REQUIREMENTS ANALYSIS

- **Course Information:** ISE 304 SW Engineering
- **Report Name:** Requirements Specification
- **Project Title:** GlobalCargo Connect
- **Group Members:**
    - **Emre Demir** (Software Engineer)
    - **Samet Genç** (Software Engineer)
    - **Abdullah Salih Özgüven** (Data Engineer)
    - **Selim Albayrak** (Product Manager, Marketing Manager)
- **Date:** 28/12/2023

1. INTRODUCTION

1.1 Goal

The goal of this document is to outline the detailed requirements for the development of the mobile application designed to facilitate the exporting and importing processes for companies and traders worldwide. It provides a comprehensive overview of both functional and non-functional requirements, serving as a foundation for the development team and guiding the implementation process.

1.2 Contents and Organization

The document is organized into key sections:

- **Introduction**: Provides an overview of the document's goal and organization.
- **System Requirements:** Enumerates both functional and non-functional requirements, detailing the expected behavior and characteristics of the application.
- **Use Cases:** This section provides detailed user scenarios and interactions, including user types and scenarios such as Ship Owners posting advertisements, Customers searching for shipping services, and interactions between them. The use case diagram and individual use cases offer a comprehensive view of the application's functionalities and user interactions, guiding the development process.

2. SYSTEM REQUIREMENTS

2.1 Functional Requirements

User Authentication:

The system shall allow users to create accounts easily, catering to end-users and customers without technical backgrounds.

The system shall provide a secure login mechanism, ensuring user data protection.

The system should include a forget password functionality for account recovery.

Application Interface:

The user interface shall be designed to be intuitive, ensuring ease of use for end-users.

Navigation between different sections of the application shall be seamless, promoting a user-friendly experience.

A dashboard shall be implemented to provide a summarized view of ongoing shipments, available transportation, and notifications.

Shipment Requests:

Users shall be able to effortlessly create shipment requests through a user-friendly interface.

Shipment requests shall capture essential details such as destination, quantity, and preferred transportation.

Transportation Matching:

The application shall efficiently match shipment requests with available transportation options.

Transportation providers shall be notified promptly of relevant shipment requests.

2.2 Non-Functional Requirements

Security:

User data shall be encrypted during both transmission and storage to ensure data security.

Access to sensitive information shall be restricted based on user roles, enhancing overall system security.

Performance:

The application shall respond to user interactions within a maximum of 2 seconds, optimizing user experience.

The system shall handle a concurrent user load of at least 1000 users, ensuring robust performance.

Usability:

The application shall adhere to established design principles, promoting a positive and intuitive user experience.

Help and support features shall be easily accessible, providing assistance as needed.

Scalability:

The system shall be scalable to accommodate a potential increase in the number of users and transactions.

2.3 General Requirements Guidelines

- Each requirement shall be assigned a unique number for clear identification and traceability.
- The language used in the requirements document shall be consistent, using "shall" for mandatory requirements and "should" for desirable requirements.
- Key parts of each requirement shall be highlighted for easy identification.
- Computer jargon shall be avoided to enhance understanding among users.
- Each requirement shall include an explanation (rationale) outlining why it is necessary, contributing to a comprehensive understanding of the system's needs.

3. USE CASES

3.1 User types

The following definitions describe the actors in the system.

Ship Owners: Users who own ships and have empty cargo space. Can post advertisements to offer their shipping services.

Customers: Users looking to send cargo via available shipping services. Can browse advertisements to find suitable shipping options.

System: The system refers to the computer hardware and software that controls the application. It accepts user input, displays user output, and interfaces to the Web Server through the Internet.

Web Server: Manages user authentication and authorization.

