

Final Report

Nicholas Faylor, Email: nicholas.faylor@sjsu.edu, ID: 015663011

Timothy Phan, Email: timothy.t.phan@sjsu.edu, ID: 014819857

Raul Garcia, Email raul.garcia01@sjsu.edu, ID: 016289572

Ibrahim Dobashi, Email ibrahim.dobashi@sjsu.edu, ID: 015117414

Joseph Hawkins, Email: joseph.hawkins@sjsu.edu, ID: 015744001

San Jose State University

CS157A - Database Management Systems

Prof. Ramin Moazeni

May 9, 2024

Goals and Description of the Application

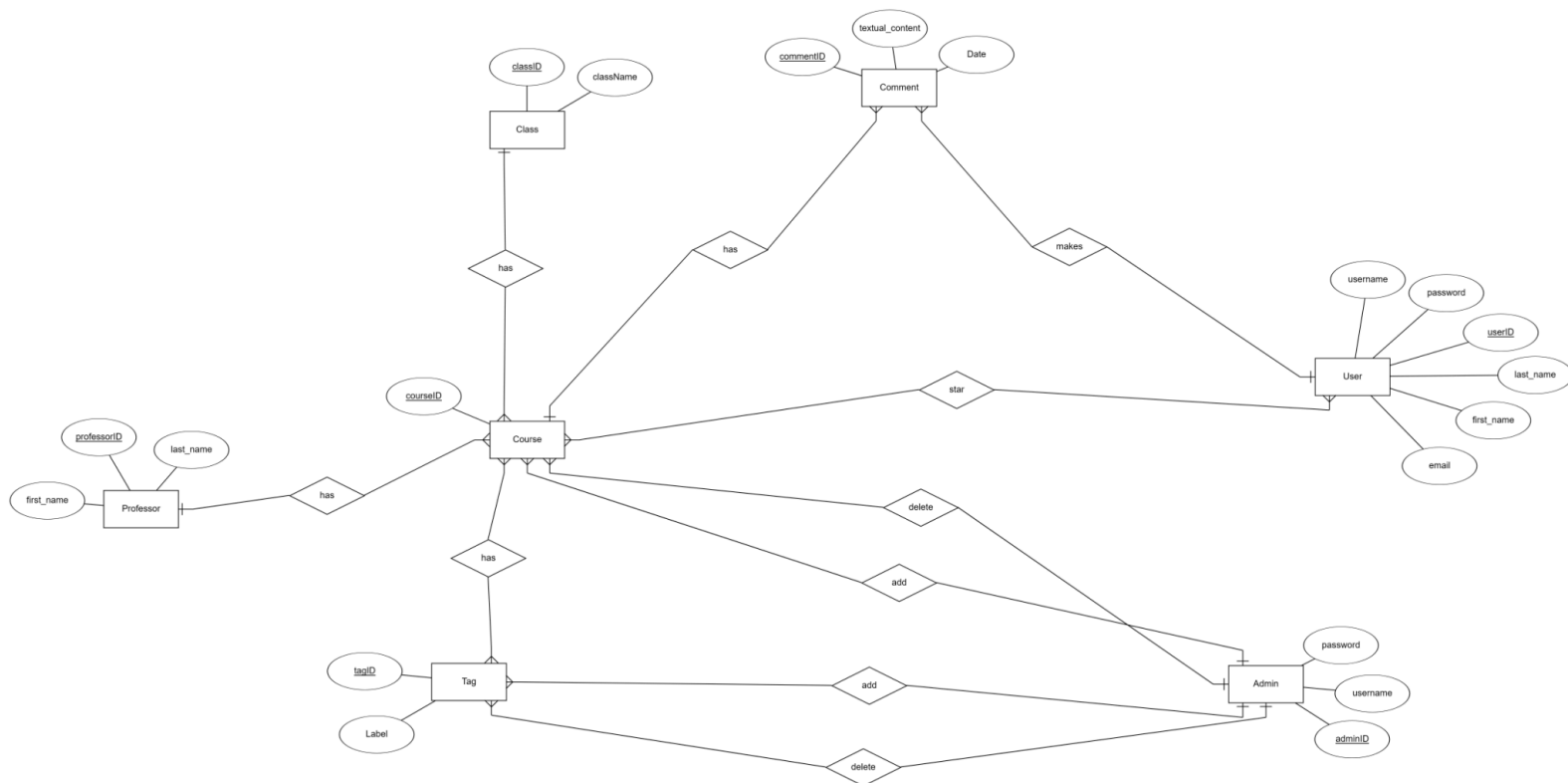
CourseBrief is a navigational web-app for SJSU CS and CMPE courses that detail specific courses down to the professor, about the specific topics taught in that course. For example, a student could search CourseBrief for courses that teach the concept of ‘sorting algorithms.’ This should help both prospective students and current students help choose classes that can tailor to their interests or explore courses that they must take. Oftentimes, students will only have access to a certain extent of information about a specific course taught by a professor. This app aims to prevent this, and to offer another layer of depth to the intricate topics taught in CS and CMPE courses. Our goals are to create a user-friendly application that ultimately allows users to search courses by a class name, professor’s name, and the ‘tags’ associated with the course. This ‘tag’ system allows for querying and searching the database for courses that contain these tags.

