Rental Portal Software System

Final Proposal

Prepared By:

Anirudh Venkatesh

Submitted to:

CPSC 589: Computer Science Seminar

Date: 12th May 2024

Professor: Dr. Kenneth Kung

Department of Computer Science

College of Engineering and Computer Science

California State University Fullerton

CALIFORNIA STATE UNIVERSITY FULLERTON™

Department of Computer Science

CPSC 597 / 598 PROJECT / THESIS DEFINITION

To the graduate student:
1. Complete a project proposal, following the department guidelines.
2. Have this form signed by your advisor and reviewer / committee.
3. Submit it with the proposal attached, to the Department of Computer Science.

- [x] Project
- [ ] Thesis

Please print or type.

Student Name: ANIRUDH VENKATESH Student ID: 885174318

Address: 2452 Nutwood Ave Apt D35 (Street) Fullerton (City) 92831 (Zip Code)

Home Phone: (657)525-8035 Work Phone: N/A

E-Mail: anirudh.venkatesh@csu.fullerton.edu Units: 3 Semester: Spring 2024

Are you a Classified graduate student?
- [x] Yes
- [ ] No

Is this a group project?
- [ ] Yes
- [x] No

Proposal Date: 12th May 2024

Tentative Date for Demonstration /Presentation/Oral Defense: _____

Completion Deadline: _____

Tentative Title: Rental Portal Software System

We recommend that this proposal be approved:

	Printed name	Signature	Date
Faculty Advisor	Kenneth Kung	Kenneth Kung	May 15, 2024
Faculty Reviewer			
Faculty Reviewer			

Table of Contents

Table of Figures

List of Tables

Introduction

Real Estate prices are always on the rise. Everyone believes in investing in real estate as it is considered worthwhile, something whose value will increase many folds in the near future. There are countries that provide Citizenship on Investment policies that attract potential investors to invest in their country and gain citizenship in return for their investment. One can also see the property values of various buildings and houses in densely populated cities, and their value rises with time. Property values have increased so much that buying property in the world's major cities is almost impossible. Despite this, people still believe in investing in property because of its immense potential.

There's always a market for real estate, and connecting sellers with potential buyers has always been a need in the property market. While many platforms serve that purpose, there's always a requirement for an application that comprises the latest technology that is better able to serve the needs of the people.

In recent years, there has been a trend in how Software is delivered to customers. Traditional software has had issues with Development costs, maintenance, and upgrade costs, and that typically leaves customers unhappy with the software product. The rise of Software as a Service (SaaS) has changed the way it is developed and delivered to customers.

Background

The rental market has long been a cornerstone of the housing market, providing individuals and families with affordable housing options and landlords with income streams. However, navigating the rental or purchase process, whether from the landlord's or buyer's perspective, often presents numerous challenges and inefficiencies.

Recognizing the need for modernization and streamlining in the rental industry, this project aims to develop a Rental Portal Software System that encompasses the latest trends in technology in a modern software development environment. This innovative software solution is envisioned to revolutionize how buyers or tenants search and connect with landlords.

Traditional software development processes have their shortcomings. Software as a Service (SAAS) is a software development model that delivers software solutions to businesses and consumers. In the modern world, we use software such as Microsoft Office, a classic example of SAAS. It is a commercial product where several businesses and individuals purchase the same software for different purposes. The property portal is envisioned to be developed similarly.

In recent years, how we work with data has revolutionized software development. Handling data, susceptible data such as User Data, Images, and House addresses, must be securely worked on and stored, which has changed how we develop software. Older software typically struggled to cope with large volumes of data and could not handle large volumes of data.

Modern Software Techniques have been found to develop software efficiently. As mentioned, maintaining and updating software is also crucial to keeping happy customers. There have been changes in software design that allow integration with third-party companies to provide a good experience at the front end. The Layered Architecture is a software architecture that follows the

design principle of separation of concerns. Each software layer is only concerned with what it is supposed to do and only calls upon the service of the layer below it. Each layer does not know how other layers work and treats every lower layer as an interface, simply giving an input with specific parameters and receiving an output.

Current Approaches

There are currently several exciting approaches to developing a rental portal software system. They are:

Smart Rental Portal using Augmented Reality

The paper from Satapathy et al. (2020) shows a rental portal developed using a recommendation system and augmented reality. The paper proposes a method to use collaborative filtering and shallow filtering to display properties of interest to users.

