



**slington college**  
(इस्लिङ्टन कलेज)

**Module Code & Module Title**  
**CS6P05NI Final Year Project**

**Assessment Weightage & Type**  
**25% FYP Interim Report**

**Semester**  
**2021 Autumn**

**Project Title: Rent Management Web app, Tarif**

**Student Name: Suyogya Luitel**

**London Met ID: 19031784**

**College ID: np01cp4a190035**

**Internal Supervisor: Mohit Sharma**

**External Supervisor: Ankit Tandukar**

**Assignment Due Date: 15<sup>th</sup> December, 2021**

**Assignment Submission Date: 15<sup>th</sup> December, 2021**

**Word Count (Where Required): 3801**

*I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.*

## **Abstract**

This report demonstrates the early phases of the development of the Rent Management Web app based on React and Django. Even as the world has entered the stage of electronic data storage, people in the rental domain still either keep a record of rent data on paper or do not store it at all. Keeping in mind the importance of data analysis to make educated decisions, this project aims to digitally store the rental data of the users and also make it easier to find houses to rent.

## Table of Contents

1. Introduction.....	1
1.1. Introduction to the topic.....	1
1.2. Current Scenario .....	1
1.3. Problem Scenario.....	2
1.4. Project as solution.....	2
1.5. Aim and Objectives: .....	3
1.5.1. Aim.....	3
1.5.2. Objectives .....	3
1.6. Report Structure.....	4
1.6.1. Background .....	4
1.6.2. Development .....	4
1.6.3. Progress Breakdown .....	4
1.6.4. Future work .....	4
2. Background .....	5
2.1. Client description and requirements.....	5
2.1.1. Description .....	5
2.1.2. Requirements.....	5
2.2. Understanding the project .....	8
2.2.1. Web app as a medium .....	8
2.2.2. Project elaboration .....	8
2.2.3. Project deliverables.....	8
2.3. Similar systems.....	10
2.3.1. System 1: Onsitepropertymanager.....	10
2.3.2. System 2: Vacationlabs.....	10
2.3.3. System 3: Buildium .....	11

2.3.4.	System 4: Neopropertynepal.....	12
2.3.5.	System 5: Property Care Guru .....	13
2.4.	Similar Systems Comparison .....	14
2.4.1.	Comparison Table.....	14
2.4.2.	Comparison Result.....	14
3.	Development .....	15
3.1.	Methodologies.....	15
3.1.1.	Iterative Waterfall .....	15
3.1.2.	Prototyping.....	16
3.1.3.	Feature Driven Development .....	17
3.2.	Adopted Methodology (FDD) .....	19
3.3.	Development Till Date.....	20
3.3.1.	Entity Relation Diagram.....	20
3.3.2.	UI Designs.....	21
3.3.3.	Use Case Diagram .....	29
4.	Progress analysis .....	38
4.1.	Progress Analysis Table .....	38
4.2.	Progress Analysis Report.....	39
4.2.1.	Current scenario of progress.....	39
5.	Future work.....	40
5.1.	Phases to complete.....	40
5.1.1.	Build Feature List .....	40
5.1.2.	Plan by Feature.....	40
5.1.3.	Design by feature .....	40
5.1.4.	Build by Feature .....	40
6.	References .....	41

7. Appendix.....	43
7.1. Appendix A (Development) .....	43
7.1.1. Gant Chart.....	43
7.1.2. Work Breakdown Structure .....	44
7.2. Appendix B (Development Till Date) .....	45
7.2.1. Code Snippets.....	45
7.2.2. Viewable Development .....	50

## Table of Figures

Figure 1: System 1 .....	10
Figure 2: System 2.....	11
Figure 3: System 3.....	12
Figure 4: System 4.....	13
Figure 5: System 5.....	13
Figure 6: Iterative waterfall model (Ruparelia, 2010) .....	15
Figure 7: Prototyping Model (Martin, 2021).....	16
Figure 8: FDD project Lifecycle (Abler, 2021) .....	18
Figure 9: Entity Relation Diagram .....	20
Figure 10: Sign-in page UI .....	21
Figure 11: Sign up page UI .....	22
Figure 12: Home Page UI .....	23
Figure 13: Houses page UI .....	24
Figure 14: Bills page UI.....	25
Figure 15: Analysis page UI.....	26
Figure 16: Profile page UI .....	27
Figure 17: Map page UI .....	28
Figure 18: Use Case Diagram .....	29
Figure 19: Updated Gantt Chart.....	43
Figure 20: Gantt chart Tasks for Clarity .....	43
Figure 21: Work Breakdown Structure .....	44
Figure 22: App.js Code .....	45
Figure 23: App.css Code .....	46
Figure 24: Button Component Code .....	47
Figure 25: Login form Code .....	47
Figure 26: Button style Code .....	48
Figure 27: Login Page Code .....	49
Figure 28: Sign-in page.....	50

**Table of Tables**

Table 1: System Comparison Table.....	14
Table 2: Progress Table.....	38

## **1. Introduction**

### **1.1. Introduction to the topic**

Web apps have revolutionized the way we use the internet. While initially, web browsers could only render simple HTML contents, nowadays browsers are powerful enough to run heavy web apps and even heavy 3D RPG games, thanks to the JavaScript engines built into browsers. People use the internet very frequently, making it the perfect platform for reaching a huge mass of people. The beauty of a web app is that it runs on any platform so long as you have an internet connection and a browser, be it IOS, Windows, Android, Linux, or any other OS. Using a web app, both the tenants and householders can find one another more easily and communicate. The ratings help both the tenants and householders in screening the available options.

### **1.2. Current Scenario**

The pandemic made people more aware and enthusiastic about cashless payment options. While managing rents looks trivial, it can get out of hand very quickly if you own multiple properties to rent. Besides, traditionally, people either do not keep a record of the rents they push or pay; or record it on paper. Keeping a record of rents is extremely useful as it can provide statistical benefits. Comparing the rents paid for different houses can give one an idea of what the market is offering them which can be used in decision making while searching for rents. On the flip side, keeping tabs of bills pushed or rent collected can be valuable to householders to gauge what people are willing to pay and what their property's worth should be to tenants. By keeping a record of rents paid/collected, the users can make educated decisions about what properties are worth and what amount they should be paying.



### **1.3. Problem Scenario**

The problems that the web app intends to solve are as follows:

- i. Finding a rent or finding a tenant is very difficult given the limited means of reaching out to one another in Nepal.
- ii. People are opting to spend money via cashless means due to the pandemic. However, house rents are paid mostly via cash when the householder visits the tenant door to door.
- iii. Pushing consistent bills to tenants takes quite a bit of time as householders mostly need to do all the calculations themselves, write it all down on a piece of paper, sign it, and manually send it to tenants.
- iv. While pushing a bill or paying a bill, most of the parties involved in house rents, do not keep track of the transactions being made. That is ignorance of data which could serve as a valuable asset in decision making.
- v. Often there is no reliable way for screening available options of householders and tenants. Thus, after a contract is signed, either party could end up stuck with a bad or unwanted company.

### **1.4. Project as solution**

The project solves the aforementioned issues by the following means:

- i. The app displays a map-based view which helps users locate houses ready to be rented. Similarly, since the houses are now viewable on a map, the householders can find tenants more frequently.
- ii. The app provides a cashless means to pay rent if the user desires to do so.
- iii. The app supports the development of a bill template which the householders can then use to push consistent bills to all tenants.
- iv. The app keeps track of all rent-related transactions and visualizes the data in form of a graph so that the user can make educated decisions.

The app supports ratings and reviews which help in screening both tenants and householders.

## **1.5. Aim and Objectives:**

### **1.5.1. Aim**

The major aim of the project is to make finding and managing rents easier. The project helps tenants find reliable rent and help householders find reliable tenants. To help tenants and householders make educated guesses based on their previous data. To provide tenants and householders a reliable way to pay and push bills respectively via e-payment.

### **1.5.2. Objectives**

The objectives of this web app are:

- To learn about web development and its required paradigms
- To learn about database designing in a real-world scenario
- To learn how restful APIs work
- To better understand third-party APIs in development such as map APIs.
- To better understand frontend technologies like react, and JS in general.
- To better understand backend technologies such as Django and Django REST framework.

## **1.6. Report Structure**

### **1.6.1. Background**

The background section of the report clarifies the project requirements, project description, and end-users. The section also covers a comparison of the project with other similar projects to establish how the project is different from some of the existing solutions.

### **1.6.2. Development**

The development section of the report elaborates how the project is to be developed. The section also covers the considered and selected methodologies while also describing the several phases of the selected methodology. The development process is also visually represented with the help of the Work Breakdown Structure and Gantt Chart.

### **1.6.3. Progress Breakdown**

The progress breakdown section of the report specified the progress made to date for the project.

### **1.6.4. Future work**

The last section of the report, future work, describes the work that needs to be done for the project to be completed.

## 2. Background

### 2.1. Client description and requirements

Client name: Saral Karki

#### 2.1.1. Description

The client is Saral Karki, an individual, who resides in a rental house and wants a web app to enhance their rent experience. Having experienced rental issues first-hand, the client knows the common issues faced in the house rental domain.

Upon presenting this project to Saral, he found the project helpful in resolving the issues he has seen in the house rental domain. He agreed to be the client for this project as he sees practicability in the project and is thus willing to provide his insight on the rental domain.

#### 2.1.2. Requirements

##### 1. The householder and Tenant registration with citizenship

The user registers as a householder or a tenant. While registering, the user must upload a scanned copy of their citizenship which is also stored for security purposes. While it is not a common practice in Nepal for householders to take a copy of the tenant's citizenship, it is a good practice to have it available. Having citizenship attached to an account also significantly reduces the chances of fraud or run-away cases.

##### 2. Houses must be visible on a map

The web app must contain a map displaying the houses available for rent which is visible to users after they sign in. This makes it easier for tenants to find houses for rent in their desired location.

### 3. Push bills as a householder

The web app must enable householders to make house bills and push the bills via email. This makes it easier to calculate the rent amount as well as removes the hassle of physically visiting the tenant or handing out bills.

### 4. Pay bills as Tenant

The pushed bills must have an e-payment option so that tenants can pay the bills virtually. This removes the hassle of physically meeting the householder just for paying the rent amount.

### 5. Leave review and rating for tenant and householder

Both householders and tenants must be able to leave a rating on one another after their term has ended. This ensures that there is a way for screening the tenants/householders even before the interview takes place.

### 6. Register houses as a householder

A householder must be able to have multiple houses registered since a user may have multiple houses.

### 7. Keep track of payment amounts and dates and visualize these data

The web app must also ensure that it stores bill details and bill dates paid by each user. The web app must visualize the given user data in form of a graph to enable educated decision making.

8. Visualize user data for educated decision making.

The web app must visualize the given user data in form of a graph to enable educated decision making.

## **2.2. Understanding the project**

### **2.2.1. Web app as a medium**

Web apps have revolutionized the way we use the internet. While initially, web browsers could only render simple HTML contents, nowadays browsers are powerful enough to run heavy web apps and even heavy 3D RPG games, thanks to the JavaScript engines built into browsers. People use the internet very frequently, making it the perfect platform for reaching a huge mass of people. The beauty of a web app is that it runs on any platform so long as you have an internet connection and a browser, be it IOS, Windows, Android, Linux, or any other OS.

