













## **Revised Features List:**

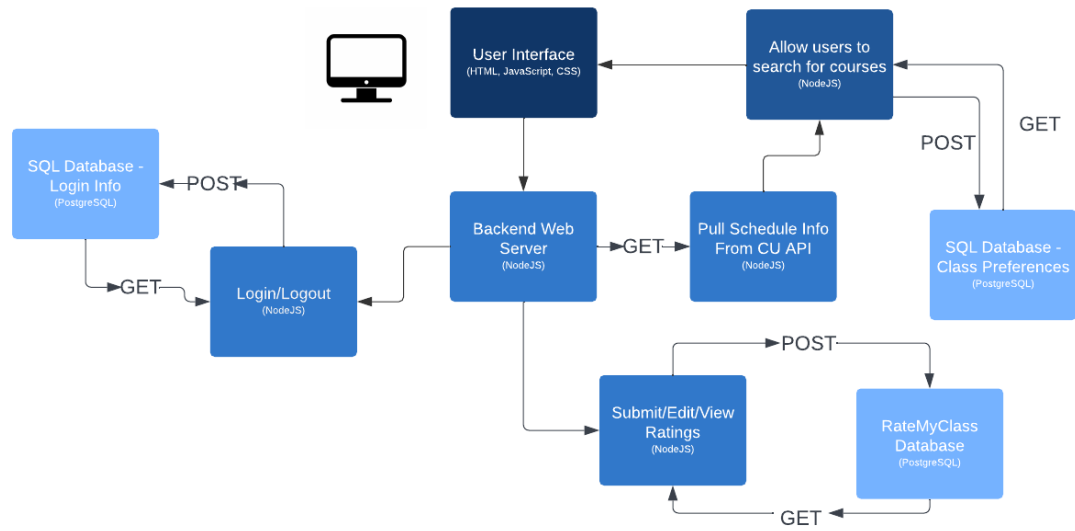
 - high priority

 - medium priority

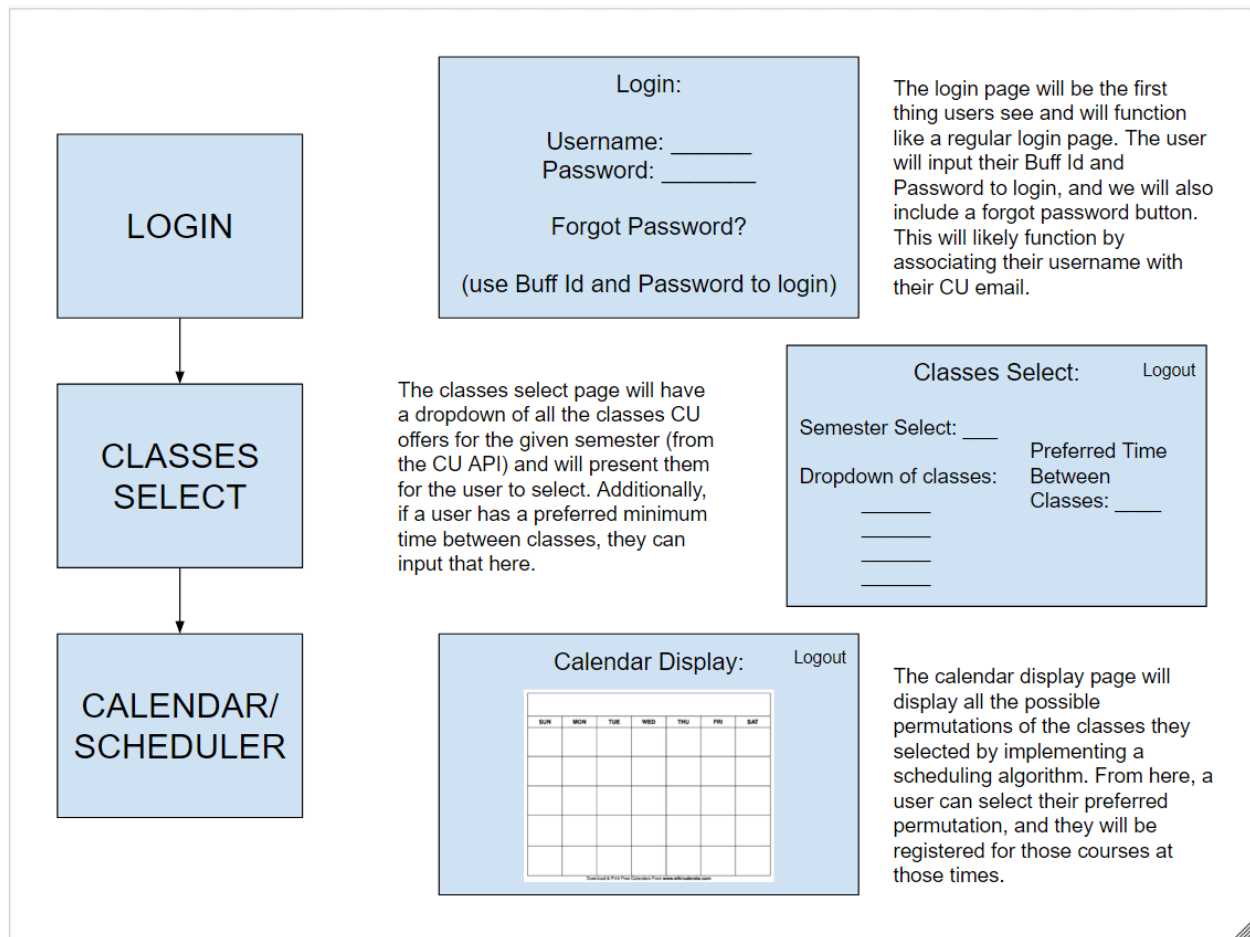
 - low priority

-  Login/Sign Up Page
  - Page to sign up/login to the site
-  Scheduling algorithm
  - Takes user selection and calculates possible options that are compatible
-  Calendar display
  - Displays information from scheduling algorithm visually
-  Search Page
  - Display available classes, allow searching for particular classes to take
  - Show availability, number of seats, professor, etc.
-  Cart system
  - Add/delete classes from cart
-  Dark/Light Mode (NEW)
  - Changes style of any page from light mode to dark
-  Preferences
  - Save preferences to account, restore them when revisiting page
-  RateMyClass Page
  - Allows users to rate the class using a form
  - Info will then be displayed to other users along with the course from database
-  Security Check (will likely be dropped)
  - Check if user is or was a student/actually took the class

## Architecture Diagram:



## Front End Design:



## Web Service Design:

No an official api but we'll be making calls to the cu classes api found at <https://classes.colorado.edu/api/>

By making POST requests to

- <https://classes.colorado.edu/api/?page=fose&route=details>
- <https://classes.colorado.edu/api/?page=fose&route=search>

with encoded JSON in the body such as:

JSON to search for “CSCI” - responds with list of matching classes

```
{
  "other":{
    "srcdb":"2227"
  },
  "criteria":[
    {
      "field":"keyword",
      "value":"CSCI"
    }
  ]
}
```

