REAL ESTATE Project

YUSUF ÖMER TURSUN

UMUT KURULUK

EFE EROL

- JUNE 7 2024 -

Group No | 16

- CEN 303 -
SOFTWARE ENGINEERING

1. INTRODUCTION

1.1 Purpose and Scope

1.1.1 | The Purpose of the Project

- The purpose of our project is to establish a robust and efficient system that seamlessly integrates advanced database management systems with a user-friendly web interface. This initiative aims to revolutionize the real estate industry by providing a comprehensive and streamlined solution for managing property listings, transactions, and customer interactions.

1.1.2 | Scope of the Project

- This project has been specifically designed to meet the various needs of both property professionals and regular users. Accessible to a wide audience, the system efficiently manages property data, facilitates transparent transactions and provides a user-friendly experience. The system prioritises functionality and simplicity to ensure ease of use for all individuals interacting with the platform.

1.2 Goals and Success Criteria

Goals

- To create a centralised and scalable property database capable of efficiently storing and managing a variety of property data.
- To develop an intuitive web platform accessible to both property professionals and general users, promoting a seamless property search and transaction experience.
- Increase user engagement and satisfaction by implementing advanced search and filtering functionalities for property listings.
- Facilitate secure and transparent property transactions through the integration of a robust transaction management system.
- Implement a user authentication and authorisation system to ensure data security and provide role-based access control.
- Optimise system performance and scalability to accommodate increasing data volumes and user traffic.
- Perform extensive testing and quality assurance to deliver a stable and error-free property database and web platform.

Success Criteria

- Successful deployment of a fully functional and responsive web platform accessible from a variety of devices.
- Ensuring a high level of user satisfaction as measured by user feedback and engagement metrics.
- Implementation of an efficient and user-friendly property search and filtering system, resulting in reduced search times for users.
- Successful completion of property transactions through the platform, as measured by transaction completion rates.
- Implementation of a secure user authentication and authorisation system with no reported security breaches.
- System performance metrics, including response times and resource utilisation, meeting predefined benchmarks.
- Successful handling of increasing data volumes and user traffic and demonstration of system scalability.
- Completion of extensive testing phases with minimal reported issues and high levels of system stability.

1.3 Overview

1.3.1 | System Requirements

- Implement a secure user authentication system requiring a unique username and password for access to the platform which includes password encryption to enhance security.
- Implement database indexing and optimization techniques for efficient data handling.
- Optimize system performance to ensure responsiveness, even under increased user traffic.
- Advanced search and filtering functions to enable users to refine property searches based on criteria such as location, price range, amenities and property type.
- Admin can review, approve or delete listings.

1.3.2 | Member Requirements

- Each member has a unique username and password
- Users have appropriate permissions based on their role (e.g. real estate agent, buyer, seller).
- Users can provide feedback through feedback mechanisms

1.3.3 | Test Items

1.3.3.1 | Features To Be Tested

- Login/Logout functionality for different user roles (e.g., admin, real estate agent, customer).
- Database Integration
- Web Interface
- Data encryption during transmission and protection against SQL injection and other common web vulnerabilities.
- Search and Filtering: Property search functionality based on various criteria.

1.3.3.2 | Features Not To Be Tested

- Physical hardware failures
- System behavior under extreme load conditions

1.3.3.3 | Item Pass/Fail Criteria

- Security – Pass if no unauthorized access, data breaches, or security vulnerabilities.
- Database Operations – Pass if CRUD operations result in correct storage and retrieval of data.
- Web Interface – Pass if intuitive navigation, responsive design, and error-free user interactions.
- Search and Filtering – Pass if accurate property search results and proper functioning of filters.
- Functionalities – Pass if all functions perform as expected under normal usage scenarios.

1.3.3.4 | Software Risk Issues

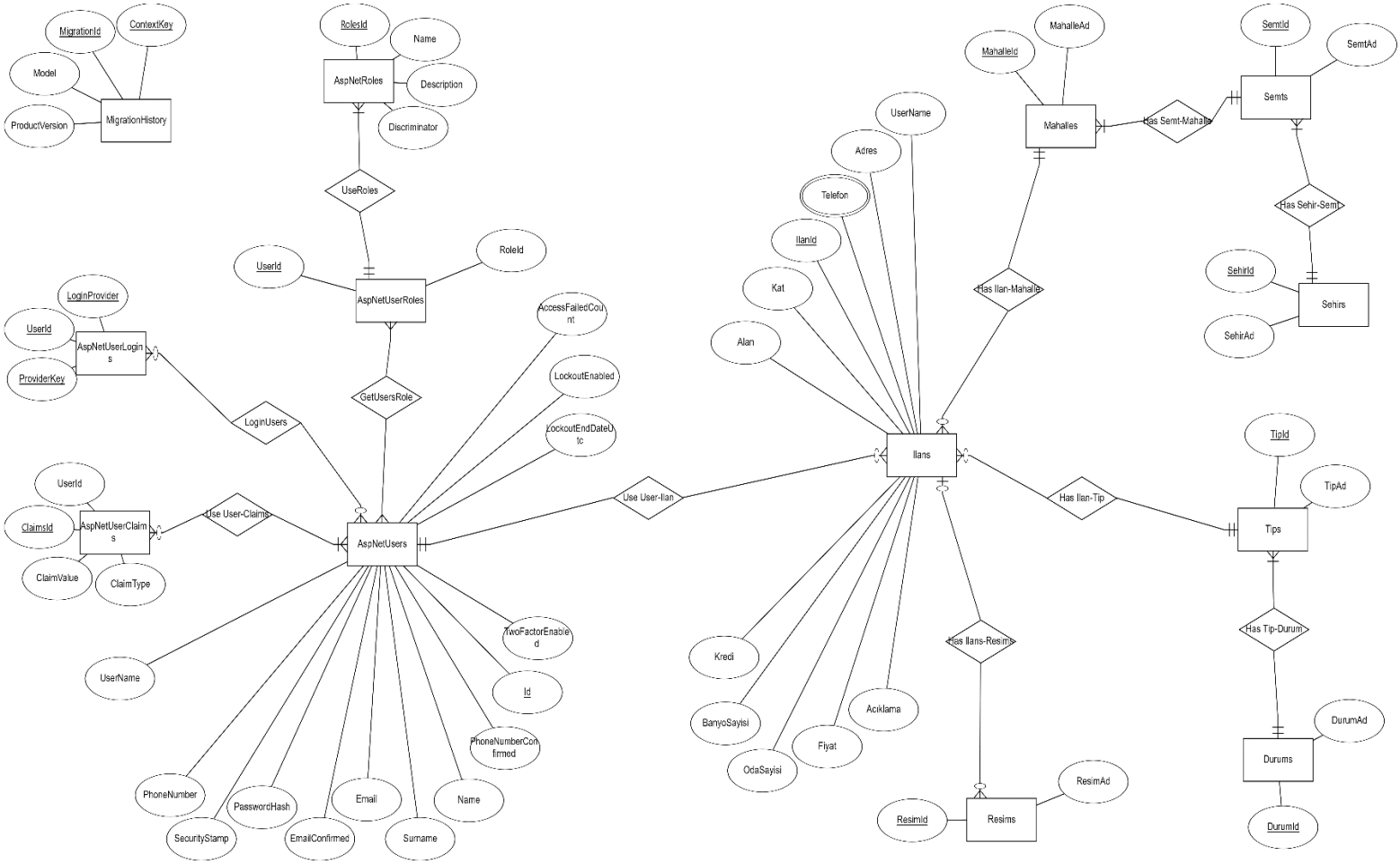
- Open Source Code
  - Risk: Dependency on external code with potential security and compatibility issues.
  - Mitigation: Regularly update and patch dependencies, security audits.

- Complexity
  - Risk: Increased chances of bugs and maintenance challenges.
  - Mitigation: Code reviews, documentation, and modularization.

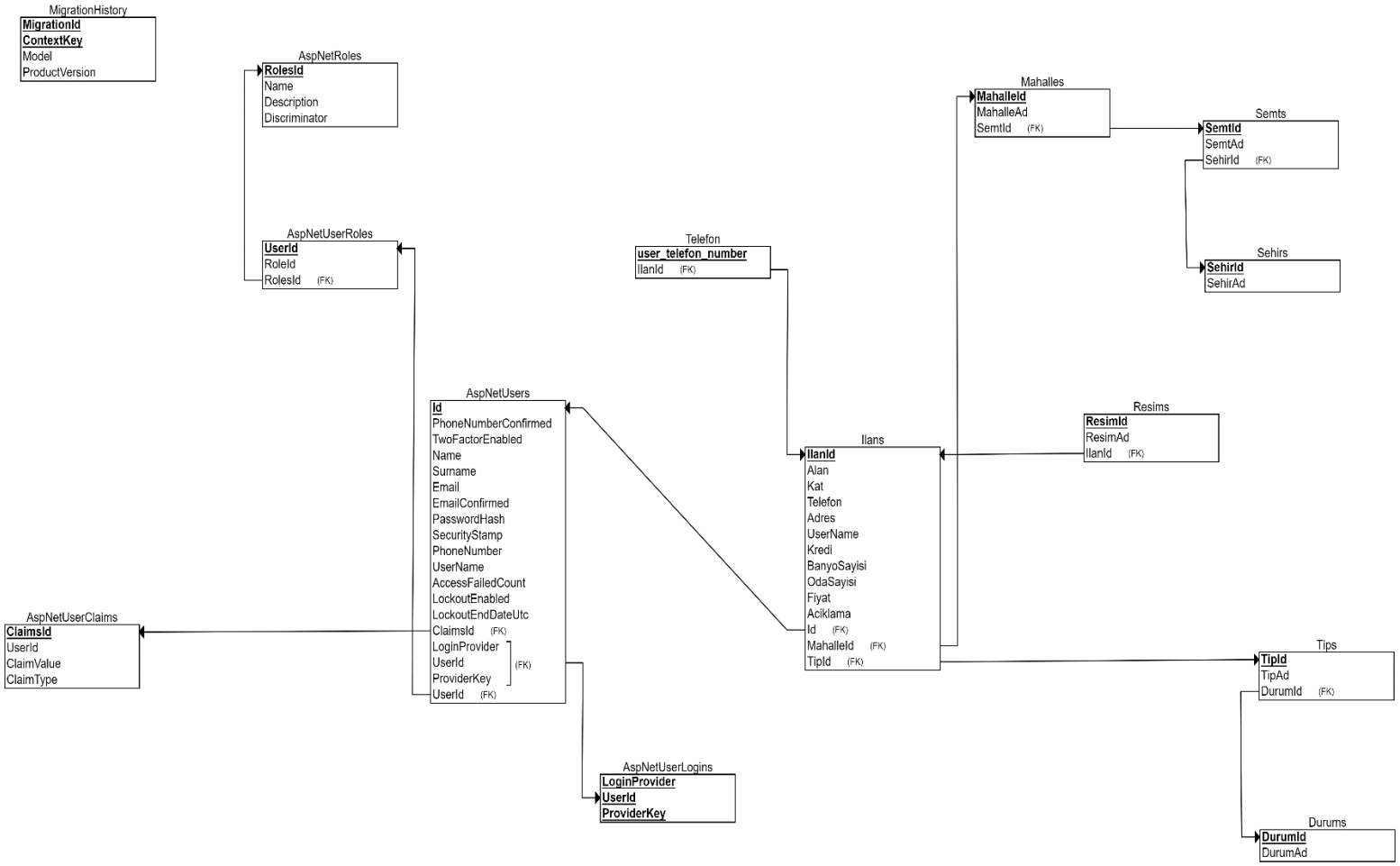
- Logic Issues
  - Risk: Incorrect processing leading to unexpected behaviors.
  - Mitigation: Thorough testing, code reviews, and continuous integration.

1.4 | Data Models

1.4.1 | E-R Diagram



1.4.2 | Relational Tables



1.5 | Software Processes and Process Models

1.5.1 | Selected 5 Software Process Models and Reasons

- ***Waterfall Model****:* The Waterfall model is a linear and sequential approach to software development, where each phase must be completed before moving on to the next. This model is not suitable for the Real Estate project due to its rigid structure. In real estate, requirements often evolve over time, and the waterfall model lacks flexibility to accommodate changes once the project has started. The inability to adapt to evolving requirements makes the Waterfall model inappropriate for a project where client needs and market trends can change rapidly.

- ***Agile Model:*** Agile is known for its flexibility and adaptability, making it more suitable than Waterfall for a real estate project. However, the Agile model might be only slightly suitable because the project may still require a certain level of upfront planning, and Agile's iterative nature might lead to challenges in managing long-term development goals. Real estate projects often involve complex business rules and regulatory requirements, and Agile's emphasis on quick iterations might not align perfectly with the comprehensive planning needs of such projects.

