Project 6
CSCI 5448

CafèConnect

Jayant Duneja
Jayant.Duneja@colorado.edu

Tilak Singh
Tilak.Singh@colorado.edu

November 29, 2023

Status Summary:

Work Accomplished:

1. Database Setup:
    - Established an AWS Cluster utilizing CockroachDB.
    - Configured the database by creating the following tables:
        - Cafe
        - Menu
        - Review
        - Student
    - The data models for these tables were consistent with those provided for Project 5
2. Connection to Database through SpringBoot Application:
    - Implemented connectivity to the database through a SpringBoot application.
    - Utilized a Configuration class to connect to the database by providing details such as username, password, and the link as a Datasource.
    - Employed a Singleton Pattern with Lazy Instantiation for the Datasource.
3. **MVC Pattern** Implementation for Interaction with Tables:
    - Adopted the MVC Pattern (Model-View-Controller) for interacting with the created tables.
    - Developed models, views, and controllers for each of the tables.
4. API Development:
    - Cafe API Endpoints:
        - api/cafe
            - GET Mapping: Retrieves all cafes in the application.
        - api/cafe/find/{cafeId}
            - GET Mapping: Retrieves the cafe object with the specified CafeId.
            - PUT Mapping: Updates the cafe element in the database with the specified CafeId.
        - api/cafe/add
            - POST Mapping: Creates a new cafe, with the CafeId automatically instantiated by the database.
        - api/cafe/delete/{cafeId}
            - DELETE Mapping: Removes a cafe with the specified cafeId.
    - Menu API Endpoints:
        - api/menu
            - GET Mapping: Returns the list of all menu items across all cafes.
        - api/menu/find/{cafeID}
            - GET Mapping: Returns the list of menu items for a specific cafeID.
        - api/menu/add

        - POST Mapping: Adds a new menu item for a particular cafe, with the cafeId passed in the request body.
    - api/menu/update/{ItemId}
        - PUT Mapping: Updates the menu item with the specified ItemId.
    - api/menu/delete/{ItemId}
        - DELETE Mapping: Deletes the menu item with the specified ItemId.

- Student API Endpoints:
    - api/student
        - GET Mapping: Returns the list of all students.
    - api/student/find/{studentId}
        - GET Mapping: Returns the student object with the specified studentId.
    - api/student/add
        - POST Mapping: Adds a new student. The studentId would be automatically instantiated by the database.
    - api/student/delete/{studentId}
        - DELETE Mapping: Removes a student with the specified studentId.
- Review API Endpoints:
    - api/review
        - GET Mapping: Returns the list of all reviews.
    - api/review/find/{reviewId}
        - GET Mapping: Returns the review object with the specified reviewId.
    - api/review/add
        - POST Mapping: Adds a new review. The reviewId would be automatically instantiated by the database.
    - api/review/delete/{reviewId}
        - DELETE Mapping: Removes a review with the specified reviewId.

5. Frontend development
    - We successfully developed a student-centric interface that enhances their interaction with campus cafes, empowering them to explore and discover improved food options.
    - To ensure a cohesive and aesthetically pleasing user experience, we adhered to design principles by strategically incorporating reusable patterns throughout the interface. This not only streamlined the development process but also contributed to a consistent and visually appealing design.
    - While user authentication through login functionality has not been implemented, we have established a designated route,` */student*`, to provide seamless access to the student view. This approach not only allows users to engage with the platform without barriers but also sets the foundation for a secure and personalized experience once login features are integrated in the future.
    - While interacting with the cafe, students can look for the food menu and provide the review for a particular cafe.
6. Division of Work:
    - Backend Development:
        - Student Flow: Tilak Singh
        - Menu Flow: Tilak Singh
        - Review Flow : Jayant Duneja
        - Cafe Flow : Jayant Duneja
    - Frontend Development:
        - /student : Jayant Duneja
        - /cafe/cafeId : Tilak Singh

Changes/Issues Encountered:

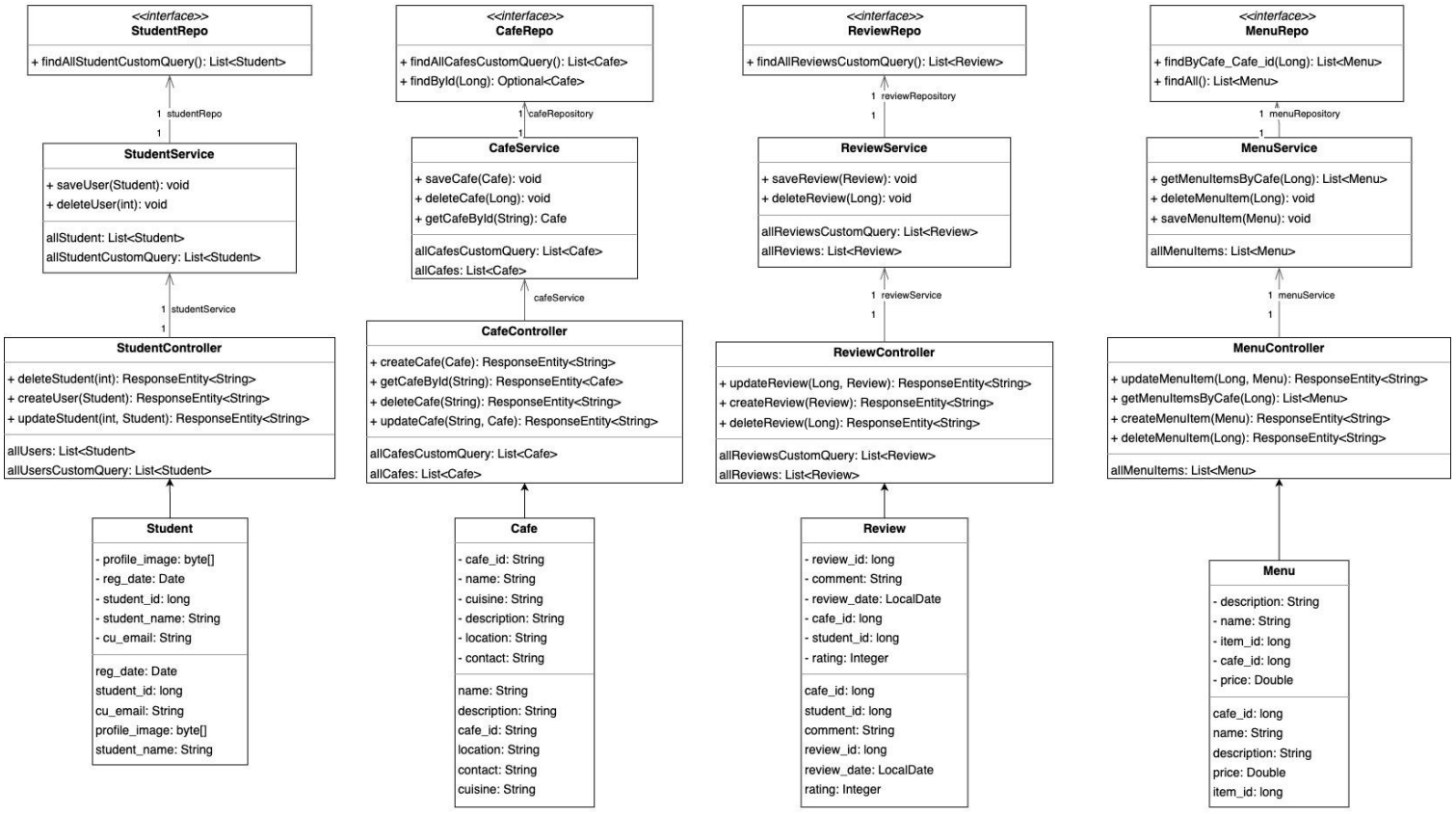
- We had specified that we would be using Next.js for the frontend components, but we transitioned to React.js since we were a bit more familiar with it and we did not have enough time to learn about Next.js
- We haven't implemented the login page for the students as of yet. Due to this, giving a review would not be possible using the UI. We can implement this using Postman since we send the Student Id in the body

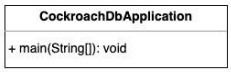
Patterns Used:

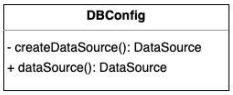
1. Singleton Pattern:
    - Purpose: The Singleton Pattern was employed in the connection to the database through the SpringBoot application.
    - Usage: The Singleton Pattern ensures that a class has only one instance and provides a global point of access to that instance. In this context, it was used for Lazy Instantiation of the Datasource, meaning that the connection to the database is created only when it is first needed.

2. MVC Pattern (Model-View-Controller):
    - Purpose: MVC was implemented for interacting with the tables (Cafe, Menu, Review, Student) in the database.
    - Usage:
        - Model (M): Represents the data and business logic of the application. In this case, models were created for each table (Cafe, Menu, Review, Student), defining the structure of the data and any necessary business logic.
        - View (V): Represents the user interface or the presentation layer. We used React.js to show the UI and for our project 6, we have implemented the `/student` route to show the student view.
        - Controller (C): Manages user input and updates the model accordingly. Controllers were implemented for each table to handle the logic of processing requests and updating the data models.

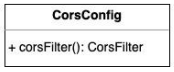
UML:







Singleton Pattern for DB Configuration



Future Work:

- Sending the notification and friend request will be our next task on the roadmap.
- We will incorporate Factory and Observer patterns.
- In our next release we will have a cafe view as well where owners will be able to manage and control their business and send notification to promote their offers.