Department of Computer Engineering

Senior Design Project
T2306
SecuRent

Design Project Final Report:

Team Members:

Mustafa Kaan KOÇ - 21903358 - kaan.koc@ug.bilkent.edu.tr
Yusuf ŞENYÜZ - 21903105 - yusuf.senyuz@ug.bilkent.edu.tr
Cengizhan TERZİOĞLU - 22003014 - cengizhan@ug.bilkent.edu.tr
Yunus Eren TÜRKERİ - 22001842 - eren.turkeri@ug.bilkent.edu.tr
Arda YILDIZ - 22003093 - yarda@ug.bilkent.edu.tr

Supervisor: Özcan ÖZTÜRK

Course Instructors: Atakan ERDEM, Mert BIÇAKÇI

Date: 12.05.2024

Table of Contents

Detailed Design Report: *SecuRent*

1. Introduction

1.1 Purpose of the system

In Turkey, the renewal time for a rent contract is once per year, according to law [1]. However, this is not the case when it comes to practice. Some landlords still believe that they have the right to increase the rent price before a contract expires in one year. Hence, tenants complain about the unlawful increments in the rent prices, and unfortunately, the complaints lead to heavy discussions between tenants and landlords. Unfortunately, discussions often lead to violence in between two parties. What is more, tenants and landlords often argue about who should keep the deposit of the house, with regards to the conditions of the house after the contract expiration date. Hence, the tension between the two parties tends to increase because of this situation as well.

The SecuRent application resolves/alleviates the aforementioned issues in the following ways: Firstly, SecuRent introduces the government as the guarantor to the renting process. Henceforth, The tenant can request a government official to evaluate the renting price of the house before a contract is signed. Next, within the contract period, if an increment has been made to the rent price by the landlord, the government will interfere, stating that the price increment is illegal. According to the law, no increment can be made to the house rental prices during the contract. Only increment can be made before the contract renewal with 25% increment rate, in regards to the law passed on July 1st 2023 [2]. Hence, as the government will be the authority in the rent process and act as the guarantor for the contract, the unlawful rent price increases will be prevented. Furthermore, with the SecuRent application, the deposit of the rented house will be kept in the government vault. As the contract expires, the conditions of the house will be evaluated by a real estate agent. Therefore, the real estate agent can evaluate the house conditions and register his/her evaluations to the SecuRent system, concluding with a decision on who should keep the deposit. Henceforth, the government can send the deposit to either the tenant or the landlord, upon the real estate agent's decision. Thus, the disputes between tenants and landlords caused by deciding on who should keep the deposit will be resolved with the contribution of third parties such as the real estate agent and the government, given that they should act objectively.

1.2 Design goals

In this section, the design goals of the system are addressed. The design goals are the quality attributes of SecuRent system.

1.2.1 Usability

To enhance the usability, the user interface must be simple and easy to understand. Users of the application can vary substantially. Users from all age groups (starting from 18 years old minimum) should be able to use the application without difficulty. Our aim is to make an application that the most of the citizens of Turkey will use. Possible tenants should be able to access the particular home within at most 10 clicks. To ensure the usability, we are designing simple and easy to understand home pages while utilizing the React.js and Bootstrap libraries to design easy to understand navigation bars and user interfaces in general.

1.2.2 Reliability

SecuRent application has millions of prospective users. For this reason, the application must be reliable and it should not crash in millions of user traffic. Hence, we will use AWS services to maintain reliability for our application. Specifically, we will use AWS EC2 along with AutoScaling group and Elastic Load Balancer to balance the traffic by distributing on different processors. Moreover, we will use AWS VPC to establish our own private network for our application to ensure that it works smoothly. Furthermore, to ensure the reliability of our application, we will write unit tests for our front end and back end source codes. For the back end we will use JUnit test library to write our tests and Spring annotations for configuration. For the front end we will use Jasmine test library to write our tests and Karma for configuration. Hence, we will test the code that we have written. The standard that we have come to consensus is that we will have a coverage rate of 80% to ensure reliability via unit tests. Last but not least, we will implement rate limits to our front end and back end source codes, to prevent our application crashing from DDoS attacks. Hence, our application will not be met with an enormous number of requests (such as 10000 requests) within a millisecond, and will not crash.

1.2.3 Supportability

SecuRent is aimed to be used for at least 10 years. In its lifetime, new technologies may arise and some of the users might start to use these technologies. SecuRent should be accessible in these new technologies because it will have millions of users all around Turkey. For instance, SecuRent should work smoothly in different browsers such as Chrome, Opera, Edge etc. In addition, since we will use cloud based solutions (AWS services), the system we provide will be long lasting as long as the hosting fees are paid. Thus, the application should also support a new/emerging browser as well to ensure supportability.

1.2.4 Scalability

SecuRent is aimed to be used by all the tenants, landlords and real estate agents in the country. Therefore, it will already be used by possibly thousands or millions of users beginning from the first day it becomes live after the public release. Also the number of the potential users might increase as the population of the country increases or more houses are built in the long term. Hence, SecuRent should be scalable to be able to increase the capacity of the servers when it is needed. Therefore, we are integrating Amazon EC2 in our system to host and respond to the increasing number of users. With EC2's tailored scalability options thanks to AutoScaling group and Elastic Load Balancer (ELB), we will address the issues varying from increasing number of users to traffic spikes with respect to the Standard and Spot options provided by the AWS.

1.2.5 Security

SecuRent collects countless sensitive data such as Turkish Republic identity number, address and phone number of the users, contract information, etc. All of this data will be protected by a set of measures. First of all, AWS WAF (Web Application Firewall) will be integrated into the system as it is a service that helps to protect our web application from common web exploits that could affect application availability, compromise security, or consume excessive resources. With AWS WAF, we will create rules to filter web traffic based on conditions that we define to protect our application from malicious third parties and users. In addition, as it was mentioned in section 1.2.2, AWS VPC will be used to provide a secure and isolated private network which we will use to host our application. What is more, we will use AWS IAM to manage access to AWS services and resources securely which are integrated to our system. We will create user roles and security rules to prevent unauthorized access to the system and its resources. Furthermore, the multi factor authentication must be added to the system, with a confirmation mail sent to a user. In the system, as the tenants will make payments through the application, money transactions must be secure so that no user loses their money. Hence, we will cooperate with multi factor authentication again to provide security. Last but not least, we will implement rate limits to our front end and back end source codes, to prevent our application crashing from DDoS attacks. Hence, our application will not be met with an enormous number of requests (such as 10000 requests) within a millisecond, and will not crash, as mentioned in the reliability section as well.

1.2.6 Maintainability

To ensure maintainability we have made a set of architectural and design decisions. Here is a list of the decisions we have made to provide maintainability in our application:

- We will provide documentation for our codebase through README.md files and our webpage. Hence, the documentation will act as a guide when maintenance is required.
- We have decided to use module style as our architectural style. Hence, we will implement our system in modules, which comes with meaningful units of development in itself. Thus, when maintaining the codebase, each module can be put into investigation separately.
- We are using version control tools such as Git. We are committing with meaningful and self descriptive commit messages. Hence, it becomes easier to backtrack and build upon a certain commit when needed.
- We will use SonarQube to have clean code in our codebase. Therefore, we will follow the common coding practices and correct our critical issues via SonarQube tool to have a more maintainable code.

- We are providing comments in our codebase to help each other work efficiently. Therefore, multiple people can work and communicate effectively on the same source code with the help of the comments. Moreover, when someone takes over the codebase in the future, the comments will help the future developers to understand the legacy code easier and ultimately maintain it easier as well.

1.2.7 Marketability

SecuRent is a groundbreaking application that revolutionizes the house rental process by addressing key issues faced by tenants and landlords. One of its primary features is the ability for tenants to request government-appointed officials to determine a fair rental price if they disagree with the landlord's set price, ensuring tenants do not overpay for their accommodation. Additionally, the application allows landlords to upload rental contracts, which are monitored by the government to prevent unauthorized rent increases. Furthermore, SecuRent streamlines the deposit process by collecting deposit money at the government treasury and disbursing it based on the house's condition at the end of the contract, eliminating conflicts over deposit refunds between tenants and landlords. By offering these innovative solutions, SecuRent enhances marketability by addressing critical pain points in the house rental process, making it an attractive and reliable option for both tenants and landlords alike. Last but not least, there is no direct competitor for SecuRent. Currently, there are 2 similar applications however, they lack the essential features that SecuRent provides: Endeksa and Mydeposits. These two applications will be discussed in section 2.1.

1.2.8 Modularity

SecuRent is composed of separate, independent modules that interact with each other through well-defined interfaces. This allows for easier development, as each module can be developed, tested, and maintained independently. Additionally, modularity improves system flexibility and scalability, as modules can be added, removed, or modified without affecting the entire system. In the context of SecuRent, modularity enables the system to evolve and adapt to changing requirements and technologies, ensuring that it remains efficient, reliable, and easy to maintain throughout its lifecycle. For these regions, the system has a modular architecture which comprises decomposition, layered. In sections 3.1 and 3.2, the modular structure of the system will be investigated via diagrams.

1.3 Definitions, acronyms, and abbreviations

- AWS: Amazon Web Services, is a cloud computing platform by Amazon that offers a variety of cloud-based services, including computing power, storage, database management, and content delivery. Currently, we utilize its computing, network, and storage services.
- EC2: Amazon Elastic Compute Cloud. It provides scalable computing capacity in the cloud.
- Spot: An Amazon EC2 pricing model that allows you to bid on spare Amazon EC2 computing capacity.
- VPC: Virtual Private Cloud. It allows you to create a private network within the AWS cloud.

- S3: Simple Storage Service, is an Amazon AWS offering that provides object storage. It stores objects in buckets and is used in our project to store large files like PDFs and images uploaded by driver users.
- WAF: Web Application Firewall. A firewall that protects web applications from common web exploits.
- IAM: Identity and Access Management. It helps you manage access to AWS services and resources securely.
- ELB: Elastic Load Balancer. It distributes incoming application or network traffic across multiple targets, such as EC2 instances, in multiple Availability Zones.
- DDoS: Distributed Denial of Service. An attack where multiple compromised systems are used to target a single system, causing a denial of service for users.
- JUnit: A unit testing framework for Java programming language.
- React.js: A JavaScript library for building user interfaces.
- Bootstrap: A front-end framework for developing responsive and mobile-first websites.
- Jasmine: A behavior-driven development framework for testing JavaScript code.
- Karma: A test runner for JavaScript that runs tests in real browsers.
- API: Application Programming Interface. It defines the methods and data formats that applications can use to communicate with each other.
- JSON: Abbreviation for JavaScript Object Notation, is a human-readable format for representing objects. It is easy to parse and serves as an excellent format for exchanging data.
- UI: Stands for User Interface, encompasses the visual and interactive components of software or hardware systems that enable user-system interaction, aiding users in achieving their goals efficiently.
- UML: The Unified Modeling Language, establishes standards for object modeling and visualization. All our diagrams were created using UML.

