Table of Contents

1. Introduction

Webapps have revolutionized the way we use the internet. Initially web browsers could only render simple HTML contents. However, nowadays browsers are powerful enough to run 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. Several businesses in Nepal have already moved or have been moving to the internet for that reason. The beauty of webapp 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 webapp, both the tenants and landlords can find one another more easily and communicate before having an on-site meeting. This helps both the tenants and landlords in screening the available options.

The pandemic made people more aware and enthusiastic for 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 benefit. Comparing the rents paid for different houses can give one an idea on 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 landlords to gauge what people are willing to pay and what their property's worth should be to tenants. By keeping record of rents paid/collected, the users can make educated guesses about what properties are worth and what amount they should be paying.

This is where Rent Management WebApp (Tarif) comes to play. The webapp makes it easier to find rentable houses/flats and tenants by displaying all the rents available in the user's vicinity in google map. It provides multiple options to both tenants and landlords along with reviews and ratings to help in screening from among the options. The webapp helps the landlords to push bills to tenants. It does so by allowing the landlords to make a bill template by selecting from available billing criteria such as monthly rental, water fees, electricity fees, waste disposal service fees, etc. which can then be used to push bills to the tenants. The webapp helps not only the landlords, but also helps the tenants to pay it via cashless methods if they so desire. The webapp also visualizes the rental data of landlord or tenant to them through graph so that they can make educated decisions on their rental journey.

a. Problem Scenario

The problems that the webapp 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 landlord visits the tenant door to door.

iii. Pushing consistent bills to tenants takes quite a bit of time as landlords 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 landlords and tenants. Thus, after a contract is signed, either party could end up stuck with a bad or unwanted company.

b. Project as solution:

The project solves the aforementioned issues by the following means:

i. The app displays a google map-based view which helps users locate houses ready to be rented. Similarly, since the houses are now viewable in map, the landlords 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 development of a bill template which the landlords 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 graph so that the user can make educated decisions.
v. The app supports ratings and reviews which help in screening both tenants and landlords.

2. Aims and Objectives

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

The objectives of this webapp 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 understand and customize google map component
- To better understand frameworks like react and Django

3. Expected Outcomes and Deliverables

The expected outcomes and deliverables of the webapp are:

- Google Maps for navigation
    - Points out house available for rent
- Make and push bills as landlord
- Receive bills and Make payments as tenant
- Leave ratings on either party
- Keep track of payment amounts and dates
- Communication portal for connecting landlords to tenants
- Data visualization in graph for both landlords and tenants

4. Risks and Contingency Plan

Some risks of the project are:

S.N.	Risks	Impact	Probability	Contingency Plan
1	System Failure	1	4	Have a reliable backup of the project so that no progress-loss occurs.
2	Client requirement variance	3	2	Define the scope and reach an agreement with the client before project initiation.
3	Difficulty in gaining client approval after project completion	1	3	Agree on proper Terms and Conditions with the client before project initiation.
4	Natural Disaster.	2	5	Have a reliable cloud backup of the project.
5	Technological Complication	3	4	Use widely used technology with significant forum support.
6	Insufficient resources	3	4	Carry out proper project planning and allocate sufficient resources.
7	Insufficient technical skills	3	4	Constantly research on the technologies used via documentations and developer forums.
8	Security Risks	1	2	Use secure frameworks and follow proper security guidelines throughout the project.
9	Impracticable deadline	2	2	Define the project scope and follow the selected methodology properly.
10	Unexpected time sink while bug fixing	3	2	Use proper coding practices and consult developer forums.

**Note: The impact and probability are determined on a scale of 1-5 where lower the value, higher the impact/probability. **

5. Methodology

a. Considered Methodologies

i. 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 preexisting project architecture can result in changes being made to every unit that has already been completed.

ii. 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.

iii. 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 project milestones, FDD uses features to organize their activities. The first three steps are completed initially and the rest of the steps are repeated for every feature.