The paper uses MongoDB to handle property data. The server is implemented using Node JS and a software interface using HTML, CSS, and Javascript. There's no layered architecture, and the tool is developed as a web app. The tool works as follows: Users enter the portal's home page and type their requirements, and the system responds with houses that meet the user's requirements. Users can explore each home via augmented reality.

Smart house renting web application

The Voumick et al. (2021) paper shows another version of a property portal with a user interface designed in CSS, Javascript, and AJAX, using MySQL Database Service to store user data and LARAVEL to develop secure features.

Interestingly, their integration of Google Maps with their application allows users to see the house's location over a map. The authors identify different kinds of users that will be associated with the system and have designed the system accordingly. There is no search functionality implemented. A prospective tenant logs into the system, and the system displays all the available houses. The tenant can choose a home of their liking and drop a message to the owner directly.

House Rental Platform Using Blockchain Technology

The paper from Kang et al. (2020, pp. 300-301) shows another interesting approach for a house rental platform using Blockchain Technology. The paper shows the use of the Ethereum platform. Users need a MetaMask account to access the Ethereum wallet. Smart contracts are validated and opened at Etherscan. Bootstrap is used to develop the house rental platform. Web3.js connects the website to the smart contract to display information on the smart contract. The platform also allows landlords to upload pictures of the house on the smart contract.

Project Objective

To develop an end-to-end application that provides users and businesses with a modern experience, with the ability to connect, transact, and manage their property needs.

This is a:

- A software product that is designed to be commercially viable.
- A software product to compete with existing design implementations.

The aim is to develop an end-to-end Rental Portal Software System using a layered software architecture encapsulating the latest technologies, such as Natural Language Processing (NLP) techniques at the front end and the capability of storing large amounts of data using big data technologies.

Designing the software as a SAAS aims for a pleasant user experience and a worthy competitor to existing software solutions.

Key Challenges

- **Large Data:** A typical rental portal software system has to deal with user information (which includes their names, ages, and email addresses) and also homes (their address, dimensions, and amenities). So, efficient storing of such data is essential.
- **Security Concerns:** Ensure that renters' and landlords' information is secure.
- **User Experience:** An excellent user experience is vital for any software product. Customers get to interact with landlords easily. This also allows customers to refine their search.
- **Unresponsive landlords or property managers:** Difficulty contacting someone to answer questions or schedule viewings can be a significant pain point for renters.
- **Ensuring fair and ethical practices:** The platform must ensure that all users are treated fairly and that listings are accurate and non-discriminatory.

This is an exciting graduate-level project because:

- Integration of new technologies such as Natural Language Processing Techniques – something OpenAI uses. Handling large amounts of data requires a good system design.
- Providing a competitive solution to a demanding market. Demand for property is always on the rise.
- An experience in developing end-to-end commercial software.
- Great Future Potential, such as involving Augmented Reality or Virtual Tours in Houses, which are trending concepts in computer science.

Proposed Approach

While existing approaches promise to give an excellent solution to the problem they set out to tackle, there’s always an opportunity to provide better solutions to existing solutions. The following subsections will discuss how this project will be designed.

Layered Software Architecture

A Layered Software Architecture is a design pattern used in software development to design applications. The system is organized into separate layers, each responsible for a specific set of tasks. Each layer is designed to work independently from the other layers.

A Layered Architecture is typically structured as follows:

1. User Interface (UI) Layer: This layer is responsible for interacting with users and presenting and collecting their information. The type of User Interface may vary; it can be a Graphical User Interface (GUI) or a Command Line Interface (CLI); however, everything is coded to the interface, so the system is designed to work with multiple interfaces.

   In the context of this project, the idea is to develop the front end as a Web Interface with Natural Language Processing (NLP) techniques that interpret user input as a chat and convey the information to the lower layers. It is like a bot used in modern applications.

2. Core Business Logic Layer: This layer handles the core functionality and rules of the application. It implements the algorithms, business processes, and data handling required by the system.

   In the context of this project, the core business logic layer will comprise different handlers. Requests from the UI Layer (the upper layer) will delegate the requests to their respective handlers. For example, an inquiry for a property will arrive at the Session Handler and then be transferred to the Inquiry Handler to handle the request.

