CS 2300 - Project Phase II - Relational Data Modeling

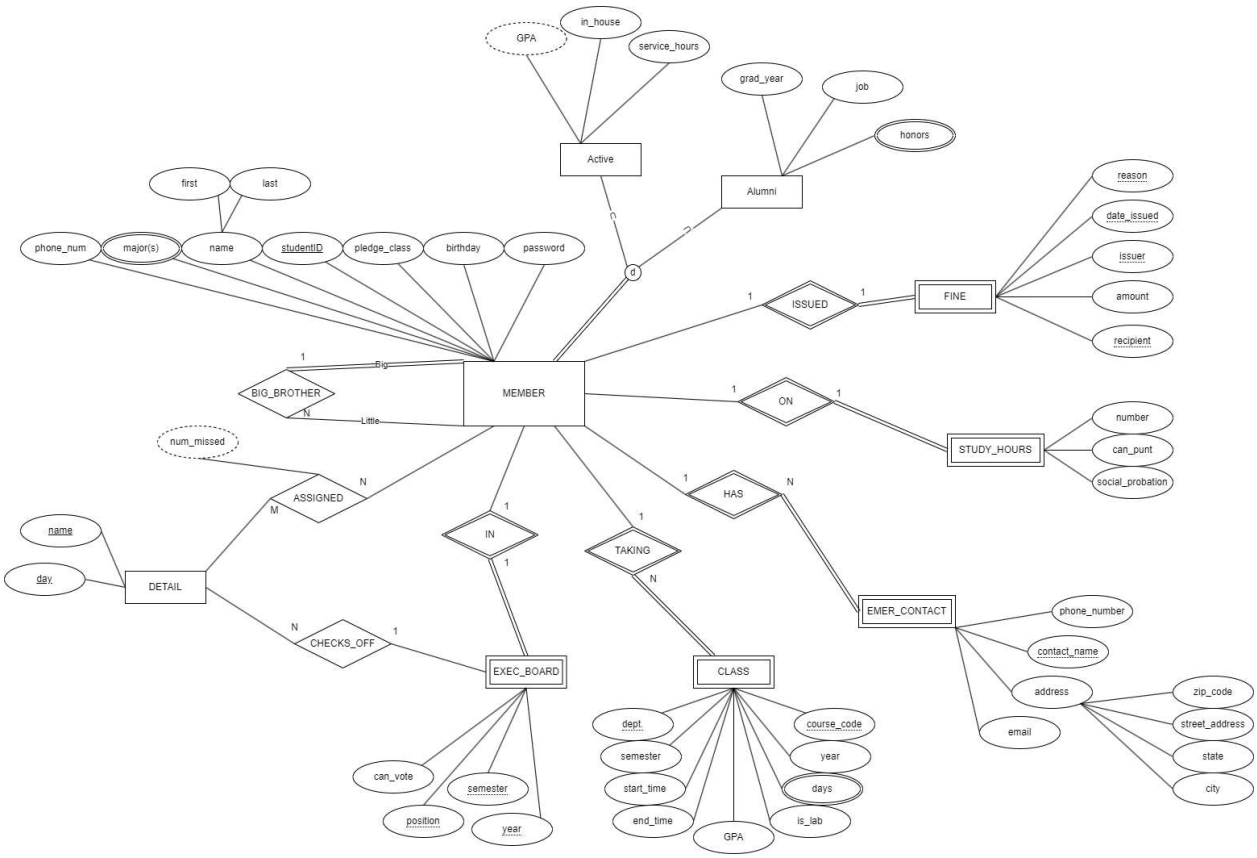
Members: Jimmy Hoerschgen , Dakota Smith

Team Name: Beta Sig

Problem Statement

Beta Sigma Psi is a fraternity and student organization at Missouri S&T. Like most organizations with many members, there is quite a bit of information that needs to be stored and manipulated. The information about each member includes what classes they're taking, what responsibilities they have around the house, and what positions they hold, among other things. This data is currently stored in a flat model, through a smattering of spreadsheets. Our team's objective is to create a database system that would act as an alternative to the system that is currently in use. This database will contain general information about each member like their name, major, year they joined the house, etc. The database will also be used to store specific information about each member including what study hours they are required to have and emergency contact information. This database would have all this data in one location, instead of separated over multiple disconnected files spread out over multiple drives. Once this data is established in the database, it will become easier to adjust and manipulate. A database would also allow for multiple user views and permissions; This would be important if information about member grades is stored. Creating a front end for data stored in the database would be a priority as well; Allowing for a less technical way for users to get value from the database. A potential use would be to generate a schedule and a report on a member's grades from just the front end using data in the database. This database will hopefully add organization and efficiency to how the house is run.

