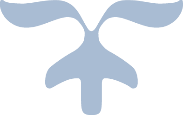


**College for me**

Software Design Specification





Contents

[1. Introduction 2](#_30j0zll)

[2. Use Case 3](#_1fob9te)

[3. Use case Diagram 13](#_3znysh7)

[4. Other requirements 15](#_2et92p0)

[5. System Architecture 17](#_tyjcwt)

[6. Class Model 18](#_3dy6vkm)

[7. Graphical User Interface and System Flow 19](#_1t3h5sf)

[8. Languages, Technologies, and Tools 35](#_2s8eyo1)

[9. Algorithm 36](#_17dp8vu)

[10. Dynamic model 41](#_26in1rg)

# Introduction

C4me helps students decide where to apply for college.  Its functionality is similar to several successful commercial software systems such as Naviance (licensed by many high schools) and commercial websites such as CollegeData.com (a service provided by 1st Financial Bank).  c4me is for students planning to get a Bachelor's degree.  An interesting challenge, beyond the scope of this project, is to make a flexible system that can easily be specialized to be suitable for students looking for different kinds of schools and degrees: undergraduate school, graduate school, law school, medical school, etc.[1]

* 1. Scope

The project goal is for users to easily find the possible colleges they should apply to by referencing other students’ information or recommender.

# Use Case

Overview of Use-Case

|  |  |  |
| --- | --- | --- |
| **Use Case** | **UI Context** | **Use Case** |
| 2.1 | Register Screen | Create new account |
| 2.2 | Register Screen | Validation check |
| 2.3 | Login Screen | Login to Account |
| 2.4 | Multiple Screens | Logout of Account |
| 2.5 | Multiple Screen | Select profile to view |
| 2.6 | Profile Screen | Change the profile |
| 2.7 | College List Screen | Search for colleges |
| 2.8 | College List Screen | Filters for search |
| 2.9 | College List Screen | Sorting for search results |
| 2.10 | College List Screen | View college information |
| 2.11 | College List Screen | College Recommender |
| 2.12 | College List Screen | Find Similar High School |
| 2.13 | Student List Screen | Get list of matching student profiles |
| 2.14 | Student List Screen | Get scatterplot of matching student profiles |
| 2.15 | Administrator Screen | Scrape College Rankings |
| 2.16 | Administrator Screen | Import College Scorecard Data |
| 2.17 | Administrator Screen | Review questionable acceptance decisions. |
| 2.18 | Profile Screen | Validate the profile |
| 2.19 | Profile Screen | Invalidate the profile |

|  |  |
| --- | --- |
| **Student** | This student can create a profile, view/edit their profile, find similar profile students, get college list |
| **Administrator** | This administrator can check the student profile, remove questionable mark, confirm questionable mark, update college rank info, update college list, update college info, delete student profile, update student profile. |

Note that there are two different actors who will make use of our application:

* 1. **Create account**

|  |  |
| --- | --- |
| Use-Case: | Create account |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | Allows new student to create account by choosing their username and password |
| Preconditions: | Database has been created and all suitable credentials have been setup |
| Trigger: | Student registers the account in Register Screen |
| Workflow: | 1. Include student input validation use case 2. Student creates account successfully |
| Exceptions: | Cannot pass invalidation check |
| Priority: | Essential, must be implemented |

* 1. **Username and password validation**

|  |  |
| --- | --- |
| Use-Case: | Username and password validation |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | Validates username and password that users choose |
| Preconditions: | User have not logged in |
| Trigger: | User enter username and password when creating account |
| Workflow: | 1. User enters username and password while creating account 2. System validates password and username |
| Exceptions: | Validation pass but database throw error |
| Priority: | Essential, must be implemented |

* 1. **Login to account**

|  |  |
| --- | --- |
| Use-Case: | Login to account |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | Login |
| Preconditions: | User is not logged in |
| Trigger: | User presses login button or user clicks view/edit profile but is not logged in to their account |
| Workflow: | 1. Include validate credentials 2. User enters a valid account with corresponding password 3. User logged in the system |
| Exceptions: | * 1. User input empty email or password   2. User input wrong email or password |
| Priority: | Essential, must be implemented |

* 1. **Logout of account**

|  |  |
| --- | --- |
| Use-Case: | Logout of account |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | Student wants to logout |
| Preconditions: | Student is logged in |
| Trigger: | Student clicks the logout choice after hover at profile icon |
| Workflow: | * + 1. Logout of account     2. User returns to search college list screen |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |

* 1. **View profile**

|  |  |
| --- | --- |
| Use-Case: | View profile |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | Allows existing user to view and edit all information in his/her profile, check the statistical consistency |
| Preconditions: | Student has login |
| Trigger: | User is logged in |
| Workflow: | Redirect to the profile screen and load the profile |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |

* 1. **Edit profile**

|  |  |
| --- | --- |
| Use-Case: | Edit profile |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | Allows existing user to edit all information in the profile, if acceptance decisions are questionable then they would be ignored by the system. |
| Preconditions: | User is logged in |
| Trigger: | User chooses any profile information he wants to edit |
| Workflow: | 1. User changes profile information 2. User saves his/her profile 3. Include System checks for statistical consistency between decision and information that he/she saved 4. Questionable decision will be ignored and prompt the student about it 5. Reasonable decision will be saved to the system. |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |

* 1. **Search for colleges**

|  |  |
| --- | --- |
| Use-Case: | Search for colleges |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | Searching colleges by applying filters, the user can choose lax or strict interpretation for filters. |
| Preconditions: | N/A |
| Trigger: | User is logged in |
| Workflow: | 1. User chooses lax or strict interpretation for filters 2. User enters desired information for each filter 3. User clicks search and system proceed the request 4. System displays the list of the results 5. User could sort this list by name, admission, rate, cost of attendance, and ranking 6. User can modify the filters and re-run the search |
| Exceptions: | No college list |
| Priority: | Essential, must be implemented |

* 1. **Filter college**

|  |  |
| --- | --- |
| Use-Case: | Filter college |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | User filters search results by their desired attributes such as location, cost, admission rate, ranking, etc |
| Preconditions: | User is logged in and got search college results |
| Trigger: | User click the filter icon in the college list screen |
| Workflow: | 1. User just got search results from a search or is about to search 2. User clicks the filter button 3. User checks all desired filters 4. User presses search 5. Colleges with the applied filters are displayed |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |

* 1. **Sort Colleges**

|  |  |
| --- | --- |
| Use-Case: | Sort Colleges |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | User sorts search results by their desired attributes such as lowest-highest cost, closest-furthest distance, closest scores, etc |
| Preconditions: | User is logged in and got search college results |
| Trigger: | User use the sort icon in college list screen |
| Workflow: | 1. User just got search results from a search  2. User clicks the “sort by” button  3. User selects a sort option  4. A list of colleges sorted by the desired format is returned |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |

* 1. **View college information**

|  |  |
| --- | --- |
| Use-Case: | View college information |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | User wants to see more details about a college |
| Preconditions: | User is logged in and got search college results |
| Trigger: | User selects a college in the list |
| Workflow: | 1. User selects a college 2. Change to college information Screen 3. Show the college information |
| Exceptions: | Incomplete college info |
| Priority: | Optional |

* 1. **College Recommender**

|  |  |
| --- | --- |
| Use-Case: | College Recommender |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | Student can ask the system to compute a recommendation score for each college in the search results, and sort the list by that score. The recommendation is based on where students with similar profiles applied and possibly other information. |
| Preconditions: | User is logged in |
| Trigger: | Student clicks the college recommend button at college list screen |
| Workflow: | 1. User just got search results from a search 2. User clicks the recommend button 3. A recommend score appears next to search results 4. Results automatically sorted by highest recommendation score |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |

* 1. **Find Similar High School**

|  |  |
| --- | --- |
| Use-Case: | Find Similar High School |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | Helps student identify a list of high schools similar to their own |
| Preconditions: | User is logged in |
| Trigger: | Student clicks the similar high school button in the college list screen |
| Workflow: | 1. User enters the name of a high school 2. User clicks the enter button 3. A list of similar high schools appears |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |

* 1. **Get list of matching student profiles**

|  |  |
| --- | --- |
| Use-Case: | Get list of matching student profiles |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | User selects college and searches for student profiles from this college |
| Preconditions: | User is logged in |
| Trigger: | User clicks the find matching student in college search screen |
| Workflow: | 1. User selects a college 2. System prompts user to select filters 3. User selects which filters to use 4. System gets list of matching profiles from database 5. System removes questionable profiles from the list 6. System displays all the profiles to the user |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |

