Design Document for Campus Critics

Group 1_HB_7

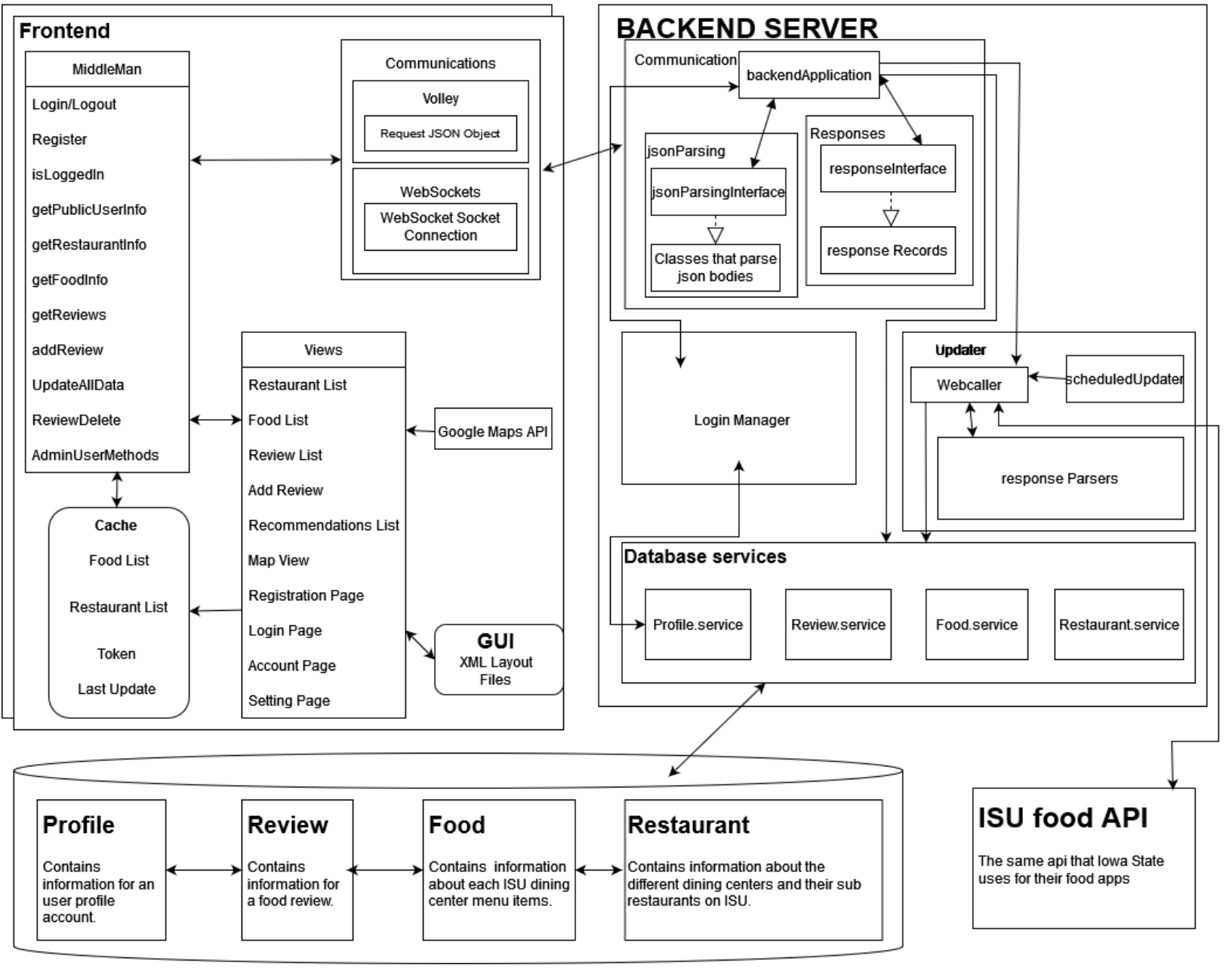
Member1 Ryan Ledbetter: 25%

Member2 Ryan Leska: 25%

Member3 Parker Pederson: 25%

Member4 Nick Thoms: 25%

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



Frontend:

Cache: The cache is where we store the food list, restaurant list, user token, and a variable tracking when these were all last updated. These are stored locally to reduce bandwidth on the server and make loading far faster in the app. There is an option in the settings to clear this.

Communications: The two forms of communication we will have with the backend server are through volley and using a websocket. With Volley, we send JSON objects of the data that is being cached on the first startup of the app, or if the data has been cleared. We also send reviews using JSON objects. We will be retrieving reviews through websockets as we believe this will allow for instantaneous refreshing of reviews as opposed to using a JSON call to pull the data as it is being refreshed.

Views: The way our app is segmented is into three main portions being the main tab which houses restaurants with links to the associated foods and associated reviews as well as a view to add reviews. We did this in such a way to make sure that the food lists per restaurant are still accessible from the map view and recommendations in the center tab. The final tab is the profile section that houses the registration/login page which transforms into the account page when a user is signed in. This is then linked to a settings page where the user can change settings such as enabling dark mode or clearing the app's cache.

GUI: The GUI of our app is entirely built with XML from within Android Studio. The ones not accessible to users are designed in such a way that they are encompassed by a larger view such as with the cards created for restaurants, food, and reviews. These are each encompassed into a larger view made up of multiple of the aforementioned card views.

Google Maps API: The Google Maps API is used to host the map in the center fragment of our app. We are using it as a way to view a map of campus and each location is represented with a marker that acts as a clickable object that leads to the restaurant's page.

Middle Man: The Middle Man class manages all of the inbetweens from the views/gui to the cache and communications. The middle man simplifies how all server requests are implemented throughout the whole frontend.

Backend:

Communication: The communication package contains all of the endpoints. Also contains all of the classes that the incoming json bodies will be mapped onto and the response records. The basic flow of the package is. 1) The request comes in. 2) The body gets parsed into data.3) The processing gets passed to an external package. 4) The response gets formatted into the record.

Login Manager: Is a single standalone class that manages the list of currently logged in users. Encapsulates all services involved in user authentication including login, registration, token checking, and getting user information.

Updater: Contains two main parts, the scheduler and the webCaller. The scheduler just periodically calls the webCaller. The webCaller sends requests to the ISU Food API and parses it into usable data for the database to use. The webCaller can be called by either the scheduler or the backend Application. All of the time stamping happens inside of the webCaller which notes the time that each entity gets updated.

Database Services: The database services are a group of subpackages each consisting of a service, a repository, and an entity. The services handle the bulk of the logic for the entire backend. Generally when an endpoint is called the appropriate services will be called and then the correct data will be returned. The design is generally layered with the services only accessing the repositories and entities.

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