3.2 User Scenarios

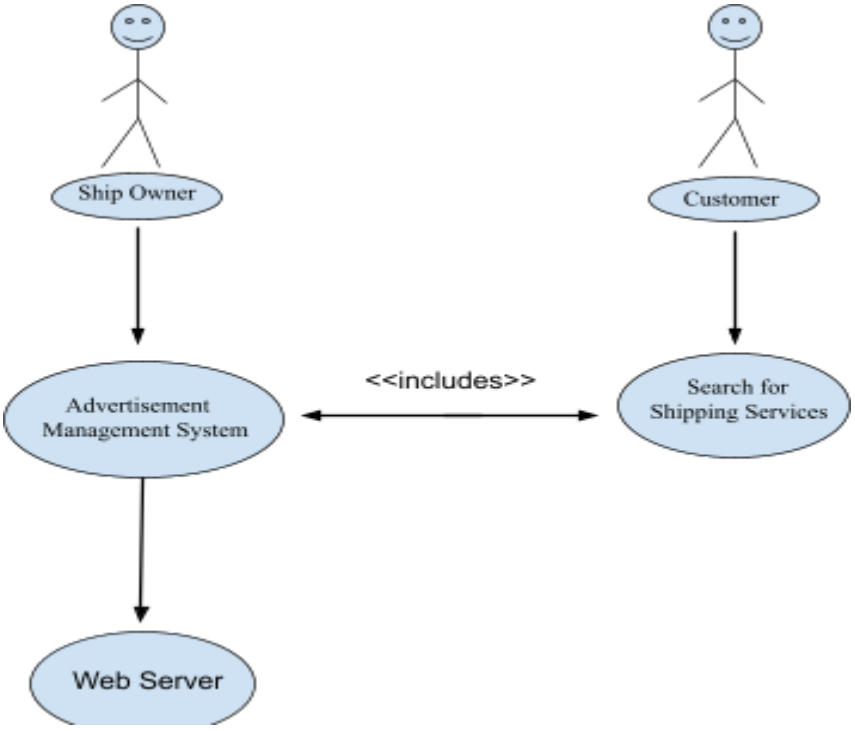
Scenario 1: Ship Owner Posts Advertisement
-Persona: John, Ship Owner
-John, who owns a ship, logs in and posts an advertisement.
-Inputs ship details, contact information, and available cargo space.

-System publishes the advertisement to the platform.

Scenario 2: Customer Searches for Shipping Services
-Persona: Emily, Customer
-Emily, needing to send cargo, searches for available shipping services.
-Uses search filters to find relevant advertisements.
-Reviews ship details and contacts the ship owner.

Scenario 3: Customer Contacts Ship Owner
-Persona: Alex, Customer
-Alex finds a suitable shipping service and contacts the ship owner.
-Initiates communication through the app's messaging system.
-Negotiates terms and finalizes the shipping arrangement.