* 1. **Get scatterplot of matching student profiles**

|  |  |
| --- | --- |
| Use-Case: | Get scatterplot of matching student profiles |
| Primary Actor: | Student |
| Secondary Actor | Database |
| Brief Description: | User selects college and views a scatterplot for student profiles from this college |
| Preconditions: | User is logged in and viewing matching student list |
| Trigger: | User selects the view scatterplot option |
| Workflow: | The normal flow of events is the same as in Use Case 2.13, except step 6 is replaced with:  6.  User selects scatterplot option  7.  System generates a scatterplot of the matching profiles, and displays it to the user |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |

* 1. **Scrape College Rankings**

|  |  |
| --- | --- |
| Use-Case: | Scrape College Rankings |
| Primary Actor: | Administrator |
| Secondary Actor | Database |
| Brief Description: | Administrator updates college rankings for the system |
| Preconditions: | Administrator has logged in |
| Trigger: | Administrator presses button to update college rank file |
| Workflow: | 1. Administrator presses button to update college rankings 2. System scrapes data from The Wall Street Journal webpage and updates rankings in database 3. System notifies administrator of successful update |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |

* 1. **Import College Scorecard Data**

|  |  |
| --- | --- |
| Use-Case: | Import College Scorecard Data |
| Primary Actor: | Administrator |
| Secondary Actor | Database |
| Brief Description: | Administrator updates college scorecard data files |
| Preconditions: | Administrator has logged in |
| Trigger: | Administrator presses button to update college scorecard |
| Workflow: | 1. Administrator presses button to import College Scorecard Data 2. System scrapes data from the College Scorecard webpage and updates college information in system 3. System notifies administrator of successful update |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |

* 1. **Review questionable acceptance decisions**

|  |  |
| --- | --- |
| Use-Case: | Review questionable acceptance decisions |
| Primary Actor: | Administrator |
| Secondary Actor | Database |
| Brief Description: | The system displays student profiles with questionable acceptance decisions. |
| Preconditions: | Administrator has logged in |
| Trigger: | Administrator reviews the questionable user list |
| Workflow: | * 1. Administrator get authorization   2. Administrator queries questionable profile list   3. Change to Student list Screen |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |

* 1. **Validate the profile**

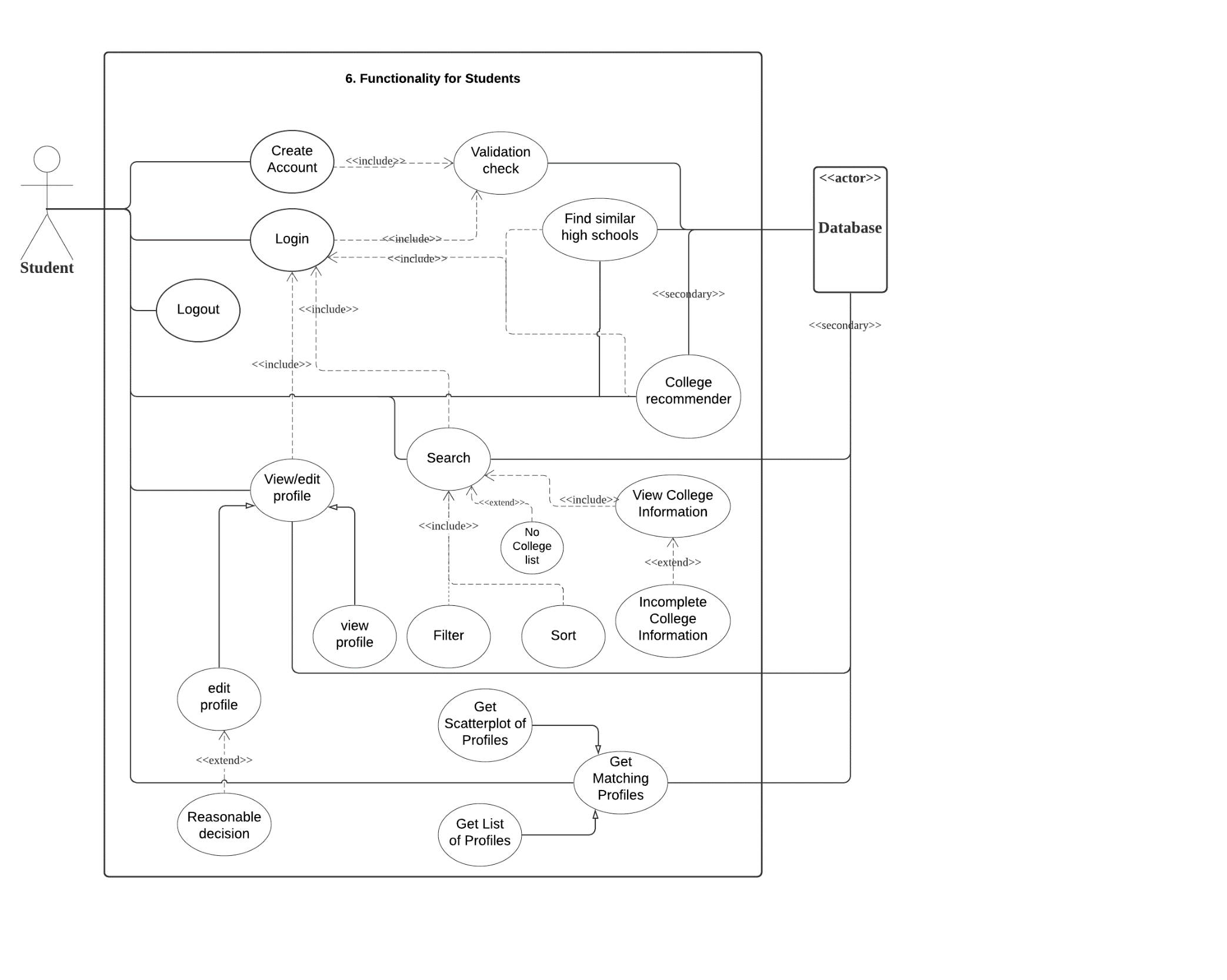
|  |  |
| --- | --- |
| Use-Case: | Validate the profile |
| Primary Actor: | Administrator |
| Secondary Actor | Database |
| Brief Description: | The administrator validates some questionable acceptance decisions, so that they are no longer marked as questionable. |
| Preconditions: | Administrator has logged in and gets the questionable profile list |
| Trigger: | Administrator clicks the questionable student profile he wants to review |
| Workflow: | 1. Change to profile page 2. Load student profile 3. Administrator mark the profile as valid 4. Database delete the “questionable” mark |
| Exceptions: | Students input empty name, password, email |
| Priority: | Essential, must be implemented |

