**Design Document for <<Cyclone Connect>>**

**Group MU3\_7**

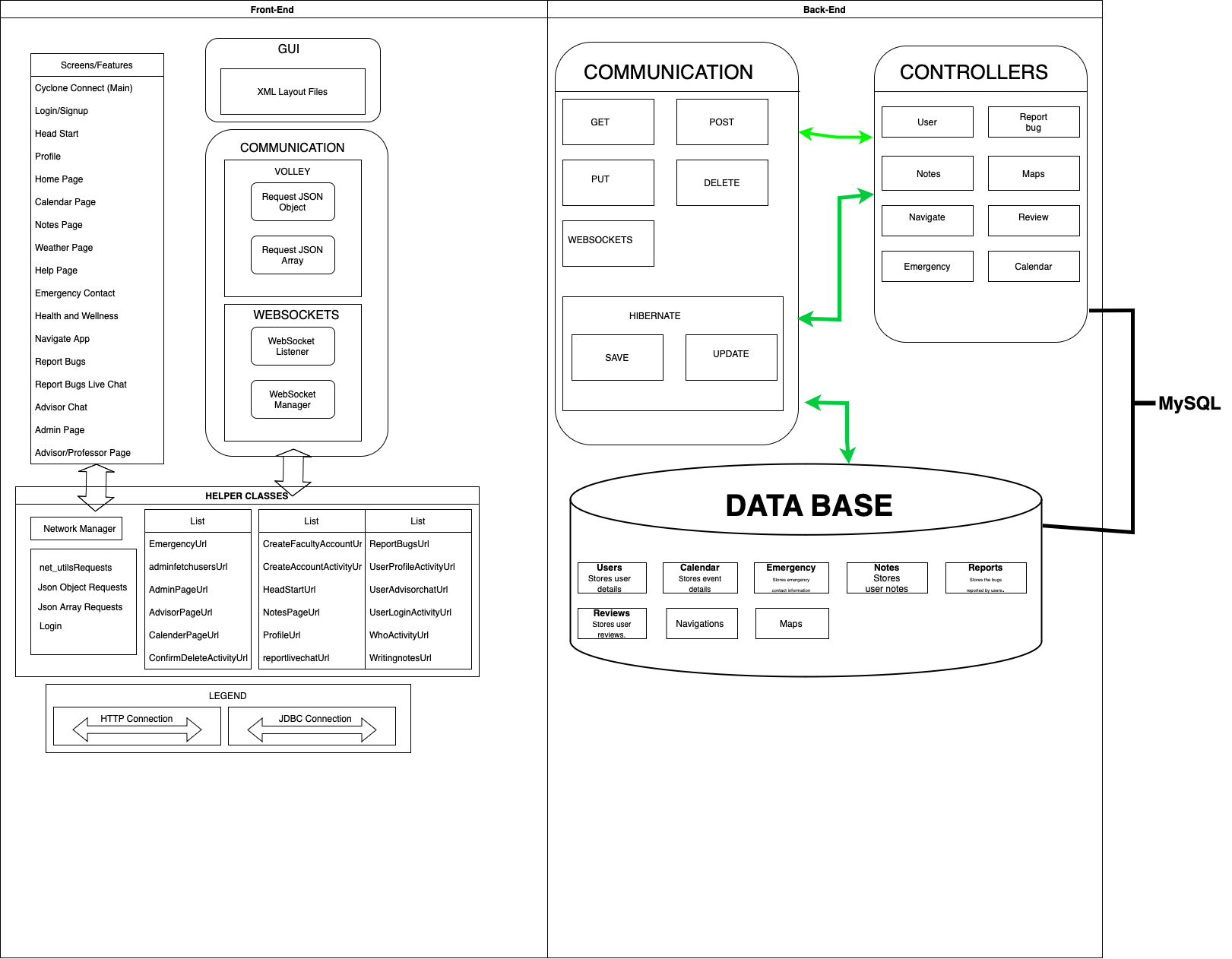
**Abhay Prasanna Rao:** Frontend [Description/working of the app, Views: Cyclone Connect Main Page, Login/Signup, Profile, Weather Page, Help Page, Emergency Contacts, Health and Wellness, Navigate App, Report Bugs and Bugs Chat, Admin Page etc; Block Diagram ( Frontend: Views, GUI, Legend), API doc generation]

