****

**Software Requirements Specification**

**ST{AU}CKTION**

**Team 10**

**Ahmet Oğuz Ergin, 21903070**

**Ayşe Özge Erkan, 21902107**

**Bertan Özer, 21902475**

**Damla Köklü, 22003189**

**Göktuğ Yeşilyurt, 22003697**

Bilkent University

Department of Information Systems and Technologies

09.11.2024

Change History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **File Name** | **Document Type** | **Deliverable** | **Version** | **Submission Date** |
| Deliverable\_2.docx | MS Word 2024 | 2 | 1 | 09.11.2024 |
| Team10\_SRS\_FR1.docx | MS Word 2024 | 2 | 2 | 03.12.2024 |

Team 10

|  |  |  |
| --- | --- | --- |
| **Name, Surname** | **Student Id Number** | **e-mail** |
| Ahmet Oğuz Ergin | 21903070 | oguz.ergin@ug.bilkent.edu.tr |
| Ayşe Özge Erkan | 21902107 | ozge.erkan@ug.bilkent.edu.tr |
| Bertan Özer | 21902475 | bertan.ozer@ug.bilkent.edu.tr |
| Damla Köklü | 22003189 | damla.koklu@ug.bilkent.edu.tr |
| Göktuğ Yeşilyurt | 22003697 | goktug.yesilyurt@ug.bilkent.edu.tr |

Project Details

|  |  |
| --- | --- |
| **Project Name** | **ST{AU}CKTION** |
| **Software Name** | **ST{AU}CKTION** |
| **Company Name** | **No Sponsor Company** |
| **Academic Advisor** | **Dr. Cüneyt Sevgi** |
| **Github URL** | **https://github.com/staucktion** |
| **WEB page** |  |

Individual Contributions Overview

|  |  |
| --- | --- |
| **Name, Surname** | **Summary of Contributions to the SRS Document** |
| Ahmet Oğuz Ergin | Main responsible of the Functional requirements, Non-functional Requirements sections. Formatting the document. |
| Ayşe Özge Erkan | Main responsible of the User Characteristics, System Model sections. |
| Bertan Özer | Main responsible of the Product Perspective, and Verification sections. |
| Damla Köklü | Main responsible of the Scope, Assumption and Dependencies sections. |
| Göktuğ Yeşilyurt | Main responsible of the Executive Summary, Requirements Prototypes and Logical Database Requirements sections. |

Executive Summary

The Software Requirements Specification for our project "St{au}cktion" shows the software related requirements of the project that we are planning to build, which is mainly planned to be a combination of Android mobile application and a web application. The software is planned to be used as an application that will ease the process of selling a photograph to stock photograph companies through an auction system while also involving a user community to make it easier for stock photograph companies to quickly identify the most liked photographs. The app is planned to include integration with PostgreSQL for secure database management, simulated banking for reliable financial transactions and to make it easier for showcasing the demo of the application, and location services through the Google Maps API for enhanced photo authenticity and security. The prototype focuses on two of the main functionalities, which are taking a photograph and validating a photograph, providing an early glimpse into the system’s architecture. One of the interfaces is from the mobile application to take the photograph, and the other one is from the web application's photograph validation dashboard. These interfaces were prototyped to make it easy to use for different user types including photographers, stock photo companies, validating agents, and system administrators to ensure a purpose oriented and easy to learn user experience. Several use cases have been analyzed to reflect real world scenarios such as photographers uploading and managing the photographs they took, companies participating in auctions, and validating agents approving photographs. These use cases make sure that our product will have a system that relates to the existing context of the stock photography market. Major functional components are detailed, such as the auction dashboard, user interface design, and photo management, supported by UML diagrams and prototyping insights. Non-functional requirements emphasize usability, system performance, and security, including standards like TLS encryption for data protection. The SRS also addresses challenges in user experience and performance optimization, setting a foundation for successful implementation while ensuring compliance with data privacy regulations.[[1]](#footnote-1)

Table of Contents

**ii**

[Executive Summary i](#_Toc184151655)

[Table of Contents ii](#_Toc184151656)

[1. Scope 1](#_Toc184151657)

[2. Product Perspective 3](#_Toc184151658)

[3. User characteristics 11](#_Toc184151659)

[4. Assumptions and dependencies 13](#_Toc184151660)

[5. Functional requirements 15](#_Toc184151661)

[a. Epics of User Stories 15](#_Toc184151662)

[b. User Stories 15](#_Toc184151663)

[b. Functional Requirements 17](#_Toc184151664)

[c. Use Case Diagram 35](#_Toc184151665)

[d. Requirement Associations 36](#_Toc184151666)

[6. System Model 39](#_Toc184151667)

[7. Requirements Prototypes 44](#_Toc184151668)

[a. UC\_19 Take Photograph 44](#_Toc184151669)

[b. UC\_22 Validate Photograph 45](#_Toc184151670)

[8. Non-functional Requirements 48](#_Toc184151671)

[1. Usability requirements 48](#_Toc184151672)

[2. Performance requirements 48](#_Toc184151673)

[Capacity Requirements 48](#_Toc184151674)

[Dynamic Requirements 48](#_Toc184151675)

[3. Software system attributes 49](#_Toc184151676)

[4. Constraints 50](#_Toc184151677)

[5. Error Handling Requirements. 50](#_Toc184151678)

[User Errors 50](#_Toc184151679)

[External System Errors 50](#_Toc184151680)

[Developer Errors 50](#_Toc184151681)

[6. Other Non-Functional Requirements 51](#_Toc184151682)

[9. Logical Database Requirements 52](#_Toc184151683)

[10. Verification 58](#_Toc184151684)

[11. Discussions 60](#_Toc184151685)

[1. Limitations and Constraints 60](#_Toc184151686)

[2. Health and Safety Issues 60](#_Toc184151687)

[3. Legal Issues 60](#_Toc184151688)

[4. Economic Issues and Constraints 60](#_Toc184151689)

[5. Sustainability 60](#_Toc184151690)

[6. Ethical Issues 60](#_Toc184151691)

[7. Multidisciplinary Collaboration 60](#_Toc184151692)

List of Tables

**iv**

[Table 1: User Characteristics 11](#_Toc184151693)

[Table 2: Cross Referencing Functional Requirements to Use Cases 36](#_Toc184151694)

[Table 3: Relationships Between Entities 54](#_Toc184151695)

List of Figures

**v**

[Figure 1: Use Case Diagram 35](#_Toc184151702)

[Figure 2: Take Photo Activity Diagram 39](#_Toc184151703)

[Figure 3: Take Photo Class Diagram 40](#_Toc184151704)

[Figure 4: Take Photo Sequence Diagram 41](#_Toc184151705)

[Figure 5: Validate Photo Activity Diagram 42](#_Toc184151706)

[Figure 6: Validate Photo Class Diagram 42](#_Toc184151707)

[Figure 7: Validate Photo Sequence Diagram 43](#_Toc184151708)

[Figure 8: Take Photograph – Initial State 44](#_Toc184151709)

[Figure 9: Take Photograph – Error State 45](#_Toc184151710)

[Figure 10: Validate Photograph – Initial State 46](#_Toc184151711)

[Figure 11: Validate Photograph – Rejection Reason Modal 47](#_Toc184151712)

[Figure 12: EER Diagram of St{au}cktion’s Database Using Chen’s Notation 57](#_Toc184151713)

Abbreviations

|  |  |
| --- | --- |
| API | Application Programming Interface |
| CCPA | California Consumer Privacy Act |
| CPU | Central processing unit |
| CRUD | Create, Read, Update, and Delete |
| CSS | Cascade Style Sheets |
| CTIS | Computer Technology and Information Systems |
| DBMS | Database Management System |
| EBS | Elastic Block Store |
| EER | Enhanced Entity Relationship |
| EP | Epic |
| GDPR | General Data Protection Regulation |
| GB | Gigabyte |
| GPS | Global Positioning System |
| GUI | Graphical User Interface |
| HTML | HyperText Markup Language |
| HTTP | HyperText Transfer Protocol |
| HTTPS | HyperText Transfer Protocol Secure |
| ID | Identifier |
| IP | Internet Protocol |
| JSON | JavaScript Object Notation |
| MB | Megabyte |
| RAM | Random Access Memory |
| SDK | Software Development Kit |
| SRS | Software Requirements Specification |
| SQL | Structured Query Language |
| TDPA | Turkish Data Protection Authority |
| TLS | Transport Layer Security |
| UC | Use Case |
| UI | User Interface |
| UML | Unified Modeling Language |
| vCPU | Virtual Central Processing Unit |

# Scope

The software to be developed is called "St{au}cktion," a web-based application focused on photo uploading, location-based categorization, and photo sales, as stated in use-case diagrams, and requirements. Software products that will be used in this project are Android, Host DBMS created by PostgreSQL, Simulated Banking Application, and Web User Interface written in web application languages, HTML, CSS, and JavaScript. The Three software Products are as follows.

**Host DBMS:** Application’s main database that holds a class for member’s information (Name, surname, e-mail, address, payment information), and a class for holding the photo information (Image id, like count, comment count, member id of the person who is uploaded the photo). The goal of creating a Host DBMS is holding the sensitive information related to the application in one structured, secure database environment.

**Simulated Banking Application:** As St{au}cktion project includes sensitive bank transaction. A simulated banking application will be produced.  This software product ensures secure money transactions. Companies will buy images and users will earn money. Banking application is necessary in this system. Therefore, a simulated banking application will be generated for handling money transactions in the app. Please note that this application will only hold a simple database and handle money transactions for our system.

**Web User interface:** This will be the main web application that will be in relation to Host DBMS, and simulated banking application. This will perform operations related with Photo upload delete, and auction referring the use cases in SRS document. This software product will be produced with JavaScript, HTML, CSS, and Sharp, Multer, React libraries.

**Web API:** Web Api is the software product that will be produced for enabling communication of the web client with the application itself. It acts as a bridge between the user interface and backend applications. This will be produced with the JavaScript and libraries that are used in the production of Web User Interface. Therefore this product is a fundamental part of the project as it involves all the key operations related with the database management, and regular site operations such as bidding, auction management, and voting.

**Android:** The android software product is going to enable users to use the application through mobile devices. Java and Kotlin languages will be used to develop this product. This product is the mobile version of the web user interface.