Conceptual Database Design

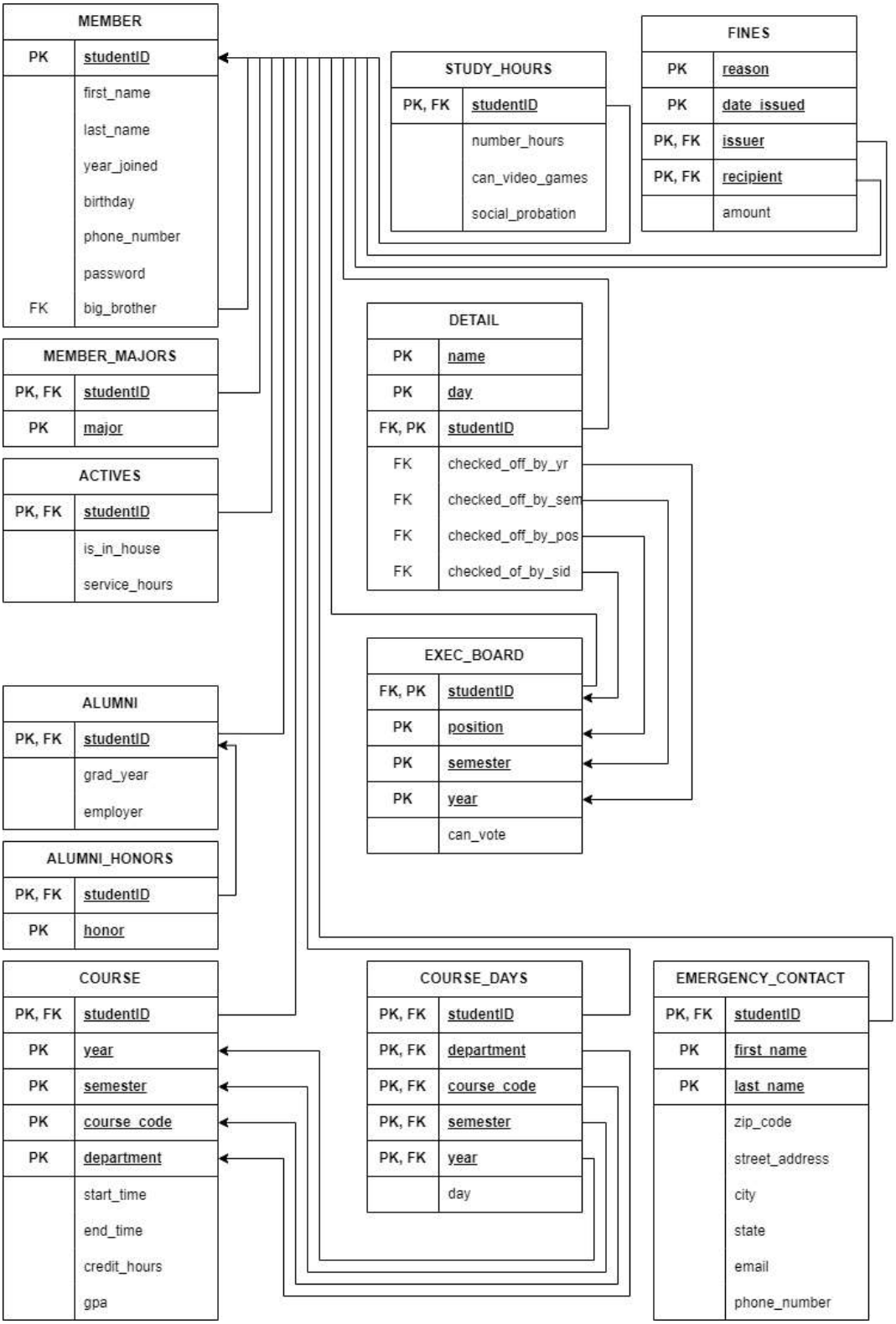


Assumptions:

- DETAIL ASSIGNED to MEMBER
  - M:N
    - Details can have multiple members assigned to it, and members can have multiple details.
- EXEC_BOARD CHECKS_OFF DETAIL
  - 1:N
    - A detail can be checked off by only one executive board member, and an executive board member can check off multiple details

- MEMBER IN EXEC_BOARD
  - 1:1
    - A position in the executive board can only be occupied by a single member.
- MEMBER TAKING CLASS
  - 1:N
    - A member can take multiple classes, each class is recorded distinctly for each member.
- MEMBER HAS EMER_CONTACT
  - 1:N
    - A member can have multiple emergency contacts, but an emergency contact is only counted once per each member.
- MEMBER ON STUDY_HOURS
  - 1:1
    - Not every member is necessarily on study hours, if a member is on study hours the information is for that member only.
- MEMBER ISSUED FINE
  - 1:1
    - A unique fine can only be issued once per member.
- BIG_BROTHER
  - 1:N
    - A member can only have one big brother, but multiple little brothers

Logical Database Design



Summary Table of Data Types

Table	Attribute	Type	Constraint
MEMBER	studentID	INTEGER	Primary Key
MEMBER	first_name	TEXT	Not NULL
MEMBER	last_name	TEXT	Not NULL
MEMBER	year_joined	INTEGER	Not NULL
MEMBER	birthday	DATE	
MEMBER	phone_number	TEXT	
MEMBER	password	TEXT	Not NULL
MEMBER	big_brother	TEXT	Foreign Key Not NULL
MEMBER_MAJORS	studentID	INTEGER	Primary Key Foreign Key
MEMBER_MAJORS	major	TEXT	Primary Key
ACTIVES	studentID	INTEGER	Primary Key Foreign Key
ACTIVES	is_in_house	BOOL	Is ‘TRUE’ or ‘FALSE’
ACTIVES	service_hours	INTEGER	
ALUMNI	studentID	INTEGER	Primary Key Foreign Key
ALUMNI	grad_year	INTEGER	
ALUMNI	employer	TEXT	
ALUMNI_HONORS	studentID	INTEGER	Primary Key Foreign Key
ALUMNI_HONORS	honor	TEXT	Primary Key
CLASS	studentID	INTEGER	Primary Key Foreign Key

CLASS	year	INTEGER	Primary Key
CLASS	semester	TEXT	Primary Key
CLASS	course_code	INTEGER	Primary Key
CLASS	department	TEXT	Primary Key
CLASS	start_time	TIME	Not NULL
CLASS	end_time	TIME	Not NULL
CLASS	credit_hours	INTEGER	Not NULL
CLASS	gpa	FLOAT	Between 0 and 4
CLASS_DAYS	studentID	INTEGER	Primary Key Foreign Key
CLASS_DAYS	year	INTEGER	Primary Key Foreign Key
CLASS_DAYS	semester	TEXT	Primary Key Foreign Key
CLASS_DAYS	course_code	INTEGER	Primary Key Foreign Key
CLASS_DAYS	department	TEXT	Primary Key Foreign Key
CLASS_DAYS	day	DATE	Not NULL
EMERGENCY_CONTACT	studentID	INTEGER	Primary Key Foreign Key
EMERGENCY_CONTACT	first_name	TEXT	Primary Key
EMERGENCY_CONTACT	last_name	TEXT	Primary Key
EMERGENCY_CONTACT	zip_code	INTEGER	
EMERGENCY_CONTACT	street_address	TEXT	
EMERGENCY	city	TEXT	