**Meghasyam Peddireddy:** Frontend [User Roles (Students, Professors, Advisors and Administrators), Login, Home Page, Head Start] Block Diagram (Frontend: GUI, Communication, Helper Classes), API Doc generation

**Neha Tirunagari:** Block Diagram (Backend and Database), Relationship Diagram, Backend Design Descriptions(Users, Emergency, Maps, Navigate, Review, Report bugs, Advisor Chat, Notes, Calendar, Home Page), Relationship Diagram, API Docs

**Umesh Sai Teja Poola:** Block Diagram (Backend and Database), Relationship Diagram, Backend Design Descriptions(Users, Notes, Calendar, Advisor chat, chat message), Relationship Diagram, API Docs(Swagger)

PUT THE BLOCK DIAGRAM PICTURE ON THIS PAGE! (Create the picture using a pencil or drawing)



**FRONT-END API:**

1. Account Type Holder
2. Admin Bugs Chart
3. Admin\_Emergency\_addcontacts
4. admin\_EmergencyContactUpdatesActivity
5. Admin\_fetchusers
6. AdminPage
7. ConfirmDeleteActivity
8. Create\_Faculty\_Account
9. CreateAccountActivity
10. EmergencyContacts
11. health and wellness
12. Help\_page
13. NavigateApp
14. Profile
15. Report\_live\_chat
16. ReportBugs
17. UpdateProfileActivity
18. UserLoginActivity
19. Weather\_Page
20. WhoActivity
21. Writing Notes
22. WebSocketManager
23. WebSocketManager1
24. WebSocketListener
25. VolleySingleton
26. User\_Advisor\_chat
27. Notes\_page
28. MainActivity
29. HomePage
30. Headstart
31. Calendar\_page
32. AdvisorView
33. AdvisorPage

**FRONT-END Design Description:**

DESCRIPTION/WORKING OF THE APP:

● Cyclone Connect starts with a welcoming splash screen, presenting users with options to either sign up or log in. New users are directed to a signup page to create an account while existing users can log in to access the app’s features.

* New users choosing to sign up can select their role (student, faculty, or admin) before being prompted to enter personal information (email, first name, last name, and password) and create their account. Account details are then stored in the backend.
* Existing users can log in by entering their registered email and password. There’s an option to "Remember Me" for storing login credentials on the frontend for easier access in the future.

● Upon successful login, users are navigated to the app’s homepage, which serves as a central hub for accessing various features like Head Start, weather, calendar, notes, help, and advisor chat. The homepage also allows users to update their profiles or navigate to specific views based on their role (advisor/professor or admin).

● The Head Start feature helps users select their major and class number to get a head start on specific courses.

● The Weather feature displays current location weather, including temperature, condition, wind speed, and max/min temperatures for the day.

● The Calendar feature allows users to view, add, or update their schedule, encompassing classes, work, meetings, and events all in one view.

● The Notes feature enables creating tasks, writing notes, and saving them in the backend for planning and reminders.

● The Help feature offers access to emergency contacts, navigation tips, bug reporting, live chat support, and map services for route finding and reviews.

● The Advisor Chat connects users with advisors for confidential discussions about course planning and academic advice.

● The Profile view allows users to manage account details and access additional functionalities depending on their role. Advisors can view and edit courses and calendars and initiate chats with students. Admins can update or delete app features.

VIEWS:

SPLASH SCREEN

● A welcoming screen offering "Login" or "Sign Up" options, guiding users based on their account status.

SIGN UP & ACCOUNT CREATION

● Users new to the app are guided to create an account, choosing their role and entering personal information to set up their account. The process includes form fields for email, name, password, and a button to finalize account creation.