# Product Perspective

St{au}cktion is a platform for stock photography similar to platforms like Getty Images. However, St{au}cktion differentiates itself from these platforms by serving as a digital marketplace where photographers can auction their stock photography to end users. In order to accomplish/facilitate these auctions, the system depends on components such as payment processing, photo storage, and location services for photo uploading and verification. Other systems are integrated to fulfill these dependencies.  Like a banking API to ensure a seamless and secure monetary transaction process for photographers and stock image companies, and the Google Maps API to enable the location features of the system.

-Banking API: The system will be connecting to a banking service to handle the secure transactions between photographers and the companies. The API will transfer payment information between photographers and end users buying stock images, ensuring that earnings and commissions, like our payment, are correctly processed.  In order to avoid any problems with official banking APIs during, and to make things easier for us, we will not be using any official banking API’s but instead using a simulated API we will develop to make basic CRUD operations and simulate transactions between photographers and companies.

-Google Maps API: This API will be used to get location data necessary to the core functionality of the app. It will be used to show where a stock image was taken, and additionally for security purposes as the location data will be used to limit the number of photos that are taken in a surrounding area.

-Node.js: Node.js will act as the backend server of the project. It will process front-end requests and interact with services like our database, PostgreSQL. For the PostgreSQL connection TCP/IP protocols will be used, while the other interactions, like with APIs, will be done over HTTPS.

-Google Maps API: Google Maps API will be used in the project for tasks that require precise location, like one of our security measures to avoid photo upload spamming. The application will send the requests through REST API and receive the results in JSON format.

-PostgreSQL: PostgreSQL used for data storage purposes in the project. Data such as photo metadata, user data, and financial data for the banking API simulation will be stored using it. SQL queries will be used for communication over TPC/IP protocol. Connection strings are used for authentication and data exchange purposes.

1. **System interfaces**

-Android: Android OS will be an essential system for the project. Enabling us access to device hardware and software such as the phone’s camera and the GPS. With these interactions the app will be able to fulfill key requirements. The app will interface with the Android OS using Android SDK and APIs to access the camera, get location data, and have access to storage.

1. **User interfaces**

-Auction Dashboard:

Characteristics: On the website the auction dashboard features a grid layout showing images on auction with high quality thumbnails. Each thumbnail includes the location of where the photo was taken, current bid, the name of the user who took the photo and their profile picture displayed next to it, and a button for bidding. There is a sidebar navigation menu to filter the images being shown by different criteria. These criteria will be landmarks like the Eiffel Tower. Upon pressing the bidding button, a different page will load.

On the mobile app the layout of the Auction Dashboard will remain similar. The interface will keep the grid layout, and key characteristics such as the navigation menu and the filters will remain, but they will be accessed differently. Users will be able to access these filters by pressing a filter button near the top of the screen. Upon pressing a dropdown menu will reveal the same navigation menu from the website. In the app there will be no bidding button instead users will press the image and the surrounding box to go to the bidding page.

Optimization: The dashboard is optimized with clear, concise labels to maintain readability and navigation. Error messages will be utilized to communicate errors should they occur.

-User settings:

Characteristics: User settings will have a tabbed layout for quick access to different categories such as “Privacy Settings” and “Account Information”.  In each tab a two-column layout will be used to display options on the left (e.g., “Password”) details on the right. This way the user will be able to easily navigate between options under the different categories and find specific settings without scrolling excessively. After making changes a pop-up window will ask for confirmation and ask the user if they want to commit these changes.

Optimization: Tooltips/hints will be available near textboxes to give users information. After critical changes a prompt will ask for confirmation and ask the user if they want to commit these changes to avoid unintended changes.

1. **Hardware interfaces**

-Mobile Devices:

Characteristics: The application supports Android devices. It accesses GPS location data to enhance the accuracy of location-based features. Additionally, the app will require access to the phone’s camera for users to take pictures.

Configuration: GPS must be enabled for image uploads as the app requires location data for functionality. Similarly, the app’s access to the camera must be enabled for basic functionality like taking photos and the app must support both portrait and landscape orientations.

-Amazon Web Services Hardware Details:

Characteristics: The application will be using an Amazon Web Services’ cloud server for different tasks, such as using the capabilities of Lambda for functions like image processing.

Configuration: Since Lambda allocates CPU power proportional to the amount of memory provisioned, we will have access to up to 6 vCPUs (virtual CPUs). RAM for the server will be allocatable as well with access to configurable memory from 128 MB to 10,240 MB. In addition, the amount of CPU power is proportional to the memory allocation. Meaning that functions with more memory allocated to them will have access to more CPU power. When it comes to ports and protocols, Lambda can TCP/IP and HTTPS protocols for communication with different services and systems involved with the project.  Communication with the PostgreSQL database will be done using TCP/IP protocol while HTTPS is used for API calls to necessary third-party services. An advantageous aspect of Lambda is that it does not require direct interaction with physical hardware, instead relying on AWS infrastructure for its processing power, storage, and networking.

1. **Software interfaces**

-PostgreSQL 16:

Purpose: This interface enables the system to store and retrieve all user data, image metadata, and transaction data from the PostgreSQL database. It ensures that all data related to the system will be maintained in a secure environment and allow for effective data management.

Interface: We will use TCP/IP protocol and remotely access the database using a connection string. The results will return in JSON format.

 -Google Built in authentication tools:

 Purpose: This interface enables the system to authenticate users and authorize access to their accounts when logging in.

Interface: API calls will handle authentication tokens and session management in JSON format.

-Android OS:

Purpose: This interface allows the associated app to interact with the Android operating system to gain access to essential system features such as GPS and the camera hardware/software.

Interface: Android’s built-in libraries will be integrated to access essential system features like the GPS and camera, with requests managed in JSON format when handling location data.

-Banking API:

Purpose: This will be a simulated API that will be used to simulate an API facilitation the secure monetary transactions between photographers and end users purchasing stock images. It will achieve this by making simple CRUD operations that will effectively serve as a simplified version of payment transfers and commission distribution.

Interface: Since it is a simulated app and not an official API, HTTP or HTTPS will be used for secure communication and to get simple CRUD operations done to simulate financial transactions.

-Google Maps API:

Purpose:  This API will be used to get location data necessary to the core functionality of the app. It will be used to show where a stock image was taken, and additionally for security purposes as the location data will be used to limit the number of photos that are taken in a surrounding area.

Interface: REST API will be used to request geographic coordinates. The results will be formatted in JSON.

-Nginx-1.26.2:

Purpose: Nginx will be used as the web server for the project. It will control how web users access the hosted files.

Interface: nginx’s requests tend to be handled over HTTP, however for the additional security it provides we will be using HTTPS to handle the requests instead.  With this interface we will be able to forward users to the appropriate content.

-Node.js:

Purpose: Node.js will act as the backend server of the project. It will process front-end requests and interact with services like our database, PostgreSQL. For the PostgreSQL connection TCP/IP protocols will be used, while the other interactions, like with APIs, will be done over HTTPS.

Interface: When it comes to the interfacing of Node.js and the nginx web server, that will be handled via either HTTP or HTTPS to exchange data.  The results will be formatted in JSON. With the PostgreSQL database interface, we will handle that via TCP/IP.

-Windows OS:

Purpose: The operating system that will be used by the members of the team who use computers with Microsoft Windows.

 Interface: Development tools such as Android Studio, Postman, and Node.js runtime will be installed and run-on Windows. The operating system will provide access to features like file storage, networking, and local database testing tools.

-MacOS:

Purpose: The operating system that will be used by the members of the team who use Apple’s Mac laptop/desktops.

Interface: Development tools such asAndroid Studio, Postman, and Node.js runtime will be installed and run-on MacOS. The operating system will provide access to features like file storage, networking, and local database testing tools.

-Debian GNU/Linux 11:

Purpose: Debian is the operating system being used for the Nginx web server and the PostgreSQL database server.

Interface: When it comes to interfacing the machine running on Debian will interface with Node.js. This interface will consist of Debian serving as the infrastructure for hosting Nginx and PostgreSQL, both of which interface with Node.js for the backend operations of the project.

1. **Communications interfaces**

-HTTPS: The platform supports HTTPS for all communications to ensure data encryption and security. Additionally, this is how the mobile app component on Android devices will communicate with the servers.  All API requests from the mobile app to the backend will be made using HTTPS as well for security and data protection.

-TPC/IP: This protocol is used by the project to communicate with the PostgreSQL database. The backend will connect to the database through TCP/IP using a connection string that includes the database’s host, port, username, and password. Allowing the backend to execute SQL queries for data retrieval or manipulation.

-Android: Using Android SDK and related APIs, the mobile app is able to access hardware and software features of Android phones. In the case of our project these features are specified to location services, the phone’s camera, storage to be able to save the watermarked device and for photo uploading, and push notifications to alert the user.

1. **Memory constraints**

-Phones:

Primary Memory (RAM): A minimum of 4 GB of RAM is required for the Android phones. We came to this decision after members of the team started doing tests with the basic functionality of the mobile app. Additionally, we checked the RAM of phone models released within the last 8 years, and it showed that 4 GB was a good baseline for the RAM.

Secondary Memory: When it comes to the secondary memory of the phones, after examining apps that do similar functions to ours with photo caching and other operations, it became clear that a size of around 500 MB to 5 GB seems appropriate.

-Amazon Lambda:

Primary Memory (RAM): With AWS Lambda functions have a configurable memory from 128 MB to 10,240 MB. We can use this configurability to allocate memory to different functions. The more key functions of the project database interaction will be given more memory. Additionally, with Lambda different instances run independently from each other so multiple requests will get their own memory allocation.

Secondary Memory: Lambda provides us with 512 MB of temporary storage that will be used for execution tasks.

-Nginx:

Primary Memory (RAM): Nginx does not require too much memory to function. 2 GB of memory will be allocated for it to serve this project.

Secondary Memory: Nginx uses minimal disk storage for temp files. Therefore, secondary memory usage will not be a big factor.

-PostgreSQL:

Primary Memory (RAM): The minimum requirement for PostgreSQL functionality is 2 GB. However, to give ourselves more freedom and avoid future problems we will be allocating 8 GB of RAM for PostgreSQL.

Secondary Memory: The database will be stored using Amazon services, like EBS, that qualify for this task. Our goal is to ensure that it can efficiently process multiple concurrent queries, store large datasets, and manage transaction logs.

