Functional Requirements

1. The system must allow users to query information about schools.

1.1. The system must allow the user to submit a query by selecting query options.

1.1.1. The system must display a list of query options.

1.1.1.1. Co-curricular activities(CCAs) offered must be a query option.

1.1.1.1.1. The system must display a list of CCAs to choose from.

1.1.1.1.2. The system must allow the user to choose from the list of CCAs.

1.1.1.2. The school type must be a query option.

1.1.1.2.1. The system must display a list of school types to choose from.

1.1.1.2.2. The system must allow the user to choose from the list of school types.

1.1.1.3. The subjects offered must be a query option.

1.1.1.3.1. The system must display a list of subjects to choose from.

1.1.1.3.2. The system must allow the user to choose from the list of subjects.

1.1.1.4. The gender of the students must be a query option.

1.1.1.4.1. The system must display mixed schools and both single-sex schools as options to choose from.

1.1.1.4.2. The system must allow the user to choose from the 3 gender options.

1.1.1.5. Distinctive programmes offered must be a query option.

1.1.1.5.1. The system must display a list of distinctive programmes to choose from.

1.1.1.5.2 The system must allow the user to choose from the distinctive programmes.

**1.1.2.5. All query options must be optional.**

1.1.3. The system must allow the user to submit a query after the user has completed selection of all query options they deem necessary.

1.2. The system must allow the user to query by entering a school name.

1.3. The system must display the results after the user’s query has been submitted.

1.3.1. The results must display a list of school(s) that fulfils the user’s query if any such school is found.

1.3.1.1. The system must provide the option of sorting the list of schools in the results in different ways.

1.3.1.2.1. The system must allow the schools in the results to be sorted alphabetically.

1.3.1.2.2. The system must allow the schools in the results to be sorted by distance from the user’s home after logging in.

1.3.3. The system must allow the user to access more information about a selected school in the results.

1.3.3.1. The information must include additional information about the selected school

1.3.3.2. The information must include the selected school’s mission statement.

1.3.3.3. The system must allow the user to view the school’s location on a map.

1.3.3.4. The information must include a link to the school’s official website.

1.3.4. The system must allow the user to save a school from the list of school(s).

1.3.4.1 The system must allow the user to save a school if they are logged into their account.

1.3.4.2 The system must prompt the user to log in or create an account if they attempt to save a school but are not logged into their account.

1. The system must allow the user to create an account.

2.1. The system must allow the user to enter a username for the account

2.1.1. The system must prevent the user from proceeding if no username is entered.

2.1.2. The system must reject the entered username if an account with the same username already exists within the system.

2.2. The system must allow the user to enter a password for the account

2.2.1. The system must prevent the user from proceeding if no password is entered.

2.2.2. The password must contain at least 8 characters.

2.2.3. The password must contain at most 12 characters.

2.2.4. The password must contain 1 digit.

2.2.5. The password must contain 1 uppercase letter.

2.2.6. The system must reject the password if the password does not adhere to the password requirements.

1. The system must allow the user to log into their account

3.1 The system must allow the user to enter a username to log in.

3.2 The system must allow the user to enter a password to log in.

3.4 The system must prevent the user from proceeding if no username is entered.

3.5 The system must prevent the user from proceeding if no password is entered.

3.3 The system must display the users list of saved schools upon log in.

3.3.1 The system must allow the user to delete a school from the list of saved schools.

3.3.2 The system must allow the user to sort the saved schools by saved school dates.

Non-functional Requirements

1. The system must enable at least 80% of first-time users to learn how to enter and submit a query within 3 minutes of starting to use the system.
2. The system must display the results within 3 seconds of a query’s submission.

