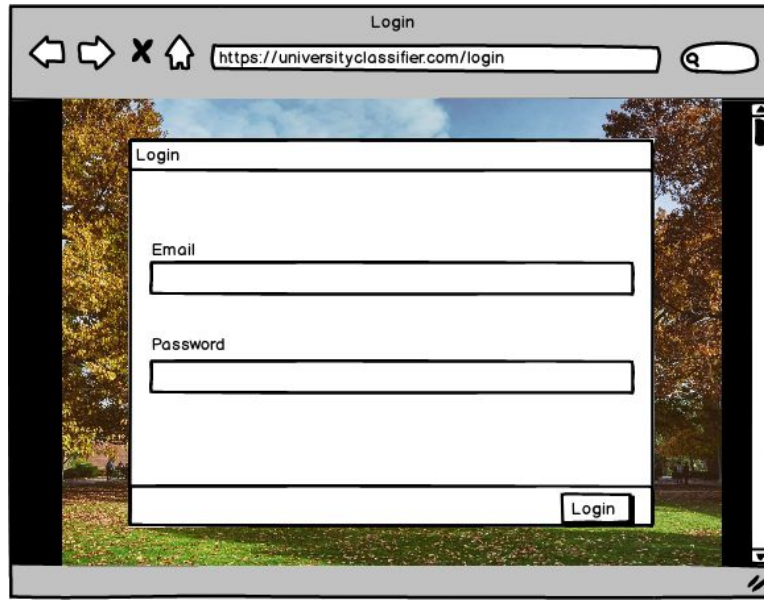


Figure 9 - The Login page

Figure 9 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 10 - The Register page

Figure 10 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.

Browser: Edit Profile
Address: https://universityclassifier.com/profile

Navigation: Edit Profile | Input Information | Log Out | Search

Form: Edit Profile

| | |
|-----------------------------------|------------------------------------|
| First Name | Last Name |
| <input type="text" value="John"/> | <input type="text" value="Smith"/> |

Email:

Password:

Buttons: Cancel | Save Changes

Figure 11 - The edit profile page

The user is brought to Figure 11, 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.

Browser: Input Information
Address: https://universityclassifier.com/inputinformation

Navigation: Edit Profile | Input Information | Log Out | Search

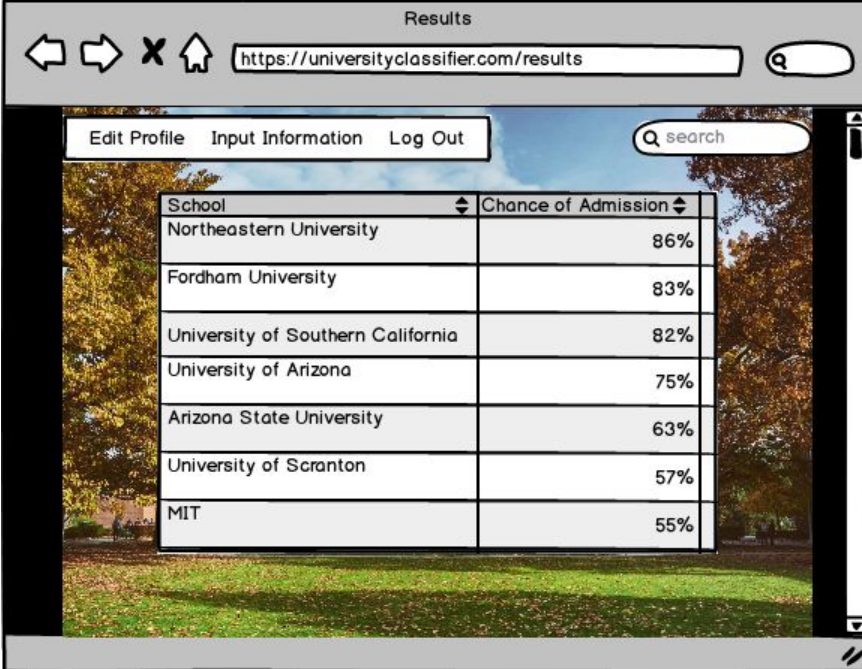
Form: Input Academic Information

| | |
|----------------------|--|
| GPA | Statement of Purpose Score |
| <input type="text"/> | <input type="text"/> |
| GRE Score | Letters of Rec Score |
| <input type="text"/> | <input type="text"/> |
| TOEFL Score | Research Conducted |
| <input type="text"/> | <input type="checkbox"/> Yes <input type="checkbox"/> No |

Buttons: Cancel | Submit

Figure 12 - This is the page where the user enters their academic information

Figure 12 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 13 - An example results page generated after the user presses “Submit”

Figure 13 shows an example results page after the user presses “Submit” on the Input Information page.

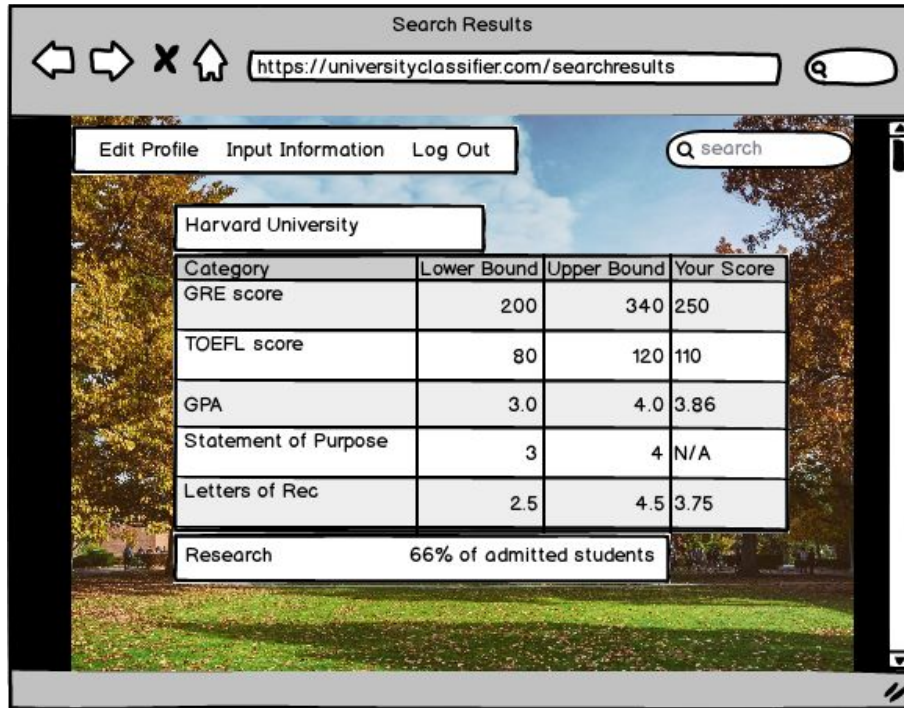


Figure 14 - The results of a university search

Figure 14 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.

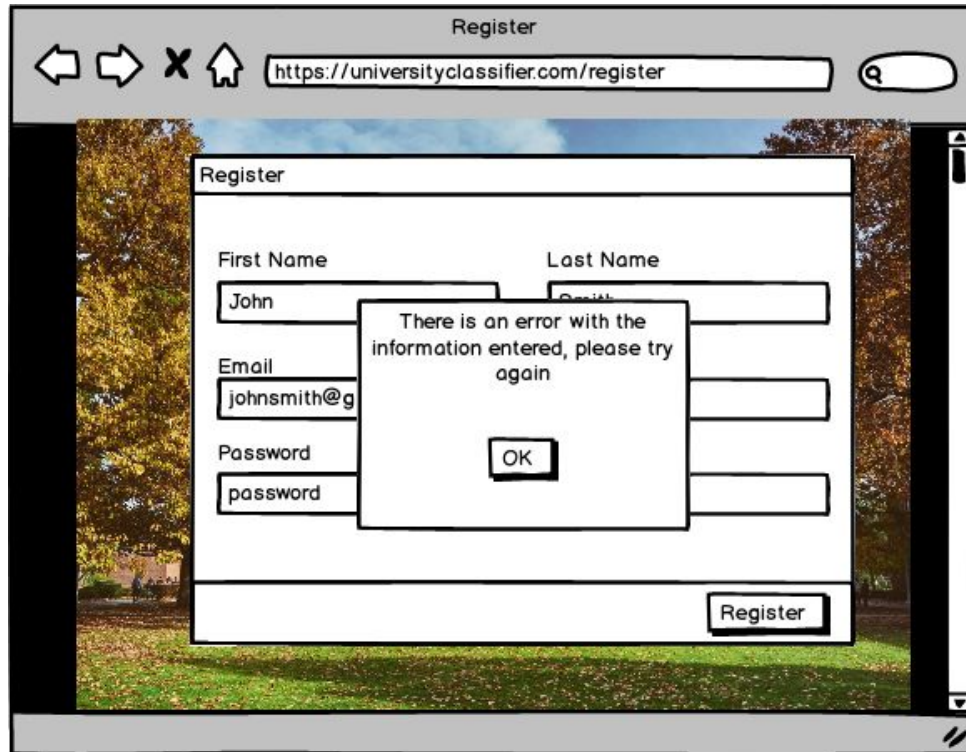


Figure 15 - The error message

Figure 15 shows an example of an error message. The error message can be displayed anywhere in the system when the user enters invalid information.