3. Technical Services Layer: Also known as the Data Access Layer, this layer handles the interaction with various other systems, such as databases, files, or payment services. It encapsulates the details of data retrieval, storage, and manipulation, providing a consistent interface for the rest of the application to access and modify data.

   Like the Core Business Logic Later, the Technical Services Layer is also envisioned to contain different handlers that cater to requests from the Business Logic Layer. Examples include a Payment Handler and a Database Handler.

   Note: Only the Database Handler will be implemented as part of this project. Other components will be stubbed as part of this project.

Advantages of Layered Architecture:

A paper from Tu (2023, p.35) mentions the advantages of implementing a software design in a layered architecture. A layered architecture works like an MVC (Model-view-controller) architecture with separation between each layer. Each layer can focus on specific responsibilities, allowing other areas of the software to be implemented separately (or outsourced.) This

architecture makes the code easy to understand, reuse, and test. This also reduces efforts in redundant code and improves overall efficiency.

The following figure depicts how a typical layered architecture would look like in the project:

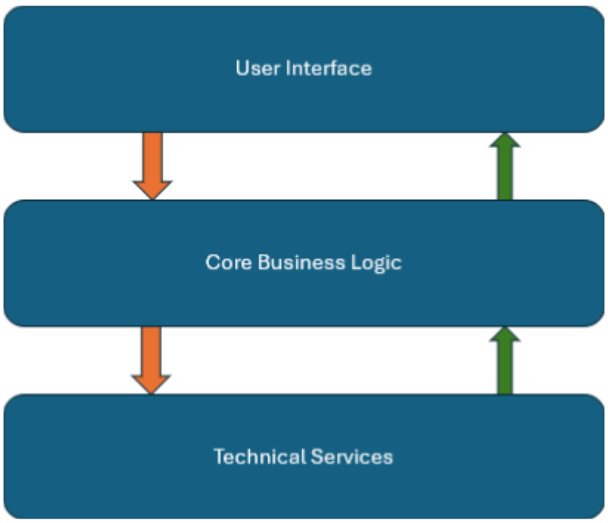

Figure 1: Layered Architecture used in the project.

Design as a Software as a Service (SaaS)

Traditional software requires end-users to purchase a license before running it on their machines. A paper from Waters (2005, pp. 34-35) mentions some issues in traditional software models:

- Unexpected costs: A customer might be under the impression that purchasing a license will suffice. However, when purchasing software, the end-user is unaware of the other hidden costs. Some hidden costs are:
    1. Training
    2. Downtime
    3. Server Hardware
    4. Installation and Configuration
- Implementation Delays: The paper mentions that over 85% of software projects are delivered well behind schedule and over budget. Some issues that contribute to this are:
    1. Waiting for approvals
    2. Scope-Creep
- Administrative burdens: The administrative challenges that an organization faces today mainly in:
    1. Managing a heterogeneous environment
    2. Capacity building and utilization
    3. The snowball effect of upgrade costs

The paper from Waters (2005, pp. 36-38) mentions the benefits of implementing SAAS. They are:

- Total Cost of Ownership: Costs are known to customers well in advance. Since customers can easily manage their applications, ownership costs are much lower.
- Speed of Deployment: Customers have no software to deploy and configure; SAAS implementation can instantly be activated for a client.
- Security: Since the developing company provides it, customers can be assured that it is of the highest quality.
- Easy upgrades: Upgrades are performed from the developing company's end, so customers do not have to install the latest software version. This ensures that customers are always using the newest version of the software.

Container will be replaced by Virtual Machine

In the context of this project, the plan is to develop it as a container to deploy applications as SaaS.

Since this project deals with a specific industry, i.e., the housing industry, the project will be designed using vertical SaaS architecture. Some of the benefits of a Vertical SaaS Architecture include:

1. Industry Specific Features
2. Faster Implementation
3. Enhanced User Experience
4. Scalability and Growth

Another design aspect of this project is using the Single-tenant architecture of SaaS. The Single-tenant architecture is a concept where each customer has their own instance of the application and database. This contrasts with Multi-tenant architecture, where customers share either the database or application. While Multi-tenant architecture has some unique benefits, it's more complicated to implement, which is why the project will be implemented as a Single-tenant architecture given the timeframe.