- ***Spiral Model:*** The Spiral model combines aspects of both waterfall and prototype models. It involves the repetition of a set of framework activities in a spiral fashion, allowing for incremental releases. While it offers some flexibility and risk management, the Spiral model might be only slightly suitable for a real estate project. Its focus on risk analysis and prototyping phases may introduce unnecessary complexity for a project where the core functionality can be well-defined without extensive prototyping.

- ***Incremental Model:*** The Incremental model involves building a system in small, manageable parts, allowing for the development of partial systems quickly. This model is fully suitable for a real estate project as it enables the development of specific functionalities in increments. Real estate projects often have diverse requirements, and incremental development allows for the gradual expansion of the system, accommodating changing needs over time. This model aligns well with the nature of real estate projects, providing flexibility and adaptability.

- ***Prototyping Model:*** The Prototyping model is particularly suitable for the Real Estate Database and Website project. This model involves the creation of a prototype (a working model of the system) early in the development process, allowing for user feedback and refinement. In real estate, where user interactions and visual appeal are crucial, prototyping enables stakeholders to visualize the system early on. This iterative approach aligns perfectly with the dynamic requirements of the real estate domain, allowing for continuous refinement based on user feedback. The Prototyping model's emphasis on user involvement, rapid iterations, and visual representation makes it the most appropriate choice for the Real Estate project.

1.5.2 | Decided Software Process Model and Reason

- The Prototyping Model proves to be the optimal choice for the Real Estate Database and Website project due to its agility in handling the dynamic requirements of the real estate domain. By rapidly creating a working prototype, this model enables stakeholders to visualize and interact with the system early in development. This iterative approach, crucial for real estate where user interfaces are paramount, allows for quick adjustments based on user feedback. The Prototyping Model fosters collaboration, ensuring the system aligns with evolving needs. Additionally, it supports early issue identification, a valuable feature for real estate projects with stringent regulatory requirements and integration needs. In summary, the Prototyping Model excels in providing visual clarity, fostering collaboration, and accommodating dynamic requirements, making it the most suitable for the Real Estate Database and Website project.

2. REQUIREMENTS

2.1 Functional Requirements

- Users (real estate agents, administrators) must be able to log in with unique credentials.
- A list of available properties for sale or rent is displayed. It is possible for users to search and filter properties by criteria (e.g. property type).
- Collect and store customer personal information (name, contact details, etc.).
- Registering new customers in the system.
- Updating information for existing customers (e.g. contact details, preferences).
- Adding house information such as number of rooms, number of bathrooms.
- Users can see the advertisements posted on the page.
- The user can place an advertisement on the site.
- The user can edit (delete or update) their own advertisement.
- The user can filter the ads on the page.
- The website is responsive and accessible on a variety of devices, including desktops, tablets and smartphones.
- Admin can edit and delete any post.

2.2 Non-Functional Requirements

2.2.1 | In Terms of Security

- Only authorized administrators have access to sensitive functionalities.
- User authentication is required with a unique username and password.
- Sensitive user-specific data such as user IDs and user passwords are encrypted during transmission and storage.

2.2.2 | In Terms of Usability

- The system should be available 24/7 to accommodate users in different time zones.
- The system should have a user-friendly and intuitive login screen.

2.2.3 | In Terms of Performance

- The system should respond instantly to user interactions.
- It has a fast and responsive user interface to enhance the user experience.
- Database queries and application logic have been optimized to minimize response times.
- With Session Cache, user-specific information can be displayed to the user with short response times.

2.2.4 | In Terms of Supportability (Maintenance)

- Implement a regular maintenance schedule for system updates and improvements.
- Tools are provided for administrators to perform routine maintenance tasks.
- Maintain version control for software.
- Maintain comprehensive documentation for the system architecture, database structure and code base to facilitate future updates and troubleshooting.

2.2.5 | In Terms of Constraints (System Constraints)

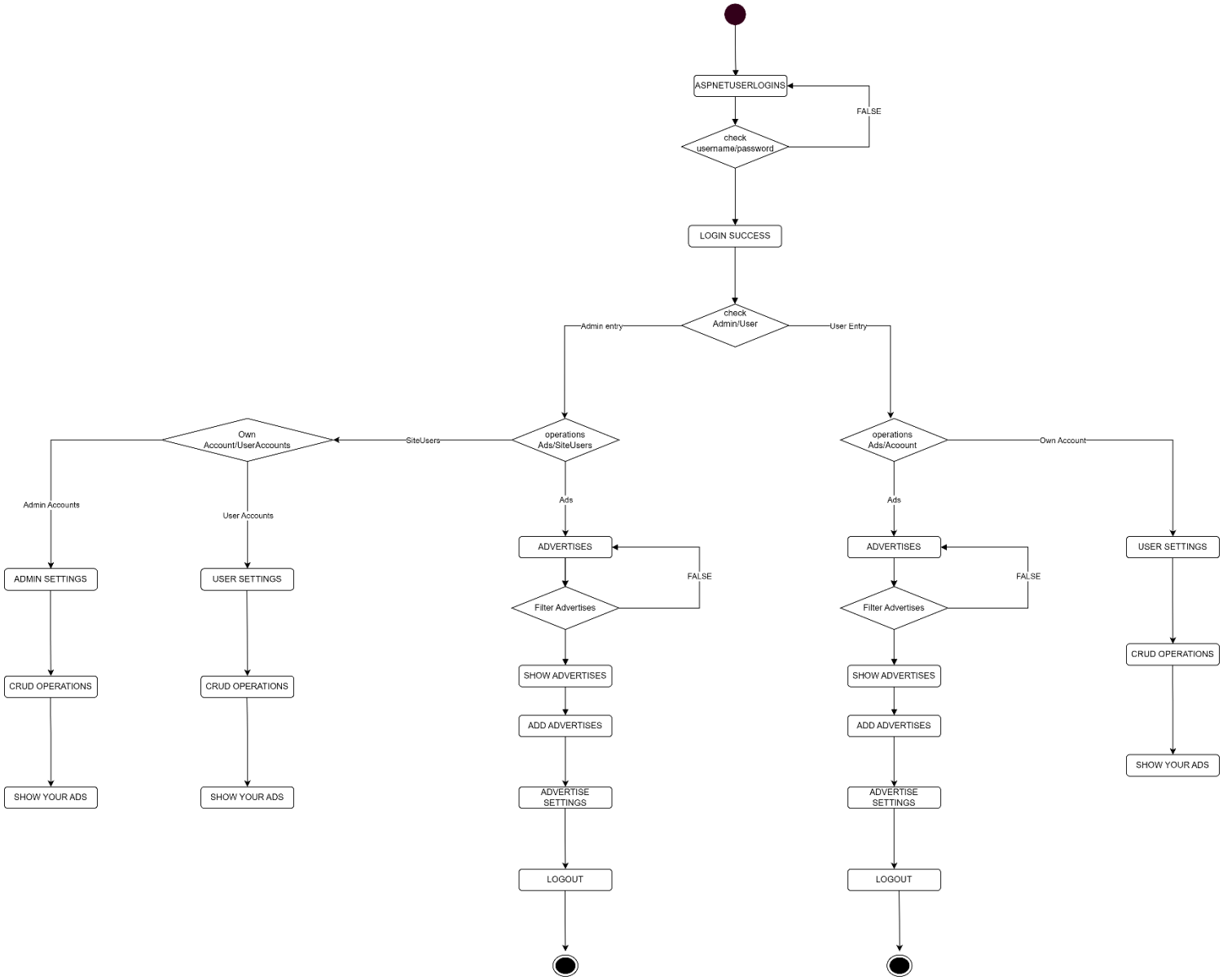
- Clearly define and enforce user roles (administrator and customer).
- Assign appropriate permissions to each role to ensure proper access control.
- Ensure that the system can handle growth without compromising performance.