Application/Functional Requirements and Architecture:

- Create Account
 - User can create account with email, username and password.
- Search
 - Users will be able to search specific courses.
 - Users will be able to search specific course tags/topics taught in that course.
- Comment
 - Users may comment text on a specific course.
- Add Courses
 - When new courses are added - admin feature.
- Delete Courses

- When courses are deprecated and must be removed - admin feature.
- Sign in
 - Input validation, user name, email, password.
- Log out
- Delete Account
- Star a Course
 - Users can star courses.

ER Data Model:



Database Design:

CREATE TABLE User

```
(  
    userID VARCHAR(255) NOT NULL,  
    email VARCHAR(255) NOT NULL,  
    first_name VARCHAR(255) NOT NULL,  
    last_name VARCHAR(255) NOT NULL,  
    pass_word VARCHAR(255) NOT NULL,  
    username VARCHAR(255) NOT NULL,  
    PRIMARY KEY (userID)  
);
```

CREATE TABLE Professor

```
(  
    professorID VARCHAR(255) NOT NULL,  
    first_name VARCHAR(255) NOT NULL,  
    last_name VARCHAR(255) NOT NULL,  
    PRIMARY KEY (professorID)  
);
```

CREATE TABLE Tag

```
(  
    tagID VARCHAR(255) NOT NULL,
```

```
Label VARCHAR(255) NOT NULL,  
PRIMARY KEY (tagID)  
);
```

```
CREATE TABLE Class  
  
(  
    classID VARCHAR(255) NOT NULL,  
    className VARCHAR(255) NOT NULL,  
    PRIMARY KEY (classID)  
);
```

```
CREATE TABLE Admin  
  
(  
    adminID VARCHAR(255) NOT NULL,  
    username VARCHAR(255) NOT NULL,  
    pass_word VARCHAR(35) NOT NULL,  
    PRIMARY KEY (adminID)  
);
```

```
CREATE TABLE ActionTag  
  
(  
    actionID VARCHAR(255) NOT NULL,  
    action_type VARCHAR(255) NOT NULL,
```

```
tagID VARCHAR(255),  
adminID VARCHAR(255) NOT NULL,  
PRIMARY KEY (actionID),  
FOREIGN KEY (tagID) REFERENCES Tag(tagID) ON DELETE CASCADE,  
FOREIGN KEY (adminID) REFERENCES Admin(adminID) ON DELETE CASCADE  
);
```

```
CREATE TABLE Course
```

```
(  
courseID VARCHAR(255) NOT NULL,  
professorID VARCHAR(255) NOT NULL,  
classID VARCHAR(255) NOT NULL,  
PRIMARY KEY (courseID),  
FOREIGN KEY (professorID) REFERENCES Professor(professorID) ON DELETE  
CASCADE,  
FOREIGN KEY (classID) REFERENCES Class(classID) ON DELETE CASCADE  
);
```

```
CREATE TABLE Comment
```

```
(  
commentID VARCHAR(255) NOT NULL,  
textual_content VARCHAR(255) NOT NULL,  
Date DATE NOT NULL,
```

```
userID VARCHAR(255) NOT NULL,  
courseID VARCHAR(255) NOT NULL,  
PRIMARY KEY (commentID),  
FOREIGN KEY (userID) REFERENCES User(userID) ON DELETE CASCADE,  
FOREIGN KEY (courseID) REFERENCES Course(courseID) ON DELETE CASCADE  
);
```

```
CREATE TABLE CourseTags  
(  
courseID VARCHAR(255) NOT NULL,  
tagID VARCHAR(255) NOT NULL,  
PRIMARY KEY (courseID, tagID),  
FOREIGN KEY (courseID) REFERENCES Course(courseID) ON DELETE CASCADE,  
FOREIGN KEY (tagID) REFERENCES Tag(tagID) ON DELETE CASCADE  
);
```

```
CREATE TABLE StarredCourses  
(  
courseID VARCHAR(255) NOT NULL,  
userID VARCHAR(255) NOT NULL,  
PRIMARY KEY (courseID, userID),  
FOREIGN KEY (courseID) REFERENCES Course(courseID) ON DELETE CASCADE,  
FOREIGN KEY (userID) REFERENCES User(userID) ON DELETE CASCADE
```

);

CREATE TABLE ActionCourse

(

actionID VARCHAR(255) NOT NULL,

action_type VARCHAR(5) NOT NULL,

adminID VARCHAR(255) NOT NULL,

courseID VARCHAR(255),

PRIMARY KEY (actionID),

FOREIGN KEY (adminID) REFERENCES Admin(adminID) ON DELETE CASCADE,

FOREIGN KEY (courseID) REFERENCES Course(courseID) ON DELETE CASCADE

);

Major Design Decisions

Three Tier Model:

We decided to use a three tier architecture with java using spring boot as middleware to connect to a MySQL database. This makes sure that users cannot have direct access to the data and if inputs are sanitized then we can assure that there will be no issues with the data. This also means that the api of MySQL is not directly connected to the user interface and has to go through the middleware api that can handle all the transactions between the user and the database.