* 1. **Invalidate the profile**

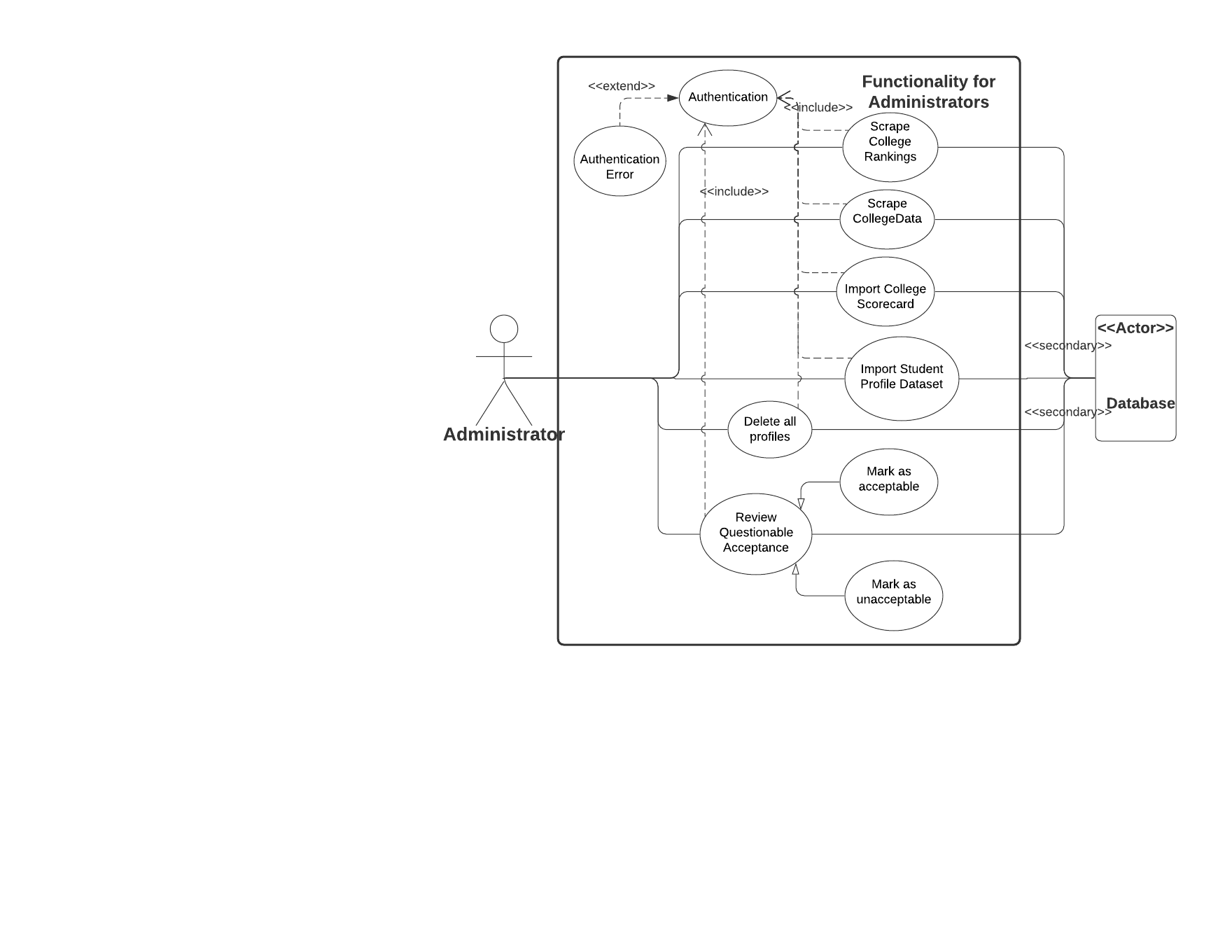
|  |  |
| --- | --- |
| Use-Case: | Invalidate the profile |
| Primary Actor: | Administrator |
| Secondary Actor | Database |
| Brief Description: | The administrator invalidates some questionable acceptance decisions, so that the profile will be marked as invalid. |
| Preconditions: | Administrator has logged in and get the questionable profile list |
| Trigger: | Administrator clicks the questionable student profile he wants to review |
| Workflow: | 1. Change to profile page 2. Load student profile 3. Administrator mark the profile as invalid 4. Database change the “questionable” to “invalid” mark 5. Database change profile attribute validation to invalid, so that student will know they need to modify their profile or give a feedback |
| Exceptions: | Students input empty name, password, email |
| Priority: | Essential, must be implemented |

# Use case Diagram

* 1. **Student use case diagram**



* 1. Administrator use case diagram



# Other requirements

The College for me application will require a number of user interface screens including:

* Login Screen
* Register Screen
* College list Screen
* College detail Screen
* Student list Screen
* Profile Screen
* Administrator Screen
  1. **Performance requirements**

**Authentication**

All access to the system, [2020-01-30 added following phrase] except for student account creation, requires authentication with a password.  Passwords stored in the system database are salted and hashed.  It is sufficient to have one pre-defined administrator account, with a fixed username and password.  The system does not need to support creation of additional administrator accounts.

**Concurrency**

Synchronization is used to ensure correct behavior when multiple users access the system concurrently.

**Configuration**

It should be easy (e.g., by editing one line in one file) to change the hostname and path prefix of URLs for all data source websites, for example, to change from https://CollegeData.com/ to https://www.cs.stonybrook.edu/CollegeData/. I might create partial mirrors of some data source websites, to ensure availability and consistency of the content during demos.

**Ease of use**

The system provides an easy-to-use, user-friendly web interface consistent with established UI design principles.  The system handles invalid inputs gracefully, provide helpful feedback to the user when appropriate, etc.

**Multi-host operation**

The client and server can run on different hosts.

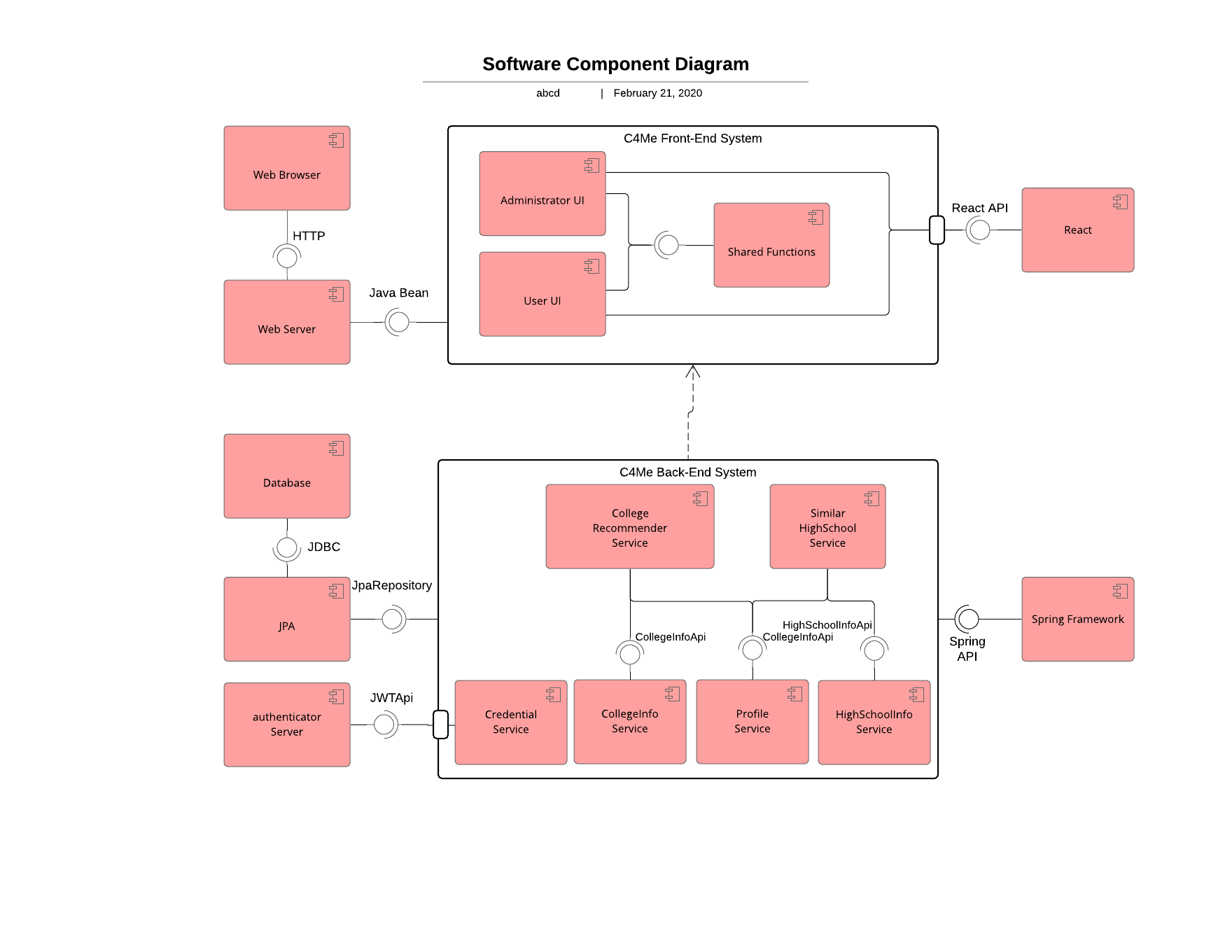
**Network security**

Network communication is secured using HTTPS or SSL.  If your server does not have a public-key certificate signed by a certification authority trusted by the web browser, the web browser will show a security warning.  The warning can be eliminated by creating a self-signed certificate, and installing the key in the browser, but that is optional.  Requiring the user to tell the browser to proceed despite the security warning is acceptable.