5. Test Document

The testing was divided into three categories - algorithm testing, database testing, and application testing. A modified sandwich approach was used - each module was tested with the necessary stubs/tickets before being integrated with other modules. The tables below show the outcomes of each test.

a. Algorithm Testing

The algorithm tests involve modifying one input at a time with all other values the same and observing the change in the output.

| Test | Outcome |
|----------------|--------------------------------------|
| Decrease GRE | Success- Admission Percent Decreases |
| Increase GRE | Success- Admission Percent Increases |
| Decrease TOEFL | Success- Admission Percent Decreases |

| | |
|----------------------------|--------------------------------------|
| Increase TOEFL | Success- Admission Percent Increases |
| Decrease SOP | Success- Admission Percent Decreases |
| Increase SOP | Success- Admission Percent Increases |
| Decrease LOR | Success- Admission Percent Decreases |
| Increase LOR | Success- Admission Percent Increases |
| Decrease CGPA | Success- Admission Percent Decreases |
| Increase CGPA | Success- Admission Percent Increases |
| Change research value to 0 | Success- Admission Percent Decreases |
| Change research value to 1 | Success- Admission Percent Increases |
| Make school rank lower | Success- Admission Percent Decreases |
| Make school rank higher | Success- Admission Percent Increases |

b. Database Testing

Note - In these tables, “valid” means that all fields are filled with the correct data type.

i. User Table

| Test | Outcome |
|---------------------------------------|---|
| Insert valid user into user table | User successfully inserted |
| Insert duplicate user into user table | User successfully inserted - this is handled at the application layer |
| Insert user with number as first name | User not inserted - invalid character |
| Insert user with number as last name | User not inserted - invalid character |
| Insert user with number as email | User not inserted - invalid character |
| Insert user with number as password | User not inserted - invalid character |
| Insert user with number as username | User not inserted - invalid character |
| Insert user with missing username | User inserted - this is handled at application layer |

| | |
|-------------------------------------|--|
| Insert user with missing password | User inserted - this is handled at application layer |
| Insert user with missing first name | User inserted - this is handled at application layer |
| Insert user with missing last name | User inserted - this is handled at application layer |
| Insert user with missing email | User inserted - this is handled at application layer |
| Alter user first name | User updated |
| Alter user last name | User updated |
| Alter user password | User updated |
| Alter user email name | User updated |
| Alter username | User updated |

ii. University Table

| Test | Outcome |
|---|---|
| Insert university with number as school name | University successfully inserted |
| Insert university with string as Times ranking | University not inserted - invalid character |
| Insert university with string as CWUR ranking | University not inserted - invalid character |
| Insert university with string as Shanghai ranking | University not inserted - invalid character |
| Insert university with string as average ranking | University not inserted - invalid character |
| Insert university with string as overall ranking | University not inserted - invalid character |
| Insert university with string as TOEFL | University not inserted - invalid character |
| Insert university with string as Statement of Purpose Score | University not inserted - invalid character |

| | |
|--|---|
| Insert university with string as Letter of Recommendation Score | University not inserted - invalid character |
| Insert university with string as CGPA | University not inserted - invalid character |
| Insert university with string as GRE | University not inserted - invalid character |
| Insert university with negative number as Times Ranking | University successfully inserted |
| Insert university with negative number as CWUR Ranking | University successfully inserted |
| Insert university with negative number as Shanghai Ranking | University successfully inserted |
| Insert university with negative number as average ranking | University successfully inserted |
| Insert university with negative number as overall ranking | University successfully inserted |
| Insert university with negative number as GRE Score | University successfully inserted |
| Insert university with negative number as TOEFL | University successfully inserted |
| Insert university with negative number as Statement of Purpose Score | University successfully inserted |
| Insert university with negative number as Letter of Recommendation Score | University successfully inserted |
| Insert university with negative number as CGPA | University successfully inserted |
| Insert university with too-high number for Times ranking | University successfully inserted |
| Insert university with too-high number for CWUR ranking | University successfully inserted |
| Insert university with too-high number for Shanghai ranking | University successfully inserted |
| Insert university with too-high number for GRE score | University successfully inserted |

| | |
|---|----------------------------------|
| Insert university with too-high number for TOEFL score | University successfully inserted |
| Insert university with too-high number for Statement of Purpose score | University successfully inserted |
| Insert university with too-high number for Letter of Recommendation score | University successfully inserted |
| Insert university with name that does not exist | University successfully inserted |
| Insert university without name | University successfully inserted |
| Insert university without Times ranking | University successfully inserted |
| Insert university without CWUR ranking | University successfully inserted |
| Insert university without Shanghai ranking | University successfully inserted |
| Insert university without average ranking | University successfully inserted |
| Insert university without overall ranking | University successfully inserted |
| Insert university without GRE score | University successfully inserted |
| Insert university without TOEFL score | University successfully inserted |
| Insert university without Statement of Purpose score | University successfully inserted |
| Insert university without Letter of Recommendation score | University successfully inserted |
| Insert university without CGPA score | University successfully inserted |

The database testing revealed several flaws in the university table. It will accept negative numbers as rankings, and will allow impossible scores and rankings to be inserted, as well as missing fields. However, this should not be an issue in the final product because the user does not have access to the university table, so they cannot insert invalid characters. Maintainers must be careful to insert only valid data into these fields when updating the table.

iii. Chance Admission Table

| Test | Outcome |
|------|---------|
|------|---------|

| | |
|---|--|
| Insert valid data into table | Chance admission successfully inserted |
| Insert row with person id that does not exist | Row not inserted - invalid foreign key |
| Insert row with school id that does not exist | Row not inserted - invalid foreign key |
| Insert row with negative person id | Row not inserted - invalid foreign key |
| Insert row with negative school id | Row not inserted - invalid foreign key |
| Insert row with negative chance of admission | Row successfully inserted- error handled at application layer |
| Insert row without person id | Row successfully inserted- error handled at application layer |
| Insert row without school id | Row successfully inserted - error handled at application layer |
| Insert row without chance of admission | Row successfully inserted - error handled at application layer |

iv. Academic Data Table

| Test | Outcome |
|---|--|
| Insert valid row into table | Row successfully inserted |
| Insert row missing GRE score | Row successfully inserted - no error |
| Insert row missing TOEFL score | Row successfully inserted - no error |
| Insert row missing Statement of Purpose score | Row successfully inserted - no error |
| Insert row missing Letter of Recommendation score | Row successfully inserted - no error |
| Insert row missing CGPA score | Row successfully inserted - no error |
| Insert row missing research | Row successfully inserted - no error |
| Insert row missing person id | Row successfully inserted - no error |
| Insert row negative GRE score | Row successfully inserted - handled at application layer |

