The Walruses

Code 1

Kenny Liang (kenny.liang.1@stonybrook.edu)
Adam Tringali (adam.tringali@stonybrook.edu)
Dylan Singh (dylan.singh@stonybrook.edu)
Nathan Melnyk (nathan.melnyk@stonybrook.edu

Table Of Contents

1. Persistence

1.1 Data Model

Our group is using MongoDB (NOSQL database) with an object document mapper (ODM).

For each collection (Student and College) we define schemas that allow us to define the fields,types, default values of a document for a collection. From schemas, we can create our models.

These schemas are located in our src/c4me_express/database/schemas/ directory. Only College and Student schemas are completed at the moment.

1.2 Queries

Filtering has not been implemented but we plan to use application code and database queries to filter the data. Mongodb uses Javascript as its query language with various operators\expressions and functions for CRUD operations.
Example of difference vs SQL system.

```
db.inventory.find( { status: { $in: [ "A", "D" ] } } )
```

```
SELECT * FROM inventory WHERE status in ("A", "D")
```

2. Code Conventions

Use of google's JavaScript coding conventions - https://google.github.io/styleguide/jsguide.html

1. ES Modules - use JavaScript import and export to use variables,functions from other files or 3rd party libraries. (E.g. `import {hostname} from './constant.js';`)
2. Exceptions - try catch with proper error message and response
3. Variable declaration with "let" keyword or "const" keyword

4. Many more…

3. Implementation Status Report

1. Claiming full credit on section 7.1 to 7.5. Partial claim of (6.2) view\edit of student profile and partial credit of (6.3) displaying colleges and sorting.
2. See implementation file located in docs directory

4. Installation Manual

1. Installation of Nodejs/npm (Due to being dependent on the system and probably already installed on the test system a command will not be included). No database setup is required as our backend runs a script automatically to create a connection to **Mongodb in the cloud** and creates the database and collection. In the event, you may want to use a different mongodb server, visit the src/c4me_express/constants/databaseconst.js file to change the ip address.

2. Download project files and go into 416 file.

3. Running the backend server (Express)
    1. Cd src/c4me_express
    2. Npm install (downloads the dependencies, node_modules)
    3. Npm start

4. Running the client (React)
    1. Cd src/c4me_react
    2. Npm install (download the frontend dependencies, node_modules)
    3. Npm start

5. Test Report

Test Case Name	Import College Scorecard,,Scrap CollegeData,Scrap Ranking,Delete Profile
Description	Testing admin import\scraping data functionality. This test case verified database, backend server, and client functionality.
Precondition	Database can be reset or contains college information
Flow of events	1. Tester visits admin and clicks on one of the functionality buttons. 2. System executes the function to import or scrap. 3. Tester then clicks on 'college' on navigation bar 4. Tester expects to see a list of colleges from college.txt
Outcome	Success : To see if the test passed, the tester expects to see a long list of colleges (101 colleges in list) with the column name,admission,instate\outstate cost, ranking. Errors: no colleges are displayed or missing colleges or data for columns are missing or errors in backend (check logs)

Test Case Name	View\Edit Student Profile
Description	Testing Viewing\Editing a student profile
Precondition	Database can be reset or contains information on students.
Flow of events	1. Tester visits "all Profiles" on the navigation bar to see a list of students in the system. This page is a testing page. 2. Tester clicks on a particular row for a particular student 3. Tester views the profile and can make edits\saving changes
Outcome	Success : To see if the test passed, the tester expects to see a list of students if student profiles were imported. Then each student's profile can be viewed and information is editable and can be saved. Errors: if a user's info was edited and saved, the tester expects to see the saved data once the tester refreshes the page. If a profile does not appear and the tester is expecting the information of the student to be displayed. In addition the expected list of students is the number of students rows imported from the cv file.

Test Case Name	Server Routes
Description	Testing backend API routes using Postman
Precondition	Backend server running.
Flow of events	1. Tester enters a particular backend API route on Postman to see the expected status code (200) and any addition json data if applicable
Outcome	Success : status code (200) and expected json data (E.g route returns a list of colleges or route returns a particular college object) Error: status code (500) and error messages.

6. Contributions

Kenny Liang

1. Worked on backend routes and database and stored data from importing\scraping of data to database.

Adam Tringali

1. Worked on frontend screens to display the fields of students and colleges

Dylan Singh

1. Worked on the scraping and parsing the html content of the college websites to scrap

Nathan Melnyk

1. Worked on frontend and testing of backend and frontend systems.

9. Requirements

1. See requirements.txt for requirements of project

10. Design

1. See design.pdf for design document of project (HW1-HW3 designs)

11. Video

1. See demo.mp4 for demonstration video of current working functionality