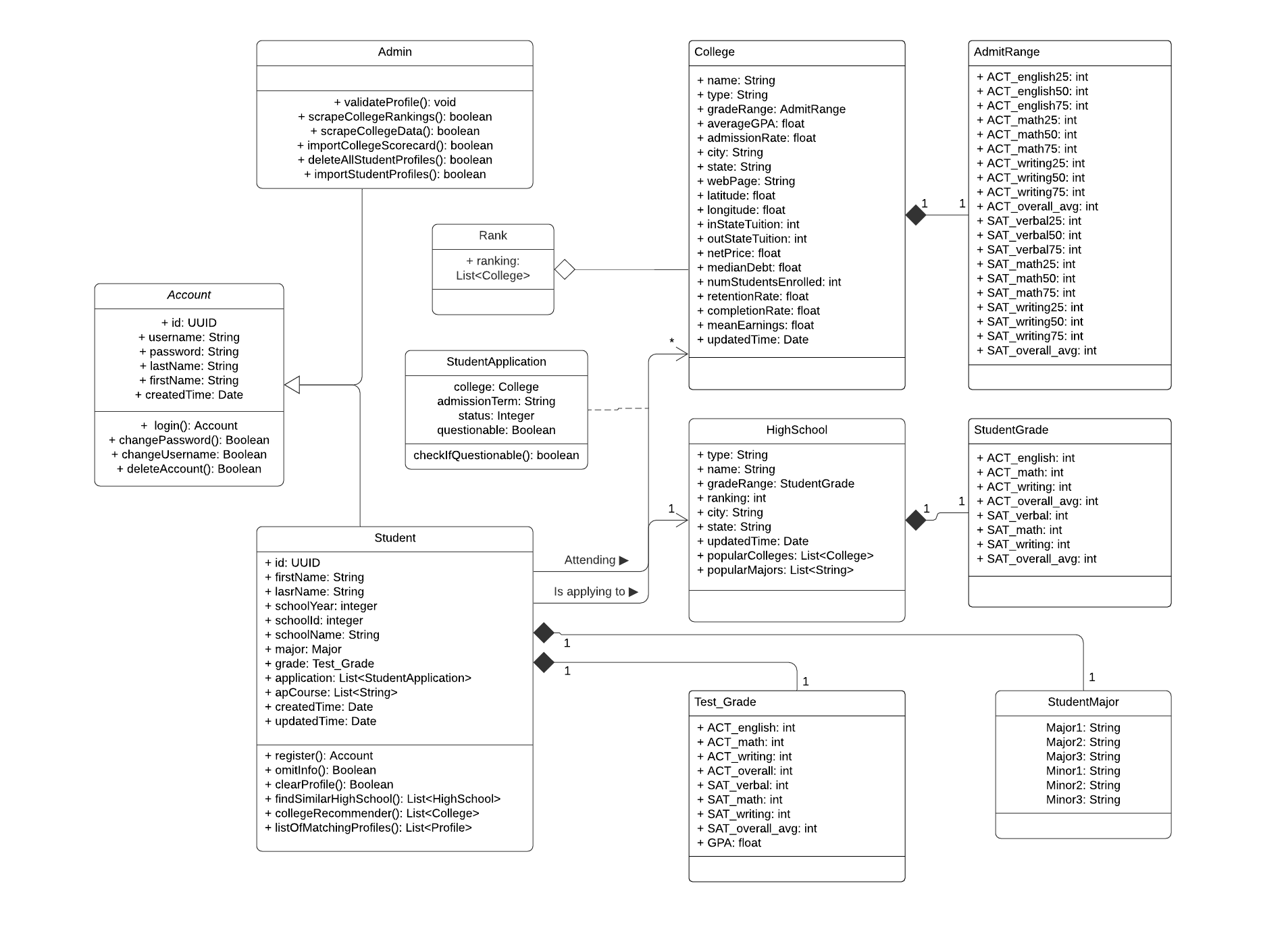
**Robustness**

The system handles missing information gracefully.  For example, almost all information in a student profile is optional (students should omit information they don't want to share, since all information in the profile is public), and some information about colleges is missing on CollegeData.com (it's shown as "Not reported").  The system handles unreachable data sources gracefully, for example, if CollegeData.com is unreachable when scraping, or niche.com is unreachable when querying high school information. The system is robust to changes in the layout of scraped web pages, provided the part of direct interest is unchanged. The system can assume College Scoarecard data files contain valid data. The system thoroughly validates data entered by students: high schools, colleges, test scores, etc. [1]



1. **System Architecture** 

# Class Model



# Graphical User Interface and System Flow

7.1**图片包含 屏幕截图

描述已自动生成**

Description:

1. Once the user enters correct username and password and clicks **Login** button, screen would switch to **Home Screen**
2. When user clicks **Create account**, screen would switch to **Create account Screen**
3. When user clicks **Forgot your password**, screen would switch to **Find account Screen**

7.2

图片包含 屏幕截图

描述已自动生成

Description:

1. When user clicks **My Profile** on nav bar, screen would switch to **My Profile Screen**

2. When user clicks **Find Similar High School** on nav bar, screen would switch to **Find Similar High School Screen**

3. When user clicks **College Search** on nav bar, screen would switch to **Search Screen**

4. When user clicks **Log out** on nav bar, screen would switch to **Login Screen**

7.3

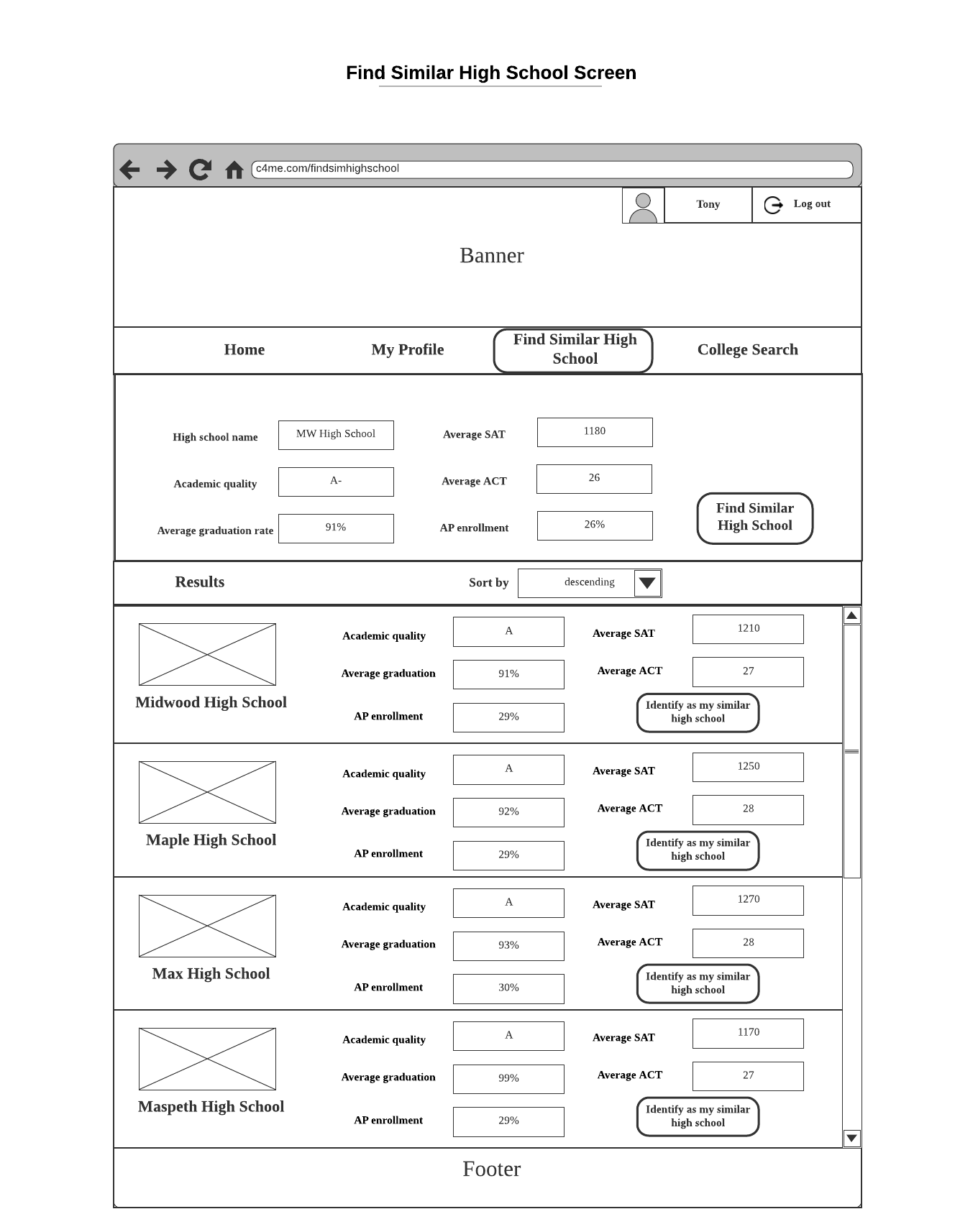
图片包含 屏幕截图

描述已自动生成

Description:

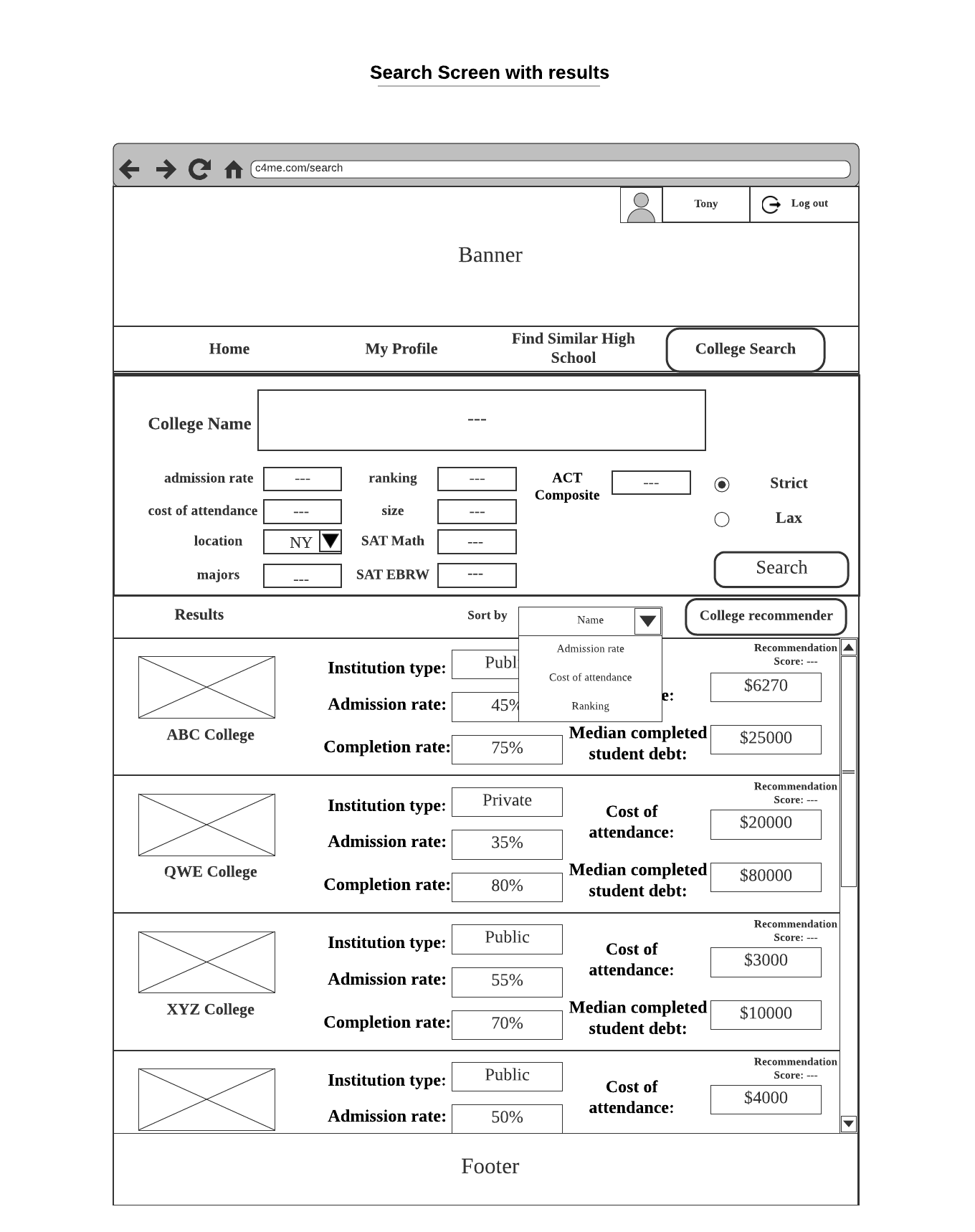
1. When user clicks **Edit profile** button, **School, Academic profile, Ap course, Application** sections become editable, and **Save profile** button becomes clickable.
2. When user clicks **Save profile** button, System would save all data have changed.
3. When user clicks **Home** on nav bar, System would switch screen to **Home Screen**.
4. When user clicks **Find Similar High School** on nav bar, System would switch screen to **Find Similar High School Screen**.
5. When user clicks **College Search** on nav bar, System would switch screen to **College Search Screen**.
6. When user clicks **log out** button, system would switch back to **Login** **Screen**.

7.4



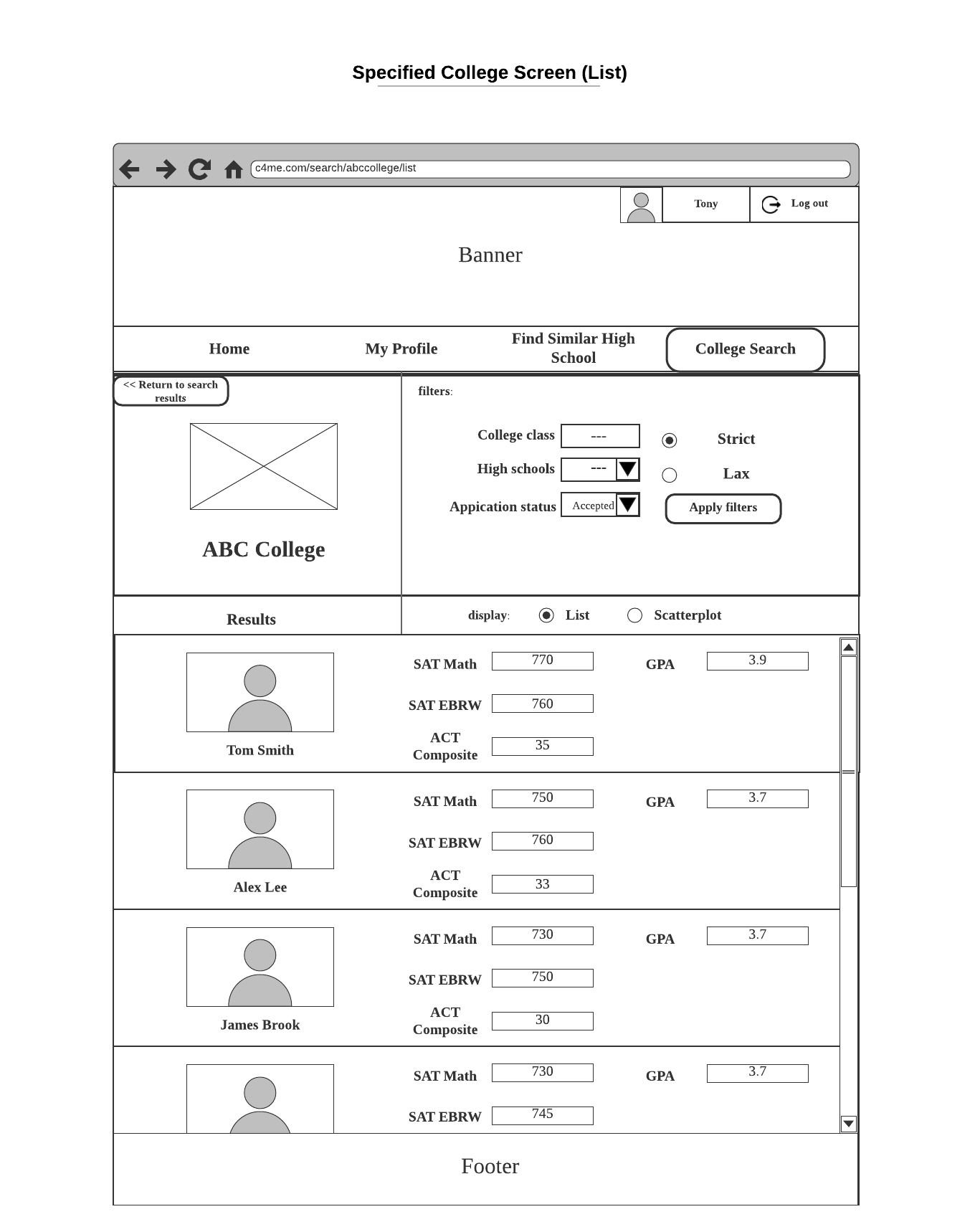
Description:

1. When user clicks **Home** on nav bar, system would switch screen to **Home Screen**.
2. When user clicks **My Profile** on nav bar, system would switch screen to **My Profile Screen**.
3. When user clicks **College Search** on nav bar, system would switch screen to **Search Screen**.
4. When user enters data for some fields for his/her high school and clicks **Find Similar High School** on nav bar, system would **display the list of similar high schools in descending order**.
5. When user clicks **Identify as my similar high school** button on one of the tab, system would change the similar high school field in user profile to the one that listed in the tab
6. When user clicks **log out** button, system would switch back to **Login** **Screen**.

7.5

Description:

1. When user clicks **Home** on nav bar, system would switch screen to **Home Screen**.
2. When user clicks **My Profile** on nav bar, system would switch screen to **My Profile Screen**.
3. When user clicks **Find Similar High School** on nav bar, system would switch screen to **Find Similar High School Screen**.
4. When user enters inputs for some filters and clicks **Search** button, system would display list of colleges that fit in these criteria in **result** section.
5. User can use the toggle button to switch **Strict** or **Lax** search mode
6. User can use the dropdown list to choose sorting criteria.
7. When user clicks on **College recommender** button, system would compute and display recommendation score for each college.
8. When user clicks on one of the **college tab** on the list, system will switch screen to that **Specified College Screen**
9. When user clicks **log out** button, system would switch back to **Login** **Screen**.