# User characteristics

Understanding and analyzing user characteristics is a critical step in successful project development. The table below offers a comprehensive evaluation of various user groups within the stock photography market, examining their educational backgrounds, levels of experience, accessibility needs, technical proficiency, and motivations. These user groups include photography market stakeholders (such as advertisers, marketers, and design agencies), independent buyers (like small business owners and individuals), and stock photographers. By recognizing the unique needs and preferences of each group, the table highlights the importance of designing platform features and interfaces that are inclusive and customized to diverse user profiles, ultimately enhancing user engagement and satisfaction.

Table 1: User Characteristics

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User Group | Educational Level | Experience | Disabilities | Technical Expertise | Motivation |
| Photography Market Stakeholders (Advertisers & Marketers, Design Agencies, Independent Buyers, Stock Photographers, Stock Picture Companies) | Primarily educated in media, design, or marketing, though education level may vary across user groups. | Moderate to high experience; generally accustomed to regularly using stock photo sites. | Accessibility features may be needed for visual or cognitive disabilities. | Varying levels of technical knowledge; some users expect advanced interfaces, while others prefer simpler ones. | Access high-quality visuals without subscription costs, maintain control over content creation and distribution, discover popular and unique images. |
| Independent Buyers | Varies; could include small business owners, bloggers, or individuals. | Little to moderate experience with stock photos . | Accessibility considerations for visual and cognitive disabilities. | Basic to moderate; needs intuitive interface. | Obtain high-quality images quickly without high subscription costs. |
| Stock Photographers | Ranges from self-taught individuals to those with formal training in photography. | Moderate to high experience in photography, some may already sell on stock sites. | Accessibility considerations for visual and physical disabilities. | Varies; some are very tech-savvy, others may prefer simpler interfaces. | Earn revenue by selling their images directly, maintain control over pricing and distribution. |
| Validating Agent | Technical expertise. | Experience in content moderation or quality assurance | Accessibility considerations are minimal, but ergonomic tools or visual aids can assist agents with physical or visual disabilities. | Strong technical expertise and often use specialized software to ensure compliance with platform policies. | Maintain the platform's credibility by approving high-quality, policy-compliant images.[[2]](#footnote-2) |

# Assumptions and dependencies

There are several factors that can affect the requirements demonstrated in Functional Requirements, and nonfunctional requirements section in this SRS document.

**Assumption 1:** It is assumed that the database we will create will not require additional money from the application. For thisproject, the Host DBMS is expected to store a good amount of data. The software, which will be used to store data, requires an additional price, as the size of the database increases, and operations get more complex. The Database management system is needed to be updated to a simpler form if software used for database requires additional money.  This change can affect the functional requirements that involve database management, like FR2.1. and EP4-US1 to EP4-US7 are also affected, due to its high database requirements in photo management. If this problem occurs, the team will purchase the database at a suitable price.

**Assumption 2:** It is assumed that Integration with Other Systems will be performed without compatibility issues. If APIs that will be used for location verification and map operations are not made compatible with the systems, Requirements starting from uploading images, can be affected. The user stories, EP3-US1, EP3-US2, EP3-US3, will be affected if this assumption is not assumed correctly. If this problem occurs, system architecture can be updated, by generating new APIs in the project.

**Assumption 3:** It is assumed that system will not have any problems related with network. Meaning, network will be reliable and consistent. Therefore, it is assumed that there will be no Concurrency Problems. Meaning, multiple users bidding simultaneously could lead to race conditions, and bids are processed with the right order. If users experience inconsistent network speeds or interruptions, functionalities like real-time bidding can be affected. If this problem occursEP5-US1, EP4-US7, can be affected. These stories are related with keeping the auction time, and voting for photos. If there is a problem with the network operations, the team will consider buying enterprise networking tools.

**Assumption 4:** It is assumed that the cloud service provider, Google Cloud Free tier will maintain high uptime and availability. Downtime or maintenance by the provider could lead to service disruptions, affecting critical functionalities such as database access, user authentication, and auction management. Therefore, EP1-US2 (login functionality), EP4-US4 (photo upload), and EP5-US1 (auction time management) would be heavily impacted by these disruptions. If this problem occurs, the team will upgrade its service provider with additional costs.

# Functional requirements

a. Epics of User Stories

**EP-1:** Authentication and authorization

**EP-2:** User management

**EP-3:** Location management

**EP-4:** Photo management

**EP-5:** Auction management

**EP-6:** Purchase management

**EP-7:** Notification management

b. User Stories

**EP1-US1:** As a guest user, I want to register to system so that I will use the system functionalities which registered users can perform.

**EP1-US2:** As a registered user, I want to login to a system so that I can authenticate and perform functionalities which need authorization.

**EP1-US3:** As a registered user, I want to logout so that I can either login with a different account, register a new one, or simply end my session.

**EP2-US1:** As a system admin, I want to manage all user information so that I can ensure accuracy and enforce policies, such as removing inappropriate information.

**EP2-US2:** As a registered user, I want to manage my own information so that I can correct any outdated or incorrectly entered details.

**EP3-US1:** As a system admin, I want to manage location categories so that I can ensure accuracy and enforce policies, such as removing inappropriate or inaccurate information.

**EP3-US2:** As a validating agent, I want to approve location category requests so that the new location category will be accurate and appropriate.

**EP3-US3:** As a registered user, I want to make a request to create a new location category for a photo upload so that I will be able to upload photos for that category which does not exist.

**EP3-US4:** As a registered user, I want to display a map and click on location category markers on the map so that I can view and vote on the watermarked photos.

**EP4-US1:** As a system admin, I want to manage all watermarked photos so that I can ensure proper control, monitoring, and enforcement of policies regarding their display and sale.

**EP4-US2:** As a validating agent, I want to approve photos so that they can be included in the auction and made available for stock image companies to bid on.

**EP4-US3:** As a photographer, I want to take a photo using the system's user interface (instead of the default camera application) while GPS is on so that the photo's location cannot be mimicked.

**EP4-US4:** As a photographer, I want to upload a photo to the system so that the system can place it in the auction with a watermark.

**EP4-US5:** As a photographer, I want to manage my uploaded photos so that I can update, delete, or review them as needed before they are included in the auction.

**EP4-US6:** As a registered user, I want to display watermarked photos using a location filter so that I can view photos based on specific geographic areas.

**EP4-US7:** As a photographer, I want to vote for the watermarked photo of other users uploaded so that I would earn money if it the photo selected in the auction.

**EP5-US1:** As a timer, I want to start and end the auction at specified times so that the auction process is effectively managed.

**EP5-US2:** As a system admin, I want to manage auction so that I can ensure accuracy and enforce policies, such as terminating auction with inappropriate situation.

**EP5-US3:** As a stock image company, I want to bid for a watermarked photo so that I can purchase it at the end of the auction.

**EP5-US4:** As a stock image company, I want to purchase a photo so that I receive a watermark-free version with full rights.

**EP5-US5:** As a photographer, I want to receive a 10% commission for voting on photos that make it into the top 10%, so that I am rewarded for my participation in the voting process.

**EP5-US6:** As a system administrator, I want the system to automatically retain 20% commission from the payment of a photo purchase, so that the platform can take its share from successful transactions.

**EP5-US7:** As a photographer, I want to receive 70% of the payment for the sale of my photo, so that I am fairly compensated for my work.

**EP6-US1:** As a photographer, I want to assign a value to my photo even if it is not in the top 5% voted photos, so that stock image companies can still purchase it, allowing me to earn a profit.

**EP6-US2:** As a stock image company, I want to view photos that are not in the top 5% of votes, so that I can evaluate and purchase them as well.

**EP7-US1:** As a registered user, I will display system notifications so that I can display important information such as auction result gains.

b. Functional Requirements

**EP1:**

**FR-1.0.0:**

**Description:** The system shall display an app bar and an image slider to guest users.

**Acceptance Criteria:** App bar and image slider are visible to guest users.

**FR-1.1.0:**

**Description:** The system shall display an app bar at the top of the page with two buttons, "Register" and "Login," for guest users to access further features.

**Acceptance Criteria:** App bar shows "Register" and "Login" buttons for guest users.

**FR-1.2.0:**

**Description:** The system shall display a slider showcasing user comments to guest users.

**Acceptance Criteria:** Comment slider is visible to guest users and rotates through comments.

**FR-2.0.0:**

**Description:** The system shall direct guest users to the registration page upon clicking the "Register" button.

**Input:** Guest user clicks the "Register" button.

**Output:** Registration page is displayed.

**Acceptance Criteria:** Clicking the "Register" button redirects guest users to the registration page.

**FR-2.1.0:**

**Description:** The system shall allow guest users to register by entering their information and clicking the "Register" button.

**Input:** Guest user enters required information and clicks "Register."

**Output:** User is directed to the home page.

**Acceptance Criteria:**

* Guest user is successfully redirected to the home page after registration.
* System admin display information about newly registered user.

**FR-2.1.1:**

**Description:** The system shall validate user input during the registration process.

**Input:**

* User type: Radio button with 2 values as photographer and stock image company, required.
* Name: Up to 32 alphabetic characters, required.
* Surname: Up to 32 alphabetic characters, required.
* Email: Must be in a valid email format, required, unique.
* Password: Must be at least 8 characters long up to 32, contain at least one uppercase letter, one lowercase letter, one digit, and one special character (e.g., @, #, $, %), required.
* Phone number: Must consist of 10 digits, required.
* Credit card information: Must follow standard credit card format (e.g., 16 digits, valid expiration date, CVV), required.

**Output:**

* Success color for valid inputs
* Error color for invalid inputs.

**Acceptance Criteria:**

* All fields must accept only the specified formats and lengths.
* Valid inputs are visually indicated with a success color.
* Invalid inputs are visually indicated with an error color and display error message.
* Users cannot proceed with registration until all inputs are valid.

**FR-3.0.0:**

**Description:** The system shall direct guest users to the login page upon clicking the "Login" button.

**Input:** Guest user clicks the "Login" button.

**Output:** Login page is displayed.

**Acceptance Criteria:** Clicking the "Login" button redirects guest users to the login page.

**FR-3.1.0:**

**Description:** The system shall allow guest users to authenticate via token by entering their credentials and clicking the "Login" button.

**Input:** Guest user enters required login information and clicks the "Login" button.

**Output:** User is directed to the home page upon successful authentication.

**Acceptance Criteria:**

* Guest user is successfully authenticated with valid credentials.
* User is directed to the home page after successful login.
* Appropriate error message is displayed for invalid credentials.