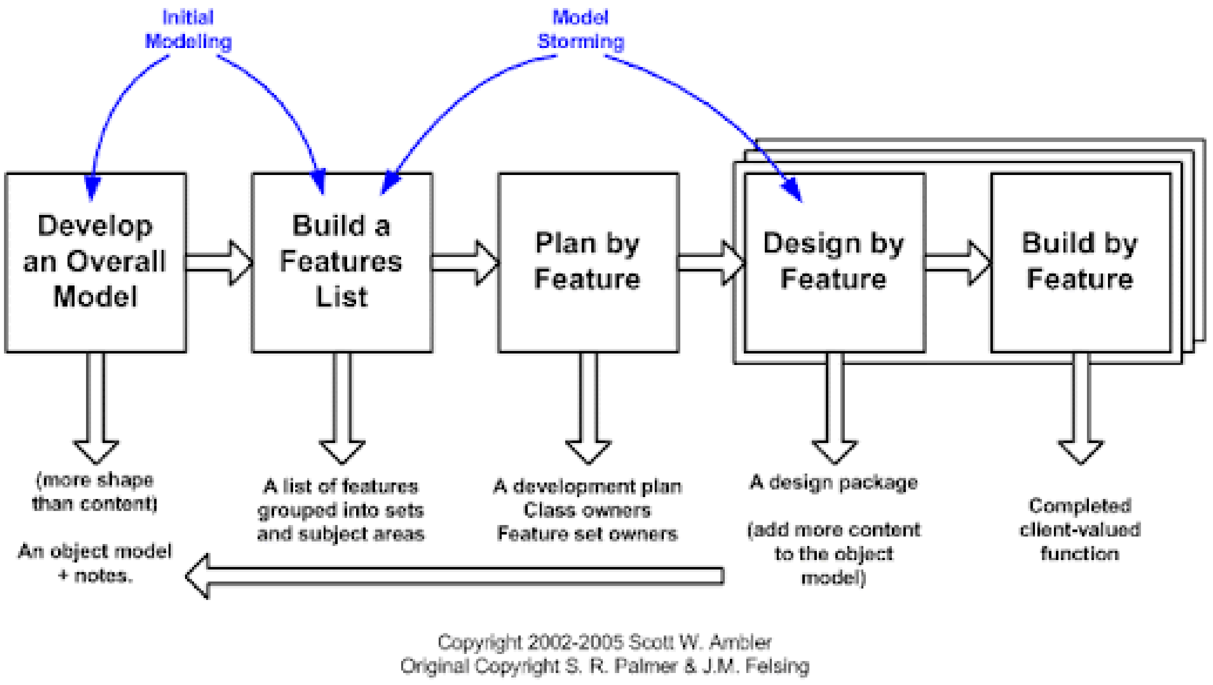

Initial Modeling
Model Storming
Develop an Overall Model
Build a Features List
Plan by Feature
Design by Feature
Build by Feature
(more shape than content)
An object model + notes.
A list of features grouped into sets and subject areas
A development plan Class owners Feature set owners
A design package
(add more content to the object model)
Completed client-valued function
Copyright 2002-2005 Scott W. Ambler
Original Copyright S. R. Palmer & J.M. Felsing

Figure 1: FDD project Lifecycle (Abler, 2021)

