Project Report
Team 066 ---- Ease Lease

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).

Our project remained largely consistent with the initial proposal, with minor adjustments only to the UI interface. Additionally, we changed the bidding feature to an application process, enhancing the practicality of the app.

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

Minor UI Adjustments: These adjustments likely involved improving user interaction design and optimizing the visual layout to enhance usability and overall user experience.
Feature Modification: Replacing the bidding system with an application process simplified user operations and better met the real-world demands of the rental market, increasing the app's usability and applicability.

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

Source of data: We do not change the source of data and our data source is still from Airbnb (http://insideairbnb.com/get-the-data/).
Changes of Schema: We added a new attribute 'image_url' to the listings table, and included 'user_id' in both the review and rating 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?

The changes about Listings Schema:
Previously, our listings table is like:
listing(listing_id, room_type, description, price, from_date, to_date, lanlord_id, longitude, latitude)
Our creative components include listing images; however, our current listing dataset lacks image information. To address this, we have introduced a new column of attributes for listing image URLs in the listing table, allowing us to store corresponding image information for each listing.

The changed listing table is like:
listing(listing_id, room_type, description, price, from_date, to_date, lanlord_id, longitude, latitude, **image_url**)

The changes about review Schema and Rating Schema

At first, we designed review table and rating table were like:

review(review_id, reviewer_name, content, listing_id)

rating(rating_id, scores_rating, scores_accuracy, score_cleanliness, score__checkin, scores_communication, scores_location, scores_value, listing_id)

However, during implementation, we realized that to limit each user to only one review and rating per listing, we should use 'user_id' as a foreign key constraint.

The following is implementation details:

1) Inserting the information of anonymous users into the User table.
   We have created an entry in the User table with a user_id = 1 to represent anonymous users. This implementation addresses the issue of the review table and rating table that currently lack corresponding user_id data.

```
INSERT INTO User (user_id, user_name, password, phone_number, first_name, last_name)
VALUES (1, 'anonymous', 'N/A', 0, 'Anonymous', 'User');
```

2) Alter the review table and rating table
   Add a user_id Column to Rating and Review Tables

```
ALTER TABLE Rating ADD user_id INT;
ALTER TABLE Review ADD user_id INT;
```

   Set Default Values for Existing Rows

```
UPDATE Rating SET user_id = 1 WHERE user_id IS NULL;
UPDATE Review SET user_id = 1 WHERE user_id IS NULL;
```

   Add Foreign Key Constraints

```
ALTER TABLE Rating ADD CONSTRAINT fk_rating_user_id FOREIGN KEY (user_id)
REFERENCES User(user_id) ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE Review ADD CONSTRAINT fk_review_user_id FOREIGN KEY (user_id)
REFERENCES User(user_id) ON DELETE CASCADE ON UPDATE CASCADE;
```

After changing review & rating table structures, the current tables are like:

review(review_id, reviewer_name, content, listing_id, **user_id**)

rating(rating_id, scores_rating, scores_accuracy, score_cleanliness, score__checkin, scores_communication, scores_location, scores_value, listing_id, **user_id**)

The following UML diagram has been revised to reflect changes made to the listing, review, and rating tables.

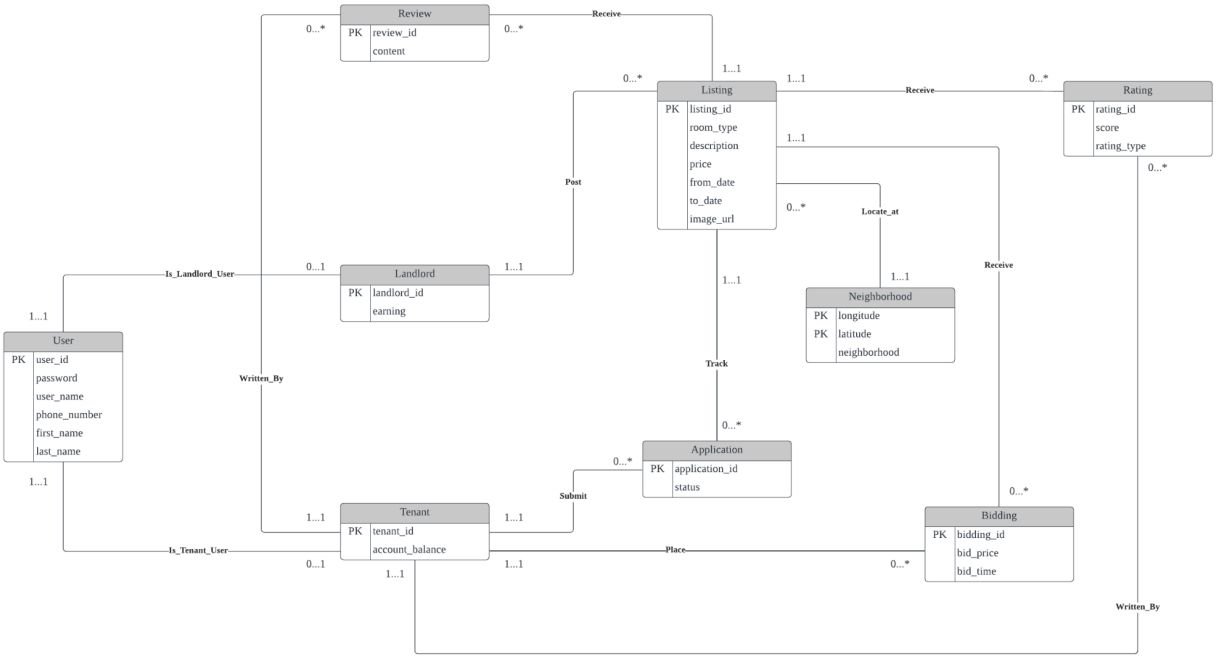

Figure 1 The Revised UML

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

An image enhancing function was added for the landlords that allows them to upload the picture of their properties. We believe this function is practical in house-leasing apps, and it is helpful for landlords to give better descriptions of their properties.
We removed the financial reporting because there weren't enough application records for us to create a financial report for the landlord. Similarly, there weren't enough records to support the rewards mechanism functions.