**FR-3.1.1:**

**Description:** The system will require non-bot validation after 5 unsuccessful login attempts to prevent automated access.

**Input:** User attempts to log in and fails to provide valid credentials for 5 times.

**Output:** User is prompted to complete a non-bot validation (e.g., CAPTCHA or security question) after five failed login attempts.

**Acceptance Criteria:**

* Non-bot validation is triggered after five consecutive unsuccessful login attempts.
* User must successfully complete the non-bot validation to attempt logging in again.
* The count of unsuccessful attempts resets after a successful login or after the user completes the validation.

**FR-4.0.0:**

**Description:** The system shall display an app bar and a map to registered users.

**Acceptance Criteria:**

* App bar and map are visible to guest users.
* Guest users do not have access to the app bar and map.

**FR-5.0.0:**

**Description:** The system shall log out registered users upon their request.

**Acceptance Criteria:**

* Registered users are logged out when they click the "Logout" button, and the system redirects them to the login page.
* After successful logout, pages require authentication shouldn’t be available.

**EP-2:**

**FR-6.0.0:**

**Description:** The system shall display an app bar at the top of the page with a user icon.

**Acceptance Criteria:** App bar shows user icon.

**FR-7.0.0:**

**Description:** The system shall direct registered users to the user information page upon clicking the user icon on appbar.

**Input:** Registered user clicks the user icon on the app bar.

**Output:** The user's own information page is displayed.

**Acceptance Criteria:** Registered users are directed to the user information page when they click the user icon on the app bar.

**FR-8.0.0:**

**Description:** The system shall allow the system administrator to view a list of all registered users.

**Input:** System administrator accesses the user management section of the system.

**Output:** A list of all registered users, including relevant details (e.g., username, email), is displayed.

**Acceptance Criteria:** The system administrator can access and view a comprehensive list of all registered users in the user management section.

**FR-9.0.0:**

**Description:** The system shall allow the system administrator to update information for any registered user.

**Input:** System administrator selects a user from the user management section and modifies the user’s profile information.

**Output:** The updated user information is saved and displayed for the selected user.

**Acceptance Criteria:** The system administrator can successfully update any registered user’s information, and the system reflects the changes immediately upon saving.

**FR-10.0.0:**

**Description:** The system shall allow the system administrator to delete a registered user.

**Input:** System administrator selects a user from the user management section and confirms the delete action.

**Output:** The selected user is permanently removed from the system, and the user no longer appears in the user list.

**Acceptance Criteria:** The system administrator can successfully delete a registered user, and the user is removed from all relevant records.

**FR-11.0.0:**

**Description:** The system shall display user’s own information to registered user.

**Input:** Registered user clicks the user icon on the app bar.

**Output:** The user's personal information is presented accurately on the information page without password field. There are also buttons for update delete account.

**Acceptance Criteria:** Registered users are directed to the user information page when they click the user icon on the app bar.

**FR-12.0.0:**

**Description:** The system shall allow registered users to update their own user information.

**Output:** The updated user information is saved and displayed.

**Acceptance Criteria:** Registered users can successfully update their profile information, and the system reflects the changes immediately upon saving.

**EP-3:**

**FR-13.0.0:**

**Description:** The system shall allow the system administrator to remove location categories to ensure accuracy and enforce policies.

**Input:** System administrator accesses the location management section and selects a location category to remove.

**Output:** The selected location category is permanently removed from the system.

**Acceptance Criteria:** The system administrator can successfully delete a location category, and it no longer appears in the location categories list.

**FR-14.0.0:**

**Description:** The system shall allow the validating agent to approve location category requests to ensure they are accurate and appropriate.

**Input:** Validating agent reviews a pending location category request and selects the option to approve it.

**Output:** The approved location category is added to the system and made available to users.

**Acceptance Criteria:** The validating agent can successfully approve location category requests, and approved categories are immediately added to the list of available location categories.

**FR-15.0.0:**

**Description:** The system shall allow registered users to request the creation of a new location category for photo uploads.

**Input:** Registered user submits a request to create a new location category during the photo upload process if the desired category does not already exist.

**Output:** The system records the request for a new location category and forwards it for validation by an authorized agent.

**Acceptance Criteria:** Registered users can submit requests for new location categories during photo upload, and the request is successfully forwarded for validation.

**FR-16.0.0:**

**Description:** The system shall allow registered users to display a map with location category markers that can be clicked to view and vote on watermarked photos.

**Input:** Registered user accesses the map and clicks on a location category marker.

**Output:** The system displays watermarked photos associated with the selected location category.

**Acceptance Criteria:** Registered users can view location category markers on the map, click on a marker to see associated watermarked photos, and vote on the displayed photos.

**EP-4:**

**FR-17.0.0:**

**Description:** The admin shall manage all watermarked photos to ensure proper control, monitoring, and enforcement of policies regarding their display and sale.

**Input:** The system administrator accesses the photo management interface.

**Output:** The system displays a list of all watermarked photos, allowing the administrator to review, approve, update, or remove them as necessary.

**Acceptance Criteria:** The system administrator can view and manage all watermarked photos, including taking actions to enforce display and sale policies, ensuring photos are appropriately handled and monitored.

**FR-18.0.0:**

**Description:** The system shall allow a validating agent to approve photos so that they can be included in the auction and made available for stock image companies to bid on.

**Input:** The validating agent accesses the photo approval interface and reviews submitted photos.

**Output:** Approved photos are added to the auction pool, while rejected photos are flagged for further review or removal.

**Acceptance Criteria:** The validating agent can approve or reject photos, with only approved photos being included in the auction, ensuring they meet the necessary criteria for auction eligibility.

**FR-19.0.0:**

**Description:** The system shall allow photographers to take a photo using the system's user interface with GPS enabled to ensure the photo's location cannot be mimicked.

**Input:** The photographer accesses the system's camera interface and enables the GPS function.

**Output:** The system captures the photo and records the GPS coordinates of the location where the photo was taken.

**Acceptance Criteria:** The system ensures that the photo is taken using the system’s interface with the GPS location data embedded, preventing the photo’s location from being spoofed or altered.

**FR-20.0.0:**

**Description:** The system shall allow photographers to upload a photo so that the system can automatically place it in the auction with a watermark.

**Input: T**he photographer selects and uploads a photo through the system’s upload interface.

**Output:** The uploaded photo is automatically watermarked and added to the auction pool for stock image companies to bid on.

**Acceptance Criteria:** The system ensures that every photo uploaded by the photographer is watermarked before being added to the auction, and that the photo is available for bidding once the watermark is applied.

**FR-21.0.0:**

**Description:** The system shall allow photographers to manage their uploaded photos so that they can update, delete, or review them before they are included in the auction.

**Input:** The photographer accesses the photo management interface to view their uploaded photos.

**Output:** The system provides options to update, delete, or review the details of the uploaded photos.

**Acceptance Criteria:** Photographers can successfully view, update, delete, or review their uploaded photos, with any changes reflected before the photos are included in the auction.

**FR-22.0.0:**

**Description:** The system shall allow registered users to display watermarked photos using a location filter so that they can view photos based on specific geographic areas.

**Input:** The registered user selects a location filter within the system's photo display interface.

**Output:** The system displays watermarked photos that match the selected geographic area.

**Acceptance Criteria:** The system successfully filters and displays watermarked photos based on the location criteria selected by the registered user, allowing them to view photos from specific geographic areas.

**FR-23.0.0:**

**Description:** The system shall allow photographers to vote for watermarked photos uploaded by other users so that they can earn money if the photo is selected in the auction.

**Input:** The photographer selects a watermarked photo uploaded by another user and casts a vote.

**Output:** The system records the vote and associates it with the photo for potential inclusion in the auction.

**Acceptance Criteria:** Photographers can successfully vote for other users' photos, and if the photo is selected in the auction, the photographer earns money.

**EP-5:**

**FR-24.0.0:**

**Description:** The system shall automatically start the auction on the 20th of every month and end it on the 1st of the following month.

**Input:** The system detects the current date and time.

**Output:** The auction starts automatically at midnight on the 20th of each month and ends at midnight on the 1st of the following month.

**Acceptance Criteria:** The system successfully initiates the auction on the 20th and closes it on the 1st of each month without manual intervention, ensuring auctions are consistently managed on a monthly cycle.

**FR-25.0.0:**

**Description:** The system shall allow the system administrator to terminate auctions to ensure policy compliance and prevent inappropriate situations.

**Input:** The system administrator accesses the auction management interface and reviews ongoing auctions for policy violations.

**Output:** The auction is terminated if it is found to have inappropriate content or violates policies.

**Acceptance Criteria:** The system administrator can terminate auctions.

**FR-25.1.0:**

**Description:** The system shall refund any paid provision to the user if an auction is terminated by the system administrator.

**Input:** The system administrator terminates an auction due to a policy violation or inappropriate content.

**Output:** The provision paid by the user is automatically refunded to their account.

**Acceptance Criteria:** When an auction is terminated by the system administrator, any provision paid by the stock image company is refunded to their account.

**FR-26.0.0:**

**Description:** The system shall allow stock image companies to place bids on the top 10% voted watermarked photos during an auction, with the provision to be paid upfront.

**Input:** A stock image company selects a top 10% voted watermarked photo and places a bid through the auction interface, paying the required provision upfront.

**Output:** The bid is recorded along with the provision payment, and the company's bid is considered in the auction for the selected photo.

**Acceptance Criteria:** Stock image companies can place bids on watermarked photos that are in the top 10% of voted photos, and the provision is paid upfront before the bid is placed.

**FR-27.0.0:**

**Description:** The system shall grant the original photo to the stock image company that places the highest bid in the auction when the auction ends.

**Input:** The auction ends, and the system evaluates the highest bid placed by the stock image companies.

**Output:** The stock image company with the highest bid is awarded the original photo for purchase.

**Acceptance Criteria:** The system automatically identifies the highest bid in the auction, and the stock image company that placed it is granted the original photo once the auction concludes.

**FR-28.0.0:**

**Description:** The system shall distribute a total 10% commission from the payment of the stock image company to the photographer who successfully votes for a photo in the top 10% of voted photos.

**Input:** A stock image company completes a purchase of a photo that was voted into the top 10%, and a photographer votes for that photo.

**Output:** The photographer who voted for the top 10% photo receives the total 10% commission from the stock image company’s payment.