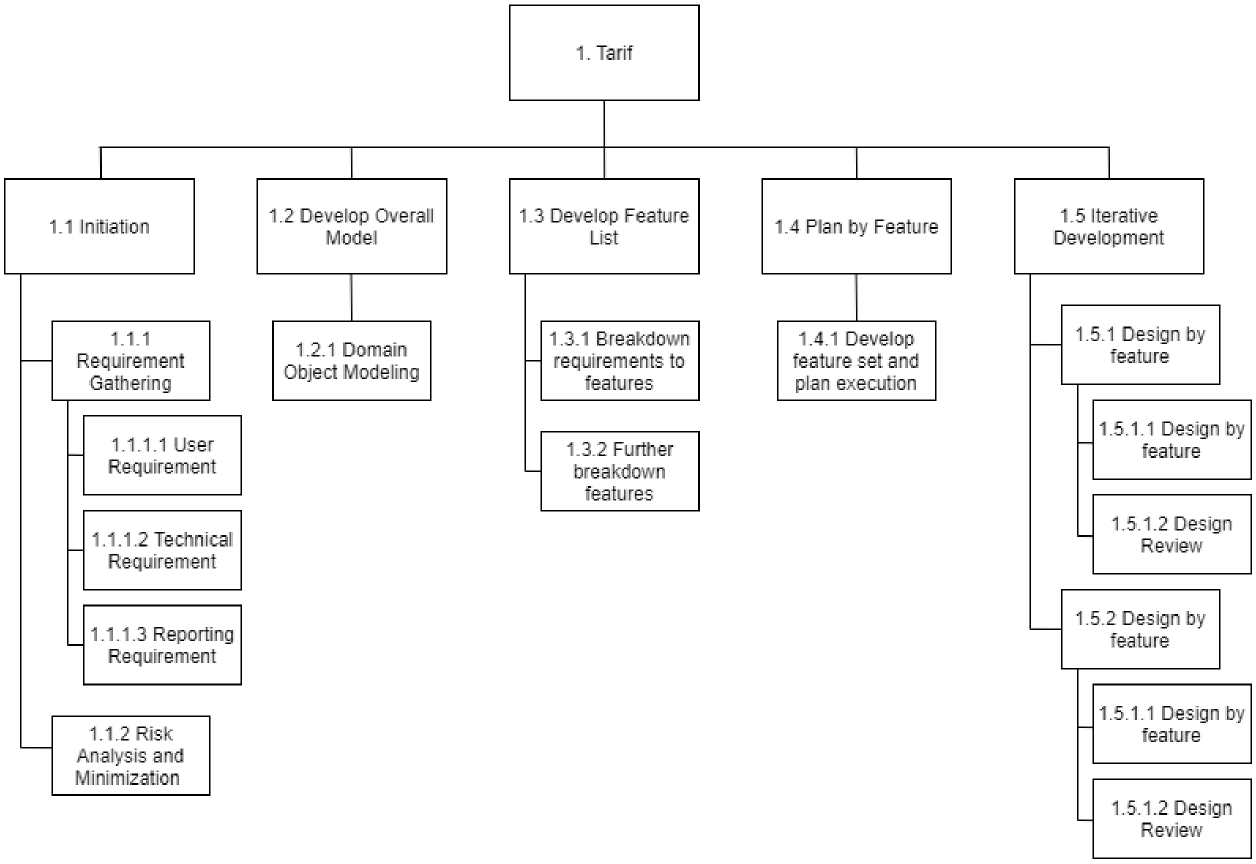
6. Resource and Requirement

Some requirements of the project are:

1. A working Internet connection
2. A working computer capable of running a web browser, IDE, and other tools.
3. An IDE: "VS Code"
4. Balsamiq Mockups for wireframing
5. Figma for UI design
6. Gantt Project for generating Gantt chart
7. Django for backend development along with SQLite
8. React JS for frontend development
9. Git and GitHub for version control

7. Work Breakdown Structure

Graphical Representation:


1. Tarif
1.1 Initiation
1.2 Develop Overall Model
1.3 Develop Feature List
1.4 Plan by Feature
1.5 Iterative Development
1.1.1 Requirement Gathering
1.1.1.1 User Requirement
1.1.1.2 Technical Requirement
1.1.1.3 Reporting Requirement
1.1.2 Risk Analysis and Minimization
1.2.1 Domain Object Modeling
1.3.1 Breakdown requirements to features
1.3.2 Further breakdown features
1.4.1 Develop feature set and plan execution
1.5.1 Design by feature
1.5.1.1 Design by feature
1.5.1.2 Design Review
1.5.2 Design by feature
1.5.1.1 Design by feature
1.5.1.2 Design Review

Textual Representation:

1. Tarif
    - 1.1 Initiation
        - 1.1.1 Requirement Gathering
            - 1.1.1.1 User Requirement
            - 1.1.1.2 Technical Requirement
            - 1.1.1.3 Reporting Requirement
        - 1.1.2 Risk Analysis/Minimization
    - 1.2 Develop Overall Project Model
        - 1.2.1 Domain Object Modelling
    - 1.3 Develop Feature List
        - 1.3.1 Breakdown requirements into features