Indexing:

Our project uses the primary key indexing which is something that MySQL and Spring Boot JPA does automatically.

Entity Relationship Model:

In regards to our entities, we included Professor, Course, Tag, Class, Comment, Admin and User. These were the entities we felt were tables that held data specific to themselves, aside from their foreign keys. The relationship entities we included in our diagram were representative of the relationships between two tables, which include: StarredCourse, CourseTag, ActionTag and ActionCourse. Here is a brief description of each entity and our design intentions for each. They have specific terminology to describe their purpose and function:

User: Represents a CourseBrief user, which has attributes: userID, username, password, firstName, lastName and email.

Professor: Represents a professor at SJSU, which has attributes: firstName, lastName and a professorID.

Course: Represents a course at SJSU, which is made up of a class and a professor. It has attributes: courseID and contains a FK to Professor (professorID) and a FK to Class (classID).

Class: Represents a class or a subject at SJSU. It has attributes classID and className.

Tag: Represents a brief description that is used to group courses based on the criteria taught in a specific course. It has attributes tagID and label.

Comment: Represents a comment made by a User, on a course. It has attributes commentID, textual_content, and Date. It has a FK to User (userID) and a FK to Course (courseID).

Admin: Represents a 'CourseBrief' admin, with special privileges. It has attributes adminID, username and password.

StarredCourse: Represents the starred(saved) courses by a User. It has a composite primary key: userID, courseID. It also contains a FK to User (userID) and Course (courseID).

CourseTag: Represents the table that contains all of the 'tags' belonging to each course. It has a FK to Course (courseID) and Tag (tagID).

ActionTag: Represents the table containing all of the tags created/deleted by an Admin. It has attributes actionID and actionType. It contains a FK to Tag (tagID) and Admin (adminID).

ActionCourse: Represents the table containing all of the courses created/deleted by an Admin. It has attributes actionID and actionType. It contains a FK to Course (courseID) and Admin (adminID).

Functional Dependencies:

UserId → username, password, firstName, lastName and email

A → B

ProfessorId → firstName, lastName

C → D

CourseID → classId, profID

E → F

ClassID → className

H → I

TagID \rightarrow label

J \rightarrow K

commentID \rightarrow textual_content, Date

L \rightarrow M

adminID \rightarrow username, password

N \rightarrow O

actionID(tag) \rightarrow actionType, tag, admin

Q \rightarrow R

actionID(course) \rightarrow actionType, course, admin

S \rightarrow T

All of these functional dependencies are in BCNF, as each of them are non-trivial, and X (left hand side) contains a candidate key.

Implementation Details

Front-end:

For the front-end component of the application, we decided to go with React-Bootstrap as it provides easy to use components and creates fast UI designs. We also used the Axios API to handle the HTTP requests connecting the frontend to the backend. To handle the routing and authorized/unauthorized routes, we used the react-router library.

Back-end:

For the back-end component, we decided to use Java Spring boot due to team members having experience with the framework. It also includes various dependencies that help streamline the application process. Spring Boot follows a MVC architecture. The java files in the models folder are essentially the entity classes that will be created whenever we run the application. This is due to the Spring Boot JPA dependency which maps the Java entity classes to be tables in the database. The controller folder is where the communication between the frontend and backend happens. That is where we create the POST and GET requests. The repository folder is where the methods to save to or fetch from the database are stored. The service folder is where the business logic goes and is usually how the controller gets the data from the database. The controller would call the corresponding service method. Then the service method would call the corresponding method in the repository where it would either query from the database if the request was a GET or save to the database if the request was a POST.

Database Server:

For the database, we are using MYSQL as our DBMS as that was what we have been learning throughout the semester, so everyone is familiar with how everything should be structured.

Demonstration of example system run

This is a demo of our application. In order to follow along, please make sure to have these prerequisites:

- Java JDK 17 or higher
- Node
- NPM
- MySQL

You would also need to create a system environment variable called 'DB_PASSWORD' and set the variable to the MySQL server instance password. This is only if the root user has a password.


If you are running mysql using XAMPP, you don't need to set an env variable to set the password. Instead, navigate to the application.properties file which is inside of the resources folder found here: \backend\src\main\resources. You should see something like this:

```
1    spring.application.name=CourseBrief
2
3    #configuration
4    spring.jpa.hibernate.ddl-auto=update
5    spring.datasource.url=jdbc:mysql://localhost:3306/coursebrief_db
6    spring.datasource.username=root
7    # spring.datasource.password=${DB_PASSWORD}
8    spring.datasource.password=
9
10   spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

In the current repo, the password with the environment variable has been already commented out. In this case, you would be ready to run the server and connect it to your local database. Vice Versa if you have a password for the root.

Now, you should be ready to run the application.

1. First go to the MYSQL shell and cd to the location of the project like shown below:

 Administrator: XAMPP for Windows

```
Setting environment for using XAMPP for Windows.  
Ghost@DESKTOP-10107PC c:\xampp  
# cd C:\\Users\\Ghost\\CS157A-CourseBrief  
  
Ghost@DESKTOP-10107PC C:\\Users\\Ghost\\CS157A-CourseBrief  
#
```

a.

2. Log into the mysql server:

```
Ghost@DESKTOP-10107PC C:\\Users\\Ghost\\CS157A-CourseBrief  
# mysql -u root -p  
Enter password:  
Welcome to the MariaDB monitor.  Commands end with ; or \g.  
Your MariaDB connection id is 8  
Server version: 10.4.28-MariaDB mariadb.org binary distribution  
  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
MariaDB [(none)]> _
```

a.

3. Run the cmd source createdb.sql to create the database:

```
MariaDB [(none)]> source createdb.sql  
Query OK, 1 row affected (0.001 sec)  
  
Database changed  
MariaDB [coursebrief_db]>
```

a.

4. After creating the database, run the spring boot server by navigating to
 \backend\src\main\java\com\CS157AProject\CourseBrief\CourseBriefApplication
 .java and running the file:
5. To check if the server is running, go back to the shell and do the following
 command: SHOW TABLES. You should see the following:

```
MariaDB [coursebrief_db]> show tables;
+-----+
| Tables_in_coursebrief_db |
+-----+
| action_course            |
| action_tag              |
| admin                   |
| class                   |
| comment                 |
| course                  |
| course_tag              |
| professor               |
| starred_course          |
| tag                     |
| user                    |
+-----+
11 rows in set (0.000 sec)

a. MariaDB [coursebrief_db]> _
```

6. Run the cmd source insertStatements.sql to populate the admin, class, tag,
 professor, courseTags and course tables:

```

MariaDB [coursebrief_db]> source insertStatements.sql
Query OK, 1 row affected (0.002 sec)

Query OK, 46 rows affected (0.002 sec)
Records: 46  Duplicates: 0  Warnings: 0

Query OK, 399 rows affected (0.003 sec)
Records: 399  Duplicates: 0  Warnings: 0

Query OK, 72 rows affected (0.001 sec)
Records: 72  Duplicates: 0  Warnings: 0

Query OK, 39 rows affected (0.002 sec)
Records: 39  Duplicates: 0  Warnings: 0

Query OK, 76 rows affected (0.002 sec)
Records: 76  Duplicates: 0  Warnings: 0

a. MariaDB [coursebrief_db]> _

```

7. Now head to your IDE's terminal where we will start up the project on the client side. Once you are in the terminal, cd to frontend/coursebrief like shown below:

```

PS C:\Users\Ghost\CS157A-CourseBrief> cd frontend/coursebrief
PS C:\Users\Ghost\CS157A-CourseBrief\frontend\coursebrief>

```

- a.
- b. Make sure that you are in the root folder before cding
8. Once you are inside of the coursebrief directory, run the cmd npm install to install the packages and dependencies needed to run the application:

```

PS C:\Users\Ghost\CS157A-CourseBrief\frontend\coursebrief> npm install

added 18 packages, and audited 332 packages in 3s

109 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\Ghost\CS157A-CourseBrief\frontend\coursebrief>

```

- a.
9. To start the server, run the cmd npm run dev. It should give a localhost link. Click on that to see the application:


```
PS C:\Users\Ghost\CS157A-CourseBrief\frontend\coursebrief> npm run dev

> coursebrief@0.0.0 dev
> vite

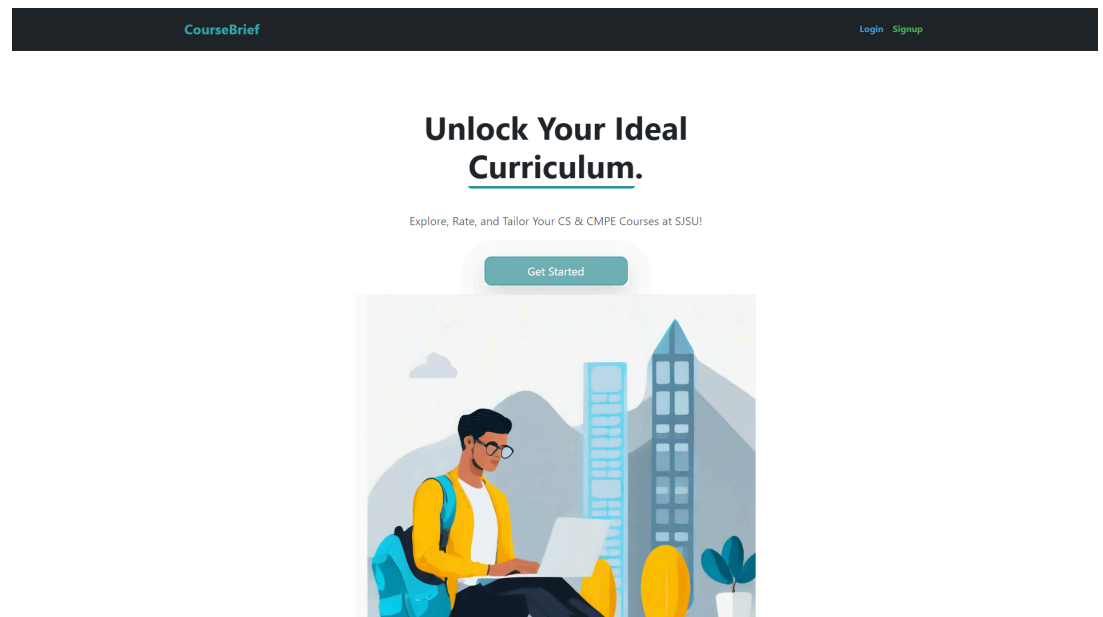
✓ Console Ninja extension is connected to Vite, see https://tinyurl.com/2vt8jxzw