Data Dictionary

|  |  |
| --- | --- |
| Term | Definition |
| School | A school is an institution for learning. |
| School Name | The name used to identify a school. |
| Postal Code | A postal code is a string of letters and/or digits that identifies a physical entity. |
| Subject | A subject is an area of study offered by a school. |
| Co-Curricular Activity (CCA) | A co-curricular activityis a programme or learning experience that complements and extends the academic curriculum. |
| School Type | A school type refers to one of the following categories: Government/Government-aided, Independent, Specialised, Boys’, Girls’, and, Co-ed. |
| Focus Area | A focus area is an area of academic specialisation assigned to a school. The school designs hands-on learning programmes related to its focus area. |
| Distance | The distance refers to that from the entered postal code to the school in question. |
| Map | A diagrammatic representation of an area of land. |

Use Case Description

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC001 | | |
| Use Case Name: | Create Account | | |
| Created By: | Germaine Kwek | Last Updated By: | Alexander Wong |
| Date Created: | 20/2/2019 | Date Last Updated: | 07/4/2019 |

|  |  |
| --- | --- |
| Actor: | User(initiating actor) |
| Description: | The user creates an account with a username and password. |
| Preconditions: | NIL |
| Postconditions: | The user has an account with a username and password.  A record is created for the user in the “Saved Schools” database. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1) User clicks on the “Create Account” button.  2) User enters a email.  3) The System verifies the “User Information” database does not already contain an account with the same email as the given email.  3) User enters a password.  4) The System verifies the password has 8-12 characters, 1 Uppercase letter and 1 Number  5) User re-enters the password.  6) The System verifies the 2 passwords match.  7) User enters postal code.  8) The user clicks the “submit” button.  9)The system writes the email, password, and postal code(valid or NULL) to a new record in the “User Information” database.  10) System prints a success message and displays the home page. |
| Alternative Flows: | AF-S3: If an account with the email already exists in the “User Information” database:   * System displays a message: “This account already exists!” for 2 seconds. * System returns to step 2.   AF-S4: If the password does not pass security requirements:   * System displays a message: “Please make sure the password fulfils the requirements” for 2 seconds. * System returns to step 3.   AF-S6: If re-entered password is different from that originally entered:   * System displays a message: “Re-entered password is different!” for 2 seconds. * System returns to step 5.   AF-S7: The user chooses not to enter a postal code for their account.   * The system proceeds with account creation as postal code is not a required field. |
| Exceptions: | NIL |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC002 | | |
| Use Case Name: | Login | | |
| Created By: | Germaine Kwek | Last Updated By: | Alexander Wong |
| Date Created: | 20/2/2019 | Date Last Updated: | 07/4/2019 |

|  |  |
| --- | --- |
| Actor: | User(initiating actor) |
| Description: | The user attempts to login with his/her email and password. |
| Preconditions: | The user has an account with a email and password. |
| Postconditions: | The user is logged in. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1) User clicks on “Login”.  2) System requests user to enter the email and password.  3) User enters a email.  4) The system verifies the email belongs to an account in the “user information” database.  4) User enters a password.  5) The system verifies the password matches the account in the “User Information” database.  6) The system logs the user in. |
| Alternative Flows: | AF-S2: If user clicks on “login” without entering a email and password:   * System display a message: “please fill out this field”. * System returns to step 2.   AF-S4: If an account with the given email is not found in the “User Information” database:   * System displays a message: “Invalid email address.” * System returns to step 2.   AF-S5: If password does not match that stored in the record identified by the user’s email in the “User Information” database:   * System displays a message: “Incorrect password.” * System returns to step 2. |
| Exceptions: | NIL |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC003 | | |
| Use Case Name: | Display Saved Schools | | |
| Created By: | Germaine Kwek | Last Updated By: | Alexander Wong |
| Date Created: | 20/2/2019 | Date Last Updated: | 07/4/2019 |