_CONTACT			
EMERGENCY _CONTACT	state	TEXT	
EMERGENCY _CONTACT	email	TEXT	
EMERGENCY _CONTACT	phone_number	TEXT	
STUDY_HOURS	studentID	INTEGER	Primary Key Foreign Key
STUDY_HOURS	number_hours	INTEGER	
STUDY_HOURS	can_video_games	BOOL	Is ‘TRUE’ or ‘FALSE’
STUDY_HOURS	social_probation	BOOL	Is ‘TRUE’ or ‘FALSE’
FINES	reason	TEXT	Primary Key
FINES	date_issued	DATE	Primary Key
FINES	issuer	TEXT	Primary Key Foreign Key
FINES	recipient	TEXT	Primary Key Foreign Key
FINES	amount	FLOAT	Not NULL
DETAIL	name	TEXT	Primary Key
DETAIL	day	TEXT	Primary Key
DETAIL	studentID	INTEGER	Primary Key Foreign Key
DETAIL	checked_off_by_studentid	INTEGER	Foreign Key
DETAIL	checked_off_by_position	TEXT	Foreign Key
DETAIL	checked_off_by_semester	TEXT	Foreign Key

DETAIL	checked_off_by_year	INT	Foreign Key
EXEC_BOARD	studentID	INTEGER	Primary Key Foreign Key
EXEC_BOARD	position	TEXT	Primary Key
EXEC_BOARD	semester	TEXT	Primary Key
EXEC_BOARD	year	INTEGER	Primary Key
EXEC_BOARD	can_vote	BOOL	Is ‘TRUE’ or ‘FALSE’

Functional Requirements:

- Insert Operations:
  - A method for adding a new member into the database
  - A method for assigning a member a detail
  - A method for adding a member to the executive board
  - A method for recording the classes a member is taking
  - A method for recording the emergency contact for a member
  - A method for giving a member study hours
  - A method for giving a member a fine
- Update/Modify Operations:
  - A method for checking off a detail
  - A method for making a member an alumnus
- Delete Operations
  - Delete a member from the database
- Aggregation Functions
  - Generate a weekly schedule for a member
  - Get total amount a member has been fined
  - Calculate the term GPA based off of the gpa for each class

Application Program Design

AddMember:

- Execute query uploading a new member's information to the database

DeleteMember:

- Execute queries deleting a member from the database using a members student id

GenWeeklySchedule:

- For a given member, semester, and year, generate a weekly schedule for that member that is sent to the frontend in the proper format

GetAllMembers:

- Retrieves information for all members as a table viewable in the frontend

LoginExec:

- Returns a Boolean value depending on if a given student id and password are valid for a current executive board member

CheckOffDetail:

- Uses the LoginExec function to validate if a user is an executive board member and if the login is successful, will check-off a given detail

AddToDetail

- Adds a member to a detail

GetDetails

- Gets all details for a given date range and returns their information as a table viewable in the frontend

GetAllDepartments

- Retrieves a list of all unique departments found in the database, helpful when generating options for the frontend input

InsertCourse

- For a member, adds a course and the days they take it to the database

GetExec

- Returns a list of all executive members for a semester as a table viewable in the frontend

CheckIsInExec

- Ensures member is not already serving in the executive board for a specific term

AddExec

- Adds member to the executive board for a given term

ModifyStudyHours

- Modifies the study hour information for a member

ModifyActive

- Modifies the service hours or if active is in house of an active

AddEmerContact

- Adds data for a members emergency contacts

ModifyEmerContact

- Modifies the emergency contact information for a member

AssignFine

- Adds a fine for a member from another member

GetAlumni

- Get all alumni in the database and return as a table viewable in the frontend

GoAlumni

- Add a member to the alumni table and remove them from the active table

AddAlumHonor

- Record an honor an alum has received

GetGrades

- Gets classes and grade information for a member of a given semester and returns as a table viewable in the frontend