### **2.2.2. Project elaboration**

The project is a web app built using React and Django frameworks. React is the frontend framework used for making the project viewable in the browser and to collect input from the users. Django is the backend framework for storing user information. The project uses restful APIs to transfer data between the backend and the frontend. The web app registers users as a householder or a tenant. Upon sign in, they are given access to further features such as finding available houses, registering available houses, e-payment for bills, and a communication portal. Once both the householder and the tenant have selected one another, the house is marked rented, and this term can be ended by the householder. After the term has ended, the house is marked again as available.

### **2.2.3. Project deliverables**

The expected outcomes and deliverables of the web app are:

- Maps for navigation
  - Points out the house available for rent

- Make and push bills as a householder
- Receive bills and Make payments as a tenant
- Leave ratings on either party
- Keep track of payment amounts and dates
- Communication portal for connecting householders to tenants
- Data visualization in a graph for both householders and tenants



## 2.3. Similar systems

### 2.3.1. System 1: Onsitepropertymanager

URL: <http://www.onsitepropertymanager.com/>

This system manages property rental with automated postings, rent roll format, and Tenant portals. However, it does not help tenants find householders or rent.



Figure 1: System 1

### 2.3.2. System 2: Vacationlabs

URL: <https://www.vacationlabs.com/rental-management-system/>

This system handles vacation-related rentals such as hotels, tours, etc. It does not solve the issue of tenants and householders.

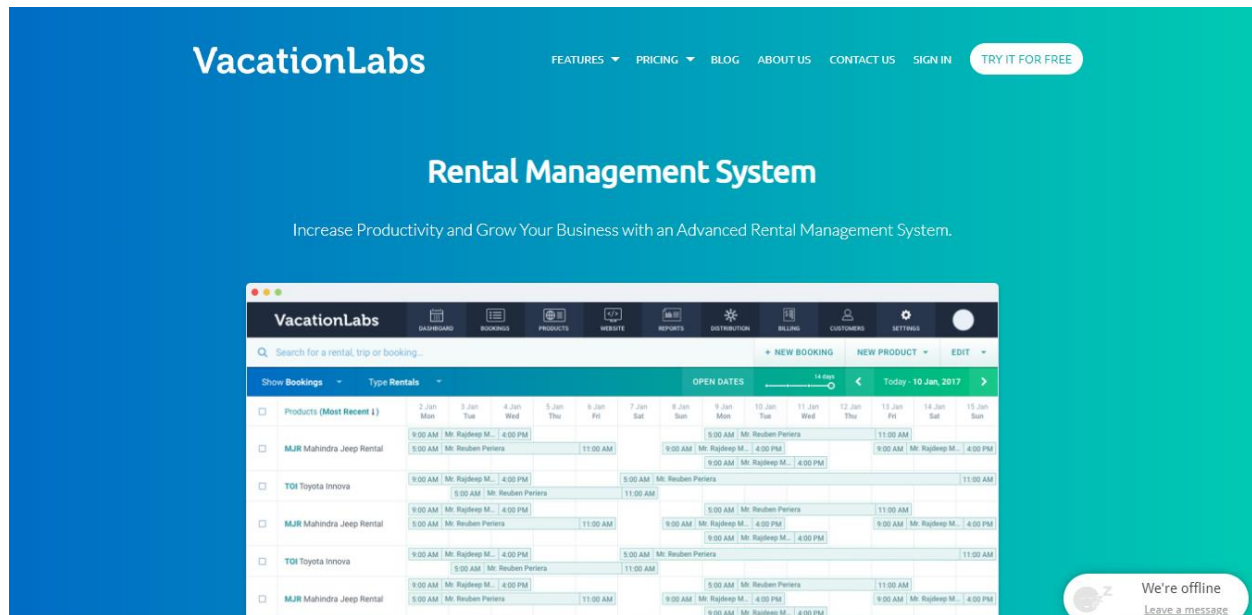


Figure 2: System 2

### 2.3.3. System 3: Buildium

URL: <https://learn.buildium.com/>

This system is a pay-to-use system that helps in ePay, Tenant Screening, Renters Insurance, and data visualization reports. While the system is great, it is not used in the case of Nepal since it is a pay-to-use system.

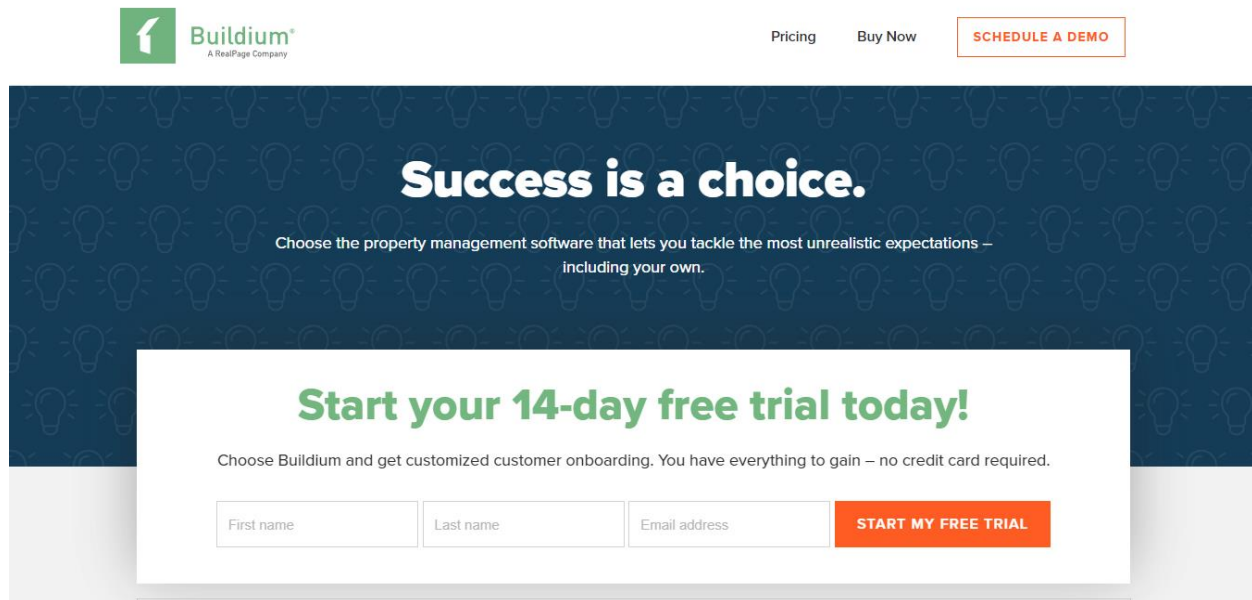


Figure 3: System 3

#### 2.3.4. System 4: Neopropertynepal

URL: <https://www.neopropertynepal.com/>

This system is based in Nepal and helps users with property sales and rental. It includes houses, offices, land, hotel, apartments, etc. While the app is great, it cannot be used for tenant or householder filtering since it shows advertisement-based property and has no system for rating.

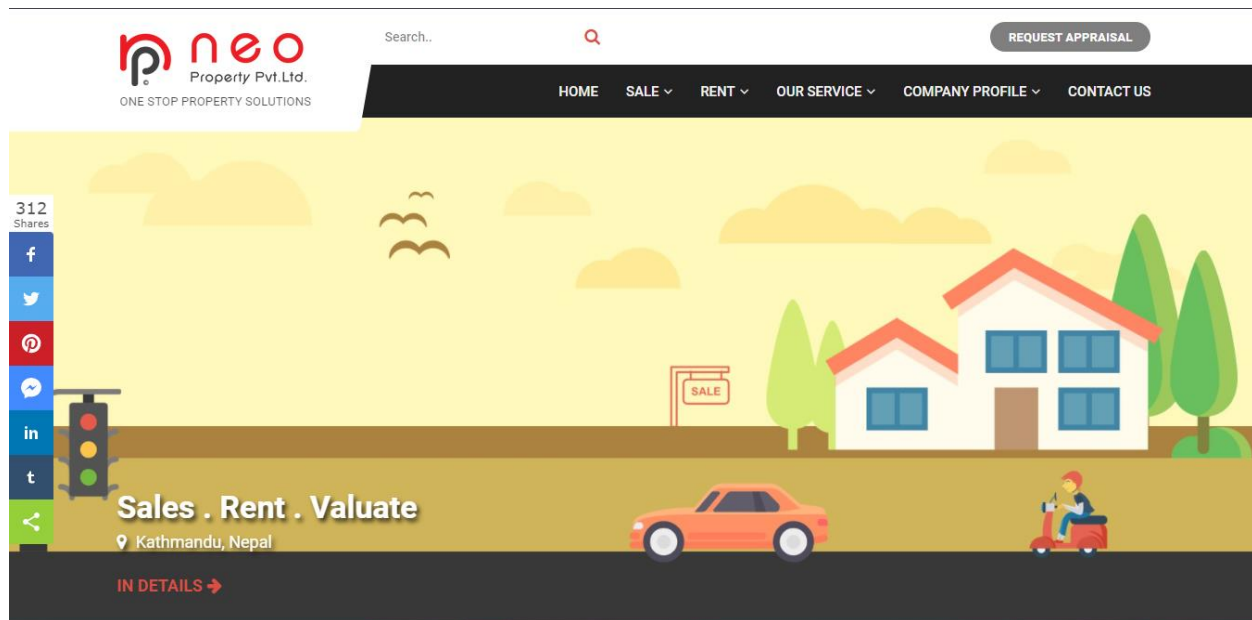


Figure 4: System 4

### 2.3.5. System 5: Property Care Guru

URL: <http://propertycareguru.com/>

This system is also based in Nepal and helps users find vacant properties. The app has a lot of property filtering options but similar to the previous system, has no system for rating and reviewing.

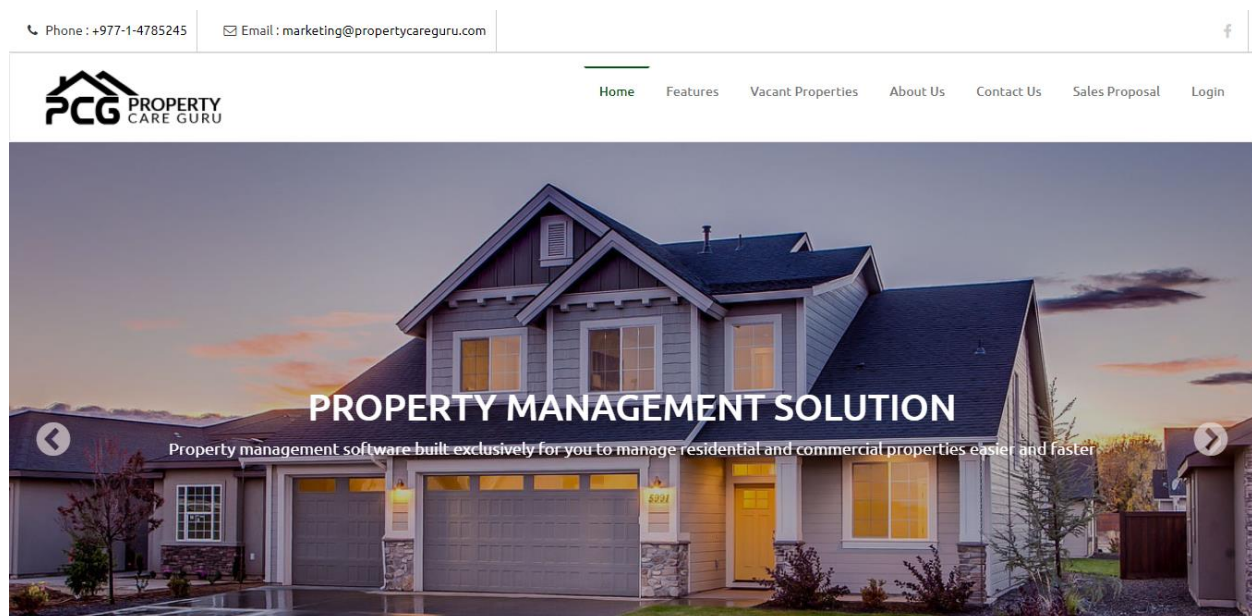


Figure 5: System 5

## 2.4. Similar Systems Comparison

### 2.4.1. Comparison Table

Features	System 1	System 2	System 3	System 4	System 5	My Project
Availability in Nepal	✗	✗	✗	✓	✓	✓
Free to use	✗	✗	✗	✓	✓	✓
Rating and Review	✗	✓	✓	✗	✗	✓
Map	✗	✗	✗	✗	✗	✓
Push bills	✓	✗	✓	✗	✓	✓
E-payment	✓	✗	✓	✗	✓	✓
Communication Portal	✓	✗	✓	✗	✗	✓
Data Analysis	✗	✓	✗	✗	✗	✓

Table 1: System Comparison Table

### 2.4.2. Comparison Result

The comparison done above shows that while there are multiple systems in the rent management domain, none have all the set of features present in my system. The system adheres to the requirements provided by the client.