| | |
|---|--|
| Insert row negative TOEFL score | Row successfully inserted - handled at application layer |
| Insert row negative Statement of Purpose score | Row successfully inserted - handled at application layer |
| Insert row negative Letter of Recommendation score | Row successfully inserted - handled at application layer |
| Insert row negative CGPA score | Row successfully inserted - handled at application layer |
| Insert row negative research | Row successfully inserted - handled at application layer |
| Insert row negative person id | Row not inserted - invalid foreign key |
| Insert row with too-high GRE score | Row successfully inserted - handled at application layer |
| Insert row with too-high TOEFL score | Row successfully inserted - handled at application layer |
| Insert row with too-high Statement of Purpose score | Row successfully inserted - handled at application layer |
| Insert row with too-high Letter of Recommendation score | Row successfully inserted - handled at application layer |
| Insert row with too-high CGPA score | Row successfully inserted - handled at application layer |
| Insert row with too-high research | Row successfully inserted - handled at application layer |
| Insert row with too-high person id | Row not inserted - invalid foreign key |
| Insert row with string GRE score | Row not inserted - invalid character |
| Insert row with string TOEFL score | Row not inserted - invalid character |
| Insert row with string Statement of Purpose score | Row not inserted - invalid character |
| Insert row with string Letter of Recommendation score | Row not inserted - invalid character |
| Insert row with string CGPA score | Row not inserted - invalid character |
| Insert row with string research | Row not inserted - invalid character |

| | |
|----------------------------------|--|
| Insert row with string person id | Row not inserted - invalid foreign key |
|----------------------------------|--|

c. Application Testing

i. Creating an account

| Test | Outcome |
|---|--------------------------------|
| Enter all valid information | User successfully created |
| Password does not match Verify Password | Error- passwords do not match |
| Do not enter first name | Error- empty field |
| Do not enter last name | Error- empty field |
| Do not enter username | Error- empty field |
| Do not enter password | Error- empty field |
| Do not enter verify password | Error- empty field |
| Enter username that already exists | Error - username already taken |
| Enter password that already exists | User successfully created |
| Enter email that already exists | User successfully created |
| Enter first name that already exists | User successfully created |
| Enter last name that already exists | User successfully created |

ii. Signing in

| Test | Outcome |
|--|---------------------------|
| Enter correct username and password | User brought to main page |
| Enter correct username, incorrect/nonexistent password | Error - invalid login |
| Enter correct password, incorrect/nonexistent username | Error- invalid login |
| Enter incorrect/nonexistent username | Error - invalid login |

| | |
|-----------------------------------|-----------------------|
| and password | |
| Enter username but not password | Error - missing field |
| Enter password but not username | Error - missing field |
| Do not enter username or password | Error - missing field |

iii. Inputting Academic Information

| Test | Outcome |
|---|---------------------------------|
| Valid information entered into every field | Report generated |
| GPA missing, all other info valid | Report generated |
| TOEFL missing, all other info valid | Report generated |
| Statement of Purpose missing, all other info valid | Report generated |
| Letter of Recommendation missing, all other info valid | Report generated |
| GRE missing, all other info valid | Report generated |
| Research missing, all other info valid | Report generated |
| GPA negative, all other info valid | Error - Outside of range 1-10 |
| TOEFL negative, all other info valid | Error - Outside range of 0-120 |
| Statement of Purpose negative, all other info valid | Error - Outside range of 0-5 |
| Letter of Recommendation negative, all other info valid | Error - Outside range of 0-5 |
| GRE negative, all other info valid | Error - Outside range of 34-260 |
| GPA too high, all other info valid | Error - Outside of range 1-10 |
| TOEFL too high, all other info valid | Error - Outside range of 0-120 |
| Statement of Purpose too high, all other info valid | Error - Outside range of 0-5 |
| Letter of Recommendation too high, | Error - Outside range of 0-5 |

| | |
|---|-------------------------|
| all other info valid | |
| GRE too high, all other info valid | Outside range of 34-260 |
| GPA string, all other info valid | Crash |
| TOEFL string, all other info valid | Crash |
| Statement of Purpose string, all other info valid | Crash |
| Letter of Recommendation string, all other info valid | Crash |
| GRE string, all other info valid | Crash |

iv. Search for School

| Test | Outcome |
|--|--|
| Enter a valid school name and press Search | List of your scores compared to scores of average participants given |
| Enter a nonexistent school name and press Search | Error - university not found |
| Enter nothing and press Search | Error - empty field |

v. Update Profile

| Test | Outcome |
|--|----------------------------------|
| Update email and password with matching password and verify password | Information successfully updated |
| Update email, leave password and verify password blank | Information successfully updated |
| Update password and verify password with same value, leave email blank | Information successfully updated |
| Update password and verify password with different values, leave email blank | Error - fields do not match |
| Update password, leave verify | Error - missing field |

| | |
|---|----------------------------------|
| password blank | |
| Update verify password, leave password blank | Error - missing field |
| Leave everything blank | Error - missing field |
| Update email with same value as current email, leave password and verify password blank | Information successfully updated |
| Update password and verify password with current password, leave email blank | Information successfully updated |
| Update email, password, and verify password with current email and password | Information successfully updated |

vi. Log Out

| Test | Outcome |
|-----------------|------------------------|
| Click "Log Out" | Return to login screen |

The testing of the application reveals two main weaknesses in our design - the system does not currently verify if an email ends in a valid email (@gmail.com, @aol.com, etc), and it does not verify if a number is entered instead of a string while creating an account. Other than that, there are no issues with the application.

6. User Manual/User Guide

a. The minimum specs you'll need:

- i. Windows 7/8/10 64-bit + Mac OSX Sierra
- ii. Core i3 2.4 GHz processor
- iii. 4GB of system RAM
- iv. Intel HD 4000 video card
- v. Wifi connection
- vi. Visual Studio Code for Windows or MonoDevelop for Mac

b. Creating an account

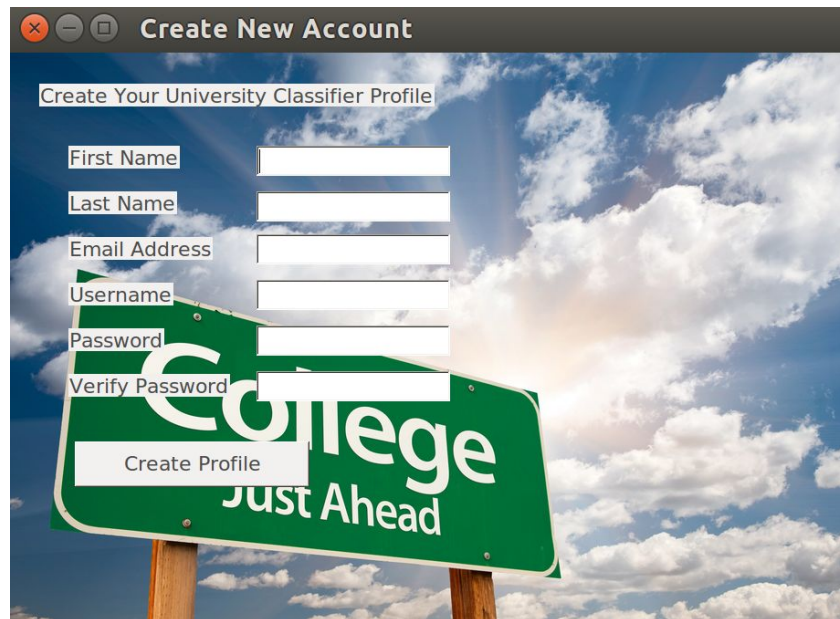


Figure 16 - The “Create Account” window

- i. When the application is run, there is an option of “New User?”. Clicking this will allow you to create your account.
 - Creation of an account requires your full name, your e-mail, a username, and a password.

c. Signing in:

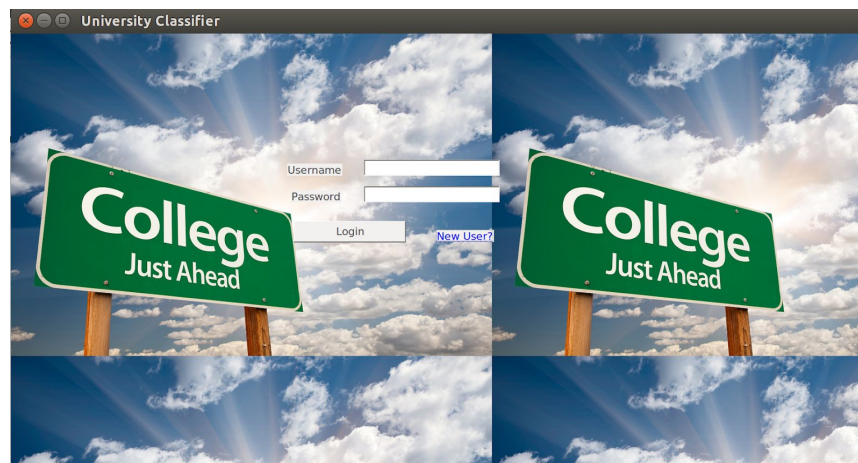


Figure 17 - The “Log In” window

- i. At the main menu of the application, enter the username and password that you created your account with.

d. Inputting Academic Information:

Figure 18 shows the "Input Information" window. It contains the following fields and controls:

- GPA:
- GRE Score:
- TOEFL Score:
- Statement of Purpose Score:
- Letters of Rec Score:
- Research Conducted: ☐ Yes ☐ No
- Buttons: Submit, Generate, Cancel

Figure 18 - The “Input Information” window

- i. After logging into the account that you made, you are sent into a menu to input your academic information, which includes the following.