**Acceptance Criteria:** The system identifies the photographer who voted for the top 10% photo, calculates the 10% commission from the stock image company’s payment, and distributes the full commission to the photographers.

**FR-29.0.0:**

**Description:** The system shall retain a 20% commission from the payment made by the stock image company for each successful photo purchase.

**Input:** A stock image company completes the payment for a purchased photo at the end of the auction.

**Output:** The system automatically deducts and retains a 20% commission from the total payment.

**Acceptance Criteria:** Upon a successful transaction, the system calculates and retains 10% of the total payment from the stock image company, ensuring the commission is deducted and not paid to the photographer.

**FR-30.0.0:**

**Description:** The system shall distribute 70% of the payment from the stock image company to the photographer who took the purchased photo.

**Input:** A stock image company completes the purchase of a specific photo through the auction.

**Output:** The photographer who took the photo receives 70% of the payment amount from the stock image company’s purchase.

**Acceptance Criteria:** Upon a successful purchase, the system automatically transfers 70% of the payment to the photographer who took the photo.

**FR-31.0.0:**

**Description:** The system may retain copies of sold photos and display them with a watermark, including information such as the sale price.

**Input:** A photo is successfully sold through the auction.

**Output:** The sold photo is stored in the system and displayed with a watermark indicating details such as the amount paid by the stock image company.

**Acceptance Criteria:** After a photo is sold, the system retains a watermarked copy with sale information (e.g., sale price), which can be displayed in a gallery or records.

**EP-6:**

**FR-32.0.0:**

**Description:** The system shall allow photographers to assign a monetary value to their photos, if the photos are not in the top 5% of votes.

**Input:** The photographer accesses their photo portfolio and selects a photo to assign a value. They enter the desired monetary value into the provided input field.

**Output:** The system updates the photo’s details to include the assigned value, making it visible to stock image companies.

**Acceptance Criteria:**

* Photographers can assign a monetary value to any photo in their portfolio.
* The assigned value is saved and displayed alongside the photo for stock image companies to view.

**FR-33.0.0:**

**Description:** The system shall allow stock image companies to view photos that are not in the top 5% of votes.

**Input:** Photographer accesses photos that are not in the top 5% of votes.

**Output:** The system displays a list of photos not in the top 5% of votes, along with assigned monetary value.

**Acceptance Criteria:** The system provides an option for stock image companies to filter and view only photos that are not in the top 5% of votes, along with their assigned monetary values.

**FR-33.1.0:**

**Description:** The system shall allow stock image companies to purchase photos that are not in the top 5% of votes.

**Input:** The stock image company selects a photo from the gallery of non-top-5% photos and initiates the purchase process by clicking a button.

**Output:** The system processes the payment, removes the watermark from the purchased photo, and grants the company access to download the high-resolution version.

**Acceptance Criteria:** The system successfully processes the payment and provides the stock image company with access to the purchased photo without the watermark.

**FR-34.0.0:**

**Description:** The system shall display an indicator to show that a photo has already been sold.

**Input:** Registered user views the gallery of photos after purchasing one.

**Output:** The system marks the purchased photo with a "Sold" label or icon in the gallery view.

**Acceptance Criteria:** The system visibly marks photos as "Sold" after purchase, ensuring other users are aware they are no longer available for sale.

**EP-7:**

**FR-35.0.0:**

**Description:** The system will notify stock image companies when an auction is terminated by the system administrator.

**Input:** The system administrator terminates an auction.

**Output:** The system display notification on the system.

**Acceptance Criteria:** Users receive relevant notifications promptly when necessary events occur, ensuring they are informed of important actions or updates.

c. Use Case Diagram

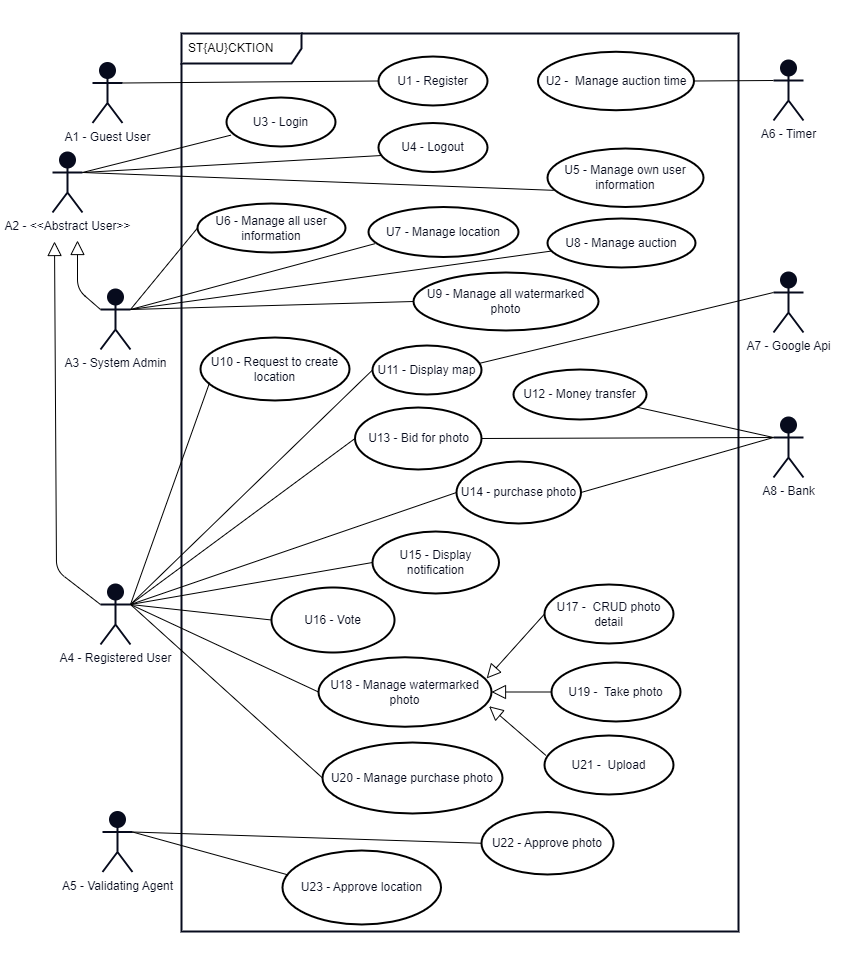


Figure 1: Use Case Diagram

d. Requirement Associations

Table 2: Cross Referencing Functional Requirements to Use Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirment ID** | **User Story ID** | **Use Case ID** | **Initial Plan Functional Requirement ID** |
| FR-1.0.0 | EP1-US1 | U1 | FReq1.1 |
| FR-1.1.0 | EP1-US1 | U1 | FReq1.1 |
| FR-1.2.0 | - | - | - |
| FR-2.0.0 | EP1-US1 | U1 | FReq1.1 |
| FR-2.1.0 | EP1-US1 | U1 | FReq1.1 |
| FR-2.1.1 | EP1-US1 | U1 | FReq1.1 |
| FR-3.0.0 | EP1-US2 | U3 | FReq1.1 |
| FR-3.1.0 | EP1-US2 | U3 | FReq1.1 |
| FR-3.1.1 | EP1-US2 | U3 | FReq1.1 |
| FR-4.0.0 | EP1-US1 | U1 | FReq1.1 |
| FR-5.0.0 | EP1-US3 | U4 | FReq1.1 |
| FR-6.0.0 | EP2-US2 | U5 | FReq1.1 |
| FR-7.0.0 | EP2-US2 | U5 | FReq1.1 |
| FR-8.0.0 | EP2-US1 | U6 | Freq2.1 |
| FR-9.0.0 | EP2-US1 | U6 | Freq2.1 |
| FR-10.0.0 | EP2-US1 | U6 | Freq2.1 |
| FR-11.0.0 | EP2-US2 | U5 | FReq1.1 |
| FR-12.0.0 | EP2-US2 | U5 | FReq1.1 |
| FR-13.0.0 | EP3-US1 | U7 | - |
| FR-14.0.0 | EP3-US2 | U23 | - |
| FR-15.0.0 | EP3-US3 | U10 | - |
| FR-16.0.0 | EP3-US4 | U11 | - |
| FR-17.0.0 | EP4-US1 | U9 | - |
| FR-18.0.0 | EP4-US2 | U22 | FReq3.1 |
| FR-19.0.0 | EP4-US3 | U19 | Freq4.1 |
| FR-20.0.0 | EP4-US4 | U21 | Freq5.1 |
| FR-21.0.0 | EP4-US5 | U17 | Freq6.1 |
| FR-22.0.0 | EP4-US6 | U17 | Freq6.1 |
| FR-23.0.0 | EP4-US7 | U16 | - |
| FR-24.0.0 | EP5-US1 | U2 | - |
| FR-25.0.0 | EP5-US2 | U8 | - |
| FR-25.1.0 | EP5-US2 | U12 | - |
| FR-26.0.0 | EP5-US3 | U13 | - |
| FR-27.0.0 | EP5-US4 | U14 | Freq8.1 |
| FR-28.0.0 | EP5-US5 | U12 | FReq9.1 |
| FR-29.0.0 | EP5-US6 | U12 | FReq9.1 |
| FR-30.0.0 | EP5-US7 | U12 | FReq9.1 |
| FR-31.0.0 | - | U17 | - |
| FR-32.0.0 | EP6-US1 | U20 | - |
| FR-33.0.0 | EP6-US2 | U20 | - |
| FR-34.0.0 | EP6-US1, EP6-US2 | U20 | - |
| FR-35.0.0 | EP7-US1 | U15 | - |

# System Model

The "Take Photo" and "Validate Photo" use cases were selected as they form the core of our project, directly supporting its objective of managing and auctioning high-quality photographs. The "Take Photo" use case is critical for capturing accurate metadata, including GPS coordinates, to prevent location data manipulation or misuse. This ensures the integrity of the system, fostering trust among users and potential buyers. Similarly, the "Validate Photograph" use case guarantees the quality and authenticity of submitted photographs by allowing users to review, approve, or reject entries with clear reasons. This process establishes high standards, making the system more reliable and attractive to photographers and buyers. Together, these use cases are key to maintaining the system’s credibility, improving its efficiency, and ensuring the project’s success.