VITE v5.2.9 ready in 1641 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

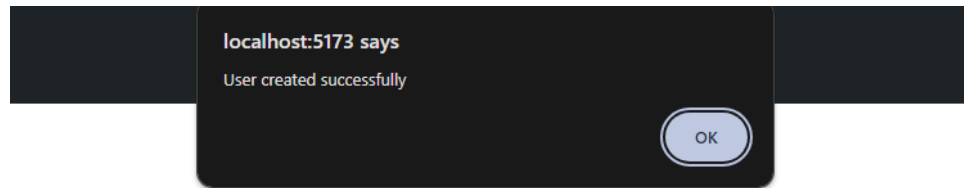
a.

10. You will be prompted to the landing page of the application. You can sign up or login by clicking on the navbar at the top:



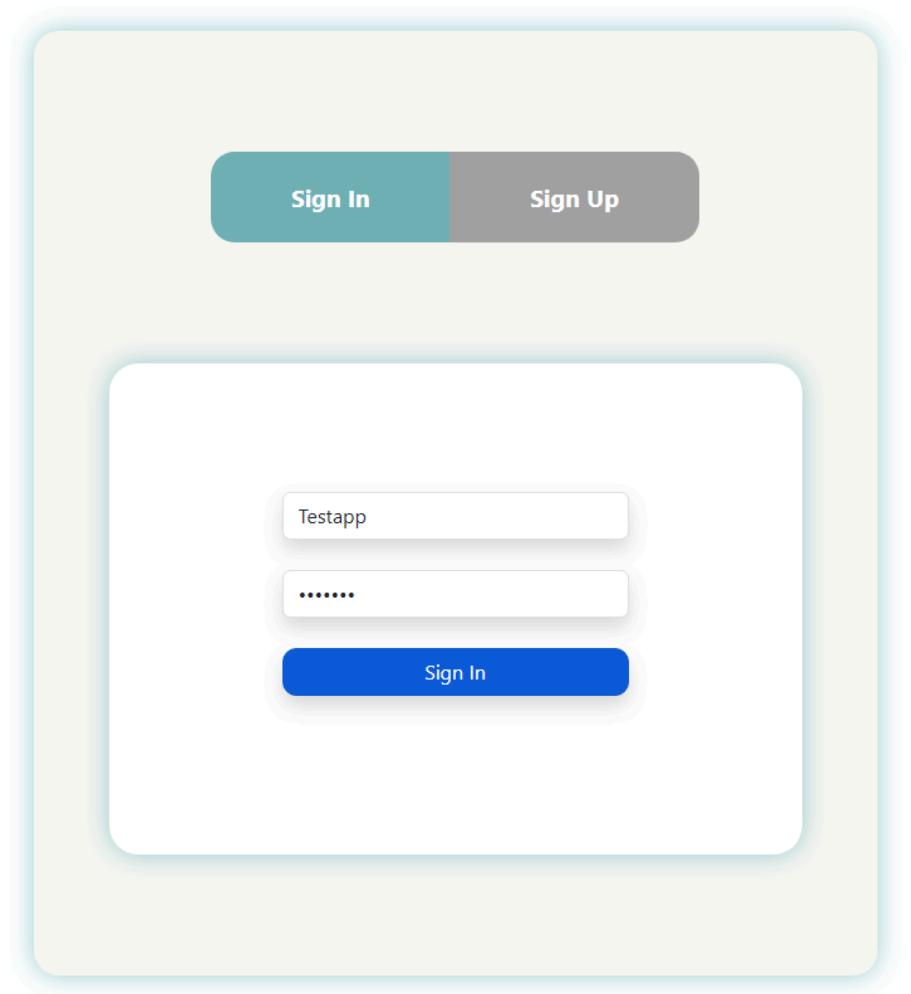
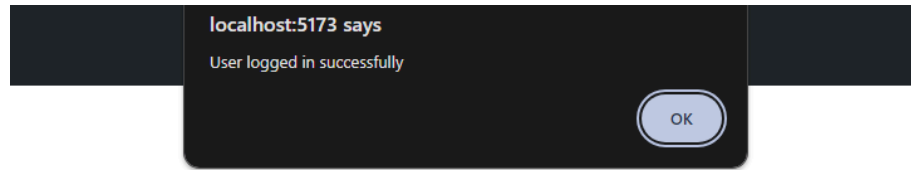
a.

11. This is the sign up page. A popup will show after if you were able to sign up successfully

A user registration form is shown on a light beige background. At the top, there are two buttons: "Sign In" (teal) and "Sign Up" (grey). Below these is a white rounded rectangle containing the registration fields. The fields are: "Test" and "App" (small input boxes), "TestApp" (a single-line input box), "testapp@gmail.com" (a single-line input box), and a password field (a single-line input box with dots). At the bottom of the white rectangle is a blue "Sign Up" button.