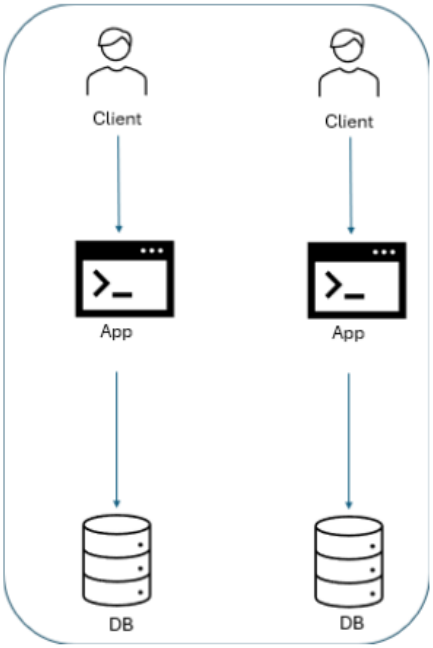

Figure 2: Single-Tenant Architecture used in the Project.

Incorporation of Big Data Techniques

With the advent of Big Data Technologies, machines can now handle and store large amounts of data. Data can be structured or semi-structured, and significant advancements have been made to efficiently handle various kinds of data.

The idea of the project is to incorporate Big Data Techniques to store House Images, Addresses, and Owner Information. This will be part of the Technical Services Layer of the Layered Architecture (as mentioned in an earlier section.)

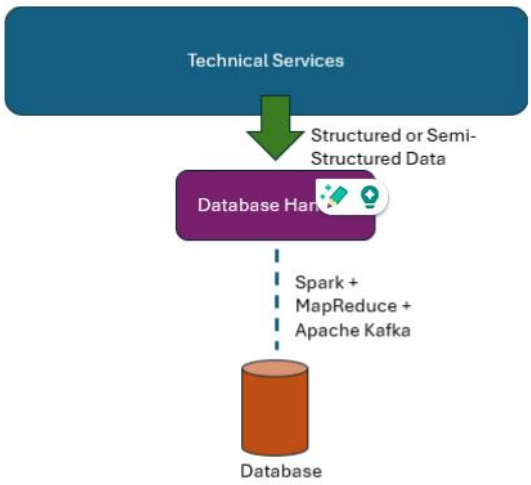

Figure 3: Overview of Database Handler using Big Data Techniques

The idea is to Implement basic NLP features like keyword search and simple text processing rather than complex question-answering systems.

Use of Natural Language Processing (NLP) Techniques

The Project is designed to incorporate Natural Language Processing (NLP) Techniques at the Core Business Logic Layer of the Architecture. If required, it might be implemented at the User Interface level. The critical concepts of Text Processing and Question Answering will mainly be implemented here as part of the Web Page. Additional functionalities, such as sentiment analysis, are not included in the design aspect of this project.

An Overview of the use of the NLP Technique is shown in the figure below.

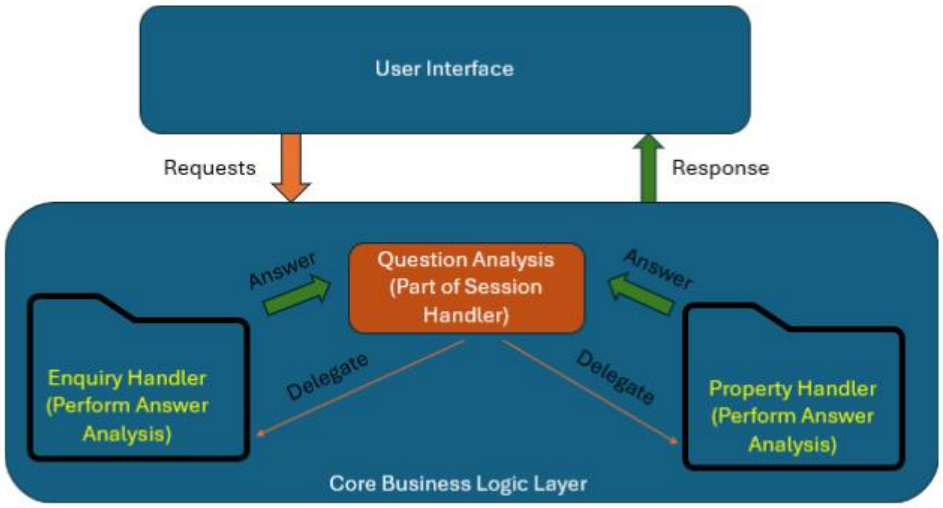

Figure 4: An Overview of the use of NLP.