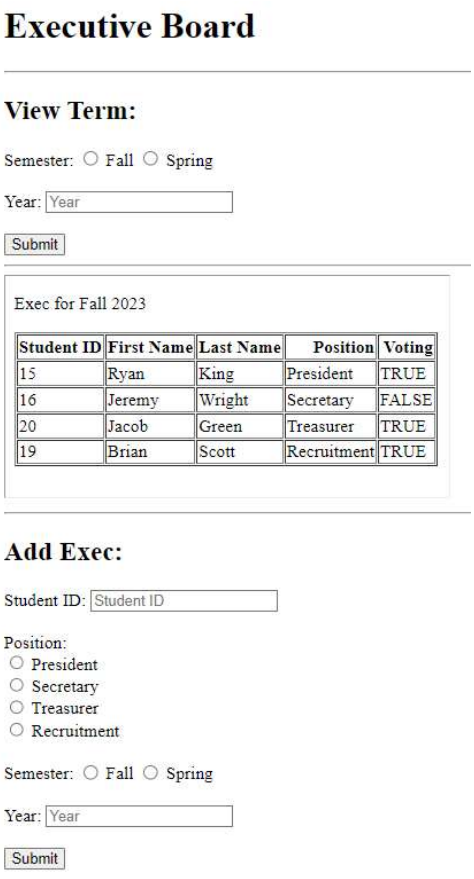
GetAvgGrade

- Gets the average grade for a member of a given semester

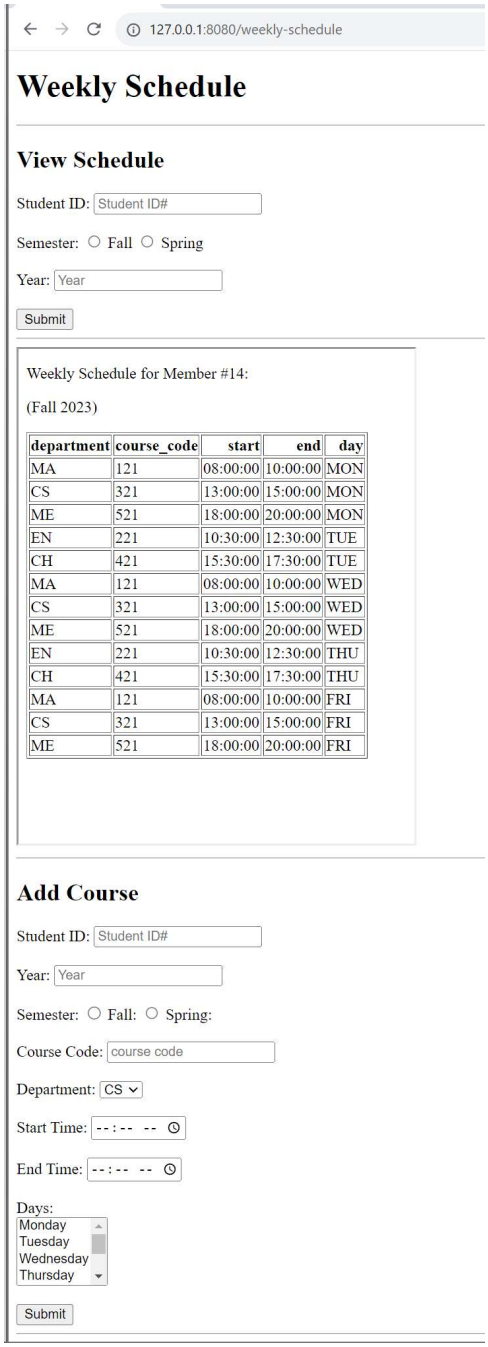
User Interface Design

Menu to select operations

Executive Board Page



Weekly Schedule Information



Details Schedule

name	day	fname	lname	studentid	checked_off
Kitchen	2023-11-07	Brian	Scott	19	FALSE
Kitchen	2023-11-08	Jacob	Green	20	FALSE
Kitchen	2023-11-09	Gary	Adams	21	FALSE
Kitchen	2023-11-10	Timothy	Nelson	22	FALSE
Kitchen	2023-11-11	Jose	Baker	23	FALSE
Bathroom	2023-11-07	Nicholas	Lopez	17	FALSE
Bathroom	2023-11-08	Kevin	Young	18	FALSE
Bathroom	2023-11-09	Brian	Scott	19	FALSE
Bathroom	2023-11-10	Jacob	Green	20	FALSE
Bathroom	2023-11-11	Gary	Adams	21	FALSE
Garbage	2023-11-07	Scott	Perez	25	FALSE
Garbage	2023-11-08	Jeremy	Wright	16	FALSE
Garbage	2023-11-09	Nicholas	Lopez	17	FALSE
Garbage	2023-11-10	Kevin	Young	18	FALSE
Garbage	2023-11-11	Brian	Scott	19	FALSE

Installation Instructions: (from project README.md)

CS2300 Project

by Jimmy Hoerschgen & Dakota Smith

Installation Instructions

Required Software:

- Windows
- Anaconda
- DB Browser for SQLite (Optional)

In the root of this repository is a YAML file with the correct config to instantiate an Anaconda environment with all of the necessary packages. Create an Anaconda environment with it using the following command in an Anaconda powershell window:

```
conda env create --file \cs2300_proj\CS2300.yml
```

A copy of the *.db* file used for this project is included in the repository. If you wish to create a fresh copy of the database run the *SQL* queries found in: **\cs2300_proj\sql_queries_initdb.sql**

If you create a new copy of the database you will need to modify the value for **DB_PATH** in **\cs2300_proj\app\constants.py**

Execution Instructions

To run the front-end application for this project run the following commands in the Anaconda powershell window:

```
conda activate CS2300
```

```
(CS2300) python app\main.py
```

Once you run these commands you should see something like this:

```
(CS2300) PS C:\...\cs2300_proj> python .\app\main.py
INFO:     Will watch for changes in these directories:
['C:\\...\\cs2300_proj']
INFO:     Uvicorn running on http://127.0.0.1:8080 (Press CTRL+C to quit)
INFO:     Started reloader process [37976] using StatReload
INFO:     Started server process [19048]
INFO:     Waiting for application startup.
CS 2300 Project
INFO:     Application startup complete.
```

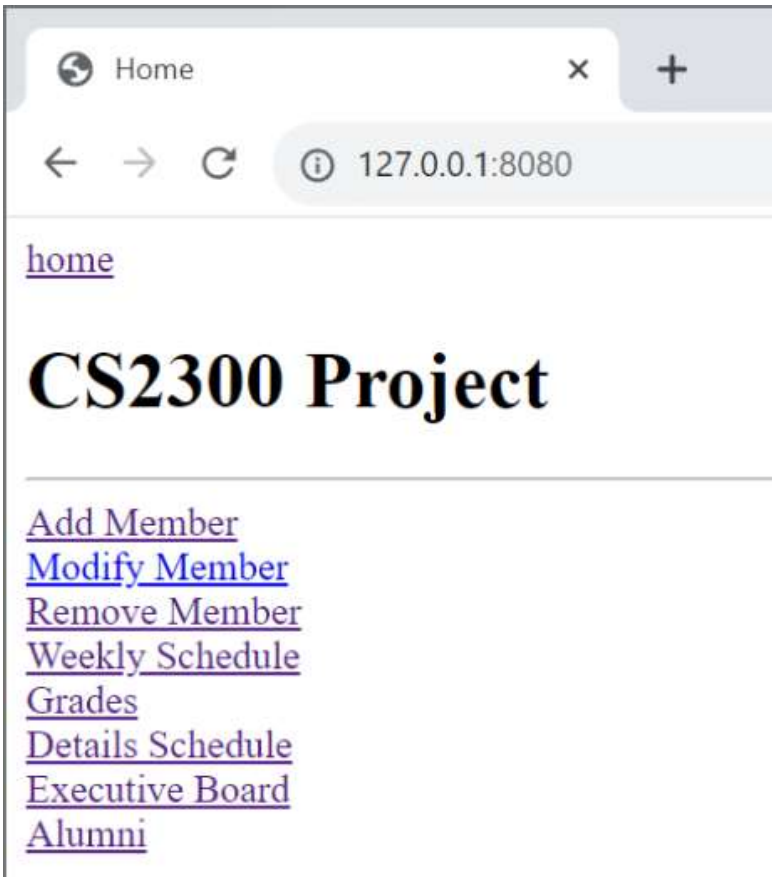
Enter the URL: **http://127.0.0.1:8080** into a web browser to access the application.