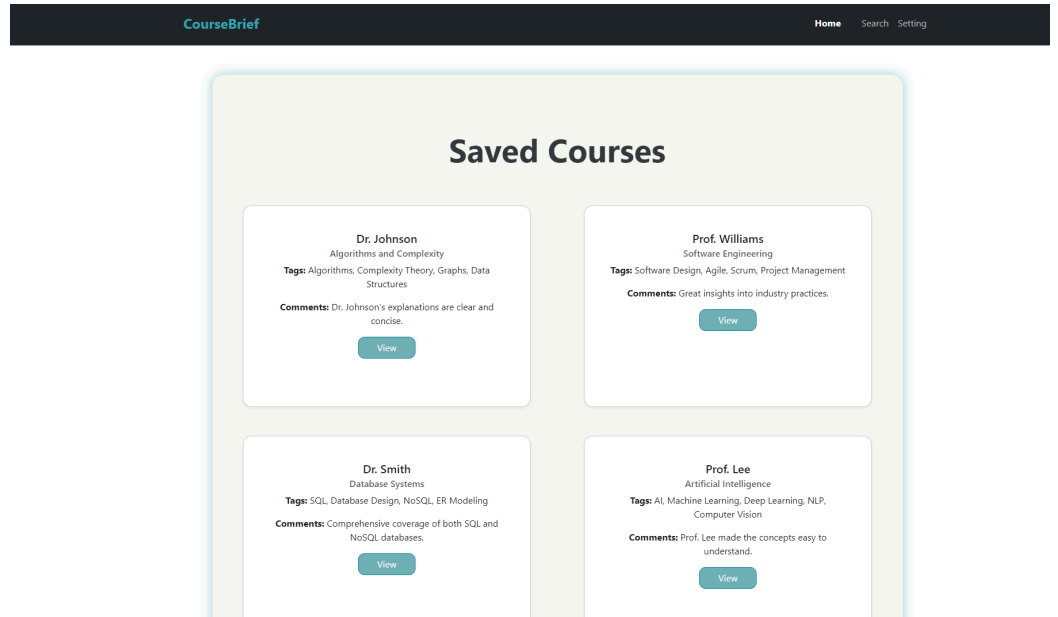
a.

12. After signing up, it redirects you to the login page:



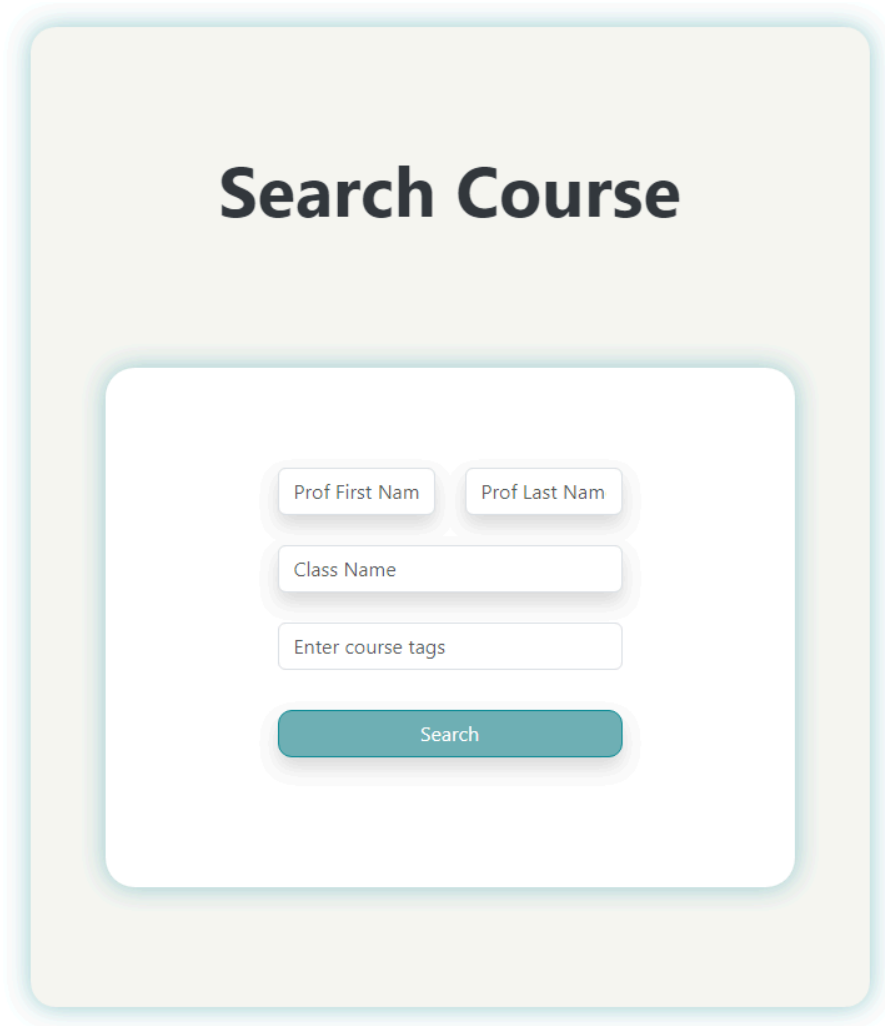
a.