1.4 Overview

The SecuRent is an application which enhances the house rental process as well as addressing the issues that are related with the house rental process. SecuRent application emerges as a pioneering solution designed to transform the dynamics between tenants and landlords. With SecuRent, tenants can request officials (who are assigned by the government) to determine the rental price of a house, if they are not in favor of the price set by the landlord. Thus, ballooned prices set by the landlords can be regulated by the officials to prevent tenants paying more than what is worth for a particular house. Furthermore, if a house is rented, the status of the house will be updated in the application and a contract uploaded by the landlord will enable the government to monitor the contract. If a landlord wants to increase the rent price before the contract expiration date, the government will be notified with a request. Hence, the rent prices will not be increased without the government's consent, ultimately preventing unwanted skyrocketing prices that cause hardships for tenants. Moreover, with SecuRent, the deposit money for houses will be collected at the government treasury. The implication of this use case is that the deposit money will be given to either the tenant or the landlord depending on the condition of the house after the contract expires. Thus, conflicts between tenants and landlords caused by deciding who will keep the deposit money will be solved, as the government will act as the mediator in this process.

2. Current Software Architecture

Currently, there is no existing application comparable to SecuRent that we can further develop. While there are renting applications available, they do not encompass the essential features of SecuRent and are typically designed for specific companies. We are also working on implementing a payment system within the app to facilitate financial operations integrated to the system. Our aim is to enhance user experience with the user-friendly SecuRent application and expand our reach to the entire renting sector.

2.1 Possible Competitors

In the market, there is no robust competitor for Securent, as mentioned in the previous paragraph. There are similar projects that include some features of SecuRent, but these projects do not function similarly and their aims are not the same. On the internet, 2 projects/websites show a similarity in the same application domain as SecuRent: Endeksa and mydeposits.

Endeksa is a service which offers the users to buy or rent a house. It is supported with machine learning which predicts the market price of a house. The website has some fields to fill in when a landlord creates a publication of a house to be sold. These fields include specific information about the floor, facade and the condition of the flat. Users can also examine in a specific location to look for the renting/selling opportunities. Though it is in the same application domain as SecuRent, the contract system, interactions with the government and the depositing system is different from SecuRent. So, in practice Endeksa is not a challenging competitor for SecuRent since they function differently and they offer different services [3].

Mydeposits is a service which is constructed by private rental sector experts. It is a deposit protection service which landlords, agents and tenants can use. The website is dedicated and designed to protect the deposits mainly, so there is no governmental profit from these deposits or detailed tracking of contracts [4]. Regarding these missing features, mydeposits is not a powerful competitor for SecuRent since the functionalities and scope is different.

3.1 Overview

SecuRent is a web application which easens the house rental process as well as addressing the unlawful rent price increment and deposit retrieval problems. SecuRent is innovative since there is no application that resolves the unlawful rent price increment and deposit retrieval conflicts.

The system followed a three layered architectural style: Presentation layer, business layer, and data layer. In the presentation layer, we enabled the clients to interact with the system. Hence, we implemented the front end of the system. To do this, we used JSX syntax of JavaScript alongside React.js library of JavaScript, as well as Bootstrap. Proceeding from the presentation layer to the business layer, we used Axios to handle the communication between these layers. In the business layer, we implemented the business logic of the SecuRent system. Hence, we implemented the back end functionality of the system, while responding to the functional and non functional constraints that will be discussed next. We used the Java programming language and Spring Boot framework to implement the business layer. Lastly, we have the data layer, which stores the entities and their properties. To implement this layer, we used PostgreSQL, a relational database, which will be in communication with the business layer through SQL queries. Last but not least, since SecuRent is intended to be an enterprise application, we used Amazon Web Services (AWS) to host our application. Specifically, we used Amazon Elastic Compute Cloud (Amazon EC2) to achieve a scalable and adjustable application, as EC2 comes with the features responding to the increasing numbers of users or sudden spikes (stark increments in the number of users) [5].

The high-level system architecture of the system is given in the next page. In the architecture, module style is used. Main features of the system are grouped into modules and their relationship with each other is displayed with arrows. Moreover, layered architectural style was used, since we have different layers to respond to front end and back end requirements of the system, as well as API layers to maintain communication in between layers. Lastly we implement the use style, since particular modules use the functionality of other modules.

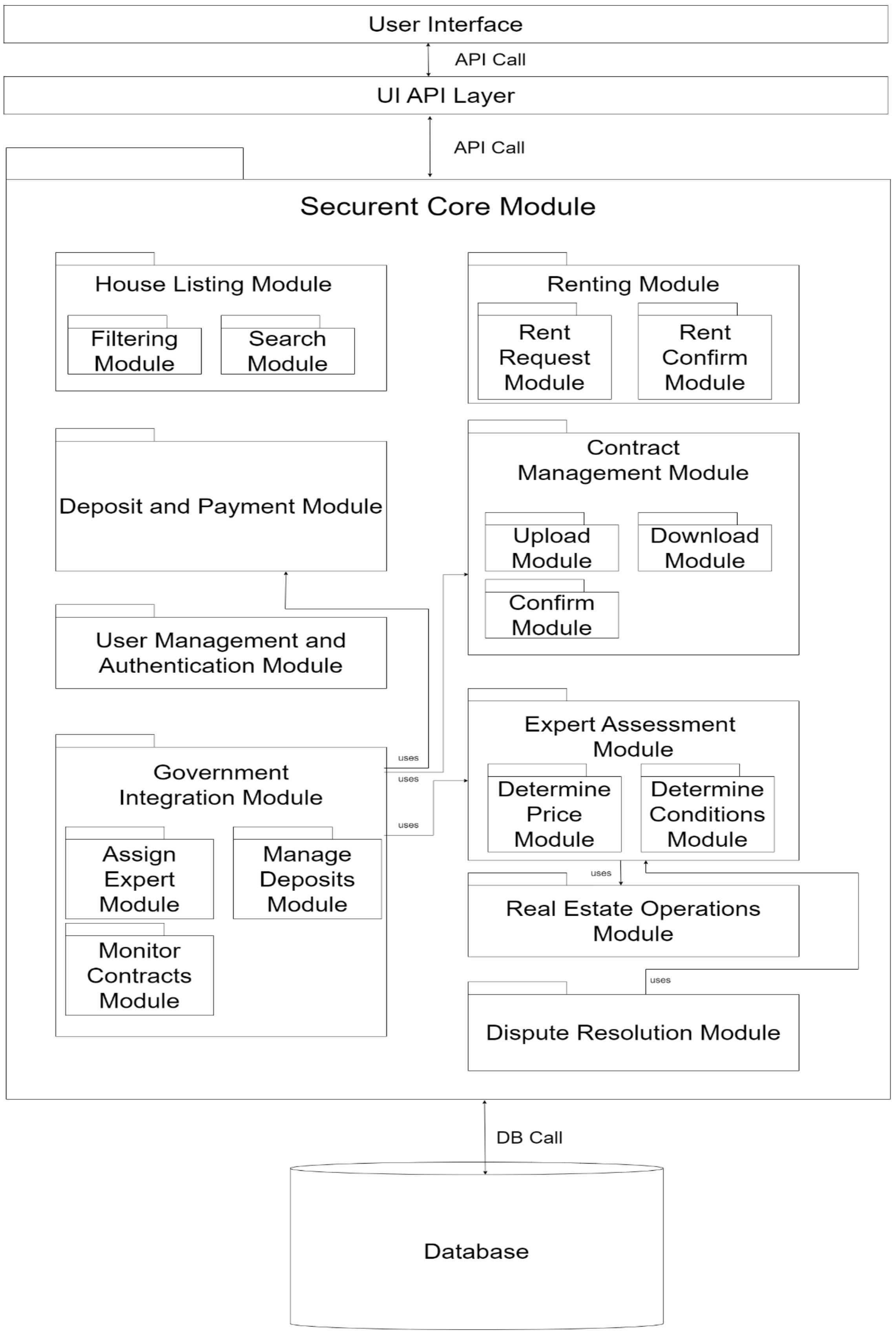

Figure 1: High Level System Architecture

3.1.1 Description of the High Level System Architecture

- Deposit and Payment Module: Deposits will be paid with this module.
- User Management and Authentication Module: In this module, authentication functionality will be covered.
- Government Integration Module: In this module, government related functionality will be covered.
  - Assign Expert Module: The government will assign experts to houses, with the functionality covered in this module
  - Manage Deposits Module: The government will collect and distribute deposits with this module.
  - Monitor Contracts Module: With this module, the government will monitor the contracts.
- Renting Module: The renting functionality will be covered within this module.
  - Rent Request Module: In this module, the tenant will request a house to be rented after negotiations are completed with the landlord.
  - Rent Confirm Module: The landlord will confirm the rent request of the tenant and the status of the house will change to the rented. Hence, the house will not be displayed to other tenants for rental purposes after the confirmation, since it has already been rented by a tenant.
- Contract Management Module: The functionality will be covered within this module.
  - Upload Module: Landlords can upload rental contracts to the system, and the government monitors these contracts.
  - Download Module: Uploaded rental contracts to the system can be downloaded with this module.
  - Confirm Module: The government will confirm the contract whether it belongs to the renting. If the uploaded document is an irrelevant document, the landlord will be notified to upload the contract.
- Expert Assessment Module: The expert judgments on the house to be rented will be carried out in this module.
  - Determine Price Module: The renting price of the house will be determined by the official assigned by the government. The functionality will be covered in this module.
  - Determine Conditions Module: The condition of the house will be determined by a real estate agent. The real estate agent will be chosen by the tenant to provide an objective judgment. Next, the conditions will be entered into the system by the real estate agent to help the government decide on which party will receive the deposit depending on the conditions.
- Real Estate Operations Module: Tenants can choose a real estate agent from multiple options in a specific area to assist in the rental process. Moreover, tenants can request a real estate agent to investigate the conditions of the house after the expiration of the contract.
- Dispute Resolution Module: Upon the real estate agent's evaluation of the house conditions, the government will determine whether the deposit goes to the tenant if the house is in good condition or to the landlord if the house is damaged.
- User Interface: The GUI in which the clients will interact with the system.