3. System Modelling with UML

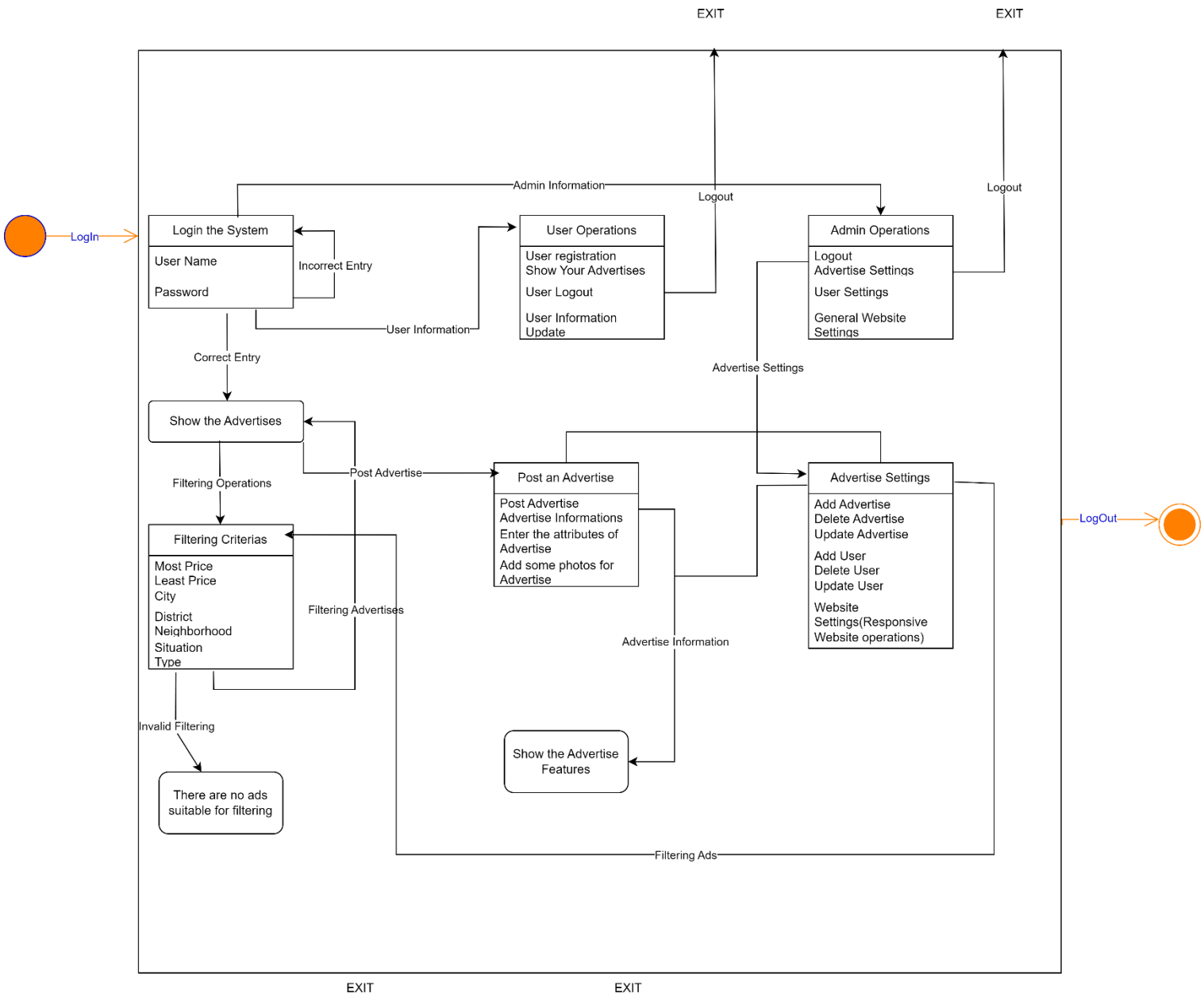
3.1 Behavioral Diagrams

3.1.1 | Use-Case Diagram

3.1.2 | Activity Diagram

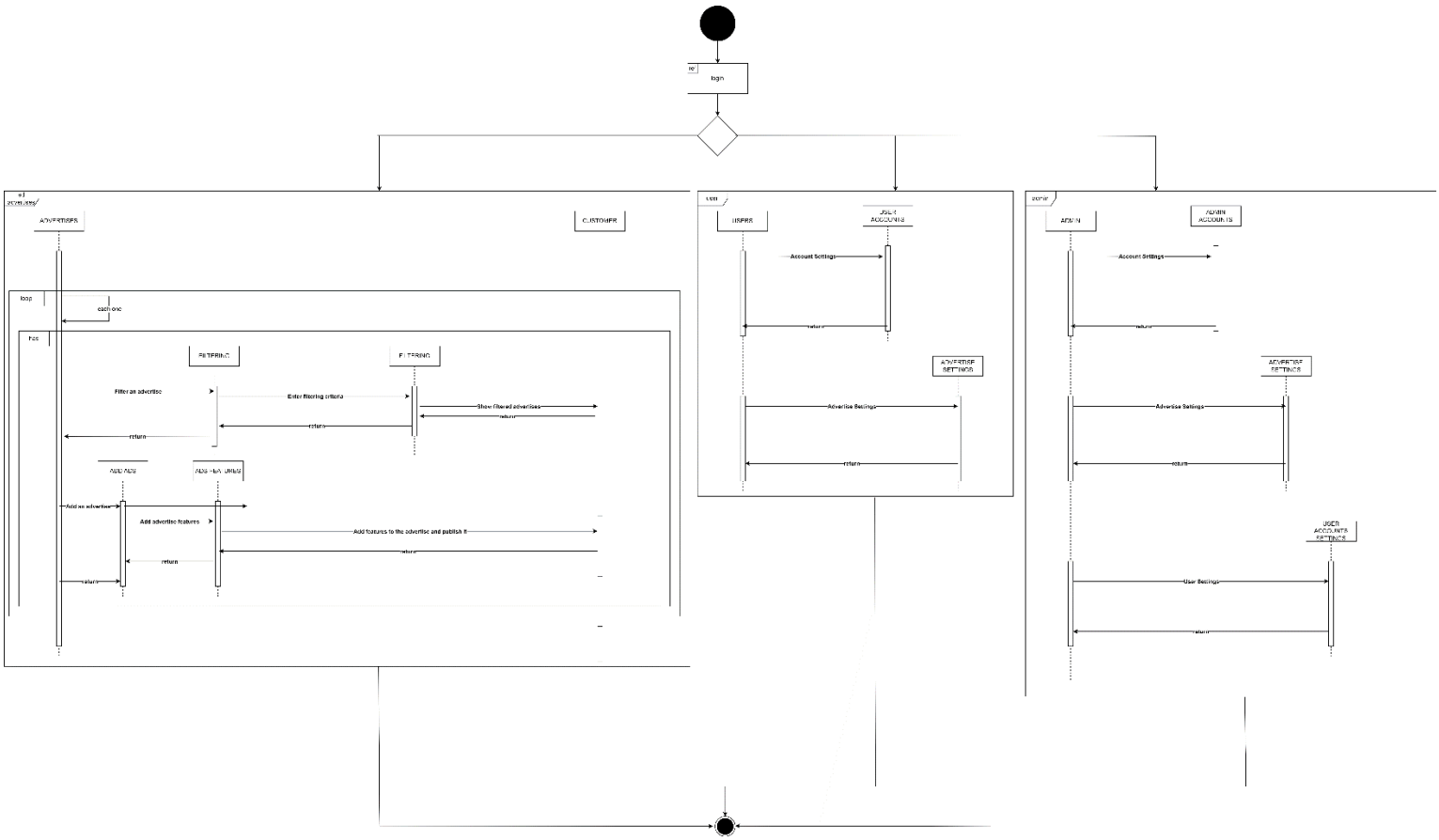


3.1.3 | State (State Machine) Diagram



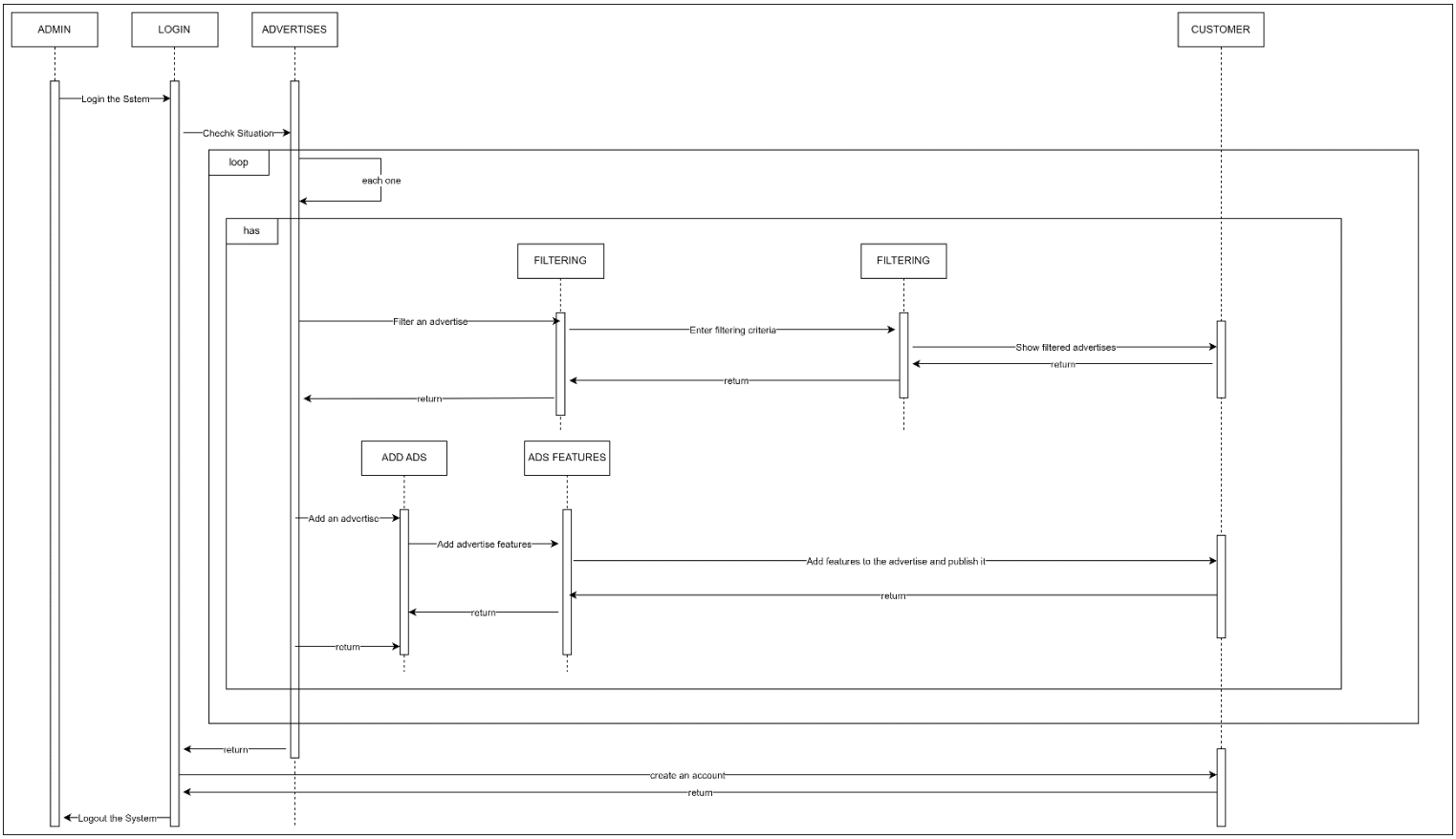
3.1.4 | Interaction Diagram