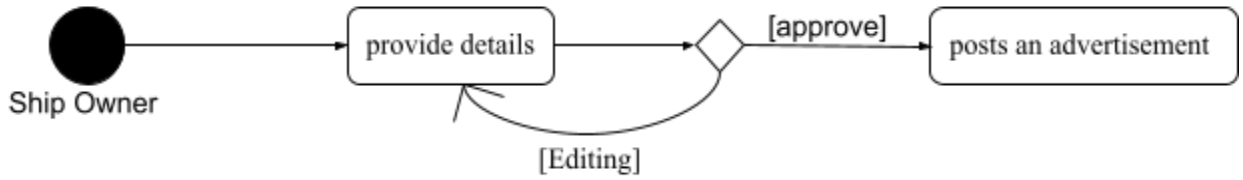
3.3 Use Case Diagram



3.4 Use Cases

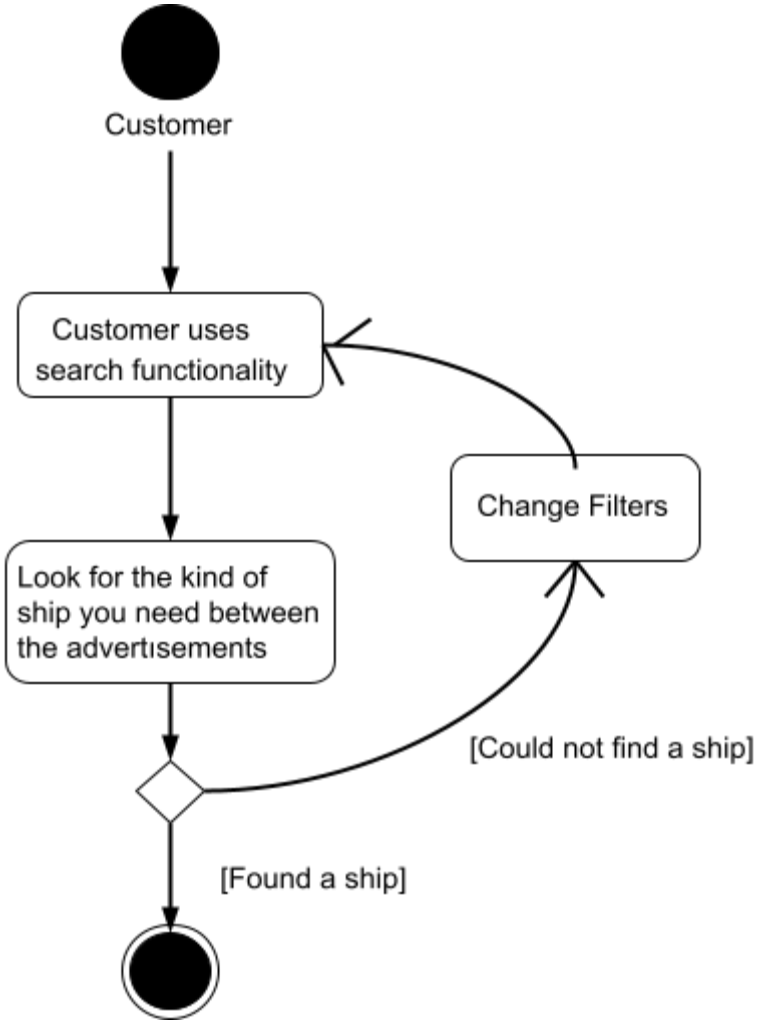
Use Case 1: Post Advertisement

-Primary Actor: Ship Owner
-Ship owner provides details and posts an advertisement.
-Alternative Flow: Editing or deleting an advertisement.

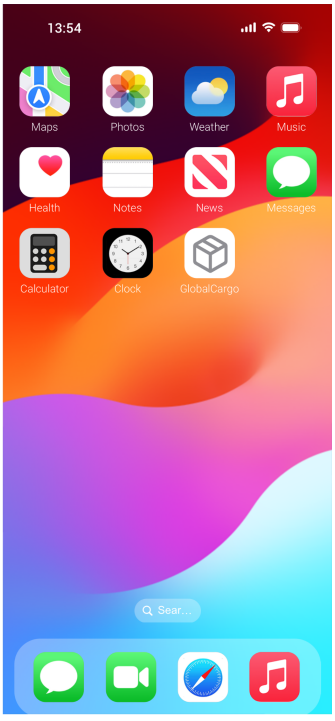


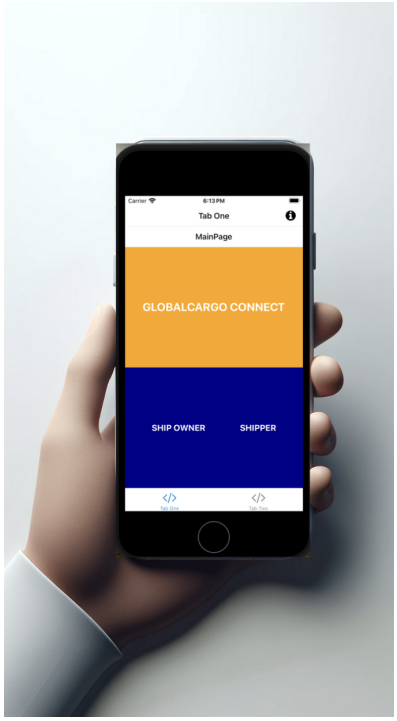
Use Case 2: Search for Shipping Services