- UI API Layer: The API layer which serves as the communication channel between the back end and the front end of the system. HTTP requests are used to make calls in between layers.
- Database: The data related to the houses and contracts will be stored in the database. A relational database will be used and SQL queries are used to make calls to the database.

3.2 Subsystem Decomposition

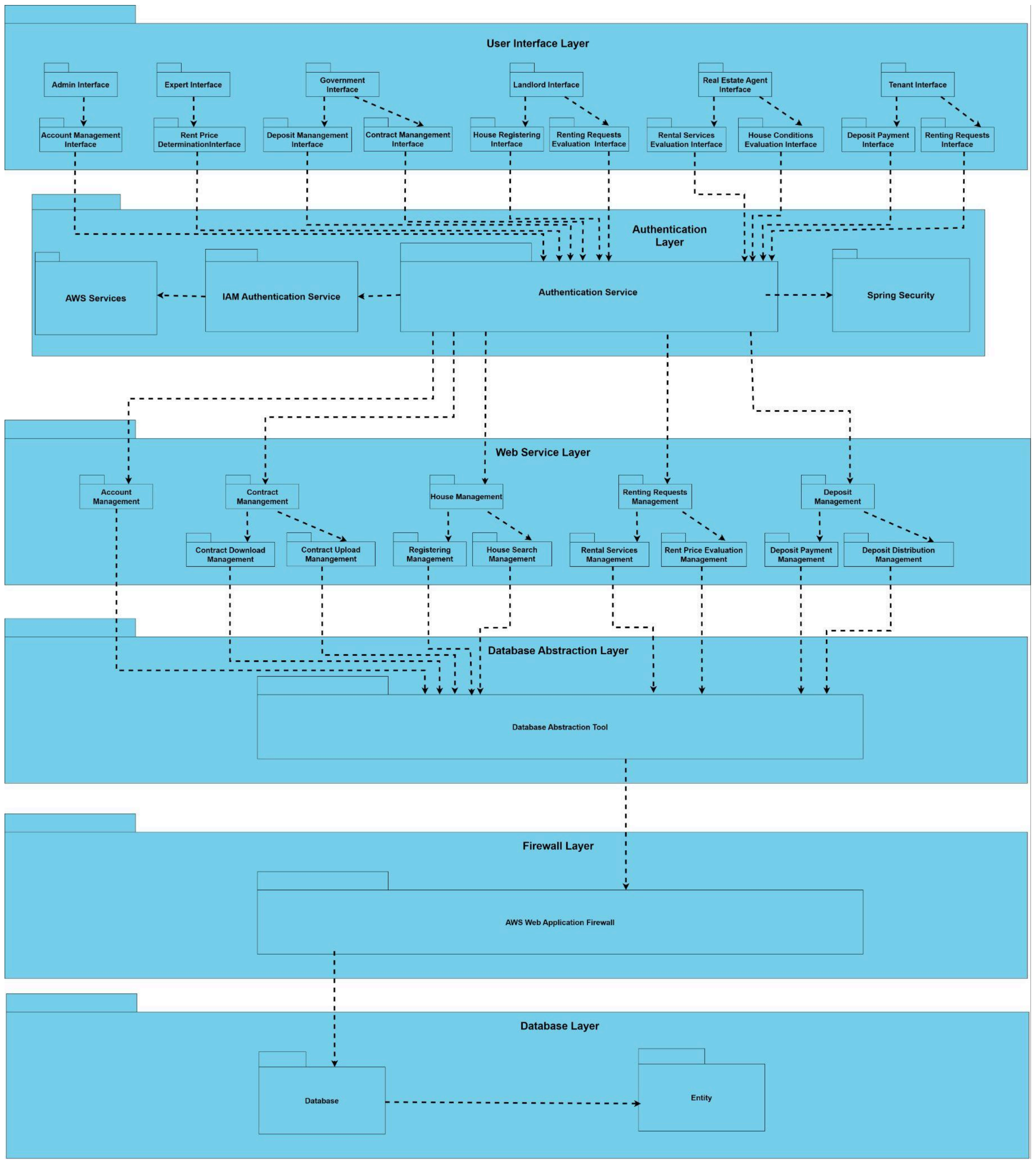
User Interface Layer
Admin Interface
Expert Interface
Government Interface
Landlord Interface
Real Estate Agent Interface
Tenant Interface
Account Management Interface
Rent Price DeterminationInterface
Deposit Manangement Interface
Contract Manangement Interface
House Registering Interface
Renting Requests Evaluation Interface
Rental Services Evaluation Interface
House Conditions Evaluation Interface
Deposit Payment Interface
Renting Requests Interface
Authentication Layer
AWS Services
IAM Authentication Service
Authentication Service
Spring Security
Web Service Layer
Account Management
Contract Manangement
House Management
Renting Requests Management
Deposit Management
Contract Download Management
Contract Upload Manangement
Registering Management
House Search Management
Rental Services Management
Rent Price Evaluation Management
Deposit Payment Management
Deposit Distribution Management
Database Abstraction Layer
Database Abstraction Tool
Firewall Layer
AWS Web Application Firewall
Database Layer
Database
Entity

Figure 2: Subsystem Decomposition of SecuRent

User Interface Layer: This layer represents the interface through which users interact with the SecuRent system and its features. It includes components such as forms, buttons, and menus that users use to input data and receive feedback.

Authentication Layer: This layer is responsible for authenticating users and ensuring that only authorized users can access the system. It may include components for login, password management, and access control. AWS IAM will be used for authentication along with the Spring Security.

Web Service Layer: This layer provides the interface for external systems or clients to interact with the application. It includes RESTful services for data exchange, which will be implemented in the controllers for each entity.

Database Abstraction Layer: This layer abstracts the database implementation details from the rest of the application. It provides a simplified interface for accessing and manipulating data, independent of the underlying database technology. We used Spring Data JPA along with Hibernate for this layer.

Firewall Layer: This layer protects the system from unauthorized access and cyber threats. It may include components for filtering incoming and outgoing network traffic, as well as monitoring for suspicious activity. AWS WAF will be utilized for this layer.

Database Layer: This layer represents the actual database where the system's data is stored. It includes components for data storage, retrieval, and management, using a PostgreSQL database management system. MongoDB will be used as well for storing contracts uploaded to the system.

3.3 Hardware/Software Mapping

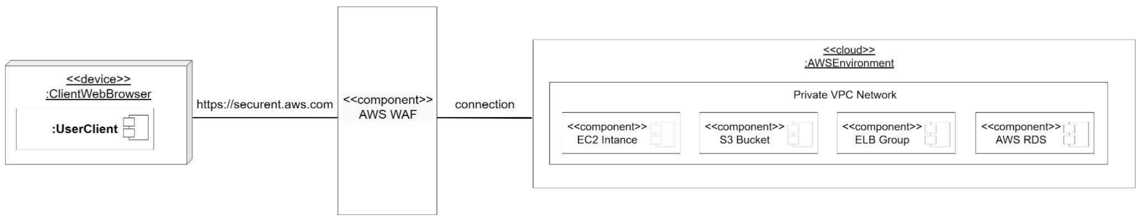

Figure 3: Hardware/Software Mapping of SecuRent

In the hardware/software mapping of SecuRent, the application's core components are allocated to specific hardware or software elements. For hardware, Amazon EC2 instances are designated for hosting, ensuring the application's availability and scalability. The Elastic Load Balancer is utilized to evenly distribute incoming traffic among these instances, enhancing performance and reliability. Auto Scaling groups are implemented to automatically adjust the number of instances based on traffic volume, optimizing resource utilization. On the software side, Java and Spring Boot are employed for the backend's business logic, providing a robust and efficient foundation. PostgreSQL is chosen as the database, offering a reliable and scalable data storage solution. Since AWS RDS is compatible with the PostgreSQL database, the service is utilized in the cloud environment.

3.4 Persistent Data Management

In SecuRent, a PostgreSQL instance will be integrated with AWS RDS, and the database instance will be maintained within the AWS environment. Given that our backend application is developed in Java Spring Boot, opting for an SQL database like PostgreSQL is logical. The backend team chose PostgreSQL due to their prior experience with it, reducing the learning curve. The database will store user data such as profile information, TCK, rented houses, and houses owned. AWS will manage the database instance, and it will be accessed through the REST API running on the Java Spring Boot framework.

3.5 Access Control and Security

We have opted to utilize the Java programming language and the Spring Boot framework for constructing the business layer of our project. To handle authentication, we have chosen SAML 2.0, facilitated by Spring Security. SAML 2.0 operates on an XML-based protocol, utilizing security tokens that carry assertions to convey information about a user (typically an end user) between a SAML authority, known as an Identity Provider, and a SAML consumer, known as a Service Provider.

Additionally, our authentication strategy includes Remember-Me Authentication, which employs a straightforward hash-based token method. This approach utilizes hashing to establish an effective remember-me functionality. Essentially, upon successful interactive authentication, a cookie is transmitted to the browser.

Moreover, we have incorporated X.509 authentication. This authentication method is commonly employed to verify the identity of a server in SSL usage, particularly with HTTPS from a browser. The browser automatically verifies that the server's certificate is issued (digitally signed) by one of the trusted certificate authorities it maintains.