### 3. Development

#### 3.1. Methodologies

##### 3.1.1. Iterative Waterfall

The waterfall methodology is a linear project management approach, where stakeholder and customer requirements are gathered at the beginning of the project, and then a sequential project plan is created to accommodate those requirements (Project Manager, 2021). The methodology phases do not overlap one another and each phase is carried out sequentially. While being cheap, the project failure chances for long-term projects are maximized in this methodology since it cannot handle client requirement variation very well. Any changes in pre-existing project architecture can result in changes being made to every unit that has already been completed. (Prepinsta, 2020)

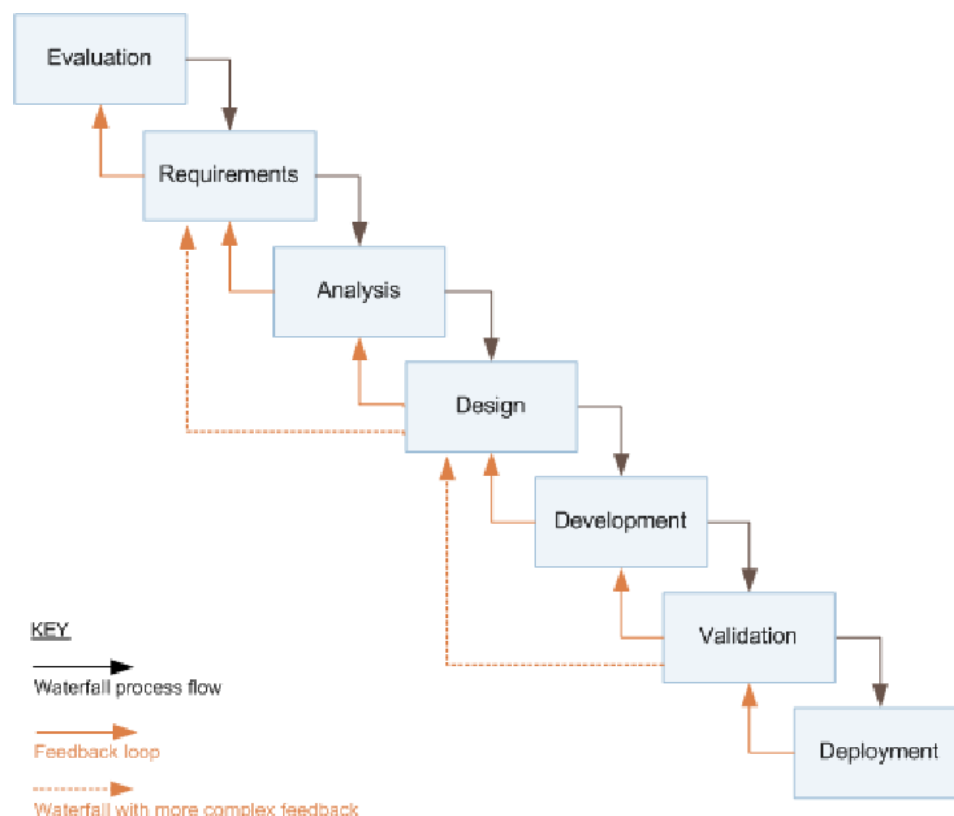


Figure 6: Iterative waterfall model (Ruparelia, 2010)

### 3.1.2. Prototyping

A prototype is an early sample, model, or release of a product built to test a concept or process or to act as a thing to be replicated or learned from (Lumitex, 2017). This implies that a prototype is an instrument to evaluate an idea. In prototype methodology, a prototype of the project is made and it is then refined over and over till the app meets client satisfaction. The methodology makes addressing client requirement variation simpler. It is well suited for long-term projects where client requirement is not clear or is prone to frequent changes. (Lewis, 2019)

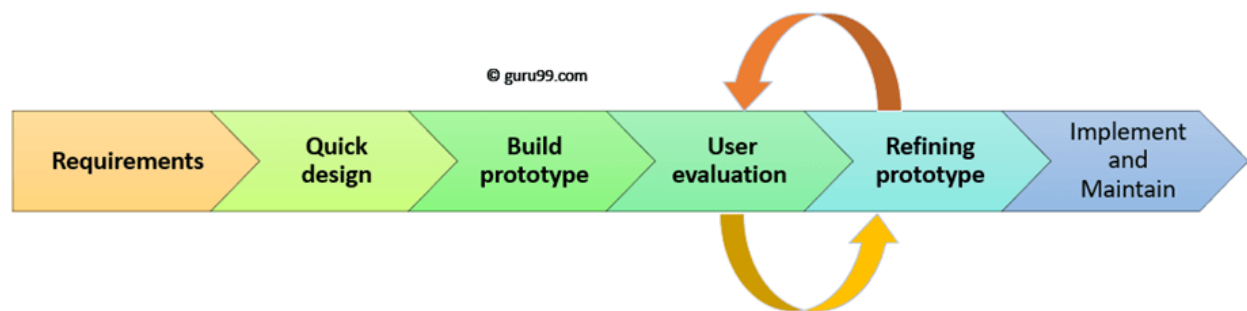


Figure 7: Prototyping Model (Martin, 2021)

### 3.1.3. Feature Driven Development

An Agile methodology for developing software, Feature-Driven Development (FDD) is customer-centric, iterative, and incremental, with the goal of delivering tangible software results often and efficiently (PlanView, 2021). As the name suggests, this methodology focuses more on features. In FDD, a feature is expected to be delivered every 2-10 days. The methodology is followed using the below steps:

- Develop overall model
- Build feature list
- Plan by feature
- Design by feature
- Build by feature

Instead of traditional project milestones, FDD uses features to organize its activities. The first three steps are completed initially and the rest of the steps are repeated for every feature. If the client decides to add any more features or changes the requirements, the entirety of the five steps is carried out again in an incremental fashion. (PlanView, 2021)



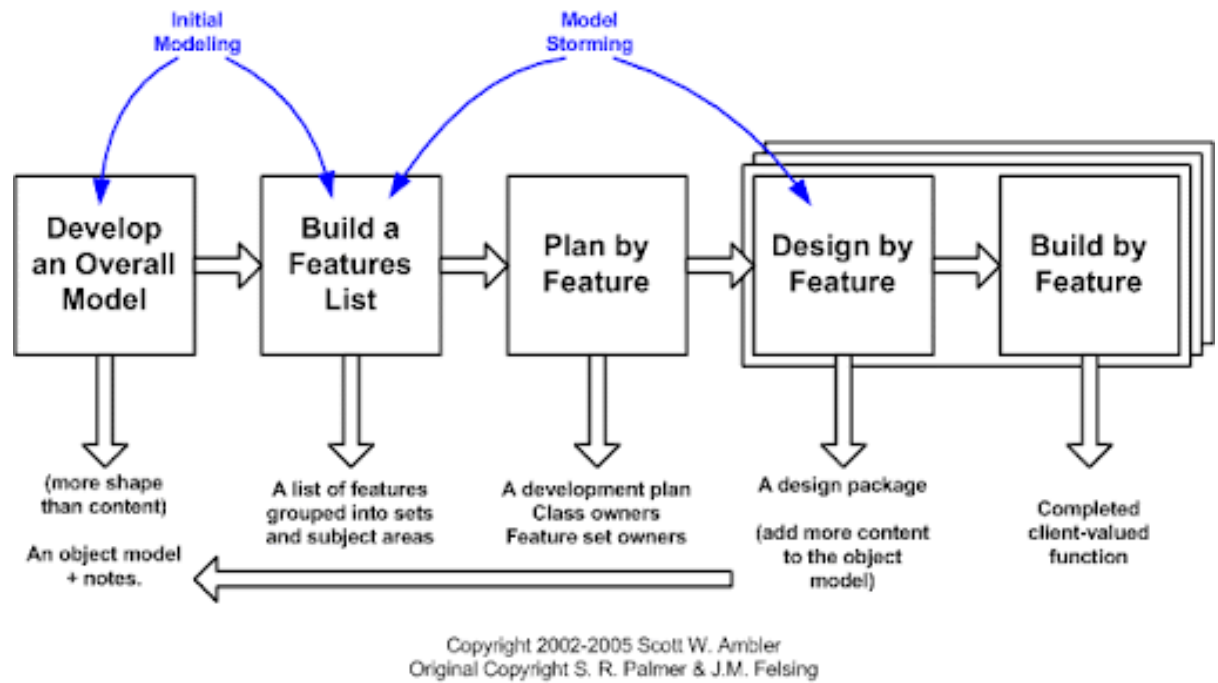


Figure 8: FDD project Lifecycle (Abler, 2021)

### **3.2. Adopted Methodology (FDD)**

Despite being a methodology for large projects, I adopted Feature Driven Development due to the following reasons:

- The methodology requires fewer meetings and is documentation-oriented.
- It uses a user-centric approach with the client in mind.
- It is very scalable and can address the growth of the project.
- It breaks feature sets into smaller chunks and regular iterative releases, which makes it easier to track and fix coding errors, reduces risk, and allows you to make a quick turnaround to meet your client's needs (Lucid Content Team, 2021).