3.1.4.1 | Interaction Overview Diagram

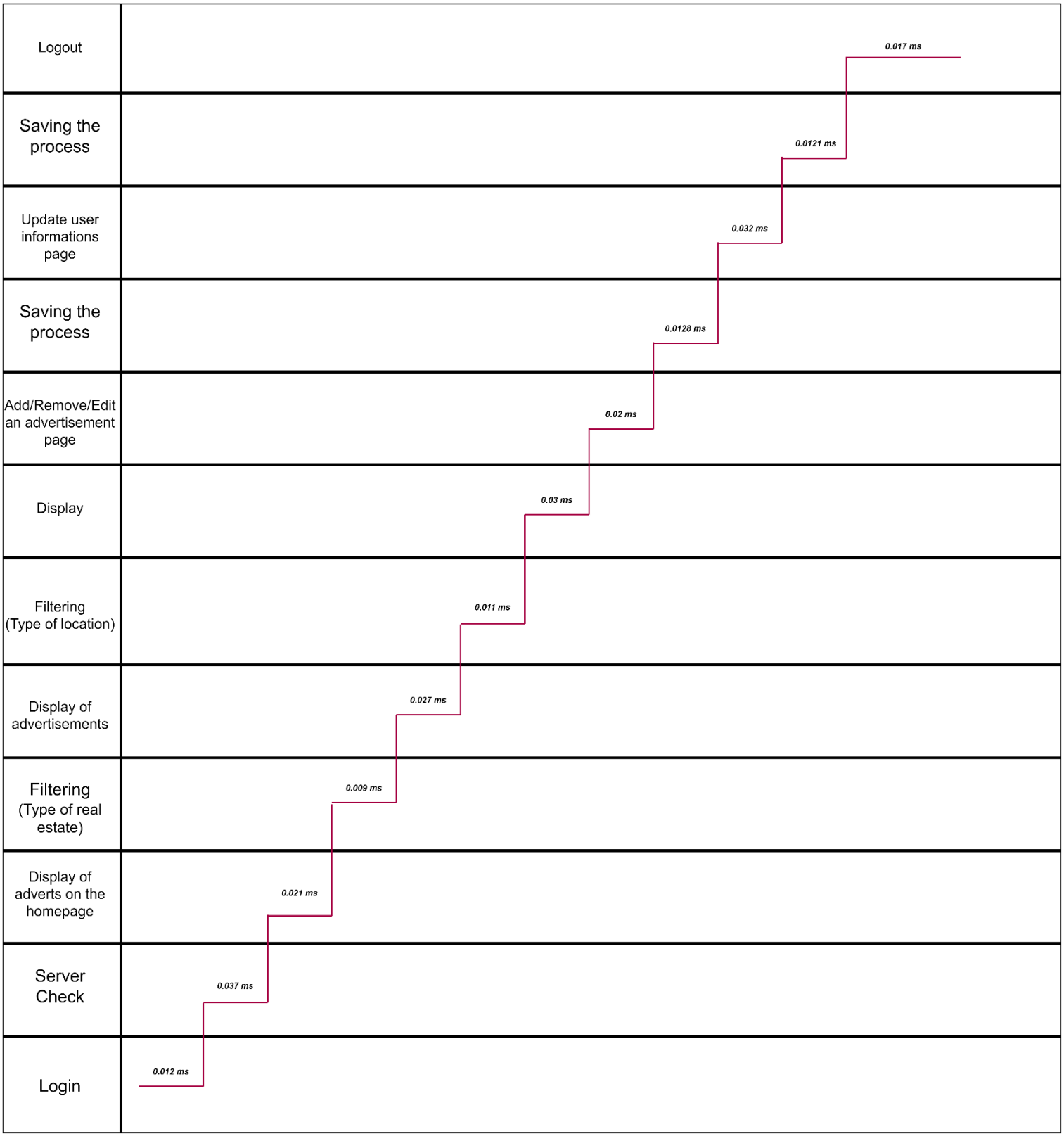


- *Bu resmin net hali, ekte paylaşılmıştır.*

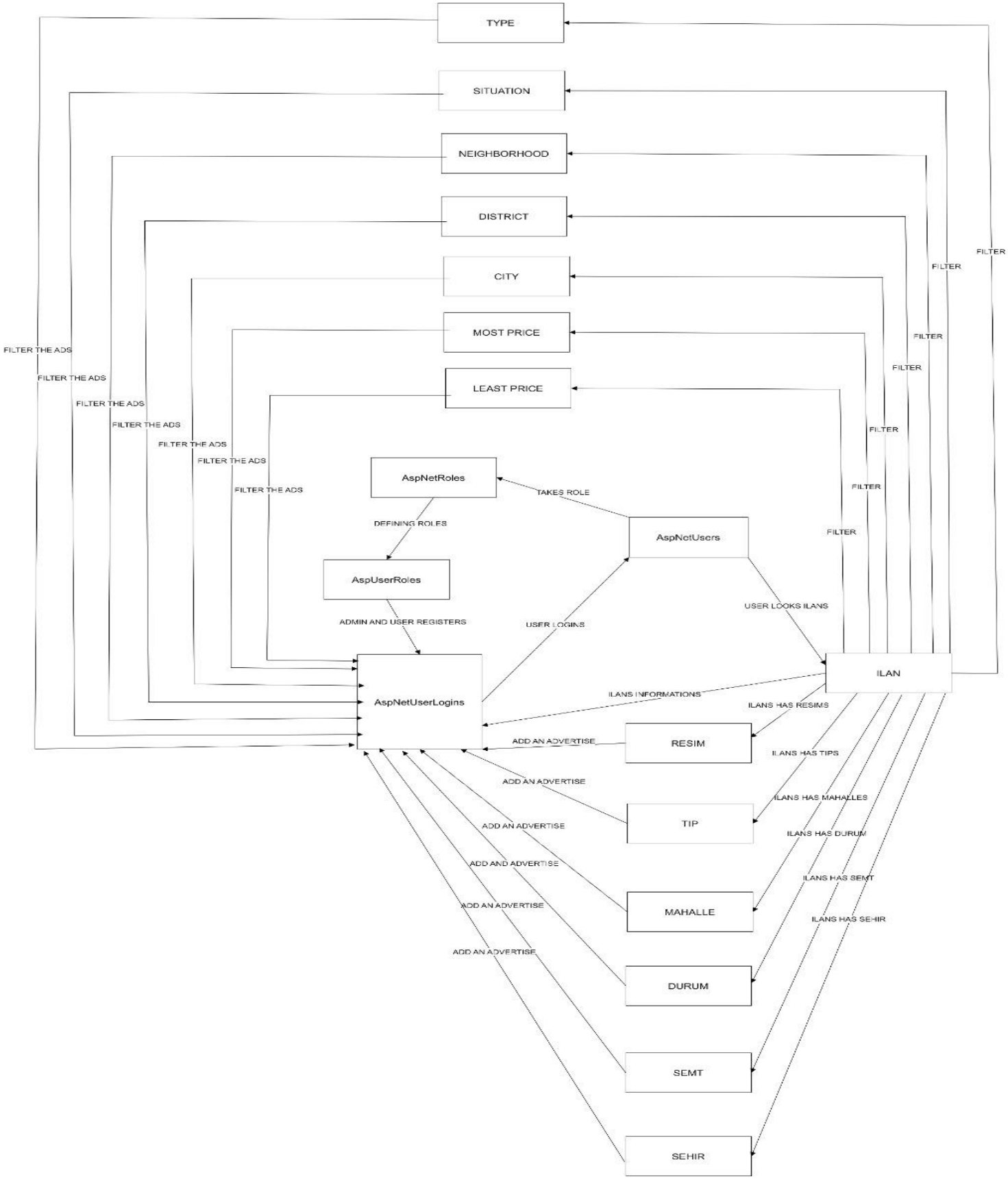
3.1.4.2 | Sequence Diagram



3.1.4.3 | Timing Diagram

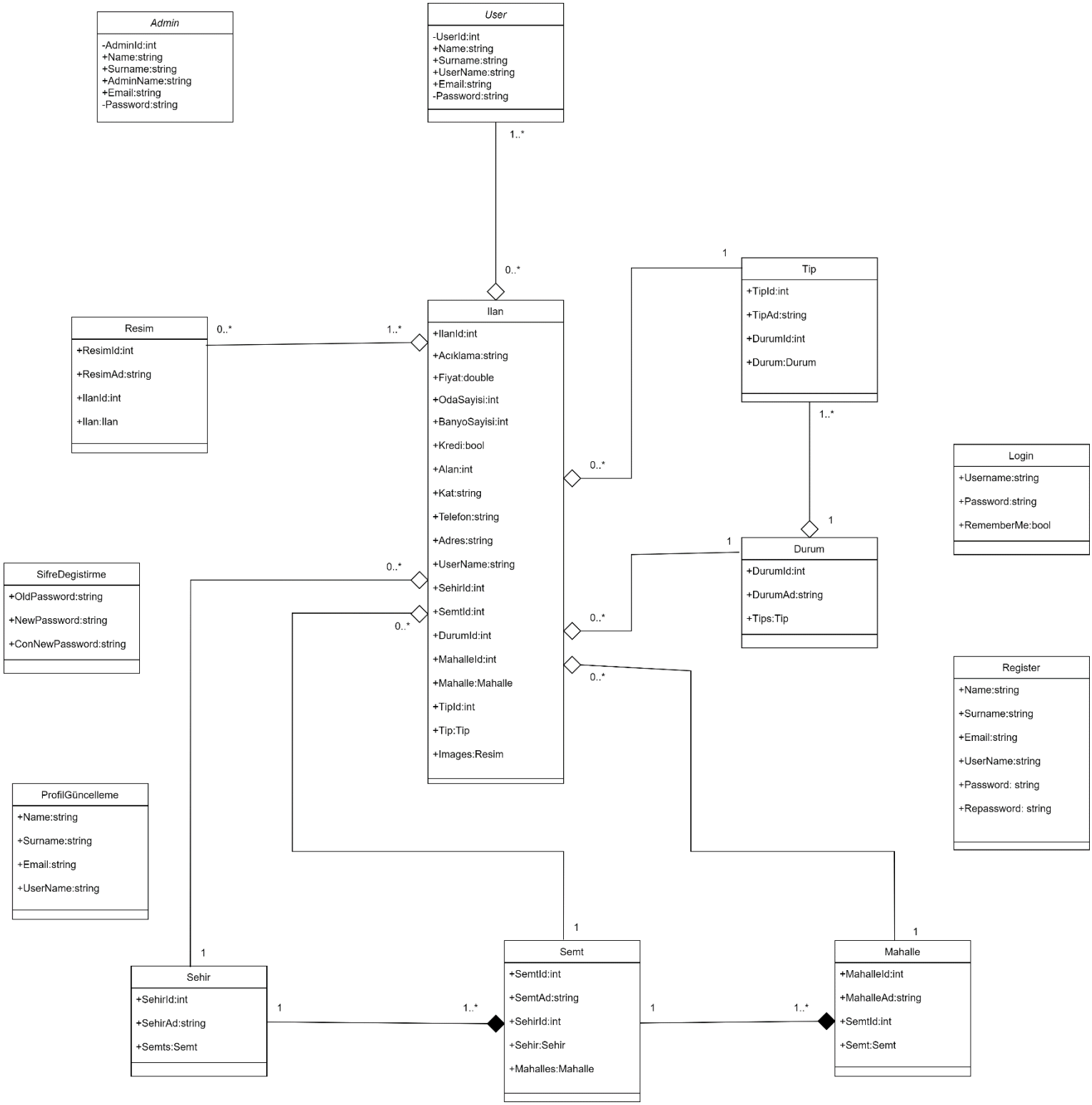


3.1.4.4 | Collaboration Diagram (Communication Diagram)