4. Subsystem Services

4.1 User Interface Layer

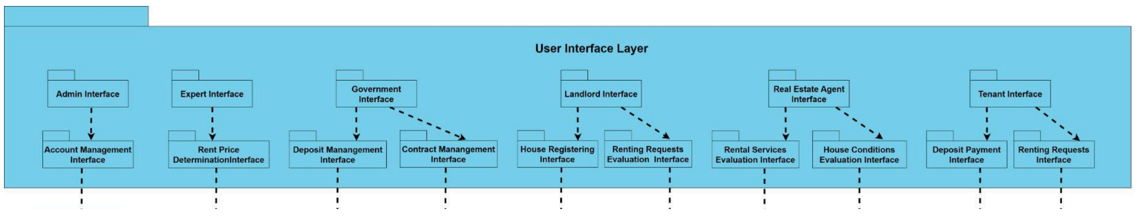
User Interface Layer
Admin Interface
Expert Interface
Government Interface
Landlord Interface
Real Estate Agent Interface
Tenant Interface
Account Management Interface
Rent Price DeterminationInterface
Deposit Manangement Interface
Contract Manangement Interface
House Registering Interface
Renting Requests Evaluation Interface
Rental Services Evaluation Interface
House Conditions Evaluation Interface
Deposit Payment Interface
Renting Requests Interface

Figure 4: User interface layer of SecuRent

User interface layer has the front-end pages that the end-users will be able to view and use. These are:

Admin Interface: Manages the pages for the admins of SecuRent

Account Management Interface: The user interface which will be used by admins to add, delete and update accounts in SecuRent

Expert Interface: Manages the pages for the experts that are assigned by the government

Rent Price Determination Interface: The user interface which will be used by experts to enter rent prices of the houses after conditions of those houses are evaluated

Government Interface: Manages the pages for the government officials

Deposit Management Interface: The user interface which will be used by government officials to retrieve and withdraw the deposits of the houses

Contract Management Interface: The user interface which will be used by government officials to monitor rental contracts between tenants and landlords

Landlord Interface: Manages the pages for the landlords

House Registering Interface: The user interface which will be used by landlords to add their houses to the SecuRent database

Renting Requests Evaluation Interface: The user interface which will be used by landlords to monitor, accept or reject the renting requests coming from the tenants for their houses

Real Estate Agent Interface: Manages the pages for the real estate agents

Rental Services Evaluation Interface: The user interface which will be used by real estate agents to enter the results of the rental services that are requested by the tenants

House Conditions Evaluation Interface: The user interface which will be used by real estate agents to enter the house conditions before and after the renting period

Tenant Interface: Manages the pages for the tenants

Deposit Payment Interface: The user interface which will be used by tenants to pay the deposits of the houses they rented and retrieve them back when they decide to leave the house

Renting Requests Interface: The user interface which will be used by tenants to send renting requests to the landlords

4.2 Authentication Layer

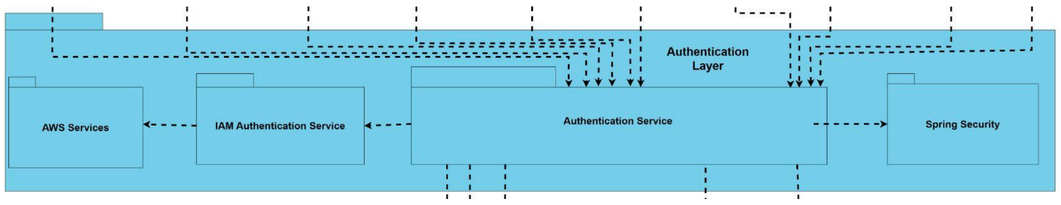

Figure 5: Authentication layer of SecuRent

Authentication layer has the services that are required to determine whether a user has access to a certain page or not. These are:

Authentication Service: The main service which determines whether the user has access to a page or not

Spring Security: Security package of the Spring Boot

IAM Authentication Service: An AWS service that securely controls access to other AWS services

AWS Services: Cloud services which are provided by Amazon

4.3 Web Service Layer

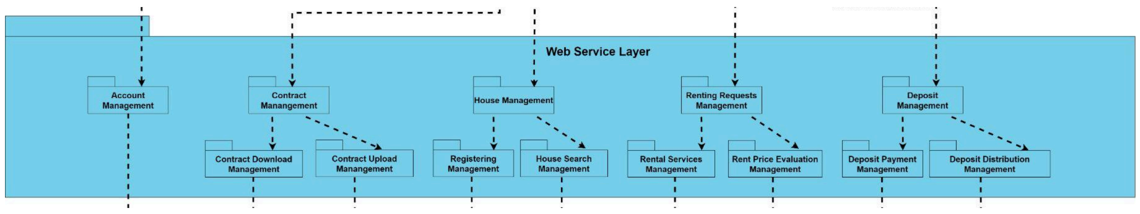
Web Service Layer
Account Management
Contract Manangement
House Management
Renting Requests Management
Deposit Management
Contract Download Management
Contract Upload Manangement
Registering Management
House Search Management
Rental Services Management
Rent Price Evaluation Management
Deposit Payment Management
Deposit Distribution Management

Figure 6: Web service layer of SecuRent

Web service layer has the services that are required to perform the actions of users. These are:

Account Management: Service which is used to add, delete and update user accounts

Contract Management: Service which is used for all rental contract related operations

Contract Download Management: Service which is used to download rental contracts to the system

Contract Upload Management: Service which is used to upload rental contracts from the system

House Management: Service which is used for all house related operations

Registering Management: Service which is used to add, delete or update houses

House Search Management: Service which is used to query houses from the database

Renting Requests Management: Service which is used for all renting request related operations

Rental Services Management: Service which is used to create or evaluate rental service requests

Rent Price Evaluation Management: Service which is used to create house price evaluation requests and enter the evaluation results of those requests

Deposit Management: Service which is used for all deposit related operations

Deposit Payment Management: Service which is used to get deposit payments of the tenants

Deposit Distribution Management: Service which is used to distribute the deposits from government back to tenants or landlords

4.4 Database Abstraction Layer

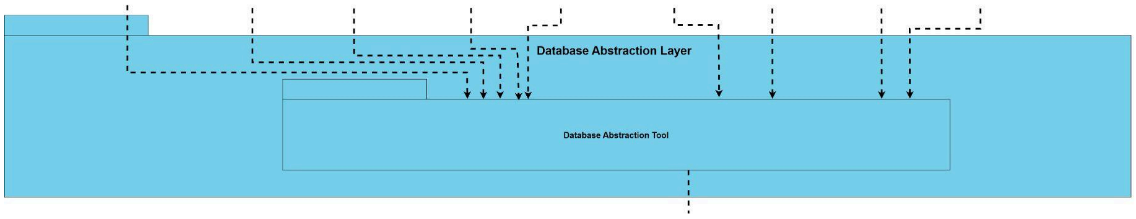

Figure 7: Database abstraction layer of SecuRent

Database Abstraction Tool: In our project, we utilize Spring Data JPA in conjunction with Hibernate as our database abstraction tool. This combination allows us to abstract the data to a format that is appropriate for the database we are using. This abstraction simplifies the interaction with the database, reducing the need for boilerplate code and providing additional features such as caching, lazy loading, and transaction management.

4.5 Firewall Layer

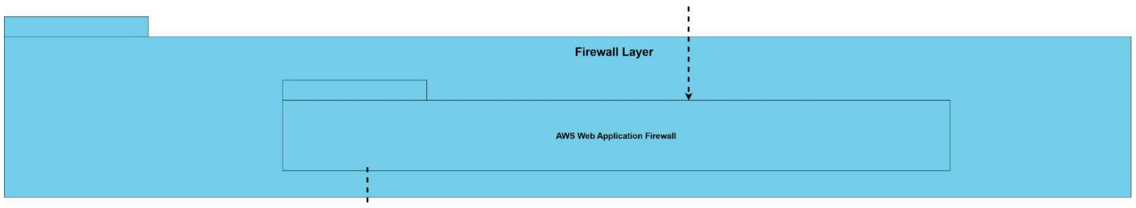

Figure 8: Firewall layer of SecuRent

AWS Web Application Firewall: The AWS Web Application Firewall (WAF) is a critical component of our infrastructure, serving as a protective barrier for our web servers against a variety of external attacks. By leveraging the AWS WAF, we can defend our web applications from common threats such as SQL injection, cross-site scripting (XSS), and other malicious activities. This security measure is essential for maintaining the integrity and availability of our web services, ensuring that they remain resilient and secure against potential threats.

4.6 Database Layer

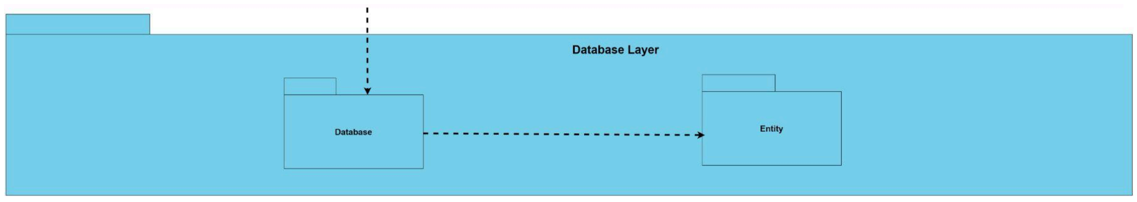

Figure 9: Database layer of SecuRent

The database layer of the SecuRent application encompasses both the database itself and the entity classes.

Database: The database serves as the repository for all the data that the SecuRent application relies on, storing information such as user profiles, rental listings, and transaction records. We are using PostgreSQL as our database management tool and pgAdmin as a graphical administration tool to manage our tables stored in the database.
Entity: The entity classes represent the various models within the application, defining the structure and behavior of the data. Here are the entities of the SecuRent application: Admin, Expert, Government, House, HouseProperties, Landlord, RealEstateAgent, RentalAd, RentalContract, RentRequest, Tenant.

Together, these components form the foundation of the application's data management system, ensuring that data is stored efficiently and accurately. The database layer plays a crucial role in the overall functionality and performance of SecuRent, providing a reliable and secure means of storing and accessing data essential to the application's operation.