Further Content: [Appendix A](#)

### 3.3. Development Till Date

#### 3.3.1. Entity Relation Diagram

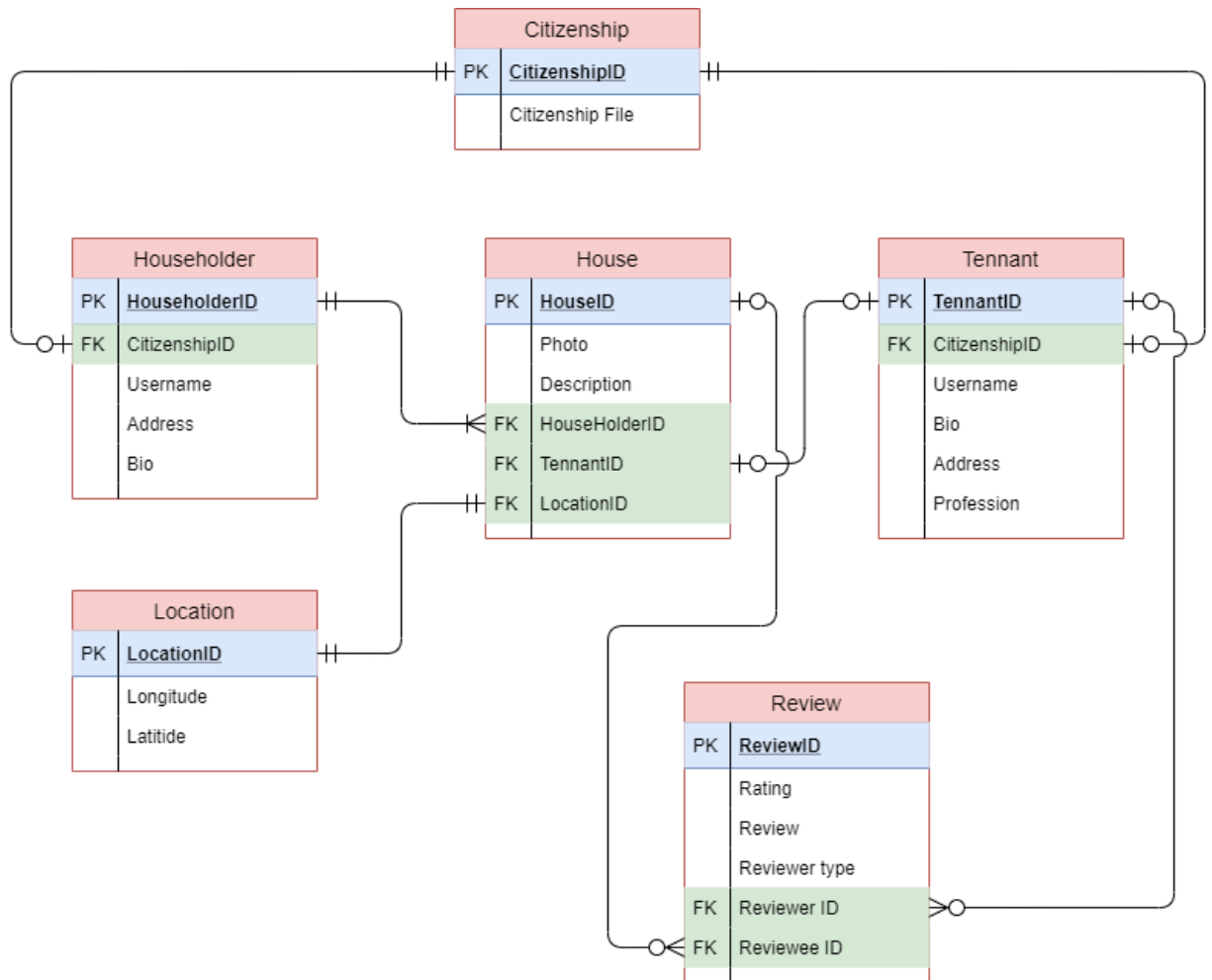
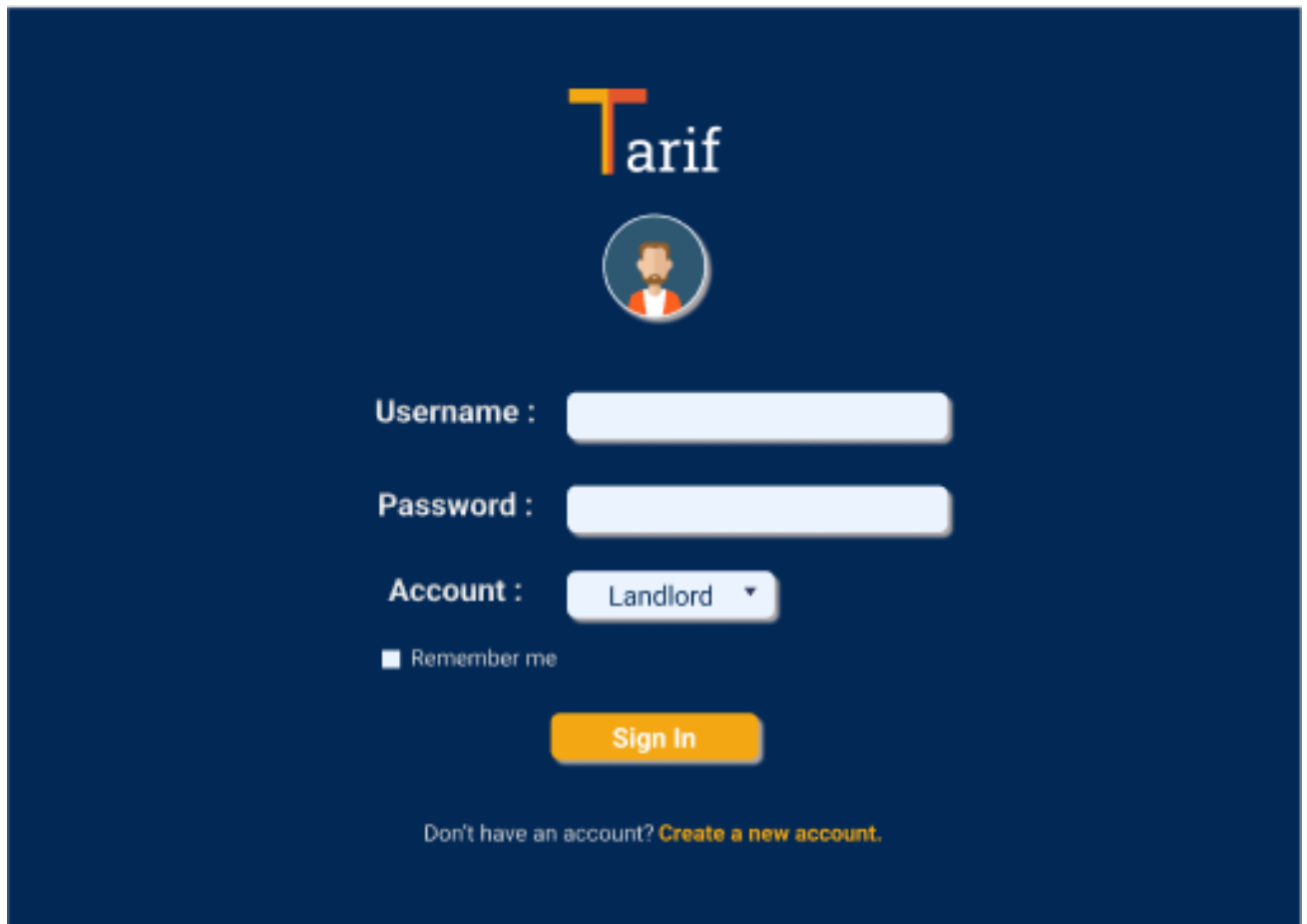


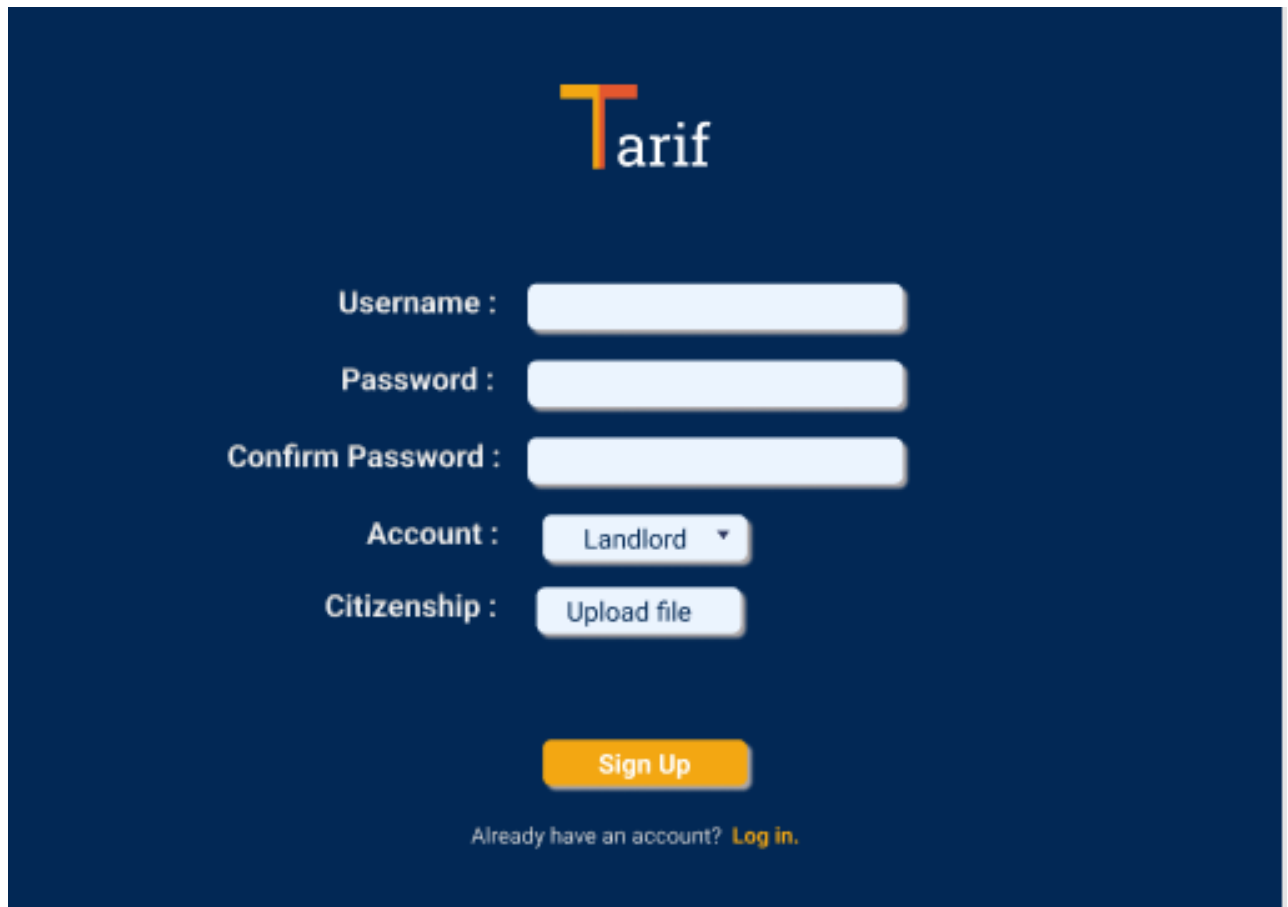
Figure 9: Entity Relation Diagram

### 3.3.2. UI Designs



The image shows a sign-in page for a service named "Tarif". The page has a dark blue background. At the top center is the "Tarif" logo, which consists of a stylized orange "T" followed by the word "arif" in white. Below the logo is a circular profile picture of a man with a beard and orange hair. The form contains three input fields: "Username :", "Password :", and "Account :". The "Account :" field is a dropdown menu currently showing "Landlord". Below these fields is a checkbox labeled "Remember me". A large orange "Sign In" button is positioned below the checkbox. At the bottom, there is a link that says "Don't have an account? [Create a new account.](#)".

Figure 10: Sign-in page UI



The image shows a sign-up page for a service called "Tarif". The page has a dark blue background. At the top center is the "Tarif" logo, with a stylized orange "T" and the word "arif" in white. Below the logo are five form fields, each with a label to its left: "Username :", "Password :", "Confirm Password :", "Account :", and "Citizenship :". The "Username", "Password", and "Confirm Password" fields are white text boxes. The "Account" field is a dropdown menu with "Landlord" selected. The "Citizenship" field is a button labeled "Upload file". Below these fields is a large orange "Sign Up" button. At the bottom, there is a link that says "Already have an account? [Log in.](#)".