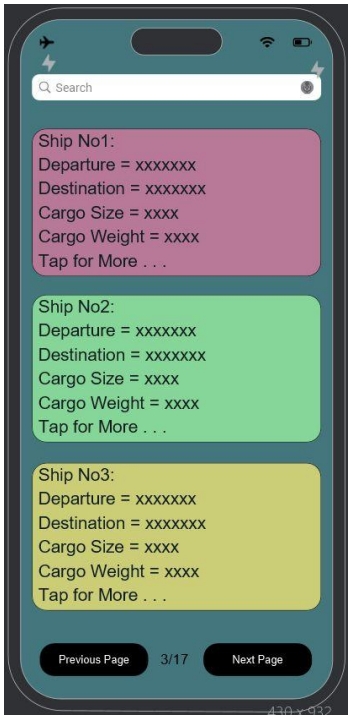
-Primary Actor: Customer
-Customers use search functionality to find relevant shipping services.
-Alternative Flow: Refining search criteria.

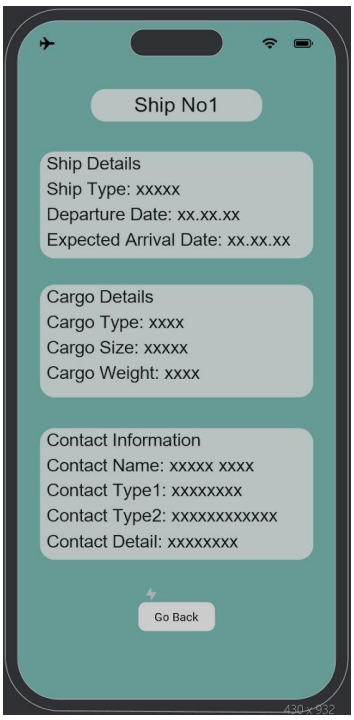

Customer
Customer uses search functionality
Change Filters
Look for the kind of ship you need between the advertısements
[Could not find a ship]
[Found a ship]

4. USER INTERFACE MODEL



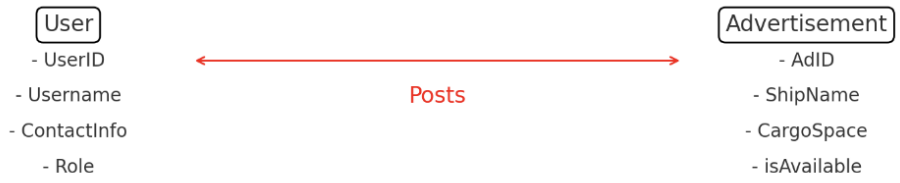






5. FLOW DIAGRAMS

5.1 General Data Model - Entity relationship diagram



5.2 Important Data Considerations

In the development of the GlobalCargo Connect application, several important data considerations are taken into account to ensure the efficient and reliable exchange of information between the client application and backend services, as well as within the client application itself.

Data Format:

- JSON:
  - The primary data format for client-server communication is JSON (JavaScript Object Notation). JSON is a lightweight data-interchange format that is easy for humans to read and write, and easy for machines to parse and generate. It is utilized for its simplicity and native compatibility with JavaScript, enabling seamless serialization and deserialization of data structures.

Data Exchange Protocol:

- HTTP/HTTPS:
  - Data exchange with the server is conducted over HTTP/HTTPS, utilizing RESTful principles. This standard protocol ensures secure communication and is widely supported across different platforms and devices.