LOGIN

● A straightforward login page where returning users enter their email and password. Options for "Remember Me" and password recovery enhance user convenience.

HOMEPAGE

● The central hub after login, presenting features like Head Start, weather, calendar, notes, and help. The homepage adapts to user roles, offering additional functionalities to advisors and admins.

HEAD START

● A feature allowing users to select their major and class to prepare for courses, accessed from the homepage.

WEATHER

● Displays current weather conditions based on the user’s location, accessible from the homepage.

CALENDAR

● Enables users to view and manage their schedule, including academic and extracurricular activities, from the homepage.

NOTES

● A tool for creating tasks and notes, with the ability to save and manage them for academic planning.

HELP & SUPPORT

● Offers access to emergency contacts, navigation help, bug reporting, live chat support, and map services for route optimization and feedback.

ADVISOR CHAT

● Connects users with advisors for confidential academic planning and course discussion.

PROFILE MANAGEMENT

● Allows users to view and update their profile details, with role-specific functionalities for advisors and admins to manage courses, calendars, and app features.

**USER ROLES:  
  
Students:**

* Users who want to access course topics and roadmaps can sign up as students.
* The signup form includes First name, Last name and email ID, password, and Password confirmation.
* After Login, They can access the calendar, Notes, Weather, Help, Advisor Chat, HeadStart, and Profiles features.
* Users can report bugs and have a live chat with admins, depending on the availability.

**Professors:**

* Users who want to sign up as professors need to select the Professor icon while signing up, which includes first name, Last name and email ID, password, and Password confirmation.
* They will need authentication from Admin.
* After Login, They can access the calendar, Notes, Weather, Help, HeadStart and Profiles features.
* They have the ability to add course topics in HeadStart.
* Users can report bugs and have a live chat with admins, depending on the availability.

**Advisor:**

* Users who want to sign up as Advisors need to select the Advisor icon while signing up, which includes first name, Last name and email ID, password, and Password confirmation.
* They will need authentication from Admin.
* After Login, They can access the calendar, Notes, Weather, Help, Advisor Chat, HeadStart and Profiles features.
* They have the ability to chat with students.
* Users can report bugs and have a live chat with admins, depending on the availability.

**Administrator:**

* The Administrator role includes special privileges, like authenticating the users.
* Administrators can log in directly.
* The admin role is not visible to other users. Admins can resolve the bugs reported by users.

**Login:**

* After Creating an account, Users can log in using their email and password.
* After login, Users will be directed to the homepage.
* The HomePage is different based on your role in the application.

**Home Page:**

* Most HomePage comprises features like Calendar, notes, Weather, and Help.
* There will be additional features depending on your role. Students will have an advisor chat, and Advisors will have a user chat.
* Users can access their calendars to create an event. They can get their topics as checkboxes on the notes page. They can check the weather and access the help tab to access the university’s emergency services and report bugs.
* They can access their profile information in the HomePage.

**HeadStart:**

* The HeadStart Page consists of the course information, topics, and roadmaps needed to ace a course.
* Users can search for a specific course and import the course topics as checkboxes to the notes.
* Professors can add the course information and topics.

Use this third page to describe complex parts of your design.

**BACKEND:   
Calendar Controller**

**Class Description:**

- The CalendarController class manages HTTP requests related to calendar events within the application. It is designed to handle operations such as creating, updating, deleting, and fetching events from the database.

**Fields**:

- `eventRepository`: An instance of the EventRepository class, enabling access to event data.

**Methods**:

- `createEvent`: Adds a new event to the calendar. Accepts event details and returns a success or failure message.

- `updateEvent`: Updates an existing event with new details. Requires event ID and updated details.

- `deleteEvent`: Removes an event from the calendar based on its unique ID.

- `getAllEvents`: Retrieves a list of all calendar events.

- `getEventById`: Fetches a single event by its ID.