Figure 11: Sign up page UI

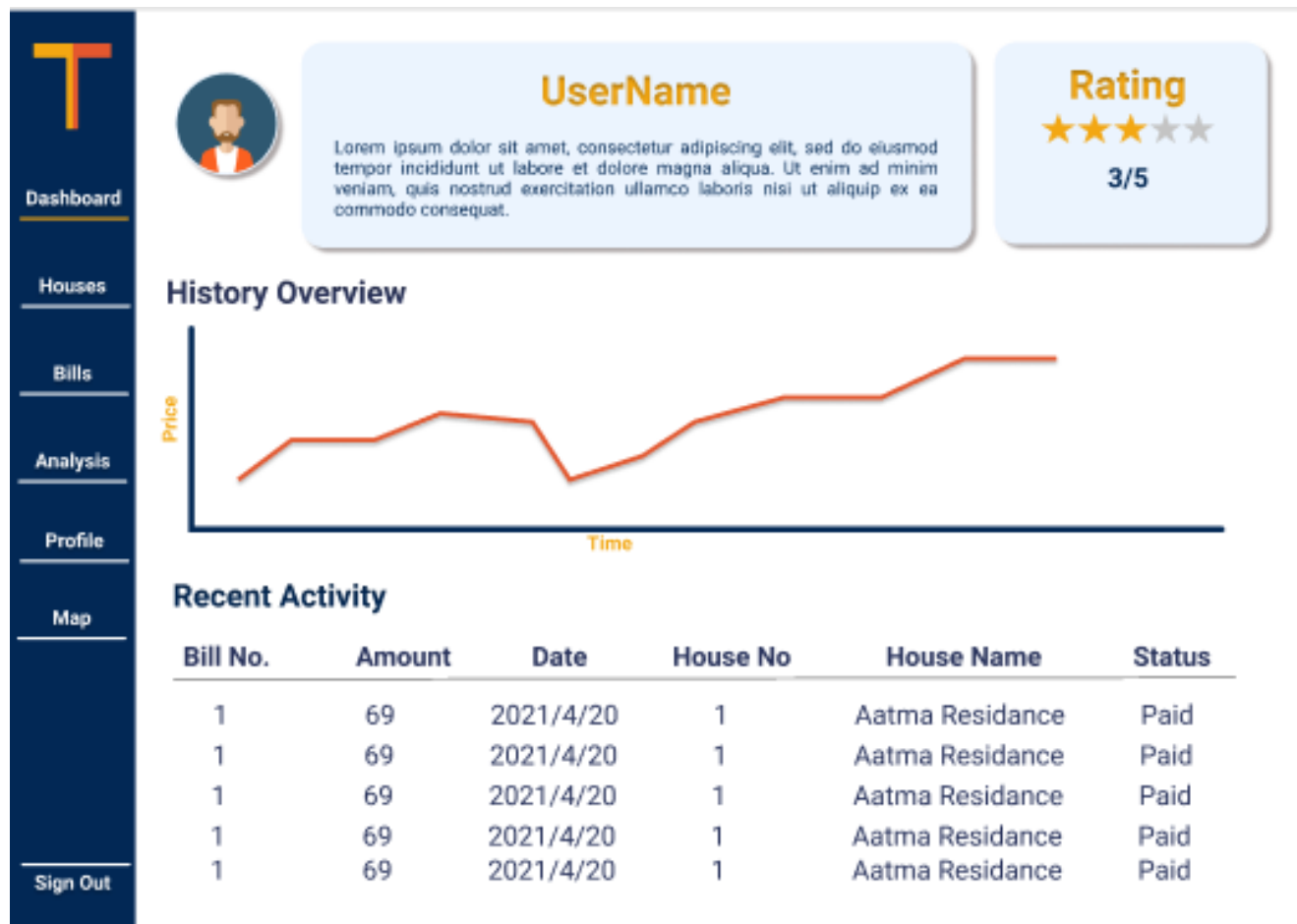


Figure 12: Home Page UI



Figure 13: Houses page UI

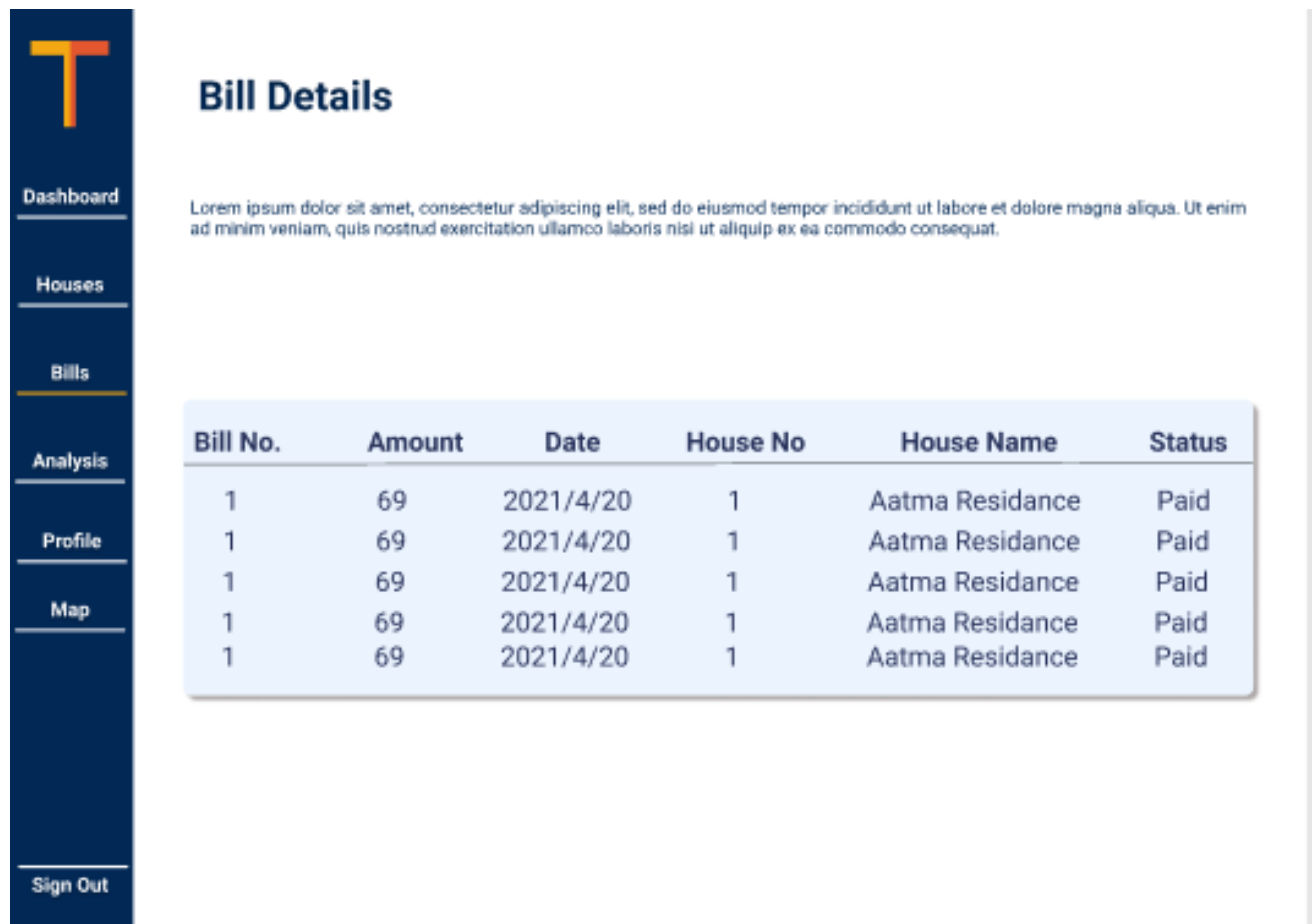


Figure 14: Bills page UI



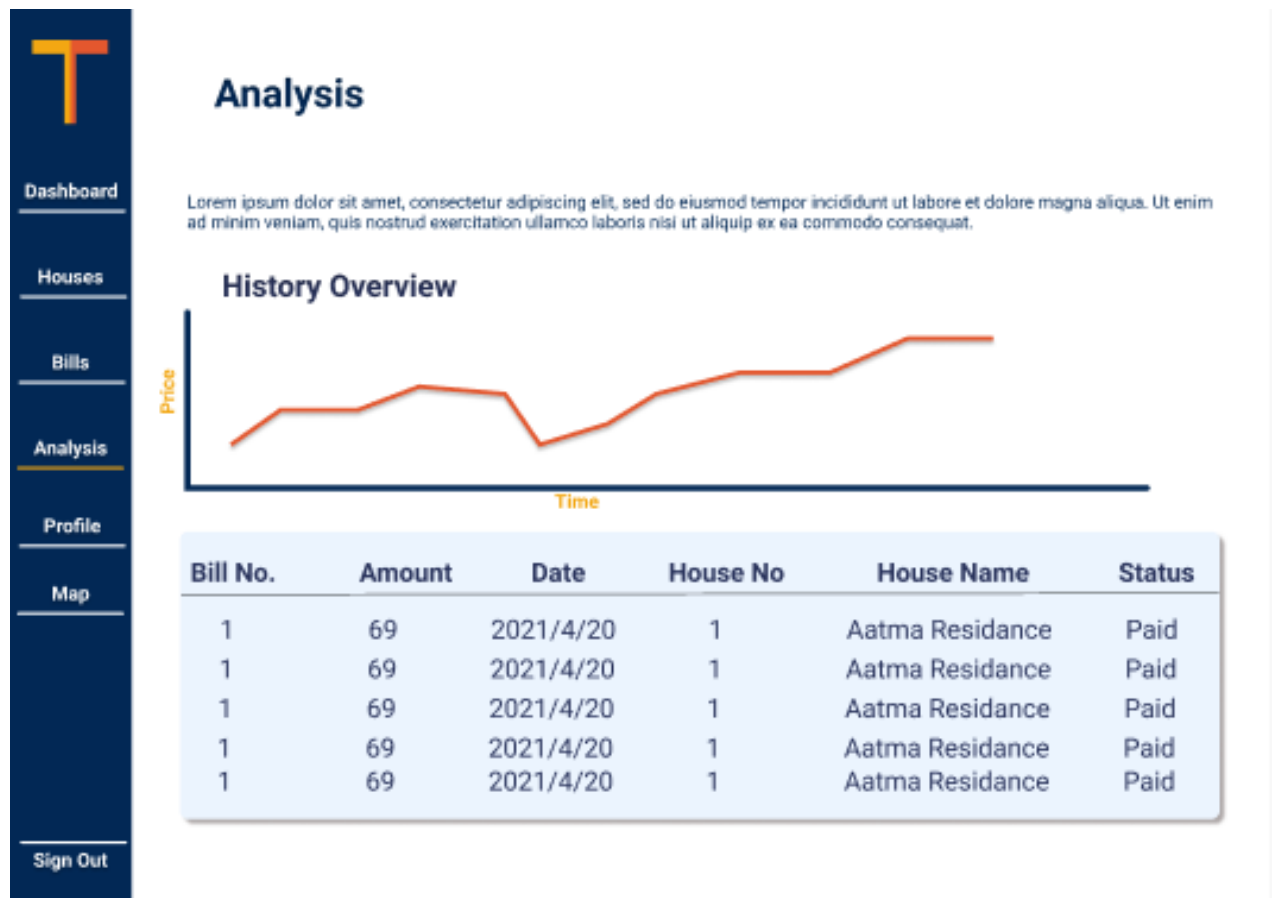


Figure 15: Analysis page UI

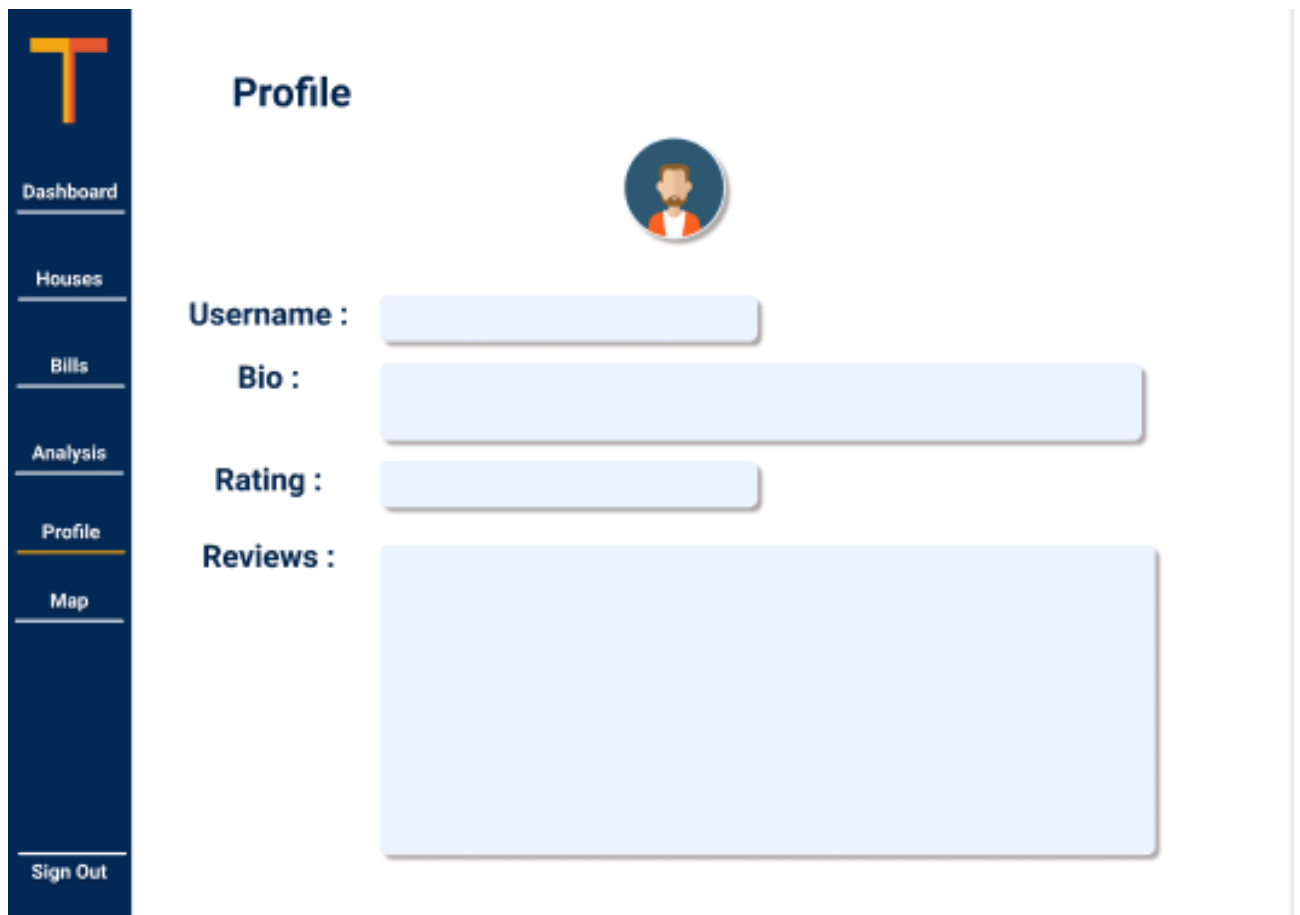


Figure 16: Profile page UI

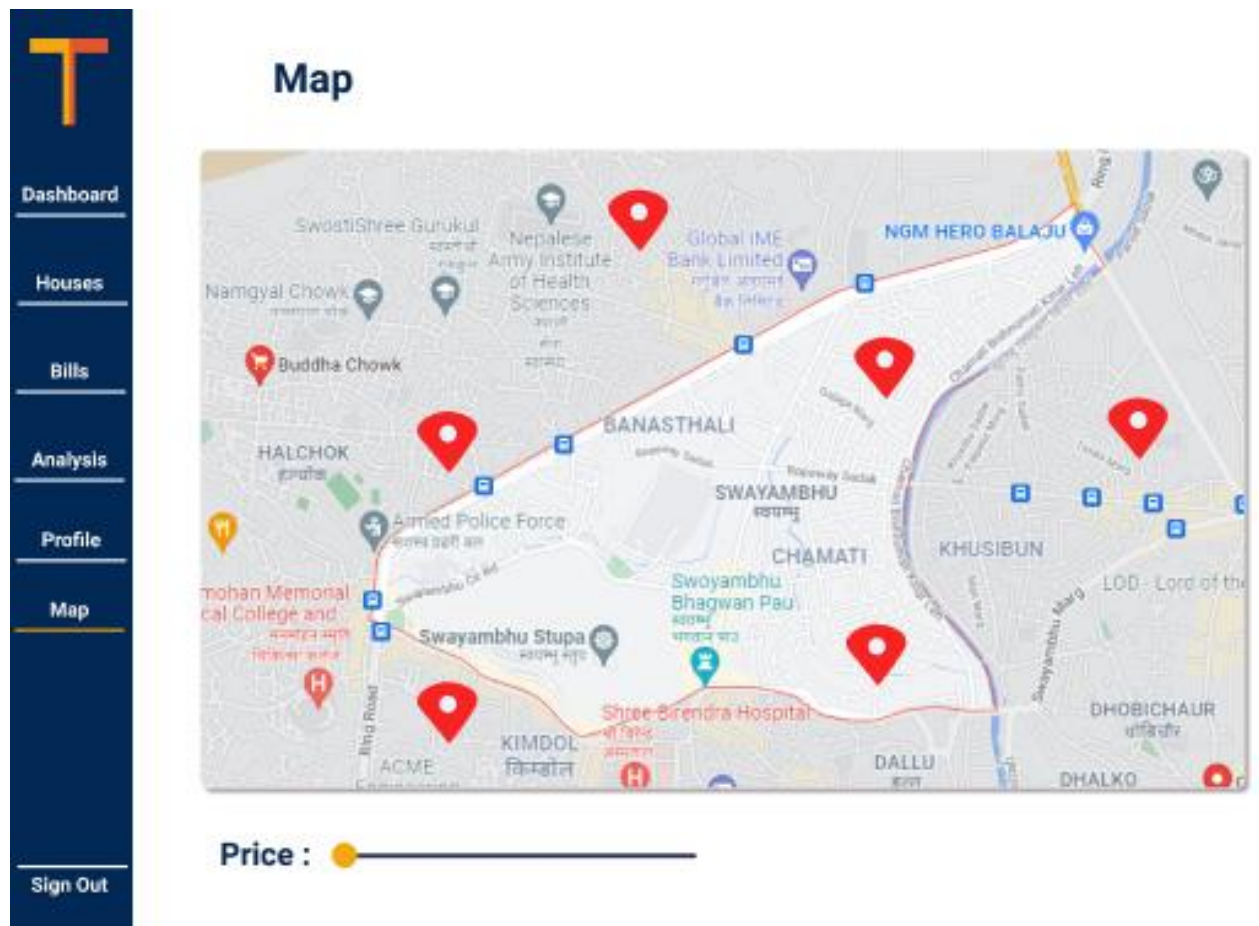


Figure 17: Map page UI