13. After logging in, you will be redirected to the homepage



a.

14. To search for courses, click on the search button on the navigation bar. It will prompt you to the search page.

A screenshot of a web form titled "Search Course". The form is centered on a light beige background with a subtle blue glow. It contains four input fields: "Prof First Nam", "Prof Last Nam", "Class Name", and "Enter course tags". Below these fields is a teal "Search" button.

Search Course

Prof First Nam

Prof Last Nam

Class Name

Enter course tags

Search

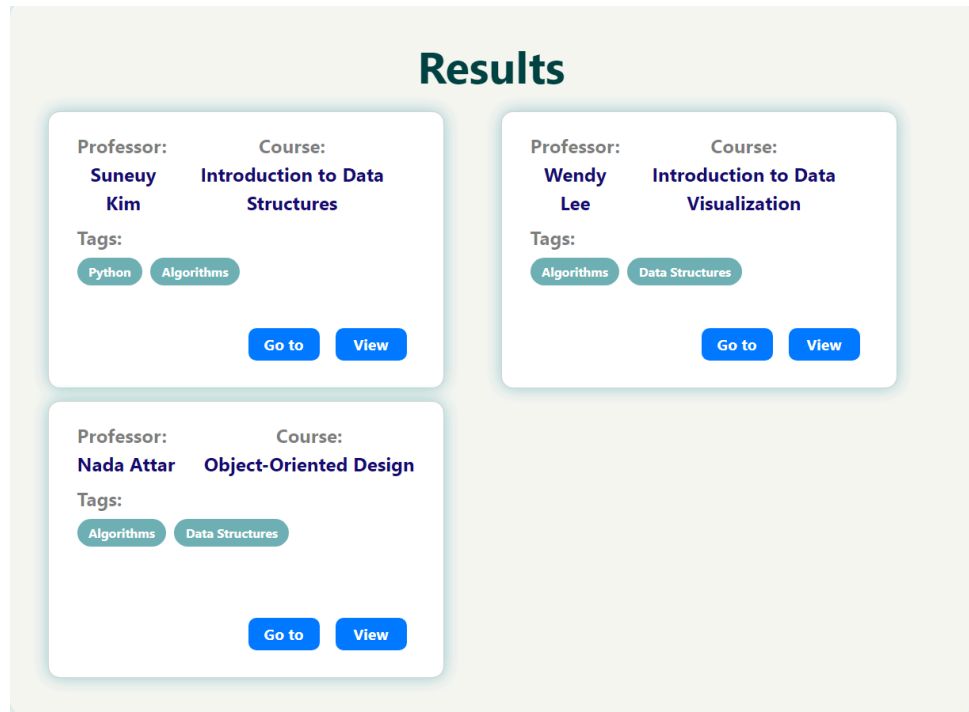
a.

15. In the search page, you are able to search for a course by providing the professor name, class name, or specific tags/keywords. For example, if I want to see courses with the algorithms tag, I would enter it in the tags like shown below:

Search Course

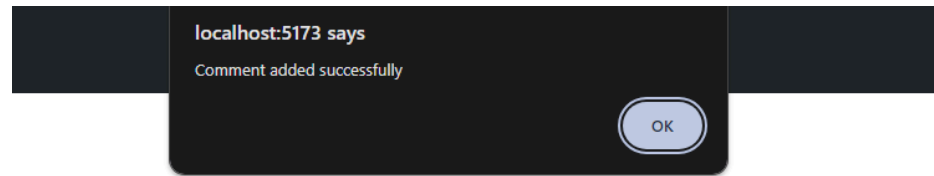
a.

16. After clicking search, it would show all courses with that tag:



a.

17. If you want to leave a comment on a course, you can click on the Go to button which would prompt you to the comment page for that specific course:



Course Page for course COURSE22

Leave a Comment:

Good class

Submit

Comments

No comments yet. Be the first to comment!

a.