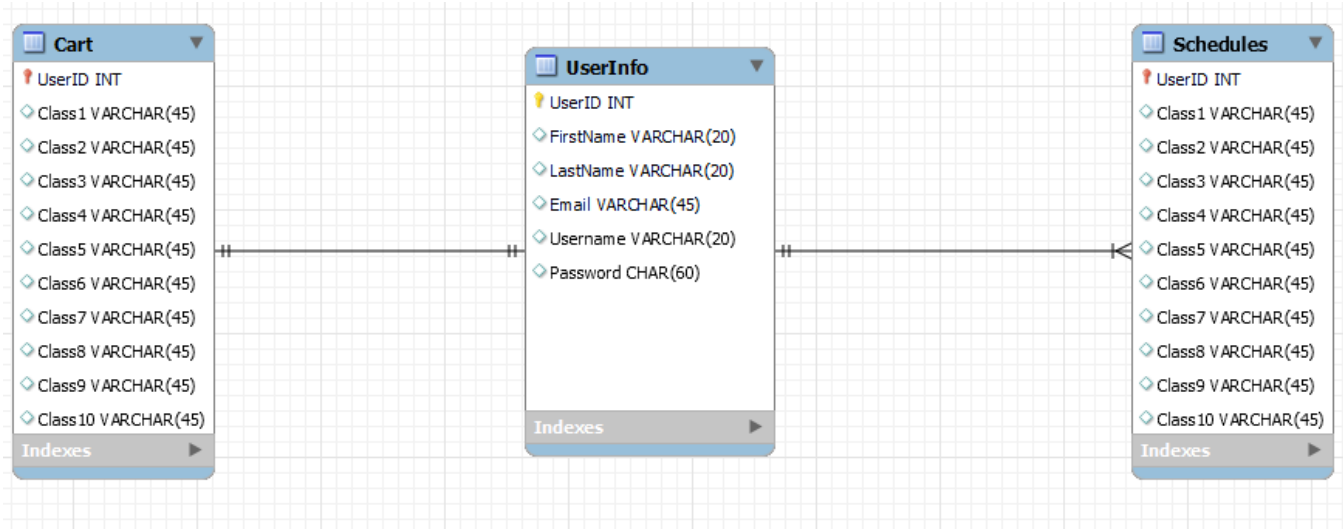
JSON to get details for “CSCI 2270” - responds with class details

```
{
  "group":"code:CSCI 2270",
  "srcdb":"2227"
}
```

We’ll get a JSON response with all the class information or search results that we can parse to get the course description, times / dates of each section, whether a lecture requires a recitation etc. [Course Details JSON Response Example](#)

### **Database Design:**

- The database will store user information and classes chosen by the user which will be run against a JSON file pulled from the CU class API. The information will later be parsed and run through an algorithm to output the possible options to the user.
- Currently the database is using MySQL but is prepared to merge over to PostgreSQL due to alleged restrictions with Heroku.
- The database code is accessible via github.
- The database will function as...



- UserID will be assigned as an auto incrementing non null unique index INT when the user creates an account.
  - The UserID is a primary key for the UserInfo table and a foreign key for the cart and schedules tables.
- UserInfo
  - Has a surrogate key for the primary key "UserID"
    - Auto incrementing non null unique index INT
  - Stores user information
    - First Name
      - Variable Character of 20
    - Last Name
      - Variable Character of 20
    - Email
      - Variable Character of 45
    - Username
      - Variable Character of 20
    - Password
      - Character of 60
        - Stores hashed password which is always 60
- Cart
  - Foreign key of UserID
  - Stores chosen class information
    - Class Name (multiple rows)
      - Variable Character of 45
        - Parses JSON for times and days
- Schedules

- Foreign key of UserID
- Stores chosen class information
  - Class Name (multiple rows)
    - Variable Character of 45
      - Parses JSON for times and days
  - May add spot to store time/days as it will be stored schedules

### **Challenges:**

- Decide what information should be stored in database for speed reasons vs api requests and parsing JSON files
  - Backup plan is to parse JSON files as that will be how information is originally retrieved.
- Creating scheduling algorithm to create schedules based on class times
  - There are likely some algorithms that we can use, but in case these don't work or we can't use them, we can work on creating our own from scratch using python or c++
- Checking whether a student actually took the class they are reviewing (RateMyClass)
  - This might be difficult due to CU security concerns
  - If we cannot do this, it not the end of the world, but we may want to add some different restrictions to combat review-bombing courses (one review per person, etc.)

### **Individual Contributions:**

Owen - for milestone 4: made API description. For project: created cart/search classes system

Tristan - for milestone 4: revised list of features, worked on architecture diagram, challenges. For project: created sign up page to be connected to database

Solomon - for milestone 4: worked on challenges, made database description. For project: set up database, wrote queries

Greyson - for milestone 4: made front-end wireframes. For project: created light/dark mode toggle for site

Gabe - For project: created site home page

GitHub Commits:

<https://github.com/cub-csci-3308-spring-2022/csci-3308-spring22-015-01/tree/main/Project%20Components>

We are organizing through discord in terms of what tasks need to be done by when. In addition to our weekly meetings with our TA, we are meeting on an as-needed basis over zoom to discuss the workload and project organization. So far, we have been giving each team member a feature to work on, and are currently working on integrating them together with consistent CSS styling.