Development Phases

Since the project is designed in a layered architecture, the following development phases will be followed:

Inception Phase:

The inception phase in software development establishes project goals, scope, stakeholders, and feasibility, laying the groundwork for planning and ensuring alignment with business objectives.

Elaboration Phase:

The elaboration phase expands upon the inception phase by refining requirements, defining the architecture, creating a detailed project plan, and setting the stage for development and implementation activities.

Construction Phase:

The construction phase focuses on building and coding the software according to the elaborated requirements and architecture, integrating components, and conducting thorough testing to ensure functionality and quality.

Testing Phase:

The testing phase involves systematically evaluating the software to identify defects, verify functionality, and validate against requirements, ensuring the system's reliability, usability, and adherence to quality standards.

The Project Schedule (described below) shows the activities performed in each phase in detail.

Overall Workflow

The Project is envisioned to work as follows:

1. The User enters a query on the User Interface.
2. The request reaches the Core Business Logic Layer.
    a. The Session Handler understands the request and delegates the request to the appropriate handler in that layer.
    b. The delegated handler handles the request and requests for data from the Technical Services Layer.
3. The Technical Services Layer receives the request from the Core Business Logic Layer.
    a. The Technical Services Layer performs the service required and returns the output to the respective handler in the Core Business Logic Layer.
4. The Handler in the Core Business Logic layer performs additional operations (if needed) and returns the final output to the Session Handler.
5. The Session Handler returns the response to the User Interface Layer.
6. The User Interface presents the information to the User.

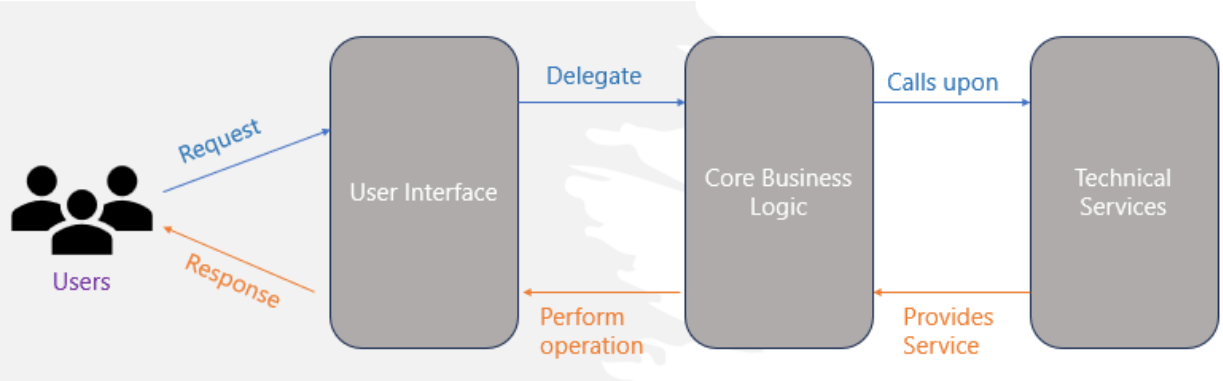

Figure 5: Overall Workflow of the Project.

Sample Use Case: Home Search

Scope: Rental Portal Software System

Level: User goal

Primary Actor: Guest

Stakeholders and Interests:

- Guest: Wants results that match their requirements.
- SaaS Company: Want user requests to be understood and responded to correctly.
- Purchasing Company: Want more prospective buyers to visit their site to search for homes and have a pleasant experience.

Success Guarantee: The system is able to provide recommendations that match the requirements.

Main Success Scenario:

1. The guest requests authentication, and the system responds with a text prompt.
2. The guests request to list the houses by entering their requirements, and the system responds with houses that closely match the guests' requirements.
3. The guests request to see more information about a chosen house, and the system responds with such details, including location and contact information.
4. The guest requests to schedule a house tour by selecting their preferred date and time, and they respond that their house tour is booked.
5. The guest signs off the system.

Project Outcomes

Core Functionalities such as property listing, searching and user registration will be in priority.

