Block Diagram for CyWalk

Group: 4_RASEL_6

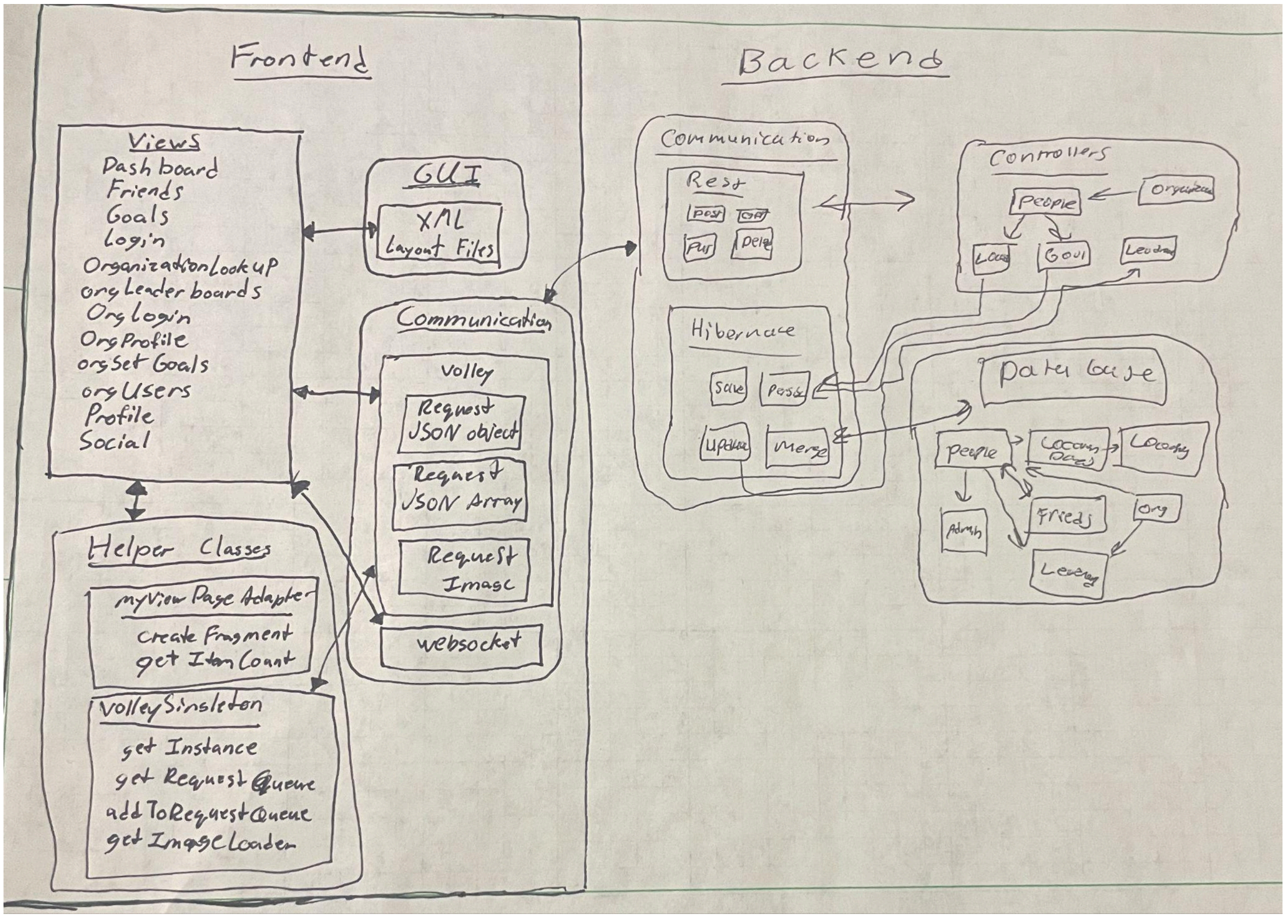
Caleb Kugel: 25%

Charlie Dolphin:25%

Ethan Ashihundu: 25%

Justin Stevens: 25%

API URL: http://coms-3090-072.class.las.iastate.edu:8080/swagger-ui/index.html#/


Frontend
Backend
Views
Dashboard
Friends
Goals
Login
OrganizationLookUP
orgLeaderboards
OrgLogin
OrgProfile
orgSetGoals
orgUsers
Profile
Social
GUI
XML Layout Files
Communication
Volley
Request JSON object
Request JSON Array
Request Image
websocket
Helper Classes
myViewPageAdapter
createFragment
getItemCount
VolleySingleton
getInstance
getRequestQueue
addToRequestQueue
getImageLoader
Communication
Rest
Controllers
People
Save
Update
Merge
People
Admin
Org

Actor:

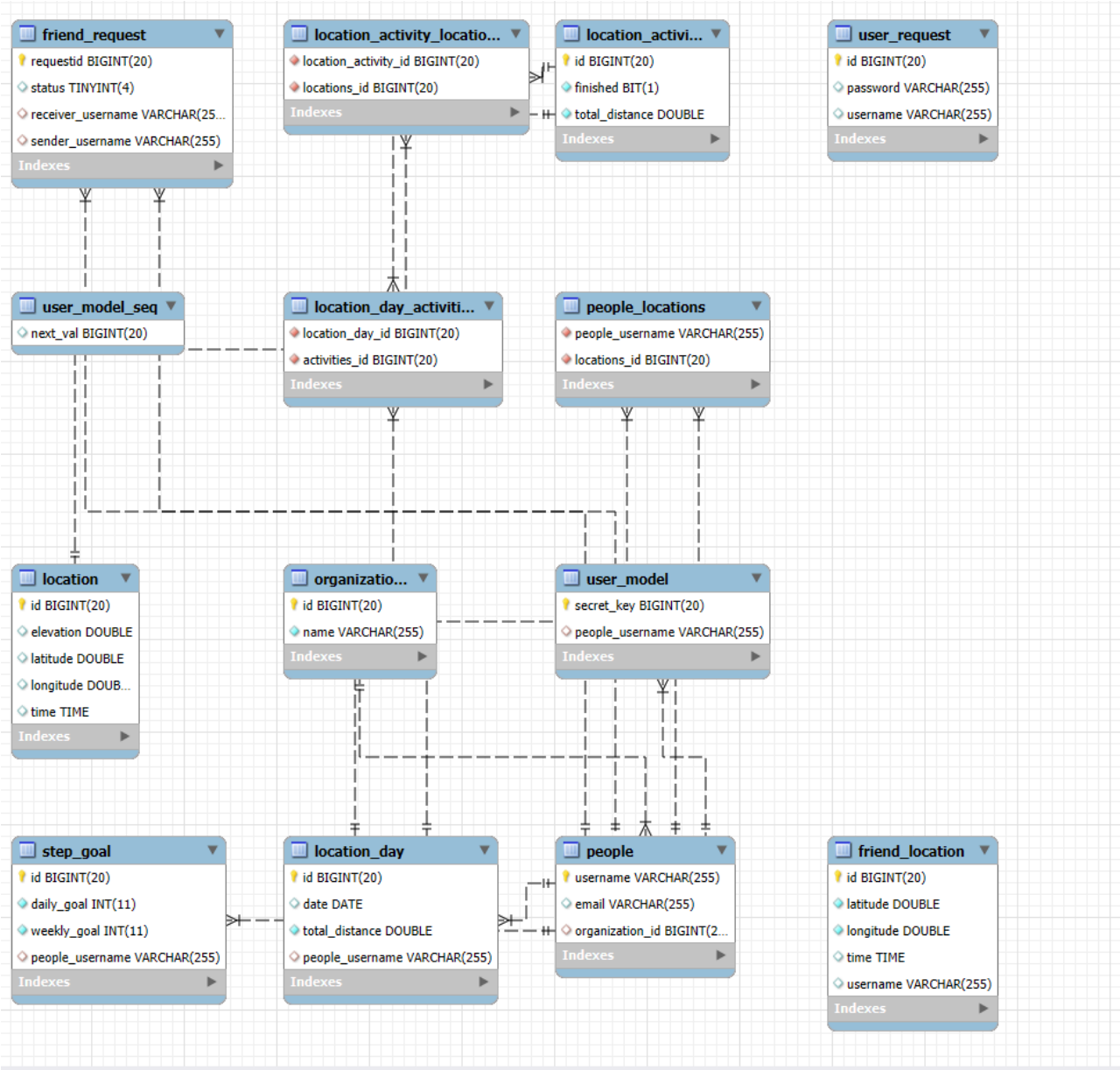
Interacts with the app through views, which contain UI elements such as buttons, input fields, and navigation. Login view gets username and password, and there is a signup button that creates a new user and sends them to dashboard view if one does not exist with the provided credentials. There is also a login button that sends the user to the dashboard view if a user exists with provided credentials. There is a Dashboard view which is basically the home page of the project, it has an integrated map view which shows the users location, and shows location of friends, etc. Locations are tracked with websockets, and user information is gathered with api calls to backend. There is also a Goals view where user can choose distance goals to try to achieve. Social view shows leaderboard comparing users, with filters for all global users, friends, and local. Profile view shows all user information, including profile picture, and options for editing profile info and log out. Organization Signup/login also exists within the login view, which is basically a group of users that can set organization-specific goals.

Server:

We are using Spring Boot as our backend for our project We use controllers to handle RESTful requests, handling POST, PUT, GET, and DELETE. Our controllers respond with response entities that give the frontend information as to what, if anything went wrong in the process of calling the request. Our repositories utilize hibernate to communicate with the database. We also listen on web sockets to provide extra functionality for the application,

Database:

Our database stores all of our users, their locations, friend relationships, and organizations they are a part of. We use MySQL as our database, which stores the relationships between our data and user information. We communicate with the database from the server


friend_request
requestid BIGINT(20)
status TINYINT(4)
receiver_username VARCHAR(25...
sender_username VARCHAR(255)
Indexes
location_activity_locatio...
location_activity_id BIGINT(20)
locations_id BIGINT(20)
Indexes
location_activi...
id BIGINT(20)
finished BIT(1)
total_distance DOUBLE
Indexes
user_request
id BIGINT(20)
password VARCHAR(255)
username VARCHAR(255)
Indexes
user_model_seq
next_val BIGINT(20)
location_day_activiti...
location_day_id BIGINT(20)
activities_id BIGINT(20)
Indexes
people_locations
people_username VARCHAR(255)
locations_id BIGINT(20)
Indexes
location
id BIGINT(20)
elevation DOUBLE
latitude DOUBLE
longitude DOUB...
time TIME
Indexes
organizatio...
id BIGINT(20)
name VARCHAR(255)
Indexes
user_model
secret_key BIGINT(20)
people_username VARCHAR(255)
Indexes
step_goal
id BIGINT(20)
daily_goal INT(11)
weekly_goal INT(11)
people_username VARCHAR(255)
Indexes
location_day
id BIGINT(20)
date DATE
total_distance DOUBLE
people_username VARCHAR(255)
Indexes
people
username VARCHAR(255)
email VARCHAR(255)
organization_id BIGINT(2...
Indexes
friend_location
id BIGINT(20)
latitude DOUBLE
longitude DOUBLE
time TIME
username VARCHAR(255)
Indexes