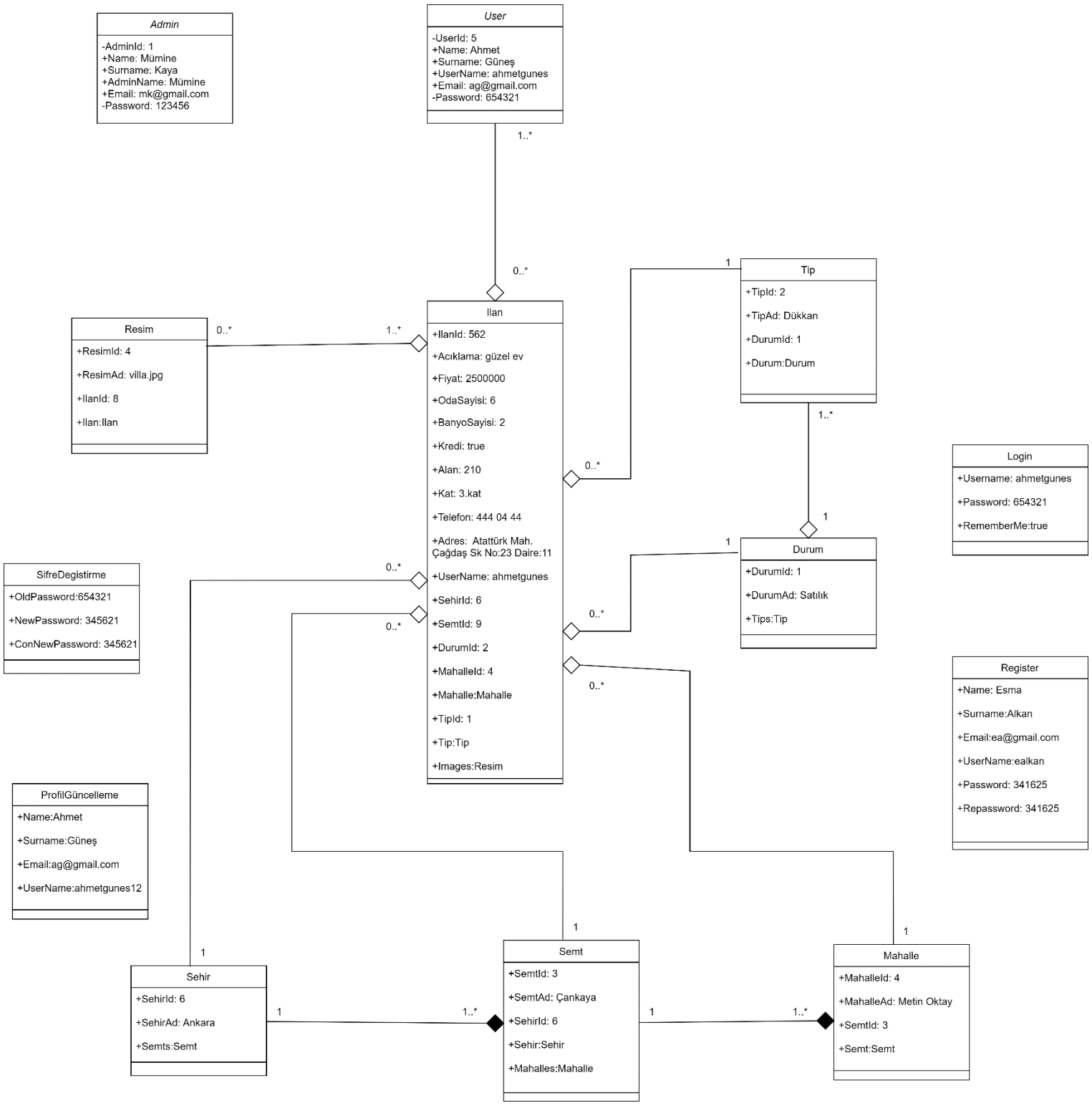


3.2 Structural Diagrams

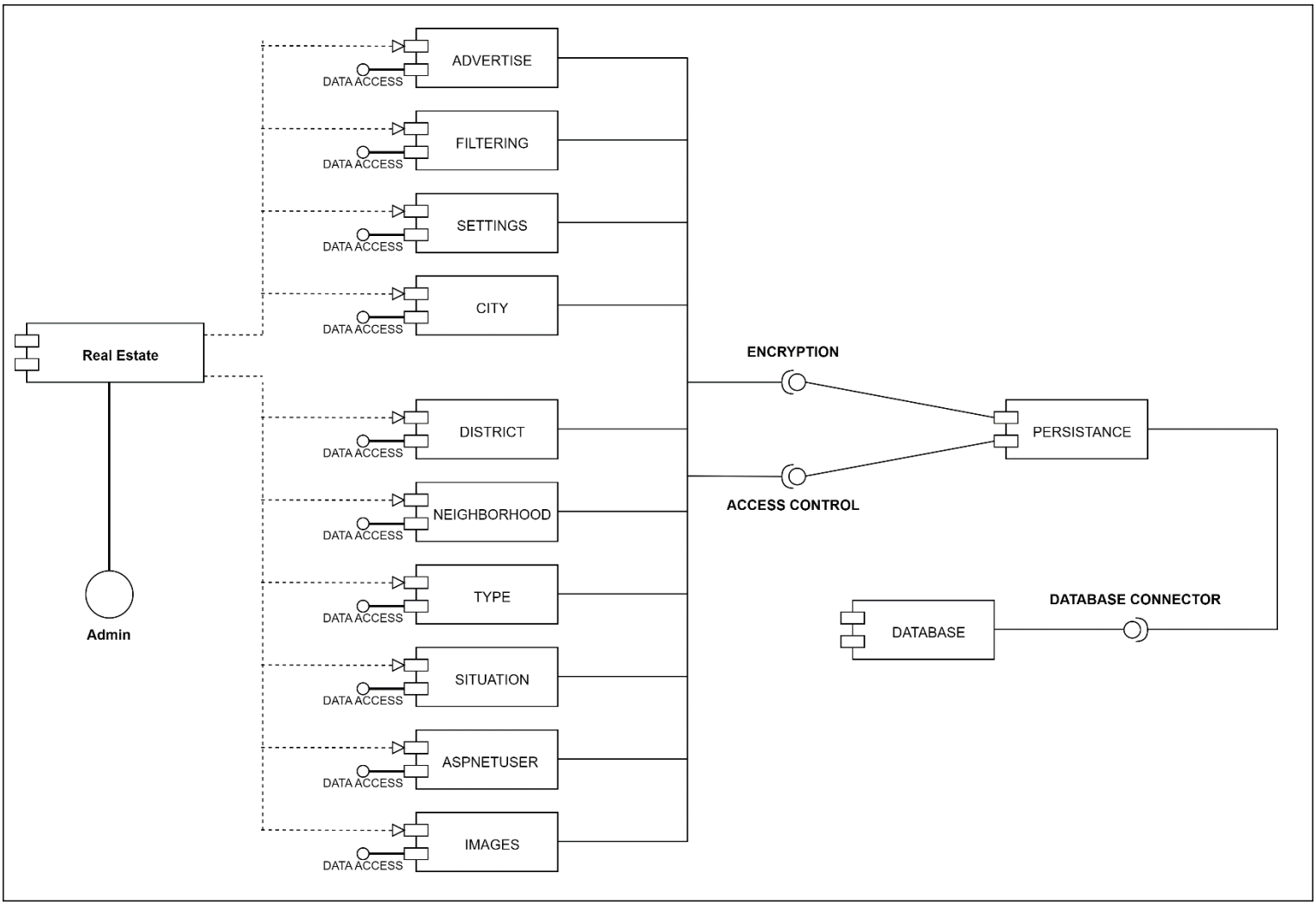
3.2.1 | Class Diagram



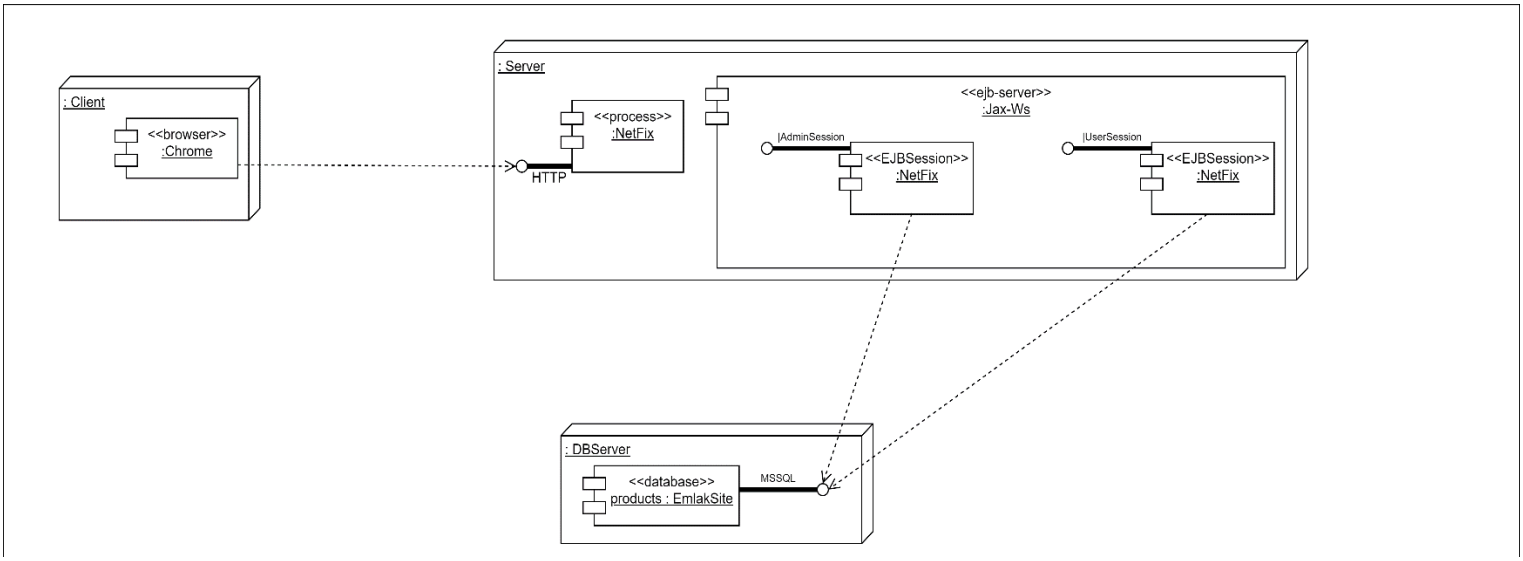
3.2.2 | Object Diagram



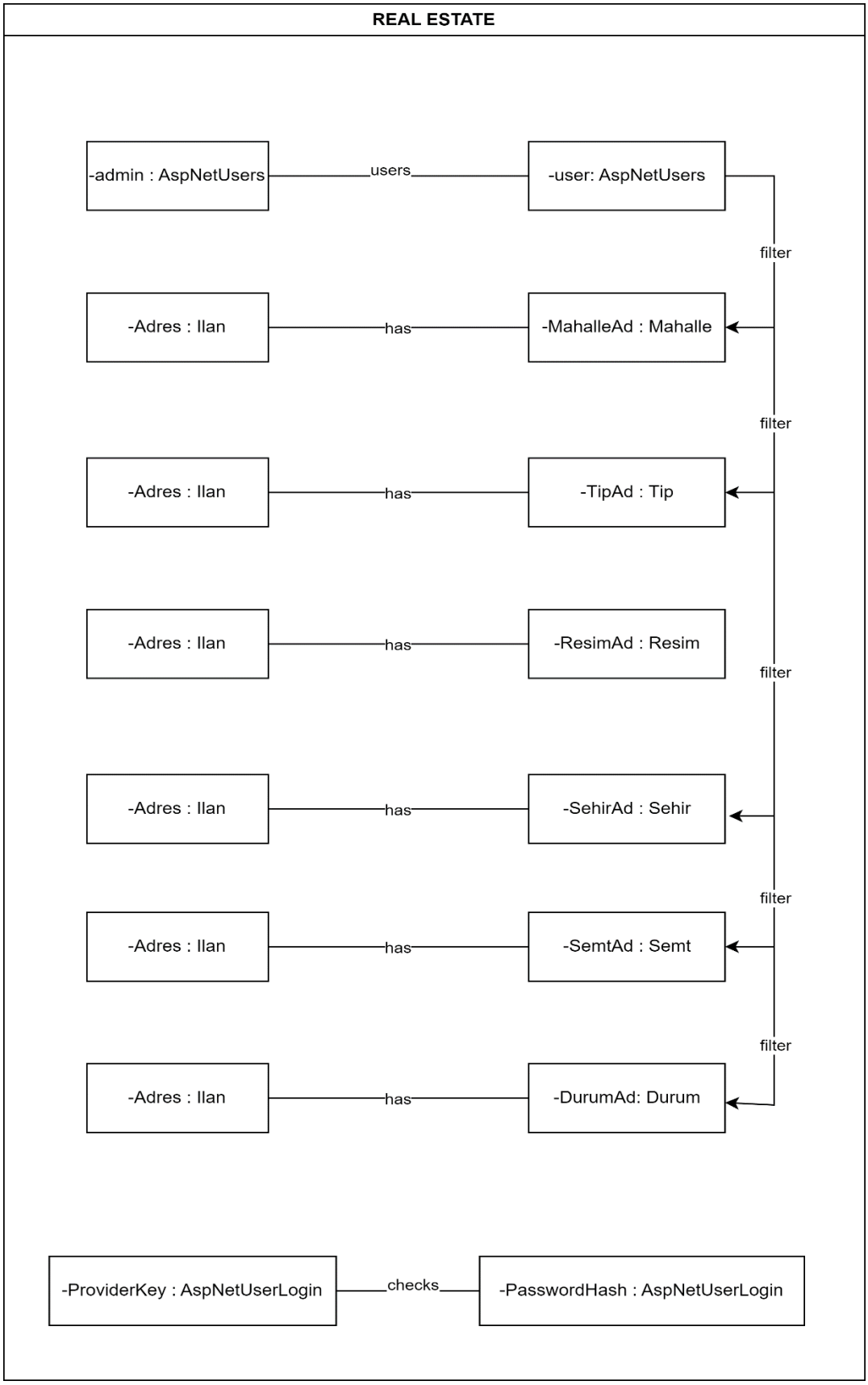
3.2.3 | Component Diagram



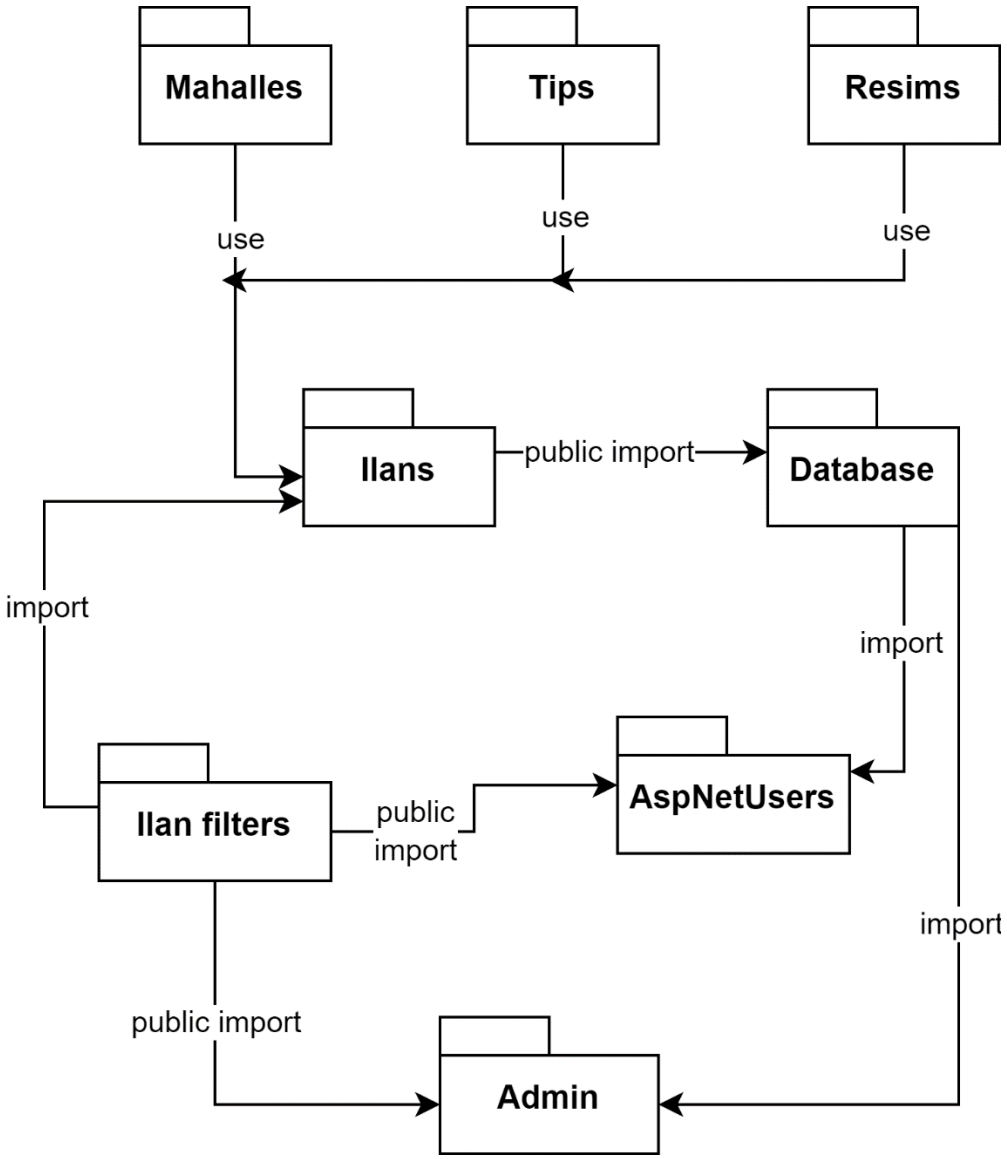
3.2.4 | Deployment Diagram



3.2.5 | Composite Structure Diagram



3.2.6 | Package Diagram



4. Software Testing

4.1 Program Testing

- **Purpose:** The purpose of program testing for the Real Estate Database and Website is to ensure the correct and reliable operation of the software, validating its functionality, performance, and security.

- **Tests Phases:**
  - Compile all code components to ensure syntactical correctness and identify compilation errors.
  - Execute the compiled code to verify its functionality and identify any runtime errors.
  - Ensure compatibility between database and website components during the compilation and execution processes.
  - Conduct Unit Tests to evaluate the individual functionalities of database and website components.
  - Perform Integration Tests to assess the seamless interaction between different modules.
  - Validate data flow and consistency between the Real Estate Database and the Website.
  - Check for the generation of error messages or unexpected results during code execution.
  - Verify error messages for clarity and accuracy.
  - Confirm that the code produces expected results and handles unexpected situations gracefully.

4.1.1 | Verification for Real Estate Database and Website

- **Purpose:**
  - To assess whether the software aligns with specified requirements, ensuring that it meets the defined criteria and standards.
  - To confirm that the Real Estate Database and Website function according to established guidelines and expectations.

- **Steps:**
    - Determination of requirements specific to the Real Estate Database and Website.
    - Checking whether the software complies with the identified requirements.
    - Define and document specific requirements for the Real Estate Database and Website.
    - Evaluate the software against the documented requirements to ensure compliance.
    - Example: Does the Real Estate Database accurately store property details as specified in the requirements?
    - Example: Does the Website correctly display search results based on user-defined criteria?
    - Example: Is the Real Estate Database storing and retrieving property data in accordance with the defined specifications?
    - Example: Does the Website adhere to design standards and user interface guidelines as outlined in the project requirements?