**Emergency Controller**

**Class Description:**

- The EmergencyController class is responsible for managing emergency contact information in the application, allowing users to add, update, delete, and retrieve emergency contact details.

Fields:

- `emergencyContactRepository`: Facilitates access to emergency contact data.

**Methods**:

- `addEmergencyContact`: Stores a new emergency contact in the database.

- `updateEmergencyContact`: Modifies details of an existing emergency contact.

- `deleteEmergencyContact`: Removes an emergency contact based on its ID.

- `getEmergencyContacts`: Lists all emergency contacts associated with a user.

**Map Controller**

**Class Description:**

- The MapController class provides functionality for navigation and location-based services within the application, such as fetching directions and points of interest.

**Fields:**

- `mapService`: An instance of a service class that interfaces with external map APIs.

**Methods**:

- `getDirections`: Offers routing information between two points.

- `searchLocation`: Allows searching for locations or points of interest.

**Review Controller**

**Class Description:**

- The ReviewController handles user reviews for services or products offered within the application, including operations to add, update, delete, and retrieve reviews.

**Fields:**

- `reviewRepository`: Access to review data.

**Methods:**

- `postReview`: Allows a user to create a new review.

- `updateReview`: Modifies an existing review.

- `deleteReview`: Removes a review.

- `getReviewsByProduct`: Retrieves all reviews for a specific product or service.

**Report Controller**

**Class Description:**

- The ReportController manages reports or feedback submitted by users, including creating reports, updating their status, and querying reports from the database.

**Fields:**

- `reportRepository`: Access to report data.

**Methods**:

- `submitReport`: Users can report an issue or feedback.

- `updateReportStatus`: Modifies the status of an existing report.

- `getAllReports`: Fetches a list of all reports.

**User Controller**

**Class Description:**

- The UserController is focused on user management within the application, including user registration, profile updates, authentication, and account deletion.

**Fields**:

- `userRepository`: Access to user profiles and data.

**Methods**:

- `registerUser`: Handles new user registration.

- `updateUserProfile`: Updates user profile information.

- `authenticateUser`: Manages user login and authentication.

- `deleteUserAccount`: Removes a user's account from the application.

**Navigation Controller**

**Class Description:**

- The NavigationController class provides functionality related to navigating within the application, including fetching directions, points of interest, and traffic updates. It interfaces with external mapping and navigation services to provide users with real-time navigation assistance.

**Fields:**

- `navigationService`: An instance of a service class that interfaces with external navigation APIs to fetch routing and location information.

**Methods:**

- `getDirections`: Provides routing information between two or more points. Accepts start and end locations along with optional parameters like mode of transportation.

- `getTrafficUpdate`: Offers real-time traffic updates for a specified route or location.

- `getPointsOfInterest`: Retrieves points of interest near a specified location or along a route. Can include filters for types of places.

**Notes Controller**

**Class Description:**

- The NotesController manages operations related to creating, updating, deleting, and retrieving notes within the application. This controller allows users to manage their personal notes or annotations, which can be linked to various entities within the application (e.g., calendar events, saved locations).

**Fields:**

- `notesRepository`: An instance of the NotesRepository class, facilitating access to notes data stored in the database.

**Methods:**

- `createNote`: Adds a new note. Accepts note content along with any associated metadata (e.g., related event or location).

- `updateNote`: Updates the content or metadata of an existing note. Requires the note's ID and the new content or metadata.