### 3.3.3. Use Case Diagram

#### 3.3.3.1. Diagram

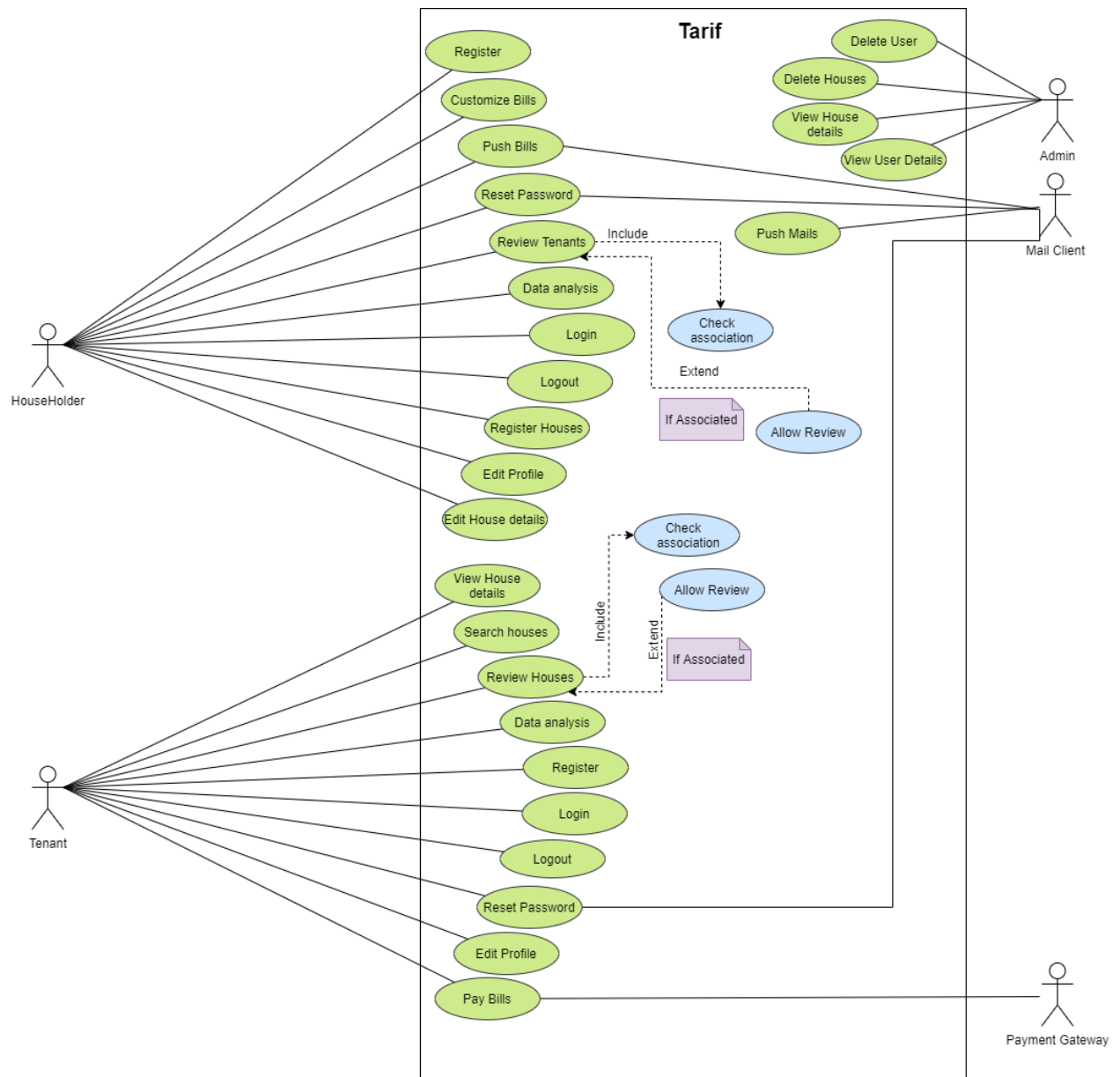


Figure 18: Use Case Diagram

### 3.3.3.2. High-Level Description

#### Use Case 1

**Use Case:** Householder Register

**Actor:** Householder

**Description:** The householder enters their details in the system. The system checks if the user is already registered. If not, the registration executes successfully, otherwise, the system gives an appropriate error message to the user.