|  |  |
| --- | --- |
| Actor: | User(initiating actor), “School Information” database(participating actor), API(distance) (participating actor) |
| Description: | The system displays the schools saved by the user with the option to sort the information in different ways. |
| Preconditions: | The user is logged in. |
| Postconditions: | All schools saved by the user are displayed. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1) User clicks on “Display Saved Schools”.  2) System retrieves the names of all schools as well as the dates they were saved from the record identified by the user’s email in the “Saved Schools” database.  3) The system displays the list of saved schools.  3) If the user clicks on “Saved Date”, then the “Sort by Saved Date” use case is activated.  4) If the user clicks on “Alphabetical”, then the “Sort By Alphabetical” use case is activated.  5) If the user clicks on “Distance”, then the “Sort By Distance” use case is activated.  Date:  1) System retrieves the date the schools are saved from the record identified by the user’s username in the “Saved Schools” database and ranks them in order from the most to least recent date.  2) System displays the schools in order from most to least recently saved.  Alphabetical:  1) System orders the schools in alphabetical order from “A” to “Z”.  2) System displays the schools in that order.  Distance:  1) Activate the “Read Postal Code” use case.  2) If the “Read Postal Code” use case returns a valid postal code, the system retrieves the addresses of the schools from the records identified by the schools’ names in the “School Information” database.  3) System makes API(distance) calls with the user’s postal code and the schools’ addresses to get the distances between the user’s home and the schools, which it orders from shortest to longest.  4) System displays the school names and their distances from the user’s home in that order. |
| Alternative Flows: | AF-S2: If there are no schools saved in the “Saved Schools” database:   * System displays a message: “There are no schools saved!” for 2 seconds. * End of use case.   AF-Distance-S2: If the “Read Postal Code” returns a NULL data:   * System displays a message: “No postal code; unable to sort by distance.” for 2 seconds. * The “Sort by Distance” button is greyed out. |
| Exceptions: | NIL |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC004 | | |
| Use Case Name: | School Search by Criteria | | |
| Created By: | Germaine Kwek | Last Updated By: | Alexander Wong |
| Date Created: | 20/2/2019 | Date Last Updated: | 07/4/2019 |

|  |  |
| --- | --- |
| Actor: | User(initiating actor), “School Information” database(participating actor), API(distance) (participating actor), API(map) (participating actor) |
| Description: | The user searches for schools that meet his/her selected criteria(s). |
| Preconditions: | NIL |
| Postconditions: | The schools that meet the user’s criteria(s) are displayed. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1) User clicks on “Search by Criteria”.  2) System requests user to select from the displayed search criterias.  3) The user selects and the relevant criteria(s).  4) The user selects the “submit” button.  5) The System filters the “School Information” database for schools that meet the criteria(s), retrieves their names and addresses and puts them into lists for each specific criteria.  6) The System compares the respective lists and puts the common schools into a final results list.  8) If the user clicks on a school displayed, system retrieves all the information about the school from the record identified by the school’s name in the “School Information” database and displays them.  9) If the user clicks on the star button, activate the “Save” use case. |
| Alternative Flows: |  |
| Exceptions: | NIL |
| Includes: | UC008 Read Postal Code |
| Special Requirements: | 1) The system must enable at least 80% of first-time users to learn how to enter and submit a query within 3 minutes of starting to use the system.  2) The system must display the results within 3 seconds of a query’s submission. |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC005 | | |
| Use Case Name: | School Search by Name | | |
| Created By: | Germaine Kwek | Last Updated By: | Alexander Wong |
| Date Created: | 20/2/2019 | Date Last Updated: | 07/4/2019 |