7.6

Description:

1. When user clicks **Home** on nav bar, system would switch screen to **Home Screen**.
2. When user clicks **My Profile** on nav bar, system would switch screen to **My Profile Screen**.
3. When user clicks **Find Similar High School** on nav bar, system would switch screen to **Find Similar High School Screen**.
4. When user clicks **Return to search results** button, system would switch back to **Search** **Screen** with previous results.
5. When user chooses some criteria in the filter sections and clicks **Apply filters** button, system would display a list of students in those criteria on result sections.
6. When user switch the toggle to Scatterplot. system would switch to **Specified College Screen(scatterplot)** which would display appropriate students on scatterplot view.
7. When user clicks one of the **student tab** on the list, system would switch to **View Student Profile screen** with that students’ profile.
8. When user clicks **log out** button, system would switch back to **Login** **Screen**.

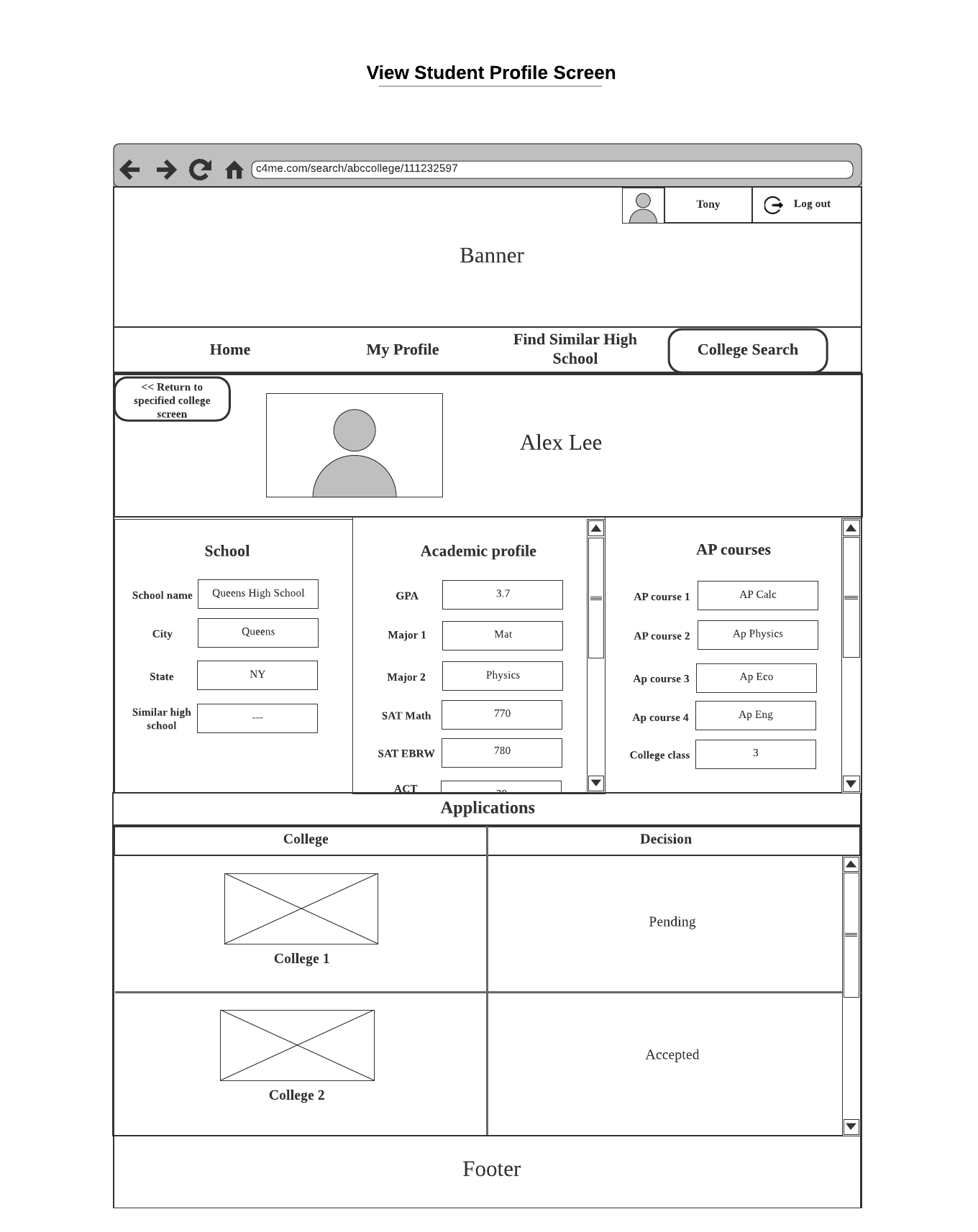
7.7

图片包含 屏幕截图

描述已自动生成

Description:

1. When user clicks **Home** on nav bar, system would switch screen to **Home Screen**.
2. When user clicks **My Profile** on nav bar, system would switch screen to **My Profile Screen**.
3. When user clicks **Find Similar High School** on nav bar, system would switch screen to **Find Similar High School Screen**.
4. When user clicks **Return to search results** button, system would switch back to **Search** **Screen** with previous results.
5. When user chooses some criteria in the filter sections and clicks **Apply filters** button, system would display a list of students in those criteria on result sections.
6. User could choose to display ACT or SAT scores by switching toggle button in the Scatterplot.
7. When user switch the toggle to Scatterplot. system would switch to **Specified College Screen(List)** which would display appropriate students on list view.
8. When user clicks **log out** button, system would switch back to **Login** **Screen**.

7.8

Description:

1. When user clicks **Home** on nav bar, system would switch screen to **Home Screen**.
2. When user clicks **My Profile** on nav bar, system would switch screen to **My Profile Screen**.
3. When user clicks **Find Similar High School** on nav bar, system would switch screen to **Find Similar High School Screen**.
4. When user clicks **return to specified college screen** button, system would switch back to **Specified College** **Screen**.

5. When user clicks **log out** button, system would switch back to **Login** **Screen**.

# Languages, Technologies, and Tools

**Language:**

* FrontEnd: JavaScript
* BackEnd: Java

**Technologies**

* Front End:
  + React
  + Ajax
* BackEnd:
  + SpringCloud
  + SpringBoot
  + Maveen
  + Hibernate
  + Gateway
  + Oauth
  + JWT
* Database
  + MySQL

**Tools:**

* UML: Lucidchart
* Code Repository: Github
* Mock data: Postman
* Monitor: Zipkin
* Java: version 1.8.241

# Algorithm

* 1. **College Recommender**

The first function we need for our college recommendation algorithm is a metric for computing the distance between two students. To do this, we first normalize each numerical attribute of both students to be in the range [0,1]. For example, a student SAT math score of 700 would normalized to 0.8333, because SAT math is on a range 200-800, and (700-200)/(800-200) = 0.8333. In general, the function for normalizing an attribute would look like this:

float normalize(attr, attr\_min, attr\_max):

return (attr-attr\_min)/(attr\_max-attr\_min)

Then we compute the distance for each of these attributes between the two students.

Next, weights are assigned to each attribute. For example, SAT scores might have a higher weight than GPA. The weights need to be adjusted for missing attributes, because a student profile may have missing information. Any attribute that is missing in either of the students is given a weight of 0.

Then, we simply compute the weighted distance between the two students, and this is our metric. In pseudocode:

float studentDistance(student1, student2):

# get normalized attributes for student1 - attr1

# get normalized attributes for student2 - attr2

# get weights

distance = 0

for i in range(attr1):

distance += weights[i] \* (attr1[i]-attr2[i])^2

distance /= (sum of weights)

distance = sqrt(distance)

return distance

Now, to compute a college recommendation score to college C for student S, we will compute a number of factors, that will be averaged together to give our final score.

The first factor is based on the popularity of C among students similar to S. First, we find the top n most similar students to the student being considered (using the metric described above). n is a number that will be adjusted based on how much student profiles we end up having in the system. Based on these profiles, we compute P(applied to college C | similar profile to S), by simply using the formula for conditional probability - count the number that applied to college C (out of the n most similar profiles) and divide by n. This number will be one factor of the college recommendation score.