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

Our application, Ease Lease, successfully provides the fundamental functions and several enhanced functions with a smooth interactive interface. We've equipped our application with a highly scalable and flexible database architecture. This ensures that as our user base grows, our application can handle larger data sets efficiently without performance hits. On our user-friendly house-subleasing platform, landlords are able to modify their self-descriptions, upload pictures of their house, and check the applications on their property. While tenants can easily submit or withdraw applications on any houses they are interested in. And our system allows them to post ratings and reviews on the property. Additionally, we included a bidding function on our app, which not only improves user engagement but also adds significant value to our application, setting it apart in a competitive market.

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.

Sizheng Zhang: While implementing the creative component, we encountered the issue of Google Cloud Storage's (GCS) access authorization. Since we used a GCP VM instance to host our application, so by default, the service account has both read and write access to our cloud storage bucket. However, the VM itself does not have the write authorization by default, so at the beginning of the implementation, although we could read data from our GCS bucket, and the authorization API shows that we have write access to the bucket, we could not perform any writes. We finally resolved this issue by manually granting our VM all the accesses to our GCS bucket.

Chenzhao Wang: During our project, we encountered significant technical challenges while updating our database schema to add new features. Initially, our listings table did not include image information, which was crucial for our application's creative components. To resolve this, we introduced a new column for image URLs in the listings table, enabling us to store corresponding image data efficiently for each listing (details are in previous parts).
Additionally, we realized the need to restrict users to a single review and rating per listing to maintain data integrity. To enforce this, we added a 'user_id' column in both the review and rating tables. For handling entries by anonymous users, we created a default user ID in the User table and updated existing reviews and ratings to reflect this. These modifications required careful implementation to ensure consistency and functionality in our application, highlighting the importance of early consideration of user constraints and feature additions in future projects.

Chenyu Liao: My challenge was to optimize a SQL query for a property listing page that I was developing, which needed to display the top 10 property listings. The initial query merged several tables but failed to include essential landlord information, which was crucial for my teammates working on the property details. I needed to revise the query to efficiently incorporate these details while ensuring the data remained accurate and the system performance was not compromised. This adjustment was key to supporting seamless collaboration and functionality across different parts of our web application.

Chloe Cai: When enabling the withdrawal of an application on the tenant profile, the initial query only deletes the application in the table. However, we realized that, since the bidding and application are submitted together, deleting the application also requires the simultaneous deletion of the corresponding bidding record. To handle this, I inserted a trigger query into our database. This trigger ensures that after each deletion from the application table, the database system checks if this tenant had a bid on that property. If a bidding record exists, it is deleted. If no corresponding bidding record can be found, the trigger simply passes.

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

Nothing. We have mentioned all our changes in prior sections of this report.

9. Describe future work that you think, other than the interface, that the application can improve on

One possible future direction is to combine our backend with generative AIs in text and images. We can have a chatbot built with GPT that can input users' requests in the form of natural language and generate suitable filters to look for the properties that best fit their requirements. Also, we can improve our creative component by adding diffusion models to create better and more creative images.

10. Describe the final division of labor and how well you managed teamwork.

Sizheng Zhang: Designed the general architecture of the application and finished the implementation of the account module, the landlord profile module, and the creative component module.
The overall teamwork was managed very well. Each member was responsible for at least one module and advanced functionality, so we learned a lot about web application development and database operations. We also had weekly scrum meetings and project tracker files to keep track of all individual progress so that we could be aware of any potential delays and difficulties in coding and were able to solve all problems in time. To sum up, it was a great collaboration experience.

Chenzhao Wang: Responsible for implementing functionalities such as keyword search, application submission, bidding, and review/rating submission, along with the corresponding stored procedures.
This collaboration was very successful. Team members chose suitable roles based on their expertise. In technical matters, we communicated actively and brainstormed solutions together; in non-technical aspects, we collectively conceived creative and practical features. Everyone actively participated in the project and learned from each other. Additionally, we utilized Google Drive and Zoom meetings to manage the entire project, ensuring smooth progress.

Chenyu Liao: I was primarily responsible for the listing page and tenant profile page. My work involved functionality to display properties once a tenant logs in, and a feature that allows tenants to access their profile where they can view their personal information and real-time avatar.
The team collaboration was excellent. Each member took charge of a specific module and its advanced features, which deepened our understanding of web application development and database management. We held weekly scrum meetings and used project tracking tools to

monitor progress, enabling us to identify and resolve coding challenges promptly. Overall, it was an exemplary team effort.

Chloe Cai: My responsibility was to design the listing details page, which included displaying the basic information of a specific property, recent ratings, and reviews. I also developed two functions on the tenant profile page: one function allowed tracking the applications submitted by the tenant, and another enabled the withdrawal of an application with a trigger query to recall the corresponding bidding record.

Our teamwork was managed very well throughout the entire project. We held weekly meetings to communicate, brainstorm, and ensure everyone was on the same page regarding our progress. Tasks were allocated reasonably, based on each team member's expertise, and task trackers were used to support the well-organized process of our project. There was a positive learning and collaborating environment within our small group. In all, it was a great teamwork experience. And I really appreciate the work done by my teammates and the help from them.