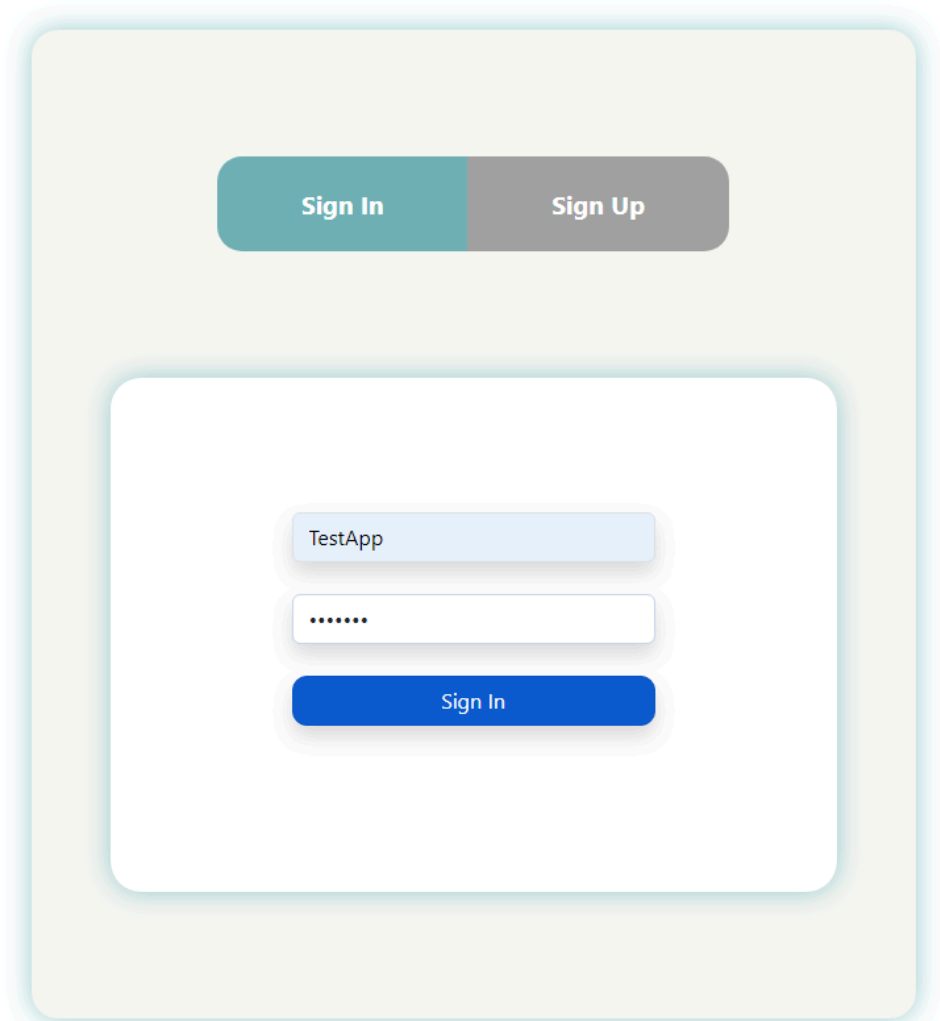
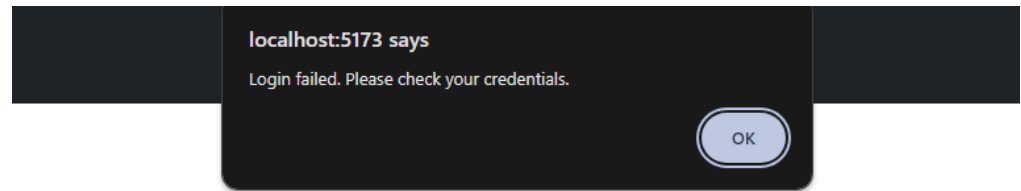
18. If you want to delete your account, you can go to the settings page on the navbar:

Setting

Log Out

Delete Account

a.



b.

19. Now moving towards the admin part of our application. To do this, stop running the CourseBriefApplication file. Then navigate to `\backend\src\main\java\com\CS157AProject\CourseBriefAdmin\CourseBriefAdmin.java` and run the file. You should see the following if it was successful:

[illegible]

a.

a.

```

shell:>course
Would you like to add or delete a course? Enter 'add' to add, and 'delete' to delete: delete
In order to delete a course, you must be logged in as an admin. Please enter your credentials.
Enter your username: ntk-admin
Enter your password: adminpassword
Enter the classID of the class: CLASS534
Enter the professorID of the professor: PROF3
Error: invoke 'com.cs157.projects.courseRepository.StarredCourseRepository.deleteStarredCourseByCourseId(String)' because "this.starredCourseRepository" is null
Details of the error have been omitted. You can use the stacktrace command to print the full stacktrace.
shell:>

```

b.

a.

```
shell:>tag
Would you like to add or delete a tag? Enter 'add' to add, and 'delete' to delete: delete
In order to delete a tag, you must be logged in as an admin. Please enter your credentials.
Enter your username: nick-admin
Enter your password: adminpassword
Enter the tagID of the tag to delete: TAG401
Tag with tagID TAG401 has been deleted successfully.
shell:>
```

b.

Conclusion

Overall, this project was a good learning experience for all of us. We were able to apply the concepts taught in class into our database design and also it gave us new frameworks and areas to learn more about. There were some challenges that we faced along the way but in the end we were able to get a working product that implements most of the features that we listed in the proposal. Some of the challenges that we faced were handling the HTTP requests particularly the POST and GET methods. Since some members of the team were using Spring boot for the first time, it took a little while to get the hang of it, fixing CORS issues, etc. Once we got them all sorted out, the request handling went more smoothly. Another challenge would be figuring out the correct query to fetch the correct courses based on the user's input through the search page. Since we had it if the user didn't enter anything in the fields, it would just display all the courses in the database. However we had trouble creating a query that would correctly output the courses based on the user inputs. After a lot of time spent brainstorming, we were able to come up with a query that correctly queried the courses based on the user's inputs in the search form. Designing an Admin-Interface using a second Spring Boot Application was difficult as this included operations like adding and deleting to the database. This required handling and accounting for foreign key constraints that would need to be addressed upon manipulating tuples in the database.

If we were to come back to this project and work on improvements, we would implement the starred course feature and have it functional as we weren't able to get to that part within the timeframe. Also we would improve the user experience, making it more user friendly and easy to use and navigate around the website.