4.1.2 | Validation for Real Estate Database and Website

- **Purpose:**
    - To check whether the software meets user expectations.
    - To validate that the software not only meets technical specifications but also satisfies the needs and expectations of end-users.
    - To ensure that the Real Estate Database and Website deliver a positive and user-friendly experience, aligning with user preferences.

- **Steps:**
    - Testing the software with real users who engage with the Real Estate Database and Website.
    - Evaluation of user feedback to ensure that the software aligns with user expectations.
    - Engage real users in testing scenarios to observe their interaction with the Real Estate Database and Website.
    - Collect and analyze user feedback to identify areas of improvement and ensure user satisfaction.
    - **Example:** Does the Real Estate Website provide an intuitive and efficient property search experience for users?
    - **Example:** Does the Real Estate Database and Website confirm user actions, such as property listings or updates, accurately and promptly?
    - **Example:** Is the Real Estate Website's search functionality intuitive and user-friendly, meeting the expectations of users with varying levels of technical expertise?
    - **Example:** Does the Real Estate Database and Website provide clear and accurate confirmations to users, enhancing their overall experience and confidence in the system?

4.2 Development Testing

4.2.1 | Unit Testing for Real Estate Database

- **Purpose:**
    - To verify the correctness of individual units within the Real Estate Database, ensuring that each unit produces the expected output and functions as intended.

- **Steps:**
    - **Selection of Unit to Test:** "Property Listing" function.
    - **Creating Test Scenarios:** Scenarios include adding a new property, updating property details, and deleting a property.
    - **Implementation of Unit Testing:** Adding a new property to the database and verifying its existence.
    - **Checking Expected Outputs:** Confirming that the property is successfully added and can be retrieved from the database.
    - **Example:** Does the unit responsible for updating property details reflect the changes accurately in the database?
    - **Example:** Can the system handle gracefully when attempting to delete a property that is linked to active contracts?

4.2.2 | Unit Testing for Web Site

- **Purpose:**
    - To validate the functionality of individual units within the Real Estate website, ensuring a seamless user experience.

- **Steps:**
    - **Selection of Unit to Test:** "User Registration" function.
    - **Creating Test Scenarios:** Scenarios include successful registration, duplicate email prevention, and validation of input fields.
    - **Implementation of Unit Testing:** Registering a new user and verifying their account creation.
    - **Checking Expected Outputs:** Confirming that the user is redirected to the login page after successful registration.
    - **Testing Error Conditions:** Does the system warn when an empty field is left?
    - **Example:** Does the unit handling property search return accurate and up-to-date results from the database?
    - **Example:** Can the system prevent unauthorized access to sensitive information by restricting certain pages to authenticated users only?

4.2.3 | Component Testing for Real Estate Database

- **Purpose:**
  - Verify that each independent module (component) of the Real Estate Database functions as intended.

- **Steps:**
  - **Component Selection to Test:** "Property Listing" Component: Ensure that the property listing component correctly displays property details, including images, descriptions, and pricing.

  - **Creating Test Scenarios:** Test the display of different property types (apartments, houses, commercial spaces) within the property listing component. Verify the accurate presentation of property amenities and features.

  - **Implementing Component Testing:** Execute test cases that involve querying the database for property details and validating the returned results. Evaluate the performance of the property listing component by assessing its response time under varying load conditions.

  - **Checking Expected Outputs:** Confirm that the property details retrieved from the database match the information displayed on the website. Ensure that the images associated with each property are displayed correctly.

  - **Testing Error Conditions:** Simulate scenarios where the property information is incomplete and assess how the system handles such situations.
    Evaluate the system's response when attempting to access a non-existent property record.

  - **Example:** Does the property listing component accurately retrieve and display details for a diverse range of properties, including apartments, houses, and commercial spaces?

  - **Example:** Can the system seamlessly handle and display property images, descriptions, and pricing information, ensuring a consistent user experience?

4.2.4 | Component Testing for Real Estate Website

- **Purpose:**
  - Validate that each component of the Real Estate Website functions independently and integrates correctly.

- **Steps:**
  - **Component Selection to Test:** "User Registration" Component. Verify the registration process, including user input validation and database storage of user information.

  - **Creating Test Scenarios:** Test user registration with various input data, including different combinations of username, password, and personal details. Assess the system's response to simultaneous registration attempts from multiple users.

  - **Implementing Component Testing:** Execute test cases that involve sending registration requests to the server and validating the registration status. Evaluate the security measures in place to protect user information during the registration process.

  - **Checking Expected Outputs:** Confirm that user registration results in the creation of a new user account with accurate and complete information stored in the database. Ensure that the system provides appropriate feedback to users upon successful registration.

  - **Testing Error Conditions:** Simulate scenarios where users attempt to register with invalid or duplicate information and assess how the system handles such cases. Evaluate the system's response to registration attempts when the database is unreachable.

  - **Example:** Does the user registration component effectively validate user input and prevent the creation of accounts with incomplete or invalid information?

  - **Example:** Can the system securely store user registration data in the database and provide timely feedback to users upon successful registration?

4.2.5 | System Testing

- **Purpose:**
  - Verify that all components within the Real Estate Database and Website function as expected when integrated into the complete system. Ensure the overall performance, functionality, and reliability of the entire software system.

- **Steps:**
  - **Creating System Test Scenarios:** Testing the end-to-end process of property listing and transaction within the real estate system. Verifying the functionality of advanced search features across various property types.

  - **Testing Different User Roles in the System:** Evaluating the user experience for both property seekers and real estate agents. Verifying that administrative roles have access to necessary management and reporting features.

  - **Testing Error Conditions:** Simulating scenarios where users input invalid property details and ensuring the system handles errors gracefully. Verifying that appropriate error messages are displayed to users in case of data entry mistakes or system malfunctions.

  - **Running Performance Test Scenarios:** Testing the system's response time during peak usage hours. Assessing the compatibility of the website and database across multiple browsers (Chrome, Firefox, Safari, etc.) and devices (mobile phones, tablets, desktop computers).

  - **Security Testing:** Checking the security measures in place to protect sensitive user information. Verifying that user authentication and authorization mechanisms are robust and prevent unauthorized access.

  - **Data Integrity Testing:** Ensuring that data stored in the Real Estate Database remains accurate and consistent throughout various transactions. Verifying the reliability of data retrieval and update operations.

4.2.6 | Integration Testing

- **Purpose:**
  - To verify the seamless integration of the Real Estate Database and the Website, ensuring that they function cohesively.

- **Steps:**
  - **Selection of Integration to Test:** Adding a new property listing to the Real Estate Database and testing the integration of this property into the website's listing display. Verifying the integration of user account creation in the website with the storage and retrieval of user data in the Real Estate Database.

  - **Creating Integration Test Scenarios:** Testing the integration between the property search functionality on the website and the data retrieval from the Real Estate Database. Checking the synchronization between property status updates in the database and their reflection on the website in real-time.

  - **Implementing Integration Testing:** Executing tests to ensure the smooth communication and data exchange between the web server and the database server. Validating the integration of location-based services, such as mapping functionalities, with property data stored in the database.

  - **Checking Expected Outputs:** Verifying that property details displayed on the website align with the information stored in the Real Estate Database. Ensuring that updates made through the website are accurately reflected in the database and vice versa.

  - **Example:** Verifies if the Real Estate Database integrates seamlessly with the website when new property listings are added, ensuring accurate and up-to-date information for users. Checks whether the website provides proper integration between user interactions, such as property inquiries, and the corresponding data stored in the Real Estate Database.

  - **Additional Steps:** Testing the integration of multimedia elements,images, videos etc., associated with property listings on the website and their storage in the database. Verifying the integration between user authentication processes on the website and the security measures implemented in the Real Estate Database to protect sensitive information.

4.3 White Box Testing

- **Purpose:** To assess the internal structure and code of the Real Estate Database and Website, ensuring that all components function as intended and that potential issues within the code are identified and addressed.

- **Tests Phases:**
  - **Understanding the Internal Structure of the Code:** Analyzing the database schema to ensure it aligns with the defined data model for property listings, user profiles, and other relevant entities. Reviewing the source code of the website to understand the implementation of features such as user authentication, property search algorithms, and data retrieval from the database.

  - **Testing All Paths:** Conducting unit tests to validate different code paths within the Real Estate Database, such as data insertion, retrieval, and update operations. Evaluating various user interactions on the website, such as property searches, form submissions, and account management, to ensure all possible paths are tested thoroughly.

  - **Example:** Verifying that database operations, such as property addition and modification, are performed correctly by examining the internal structure of the Real Estate Database code. Ensuring the accuracy of user authentication processes by testing different authentication paths within the website's codebase.

  - **Additional Steps:** Reviewing error-handling mechanisms in the code to validate that appropriate error messages are generated and logged for various scenarios, enhancing system robustness.

  - **Code Coverage Analysis:** Utilizing code coverage tools to identify areas of the codebase that may not be adequately tested, ensuring comprehensive test coverage.

  - **Security Vulnerability Assessment:** Conducting a security-focused review of the codebase to identify and mitigate potential vulnerabilities, especially in areas handling user input and authentication. Verifying that the database queries are parameterized to prevent SQL injection attacks, enhancing the overall security posture of the system.

  - Summarizing the findings from white box testing and providing recommendations for code optimizations or enhancements based on identified areas of improvement.

4.3.1 | Big Bang Testing

- **Purpose:** Big Bang Testing is an integration testing approach where all the modules or components of a system are integrated simultaneously to detect defects in the system as a whole. The purpose of Big Bang Testing for the Real Estate Database and Website is to ensure that all the components of the system work seamlessly together and to identify and rectify any integration issues that may arise.

- **Suppose a user searches for a property on the website. The Big Bang Testing scenario would involve:**
    - Verifying that the search query is processed correctly by the website.
    - Ensuring that the website communicates effectively with the Real Estate Database to retrieve property information.
    - Checking that the displayed property details match the information stored in the database.
    - Confirming that any filters or sorting options related to the search function work as expected.

4.3.2 | Top Down Testing

- **Purpose:** Top Down Testing is an integration testing approach that starts with testing the higher-level modules and gradually moves towards testing the lower-level modules. The purpose of Top Down Testing for the Real Estate Database and Website is to ensure the proper functioning of the integrated system by validating the interactions between high-level components and their underlying subsystems.

- **Consider the process of a user listing a property on the Real Estate Website. In a Top Down Testing scenario:**
    - Start by testing the high-level module responsible for property listing functionality.
    - Gradually integrate lower-level modules such as image uploading, property details validation, and database storage.
    - Verify that the integrated system correctly processes and stores property listing information without compromising the user experience.

4.3.3 | Bottom Up Testing

- **Purpose:** Bottom-Up Testing is an integration testing approach that begins with testing the lower-level modules or components and progresses towards testing the higher-level modules. The purpose of Bottom-Up Testing for the Real Estate Database and Website is to ensure that individual components function correctly when integrated into larger subsystems, thereby validating the reliability of the entire system.

- **Consider the process of retrieving and displaying property details on the Real Estate Website. In a Bottom-Up Testing scenario:**
    - Start by testing lower-level modules responsible for database access and data retrieval.
    - Gradually integrate higher-level modules, such as user interface components and frontend services.
    - Verify that the integrated system correctly retrieves and displays property details on the website, ensuring a seamless user experience.

4.3.4 | Hybrid Testing

- **Purpose:** Hybrid Testing is an integration testing approach that combines elements of both Top-Down and Bottom-Up Testing. The purpose of Hybrid Testing for the Real Estate Database and Website is to leverage the strengths of both approaches, allowing for comprehensive testing from the higher-level modules down to the lower-level modules and vice versa. This approach aims to achieve thorough coverage and efficient defect detection.

- **Consider the scenario of a user submitting a property inquiry on the Real Estate Website. In a Hybrid Testing scenario:**
    - Start with top-down testing, focusing on the user interface and frontend processing of the inquiry.
    - Incrementally integrate lower-level modules responsible for backend processing and database updates.
    - Validate that the inquiry data is correctly processed, stored in the database, and reflected in the user interface.

4.4 Black Box Testing

4.4.1 | Black Box Testing for Real Estate Database

- **Purpose:**
  - The primary goal of black box testing for the Real Estate Database is to evaluate the functionality and performance of the system by focusing solely on inputs and outputs, without delving into the internal code or architecture.

- **Steps:**
  - **Identifying Input Values:** Identify and validate various input values such as property details, client information, and location data to ensure accurate processing.
  - **Verifying Data Retrieval:** Confirm that property information, pricing details, and other relevant data are retrieved accurately from the database.
  - **Testing Search Functionality:** Evaluate the search functionality to ensure that users can effectively search and retrieve properties based on different criteria.