The Major outcomes of this project are:

- The ability of the system to understand and respond to user inputs correctly. The System must be able to perform text translations correctly so that correct handlers are identified in the Core Business Logic Layer. For example, Guest inquiries for houses in a specific location should be answered with houses available.
- Data is stored correctly and securely in the database. User Information is retrieved correctly. Also, the house information is correctly fetched for each request made.
- The Layered Architecture requires the principle of separation of concerns to be followed at each layer. This means that each layer does not implement functionality that is not a part of it. Each layer only calls upon the services of the layer below it. The Layered Architecture also requires everything to be coded to the interface, which means that Implementation must be separated from the Interface.
- Ensure that Vertical SaaS Architecture is followed in the project.
- Finally, A web user interface that can capture and display information to the user.

Certain aspects of the project will not be developed in the project. They are:

- Payment Gateways: Handling of Payments is not in the project's scope, and such functionality will be stubbed for this project.

Advanced NLP or Big Data Analytics will be for future works.

- User Authentication: Secure User Authentication and Authorization will not be a part of the project nor be tested. Basic User Functionality will be part of the project.
- Support for Multiple Languages: Implementing NLP to translate the English language will be developed and tested. Support for other languages is not part of the project.
- Any other external functionality integration and testing will not be performed.

Software Requirement Specifications

The Sample Use Case section earlier gives a brief idea of the kind of functionalities that will be required in this project. With that, this section will discuss the Functional and Non-Functional requirements of the Project.

Functional Requirements

1. User Registration: Site Users must have an account to list their properties. As mentioned earlier, basic user registration and authentication will be implemented as part of this project.
2. Property Listing: Allow property owners to list their properties for rent or sale, including details such as location, size, amenities, rental price, and availability dates.
3. Search and Filter: Users can type their query in a search box, and the system will respond with houses that match their request.
    a. Natural Language Processing techniques must be trained and tested to understand human input.
    b. Session Handler in the Core Business Logic Layer can understand and delegate requests to the correct handler.
4. Storage of Home Information: Unstructured and Semi-structured data need to be processed and stored correctly.
5. User Profile Management: Allow users to manage their profiles, including updating personal information, viewing listed properties, and editing properties available for sale or rent.
6. Integration with Mapping Services: Houses have addresses, and visitors may want to visit them. The idea is to Integrate the software with mapping services to provide location-based information.

Non-Functional Requirements

1. Performance
    - The software must handle many concurrent users without significant degradation in performance.
    - Response Times for search queries, property listings, and related activities should be minimal, typically within a few seconds.
    - Given that this project is implemented as a single-tenant architecture, users can expect high computing performance.

2. Reliability

    - The System should always be available with minimal downtimes to facilitate upgrades and maintenance.

3. Scalability

    - The Software is designed to accommodate growth in portal users over time.

4. Usability

    - The Graphical User Interface is designed to be user-friendly and intuitive.

    - The UI uses the best user experience design practices to ensure navigation efficiency.

5. Compatibility

    - The Software is designed to operate on Linux (and derivative) Operating Systems.

    - The User Interface is designed to be supported in popular web browsers like Microsoft Edge and Google Chrome.

6. Maintainability

    - The System is designed on a Layered Architecture, which requires the design to be based on interfaces. This allows flexibility in modifying implementations to certain areas of the software without affecting the other portions of the software.

    - At delivery time, detailed documentation will be provided to clients to help them understand the system architecture, configuration settings, and system requirements.

Project Environment

Development Environment

- Programming Languages: The System is envisioned to be developed in Python. Other Languages include NLTK for implementing NLP Techniques.

- Framework and Libraries: Python Django (for GUI).

Database Management: Applications such as PostgreSQL, MySQL or NoSQL (MongoDB) will be used for the project.

- Database Management: A combination of Apache Kafka, Spark, and MapReduce will be used to handle data using a NoSQL Approach.

- Developmental Tools: Tensorflow or Scikit-Learn. Version control systems such as Git will be used.

Deployment Environment

Deployment Environment will be on Virtual Machines.

- Hosting Environment: The Software will be hosted in Google gVisor or Amazon AWS.

- Containerization: Use of docker for deployment of Containers (like a docker container). Kubernetes will be used to manage deployed Containers.