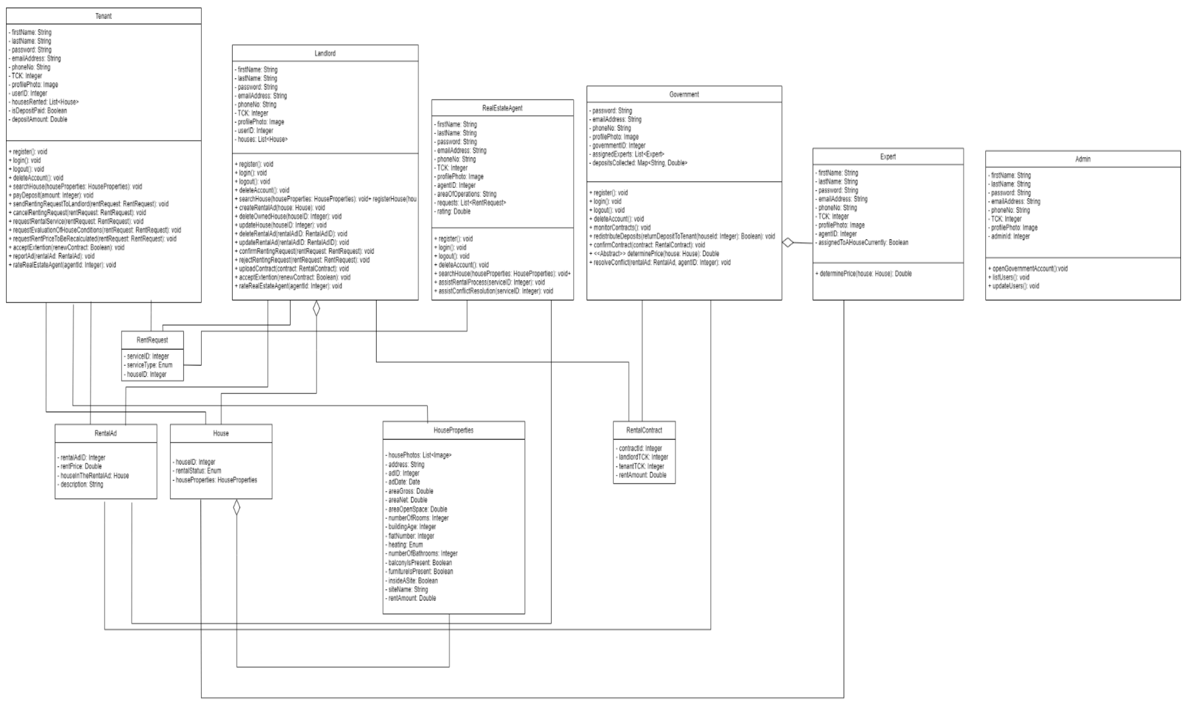

Figure 10: Entity relationship diagram of SecuRent

5. Development/Implementation Details

While developing our project, we used the agile methodology. It gave us an iterative approach that emphasizes flexibility, collaboration, and continuous improvement. We used the Jira software and used weekly sprints throughout the year.

Before we start implementing the project we divided our group to 2 smaller teams: backend and frontend teams. The frontend team finished most of the mockup screens (static design of the web page) in the first semester. Backend team started to implement business logic and model classes in the first semester.

In the second semester the frontend team started to do the following:

- They learn Swagger for establishing API endpoint connections.
- They merged UI JSX files mockup classes into a single function React component.
- They merged profile JSX files into one.
- They created test cases.
- They finished designing and coding the pages of the website. They implemented the pages such as: ApprovedContracts, Contracts, ForgotPassword, HomePage for each user, HouseDetails, LandlordCurrentRequest, NavBar for each user, Profile for each user.
- They added images of houses.

In the second semester the backend team started to do the following:

- They implemented service modules for the Government class.
- They implemented service modules for the Landlord class.
- They implemented service modules for the Tenant class.
- They implemented backend functionalities for the implemented classes.
- After the service classes, necessary repositories are created for database operations such as RentRequestRepository, TenantRepository, UserRepository, RealEstateAgentRepository, LandlordRepository, GovernmentRepository and HouseRepository.
- They developed login and sign up pages.
- They created exception classes for errors.
- They created enums files for improved code readability. Examples of those enums files are: HouseType, Role, ServiceType, and GovernmentServiceType.
- They created controller files for pages.
- They added authentication.

After, both the backend and frontend team finished their parts, we connected backend and frontend by arranging API Endpoints. We exposed the API endpoints from the backend server to allow the frontend to make requests. We ensured that CORS (Cross-Origin Resource Sharing) is properly configured to allow requests from the frontend domain. We used Axios.

After we completed our project,we used Amazon Web Services (AWS) to host our application. We deployed our project by using Amazon EC2 (Amazon Elastic Compute Cloud).

Throughout the year we used GitHub for version control, collaboration and project management. We were able to track changes in our codebase, and undo the changes if necessary. We used previous versions when needed. This helped us work on the same project concurrently without risk of overwriting each other's changes. We used various branches such as backend, frontend and additional branch for each member, we sometimes used additional branches for some important features. Then we merged changes from one branch into another.

6. Test Cases

We have used the following severity levels for our test cases: low, intermediate and critical.

6.1 Upload Photo Function

Test ID	001	**Category**	Functional	**Severity**	Intermediate
Objective	This test case verifies if the user can upload a photo (i.e., profile picture).				
Steps	1. Login as a tenant. 2. Navigate to the profile screen. 3. Click on “upload photo”. 4. Select a photo from the file browser. 5. Click “Select”.				
Expected	A new photo will be uploaded into the system.				
Date-Result	07/05/2024 - Passed.				

6.2 Filter Function

Test ID	002	**Category**	Functional	**Severity**	Low
Objective	This test case verifies if the user can filter the number of rental ad results (i.e., on the home page, where the user scrolls through the rental ads).				
Steps	1. Login as a tenant. 2. Navigate to the filter button on the home page (for tenants, real estate agents, and state officials). 3. Change the max result from the dropdown.				

	4. Click on “Filter”.
Expected	A filtered version of the rental ads will appear on the home page.
Date-Result	07/05/2024 - Failed because we decided not to use a filter component.

6.3 Search Function For Houses

Test ID	003	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if the user can use the search bar for the rental house ads.				
Steps	1. Login as a tenant. 2. Navigate to the search bar on the home page. 3. Search for any ad. 4. Click on the “Search” button.				
Expected	The ad results to be displayed will change related to the searched value.				
Date-Result	07/05/2024 - Passed.				

6.4 Display Details Function

Test ID	004	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if the user can click and view the details of an ad.				
Steps	1. Login as a tenant. 2. Scroll through the houses on the home page. 3. Click on the details of an ad.				
Expected	A new page with the details of an ad should be displayed.				
Date-Result	07/05/2024 - Passed.				

6.5 Submit Report Function For Reporting Ad

Test ID	005	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if the user can submit a report about an ad.				
Steps	1. Login as a tenant. 2. Navigate to the details of an ad by clicking the “Details” button.				

	3. Click on the “Report Ad” button.
Expected	A modal will pop up in which a user can report an ad.
Date-Result	07/05/2024 - Failed because later we decided that the report ad function is out of our scope.

6.6 Preview Function For Displaying Real Estate Agents

Test ID	006	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if the user can preview the list of real estate agents.				
Steps	1. Login as a tenant. 2. Click on the “Details” button on an ad. 3. Click on the “Preview” button in the ad details.				
Expected	A new page will load showing the list of real estate agents.				
Date-Result	07/05/2024 - Passed.				

6.7 Select Agent Function

Test ID	007	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a tenant can select a real estate agent.				
Steps	1. Login as a tenant. 2. Click on the “Details” button on an ad. 3. Click on the “Preview” button in the ad details. 4. Click on “Select Agent” in the real estate agent list				
Expected	A notification will be sent to the corresponding real estate agent.				
Date-Result	07/05/2024 - Passed.				

6.8 See Profile Function

Test ID	008	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a tenant can see the profiles of a real estate agent through a list of them.				
Steps	1. Login as a tenant. 2. Click on the “Details” button on an ad. 3. Click on the “Preview” button in the ad details.				

	4. Click on “See Profile” in the real estate agent list
Expected	A new page showing the corresponding real estate agent’s profile will appear.
Date-Result	07/05/2024 - Passed.

6.9 Display Remaining Time For Contract

Test ID	009	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a tenant can see the remaining time for the contract end.				
Steps	1. Login as a tenant. 2. Click on the “Rented House” in the top nav bar in tenant home page.				
Expected	A page will show the remaining time for the contract end, with a button that will let the user select a new agent.				
Date-Result	07/05/2024 - Failed because we changed that page’s design and the new design does not involve remaining time.				

6.10 Display Notification Function For Landlord

Test ID	010	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a landlord can see their notifications.				
Steps	1. Login as a landlord. 2. Click on the “Notifications” tab on the top nav bar.				
Expected	A page will show the rental notifications to a landlord user, with the requests and who made that request, along with their photos.				
Date-Result	07/05/2024 - Passed.				

6.11 Accept Notification Function For Landlord

Test ID	011	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a landlord can accept a rental request.				
Steps	1. Login as a landlord. 2. Click on the “Notifications” tab on the top nav bar.				

	3. Click on “Accept” for a rental request.
Expected	The rental request will be accepted and the request will drop to the current requests page.
Date-Result	07/05/2024 - Passed.

6.12 Decline Notification Function For Landlord

Test ID	012	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a landlord can decline a rental request.				
Steps	1. Login as a landlord. 2. Click on the “Notifications” tab on the top nav bar. 3. Click on “Decline” for a rental request.				
Expected	The rental request will be declined and the landlord and the rental request will be put into the past requests page.				
Date-Result	07/05/2024 - Passed.				

6.13 See The Profiles Of Contract

Test ID	013	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a landlord can see the profiles of contracts				
Steps	1. Login as a landlord. 2. Click on the “Current Requests” tab on the top nav bar. 3. Click on the “See Profiles” button in a request.				
Expected	A new page will show the tenant and real estate agent corresponding to the clicked request.				
Date-Result	07/05/2024 - Passed.				

6.14 Display Past Requests For Landlord

Test ID	014	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a landlord can see the past requests.				
Steps	1. Login as a landlord. 2. Click on the “Past Requests” tab on the top nav bar.				
Expected	A new page will show past rental requests with the associated tenant and real estate agent.				

Date-Result	07/05/2024 - Passed.

6.15 Display Current Requests For Landlord

Test ID	015	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a landlord can display the current requests list in their page.				
Steps	1. Login as a landlord. 2. Click on the “Current Requests” tab on the top nav bar.				
Expected	A new page will show the current requests related to the landlord user.				
Date-Result	07/05/2024 - Passed.				

6.16 End Operation For A Contract

Test ID	016	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a landlord end an operation for a contract.				
Steps	1. Login as a landlord. 2. Click on the “Current Requests” tab on the top nav bar. 3. Click on the “End Operation” button on any rental request.				
Expected	The operation will end and the request will drop to the past requests page.				
Date-Result	07/05/2024 - Passed.				

6.17 Upload Contract Function

Test ID	017	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a landlord can upload a contract for a rental request and officially verify the rental request by doing so.				
Steps	1. Login as a landlord. 2. Click on the “Current Requests” tab on the top nav bar. 3. Click on the “Upload Contract” button on any rental request.				
Expected	The operation will be completed with the contract being uploaded.				
Date-Result	07/05/2024 - Passed.				

6.18 Display Rental Notification For Real Estate Agents

Test ID	018	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a real estate agent can view rental notifications.				
Steps	1. Login as a real estate agent. 2. Click on the "Notifications" tab on the top nav bar.				
Expected	A page will show the rental notifications for a real estate agent user, having decline and accept buttons.				
Date-Result	07/05/2024 - Passed.				

