Design Document for GlobeGatherer

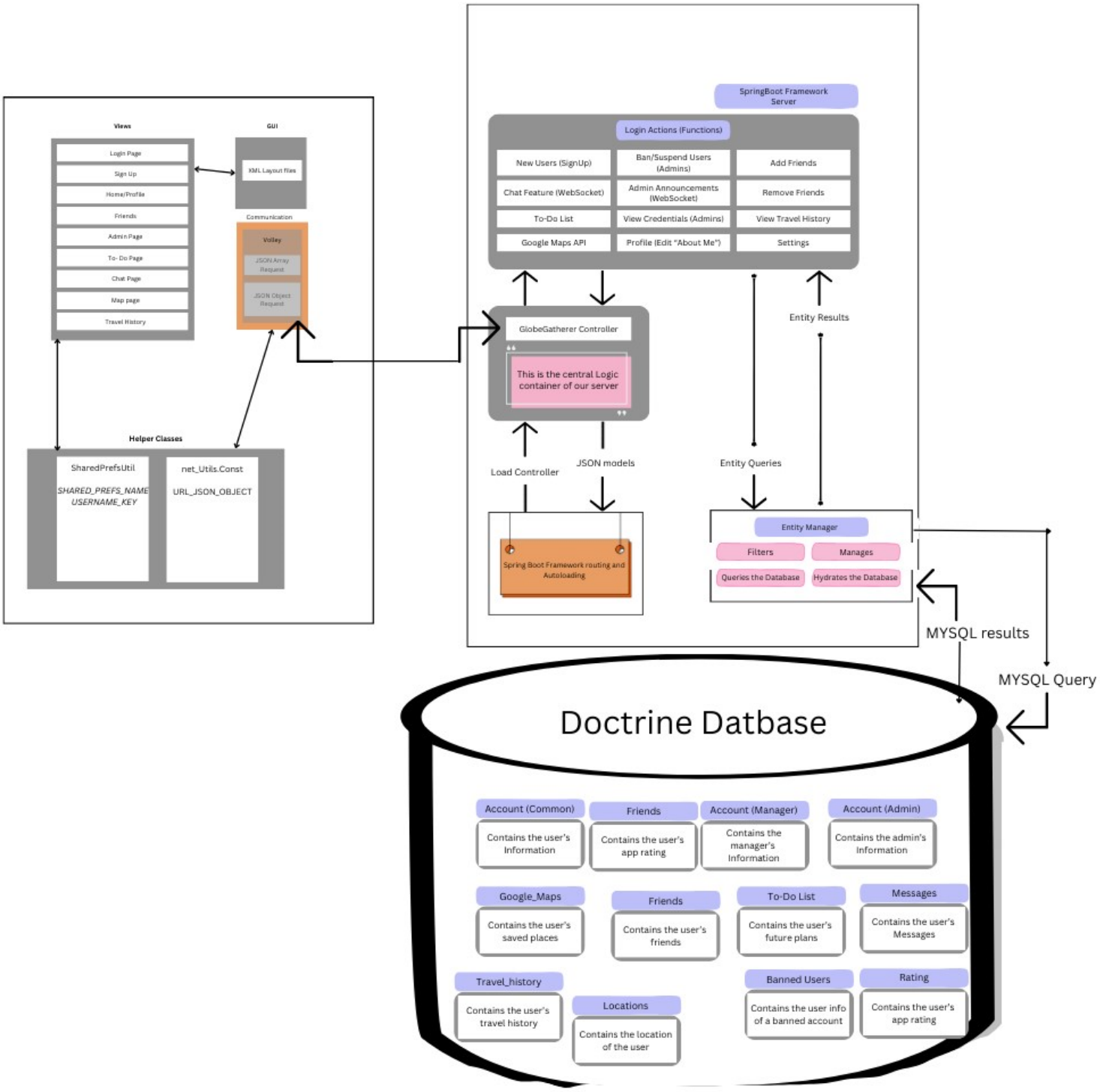
Group <IP-107>

Raghuram Guddati: 25% contribution

Tanvi Mehetre: 25% contribution

Ella Cook: 25% contribution

Tejal Debshetwar: 25% contribution


Views
Login Page
Sign Up
Home/Profile
Friends
Admin Page
To- Do Page
Chat Page
Map page
Travel History
GUI
XML Layout files
Communication
Volley
JSON Array Request
JSON Object Request
Helper Classes
SharedPrefsUtil
SHARED_PREFS_NAME
USERNAME_KEY
net_Utils.Const
URL_JSON_OBJECT
SpringBoot Framework Server
Login Actions (Functions)
New Users (SignUp)
Ban/Suspend Users (Admins)
Add Friends
Chat Feature (WebSocket)
Admin Announcements (WebSocket)
Remove Friends
To-Do List
View Credentials (Admins)
View Travel History
Google Maps API
Profile (Edit "About Me")
Settings
GlobeGatherer Controller
This is the central Logic container of our server
Entity Results
Load Controller
JSON models
Entity Queries
Spring Boot Framework routing and Autoloading
Entity Manager
Filters
Manages
Queries the Database
Hydrates the Database
MYSQL results
MYSQL Query
Doctrine Datbase
Account (Common)
Contains the user's Information
Friends
Contains the user's app rating
Account (Manager)
Contains the manager's Information
Account (Admin)
Contains the admin's Information
Google_Maps
Contains the user's saved places
Friends
Contains the user's friends
To-Do List
Contains the user's future plans
Messages
Contains the user's Messages
Travel_history
Contains the user's travel history
Locations
Contains the location of the user
Banned Users
Contains the user info of a banned account
Rating
Contains the user's app rating