When you wish to terminate the application enter *CTRL + C* or exit the Anaconda powershell terminal.

User Manual:

Upon first entering the application the user is presented with the home page. There are several links to other pages that each contain different features. These pages are:

- *Add Member*
- *Modify Member*
- *Remove Member*
- *Weekly Schedule*
- *Details Schedule*
- *Executive Board*
- *Alumni*

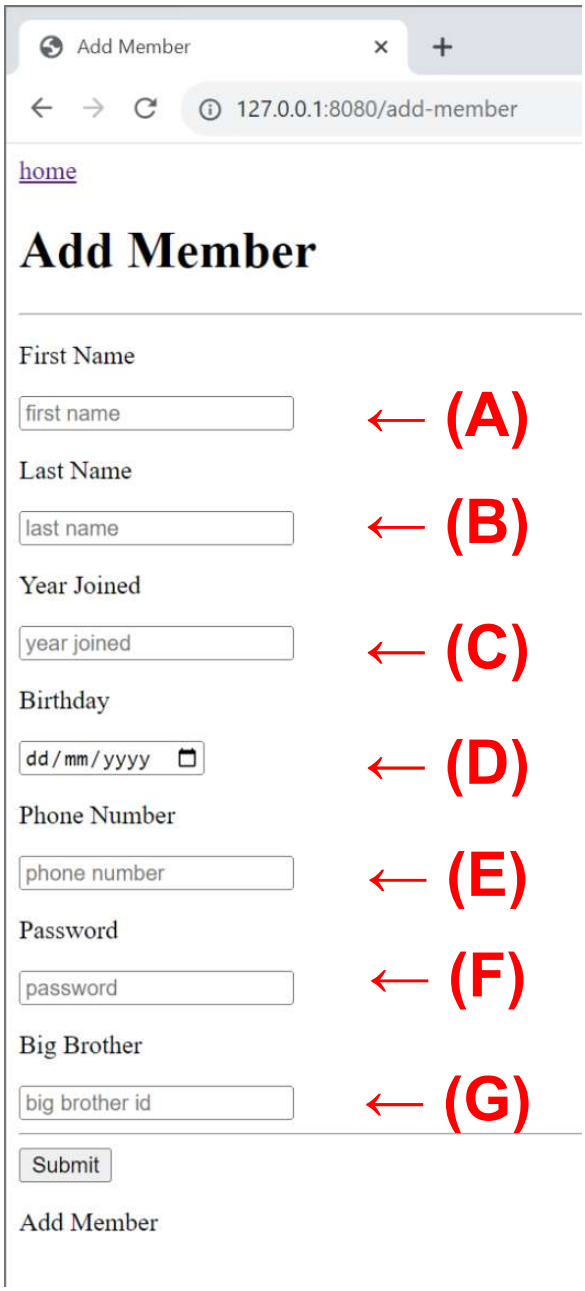


Add Member

The Add Member page allows the user to add new members to the database. This is done through a form. The user must put in valid values for the new member's:

- First name (A)
- Last name (B)
- The year they have joined (C)
- Their birthday (D)
- Their phone number (E)
- A password for the user (F)
- A "Big Brother", which the input is the Big Brother's ID#. (G)

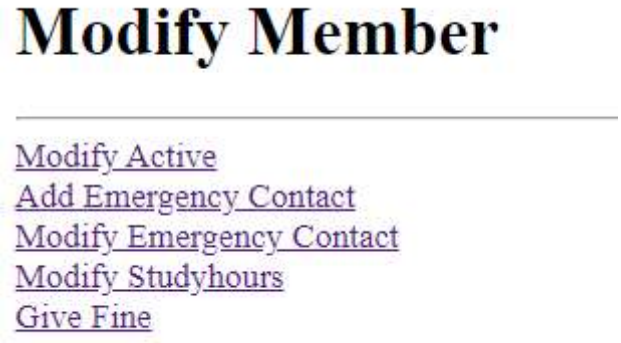
After the form is submitted the new member is assigned an ID# and is added as a member and an active member.



Modify Member

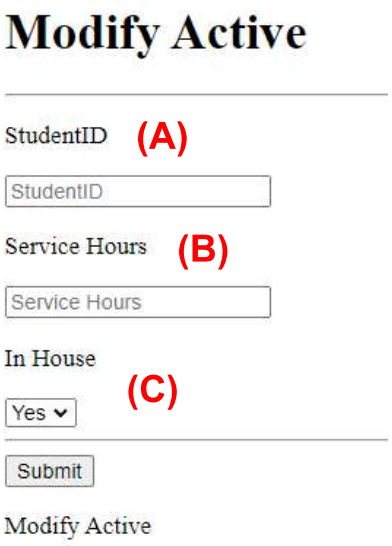
The Modify Member page is another page similar to the home page. Here there are several links to other pages to fulfill Modify and Insertion functions for members. These pages are:

- *Modify Active*
- *Add Emergency Contact*
- *Modify Emergency Contact*
- *Modify Study Hours*
- *Assign Fines*



Modify Active

The "Modify Active" page allows the user to update the amount of service hours that they currently have. As well as update wether or not the member specified is in house or not. The user must first specify a StudentID (A). The other two fields (Service Hours (B) and In House(C)) are optional, however "In House" is a dropdown and will always pass a value.