6.19 Accept Rental Notification For Real Estate Agents

Test ID	019	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a real estate agent can accept a rental notification.				
Steps	1. Login as a real estate agent. 2. Click on the "Notifications" tab on the top nav bar. 3. Click on "Accept" for a rental notification				
Expected	The accepted rental notification will move to the "Current Requests" tab accordingly.				
Date-Result	07/05/2024 - Passed.				

6.20 Decline Rental Notification For Real Estate Agents

Test ID	020	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a real estate agent can decline a rental notification.				
Steps	1. Login as a real estate agent. 2. Click on the "Notifications" tab on the top nav bar. 3. Click on "Decline" for a rental notification				
Expected	The accepted rental notification will move to the "Past Requests" tab accordingly.				
Date-Result	07/05/2024 - Passed.				

6.21 Display Past Requests For Real Estate Agents

Test ID	021	Category	Functional	Severity	Critical
Objective	This test case verifies if a real estate agent can view past requests.				
Steps	1. Login as a real estate agent. 2. Click on the "Past Requests" tab on the top nav bar.				
Expected	The user will be able to see past requests.				
Date-Result	07/05/2024 - Passed.				

6.22 Display Current Requests For Real Estate Agents

Test ID	022	Category	Functional	Severity	Critical
Objective	This test case verifies if a real estate agent can view current requests.				
Steps	1. Login as a real estate agent. 2. Click on the "Current Requests" tab on the top nav bar.				
Expected	The user will be able to see current requests.				
Date-Result	07/05/2024 - Passed.				

6.23 Display Houses In The Real Estate Agent's Location

Test ID	023	Category	Functional	Severity	Critical
Objective	This test case verifies if a real estate agent can view houses in his location.				
Steps	1. Login as a real estate agent. 2. Click on the "Home" tab on the top nav bar.				
Expected	The user will be able to see the houses only in his location				
Date-Result	07/05/2024 - Failed because real estate agents can display all of the houses. We did not filter the location instead we changed the feature to display all houses.				

6.24 House Condition Check Upload Report

Test ID	024	Category	Functional	Severity	Critical
Objective	This test case verifies if a real estate agent can upload a report for a house condition check.				

Steps	1. Login as a real estate agent. 2. Click on the “House Condition List” tab on the top nav bar. 3. Click on the “Upload Report” button on for a house.
Expected	The report will be uploaded and the house condition check card will disappear.
Date-Result	07/05/2024 - Passed.

6.25 Display House Condition Check Notification For Real Estate Agents

Test ID	025	**Category**	Functional	**Severity**	Critical
Objective	This test case verifies if a real estate agent can display the house condition check notifications.				
Steps	1. Login as a real estate agent. 2. Click on the “House Condition Notifications” tab on the top nav bar.				
Expected	A page will show the house condition check notifications.				
Date-Result	07/05/2024 - Passed.				

6.26 Accept House Condition Check Request Function

Test ID	026	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that accepting house condition checks for real estate agents is working.				
Steps	1. Login as a real estate agent. 2. Navigate to the house condition check screen. 3. Click on the notification and see the house details. 4. Accept the request if you wish.				
Expected	The notification should be removed from the house condition check screen and displayed in the house condition list page.				
Date-Result	07/05/2024 - Passed.				

6.27 Decline House Condition Check Request Function

Test ID	027	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that the declining house condition check for real estate agent function is working.				
Steps	1. Login as a real estate agent. 2. Navigate to the house condition check screen.				

	3. Click on the notification and see the house details. 4. Decline the request if you wish.
Expected	The notification should be removed from the house condition check screen and displayed in the pa page.
Date-Result	07/05/2024 - Passed.

6.28 Display Handled Complaints Function

Test ID	028	**Category**	Functional	**Severity**	Intermediate
Objective	This test case is to verify that state officials can display the handled complaints.				
Steps	1. Login as a state official. 2. Navigate to the handled complaints screen. 3. Display all the handled complaints				
Expected	When a complaint is handled by a state official, it should be displayed in this tab.				
Date-Result	07/05/2024 - Passed.				

6.29 Display Pending Complaints Function

Test ID	029	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that state officials can display the pending complaints.				
Steps	1. Login as a state official. 2. Navigate to the pending complaints screen. 3. Display all the pending complaints				
Expected	When a user sends a complaint request such as an overview for rental increment, and if the complaint is removed from the notifications screen, it should be seen under this screen.				
Date-Result	07/05/2024 - Passed.				

6.30 Mark As Handled Function

Test ID	030	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that state officials can mark a pending complaint as handled.				
Steps	1. Login as a state official. 2. Navigate to the pending complaints screen. 3. See the house details if needed. 4. Click the “mark as handled” button to flag this complaint as a resolved				

	one.
Expected	When the state official marked a complaint as handled, it should be removed from this page to “Handled Complaints” screen.
Date-Result	07/05/2024 - Failed because there were inconsistencies between backend and frontend. Due to such inconsistencies we could not make it on time.

6.31 Display Approved Contracts Function

Test ID	031	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that state officials can display the approved rental contracts.				
Steps	1. Login as a state official. 2. Navigate to the approved contract screen. 3. Display the whole list of approved contracts.				
Expected	When the state official navigates to this screen, he/she should be able to see all of the approved contracts.				
Date-Result	07/05/2024 - Passed.				

6.32 Download Contracts Function

Test ID	032	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that state officials can download the rental contracts in both approved contracts page and contracts page.				
Steps	1. Login as a state official. 2. Navigate to the approved contracts or contracts screen. 3. Download the contract by clicking the “Download Contract” button.				
Expected	When the state official navigates to one of these screens, he/she should be able to download the contract whenever they desire. The contracts should stay on that page after downloading them.				
Date-Result	07/05/2024 - Passed.				

6.33 Display Uploaded Contracts Function

Test ID	033	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that state officials can see the contracts uploaded by tenants.				
Steps	1. Login as a state official.				

	2. Navigate to the contracts screen. 3. Display the uploaded contracts to decline or accept.
Expected	When the state official navigates to the contracts screen, he/she should be able to see the whole list of contracts.
Date-Result	07/05/2024 - Passed.

6.34 Accept Contract Function

Test ID	034	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that state officials can accept the contracts uploaded by tenants.				
Steps	1. Login as a state official. 2. Navigate to the contracts screen. 3. Accept uploaded contracts.				
Expected	When a state official accepts the contract, the notification should disappear and be displayed under the “Approved Contracts” page.				
Date-Result	07/05/2024 - Failed because there were inconsistencies between backend and frontend. Due to such inconsistencies we could not make it on time.				

6.35 Decline Contract Function

Test ID	035	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that state officials can decline the contracts uploaded by tenants.				
Steps	1. Login as a state official. 2. Navigate to the contracts screen. 3. Decline uploaded contracts.				
Expected	When a state official declines the contract, the notification should disappear.				
Date-Result	07/05/2024 - Failed because there were inconsistencies between backend and frontend. Due to such inconsistencies we could not make it on time.				

6.36 Display Complaint Notifications

Test ID	036	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that state officials can display the complaint sent by the				

	other users.
Steps	1. Login as a state official. 2. Navigate to the notifications screen. 3. Display the whole list of complaint notifications.
Expected	When the state official opens the notifications page, she/he should be able to see the notifications until they click the “Handle” button. If all the notifications’ “Handle” button is clicked, the page should display “There are no notifications” text.
Date-Result	07/05/2024 - Passed.

6.37 Download Condition Status File Function

Test ID	037	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that state officials can download the condition status file to use it while deciding who will keep the deposit at the end of the rental operation.				
Steps	1. Login as a state official. 2. Navigate to the deposit side screen. 3. Click the “Download Condition File” button.				
Expected	When the state official clicks the “Download Condition File” button, it should download the pdf file uploaded by the tenants.				
Date-Result	07/05/2024 - Failed because there were inconsistencies between backend and frontend. Due to such inconsistencies we could not make it on time.				

6.38 Submit Who Will Keep The Deposit Function

Test ID	038	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that state officials can submit who will keep the deposit.				

Steps	1. Login as a state official. 2. Navigate to the deposit side screen. 3. After deciding who will keep the deposit money by examining the condition file, choose the deposit side by clicking the corresponding radio button and “Submit” button.
Expected	When the state official clicks the “Submit” button, the corresponding card should disappear and the decision should be seen by both the tenant and landlord.
Date-Result	07/05/2024 - Failed because there were inconsistencies between backend and frontend. Due to such inconsistencies we could not make it on time.

6.39 Create Account Function

Test ID	039	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that the admin account can create state official accounts.				
Steps	1. Login as admin. 2. Navigate to the create state official screen. 3. Fill the form to assign state offical’s attributes. 4. Upload a profile picture for the state official 5. Click the “Create Account” button to end the operation.				
Expected	When the admin creates a state official account, the account should be able to perform a state official’s job.				
Date-Result	07/05/2024 - Passed.				

6.40 Filter User Type To Display User List Function

Test ID	040	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that the admin can filter the whole user list to see a specific actor.				
Steps	1. Login as admin. 2. Navigate to the user list screen. 3. Click the corresponding radio button to display that actor’s list. 4. Click the “List User” button to see that actor’s list.				
Expected	When the admin chooses a user type and wants to display it, the screen should only display that type of user.				
Date-Result	07/05/2024 - Passed.				

6.41 Update Profile Function

Test ID	041	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that the admin can update an existing profile’s information.				

Steps	1. Login as admin. 2. Navigate to the user list screen. 3. Choose a user to change its profile information by clicking the corresponding “Update Profile” button 4. It will take the admin to a new page in which he/she can change the profile information. 5. After changing the attributes, click the “Update Account Information” button.
Expected	When the admin changes the profile information of a user, the changes could be seen by other users.
Date-Result	07/05/2024 - Passed.

6.42 Login Function

Test ID	042	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that all of the user types can login into the system.				
Steps	1. Open the login page 2. Enter your e-mail address and password 3. Click the “Login” button				
Expected	When a user enters his/her credentials, they need to be directed into the system.				
Date-Result	07/05/2024 - Passed.				

6.43 Forgot Password Function

Test ID	043	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that the users can use the forgot password function in case they cannot remember their passwords.				
Steps	1. Open the login page 2. Click “Forgot Password” button 3. Enter your e-mail address to receive mail from the system. 4. Enter the password in the mail sent by the system to login into your				

	account.
Expected	When users use the forgot password function, they need to receive mail to be able to login into the system.
Date-Result	07/05/2024 - Failed because we removed that feature.