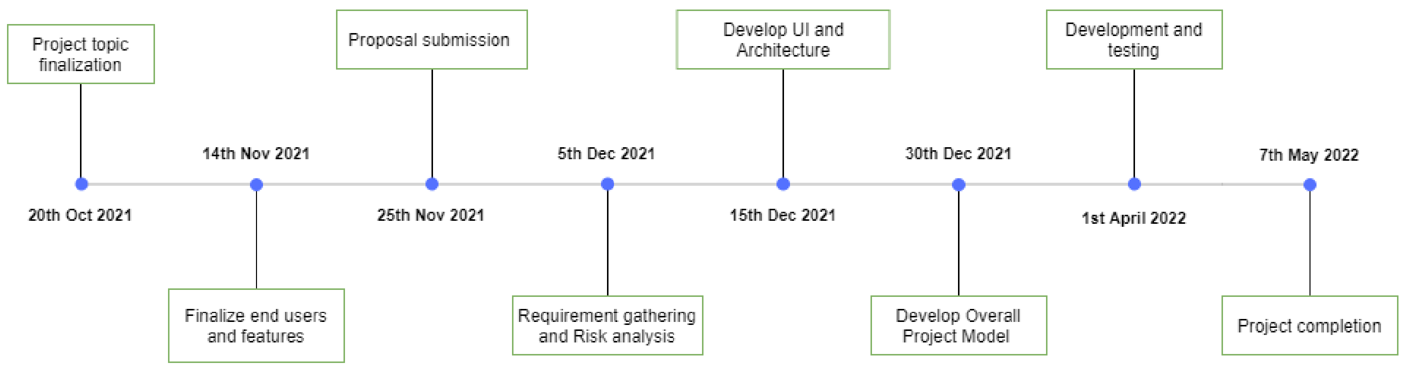
- 1.3.2 Gauge feature development feasibility and further breakdown features

1.4 Plan by Feature
- 1.4.1 Group Features into feature sets and plan execution

1.5 Iterative development
- 1.5.1 Design by Feature
  - 1.5.1.1 Design a feature
  - 1.5.1.2 Design Review
- 1.5.2 Build by Feature
  - 1.5.2.1 Code the feature
  - 1.5.2.2 Code review

8. Milestones

Visual Representation:



Textual Representation:

Project topic finalization - 20th Oct 2021

Finalize end users and features - 14th Nov 2021

Proposal submission - 25th Nov 2021

Requirement gathering and Risk analysis - 5th Dec 2021

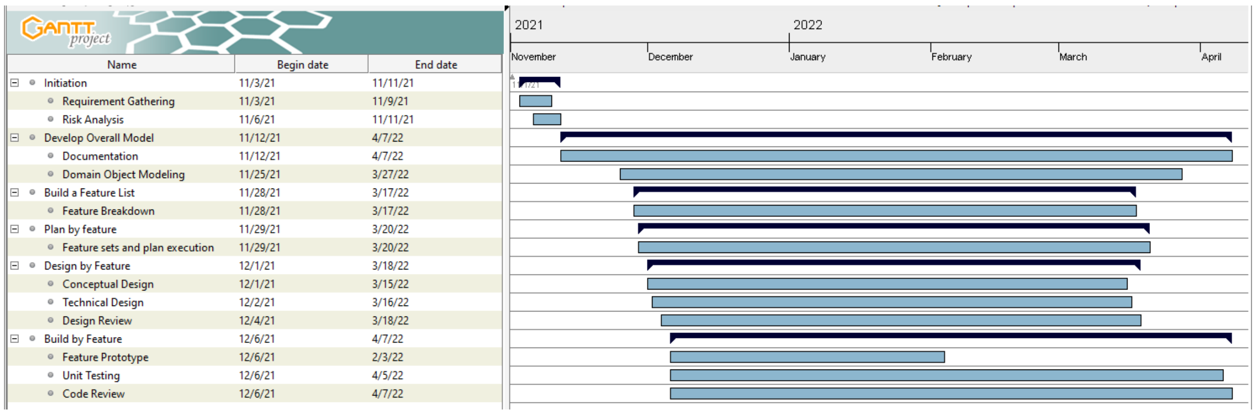
Develop UI and Architecture - 15th Dec 2021