- GPA (0 -8)
- GRE Score (34-260)
- TOEFL Score (0-120)
- Statement of Purpose Score (1-5)
- Letters of Recommendation Score (1-5)
- Research Conducted (y/n)

Enter whatever you have, then press “Submit”, then “Generate” to see which schools you have the best chance of getting into.

e. Search for School

Figure 19 shows the "Search for School" window. It contains the following elements:

- Search:
- Message: To display university stats, you must search for a school.

- i. After inputting your information, you can search for schools in the tab named “University Stats”. On that tab, you can search for schools, and see a table comparing your current academic status to the recommended scores for that school.
- f. Update Profile

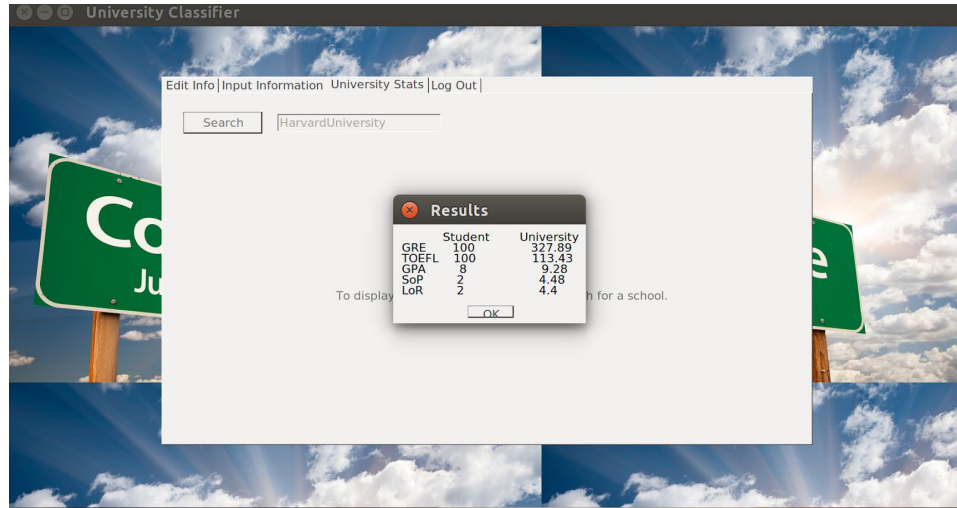


Figure 19 - The “Edit Info” tab

- i. Click on “Edit Info”. Here, you can choose to enter a new email address and/or password. “Password” and “Verify Password” must match in order for the password to be updated.
- g. Log Out
- i. On the menu, there is a tab called “Log Out”. On clicking the tab, you will be logged out of the session and returned to the login screen.

7. Glossary

| 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 |

8. 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].
- [3] *Big Future - College Search - Find colleges and universities by major, location, type, more.* [Online]. Available: <https://bigfuture.collegeboard.org/college-search>. [Accessed: 10-Apr-2019].
- [4] “Find Your Best Graduate Degree Options,” *Gradschoolmatch*. [Online]. Available: <https://www.gradschoolmatch.com/>. [Accessed: 10-Apr-2019].
- [5] “Grad School Search Engine,” *GradTrek*. [Online]. Available: <https://gradtrek.com/>. [Accessed: 10-Apr-2019].
- [6] “Find your dream Grad School,” *Graduate School Admissions | Compare Grad Schools | The Princeton Review*. [Online]. Available: <https://www.princetonreview.com/grad-school>. [Accessed: 10-Apr-2019].

9. Appendix

a. Use Cases

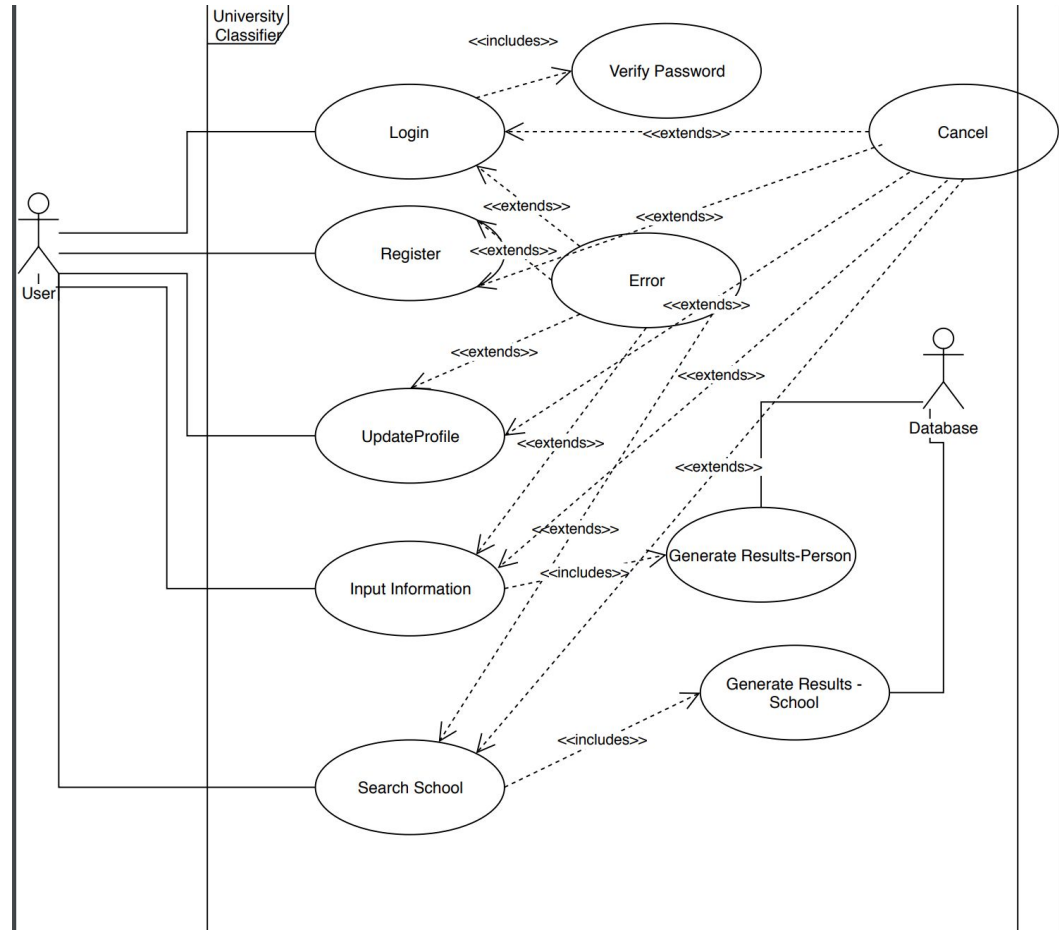


Figure 16 - The final use case diagram

USE CASE SPECIFICATIONS TEMPLATE

Use Case Identification and History

| | | |
|---------------------|---|-----------------|
| Use Case ID: | Classifier.Register.1 | |
| Use Case Name: | CreateProfile | Version No: |
| End Objective: | Create a profile that a user can sign into and modify | |
| Created by: | University Classifier | On (date): |
| Last Update by: | | On (date): |
| Approved by: | | On (date): |
| User/Actor: | User who accesses application for the first time | |
| Business Owner Name | University Classifier | Contact Details |
| Trigger: | User triggers use case by clicking 'Register' button | |
| Frequency of Use: | Once per user | |