Frontend :

Announcement:

- Admin can send in announcements to any user. Websocket implemented in this feature.
    - EditText: announce_text
    - Button: connect
    - Button: send
- On clicking the connect button you can connect in, and then type the announcement to send. Hit the send button so as to send a message to the specific user.

View Credentials:

- Admin can view the username and password of all the users.
    - EditText: UsernameEdit
    - Button: View
- On clicking the view button we can see the list of all the users with the username and password.

Map Page:

- This page lets you search any place in the world and a pin is visible. On clicking the pin you can go to google maps and search for start and end destinations. It will give you the route by bus, walk, car, etc.
    - EditText: locationInput
    - Button: search
- When the button is clicked you can see the pin visible on the map above.

Backend :

Communication:

The backend employs mappings to update the database by processing information sent to the designated URLs. These mappings encompass:

- **Post:** Used to dispatch information about an item for addition to the database.
- **Get:** Employed to request information, often accompanied by an identifier specifying the particular item sought from the database.
- **Put:** Utilized to send information for updating a specific item in the database.

- **Delete:** Employed to dispatch an identifier, triggering the removal of a specific item from the database.

Controllers:
Within the controllers lie the mappings facilitating communication between the frontend and the database. These controllers include:

- **Person:** Encompasses the mappings mentioned above for creating users, and establishing one-to-one relationships with the following types: travel-to-do, travel_history, description, ratings, and admin users.
- This class has a sub-class called 'LOGIN REQUEST'. The controller also includes checking if the password and username entered are correct and lets users switch between multiple accounts.

- **Admins:** Introduces specific variables tailored for ADMINS, such as the option to ban a user, view credentials of a particular person, send announcements to every user that exists, and is linked through the Person table.

- **Profile:** Analogous to customer controllers, it introduces additional variables like 'AboutMe,' i.e., to add a small description about yourself and to edit or update it.

- **Ratings:** Exhibits a one-to-many relationship with Persons. Customers can leave ratings (stars out of 5) and leave a comment about the app experience, and these reviews are stored in the reviews controller.

- **Google-Maps:** Similar to ratings, with a one-to-many relationship where customers can mark favourite Locations for swift access on the frontend.

- **Friends:** Exhibits a many-to-many relationship with Persons. This controller lets users search users that exist and add friends to themselves. This feature will let the other user that was added as a friend be the friend of the current user. So, this is a two-way process. Example: A is a friend of B, so B is also a friend of A.

- **SearchHistories:** Exhibits a many-to-one relationship with Persons. This controller saves the search histories of the Locations that the user made.