|  |  |
| --- | --- |
| Actor: | User(initiating actor), “School Information” database(participating actor), API(map) (participating actor), API(distance) (participating actor) |
| Description: | The user searches for a school and its information using the school’s name. |
| Preconditions: | NIL |
| Postconditions: | The school and its information is displayed. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1) User clicks on “Search by Name”  2) System requests user to enter the name of the school to search.  4) User enters the name of the school.  5) If the name of the school is found in the “School Information” database, system retrieves the entire record identified by the school’s name(including its address)  6) The system compiles a list of all the schools matching the given name and displays it in a results list. |
| Alternative Flows: |  |
| Exceptions: | NIL |
| Includes: |  |
| Special Requirements: | 1) The system must enable at least 80% of first-time users to learn how to enter and submit a query within 3 minutes of starting to use the system.  2) The system must display the results within 3 seconds of a query’s submission. |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC006 | | |
| Use Case Name: | Save School | | |
| Created By: | Germaine Kwek | Last Updated By: | Alexander Wong |
| Date Created: | 20/2/2019 | Date Last Updated: | 07/4/2019 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | The user saves a school. |
| Preconditions: | The user must be logged in.  This use case extends UC004 School Search by Criteria.  This use case extends UC005 School Search by Name. |
| Postconditions: | The school is saved by the user. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1) The user checks the boxes of the schools they wish to save.  2) The user clicks “submit”.  3) The System saves the school name to the user’s account in the “users information” database.  4) The System displays the message “The schools have been saved” |
| Alternative Flows: | NIL |
| Exceptions: | NIL |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC007 | | |
| Use Case Name: | Unsave School | | |
| Created By: | Germaine Kwek | Last Updated By: | Alexander Wong |
| Date Created: | 20/2/2019 | Date Last Updated: | 07/4/2019 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | The user deletes a school. |
| Preconditions: | The user must be logged in.  The list of saved schools must not be empty |
| Postconditions: | The school is deleted by the user. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1) The user checks the boxes of the schools they wish to delete.  2) The user clicks “submit”.  3) The System deletes the school name from the user’s account in the “users information” database.  4) The System displays the message “The schools have been successfully unsaved” |
| Alternative Flows: | NIL |
| Exceptions: | NIL |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC008 | | |
| Use Case Name: | Sort By Alphabetical | | |
| Created By: | Germaine Kwek | Last Updated By: | Alexander Wong |
| Date Created: | 20/2/2019 | Date Last Updated: | 20/2/2019 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | The system sorts the results alphabetical |
| Preconditions: | The user must be logged in.  This use case extends UC004 School Search by Criteria.  This use case extends UC005 School Search by Name. |
| Postconditions: | The results list is sorted alphabetically |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1) The user clicks on the “alphabetical” button.  2) The System sorts the list of schools alphabetically.  3) The System displays the sorted list. |
| Alternative Flows: | NIL |
| Exceptions: | NIL |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC009 | | |
| Use Case Name: | Sort By Distance | | |
| Created By: | Germaine Kwek | Last Updated By: | Alexander Wong |
| Date Created: | 20/2/2019 | Date Last Updated: | 07/4/2019 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | The system sorts the results by distance |
| Preconditions: | The user must be logged in.  This use case extends UC004 School Search by Criteria.  This use case extends UC005 School Search by Name. |
| Postconditions: | The results list is sorted by distance |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1) The user clicks on the “Distance” button.  2) The System retrieves the users postal code from the “User Information” database.  3) The System sorts the results list by distance from the users postal code.  4) The System displays the sorted list. |
| Alternative Flows: | NIL |
| Exceptions: | NIL |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC010 | | |
| Use Case Name: | Sort By Saved Date | | |
| Created By: | Germaine Kwek | Last Updated By: | Alexander Wong |
| Date Created: | 20/2/2019 | Date Last Updated: | 07/4/2019 |

|  |  |
| --- | --- |
| Actor: | User |
| Description: | The user sorts the results by saved date |
| Preconditions: | The user must be logged in.  This use case extends UC004 School Search by Criteria.  This use case extends UC005 School Search by Name. |
| Postconditions: | The results list is sorted by saved date |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1) The user clicks on the “Saved Date” button  2) The System sorts the list of schools by their saved dates  3) The System displays the sorted list |
| Alternative Flows: | NIL |
| Exceptions: | NIL |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

