Project 02: Ghetto Yelp
CST 338 – 40 Fall 2024

Ghetto Yelp

Ghetto Yelp is an application that allows users to view and create reviews and ratings for restaurants. Users will be able to add and view reviews to preexisting restaurants, while admins will be able to add restaurants to the pool.

GitHub: https://github.com/JSmevog1/GhettoYelp

Table of contents

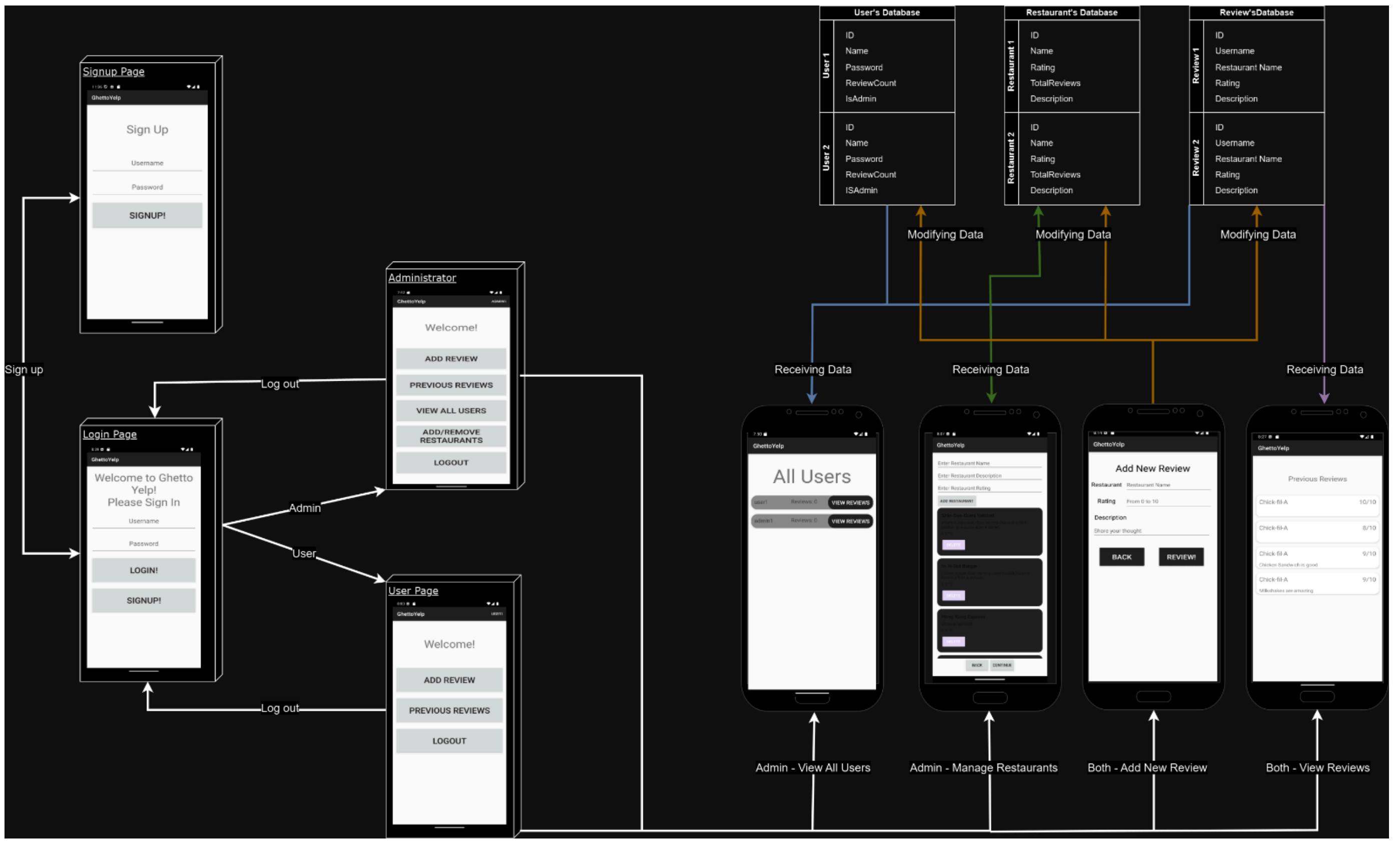
Completed Rubric Items

Please Highlight the items you have completed

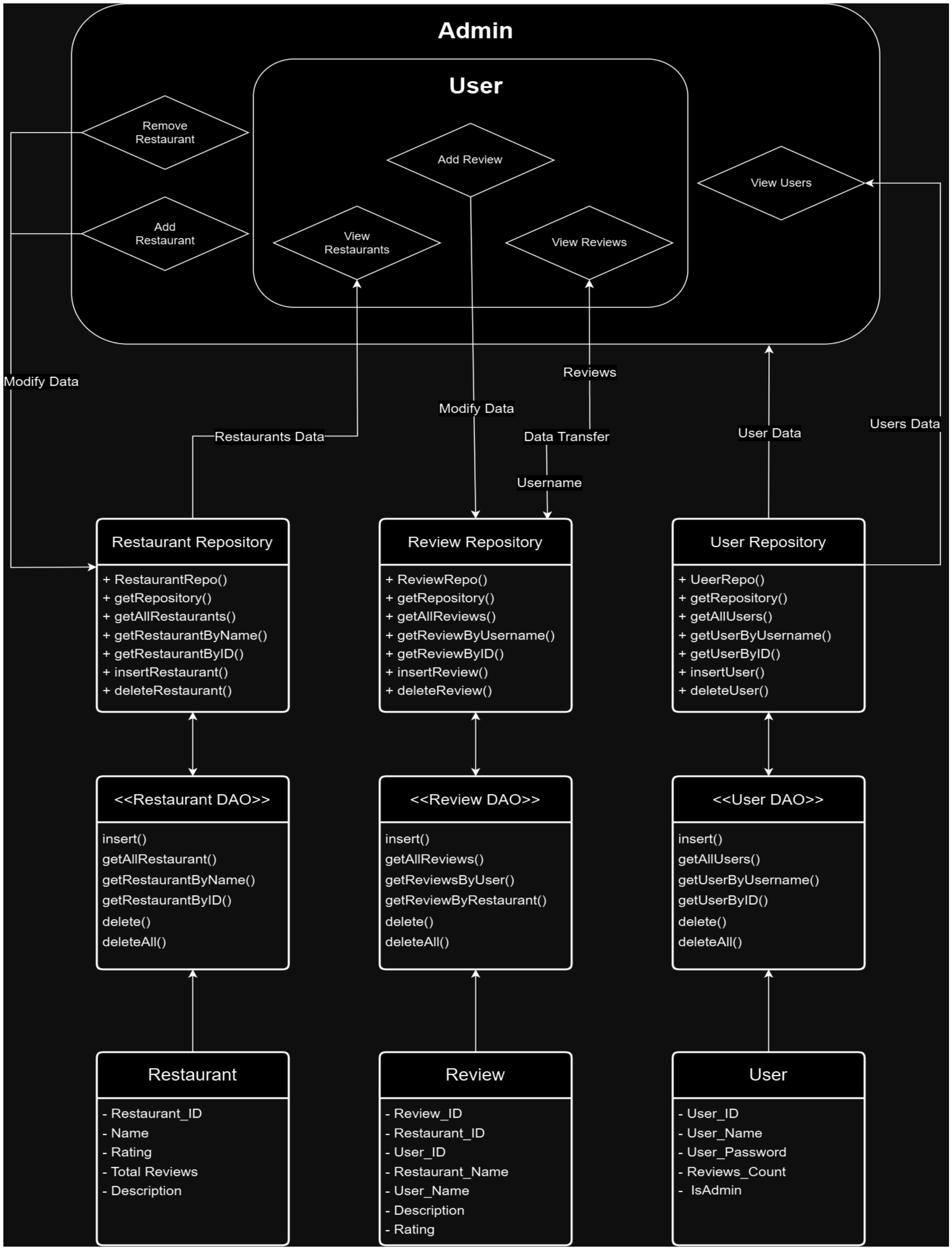
Item	The following are Required	Score
Persistence	Minimum 3 tables implemented using the Room database wrapper If you do not use the Room Database wrapper the project will be marked as 0.	30
Github issues	There must be at least 5 GitHub issues PER TEAMMATE	10
Video	Each application must have a video demonstrating its functionality. The video must be between 6 -10 minutes long. Each person must present their portion for 2-4 minutes	35
Highlighted Rubric	This rubric must be included in the video AND the submission. All completed tasks must be highlighted	5
Activities	Each team member must create 2-3 activities, one of which must interact with the database.	12
Activity: Login Page	An activity where a username and password are entered. The username and password must be stored in the local room database.	4
Activity: Landing Page	After successful login an activity should start that displays the username and some other information retrieved from the local database.	4
Admin landing page	If the user is an administrator **the landing page must indicate this** with additional options. **The same Activity must be used for both admin and non-admin users.**	4
Github	Code must be pushed to Github in a **public** repository. Each teammate must have at least 3 branches, with 5 commits EACH; each branch must be merged into main to count.	10
Unit tests	Each teammate must contribute at LEAST 2 unit tests. The tests must pass.	20