#### Use Case 2

**Use Case:** Customize Bills

**Actor:** Householder

**Description:** The householder selects the fields, along with price details, that they want on the bills. The system then generated bills for the house according to this customization made by the householder.

#### Use Case 3

**Use Case:** Push Bills

**Actor:** Householder, Mail Client

**Description:** The householder enters the bill details and presses the push bill button. The system then calculates the bill, asks the user for confirmation, and sends the details to the mail client. The mail client then pushes the bill to the tenants through their email.

#### Use Case 4

**Use Case:** Register Houses

**Actor:** Householder

**Description:** The householder fills in the details of their house and registers it in the system. Registered houses can then be displayed as available for rent by the system.

#### Use Case 5

**Use Case:** Review Tenants

**Actor:** Householder

**Description:** Once a contract with a tenant has been terminated, the householder can then review the tenant. The review consists of a rating out of five and a review text.

#### Use Case 6

**Use Case:** Data Analysis

**Actor:** Householder

**Description:** The householder views their past data collected in the system in the form of a chart for gaining information.

#### Use Case 7

**Use Case:** Householder Login

**Actor:** Householder

**Description:** The householder enters their credentials and submits them to the system. The system verifies the credentials and if verified, gives the householder access to the additional features in the system.

#### Use Case 8

**Use Case:** Householder Logout

**Actor:** Householder

**Description:** The householder logs out of the web app by pressing the logout button.

#### Use Case 9

**Use Case:** Householder Reset Password

**Actor:** Householder, Mailing Client

**Description:** The householder requests a password reset. Once requested, the mailing system sends a confirmation mail to the householder. After confirmations are done, the householder enters a new password to override their previous password.

#### Use Case 10

**Use Case:** Householder Edit Profile

**Actor:** Householder

**Description:** The householder presses the edit button. The fields in the profile page then become editable and the householder updates new data in the fields.

#### Use Case 11

**Use Case:** Householder Edit House Details

**Actor:** Householder

**Description:** The householder presses the edit button. The fields in the house page then become editable and the householder updates new data in the fields.

#### Use Case 12

**Use Case:** Pay Bills

**Actor:** Tenant, Payment Gateway

**Description:** The tenant presses the pay button. A confirmation message pops up and provides the tenant with a payment option. If the payment method involves e-payment, the payment gateway is activated.

#### Use Case 13

**Use Case:** Search Houses

**Actor:** Tenant

**Description:** The tenant is provided with a map and a price slider. The available houses matching the given criteria are displayed on the map.

#### Use Case 14

**Use Case:** Review Houses

**Actor:** Tenant



**Description:** Once a contract with a householder has been terminated, the tenant can then review the house. The review consists of a rating out of five and a review text.

#### Use Case 15

**Use Case:** Data Analysis

**Actor:** Tenant

**Description:** The tenant views their past data collected in the system in the form of a chart for gaining information.

#### Use Case 16

**Use Case:** Tenant Register

**Actor:** Tenant

**Description:** The tenant enters their details in the system. The system checks if the user is already registered. If not, the registration executes successfully, otherwise, the system gives an appropriate error message to the user.

#### Use Case 17

**Use Case:** Tenant Login

**Actor:** Tenant

**Description:** The tenant enters their credentials and submits them to the system. The system verifies the credentials and if verified, gives the tenant access to the additional features in the system.

## Use Case 18

**Use Case:** Tenant Logout

**Actor:** Tenant

**Description:** The tenant logs out of the web app by pressing the logout button.

## Use Case 19

**Use Case:** Tenant Reset Password

**Actor:** Tenant, Mailing Client

**Description:** The tenant requests a password reset. Once requested, the mailing system sends a confirmation mail to the tenant. After confirmations are done, the tenant enters a new password to override their previous password.

## Use Case 20

**Use Case:** Tenant Edit Profile

**Actor:** Tenant

**Description:** The tenant presses the edit button. The fields in the profile page then become editable and the tenant updates new data in the fields.

## Use Case 21

**Use Case:** View House Details

**Actor:** Tenant

**Description:** The tenant selects the house on the map. The system then presents the tenant with the details of that particular house.

## Use Case 22

**Use Case:** Delete User

Actor: Admin

**Description:** The admin deletes a user from the system.

## Use Case 23

**Use Case:** Delete House

Actor: Admin

**Description:** The admin deletes a house from the system.

## Use Case 24

**Use Case:** View House Details

Actor: Admin

**Description:** The admin views house information uploaded into the system by the user.

## Use Case 25

**Use Case:** View User Details

Actor: Admin

**Description:** The admin views user information uploaded into the system by the user.

## Use Case 26

**Use Case:** Push mails**Actor:** Mail Client**Description:** The mail client pushes mail to the users when the system calls it to.Further Content: [Appendix B](#)

## 4. Progress analysis

### 4.1. Progress Analysis Table

S.N.	Task	Status	Progress
1	Topic Selection	Complete	100%
2	Client Finalization	Complete	100%
3	Requirement Gathering	Complete	100%
4	Risk Analysis	Complete	100%
5	Use Case Development	Complete	100%
6	Entity Relation Diagram	Complete	100%
7	Documentation	Partially Complete	20%
9	Technical Research	Partially Complete	40%
10	UI Design	Complete	100%
11	Domain Object Modeling	Incomplete	0%
12	Build Feature List	Incomplete	0%
13	Plan by Feature	Incomplete	0%
14	Design by Feature	Incomplete	0%
15	Build by Feature	Incomplete	0%

Table 2: Progress Table

## **4.2. Progress Analysis Report**

### **4.2.1. Current scenario of progress**

The project began with topic selection. After an appropriate topic was selected, a feasibility study was done and research on other similar systems was done. A client was found for the selected project and requirements were gathered from the client via an interview. Technical research was then started on how to address the gathered requirements. After technical research, documentation was carried out with the project proposal and interim report. Some useful project management tools such as work breakdown structure and Gantt chart. So far, the project has been smoothly progressing according to the Gantt chart and project milestones set in the proposal

## **5. Future work**

### **5.1. Phases to complete**

#### **5.1.1. Build Feature List**

A feature list is required to be built to define the scope of the project and for further planning, designing, and developing the project. A feature list will be developed in the future as shown in the Gantt chart above.

#### **5.1.2. Plan by Feature**

This phase exists to create a development plan based on the created feature list. Plans will be made for the features to be developed smoothly and on time.

#### **5.1.3. Design by feature**

This phase consists of developing the technical designs. Technical design diagrams will be developed in this phase along with carrying out design inspections.

#### **5.1.4. Build by Feature**

After a successful design inspection, the code for the feature will be written. Once completed, the feature will go through unit testing and be integrated into the main system.

## 6. References

Abler, S. W., 2021. *Feature Driven Development (FDD) and Agile Modeling*. [Online]

Available at: <http://www.agilemodeling.com/essays/fdd.htm>  
[Accessed 19 November 2021].

Lewis, S., 2019. *Prototyping Model*. [Online]  
Available at: <https://searchcio.techtarget.com/definition/Prototyping-Model#:~:text=The%20prototyping%20model%20is%20a,or%20product%20can%20be%20developed.>

[Accessed 21 November 2021].

Lucid Content Team, 2021. *Why (and How) You Should Use Feature-Driven Development*. [Online]

Available at: <https://www.lucidchart.com/blog/why-use-feature-driven-development>

[Accessed 21 November 2021].

Lumitex, 2017. *Prototyping Methodology: Steps on How to Use It Correctly*. [Online]

Available at: [lumitex.com/blog/prototyping-methodology](http://lumitex.com/blog/prototyping-methodology)

[Accessed 21 November 2021].

Martin, M., 2021. *Prototyping Model in Software Engineering: Methodology, Process, Approach*. [Online]

Available at: <https://www.guru99.com/software-engineering-prototyping-model.html>

[Accessed 21 November 2021].

PlanView, 2021. *What is FDD in Agile*. [Online]

Available at: <https://www.planview.com/resources/articles/fdd-agile/>

[Accessed 14 November 2021].



Prepinsta, 2020. *Iterative Waterfall Model in SDLC*. [Online]  
Available at: <https://prepinsta.com/software-engineering/iterative-waterfall-model/>  
[Accessed 20 November 2021].

Project Manager, 2021. *What Is the Waterfall Methodology in Project Management?*. [Online]  
Available at: <https://www.projectmanager.com/waterfall-methodology>  
[Accessed 20 November 2021].

Ruparelia, N., 2010. *Waterfall model with Royces iterative feedback When referring to the waterfall model*. [Online]  
Available at: [https://www.researchgate.net/figure/Waterfall-model-with-Royces-iterative-feedback-When-referring-to-the-waterfall-model-in\\_fig1\\_220631422](https://www.researchgate.net/figure/Waterfall-model-with-Royces-iterative-feedback-When-referring-to-the-waterfall-model-in_fig1_220631422)  
[Accessed 21 November 2021].

## 7. Appendix

### 7.1. Appendix A (Development)

#### 7.1.1. Gantt Chart

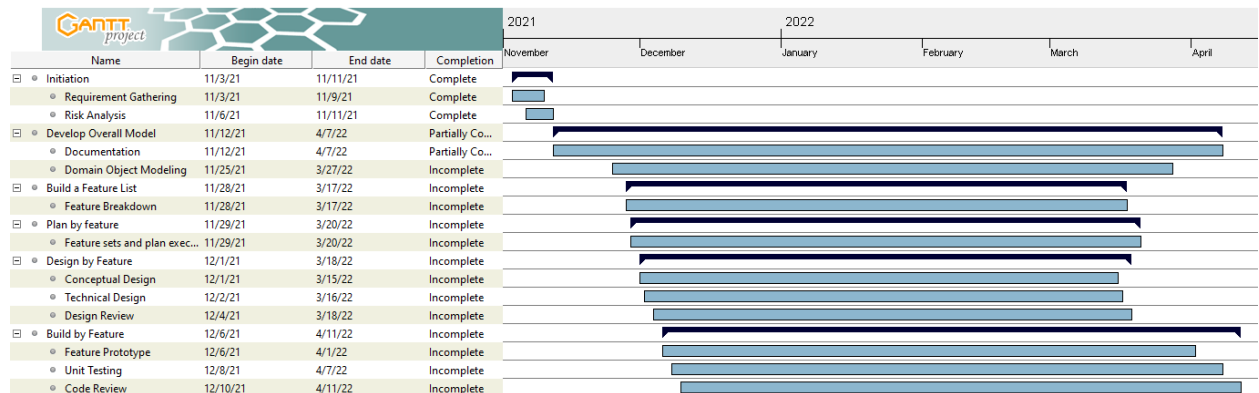


Figure 19: Updated Gantt Chart

GANTT project			
Name	Begin date	End date	Completion
Initiation	11/3/21	11/11/21	Complete
Requirement Gathering	11/3/21	11/9/21	Complete
Risk Analysis	11/6/21	11/11/21	Complete
Develop Overall Model	11/12/21	4/7/22	Partially Co...
Documentation	11/12/21	4/7/22	Partially Co...
Domain Object Modeling	11/25/21	3/27/22	Incomplete
Build a Feature List	11/28/21	3/17/22	Incomplete
Feature Breakdown	11/28/21	3/17/22	Incomplete
Plan by feature	11/29/21	3/20/22	Incomplete
Feature sets and plan exec...	11/29/21	3/20/22	Incomplete
Design by Feature	12/1/21	3/18/22	Incomplete
Conceptual Design	12/1/21	3/15/22	Incomplete
Technical Design	12/2/21	3/16/22	Incomplete
Design Review	12/4/21	3/18/22	Incomplete
Build by Feature	12/6/21	4/11/22	Incomplete
Feature Prototype	12/6/21	4/1/22	Incomplete
Unit Testing	12/8/21	4/7/22	Incomplete
Code Review	12/10/21	4/11/22	Incomplete