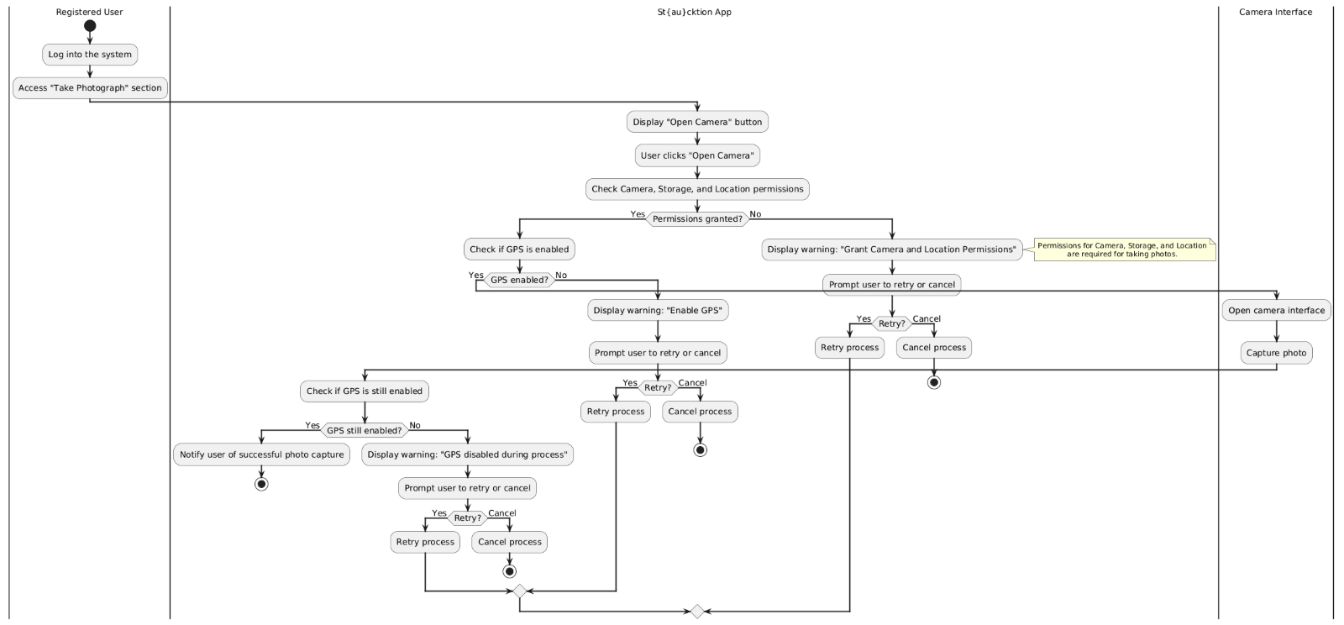


Figure 2: Take Photo Activity Diagram

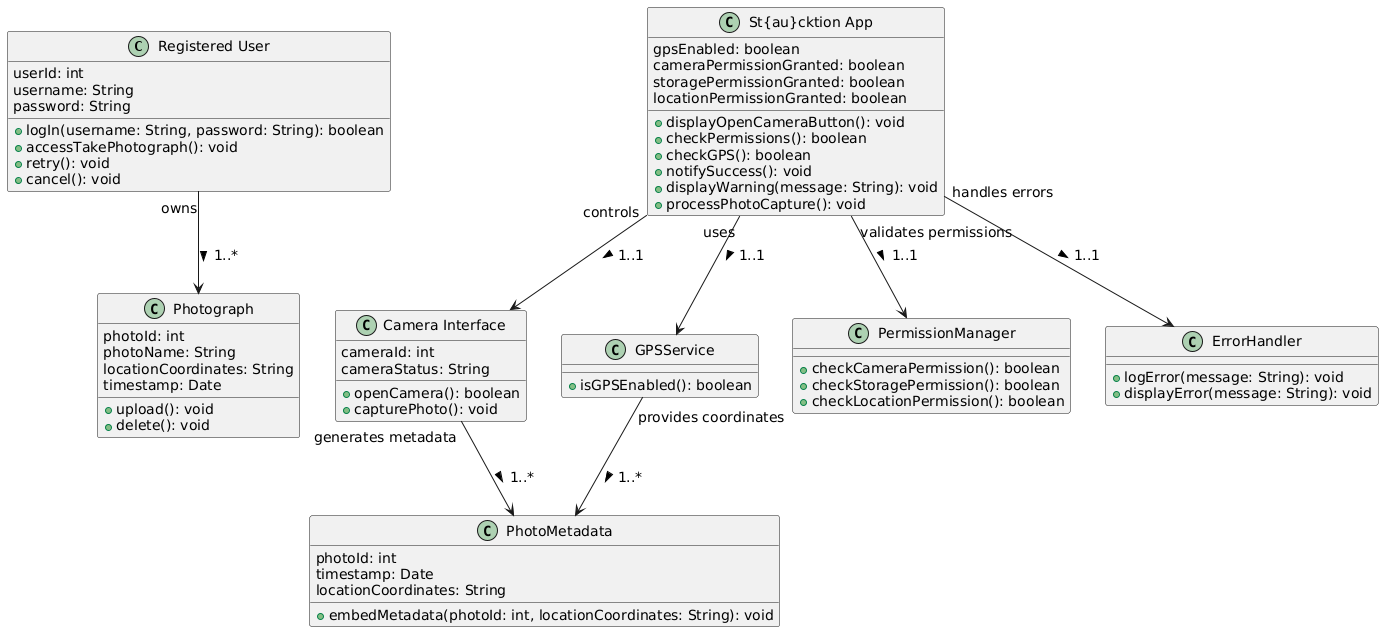


Figure 3: Take Photo Class Diagram

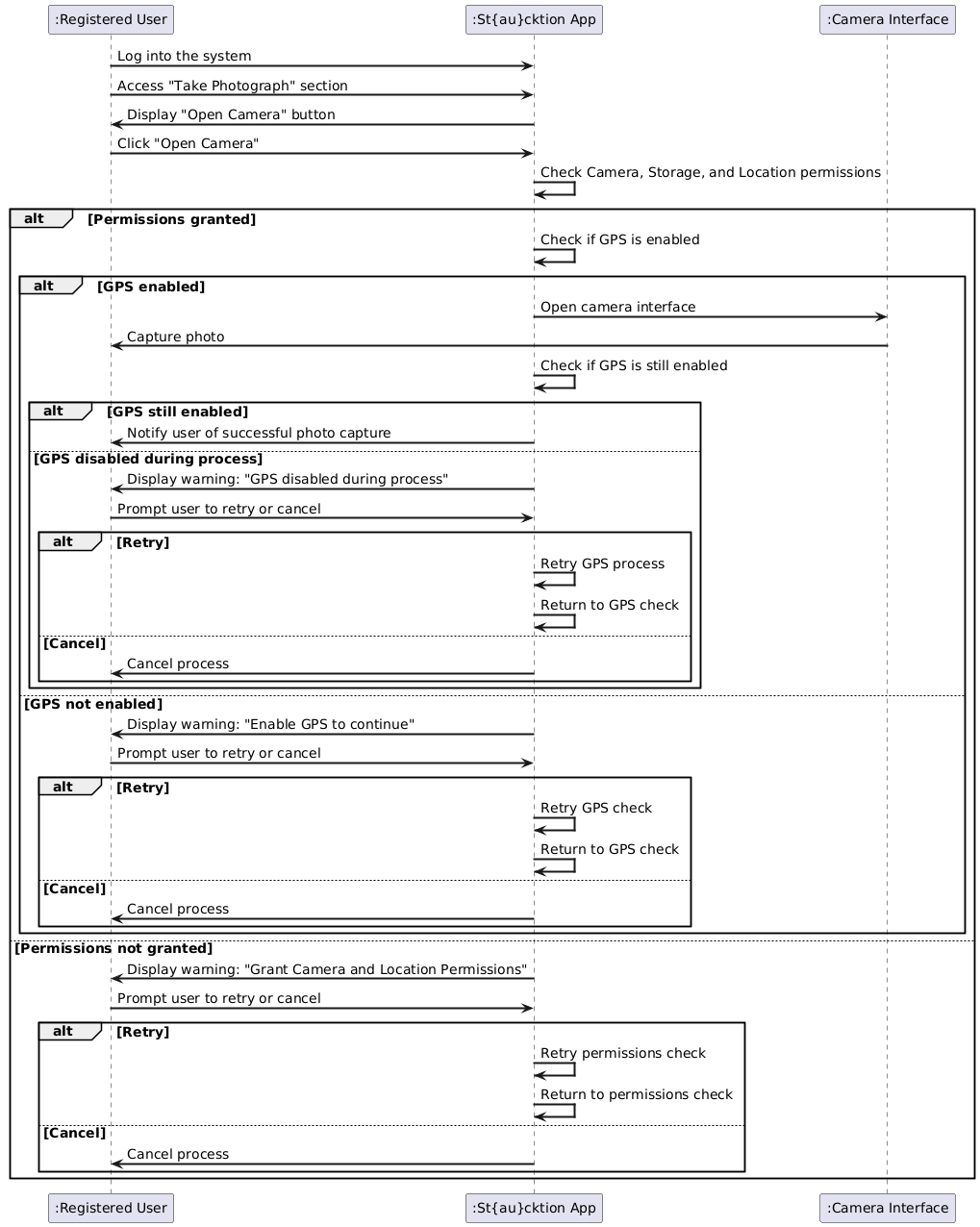


Figure 4: Take Photo Sequence Diagram

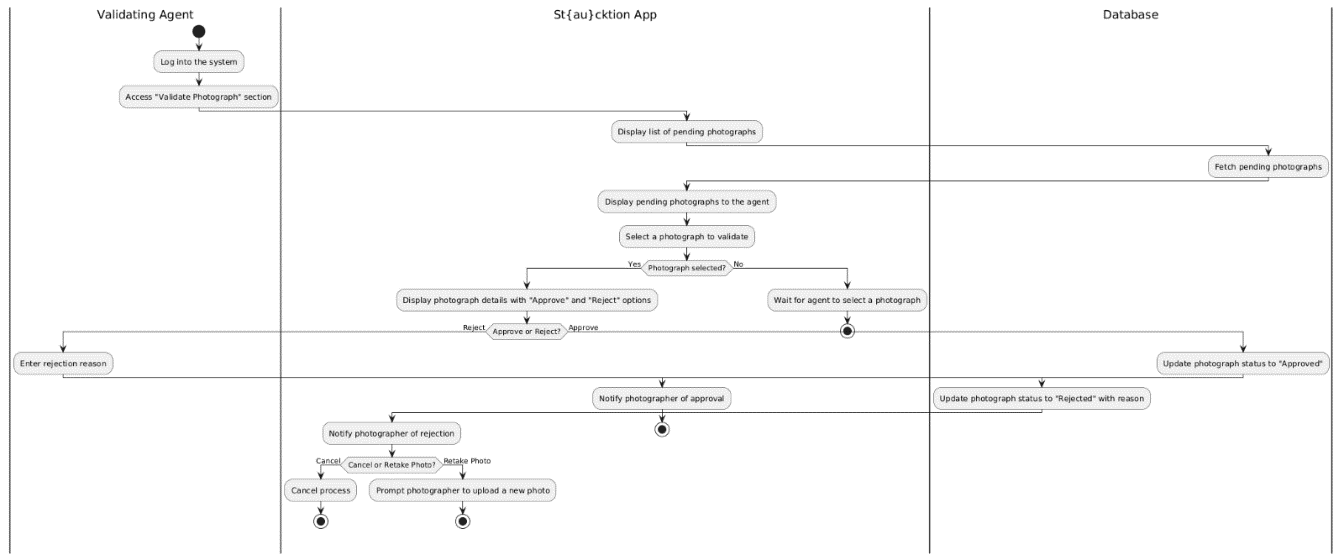


Figure 5: Validate Photo Activity Diagram

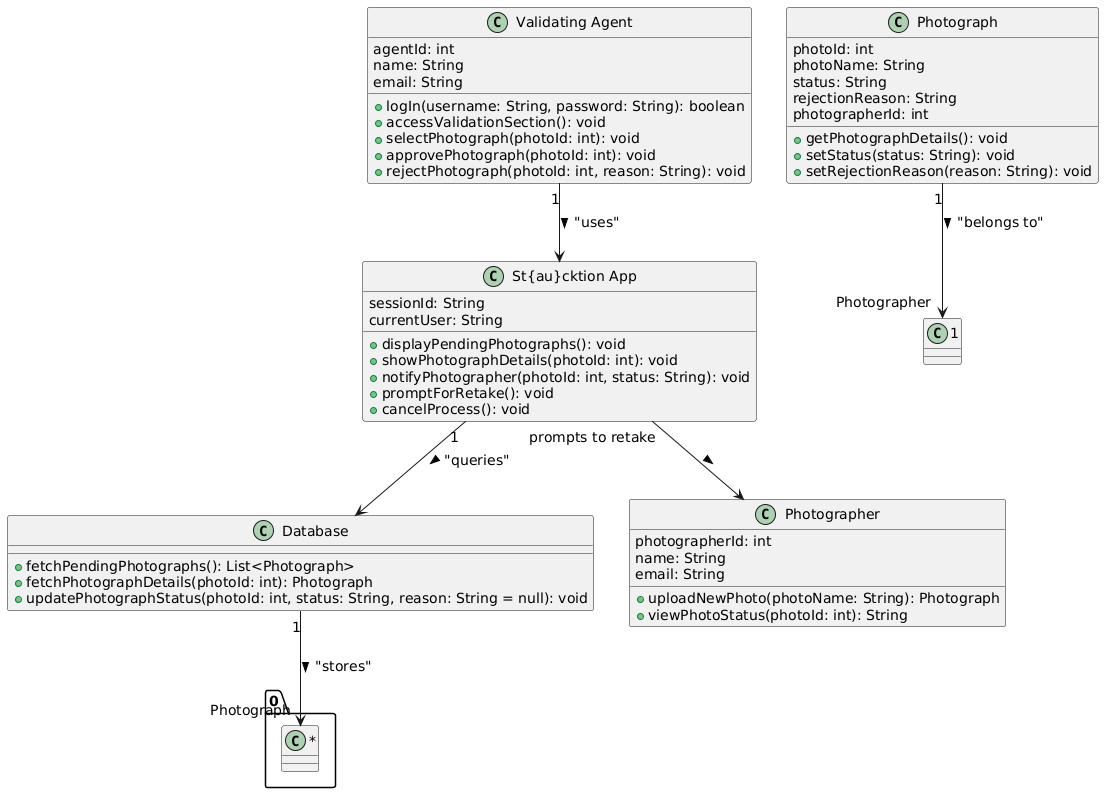


Figure 6: Validate Photo Class Diagram

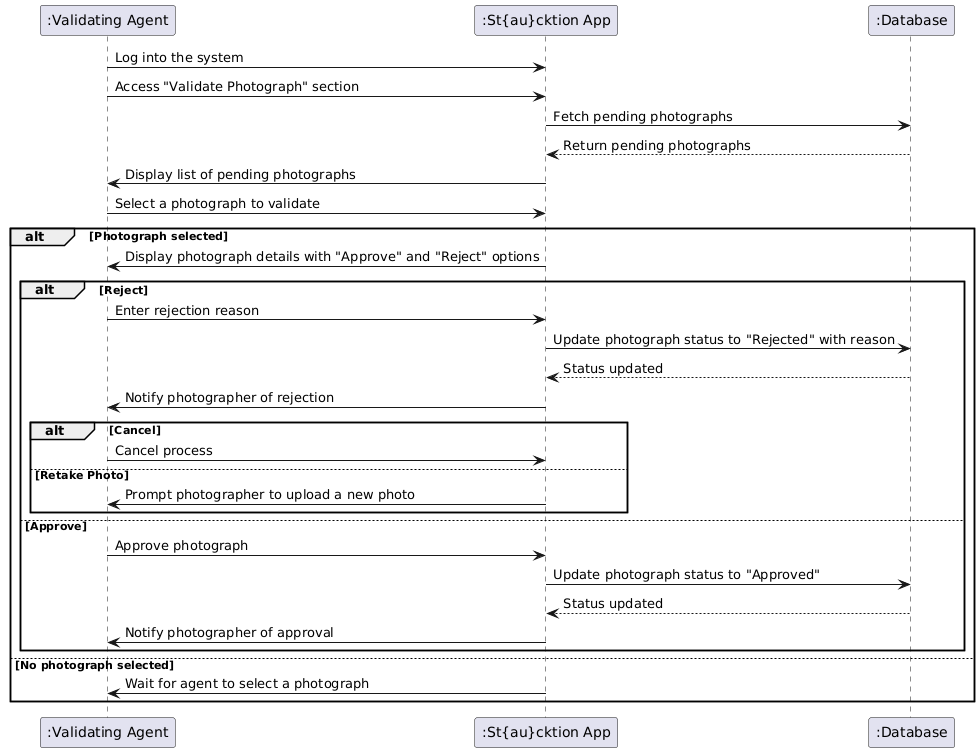


Figure 7: Validate Photo Sequence Diagram

# Requirements Prototypes

For this section, after talking with our Product Owner Cüneyt Sevgi, we decided on prototyping UC\_19 Take Photograph and UC\_22 Validate Photograph use cases. We used Balsamiq Wireframes app to create our prototypes.

1. UC\_19 Take Photograph