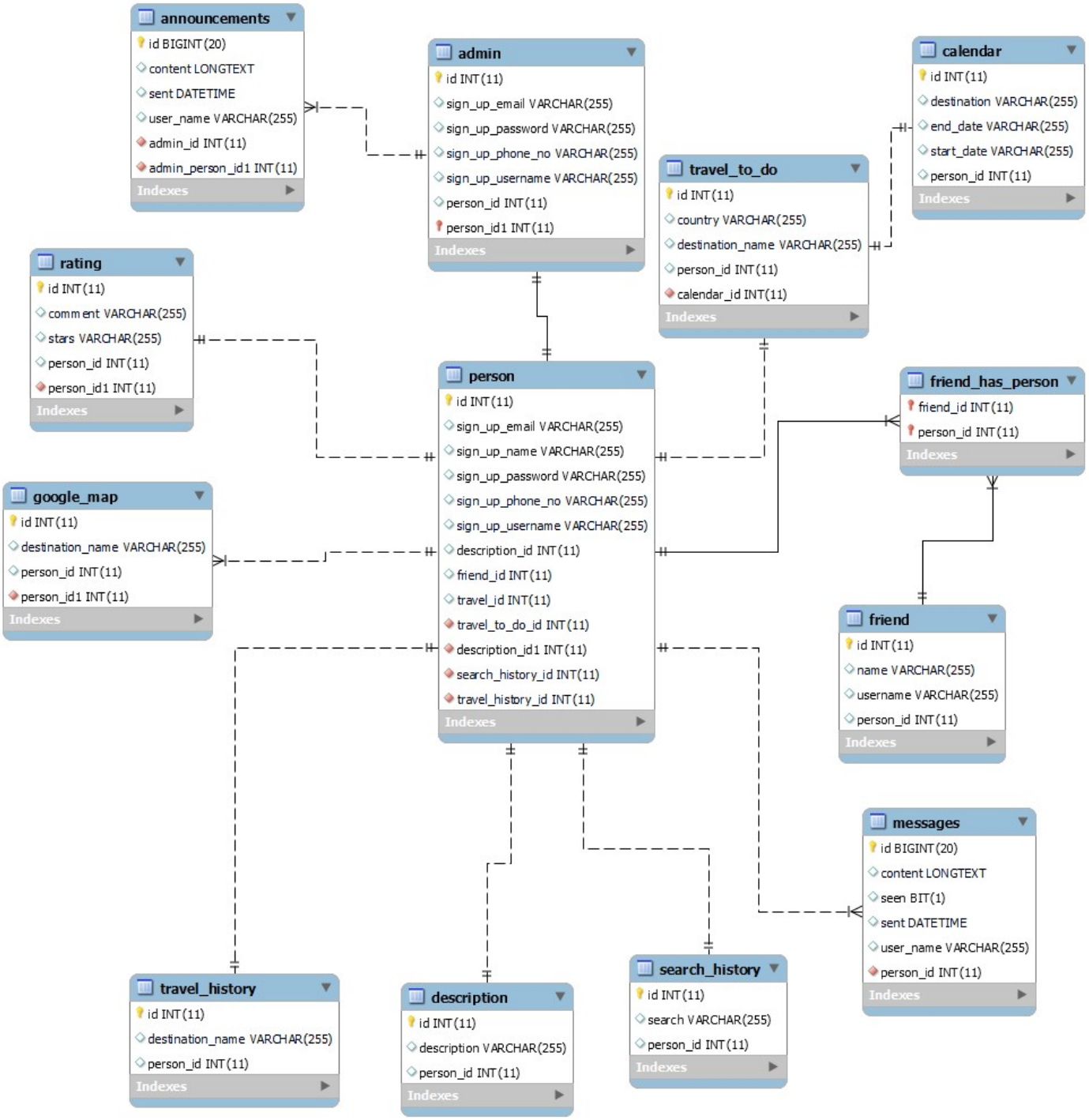
- **TravelHistories:** Exhibits a many-to-one relationship with Persons. This controller saves the past travels that the user made.
- **TravelToDos:** Exhibits a Many-to-one Relationship with Persons. This controller lets the users make a travel plan list for the future.

- **Calendar:** Exhibits a many-to-one relationship with TravelToDos. This controller lets the users add the start and end dates for their plans.

- **Texting:** This uses websockets to let users have a real-time chat and DM each other. This class is also responsible for the ADMIN announcements.

In essence, these controllers and their associated mappings form a robust system for managing interactions between the frontend and the database, ensuring seamless communication and efficient data handling.

Swagger Link:
http://coms-309-013.class.las.iastate.edu:8080/swagger-ui/#/


announcements
id BIGINT(20)
content LONGTEXT
sent DATETIME
user_name VARCHAR(255)
admin_id INT(11)
admin_person_id1 INT(11)
Indexes
admin
id INT(11)
sign_up_email VARCHAR(255)
sign_up_password VARCHAR(255)
sign_up_phone_no VARCHAR(255)
sign_up_username VARCHAR(255)
person_id INT(11)
person_id1 INT(11)
Indexes
calendar
id INT(11)
destination VARCHAR(255)
end_date VARCHAR(255)
start_date VARCHAR(255)
person_id INT(11)
Indexes
travel_to_do
id INT(11)
country VARCHAR(255)
destination_name VARCHAR(255)
person_id INT(11)
calendar_id INT(11)
Indexes
rating
id INT(11)
comment VARCHAR(255)
stars VARCHAR(255)
person_id INT(11)
person_id1 INT(11)
Indexes
person
id INT(11)
sign_up_email VARCHAR(255)
sign_up_name VARCHAR(255)
sign_up_password VARCHAR(255)
sign_up_phone_no VARCHAR(255)
sign_up_username VARCHAR(255)
description_id INT(11)
friend_id INT(11)
travel_id INT(11)
travel_to_do_id INT(11)
description_id1 INT(11)
search_history_id INT(11)
travel_history_id INT(11)
Indexes
friend_has_person
friend_id INT(11)
person_id INT(11)
Indexes
google_map
id INT(11)
destination_name VARCHAR(255)
person_id INT(11)
person_id1 INT(11)
Indexes
friend
id INT(11)
name VARCHAR(255)
username VARCHAR(255)
person_id INT(11)
Indexes
messages
id BIGINT(20)
content LONGTEXT
seen BIT(1)
sent DATETIME
user_name VARCHAR(255)
person_id INT(11)
Indexes
travel_history
id INT(11)
destination_name VARCHAR(255)
person_id INT(11)
Indexes
description
id INT(11)
description VARCHAR(255)
person_id INT(11)
Indexes
search_history
id INT(11)
search VARCHAR(255)
person_id INT(11)
Indexes