Optional Item	The following are **Optional**	Score
Intent Factories	All Intents to start an activity must use the factory pattern Either each activity must have an Intent factory method OR use a Factory class.	10
Intent Tests	At least 1 unit test per activity transition. That is each intent must have a unit test.	20
Database Tests	Each database table must have at least three tests (insert, update, delete)	20
Recycler View	At least 1 activity has a recycler view for displaying information	10
Livedata	Incorporate liveData in your project. This dovetails nicely with adding a recycler view	10
API Integration	Use Retrofit to consume an external API	20
External Library	Use an external Library. Here is a list of examples. **Retrofit** is included and would count for this.	16
Fragments	Incorporate fragments into your application	8
Pull requests	Use Github pull requests when merging a branch into main.	6
OAuth	Implement login with OAuth 2. This can replace the login procedure described above	16

Design Layout



Entity Relationship Diagram (ERD)



Team members

Tariq Kakar

- Issue: Login Page
- Issue: Login Preference and Logout
- Issue: Main Activity (UI and Back-end)
- Issue: User Landing Page (UI and Back-end)
- Issue: User Entity
- Issue: Signup Page
- Fix and merge corresponding branches into the main branch.
- Assign and update issues on GitHub
- Video: Sign Up Page, Login Page, User Page

Jason Smevog

- Issue: Main Activity (UI and Back-end)
- Issue: Admin Landing Page (UI and Back-end)
- Issue: View All Users Activity – Admin (UI and Back-end)
- Issue: Review Entity
- Issue: Manage Reviews
- Fix and merge corresponding branches into the main branch.
- Assign and update issues on GitHub
- Video: Introduction, Admin Page, View All Users

Yusra Ashar

- Issue: View All Restaurant Activity (UI and Back-end)
- Issue: Add/Remove Restaurant Activity (UI and Back-end)
- Issue: Restaurant Entity
- Issue: View All Restaurants – User (UI and Back-end)
- Fix and merge corresponding branches into the main branch.
- Setup deadlines for the project
- Assign and update issues on GitHub
- Video: Add Restaurant, Remove Restaurant

Yui Nguyen

- Issue: DAOs for Restaurant, User, Review
- Issue: Repository for Restaurant, User, Review
- Issue: Previous Reviews Activity (UI and Back-end)
- Issue: Add Review (UI and Back-end)
- Fix and merge corresponding branches into the main branch.
- Documenting the project, creating and updating diagrams
- Assign and update issues on GitHub
- Video: Add Review Page, View Reviews Page

User Stories

1. As a Ghetto Yelp user, I want to be able to login and search for restaurants in the database.
2. As a Ghetto Yelp user, I want to be able to review the restaurants in the database.
3. As an admin of Ghetto Yelp, I want to be able to add restaurants to the database.
4. As an admin of Ghetto Yelp, I want to be able to delete inappropriate reviews.

Issues Generated from Use Cases

1. Implement Databases
    a. Add the Gradle requirement.
    b. Create Repository
    c. Create Entity Objects
        i. User: User_ID, User_Name, User_Password, Reviews_Count
        ii. Restaurant: Restaurant_ID, Name, Rating, Total_Reviews, Description
        iii. Review: Review_ID, Restaurant_ID, User_ID, Restaurant_Name, User_Name, Description, Rating
    d. Create DAOs for each entity:
        i. User DAO: Verify login
        ii. Restaurant DAO: Add, search and fetch restaurants
        iii. Review DAO: Add, fetch and delete reviews
2. Core Features
    a. Login:
        i. Implement basic login page prompting for username and password
        ii. Ability to check if user is an admin
    b. Search Restaurants
        i. Implement a search page with search bar
        ii. Display results
    c. Submit Reviews
        i. Implement a review page for users to submit reviews with a rating and comment
    d. Admin Function
        i. Implement an admin page with options to add new restaurants and delete reviews

Youtube Video Link:

https://youtu.be/DN_VpEYgits