Preconditions

User does not already have a profile

Basic Flow

| Step | User Actions | System Actions |
|------|---|--|
| 1 | User selects "Register" | System displays Create Profile page |
| 2 | User fills out profile fields and clicks "Submit" | System validates input and creates account |
| 3 | | Displays Successful Creation page |

Alternate Flow

| Step | User Actions | System Actions |
|------|------------------------------|--|
| 1 | User selects "Register" | System displays Create Profile page |
| 2 | User presses "Cancel" button | System returns to home page and does not save data |

Error Flow

| Step | User Actions | System Actions |
|------|---|--|
| 1 | User selects "Register" | System displays Create Profile page |
| 2 | User fills out profile fields and clicks "Submit" | System validates input and finds an error |
| 3 | | Displays error message and prompts user to try again |

Post Conditions

Application displays Successful Creation or error page

Includes or Extension Points

Includes:
a. Verify Information
Extends:
a. Error
b. Cancel

Special Requirements

N/A

Business Rules

N/A

Other Notes

This use case creates a user profile

| | | | |
|----------------------------|--|------------------------|---|
| Use Case ID: | Classifier.Login.1 | | |
| Use Case Name: | Login | Version No: | 1 |
| End Objective: | Sign into profile that user has created | | |
| Created by: | University Classifier | On (date): | |
| Last Update by: | | On (date): | |
| Approved by: | | On (date): | |
| User/Actor: | User who has created an account | | |
| Business Owner Name | University Classifier | Contact Details | |
| Trigger: | User triggers use case by clicking 'Log In' button | | |
| Frequency of Use: | Every time application is opened | | |

| |
|------------------------------|
| Preconditions |
| User has an existing account |

| Basic Flow | | |
|------------|--|-------------------------------------|
| Step | User Actions | System Actions |
| 1 | User selects "Log In" | System displays Log In page |
| 2 | Users enters username and password and clicks "Submit" | System validates input logs user in |
| 3 | | Displays profile |

| Alternate Flow | | |
|----------------|------------------------------|--|
| Step | User Actions | System Actions |
| 1 | User selects "Log In" | System displays Log In page |
| 2 | User presses "Cancel" button | System returns to home page and does not save data |

| Error Flow | | |
|------------|--|--|
| Step | User Actions | System Actions |
| 1 | User selects "Log In" | System displays Log In page |
| 2 | User fills out username and password and clicks "Submit" | System validates input and finds an error |
| 3 | | Displays error message and prompts user to try again |

| |
|--|
| Post Conditions |
| Application displays profile or error page |

| Includes or Extension Points |
|--|
| <p>Includes:</p> <ul style="list-style-type: none"> a. Verify Information <p>Extends:</p> <ul style="list-style-type: none"> a. Error b. Cancel |

| Special Requirements |
|----------------------|
| N/A |

| Buisness Rules |
|----------------|
| N/A |

| Other Notes |
|--|
| This use case logs a user into their profile |

| | | | |
|----------------------------|---|------------------------|---|
| Use Case ID: | Classifier.UpdateProfile.1 | | |
| Use Case Name: | UpdateProfile | Version No: | 1 |
| End Objective: | User modifies the information on his/her profile | | |
| Created by: | University Classifier | On (date): | |
| Last Update by: | | On (date): | |
| Approved by: | | On (date): | |
| User/Actor: | User who is logged into account | | |
| Business Owner Name | University Classifier | Contact Details | |
| Trigger: | User triggers use case by clicking 'Edit profile' button | | |
| Frequency of Use: | Not often unless a user makes frequent changes to his/her profile | | |

| Preconditions | |
|---|--|
| User has an existing account and is logged into their profile | |

| Basic Flow | | |
|------------|---|---|
| Step | User Actions | System Actions |
| 1 | User selects "Edit Profile" | System displays Edit Profile page |
| 2 | Users edits desired information and clicks "Submit" | System validates, updates and saves information |
| 3 | | Displays profile |

| Alternate Flow | | |
|----------------|------------------------------|--|
| Step | User Actions | System Actions |
| 1 | User selects "Edit Profile" | System displays Edit Profile page |
| 2 | User presses "Cancel" button | System returns to home page and does not save data |

| Error Flow | | |
|------------|--|--|
| Step | User Actions | System Actions |
| 1 | User selects "Edit Profile" | System displays Edit Profile page |
| 2 | User fills out username and password and clicks "Submit" | System validates input and finds an error |
| 3 | | Displays error message and prompts user to try again |

| Post Conditions |
|--|
| Application displays updated profile or error page |

| Includes or Extension Points |
|---|
| Includes: a. Verify Information Extends: a. Error b. Cancel |

| Special Requirements |
|----------------------|
| N/A |

| Buisness Rules |
|----------------|
| N/A |

| Other Notes |
|--------------------------------------|
| This use case updates a user profile |

| | | | |
|----------------------------|--|------------------------|---|
| Use Case ID: | Classifier.InputInfo.1 | | |
| Use Case Name: | InputInfo | Version No: | 1 |
| End Objective: | User inputs their scores and college information | | |
| Created by: | University Classifier | On (date): | |
| Last Update by: | | On (date): | |
| Approved by: | | On (date): | |
| User/Actor: | User who is logged into account | | |
| Business Owner Name | University Classifier | Contact Details | |
| Trigger: | User triggers use case by clicking 'Enter Academic Information' button | | |
| Frequency of Use: | One time per user, but users can update their information if their scores change | | |

| |
|--|
| Preconditions |
| User must have an existing account and be logged in. Must be on the update page. |

| Basic Flow | | |
|------------|---|---|
| Step | User Actions | System Actions |
| 1 | After the user first creates their profile, they will be brought to a blank profile page. | System displays blank profile page |
| 2 | Users edits desired information and clicks "Save" | System validates, updates and saves information |
| 3 | | Displays profile |

| Alternate Flow | | |
|----------------|---|--|
| Step | User Actions | System Actions |
| 1 | After the user first creates their profile, they will be brought to a blank profile page. | System displays blank profile page |
| 2 | User presses "Cancel" button | System returns to home page and does not save data |

| Error Flow | | |
|------------|---|--|
| Step | User Actions | System Actions |
| 1 | After the user first creates their profile, they will be brought to a blank profile page. | System displays blank profile page |
| 2 | Users edits desired information and clicks "Save" | System validates input and finds an error |
| 3 | | Displays error message and prompts user to try again |

| Post Conditions |
|--|
| Application displays profile or error page |

| Includes or Extension Points |
|--|
| Includes: <ul style="list-style-type: none"> a. Verify Information Extends: <ul style="list-style-type: none"> a. Error b. Cancel |

| Special Requirements |
|----------------------|
| N/A |

| Business Rules |
|----------------|
| N/A |

| Other Notes |
|---|
| This use case allows a user to input information into their profile |

| | | | |
|--|--|-----------------|---|
| Use Case ID: | Classifier.GenerateResults.1 | | |
| Use Case Name: | GenerateResults | Version No: | 1 |
| End Objective: | Take user information and produces a list of percent chance of admission to a variety of schools | | |
| Created by: | University Classifier | On (date): | |
| Last Update by: | | On (date): | |
| Approved by: | | On (date): | |
| User/Actor: | User who has logged into their account | | |
| Business Owner Name | University Classifier | Contact Details | |
| Trigger: | User enters information and presses "Generate Results" | | |
| Frequency of Use: | Very often since users this is the main purpose of the application | | |
| Preconditions | | | |
| User must be logged in and have valid profile information. | | | |

| Basic Flow | | |
|------------|------------------------------|--|
| Step | User Actions | System Actions |
| 1 | User hits "Generate Results" | System validates profile information and searches database |
| 2 | | Displays list of schools and percent change of admission |

| Alternate Flow | | |
|----------------|---|--|
| Step | User Actions | System Actions |
| 1 | User clicks on "Generate Results" button. | System displays search page |
| 2 | User presses "Cancel" button | System returns to home page and does not save data |