Add Emergency Contact

The “Add Emergency Contact” page allows the user to add an emergency contact for a studentID. This is done through a form. The user must put in valid values for the contact:

- StudentID (A)
- First Name (B)
- Last Name (C)
- Zipcode (D)
- Street Adderss (E)
- City (F)
- State (G)
- Email (H)
- Phone Number (I)

After this is submitted a new contact information is established for the StudentID.

Modify Emergency Contact

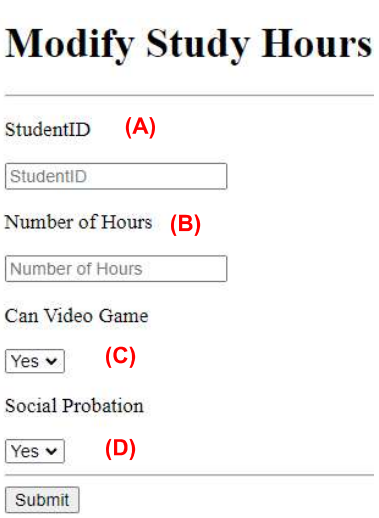
The “Modify Emergency Contact” page allows the user to modify an emergency contact for a studentID. This is done through a form. The user must put in valid values for the contact:

- StudentID (A)
- First Name (B)
- Last Name (C)
- Zipcode (D)
- Street Adderss (E)
- City (F)
- State (G)
- Email (H)
- Phone Number (I)

After this is submitted an updated contact information is established for the StudentID and Name. A-C are required values, the others are all optional so you don’t need to modify everything all at once.

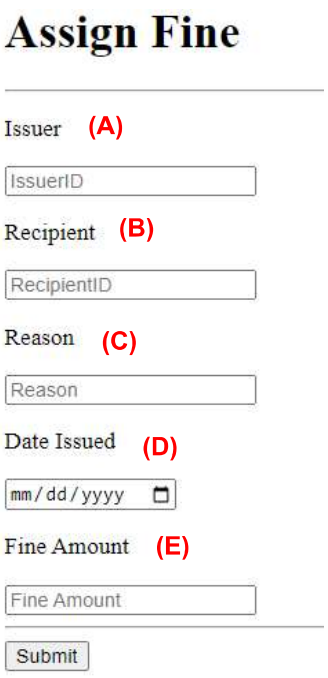
Modify Study Hours

The “Modify Study Hours” page allows the user to modify the study hours of a student on study hours. This is done by first specifying the studentID (A), the rest is all optional. Once this is sent the StudentID will have an updated amount of a combination of study hours (B), wether they can play video games (C), or if they are on social probation(D).



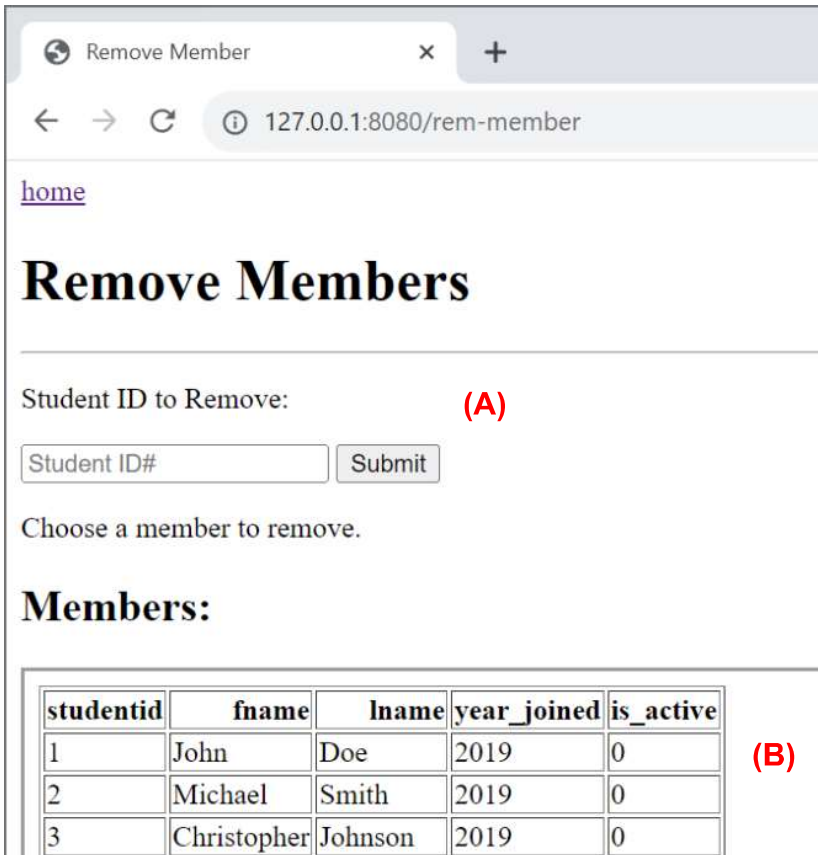
Assign Fine

The “Assign Fine” page allows the user to assign a fine to a member from a member. This is done by the User/Issuer (A) inputting their studentID, and assigning the fine to a Recipient(B) (another studentID). Then the reason is explained (C), the date of the fine (D) is input, and the amount that the recipient is being fined(E) is submitted.



Remove Member

The Add Member page allows the user to remove a member from the database. The user must enter the ID# of the member they want to delete (A). This page includes a table of current members (B) that the user may reference to avoid entering the incorrect member ID#.