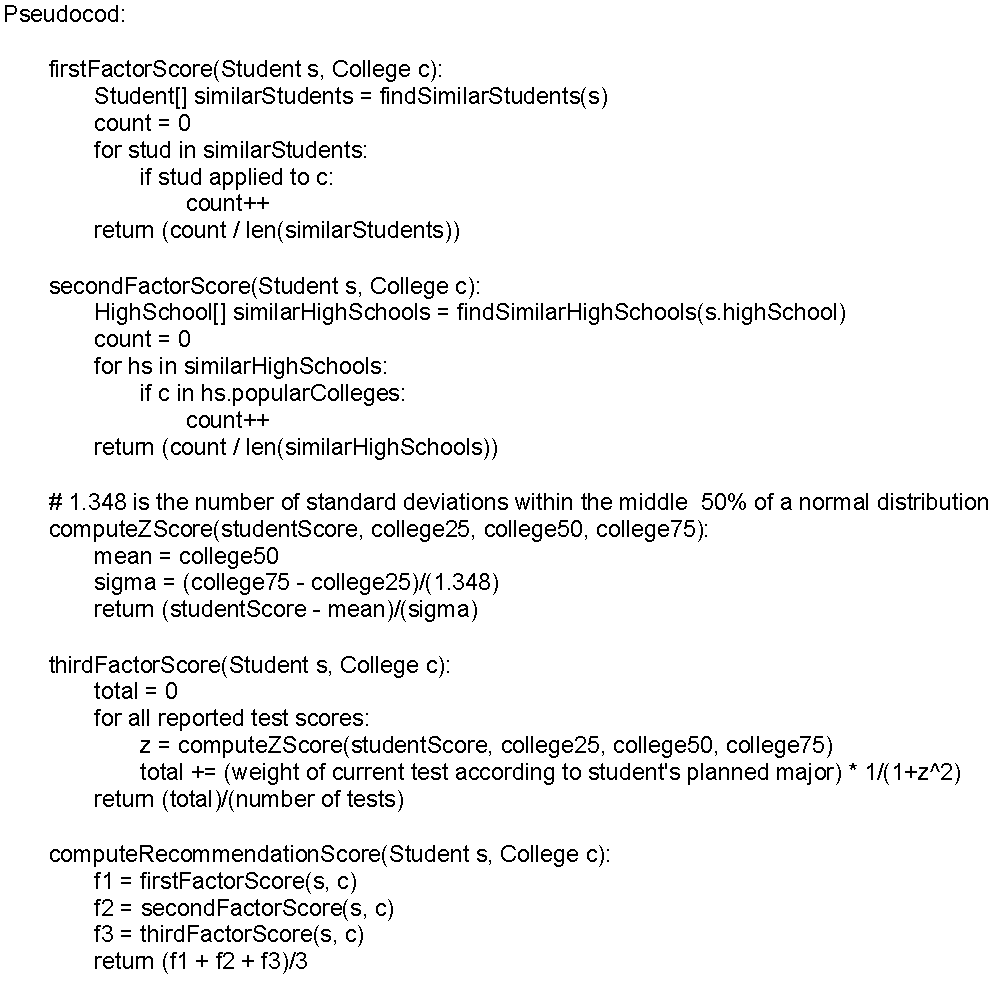
The second factor will be based on the popularity of C at the high school H (and other similar high schools) that S attends. We find the high schools most similar to S's high school, using the findSimilarHighSchools algorithm described elsewhere. Then, we compute P(C is a popular college at hs | hs is similar to H). Namely, this is the number of similar high schools where C is a popular college divided by the total number of similar high schools. This number is the second factor of the college recommendation score.

The third factor will be based on the student's test scores directly compared to the school's average scores. For each test subscore (e.g. SAT Math section), we first fit a normal distribution to the college's test scores for that section (based on the reported median, 25th, and 75th percentiles on that section for the college). Then we compute the student's z-score with respect to this distribution (and this is done for all test subscores that the student has reported). Then, the quantity

"Weighted average over all z-scores of 1/(1+z^2)"

is computed. The motive behind using 1/(1+z^2) is that it smoothly maps (-infty, infty) to (0,1], with z-scores closer to 0 getting a higher score, and z-scores further from 0 getting a lower score. We don't want to recommend a college if the student's scores are too low, but we also don't want to recommend if the student's scores are too high - this is an indication that they should aim higher. The weights that are used in this weighted average are based on the student's intended major - e.g. if they are planning to do STEM, then the math section is given higher weighting, whereas if their intended major is in the humanities, then the English section is given higher weighting.

This number is the third factor.

Finally, the three factors are averaged together to give the final college recommendation score. Note that we've carefully made sure that each factor is in the common range [0,1], so our final recommendation score is also in this range.

* 1. **Questionable Profile Detect**

**Problem:**

Student S enters his SAT/ACT score include math, verbal and writing. Then he enters he was admitted by X college.

Given:

1. Some of the student’s score(S\_math, S\_verbal, S\_AVG)
2. The college accepted score at 25th percentile and 75th percentile.

**Solve:**

Prehandle

Because the SAT has some free points for the student, we delete the free point and normalize the college admit score data. For example, the total of SAT grade range is 400 to 1600, and student A got 1300:

# (1300-400) / (1600-400) = 0.75

def normalize(SAT\_score):

return (SAT\_score - minimum\_score)/ (maximum\_score - minimum score)

Also, the math score and the verbal score should also do this separately.

After the first part, we can get student S\_AVG S\_MATH…… score after normalize.

First, compare the score with college admitted data. If all of the subject scores are higher than the college admission 25th percentile score, pass.

Second, get the list of profiles of other students in our system that were also admitted to that college.

Third, calculate the standard deviation and z-score for student S. Calculate in each subject. For example math:

def std(students):

For i in students:

variance += (i.Math-students.mathMean)^2

variance /= students.size + 1

Return sqrt(variance)

def zScore(S, students):

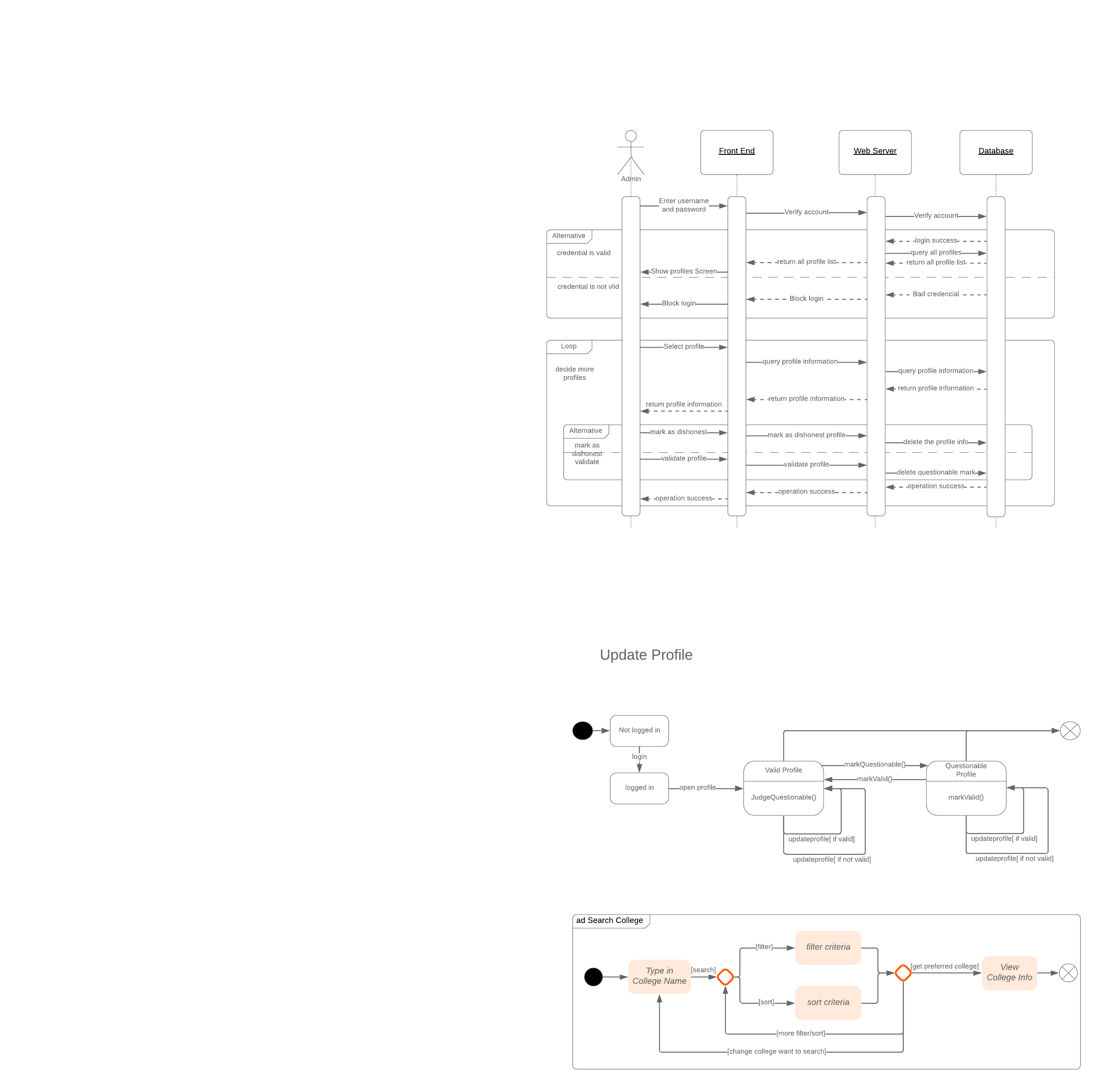
return (S.math - students.mathMean) / std(students)