- **Examples:**
  - **Property Listing:** Test the process of adding a new property to the database through the user interface, validating that all entered details are correctly stored in the system.
  - **Transaction Processing:** Perform simulated transactions such as property sales or rentals to verify that the database accurately records and updates financial transactions.

4.4.2 | Black Box Testing for Real Estate Website

- **Purpose:**
  - The primary objective of black box testing for the Real Estate Website is to assess the user interface, navigation, and overall functionality without inspecting the underlying code.

- **Steps:**
    - **User Authentication:** Test the login and registration functionalities to ensure secure access and account creation.
    - **Navigation Testing:** Verify the website's navigation, ensuring that users can easily move between pages and access various features.
    - **Form Validations:** Evaluate form submissions for property inquiries, ensuring that the website validates and processes user inputs correctly.

- **Examples:**
    - **Property Search:** Perform searches for properties using different filters, confirming that the website displays accurate results and filters properties based on user preferences.
    - **Contact Forms:** Submit contact forms through the website to validate that user inquiries are successfully recorded and processed.

4.5 Gray Box Testing

- **Purpose:** Gray Box Testing for the Real Estate Database and Website aims to gain insights into the partial internal structure of the software. This approach allows us to conduct a comprehensive assessment by combining both internal and external testing methodologies.

- **Tests Phases:**
    - **Understanding Partial Internal Structure:** Analyze the database schema to understand the relationships between different tables, ensuring data integrity. Examine the backend code to gain insights into the business logic governing the real estate operations.

    - **Testing Inputs and Outputs:** Evaluate user inputs on the website to verify if they are processed correctly and securely. Validate the outputs generated by the system, ensuring that they align with expected results and adhere to business rules.

- **Examples:**

    - **Verifying Property Listings**: Inspect the backend logic responsible for updating and displaying property listings. Perform a series of searches and verify that the displayed results on the website align with the properties stored in the database.

    - **Validating User Account Creation:** Examine the registration process on the website to ensure that user data is accurately captured and stored in the database. Validate that users can log in using the credentials created during registration.

    - **Testing Search Functionality:** Investigate the search algorithm to confirm that it accurately retrieves properties based on user-specified criteria. Cross-verify the displayed results with the database to ensure consistency and accuracy.

    - **Ensuring Transaction Integrity:** Scrutinize the internal transaction processes, especially during property transactions. Simulate property transactions through the website and verify that the corresponding changes are correctly reflected in the database.

    - **Database Transaction Auditing:** Implement auditing mechanisms to log and review database transactions.

    - **Cross-Checking User Interface with Database:** Compare the data displayed on the user interface with the actual data stored in the database. Identify any discrepancies and ensure that the user interface accurately reflects the current state of the database.

4.6 User Testing

- **Purpose:** User testing is a critical phase in the development process that ensures the software aligns with user expectations and requirements.

- **Tests Phases**

    1. **Alpha Testing**

        - **Purpose:** To evaluate the software's functionality and content by the developer team.

        - **Examples:**
            - **Initial Content and Functionality Testing:** The Real Estate Database undergoes rigorous testing by the development team to identify and rectify issues related to content and functionality.

            - **Feedback Collection on New Features:** Developers use the system internally to gather feedback on newly implemented features.

            - **Internal Use for Quality Assurance:** The Web site is utilized by the development team to ensure the software meets quality standards before reaching external users.

    2. **Beta Testing**

        - **Purpose:** To assess the software in real-world scenarios with a limited number of actual users.

        - **Examples:**
            - **Limited User Testing:** The Real Estate Database is deployed to a select group of real users who provide insights into its performance and usability.

            - **Real-world Scenario Usage:** Specific users engage with the system under real-world conditions, simulating actual usage scenarios.

            - **Feedback Collection from User Group:** The development team collects feedback from the limited user group to refine the software based on actual user experiences.

3. **Acceptance Testing**

   - **Purpose:** To verify whether end users accept and approve the software.

   - **Examples:**
     - **End User Approval Testing:** The Real Estate Database is made available to a broader audience, and end users test and evaluate the software.

     - **Real User Evaluation:** The Web site is used by real users in diverse conditions to assess its performance and suitability.

     - **End User Feedback Integration:** Feedback received from end users during the acceptance test phase is considered for any necessary refinements before the official release.

4.7 Regression Testing

- **Purpose:** To control the effects of any changes made to the software on other features, preventing unintended consequences and maintaining overall system integrity.

- **Tests Phases:**
  - **Adding New Feature or Modifying Existing Feature:** When a new feature is introduced or an existing feature is modified in the Real Estate Database or the Web site, regression testing is initiated to detect any adverse impacts on existing functionalities.
  - **Re-running Existing Tests:** The testing team re-executes previously defined test cases to verify that alterations have not negatively affected the established functionality of the software.
  - **Post-Update Verification:** After implementing a new feature, the Real Estate Database undergoes regression testing to confirm that all existing functions continue to operate as expected.
  - **Bug Fix Validation:** In the event of resolving a bug, the system performs regression testing to ensure that the correction does not introduce new issues or disrupt other features.
  - **Example:** A bug related to user authentication is identified and fixed in the Web site to enhance security and user access control.

4.8 Performance Testing

- **Purpose:** The primary objective of performance testing for the Real Estate Database and accompanying website is to gauge the software's responsiveness, scalability, and overall durability in diverse scenarios, ensuring a consistently high level of user satisfaction and system reliability.

- **Tests Phases:**
  - **System Load Test:** Simulate a realistic volume of concurrent users and evaluate the system's ability to handle the expected load. Measure response times and throughput under normal and peak conditions.

  - **Stress Test:** Apply extreme load conditions to identify system breaking points and assess its behavior under stress. Evaluate how the system recovers from stress conditions and if it can resume normal operation.
  - **Example:** Assess whether the Real Estate Website remains responsive and stable even when subjected to a sudden surge in user traffic, such as during a flash sale or a popular property listing.

  - **Running Performance Test Scenarios:** Create test scenarios that mimic common user interactions, including property searches, updates, and transactions. Measure the system's response time for each scenario and identify potential bottlenecks.
  - **Example:** Analyze whether the Real Estate Database provides quick search results and updates when users perform typical tasks like browsing properties, updating preferences, or finalizing transactions.

  - **Scalability Testing:** Investigate the system's ability to scale vertically and horizontally to accommodate an increasing number of users and data.
  - **Example:** Determine if the Real Estate Database can seamlessly expand its capacity as the volume of property listings and user interactions grows over time.

  - **Load Balancing Assessment:** Evaluate the effectiveness of load balancing mechanisms in distributing incoming traffic across multiple servers.
  - **Example:** Ensure that the Real Estate Website can maintain optimal performance even during unevenly distributed user loads, preventing individual server overloads.

- **Database Query Performance:** Assess the efficiency of database queries and optimize them for faster retrieval of property information.
- **Example:** Confirm that the Real Estate Database retrieves property details promptly, especially during peak usage, ensuring a seamless experience for users browsing and searching for properties.

4.9 Release Testing

- **Purpose:** To ensure the Real Estate Database and accompanying website are thoroughly tested and ready for release, meeting quality standards and user expectations.

- **Tests Phases:**
  - Thorough testing of all phases, including unit testing, integration testing, and system testing, to validate the functionality and reliability of the Real Estate Database.
  - Verification of data accuracy and consistency within the database, ensuring that it accurately reflects the real estate information.
  - Conducting performance testing to assess the responsiveness and efficiency of the database under various workloads.
  - Validation of data retrieval and storage processes to ensure seamless and accurate interaction with the database.
  - Comprehensive testing of the website's user interface to guarantee a seamless and intuitive experience for users browsing real estate listings and utilizing features.
  - Cross-browser and cross-device testing to ensure compatibility and responsiveness across various platforms.
  - Examination of the website's performance under different network conditions to ensure optimal user experience.
  - Verification of data synchronization between the Real Estate Database and the website, confirming that real-time updates are reflected accurately.
  - Assessment of website security measures to protect user data and maintain the integrity of the real estate platform.
  - Testing the website's search and filtering functionalities to ensure users can easily find and access relevant real estate listings.

5.0 Test-Driven Development (TDD)

5.0.1 | Test-Driven Development (TDD) for Real Estate Database

- **Purpose:** The primary objective of employing Test-Driven Development (TDD) in the Real Estate Database is to ensure robust, error-free functionality by writing tests before implementing the corresponding code.

- **Steps:**
    - **Write a Test:** Begin by articulating a test case that defines the expected behavior of a specific feature or functionality within the Real Estate Database.
    - **Test Failure:** Execute the test, anticipating its failure since the corresponding code has not been developed yet.
    - **Code Implementation:** Write the necessary code to fulfill the requirements outlined in the test, aiming for the test to pass successfully.
    - **Test Success:** Validate the success of the test to confirm that the newly implemented code meets the specified criteria.
    - **Example:** For instance, in the Real Estate Database, when introducing a feature to track property ownership changes, tests are drafted beforehand. The code is then developed to meet these predefined test scenarios.

5.0.2 | Test-Driven Development (TDD) for Real Estate Website

- **Purpose:**
    - The aim of employing Test-Driven Development (TDD) in the Real Estate Website is to guarantee a reliable and user-friendly interface by crafting tests before implementing the corresponding code.

- **Steps:**
    - **Test Creation:** Start by defining tests that describe the expected behavior or appearance of a particular feature on the Real Estate Website.
    - **Test Failure:** Execute the test, expecting it to fail initially as the code supporting the test has not been implemented yet.
    - **Code Implementation:** Write the necessary code to realize the features outlined in the test, with the goal of ensuring the test passes successfully.
    - **Test Validation:** Confirm the success of the test, verifying that the newly implemented code aligns with the specified requirements.
    - **Example:** In the context of the Real Estate Website, when introducing a feature for advanced property searches, the development process starts by outlining test scenarios. Subsequently, the website is coded to pass these predefined tests.

5.1 Coding and Debugging

5.1.1 | Coding

- **Purpose:** Writing the software code to implement desired features.

- **Steps:**
    - **Requirement Analysis:** Thoroughly understand the requirements of the Real Estate Database. Identify key functionalities such as property listings, user authentication, and search capabilities.

    - **Structured Code Development:** Implement modular and structured code to enhance maintainability. Utilize object-oriented principles for a scalable and flexible codebase.

    - **Database Integration:** Integrate the database schema with the codebase seamlessly.Optimize database queries to ensure efficient data retrieval and manipulation.

    - **Feature Expansion:** Code additional features like property details, image uploads, and user feedback mechanisms. Ensure new features align with the overall design and user experience.

- **Code Review and Collaboration:** Conduct regular code reviews to maintain code quality. Collaborate with team members to address code-related challenges and enhancements.

5.1.2 | Debugging

- **Purpose:** Identifying and correcting errors to ensure a robust and error-free system.

- **Steps:**
  - **User-Reported Issues:** Investigate and address issues reported by users promptly. Establish a systematic approach to prioritize and resolve bugs based on their impact.

  - **Error Logging and Monitoring:** Implement robust error logging mechanisms to capture and track system errors. Set up monitoring tools to detect anomalies and proactively address potential issues.

  - **Regression Testing:** Perform regression testing after each code update to identify and rectify unintended side effects. Develop automated testing scripts to streamline the regression testing process.

  - **Collaborative Debugging Sessions:** Organize collaborative debugging sessions to tackle complex issues. Encourage knowledge sharing among team members for a more effective debugging process.

```
<script src="~/scripts/jquery-1.10.2.min.js"></script>
<script>
    $(document).ready(function() {
        $("#SehirId").change(function() {
            var sehirid = $(this).val();
            debugger
            $.ajax({
                type: "Post",
                url: "/Ilan/SemtGetir?SehirId=" + sehirid,
                contentType: "html",
                success: function(response) {
                    debugger
                    $("#SemtId").empty();
                    $("#SemtId").append(response);
                }
            })
        })
    })
```

```
<script>
    $(document).ready(function() {
        $("#SemtId").change(function() {
            var semtid = $(this).val();
            debugger
            $.ajax({
                type: "Post",
                url: "/Ilan/MahalleGetir?SemtId=" + semtid,
                contentType: "html",
                success: function(response) {
                    debugger
                    $("#MahalleId").empty();
                    $("#MahalleId").append(response);
                }
            })
        })
    })
```

```
<scri <script>...
        $(document).ready
        function()
    $       eady(function() {
        $("#DurumId").change(function() {
            var durumid = $(this).val();
            debugger
            $.ajax({
                type: "Post",
                url: "/Ilan/TipGetir?DurumId=" + durumid,
                contentType: "html",
                success: function(response) {
                    debugger
                    $("#TipId").empty();
                    $("#TipId").append(response);
                }
            })
        })
    })
```

5. IDENTIFICATION OF PROJECT RISKS AND PROJECT RISK LIST

5.1 Identification of Project Risks

5.1.1 | Market Fluctuations

- Risk: Unforeseen changes in the real estate market may affect property values and demand.
- Mitigation: Regular market analysis and adaptation strategies to respond to fluctuations.

