Design Document for Trail Traveler

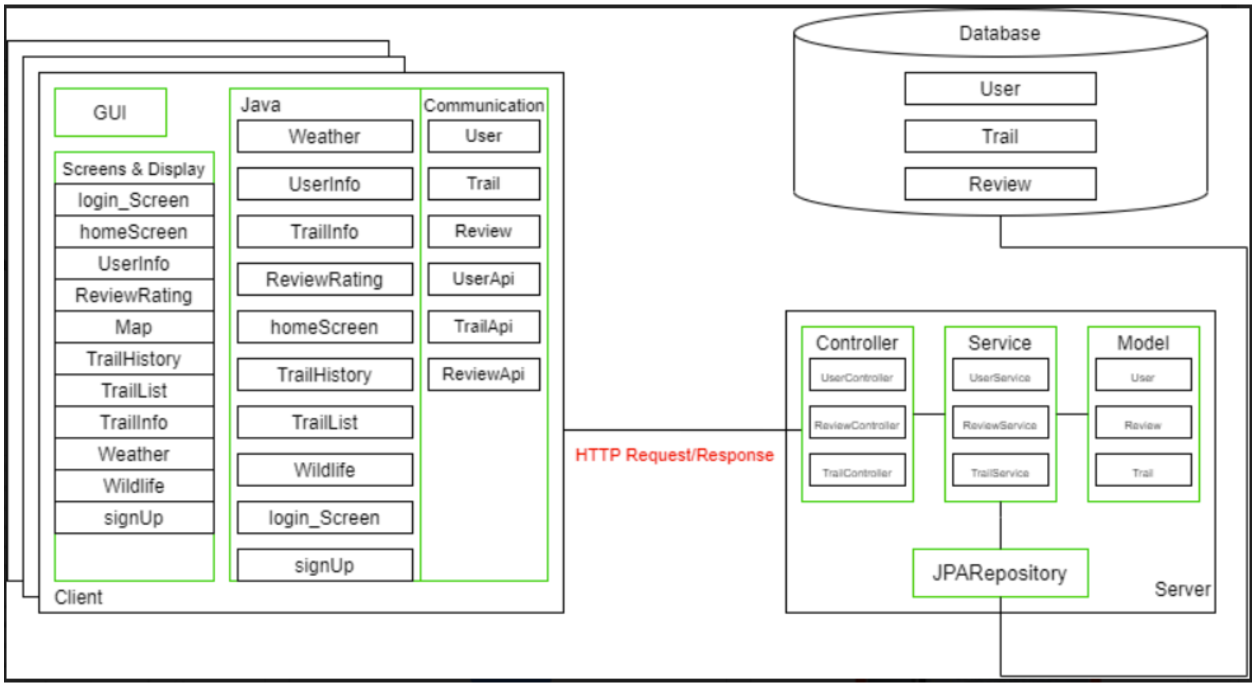
Group 2_DO_5

Daniel Chrisman: 25% contribution

Sullivan Jahnke: 25% contribution

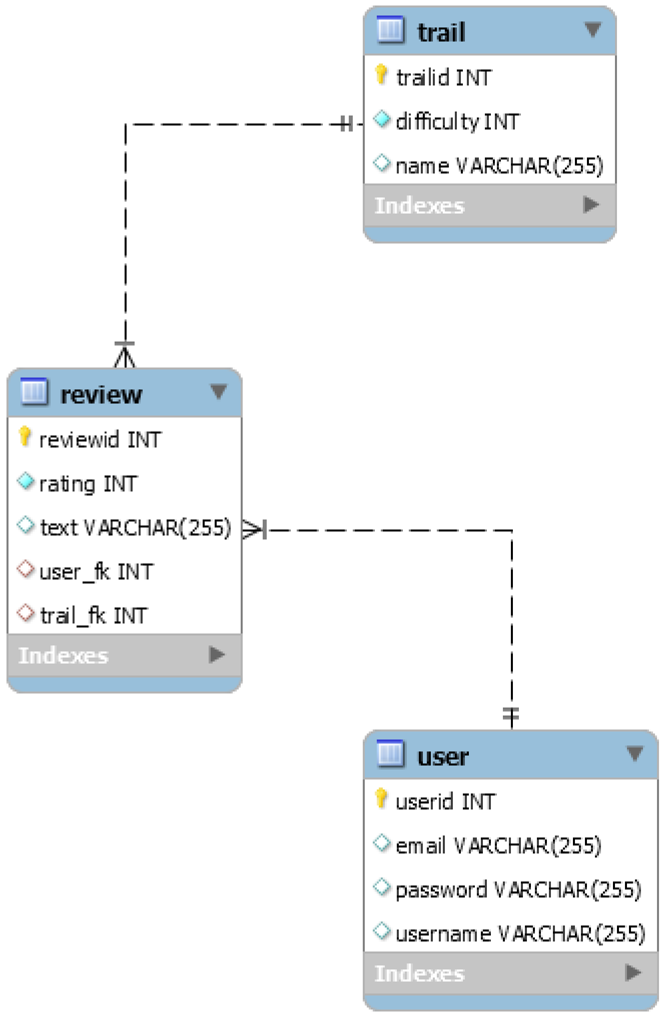
Elijah Hanson: 25% contribution

Erik Raman: 25% contribution


Database
User
Trail
Review
GUI
Screens & Display
login_Screen
homeScreen
UserInfo
ReviewRating
Map
TrailHistory
TrailList
TrailInfo
Weather
Wildlife
signUp
Java
Weather
UserInfo
TrailInfo
ReviewRating
homeScreen
TrailHistory
TrailList
Wildlife
login_Screen
signUp
Communication
User
Trail
Review
UserApi
TrailApi
ReviewApi
Client
HTTP Request/Response
Controller
UserController
ReviewController
TrailController
Service
UserService
ReviewService
TrailService
Model
User
Review
Trail
JPARepository
Server

On the client side, the user can navigate through the android application via different activities that have an individual xml file that controls what the user's screen looks like and what interactable objects will be present. There is also a java class for every activity that holds all of the logic behind the objects in the xml file like when a button is clicked it may have a listener that sends the user to another activity. There are also three java classes that are used to decipher the incoming JSON object from the tomcat server running on the backend, one for each of the tables in the database. Each of those has its own interface class that holds the methods for the Get and Post requests that the app makes in the background based on user actions like when a user tries to log into the app, a method gets the user JSON object based on the username or email that they entered. Then in the login_Screen java class it compares if the password they entered matches the one held in the database. In order to access the database the client must run the backend application to start the Tomcat server that handles communication between the database and android app. Our design allows for multiple users to be using the app at the same time, two different users can make requests to the database and they will receive the information that they requested.

On the server side, an http request is received through the controller. Based on the path that was given in the http request, the class that will handle it is determined (UserController, ReviewController, TrailController). For example, "/review/post/…" will be handled by the ReviewController because the client side specified that it wants to post a review through the path. Once the controller receives the request, it calls methods in its corresponding service class (UserController -> UserService, ReviewController -> ReviewService, TrailController -> TrailService) that is an object in the controller. The getter and setter methods for each class edit, retrieve, or retrieve and edit data from the database tables. When the controller receives the post request, the called methods in ReviewService create a new entity in the Review table with the information given by the frontend. Then, when the frontend sends an http request to view the data (to display in a list or screen for example), the backend uses the getter methods in ReviewService to pull the data from the SQL server and pass it back to the client side.


trail
trailid INT
difficulty INT
name VARCHAR(255)
Indexes
review
reviewid INT
rating INT
text VARCHAR(255)
user_fk INT
trail_fk INT
Indexes
user
userid INT
email VARCHAR(255)
password VARCHAR(255)
username VARCHAR(255)
Indexes