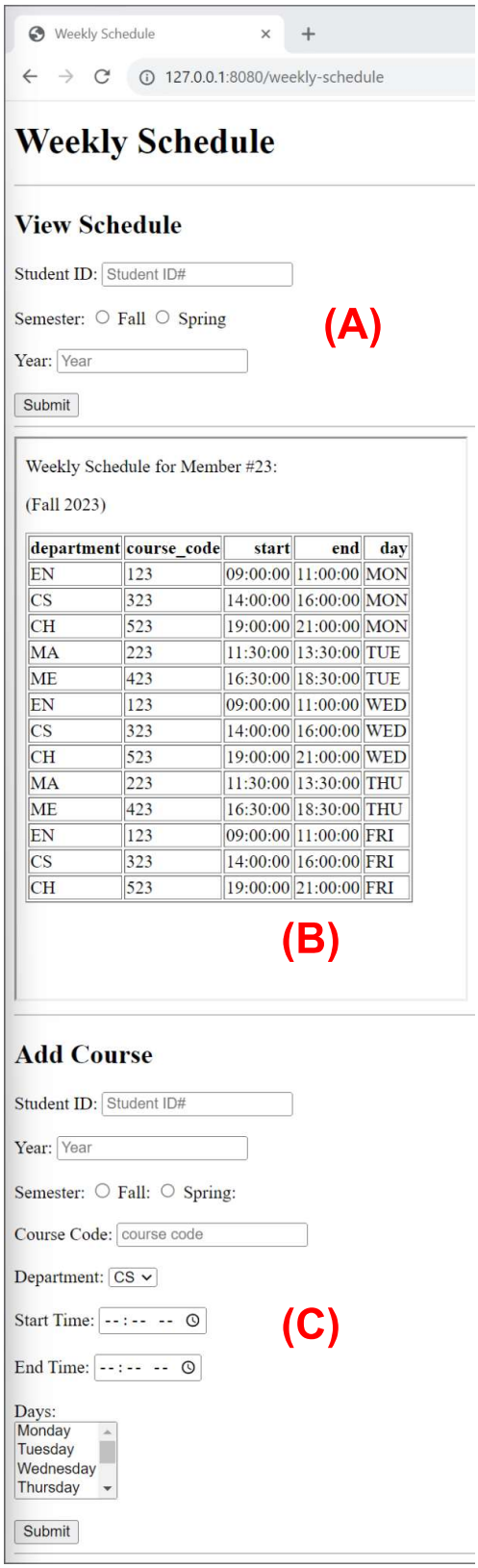


Weekly Schedule

This page allows the user to view the class schedule for a given member and semester (A).

On hitting submit the weekly schedule shown for the member as a table below the form (B).

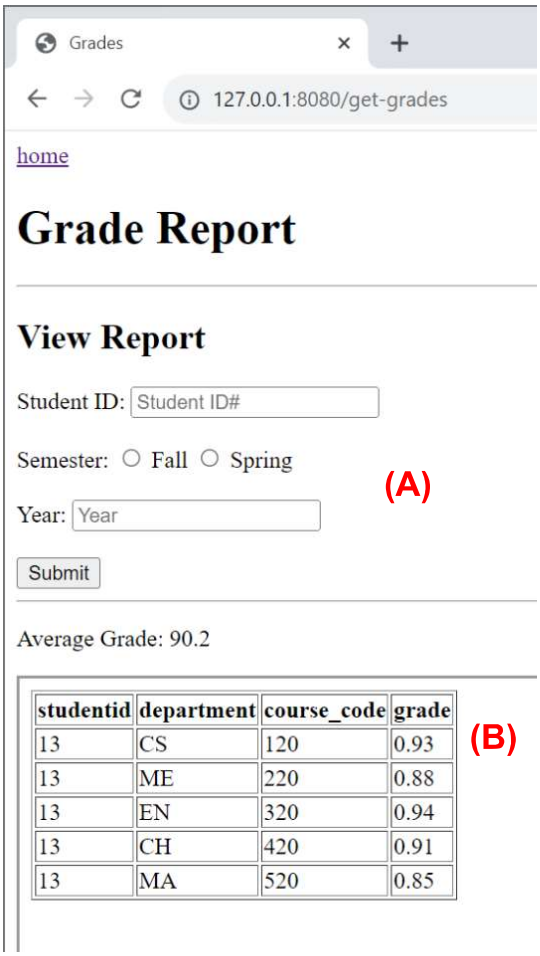
Below the schedule is another form that allows the user to add a course (C). When the course is added, the schedule refreshes, showing the updated schedule with the new class.



Grade Report

This page allows the user to view a report of a member's grades. Using the form at the top of the page (A), the user can select a member using their ID# and the semester the user wants to see the grades from.

After the form is submitted a table is generated that shows each class the member is taking and their grade (B). Their average grade for that semester is also shown.

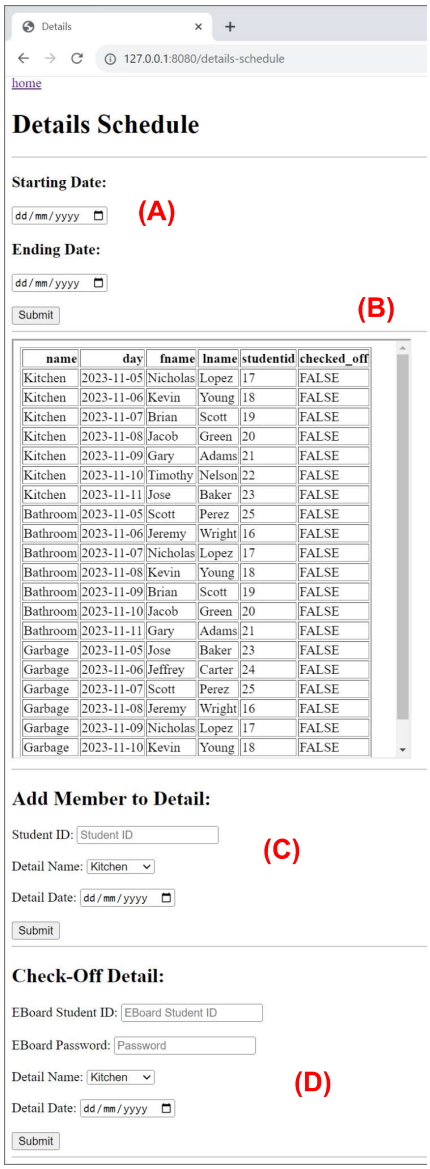


Details Schedule

This page lets a user view all assigned details for a given date range (A). Once the form is submitted a schedule is generated from the selection (B).

