SuiteMate

No More Gloomy Roomie :)

Project Track 1 : Project Reflection Report

Team034 - TeamNescafe

Mehul Oswal - mehuljo2
Daksh Gandhi - dakshg3
Daksh Pokar - dakshp2
Dhruv Kathpalia - dhruvk5

1. **Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).**

We have covered everything that was proposed in stage 1. Rather, we have added new features such as ‘analytics’ as we thought that it might be beneficial for both the ‘Customers’ and ‘Agents’. For Customers, we have added interactive graphs for average area-to-rent ratio for each pincode, which enables the user to find the best deal. For Agents, we have added a way to compare different properties based on their popularity ratio (no. of applications/ no. of units) for the property. This enables the agents to compare the property’s performance against their competitors.

2. **Discuss what you think your application achieved or failed to achieve regarding its usefulness.**

Our application is designed to help customers find not only their ideal property but also we provide them a way to match with other potential roommates registered on our platform. Our advanced SQL query determines the potential roommate by finding a similarity ratio (matched preferences / total no. of my preferences). We provide a wide range of filters along with some advanced filters for users to find their desired properties. As stated in the proposal, our agents can accept/reject the application for a particular unit in a very robust manner. We could have added another feature such as ‘push notification’ which allows the user to get updates about the status of their application. However, one aspect we needed to address was the responsiveness of our web application on mobile devices. Currently, our web app is not optimized for mobile, which may hinder the user experience for those accessing our platform on smartphones or mini-tablets.

3. **Discuss if you changed the schema or source of the data for your application**

We made the following changes to the schema with respect to the initial proposed schema:

- We added a table for storing tokens for user authentication.
- We slightly modified the schema of Unit and Property Photos by removing the photo_id column so that each photo could be uniquely identified by unit_id/property_id and photo_url.

We slightly modified the data from our initial proposal according to the schema of our application. We also generated data for several other tables.

4. **Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?**

As mentioned above, we slightly adjusted the UML diagram as we removed photo_id from both tables Unit Photos and Property Photos. Unit_Photos and Property_Photos were designed to be weak entities and it made sense to remove the photo_id column.

5. **Discuss what functionalities you added or removed. Why?**

Added:

- We added an analytics feature for both customers and agents.
- We implemented a feature to ‘review’ and ‘add rating’ to a property that was not a part of stage - 1 proposal. Users could also delete their review for a particular property.

Removed:

- We removed the feature to add/delete property for an agent as it did not fit well into real-world hierarchical roles.

6. **Explain how you think your advanced database programs complement your application.**

We successfully implemented:

- Trigger: If an application is accepted, all other applications are rejected. This enabled the user to map the unit with high correctness.
- Stored procedure: To determine which complex advanced filter to run based on a ‘flag’ parameter). This helped us filter properties based on these advanced filters to fetch the properties by multiple parameters.
- Transaction: We implemented serializable transactions for actions such as submitting an application for the unit as well as reading the application per user. This was helpful in eliminating the possibility of arising conflicts in the applications and availability of units.
- Constraint: We implemented foreign key constraints on “Application” and “Reviews” tables which enabled us to prevent multiple users having multiple applications for the same Unit. Similarly, users were not allowed to post multiple reviews for the same property.

7. **Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.**
    - DB Connection Pooling
        - Initially while we were working on the project simultaneously we noticed a particular issue that the connection randomly dropped or took a long time for a query to run. To debug this issue we created a mini-script that fired multiple parallel CURLs to keep an eye on the database connections and mimic user requests. We identified the primary cause—the application's ineffective MySQL connection management—by careful observation. Every time a CURL request was made, a new database connection was established, overwhelming the MySQL server and causing intermittent connection dropouts and slow query execution. We finally fixed it by setting pool_reset_session to True and picking a connection from the pool and returning it back for every API call. - Daksh Pokar
    - CORS
        - We faced Cross-Origin Resource Sharing (CORS) issue while calling our flask APIs from the NextJS frontend. On debugging further we realized that this was due to security reasons and we fixed it by using flask-cors middleware in our backend file. - Daksh Gandhi
    - Organizing API Structure
        - When working together on building an application it becomes a paramount challenge of concurrently doing the development on a single file. Same thing happened with this project where we had just one main.py file having all the APIs. To mitigate this, we split our project into 4 modules - customer, analytics, agent and auth. Doing so we could easily solve the challenge of concurrent development. It made it very easy to handle merge conflicts as well. This modular approach facilitated clearer code organization, improved code maintainability, and enhanced team collaboration - Mehul Oswal
    - Trigger
        - When we attempted to modify the same table on which the trigger action was defined, we were getting a locking error as the trigger was updating the same table on which the event was placed. So we had to modify the trigger such that the trigger executes when the action is performed on another table. This enabled us to use the trigger for our use-case. We were able to modify the “application” *status* based on the *availability* of the “property”. - Dhruv Kathpalia

8. **Are there other things that changed comparing the final application with the original proposal?**

   No.

9. **Describe future work that you think, other than the interface, that the application can improve on**
   - Since we are not giving access to the agents to directly remove the properties, we think that introducing hierarchy makes it easier to manage. A company supervisor role can be introduced who can manage the employment of agents as well as addition or deleting of properties.
   - We think that a push notification service can be enabled for a better understanding of the status of the application.
   - For security purposes, we can enable OTP based authorization when registering on the platform.
   - We can add spam filtering to reviews.

10. **Describe the final division of labor and how well you managed teamwork.**
    a. **Daksh Gandhi -** Analytics APIs and Frontend including interactive visualization, along with other backend APIs and util functions. UnitDetails and Application Submit logic.
    b. **Daksh Pokar -** Roommates List & Search APIs, Similarity Score Calculation Query and API, Property Filters API, Hosting Backend on GCP, and Frontend on Netlify.
    c. **Dhruv Kathpalia -** Added the Trigger and Constraints. Worked on the Backend APIs for Reviews and Preferences. Created the UI for the ‘Property Details’ page, ‘Reviews’ and ‘Add User Details’ page.
    d. **Mehul Oswal -** Worked on listing of Properties and Pagination. Implemented keyword Search on properties and advanced filtering on property using stored procedures. Set up Frontend and Middleware framework for handling the backend requests.