5.1.2 | Technology Risks

- Risk: Issues related to the implementation of new technologies or reliance on outdated systems.
- Mitigation: Thorough testing, continuous technological updates, and contingency plans.

5.1.3 | Regulatory Compliance

- Risk: Changes in local, state, or federal regulations impacting the real estate industry.
- Mitigation: Regular legal reviews, staying informed about regulatory changes, and adapting processes accordingly.

5.1.4 | Cybersecurity Threats

- Risk: Potential security breaches, data theft, or website vulnerabilities.
- Mitigation: Implementing robust cybersecurity measures, regular security audits, and employee training on cybersecurity best practices.

5.1.5 | Client Data Privacy

- Risk: Mishandling or unauthorized access to sensitive client information.
- Mitigation: Implementing strict data protection policies, securing data storage, and ensuring compliance with privacy regulations.

5.1.6 | Project Scope Creep

- Risk: Uncontrolled changes and expansions in project scope.
- Mitigation: Clearly defined project scope, regular communication with stakeholders, and a change management process.

5.1.7 | Resource Constraints

- Risk: Insufficient human or financial resources to meet project demands.
- Mitigation: Resource planning, regular assessments of resource needs, and contingency plans for unexpected resource shortages.

5.1.8 | Stakeholder Communication

- Risk: Poor communication with stakeholders leading to misunderstandings or misaligned expectations.
- Mitigation: Establishing clear communication channels, regular updates, and addressing stakeholder concerns promptly.

5.1.9 | Environmental Factors

- Risk: External factors like natural disasters affecting the project or property values.
- Mitigation: Contingency plans for potential environmental impacts and staying informed about local environmental conditions.

5.1.10 | Vendor Relationships

- Risk: Dependence on external vendors for critical components or services.
- Mitigation: Regular vendor assessments, clear contracts, and contingency plans for vendor-related issues.

5.2 Project Risk List

Risk Category	Risk Title	Likelihood	Mitigation
Market Fluctuations	Potential decrease in property values and demand.	Moderate	Continuous market analysis, agile project strategies to adapt to market changes.
Technology Risks	Potential system failures or implementation delays.	Moderate	Thorough testing, regular technology updates, and contingency plans.
Regulatory Compliance	Legal consequences and project delays.	Low	Regular legal reviews, staying informed about regulatory changes, and adapting processes accordingly.
Cybersecurity Threats	Data breaches, compromised website integrity.	High	Robust cybersecurity measures, regular security audits, and employee training on cybersecurity best practices.
Client Data Privacy	Damage to reputation and legal consequences.	Moderate	Strict data protection policies, secure data storage, and compliance with privacy regulations.
Project Scope Creep	Uncontrolled project expansion and delays.	High	Clearly defined project scope, regular communication with stakeholders, and a change management process.
Resource Constraints	Delays and compromised project quality.	Moderate	Resource planning, regular assessments of resource needs, and contingency plans for unexpected shortages.
Stakeholder Communication	Misunderstandings, misaligned expectations.	Low	Establishing clear communication channels, regular updates, and addressing stakeholder concerns promptly.
Environmental Factors	Project disruption due to external factors.	Low	Contingency plans for potential environmental impacts and staying informed about local conditions.
Vendor Relationships	Dependency-related issues.	Moderate	Regular vendor assessments, clear contracts, and contingency plans for vendor-related challenges.

5.3 Identification of Project Risks

5.3.1 | Uncertain Project Timeline

- Risk: Unforeseen delays in development.
- Contingency: Build buffer time into the project schedule, regularly monitor progress, and communicate potential delays early to stakeholders.

5.3.2 | Inadequate Resource Allocation

- Risk: Insufficient human or financial resources for planned tasks.
- Contingency: Regularly review resource needs, adjust allocations as necessary, and have contingency plans in place for unexpected resource shortages.

5.3.3 | Scope Creep

- Risk: Uncontrolled changes or expansions in project scope.
- Contingency: Establish a robust change management process, clearly define and document the project scope, and regularly communicate with stakeholders to manage expectations.

5.3.4 | Dependency on External Partners

- Risk: Delays due to dependencies on external vendors or partners.
- Contingency: Clearly outline dependencies, maintain open communication with external partners, and have backup plans or alternative solutions in case of delays.

5.3.5 | Insufficient Testing

- Risk: Inadequate testing leading to post-launch issues.
- Contingency: Implement comprehensive testing plans, including unit testing, integration testing, and user acceptance testing, and allocate sufficient time for testing phases.

5.3.6 | Communication Breakdown

- Risk: Ineffective communication among team members and stakeholders.
- Contingency: Establish clear communication channels, conduct regular status meetings, and use project management tools to ensure everyone is informed and aligned.

5.3.7 | Budget Overruns

- Risk: Exceeding the planned budget.
- Contingency: Regularly monitor expenses, implement strict budget controls, and have a contingency fund for unexpected costs

5.3.8 | Inadequate Risk Management

- Risk: Failure to identify and address risks adequately.
- Contingency: Establish a proactive risk management plan, conduct regular risk assessments, and update risk registers throughout the project.

5.3.9 | Lack of Scalability

- Risk: Inability of the system to handle increased load or future expansions.
- Contingency: Design the system with scalability in mind, conduct performance testing, and have a plan for scaling resources as needed.

5.3.10 | Market Fluctuations

- Risk: Insufficient training for end-users or project team members.
- Contingency: Develop a comprehensive training plan, provide ongoing support, and address any knowledge gaps through additional training sessions

6. RESPONSIBILITIES OF MEMBERS

- All team members collaborated closely, with each individual actively contributing to every aspect of the project. This inclusive approach ensured that each member had a hands-on involvement in various project components, fostering a comprehensive understanding and shared ownership of the entire initiative.

6.1 Position Names of Project Members

6.1.1 | Frontend Roles

- UI/UX Designer: Create user interface (UI) and user experience (UX) designs, ensuring a user-friendly and aesthetically pleasing front end.

- Frontend Developer: Work on improving the appearance and user interaction of the website using technologies such as HTML, CSS, and JavaScript.

- HTML/CSS/Javascript Developer: Develop a dynamic and interactive user interface using a specific frontend framework.

- Test Engineer (Frontend): Test frontend code, identify bugs, and verify the user experience

6.1.2 | Backend Roles

- Backend Developer: Develop the server-side logic of the web application, interact with the database, and execute the business logic.

- Database Administrator: Design, manage, and optimize database operations.

- Security Specialist (Backend): Identify and address security vulnerabilities in the backend, implement security measures.
-

6.1.3 | Reporting Roles

- Data Analyst: Analyze data related to website usage, understand user behaviors, and suggest improvements.

- Reporting Specialist: Prepare regular reports by identifying relevant metrics and performance indicators.

- Business Intelligence (BI) Specialist: Use business intelligence tools to understand business processes and assist in optimizing them.

- Development Process Monitoring and Reporting: Monitor project progress, report on goal achievement, and suggest improvements.

- The scope and responsibilities of each role may vary based on the project's specifications, size, and requirements

6.2 Task Sections Completed by Project Members

6.2.1 | Project Manager

- Develop project timelines and milestones.
- Coordinate team activities and monitor progress.
- Communicate with stakeholders to ensure alignment with project goals.
- Risk management and issue resolution.

6.2.2 | Frontend Developer

- Implement UI/UX designs for the website.
- Write code for the visual elements of the site using HTML, CSS, and JavaScript.
- Ensure the website's responsiveness and cross-browser compatibility.

6.2.3 | Backend Developer

- Develop server-side logic and database interactions.
- Create APIs to facilitate data communication.
- Implement security measures on the backend.

6.2.4 | Database Administrator

- Design and manage the database architecture.
- Optimize database performance.
- Ensure data integrity and security.

6.2.5 | Quality Assurance (QA) Tester

- Develop and execute test plans for the website.
- Identify and report bugs or issues.
- Ensure the website meets quality standards.

6.2.6 | SEO Specialist

- Optimize website content for search engines.
- Conduct keyword research and analysis.
- Implement on-page and off-page SEO strategies.

6.2.7 | Content Manager

- Curate and update content on the website.
- Ensure content aligns with the overall project goals.
- Maintain accuracy and relevance of information.

6.2.8 | IT Security Specialist

- Implement and monitor security measures to protect the website and user data.
- Conduct security audits and vulnerability assessments.

6.2.9 | Vendor Manager

- Manage relationships with external vendors.
- Ensure vendors deliver according to project requirements.

6.2.10 | Risk Manager

- Identify, assess, and manage project risks.
- Implement risk mitigation strategies.

6.3 Schedule of the Project

- Project Schedule: Real Estate Website Development (2 Months)

6.3.1 | Initiation Phase

- Define project objectives and scope (1 week).
- Assemble project team and assign roles (1 week).
- Develop project charter and obtain approvals (1 week).

6.3.2 | Planning Phase

- Gather and analyze business requirements (2 weeks).
- Develop technical specifications (2 weeks).
- Create project plan and schedule (1 week).
- Identify and assess project risks (1 week).
- Finalize project budget and resources (1 week).

6.3.3 | Design & Development Phase

- UI/UX design and approval (2 weeks).
- Frontend development (3 weeks).
- Backend development (3 weeks).
- Database design and development (2 weeks).
- Integration of frontend and backend components (1 week).

6.3.4 | Testing & Deployment Phase

- Unit testing (1 week).
- Integration testing (1 week).
- System testing and debugging (1 week).
- User acceptance testing (1 week).
- Prepare for website launch (1 week).
- Deploy website to production (1 week).
- Conduct final testing and quality assurance (1 week).

6.3.5 | Post-Launch Activities

- Monitor website performance and user feedback (ongoing).
- Implement any necessary updates or improvements (ongoing).
- Conduct training sessions for end-users (1 week).

6.3.6 | Project Closure

- Document lessons learned (1 week).
- Finalize project documentation (1 week).
- Obtain project sign-off and approvals (1 week).

6.3.7 | Notes

- The schedule has been condensed to fit within the 2-month timeframe, focusing on essential tasks for each phase.
- The testing and deployment phase has been combined to optimize the timeline.
- Ongoing activities, such as monitoring website performance and implementing updates, are planned throughout the project.

PROCESS	OCTOBER WEEK 1 & 2	OCTOBER WEEK 3 & 4	NOVEMBER WEEK 1 & 2 & 3	NOVEMBER WEEK 4	DECEMBER WEEK 1 & 2	DECEMBER WEEK 3 & 4	JANUARY WEEK 1	JANUARY WEEK 2
Identifying group members								
Research and Planning		All group members work together.						
Introduction to Frontend & Introduction Part of Report • Task1: Login, Sign Up, Admin Pages | Relational Table • Task2: Main, Detail Pages • Task3: Create, Delete, Update Pages | Form Processing, Responsive Design | E-R Diagram			Task 1 Pages (Efe) Task 2 Pages (Umut) Task 3 Pages (Omer)	Task 1 Diagrams (Efe) Task 3 Diagrams (Omer)				
Introduction to Backend & Requirements Part & System Modelling • Task1: IlanController + SehirController + SemtController | Requirements Part, Use Case Diagram, Timing Diagram, Component Diagram, Deployment Diagram, Interaction Overview Diagram • Task2: HomeController, TipController, MahalleController, OkulController | Activity Diagram, Collaboration Diagram, Composite Structure Diagram • Task3: Database Classes, AccountController, AdminController | State Machine Diagram, Sequence Diagram, Class & Object Diagrams					Task 1 Diagrams (Efe) Task 2 Diagrams (Umut) Task 3 Diagrams (Omer)	Task 1 DB (Efe) Task 2 DB (Umut) Task 3 DB (Omer)		
Testing Part • Task1: Software Testing, Development Testing, Unit Testing, White Box Testing, Integration Testing, Big-bang, Top-down, Bottom-up, Hybrid • Task2: System Testing, Black Box Testing, User Testing/Acceptance Testing, Alpha Testing, Beta Testing, User Acceptance Testing, Operational Acceptance Testing • Task3: Debugging, Debugging by Brute Force, Debugging by Induction, Debugging by Deduction, Debugging by Backtracking							Task 1 (Efe) Task 2 (Umut) Task 3 (Omer)	
Risk Part • Task1: Preparation of the Final report and cover design • Task2: Identification of Project Risks and Project Risk List, Identification of Project Risks, Project Risk List, Planning Risks and Contingencies • Task3: Responsibilities of Members, Position Names of Project Members, Task Sections Completed by Project Members, Schedule of the Project								Task 1 (Efe) Task 2 (Umut) Task 3 (Omer)

YUSUF OMER TURSUN
200101005

EFE EROL
200101066

UMUT KURULUK
210101118

Group No: 16