Figure 20: Gantt chart Tasks for Clarity

### 7.1.2. Work Breakdown Structure

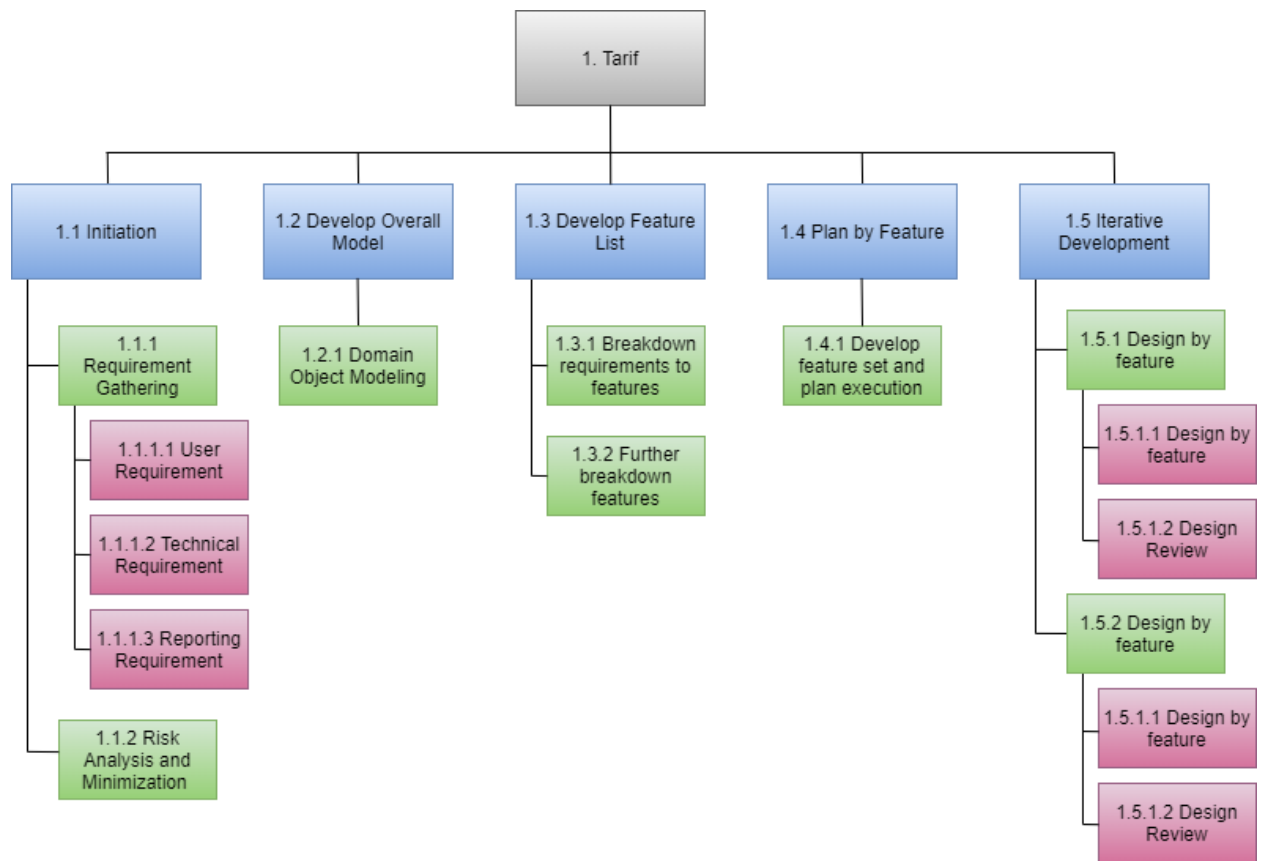


Figure 21: Work Breakdown Structure

## 7.2. Appendix B (Development Till Date)

### 7.2.1. Code Snippets

```
App.js > ...
1  import { ReactComponent as Logo } from "../assets/logo.svg";
2  import Form from "../components/Form";
3
4  function App() {
5    return (
6      <div className="App">
7        <header className="App-header">
8          <Logo />
9          <Form />
10         </header>
11       </div>
12     );
13   }
14
15   export default App;
16
```

Figure 22: App.js Code

```
# App.css > .App
1  .App {
2    text-align: center;
3  }
4
5  .App-logo {
6    height: 40vmin;
7    pointer-events: none;
8  }
9
10 @media (prefers-reduced-motion: no-preference) {
11   .App-logo {
12     animation: App-logo-spin infinite 20s linear;
13   }
14 }
15
16 .App-header {
17   background-color: #2c55a7;
18   min-height: 100vh;
19   display: flex;
20   flex-direction: column;
21   align-items: center;
22   justify-content: center;
23   font-size: calc(10px + 2vmin);
24   color: white;
25 }
26
27 .App-link {
28   color: #61dafb;
29 }
```

Figure 23: App.css Code

```
import React from "react";

const Button = ({ label, onclick, size = "default", variant = "primary" }) => {
  return (
    <button className={`g-btn ${size} ${variant}`} onClick={onclick}>
      {label}
    </button>
  );
};

export default Button;
```

Figure 24: Button Component Code

```
1  import React from "react"
2  import { useState } from "react"
3  import Button from "../Button"
4  import Google from "../../assets/GoogleButton.svg"
5  import Facebook from "../../assets/FacebookButton.svg"
6  import {Login as loginapi} from "../api"
7
8
9  const Form = () => {
10
11    const [formDetails,setFormDetails] = useState({username: "", password: ""})
12
13    const handleChange = (e) => {
14      const value = e.target.value;
15      setFormDetails({...formDetails, [e.target.name]: value});
16    }
17
18
19
20    return (
21      <form className="login">
22        <p>Enter your credentials and Log in to your dashboard</p>
23
24        <input name="username" type="text" placeholder="Username" className="textField" onChange={handleChange} />
25        <input name="password" type="password" placeholder="Password" className="textField" onChange={handleChange} />
26
27        <div style={{display: "flex",marginTop: "5px"}}>
28          <input type="checkbox" className="checkBox" name="Remember me" />
29          <label className="checkBox label">Remember me</label>
30        </div>
31      </form>
32    )
33  }
```

Figure 25: Login form Code

```
1 .g-btn {  
2   cursor: pointer;  
3  
4   font-family: "Lato";  
5   font-style: normal;  
6   font-size: 18px;  
7   font-weight: 500;  
8  
9   border: none;  
0   border-radius: 10px;  
1  
2   &.default {  
3     @include dimension(42px, 180px);  
4   }  
5  
6   &.small {  
7     @include dimension(42px, 120px);  
8   }  
9  
0   &.medium {  
1     @include dimension(42px, 270px);  
2   }  
3  
4   &.large {  
5     @include dimension(42px, 320px);  
6   }  
7  
8   &.primary {  
9     @include button-type("primary");  
0   }  
1   &.secondary {  
2     @include button-type("secondary");  
3   }
```

Figure 26: Button style Code

```
.login {
  font-family: "Lato", sans-serif;
  width: 320px;
  font-size: 16px;
  font-weight: 500;
  line-height: 19px;
  text-align: left;

  p {
    margin: 18px 0;
  }

  $TFpadLeft: 13.58px;

  .textField {
    height: 42px;
    width: calc(320px - #{$TFpadLeft});
    border-radius: 10px;
    padding-left: $TFpadLeft;
    margin: 8px 0px;
    border: 1.5px solid #dadada;
  }

  ::-webkit-input-placeholder {
    color: #dadada;
  }

  .checkBox {
    height: 20px;
    width: 20px;
    border-radius: 10px;
    border: none;
  }
}
```

Figure 27: Login Page Code



### 7.2.2. Viewable Development

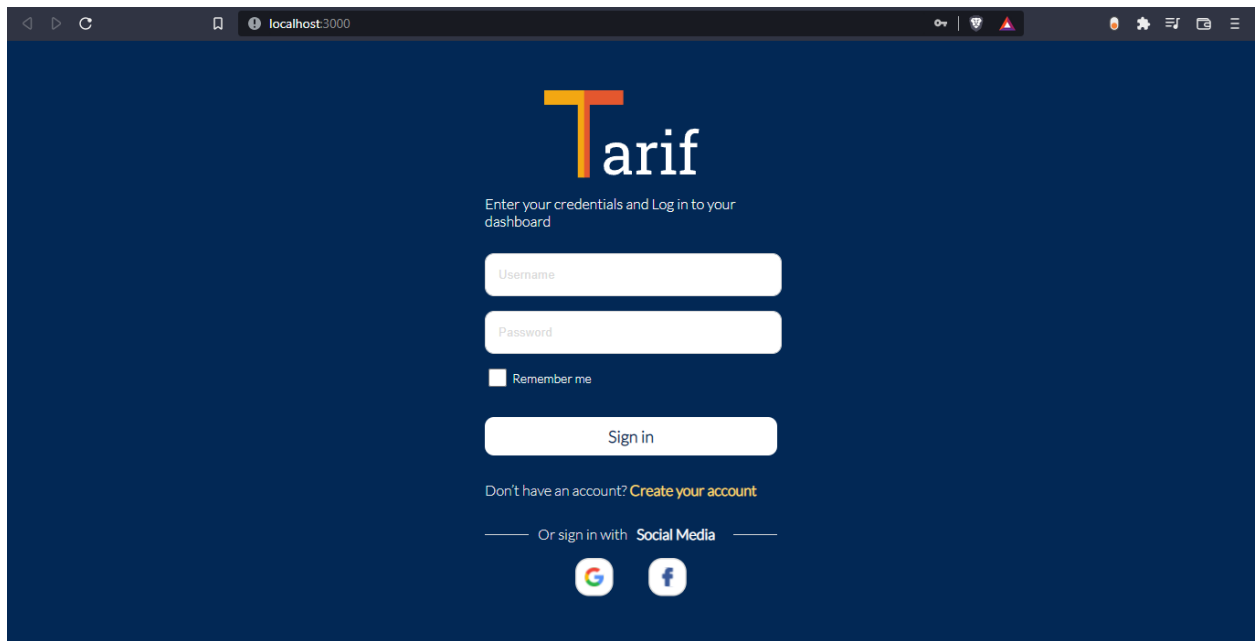


Figure 28: Sign-in page