First is the Take Photograph prototype. Since we are aiming to implement the functionality of taking photographs on our Android mobile app to get critical data regarding the photograph taken, the prototype resembles the Android user interface. When user goes into the “Take Photograph” section in the app, the user will be shown the screen pictured in Figure 8 below.

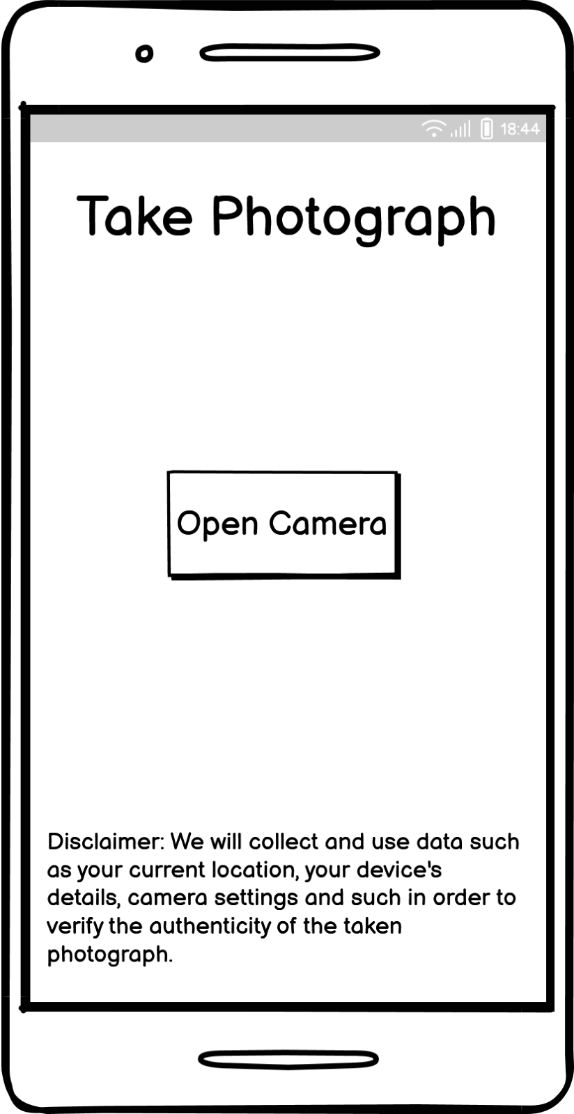


Figure 8: Take Photograph – Initial State

After user clicks the “Open Camera” button, the app will automatically check if the user has given required permissions such as Camera access, Storage system access and Location service access. The app will also check if the Location services are enabled or not on the user’s device. If it is not enabled, or if any of the required permissions are not given, then the app will show the user an error as pictured below in Figure 9 that warns the user about the problem and includes action buttons.

