Project Presentation

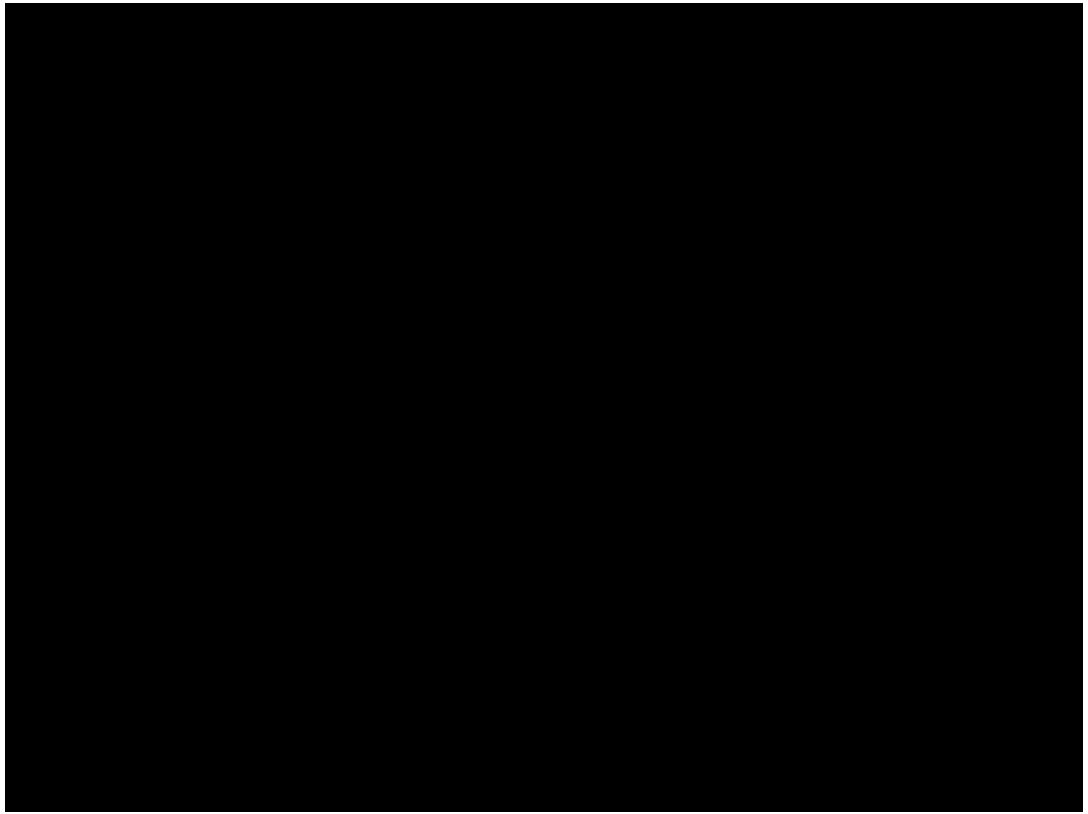
Real Estate Manager

Anudev, Rana, Eren, Felix, Andrew, Rohit

Specification

- Create a program that simulates a real-estate platform with buyers and sellers
- Sellers will be able to list properties by providing address, price, number of bedrooms/bathrooms, number of stories, etc.
- Buyers will have search parameters they can use to find out what properties interest them
    - Listing properties by: price, location, and the above features
- Sellers can remove properties from their property list
- Buyers can place offers (by sending messages to sellers) and sellers can see their offers when they login

Demonstration of Functionality



Design Decision to follow Clean Architecture

- In order to follow the Dependency Inversion rule of clean architecture, we changed our use case classes so that they weren't creating CSV readers/writers
- Created gateway classes with read and write methods that implemented interface that was contained in the use case package
- Modified the way the CSV files were being read so that the gateway classes returned the contents of the CSV file rather than attempting to store the data in containers
- Instead, populating the containers was done by the use cases
- A StoreData() class was used to call all these methods

Design Decision to uphold SOLID

- In order to uphold the Single Responsibility Principle, we created a separate class to generate unique IDs for data in the containers
- While we were originally planning on have the use case classes generate these ID's, this would give each use case class associated with a container multiple responsibilities
- Instead, ID generation happened inside GenerateUniqueID.java and each use case creates an instance of this class and uses it when storing data in containers

Design Pattern

- Used the Dependency Injection design pattern to avoid circular dependencies in the gateway classes
- Originally, the gateway classes created new instances of the use case classes and called some of their methods to populate the containers
- However, this meant the gateway classes depended on the use case classes and the constructor for the use case classes depended on the corresponding gateway class
- Using the design pattern, an instance of the gateway class was passed in the constructor to the use case, and this removed the dependencies
    - In order to support this, the implementation of the read() method had to be changed, as stated before

Something We're Proud Of

- We're proud of how coordinated we were throughout all the phases of the project
- We split up the work for the project clearly and each group member made sure to coordinate with the group if there were any changes to the expected plan
- As a result of our good coordination and communication amongst the group, we were able to smoothly arrive at a large-scale finished product
- Furthermore, there was very little confusion over who was completing which task and this allowed us to have very few merge conflicts over the course of the project

Thank You for Listening