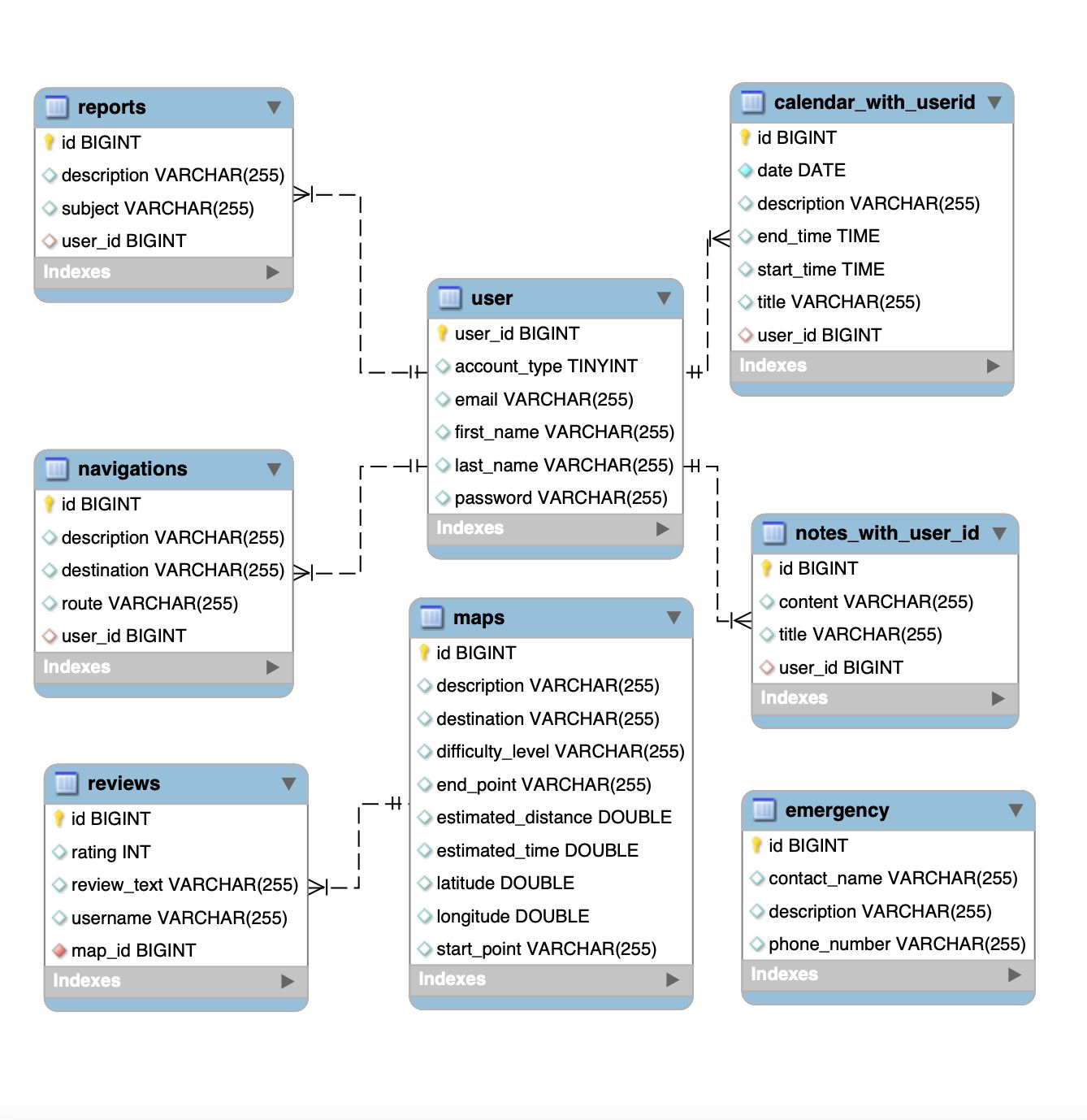
- `deleteNote`: Removes a note from the system based on its unique ID.

- `getAllNotes`: Retrieves a list of all notes stored in the application, with optional filters based on user, date, or associated entity.

- `getNoteById`: Fetches a single note by its ID, providing details of the note's content and any related metadata.

PUT THE TABLE RELATIONSHIPS DIAGRAM on this fourth page! (Create the picture using MySQLWorkbench)

**RELATIONSHIP DIAGRAM**



**SWAGGER API**

[**http://coms-309-046.class.las.iastate.edu:8080/swagger-ui/index.html#/**](http://coms-309-046.class.las.iastate.edu:8080/swagger-ui/index.html#/)