| Error Flow | | |
|------------|---|--|
| Step | User Actions | System Actions |
| 1 | User clicks on "Generate Results" button. | System validates input and finds an error |
| 2 | | Displays error message and prompts user to try again |

| Post Conditions |
|---|
| Application displays school information or error page |

| Includes or Extension Points |
|---|
| Includes: a. Verify Information Extends: a. Error b. Cancel |

| Special Requirements |
|----------------------|
| N/A |

| Buisness Rules |
|----------------|
| N/A |

| Other Notes |
|---|
| This use case allows a user to search for schools |

| | | | |
|--|--|-----------------|---|
| Use Case ID: | Classifier.SchoolSearch.1 | | |
| Use Case Name: | SchoolSearch | Version No: | 1 |
| End Objective: | User searches for graduate school they are interested in | | |
| Created by: | University Classifier | On (date): | |
| Last Update by: | | On (date): | |
| Approved by: | | On (date): | |
| User/Actor: | User who has logged into their account | | |
| Business Owner Name | University Classifier | Contact Details | |
| Trigger: | User triggers use case by clicking 'Search For Schools' button | | |
| Frequency of Use: | Very often since users will likely check for chance of admission at multiple schools | | |
| Preconditions | | | |
| User must be logged in and have profile information. | | | |

| Basic Flow | | |
|------------|---|--|
| Step | User Actions | System Actions |
| 1 | User clicks on "Search for Schools" button. | System displays search page |
| 2 | User enters a school and presses "Submit" | System searches database for information |
| 3 | | Displays school information |

| Alternate Flow | | |
|----------------|---|--|
| Step | User Actions | System Actions |
| 1 | User clicks on "Search for Schools" button. | System displays search page |
| 2 | User presses "Cancel" button | System returns to home page and does not save data |

| Error Flow | | |
|------------|---|--|
| Step | User Actions | System Actions |
| 1 | User clicks on "Search for Schools" button. | System displays search page |
| 2 | User enters a school and presses "Submit" | System validates input and finds an error |
| 3 | | Displays error message and prompts user to try again |

| Post Conditions |
|---|
| Application displays school information or error page |

| Includes or Extension Points |
|--|
| Extends: <ul style="list-style-type: none"> a. Error b. Cancel |

| Special Requirements |
|----------------------|
| N/A |

| Buisness Rules |
|----------------|
| N/A |

| Other Notes |
|---|
| This use case allows a user to search for schools |

| | | | |
|----------------------------|--|------------------------|---|
| Use Case ID: | Classifier.VerifyInfo.1 | | |
| Use Case Name: | VerifyInfo | Version No: | 1 |
| End Objective: | Ensure the user has entered valid info | | |
| Created by: | University Classifier | On (date): | |
| Last Update by: | | On (date): | |
| Approved by: | | On (date): | |
| User/Actor: | User | | |
| Business Owner Name | University Classifier | Contact Details | |
| Trigger: | User triggers use case by clicking 'Search For Schools' button | | |
| Frequency of Use: | Very often since it is included in all other use cases | | |
| Preconditions | | | |
| N/A | | | |

| Basic Flow | | |
|------------|--|--|
| Step | User Actions | System Actions |
| 1 | User inputs information and presses "Submit" | System searches database for information |
| 2 | | Displays requested information |

| Alternate Flow N/A | | |
|--------------------|--------------|----------------|
| Step | User Actions | System Actions |
| 1 | | |
| 2 | | |

| Error Flow | | |
|------------|--|--|
| Step | User Actions | System Actions |
| 1 | User inputs information and presses "Submit" | System searches database for information |
| 2 | | System validates input and finds an error |
| 3 | | Displays error message and prompts user to try again |

| |
|--|
| Post Conditions |
| Application displays information or error page |

| |
|-------------------------------------|
| Includes or Extension Points |
| Extends: a. Error b. Cancel |

| |
|-----------------------------|
| Special Requirements |
| N/A |

| |
|-----------------------|
| Buisness Rules |
| N/A |

| |
|-------------------------------------|
| Other Notes |
| This use case validates information |

| | | | |
|----------------------------|---|------------------------|---|
| Use Case ID: | Classifier.Cancel.1 | | |
| Use Case Name: | Cancel | Version No: | 1 |
| End Objective: | User is returned to home menu | | |
| Created by: | University Classifier | On (date): | |
| Last Update by: | | On (date): | |
| Approved by: | | On (date): | |
| User/Actor: | User | | |
| Business Owner Name | University Classifier | Contact Details | |
| Trigger: | User is returned to home menu | | |
| Frequency of Use: | Not often; whenever a user decides to not follow through which what they clicked on | | |

| Preconditions |
|---|
| User is in a use case with cancel functionality |

| Basic Flow | | |
|-------------------|---|--|
| Step | User Actions | System Actions |
| 1 | User inputs information and changes their mind, so presses "Cancel" | System validates, updates and saves information |
| 2 | | System returns to home page and does not save data |

| Alternate Flow N/A | | |
|---------------------------|---------------------|-----------------------|
| Step | User Actions | System Actions |
| 1 | | |
| 2 | | |

| Error Flow N/A | | |
|-----------------------|---------------------|-----------------------|
| Step | User Actions | System Actions |
| 1 | | |
| 2 | | |
| 3 | | |

| Post Conditions |
|------------------------------------|
| Application returns to home screen |

| Includes or Extension Points |
|------------------------------|
| N/A |

| Special Requirements |
|----------------------|
| N/A |

| Buisness Rules |
|----------------|
| N/A |

| Other Notes |
|-------------------------------------|
| This use case validates information |