Data Management:

- React Context API:
  - For internal state management, the React Context API is used. This enables efficient passing of data throughout the component hierarchy without the need for prop drilling, leading to cleaner code and better performance.
- State Hooks:
  - React's useState hook is employed to encapsulate local state within functional components, allowing for a reactive UI that updates in response to state changes.

Asynchronous Data Handling:

- Promises and Async/Await:
  - Asynchronous operations, such as data fetching from a server, are handled using JavaScript promises and the async/await syntax. This promotes non-blocking UI and a responsive application experience.

Data Persistence:

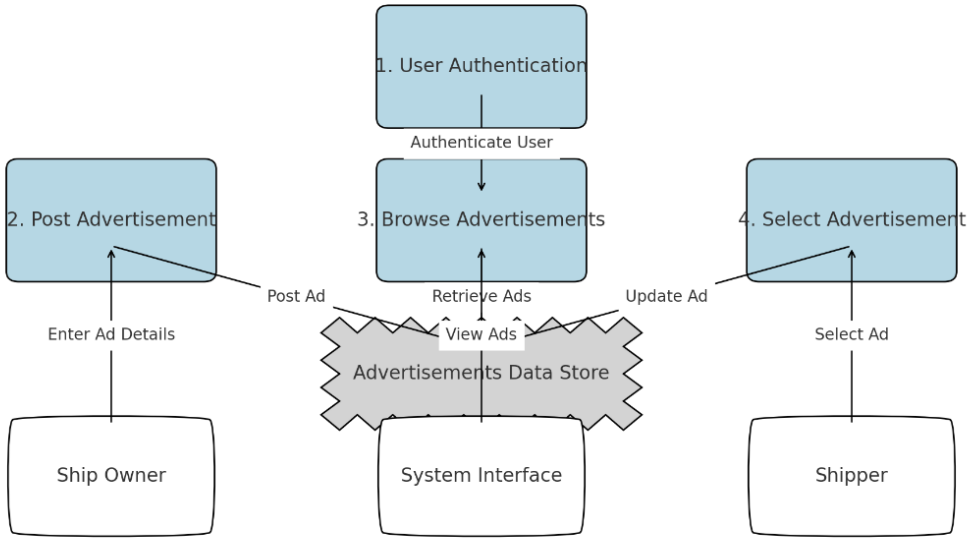
- Local and Session Storage (if applicable):
  - For data that needs to persist beyond the current session or across sessions without requiring a server call, local and session storage mechanisms may be employed, subject to the sensitivity and size of the data.

Error Handling:

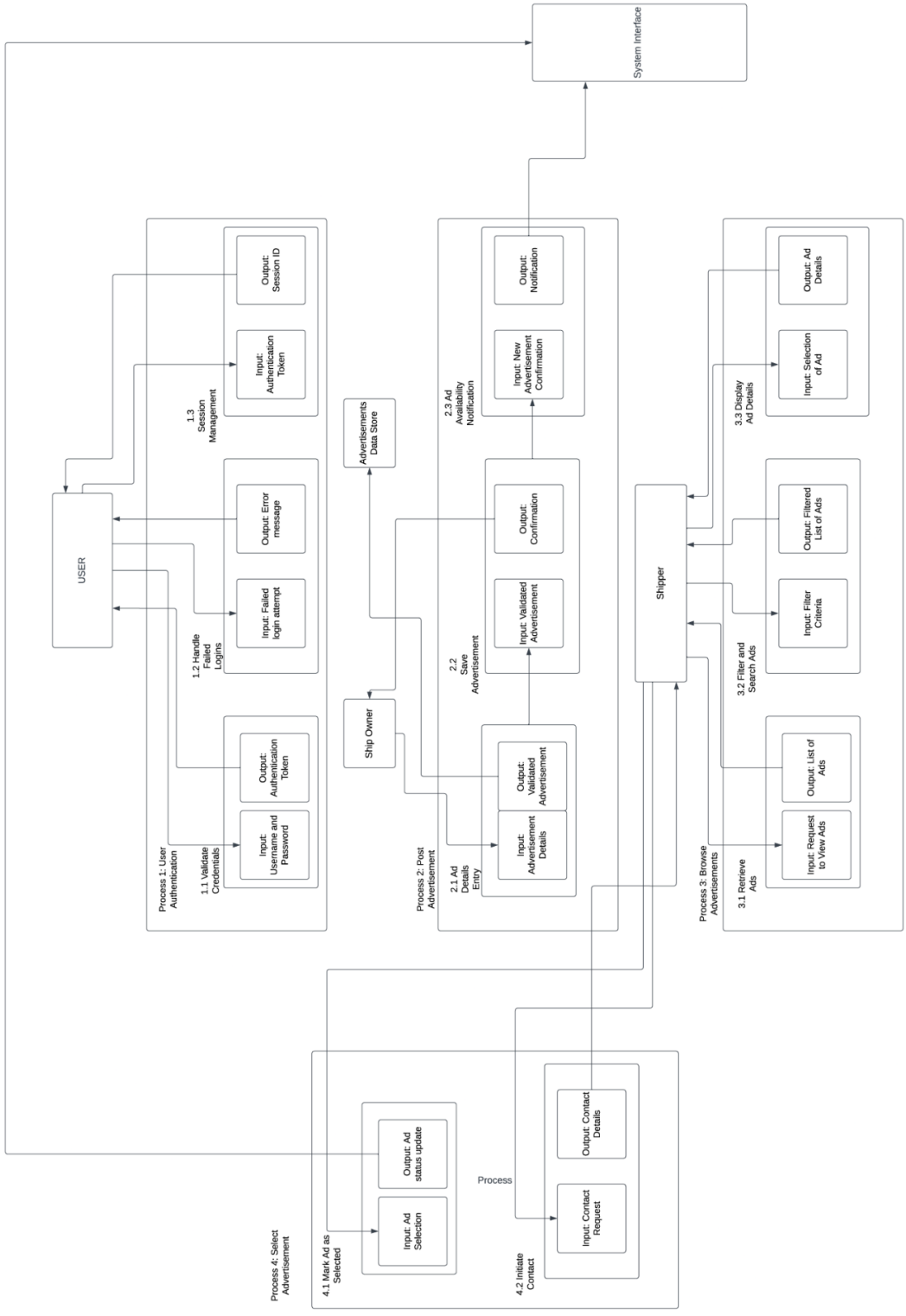
- Robust Error Handling:
  - The application implements robust error handling to manage issues during data fetching, posting, or updating operations. This ensures the application's stability and provides feedback to the user when issues arise.

5.3 Data Flow Diagram

DFD 1



DFD 2


System Interface
USER
Process 1: User Authentication
1.1 Validate Credentials
Input: Username and Password
Output: Authentication Token
1.2 Handle Failed Logins
Input: Failed login attempt
Output: Error message
1.3 Session Management
Input: Authentication Token
Output: Session ID
Advertisements Data Store
Ship Owner
Process 2: Post Advertisement
2.1 Ad Details Entry
Input: Advertisement Details
Output: Validated Advertisement
2.2 Save Advertisement
Input: Validated Advertisement
Output: Confirmation
2.3 Ad Availability Notification
Input: New Advertisement Confirmation
Output: Notification
Shipper
Process 3: Browse Advertisements
3.1 Retrieve Ads
Input: Request to View Ads
Output: List of Ads
3.2 Filter and Search Ads
Input: Filter Criteria
Output: Filtered List of Ads
3.3 Display Ad Details
Input: Selection of Ad
Output: Ad Details
Process 4: Select Advertisement
4.1 Mark Ad as Selected
Input: Ad Selection
Output: Ad status update
Process
4.2 Initiate Contact
Input: Contact Request
Output: Contact Details