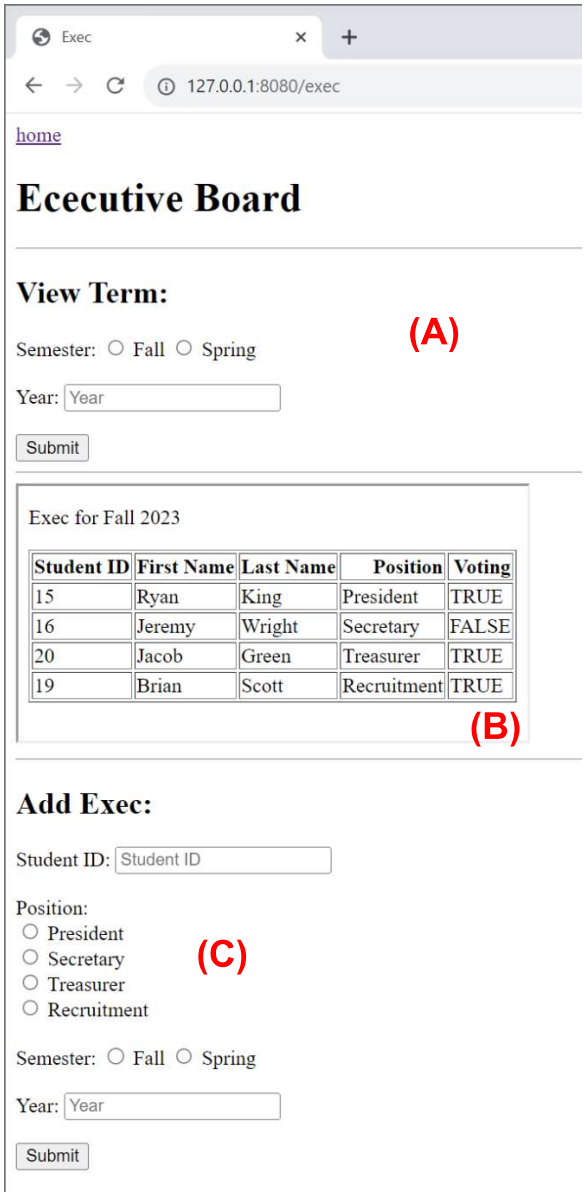
Below the schedule is a form that allows the user to add a member to a detail for a specific date (C).

Lastly, there is a form that allows a member in the current executive board to check-off a detail (D). The login must include a valid ID# and password, and the member must be a part of the executive board for that current semester and year.



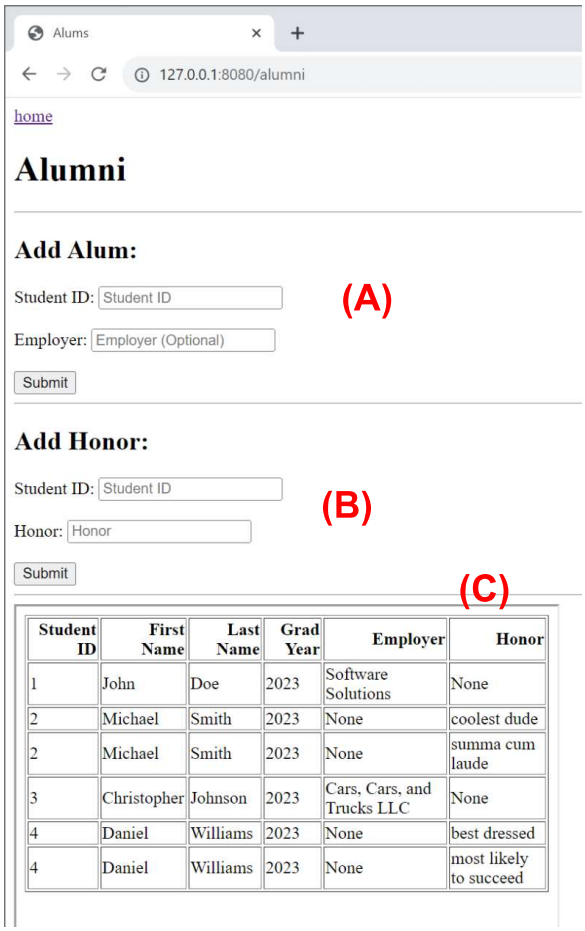
Executive Board

This page allows the user to view the executive board for a given term (A). The result is shown as a table (B). Below the table is a form to add new members to the executive board for a semester (C), if the user is already in the executive board for that term, they cannot be added as a second position.



Alumni

This page allows the user to add alumni (A) and the honors they earned in school (B) into the database. Information about alumni is in a table below the forms (C).



Student ID	First Name	Last Name	Grad Year	Employer	Honor
1	John	Doe	2023	Software Solutions	None
2	Michael	Smith	2023	None	coolest dude
2	Michael	Smith	2023	None	summa cum laude
3	Christopher	Johnson	2023	Cars, Cars, and Trucks LLC	None
4	Daniel	Williams	2023	None	best dressed
4	Daniel	Williams	2023	None	most likely to succeed