b. Class Diagram

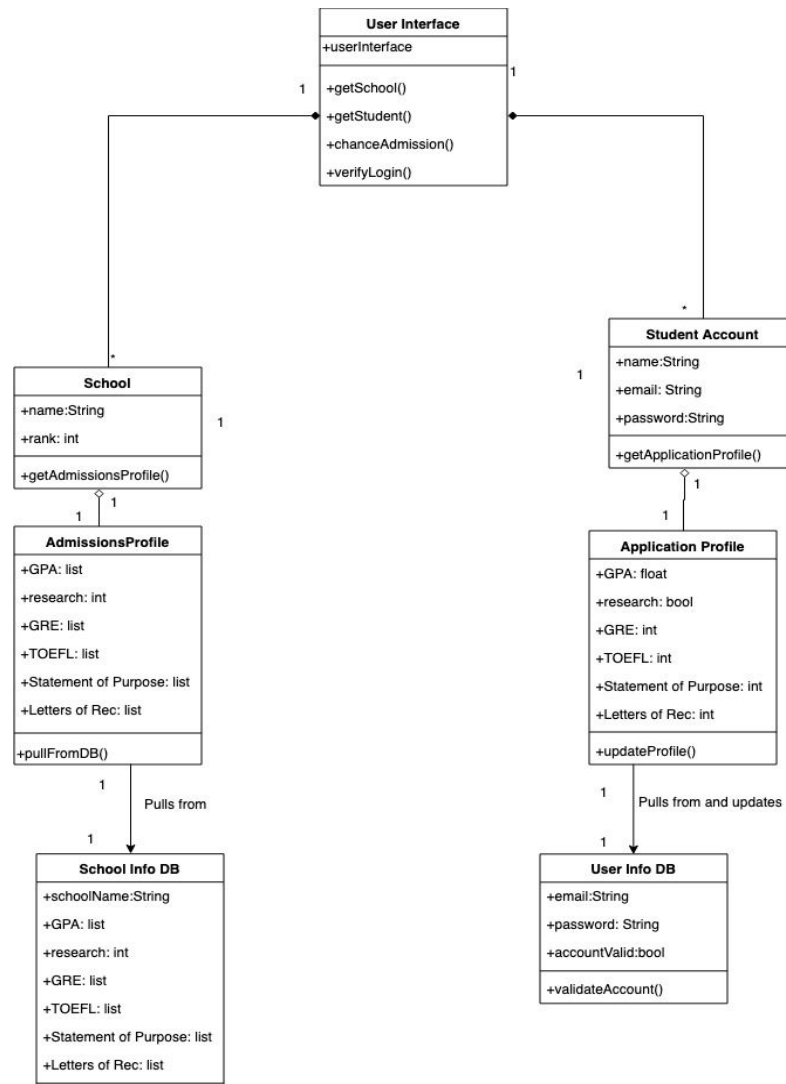


Figure 17 - Class diagram

c. Gantt Chart

| TASK NAME | START DATE | END DATE | DAYS INTO PROJECT* | DURATION* (WORK DAYS) | TEAM MEMBER | PERCENT COMPLETE | DAYS COMPLETE* | DAYS REMAINING* |
|--|------------|----------|--------------------|-----------------------|-------------|------------------|----------------|-----------------|
| Documentation Work | | | | | | | | |
| Proposal | 1/8 | 1/15 | 8 | 7 | All | 100% | 7 | 0 |
| Software Requirements Specification | 1/16 | 2/14 | 16 | 30 | All | 100% | 30 | 0 |
| Class Diagram | 2/1 | 2/21 | 31 | 21 | All | 100% | 21 | 0 |
| Sequence Diagram | 2/15 | 2/21 | 31 | 7 | All | 100% | 7 | 0 |
| State Machine | 2/15 | 2/21 | 31 | 7 | All | 100% | 7 | 0 |
| Wire Framing | 2/13 | 2/15 | 31 | 3 | All | 100% | 3 | 0 |
| Functional Requirements | | | | | | | | |
| Create university database | 2/21 | 3/2 | 51 | 10 | Marissa | 100% | 10 | 0 |
| Create an account | 2/17 | 3/14 | 47 | 26 | Jensen | 100% | 26 | 0 |
| Log into account | 2/17 | 3/14 | 47 | 26 | Jensen | 100% | 26 | 0 |
| Update profile | 3/3 | 3/14 | 63 | 12 | Jensen | 100% | 12 | 0 |
| Input scores and recieve list | 3/3 | 3/14 | 63 | 12 | Stockton | 100% | 12 | 0 |
| Search for Specific School | 3/15 | 3/30 | 75 | 16 | Jensen | 100% | 16 | 0 |
| Cancel | 3/15 | 3/30 | 75 | 16 | Jensen | 100% | 16 | 0 |
| Error page | 3/15 | 3/30 | 75 | 16 | Jensen | 100% | 16 | 0 |
| Testing | | | | | | | | |
| Test creating an account | 4/1 | 4/11 | 91 | 11 | Felix | 100% | 11 | 0 |
| Test logging into the created account | 4/1 | 4/11 | 91 | 11 | Sumer | 100% | 11 | 0 |
| Test updating profile | 4/1 | 4/11 | 91 | 11 | Marissa | 100% | 11 | 0 |
| Test inputing scores | 4/1 | 4/11 | 91 | 11 | Marissa | 100% | 11 | 0 |
| Check accuracy of result list | 4/1 | 4/11 | 91 | 11 | Stockton | 100% | 11 | 0 |
| Test searching for school | 4/1 | 4/11 | 91 | 11 | Felix | 100% | 11 | 0 |
| Check accuracy of school results | 4/1 | 4/11 | 91 | 11 | Sumer | 100% | 11 | 0 |
| Test error functionality | 4/1 | 4/11 | 91 | 11 | Marissa | 100% | 11 | 0 |
| Test cancel functionality | 4/1 | 4/11 | 91 | 11 | Felix | 100% | 11 | 0 |
| Test response time | 4/1 | 4/11 | 91 | 11 | Stockton | 100% | 11 | 0 |
| Test system crash does not result in data loss | 4/1 | 4/11 | 91 | 11 | Felix | 100% | 11 | 0 |