Project Schedule

Month 1: Initial Project Setup and Planning	
Weeks 1-2: Project Kickoff and Planning (10hrs) (Inception Phase)	• Identify the goals, deliverables, and scope of the project. • Define project roles and responsibilities • Develop a project plan detailing the significant tasks, deliverables, and milestones.
Week 3: Requirements Gathering (20hrs) (Elaboration Phase -1)	• Collect training data for Natural Language Processing Models. • Document functional and non-functional requirements in detail after considering the availability and feasibility of software tools at the time of development.
Week 4: Architecture Design (25hrs) (Elaboration Phase -2)	• Create a blueprint of the architecture design considering the database schema, dependencies, and component layers. • Upskill on certain technologies if required.
Month 2: Development (Construction Phase -1)	
Weeks 1-2: Prototype Development (25hrs)	• Create a small prototype with minimal backend and frontend development.
Week 2-4: Backend Development (45hrs)	• Build on the prototype.

	• Provide full implementation to each layer with their handlers. • Develop a backend database and integrate it with Technical Services Layer.
Month 3: Working on Frontend (Construction Phase -2)	
Week 1-3: Frontend Development (25hrs)	• Develop the full functionalities of the GUI. • Develop NLP Techniques • Integrate with Core Business Logic Layer
Week 4: Deployment (25hrs) (Deployment Phase)	• Once the front and back end components are completed, deploy the beta version of the software.
Month 4: Integration and Testing (Testing Phase)	
Weeks 1: (25hrs) (Unit Testing)	• Perform Unit tests for backend and frontend components
Week 2: (30hrs) (Integration Testing)	• Integrate frontend and backend components. • Perform Integration Testing to validate functionality.
Week 3-4: (60hrs) (Bug Fixing)	• Identify bugs or failures happening. Sort bugs from Critical to Minor. • Fix the code to resolve bugs. • Perform necessary tests to confirm the bugs are resolved.
Month 5: Documentation	
Week 1-2: (20 hours)	• Create a Use Case Document. • Create a Risk List and Risk Management Document. • Create Software Architecture Document. • Create Domain Model Document. • Create a Design Model Document.

Table 1: Project Implementation Schedule

Total Estimated Duration: 310 Hrs.

Note: Different documentation will be merged into one deliverable at delivery time.

Project Deliverables

The following will be made available to the Project Advisor and the Computer Science Department, California State University Fullerton:

1. Access to the entire working code in a zipped file. The Code will include a readme file that provides an overview of the code.
2. Detailed Documentation. The documentation will comprise different sections as detailed in the project schedule.
3. A demo video of the project will also provided.

References

Waters, B. (2005). Software as a service: A look at the customer benefits. Journal of Digital Asset Management, 1(1), 32–39. doi:10.1057/palgrave.dam.3640007

Zheng, X., Fu, M., & Chugh, M. (2017). Big Data Storage and Management in SaaS Applications. Journal of Communications and Information Networks, 2(3), 18–29. doi:10.1007/s41650-017-0031-9

Tu, Z. (2023). Research on the application of layered architecture in computer software development. Journal of Computing and Electronic Information Management, 11(3), 34–38. doi:10.54097/jceim.v11i3.08

Lauriola, I., Lavelli, A., & Aiolli, F. (2022). An introduction to Deep Learning in Natural Language Processing: Models, techniques, and tools. Neurocomputing, 470, 443–456. doi:10.1016/j.neucom.2021.05.103

Satapathy, S. M., Jhaveri, R., Khanna, U., & Dwivedi, A. K. (2020). Smart Rent Portal using Recommendation System Visualized by Augmented Reality. Procedia Computer Science, 171, 197–206. doi:10.1016/j.procs.2020.04.021

Voumick, D., Deb, P., Sutradhar, S., & Khan, M. M. (2021). Development of online based smart house renting web application. Journal of Software Engineering and Applications, 14(07), 312–328. doi:10.4236/jsea.2021.147019

Kang, T.-C., Chang, C.-H., Chan, Y.-W., Tsai, Y.-T., & Liu, T.-E. (2020). An Implementation of House Rental Platform with Blockchain Technology. In J. C. Hung, N. Y. Yen, & J.-W. Chang (Eds.), Frontier Computing (pp. 298–301). Singapore: Springer Singapore.