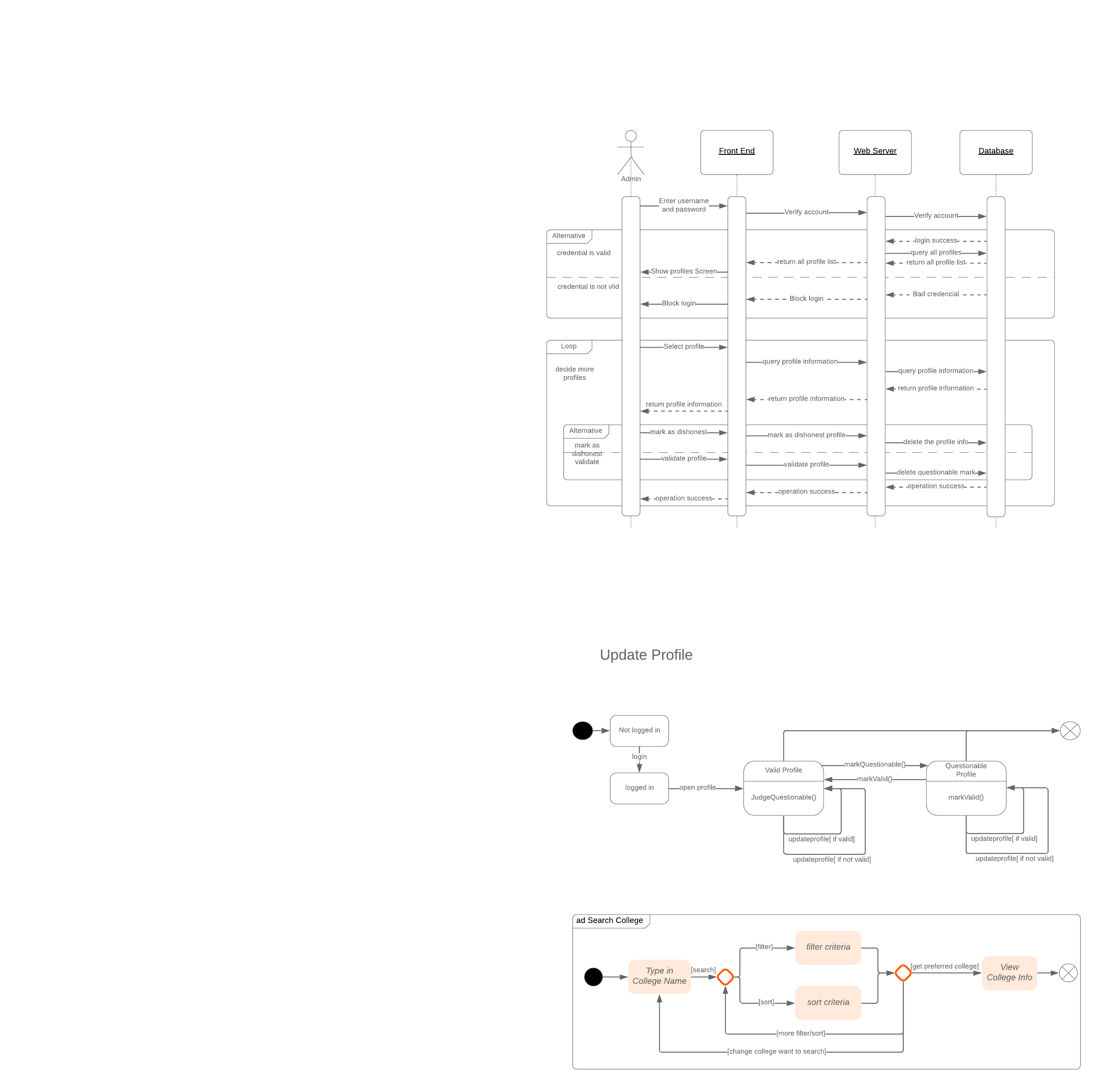
If z-score is less than -1.645, it means his math score is lower than 90% of students admitted by X college. It is possible that his acceptance is questionable.

Also, if a student was rejected from a college, yet his z-score for all tests was higher than +1.645, then we also mark his rejection as questionable.

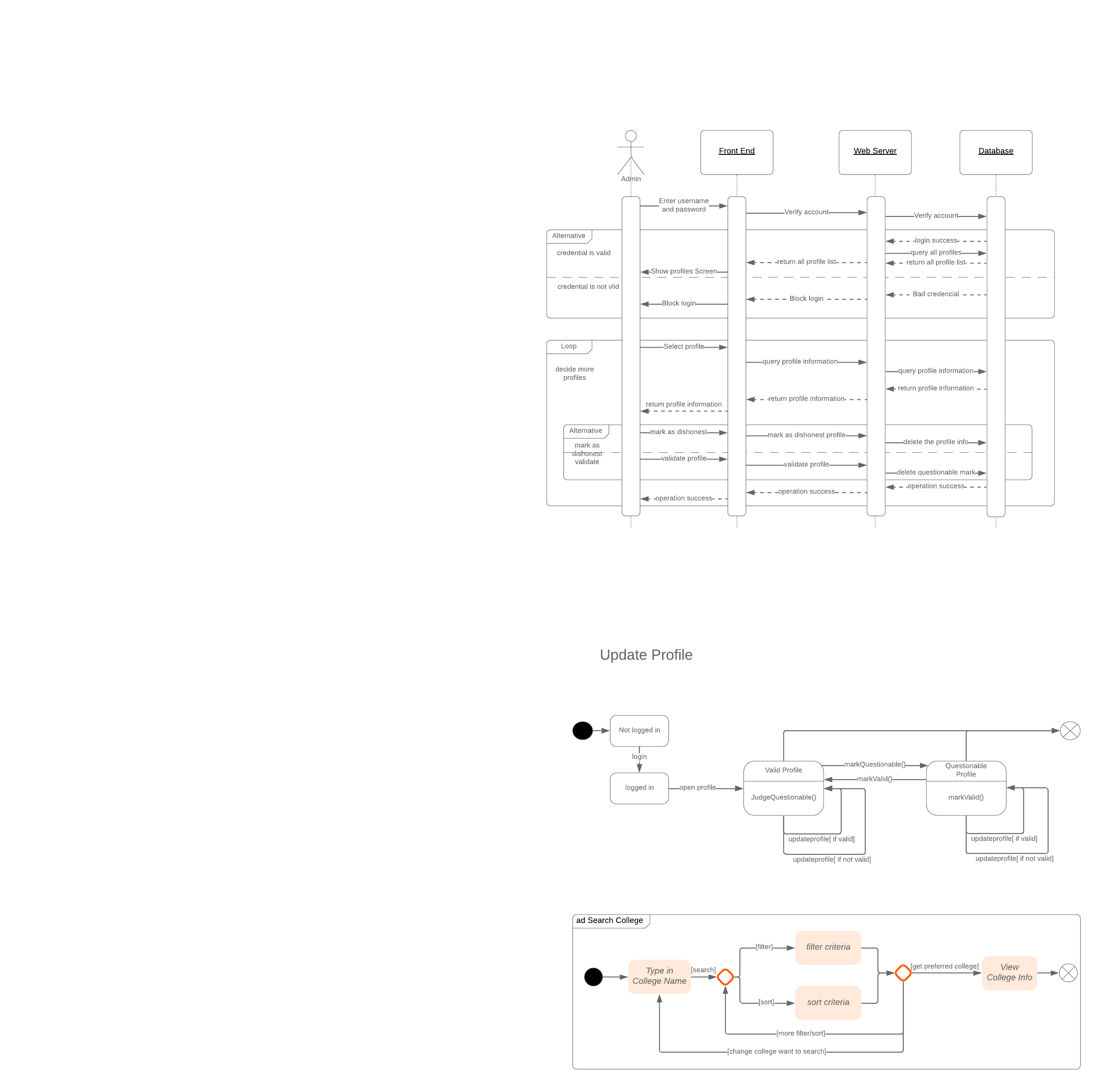
* 1. **Find similar high school**

# Dynamic model

* 1. **Questionable profile decision**
  2. **Change profile**

****

* 1. **Search College**

****