6.44 Signup Function

Test ID	044	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that users can create landlord, real estate agent and tenant accounts.				
Steps	1. Open the signup page 2. Choose the account type by clicking the corresponding radio button. 3. Fill the form by entering the necessary information. 4. Click the “Sign Up” button to create the account.				
Expected	Users can easily create landlord, real estate agent and tenant accounts by simply filling the form and the accounts should be seen in the backend.				
Date-Result	07/05/2024 - Passed.				

6.45 Logout Function

Test ID	045	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that all of the actors should be able to logout from the system.				
Steps	1. Click the button with the profile picture that lays on the upper right corner of the page. 2. Click the “Logout” button.				
Expected	When users click the “Logout” button, they should be logged out from the system and directed to the login page.				
Date-Result	07/05/2024 - Passed.				

6.46 Publish Ad Function

Test ID	046	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that landlord actor will be able to publish ads of their houses.				
Steps	1. Login as a landlord. 2. Navigate to publish ad page 3. Upload images of the house by clicking the “Upload Image” button. 4. Fill the comprehensive form that defines the house. 5. Click the “Publish Ad” button.				

Expected	After an ad is issued by a landlord, it is expected that the ad will be displayed in the system to allow other actors to see the house details.
Date-Result	07/05/2024 - Passed.

6.47 Upload Receipt Function

Test ID	047	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that after renting a house, the system can understand the deposit is paid by the tenant.				
Steps	1. Login as a tenant. 2. Navigate to the upload receipt page. 3. If the previous procedures are done, upload the receipt that is generated by the bank. 4. Click the “Upload Receipt” button.				
Expected	When the tenant uploaded the receipt, it should display a pop-up that indicates receipt is successfully uploaded. If a problem occurs, it should be displayed as a pop-up. Also, the uploaded receipt should be seen by the state officials.				
Date-Result	07/05/2024 - Failed because we decided not to take receipt in the system.				

6.48 Update Ad Function

Test ID	048	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that a published ad can be edited by the landlord if desired.				
Steps	1. Login as a landlord. 2. Navigate to the my ads page. 3. Click the “Edit Ad” button. 4. Change the parts of the ad that you want to edit. 5. Click the “Save Changes” button.				

Expected	The landlord should receive a pop-up that indicates changes are successfully saved after submitting the changes. When a landlord edits an ad, the changes should be displayed for all of the other actors.
Date-Result	07/05/2024 - Passed.

6.49 Display Rental Houses Of The Landlord Function

Test ID	049	**Category**	Functional	**Severity**	Critical
Objective	This test case is to verify that the landlord can see the list of his/her published rental houses.				
Steps	1. Login as a landlord 2. Navigate to the my houses page. 3. Display the list of the rental houses published by that specific landlord.				
Expected	When a landlord goes to my houses screen, he/she should be able to see the houses published.				
Date-Result	07/05/2024 - Passed.				

6.50 System Notification For Rental Period Extension Function

Test ID	050	**Category**	Functional	**Severity**	Low
Objective	This test case is to verify that when the contract validity period becomes under a threshold value (7 days), the system sends an e-mail to both tenant and landlord.				
Steps	1. Create a rental operation by applying the procedure. 2. Manually decrease the validity period under the threshold value. 3. Check the mailboxes of the both tenant and landlord actors.				
Expected	After the remaining time for a contract has become less than the threshold value, the system should send an email for both tenant and landlord actors as a reminder.				
Date-Result	07/05/2024 - Failed because we discarded displaying the remaining time function.				

	So, that feature automatically removed.

6.51 Loading Time Of The Images

Test ID	051	**Category**	Non-Function al	**Severity**	Low
Objective	This test case is to verify that a published ad can be edited by the landlord if desired.				
Steps	1. Login as a tenant. 2. Go to the home page where the ads are displayed. 3. Check the loading time of the images in the ad. 4. If the time was too long (more than 5 seconds), try to optimize the loading function.				
Expected	The loading time should not bore the user.				
Date-Result	07/05/2024 - Failed because we could not focus on increasing user experience.				

6.52 Login Time

Test ID	052	**Category**	Non-Function al	**Severity**	Low
Objective	This test case is to verify that an actor is logging into the system under a reasonable amount of time.				
Steps	1. Connect the log in function to the front-end 2. Check the required time for logging in. 3. If the time was too long (more than 5 seconds), try to optimize the logging function				
Expected	Actors should be able to log into the system without waiting too long.				
Date-Result	07/05/2024 - Failed because we could not focus on increasing user experience.				

Since our test cases mostly consist of functional operations, their severity level is generally critical. We wanted to give priority to the functional features because this is how we understand the system is working or not. However, we have some non functional test cases to optimize the user experience.

7. Maintenance Plan and Details

We wrote a clear code with descriptive comments to be able to handle the maintenance easier. Currently, we are not planning to do any maintenance on the application as we do not have a customer that will buy our application. We will try to find a suitable customer and arrange a meeting with them to discuss how exactly our maintenance will go. If we find such a customer, we are aiming to focus on the following actions for our maintenance.

1. **Scheduled Updates**: We are planning to schedule regular updates to address the needs of our customers and end users that emerge after deployment. We are also aiming to enhance the security of our application with these updates to ensure the safety of our users and their data.
2. **Performance Monitoring:** We will continuously check response time, server uptime, etc. of our application to take proactive measures in case of any deviation from the expected result.
3. **Security Audits:** We will regularly perform security audits to see if there are any vulnerabilities in our application. Performing such audits are very important because in case of any security vulnerabilities, people might lose their houses, money, etc.
4. **User Support:** A dedicated user support team will be available to resolve the technical problems that might arise in the application.
5. **Backup and Recovery:** We will replicate and keep at least one backup of each data so that we can recover them in case of a cyber attack or technical problem. We will prevent any data loss that might occur in such events.
6. **Scalability Planning:** As the user and data size we will scale up the backup services and the database respectively. This will allow more users to be active without any performance issues.
7. **Documentation Updates:** We will update the documentation with each change so that finding anything on the source code or the application to remain easy at all times for us, future developers/administrators, customers and end users.

8. Other Project Elements

8.1 Consideration of Various Factors in Engineering Design

In this section the possible effects of various factors on the engineering design process of the project will be discussed in detail.

8.1.1 Constraints

We considered the following constraints while regarding the functional and nonfunctional requirements of SecuRent.

Legal Considerations: Legal considerations are crucial for the success and compliance of the SecuRent project, given its involvement in the house rental process and the handling of sensitive data such as TCK. We need to protect the user's data and get consent from them regarding the collection, processing, and storage of their personal information. We should also comply with the Rental Regulations of the government. Henceforth, we need to carefully follow KVKK and prescribe a privacy policy regarding the control of data within the SecuRent application.

Usability: We should provide a simple user interface that is not complicated. We need to prioritize a clean and intuitive user interface. We should ensure that common tasks are easily achievable and that users can quickly understand and navigate through the application. We are aiming to meet a user and his/her goal in at most 10 clicks. Hence, we are developing our application to satisfy intuitiveness and simpleness.

Maintenance: We need to design the application with a modular structure, separating different components or modules such as Tenant, Landlord, Government etc. We should also provide comprehensive documentation for the application's architecture, codebase, and data structures, and database so that our project can be maintained easily (as mentioned in the NFR section in detail as well).

Accessibility: Accessibility is a critical aspect of the SecuRent project, ensuring that the application is usable by individuals with diverse abilities, including those with disabilities. Considerations for accessibility in the SecuRent application are essential to provide an inclusive user experience. Our project has potential users from all regions of Turkey. Moreover, people from different ethnic backgrounds, gender, and age will use the SecuRent, without facing any discrimination.

Constructability: Constructability in the context of the SecuRent refers to the ease with which the application can be built and implemented successfully. A focus on constructability ensures that the development process is efficient, cost-effective, and aligns with the project's goals. We need to provide incremental development, breaking the project into manageable phases or iterations. This allows for continuous testing, validation, and adjustment of features as the development progresses. Last but not least, our unit tests will help us to build an efficient and a constructible application.

Cost: SecuRent will use AWS services to manifest its functionality in the public networks. Thus, the cost of the cloud services should be constantly monitored. Moreover, AWS services need precise configurations to prevent us from paying for more than what we need. To handle the process, AWS Cloudwatch and AWS Cost Optimizer services can be used to give us insights about the cost of the services SecuRent uses.

In the following table, the effect levels range from 0 to 10 (0 having no effect, 1 having the least and 10 having the greatest effect).

	Effect level	Effect
Public health	0	Public Health does not affect any part of the solutions since the solutions are not intended for any problems in public health.
Public safety	10	Public Safety has a great effect because one of the primary aims of SecuRent is to ensure the safety of rental ads, rent amounts, deposits, and landlord-tenant contracts for houses.
Public welfare	10	Public Welfare has a great effect on the solutions since the aim of the project is to provide a service to the tenants, landlords and government, which is for public welfare specifically.
Global factors	4	Global Factors do not have a big impact on the solutions because the problems that are to be resolved or reduced by Securent are not global problems. Yet, it is for public advantage, so it can be scaled as an effect level of 4.
Cultural factors	0	Cultural Factors have no effect on the way the solutions are designed because SecuRent does not aim to provide any measures/advantages on the cultural factors: it will function for the benefit of the public in general.
Social factors	6	Social Factors have an impact on the solutions, but they are second to public safety and public welfare. The solutions for SecuRent are affected by the social factors in the way that it aims to increase the trust and work ethics between landlords, tenants and real estate agents.
Environmental factors	0	Environmental Factors have no effect on the way the solutions for SecuRent are shaped. SecuRent has no concerns (or will not deal with the issues) of environmental problems.

Economic factors	6	Economic factors have some impact on the solutions design but just like social factors, it comes after public safety and public welfare. Economic factors sparked the idea that the government should hold the deposits of rental houses for them to make a profit and add up extra income.

8.1.2 Standards

UML 2.5.1 is used through the draw.io platform to model the high level architecture of the system.