Figure 18 - The Gantt Chart

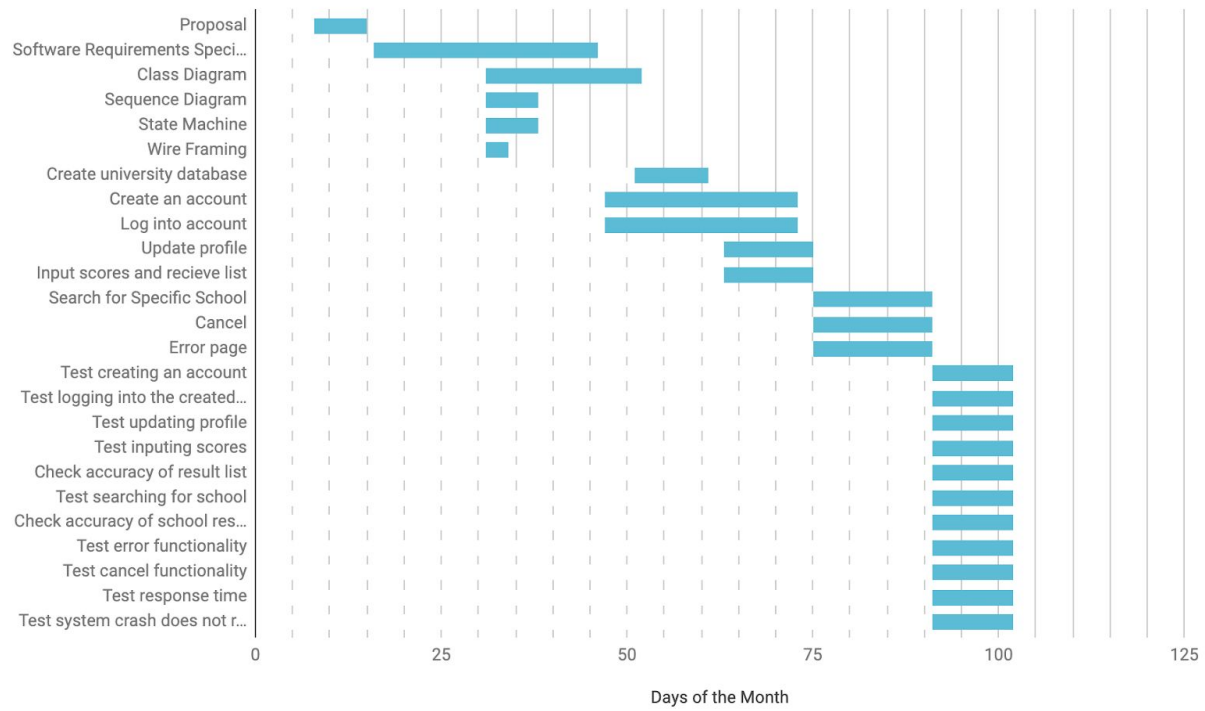


Figure 19 - A graphical representation of the Gantt chart

d. Subsystem

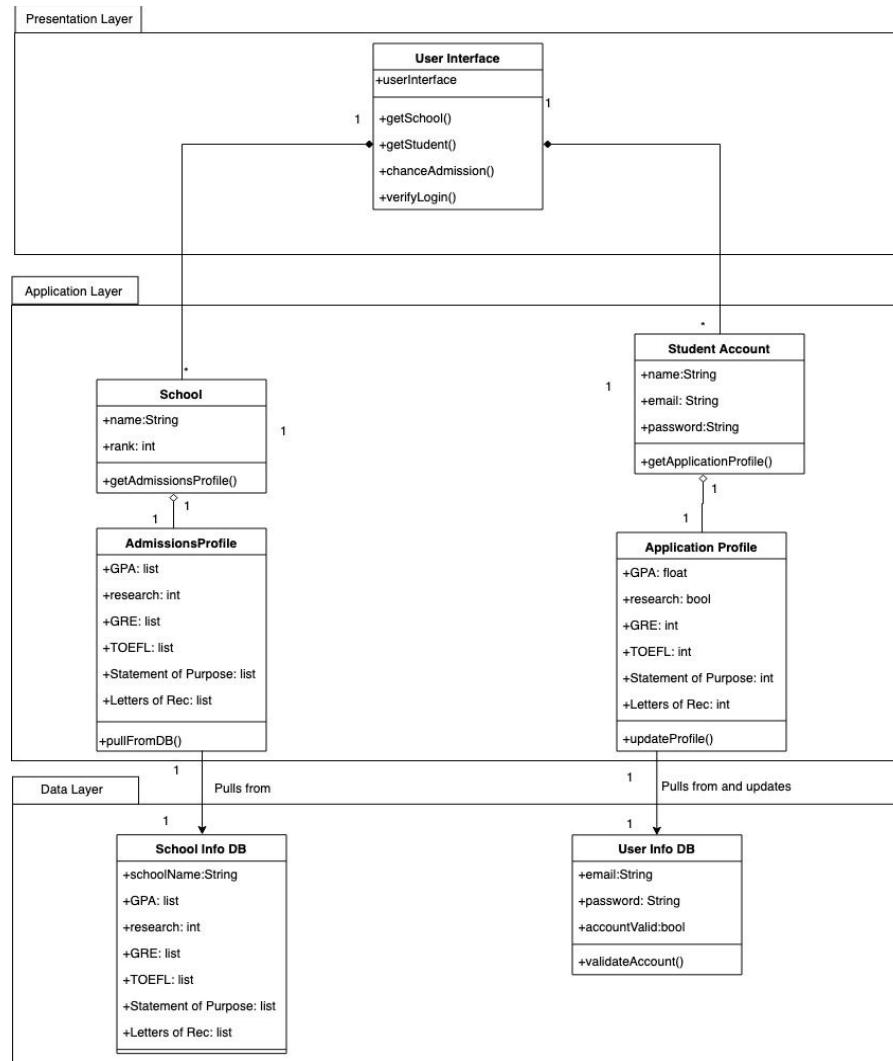


Figure 20 - The subsystem decomposition of the system

e. State Machine

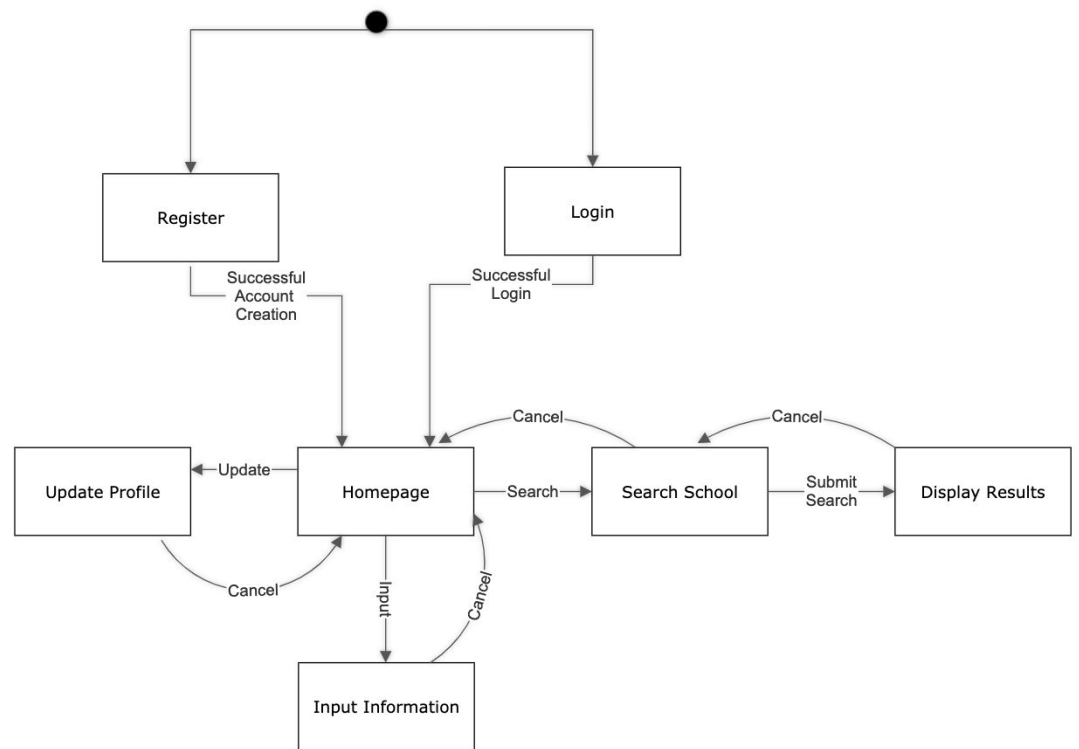


Figure 21 - State chart

f. Sequence Diagram

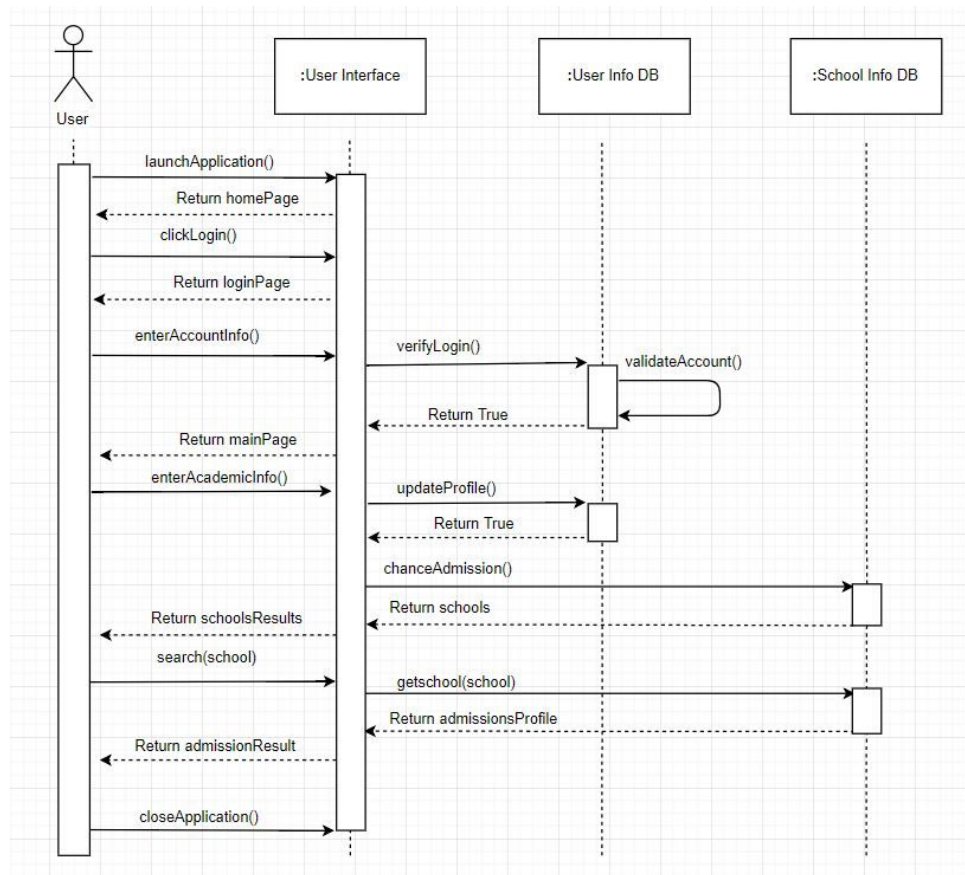


Figure 22 - Sequence Diagram

g. Extra Wire Frames

The wireframe shows a web browser window with the title 'Register'. The address bar displays the URL 'https://universityclassifier.com/register'. The main content area contains a registration form with the following fields:

- First Name
- Last Name
- Email
- Password

A 'Register' button is located at the bottom right of the form. The background of the browser window features a scenic image of a park with trees and a path.

Figure 23 - The register page

Figure 23 shows a web browser window titled "Edit Profile". The address bar displays "https://universityclassifier.com/profile". The page features a navigation bar with "Edit Profile", "Input Information", and "Log Out" buttons, along with a search bar. The main content area is titled "Edit Profile" and contains the following form fields:

- First Name: John
- Last Name: Smith
- Email: john.smith@gmail.com
- Password: password

At the bottom of the form are "Cancel" and "Save Changes" buttons.

Figure 24 - The edit profile page

Figure 24 shows a web browser window titled "Register". The address bar displays "https://universityclassifier.com/register". The page features a navigation bar with "Edit Profile", "Input Information", and "Log Out" buttons, along with a search bar. The main content area is titled "Register" and contains the following form fields:

- First Name: John
- Last Name: Smith
- Email: johnsmith@g
- Password: password

An error message dialog box is displayed in the center of the screen, stating: "There is an error with the information entered, please try again". The dialog box has an "OK" button. At the bottom right of the form is a "Register" button.

Figure 25 - The error message

