SE-2019-F1 : Project Report

Team Members:

Rohan Mohapatra	01FB16ECS307
Saahitya E	01FB16ECS322
Shailesh Sridhar	01FB16ECS349
Sailesh Gaddalay	01FB16ECS331
S Swathi Reddy	01FB16ECS320
S Jayasurya	01FB16ECS318
Saahil Jain	01FB16ECS321
Sharath N	01FB16ECS352
Shailendra Hegde	01FB16ECS348
Sharmishta Singh	01FB16ECS354
Shadan Alam Kaifee	01FB16ECS346

Summary

Harmony is a web based project that automates and provides functionality of a typical Real Estate that ensures all the customers are satisfied by mutually arriving at an agreement that favours them, with additional Recommendation Software, Filters and Chat feature to enhance the search for the user when compared to existing Real Estate application providers. Harmony provides platform for Owner to *post their properties to be rented out on the website*, Seller to *post their properties to be sold on the website*, Buyer to *buy a property of their choice*, and Renter to *look for house on rent based on their preferences*. With the suitable architecture and design, our website thrives hard to boost the user experience by using the mentioned software technologies and appropriate software engineering skills. Harmony also provides a secure login, property authentication, user authentication to

make sure the website is threat-free. Harmony's Recommendation Toolkit along with Filters recommends property and allows the customer to choose the right home and the right environment by tweaking the property list based on price, location, air quality, traffic density. Further, if Owner-Renter or Buyer-Seller are mutually interested, a chat feature enables them to communicate closely and gives a clear picture related to the property. Finally, the expected working of the application was ensured using Grey Box testing tool.
Real-time property viewing/updation and migration to a MicroService architecture can be considered as a future enhancement.

Front End

The front end acts as the integration between the various components given below. The front end was built using react and it gives a usable functionality for the users. THe front end was broken into layouts and views, seller, rentee and buyer get 2 screen to login and signup to the system. The seller/rentee has views to add a property, view the list of properties and as well chat with the list of buyers.

The buyer can view either rent/buy properties, if he/she has signed-in to the system, they can even add to the wishlist. They are also presented with a list of recommended properties that can be used to to look at properties that other user seem interested.

Properties Module

The properties module/component of the system was built as an app in a Django application. The properties modules consisted of the following files:

- models.py
- serializers.py
- views.py

Models.py: Here the models are the schemas for the properties table in the database(sqlite) is defined. Some important attributes in the models

specified are propertyName, propertyAddress, price, bhk, societyName, city, airQuality, traffic, propId.

Serializers.py: Here the way to convert the model objects to primitive python data structures like dictionary and list is specified. This is useful to convert to data in HTTP Response objects which can be in JSON or XML format.

Views.py: Here the high level control mechanisms(class based or function based) that receive specific HTTP requests and produce HTTP Responses are defined. Here the authentication controls are also specified where certain functions need authentication of the requester before an HTTP Response is sent. The high level business logic of filtering, picking attributes and serializing into HTTP response is done here.The filtering is done with a class based view.

Chat Module

The Chat Component is built using node js as Back-End and react-chat-widget.
Its built using Rest-Api available from cometchat.com this website stores the messages in its own database along with the details of the users who take part in the chat activity.
It provides real-time messaging between the buyer and seller enabling them to satisfy the queries about the properties and can help them schedule meeting about the same. Using Request module the Rest-Api are handled.The responses of the rest api is processed and the messages are displayed using functions of react-chat-widget

Login Module

The login system of harmony was made using django. The first step is the registration of a user to the website. The user has to fill 5 fields which include their username, first name, last name, email id and password. There is a serializer which parses the data sent from the frontend and converts them to be stored in the database. A unique token is created for each user. This marks the end of the registration process. To login the users then enter their username and password. This is then passed through a serializer checked if the credentials are valid and if it is, a json web token (jwt) is sent to the frontend and this marks the session of a user for every action done on the website till they logout. To logout the frontend simply gets rid of the token and the user is logged out until they send the jwt agin while logging in. The jwt is attached with each request the frontend makes to the backend and the token tells the backend which user is making the request.

Wishlist of Properties

Wishlist feature of the system is developed in django. It is quite similar to the shopping cart. The differences are, that a wishlist does not keep information about product quantities nor does a wishlist manage any extra item fields. It just allows signed-in users to create personalized collections of products and save them in their user account for future reference.

Every property has an add to wishlist option and every customer always has one and only one active wishlist. He can add and delete properties only from this wishlist. A wish list is created the first time, a customer adds a property to a wishlist.

Recommender System

The recommender system was deployed as microservice using flask, with MongoDB as the database.
It uses a collaborative filtering approach using SVD(Singular Value Decomposition) and relies on implicit ratings for this task. An implicit rating is a rating used to gauge a user's interests in properties based on actions that he/she carries out. Hover over, click and add to cart are used here.
These implicit ratings are stored in a Mongo database and SVD is applied to them to identify properties that a particular user is most likely to be

interested in. The top 7 properties are recommended to the user so that he/she may look at them. The recommender system is an important part of the product that helps showcase relevant properties and keeps the user's interest levels high while using the website.

Traffic and Air Quality

Traffic data was retrieved using an external API TomTom. Which takes latitude and longitude as input and provides output as closeness to main roads, average traffic speed, average travel time. Air quality index and pollution data for a location was retrieved using Breezometer API. It provides information such as air quality, dominant pollutant in a locality. We provide these data for every property and users may filter to look for areas with less traffic and less pollution.