- Architectural Patterns: Module style is used to model the class diagram architecture. Specifically following styles are used:
    - Decomposition Style: Modules are separated into submodules to cover functionalities within the parent module.
    - Uses Style: Different modules connect to each other with the user relationship as particular modules require the functionality of other modules.
    - Layered Style: Different layers allow different functionalities to be separate. For instance UI and core module are distinct layers and interact with each other through an intermediary API layer.
- Software Development Standards: The software development of the SecuRent should be applied by following the reports and aiming for the actual final product. The standards of the development phase should be complied which involve the language and technology choices and predefined development processes.
- Usability Standards: The User Interface (UI) and User Experience (UX) of the SecuRent should be considered so that it follows the usability standards that are user friendly, intuitive, and easy to navigate in the system.
- Quality Assurance Standards: SecuRent should comply with the quality assurance standards that are satisfying the requirements of the project. Hence, we test the application with 50 test cases currently. In the subsequent iterations, the number of test cases will increase to cover each aspect of the application.

8.2 Ethics and Professional Responsibilities

Ethical Considerations: Our project deals with the data of the landlords, tenants and houses. We are collecting them and showing them to the user. Dealing with those private data may create a privacy issue. In our project, we prioritized the careful handling of data privacy and consistently supported the principles of respecting individuals' privacy rights. Here's how we ensured that data privacy was a top priority throughout the project:

- **Data Minimization:** We minimized the collection and storage of personal data to only what was necessary for the functionality of the project. By limiting the scope of

data collection, we reduced the potential risk of unauthorized access or misuse of sensitive information. We do not require additional unnecessary private information from users. We just ask for information about the house such as number of rooms, flat number, heating type, number of bathrooms, existence of balcony, existence of furniture, and rent price.
- **Access Controls:** We implemented strict access controls to ensure that only authorized individuals had access to sensitive data. This included regular audits of user access logs to detect and prevent unauthorized access attempts.
- **Compliance with Regulations:** We conducted thorough legal research to understand the regulatory landscape governing the rental housing market. This included familiarizing ourselves with relevant local, national, and international laws, regulations, and industry standards. Furthermore, we did not add any feature that can violate any regulations or law.

Ethical AI usage: In our project, we followed ethical principles in the usage of AI and the internet to get knowledge and resources. We ensured that our usage of AI and internet resources was ethical and respectful by following these principles:
- **Originality and Integrity:** We verify that all the code developed for our project was original and created by our team members. We did not engage in any form of plagiarism, theft, or unauthorized use of code from external sources. Our commitment to integrity and originality highlight our respect for intellectual property rights and ethical conduct in software development.
- **Transparency:** We maintained transparency and accountability throughout the development process by documenting our sources, methodologies, and decision-making processes. As a result of this, how AI and internet resources were utilized in our project and integrity of our work can be observed. Our whole progress and code are available at our GitHub repository.

8.3 Teamworks

8.3.1 Contributing and Functioning Effectively on the Team

All significant decisions are made collectively after group discussions. Each project member actively participates in report writing sessions, and every team member is easily accessible and communicates effectively.

Arda Yıldız: One of the three backend developers of the project. Together with Cengizhan and Yusuf, built the server-side logic using Java Spring Boot. Implemented the Tenant, House, RealEstateAgent, RealEstateAgentOperations, RentRequest, and Contract entities along with their services, repositories and controllers. Created the database logic using

PostgreSQL and Spring Data JPA. Defined the REST API endpoints with other backend developers. Configured and began testing the API endpoints using Postman. Implemented the security of the application in terms of authentication and authorization. Implemented Axios to connect frontend and backend sides of the project. Deployed and configured the application in the AWS cloud environment. Set up the Amazon EC2 instances to run the backend, connected the database to Amazon RDS, configured/hosted the frontend of the application in the S3 bucket in the AWS cloud environment.

Cengizhan Terzioğlu: One of the backend developers of the project. Created the backend of the project with Spring Boot. Created many model classes for the database objects. Implemented Landlord, House and RentalContract entities with their services, repositories and controllers. Done modifications to services and controllers related to Tenant and RealEstateAgent. Fixed bugs in the backend. Configured the application in the AWS environment.

Yusuf Şenyüz: One of the backend developers of the project. Implemented services of the Government and House classes in the backend. Implemented, GovernmentController, GovernmentRepository, HouseRepository. Updated and changed RealEstateAgent class specifically its controller. Helped Landlord and Tenant classes as well. Created some of the interface, model and enums files to make it easier the whole implementation process. Helped the frontend team during the connecting backend and frontend by testing API endpoints.

Yunus Eren Türkeri: Worked as a team of 2 members for the frontend. Developed half of the user interfaces and specifically worked on the tenant and landlord actors and partially worked on the admin user interfaces (3 actors out of 5). Worked on connecting the frontend to backend using Axios library. Worked on adapting some UIs to suit the endpoints in the backend side. Moreover, I created 25 test cases for this report. Was responsible for running half of the test cases.

Mustafa Kaan Koç: Other member of the frontend team. Worked on developing half of the user interfaces. Focused on real estate agents and state official actors, and worked some parts of the admin actor. Worked on completing missing UI parts and removing uınnecessary components. Connected backend with frontend using axios. Worked on adapting some UIs to suit the endpoints in the backend side. Created around 30 test cases and was responsible for creating the test case tables in this report. Also, was responsible for running half of the test cases.

8.3.2 Helping Creating a Collaborative and Inclusive Environment

We utilized Jira to allocate roles and tasks within the project. Moreover, we assigned stories to different members of the backend and frontend teams to include everyone in the development process and sharing responsibilities. For project-related communications, we primarily used a WhatsApp group. However, for live communication needs and especially for iteration planning meetings, we used the Discord platform to communicate.

8.3.3 Taking Lead Role and Sharing Leadership on the Team

Collaboration is the key to success in the SecuRent project. To ensure that everyone collaborates, we have set up a Jira project management page. Next, we assigned each group member a task, giving responsibility to everyone. Thus, instead of merely talking about the distribution of the tasks, we are writing them down and assigning the tasks explicitly to everyone. Hence, everyone feels involved as they are taking responsibilities (through tasks), based on their prior experiences or willingness to commit. Moreover, we decided to have leaders for the front end and the back end side, to ensure that different people share the role of leadership in different aspects. For instance, Yunus is the leader of the front end team, while Cengizhan is the leader of the back end team. In addition, Yusuf is responsible for implementing the data access layer as well as leading the design process. Arda is the SCRUM Master due to his prior experience on Agile and SCRUM in the company he works. Kaan is the product owner as he has connections in the domain and has formulated the very idea of the SecuRent application. Consequently, each team member has assigned a leadership role in a particular field. With the sharing of leadership in different aspects of the project, we aim to increase the efficiency and joy as everyone volunteered to take leadership according to their skills and interests.

8.3.4 Meeting Objectives

In our project, we specified the following objectives at the beginning of the year:

- Filters feature that enables users to filter the houses by typing district, floor, number of rooms, direction information.
- Searching feature that enables users to search houses.
- User Management and Authentication features.
- Effective usage of the database for data related to the houses and contracts. A relational database and SQL queries are used to make calls to the database.
- User interface that enables clients to interact with the system.
- A feature that enables landlords to publish ads about their houses.
- A feature that enables tenants to see available ads that are published.
- A feature that tenants can make requests for renting available houses.
- A feature that landlords can see notifications about their published ad.
- A feature that the government can collect deposits from tenants, and give it back when necessary.
- A feature that the government can assign exprets to the houses to determine rent prices.

We met most of our objectives. Features about tenants, and landlords are done completely. User Management and Authentication features, effective database usage and effective user interface objectives are also met. However, due to time constraint, we could not finish the implementation of filtering and searching house those features are missing. In addition, features about the government such as collecting deposits and giving it back are not completed because when we discuss those ideas with the officials from the ministry of trade, they did not support our idea and project in general as a stakeholder so we did not prioritize those features. Moreover, we planned to add an image feature such that each house would have an image of itself but we faced a problem with that objective as well.

8.4 New Knowledge Acquired and Applied

Decision for choosing the tech stack involved team members' familiarity with the tool. Although both the frontend and backend team chose development tools, languages or development environments with regards to their familiarity, everyone has acquired new knowledge. For the frontend team, implementing Axios to establish communication between UI components and the database was a new knowledge. We set learning goals during our sprints for the frontend team members. Henceforth, after completing backend functionalities incrementally, the frontend team began implementing Axios. The frontend team went step by step to connect the UI pages and backend functionalities, actively communicating with the backend team in the process. On the other hand, the backend team learned how to configure the database via application.properties file. Moreover, they learned how to map the database tables via Hibernate. In addition, they learned how to deploy an application to the AWS cloud environment. With both teams achieving their learning goals and acquiring new knowledge, we as a team managed to release a functioning application running on the cloud environment. Apart from learning technologies, we learned how to cooperate and communicate actively. We communicated with each other almost everyday and learned how to work together effectively. Sometimes we worked asynchronously, sometimes synchronously, we respected each other's priorities and availability times (due to exams or other projects). Thus, we experienced working as a team from the first hand with our completion project.

9. Conclusion and Future Work

After successfully implementing the SecuRent project, what lies ahead is finding a potential customer. The biggest mystery of this project has always been a customer. In that regard, we as the SecuRent team went to the Ministry of Environment, Urbanisation and Climate Change with the hopes of earning a sponsorship for our project. However, since we met with feasibility concerns and the problem of acquiring real data from E-devlet, our project is not welcomed. Thus, we decided to complete our project and present it as a Proof of Concept work. Therefore, we still need to find prospective customers for our project.

10. Glossary

Deposit: Amount of money paid at the beginning of the renting period. In principle, tenants are paying the deposit to the landlords. Throughout the duration of the contract, the landlord keeps the deposit. When the contract expires, the landlord keeps the deposit if the house is damaged. Otherwise, the landlord returns the deposit to the tenant.

11. References

[1] *trthaber.com.* [Online]. Available: https://www.trthaber.com/haber/ekonomi/kiracinin-kontrat-suresi-dolmadan-kira-artisi-yapilabilir-mi-656837.html. [Accessed: Dec. 8, 2023].

[2] *ntv.com.tr.* [Online]. Available: https://ntv.com.tr/galeri/ntvpara/ekim-ayi-kira-artis-orani-ne-kadar-oldu-2023-ekim-ki

[3] *endeksa.com*. [Online]. Available: https://www.endeksa.com/tr/. [Accessed: Nov. 15, 2023].

[4] *mydeposits.co.uk*. [Online]. Available: https://www.mydeposits.co.uk/join-now/. [Accessed: Nov. 16, 2023].

[5] *amazon.com.* [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html. [Accessed: Dec. 8, 2023].