Develop Overall Project Model - 30th Dec 2021

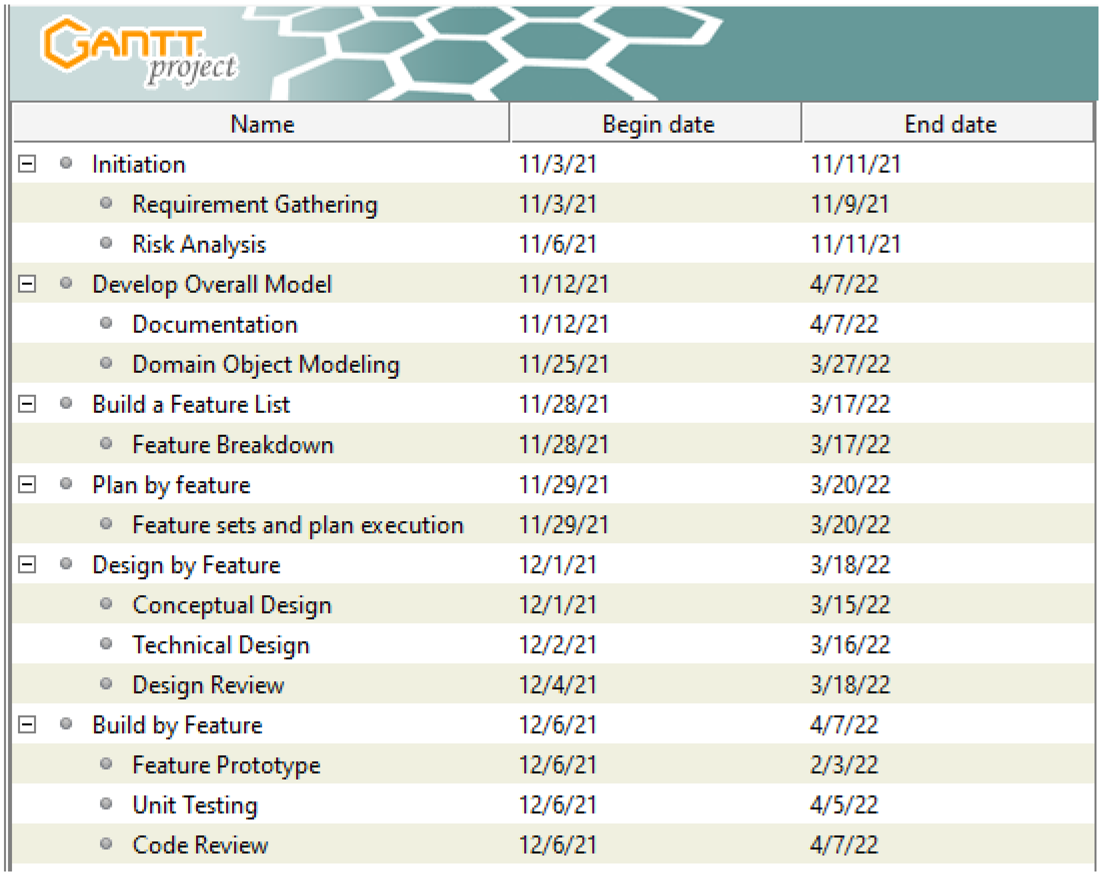
Development and testing - 1st April 2022

Project completion - 7th May 2022

9. Gantt Chart



Name	Begin date	End date
Initiation	11/3/21	11/11/21
Requirement Gathering	11/3/21	11/9/21
Risk Analysis	11/6/21	11/11/21
Develop Overall Model	11/12/21	4/7/22
Documentation	11/12/21	4/7/22
Domain Object Modeling	11/25/21	3/27/22
Build a Feature List	11/28/21	3/17/22
Feature Breakdown	11/28/21	3/17/22
Plan by feature	11/29/21	3/20/22
Feature sets and plan execution	11/29/21	3/20/22
Design by Feature	12/1/21	3/18/22
Conceptual Design	12/1/21	3/15/22
Technical Design	12/2/21	3/16/22
Design Review	12/4/21	3/18/22
Build by Feature	12/6/21	4/7/22
Feature Prototype	12/6/21	2/3/22
Unit Testing	12/6/21	4/5/22
Code Review	12/6/21	4/7/22



Name	Begin date	End date
Initiation	11/3/21	11/11/21
Requirement Gathering	11/3/21	11/9/21
Risk Analysis	11/6/21	11/11/21
Develop Overall Model	11/12/21	4/7/22
Documentation	11/12/21	4/7/22
Domain Object Modeling	11/25/21	3/27/22
Build a Feature List	11/28/21	3/17/22
Feature Breakdown	11/28/21	3/17/22
Plan by feature	11/29/21	3/20/22
Feature sets and plan execution	11/29/21	3/20/22
Design by Feature	12/1/21	3/18/22
Conceptual Design	12/1/21	3/15/22
Technical Design	12/2/21	3/16/22
Design Review	12/4/21	3/18/22
Build by Feature	12/6/21	4/7/22
Feature Prototype	12/6/21	2/3/22
Unit Testing	12/6/21	4/5/22
Code Review	12/6/21	4/7/22