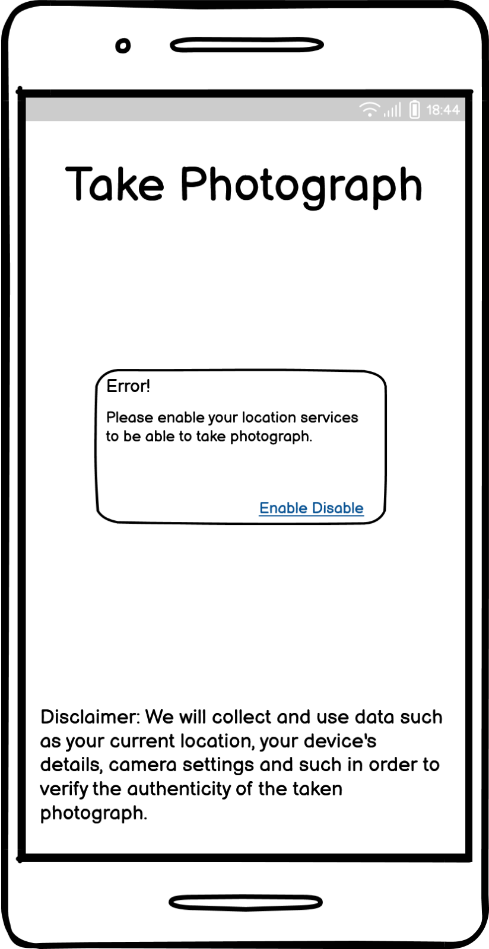


Figure 9: Take Photograph – Error State

1. UC\_22 Validate Photograph

Our next prototype is about the Validate Photograph use case. Even though we are taking photographs on mobile devices, most of the other actions such as bidding, validating photographs and such will mainly be on our web application. Therefore, this prototype resembles a web browser interface. When validating agent goes into the “Validate Photograph” section in the app, the user will be shown the screen pictured in Figure 10:

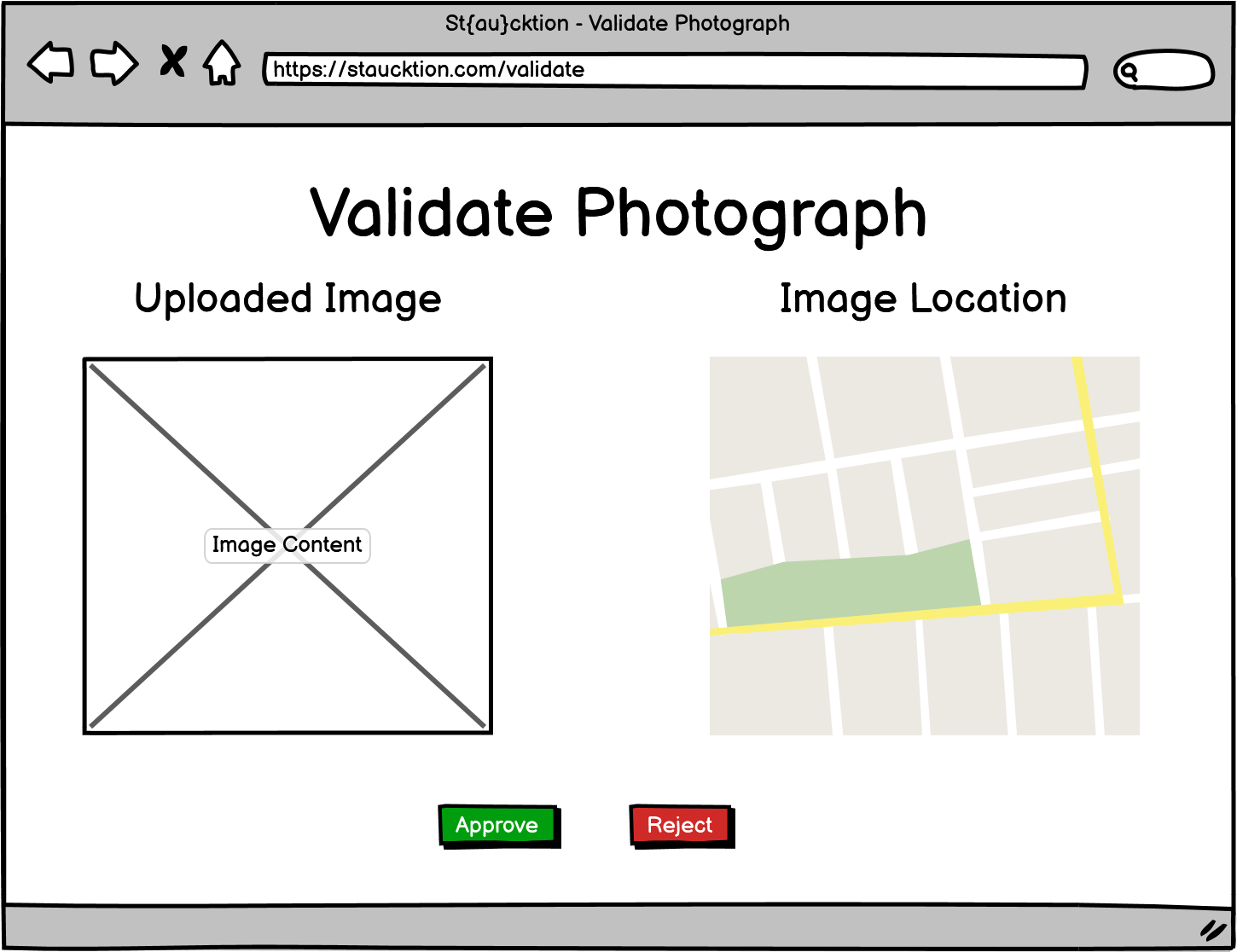


Figure 10: Validate Photograph – Initial State

After validating agent clicks the “Approve” button that is shown at the bottom of Figure 10, the app will automatically update the image’s status as approved and will go back to list of validation pending photographs. However, if the validating agent clicks on the “Reject” button that is next to the “Approve” button, then the app will show user the modal shown in Figure 11 to enter rejection reason, along with options as “Confirm” and “Cancel” to prevent issues that might happen due to misclicks and such. The modal is expected to look like shown below in Figure 11.

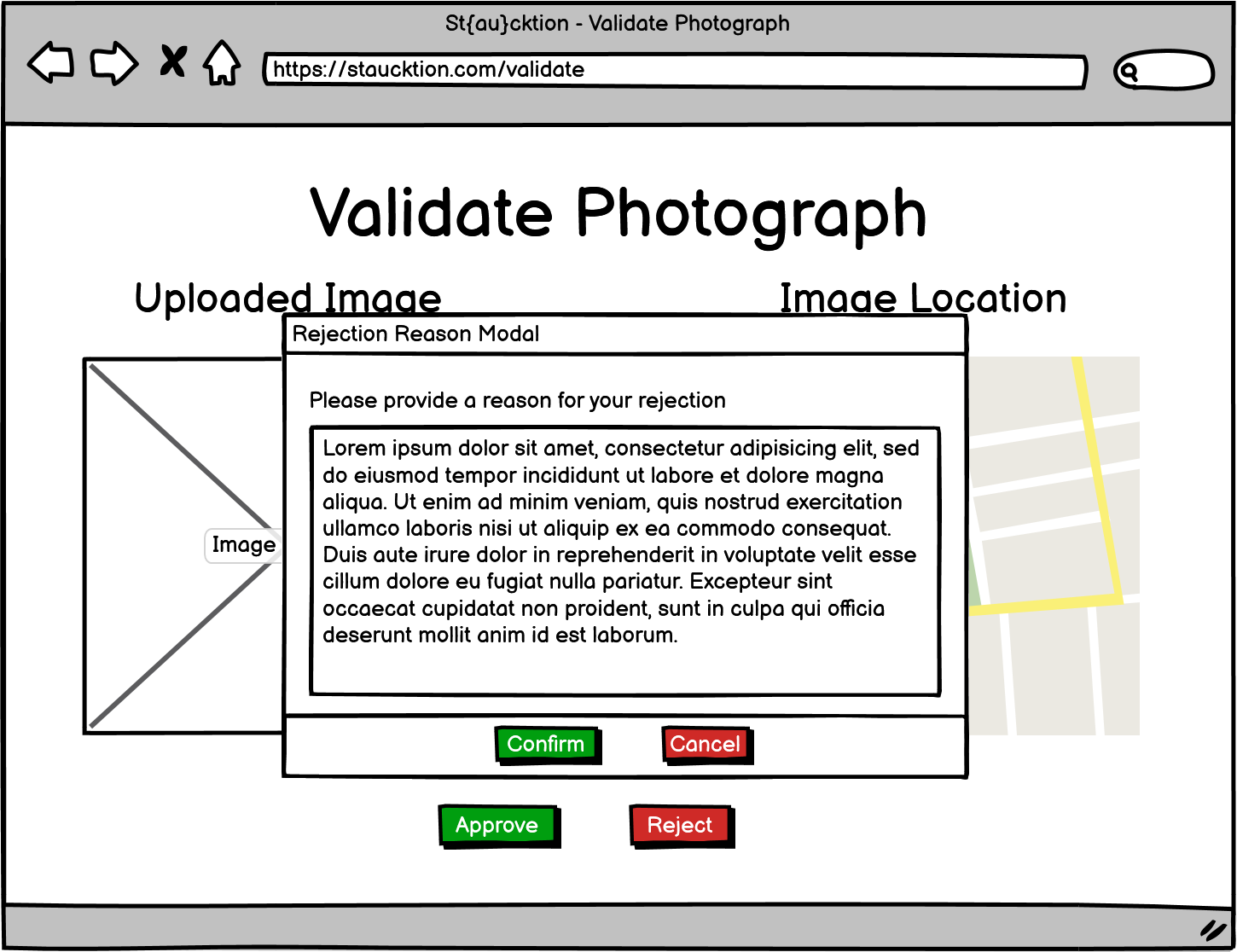


Figure 11: Validate Photograph – Rejection Reason Modal

# Non-functional Requirements

## Usability requirements

**NReq1.1:** 3 out of 5 new users, representing diverse personas, shall complete key user experience tasks such as updating profile information, uploading a photo and reviewing a photo within 5 minutes, using no more than 10 GUI interactions (e.g., pressing buttons or selecting values from dropdown menus).

**NReq1.2:** 3 out of 5 experienced users who have experience with system at least 8 hours, representing diverse persona, shall complete key user experience tasks such as updating profile information, uploading a photo and reviewing a photo within 2 minutes, using no more than 10 GUI interactions (e.g., pressing buttons or selecting values from dropdown menus).

**NReq2.1:** 3 out of 5 new users, should report a satisfaction level of at least 3 out of 5 after completing key user tasks, such as updating profile information, uploading a photo and reviewing a photo.

**NReq2.2:** 3 out of 5 experienced users who have experience with system at least 8 hours, should report a satisfaction level of at least 4 out of 5 after completing key user tasks, such as updating profile information, uploading a photo and reviewing a photo.

## Performance requirements

Capacity Requirements

**NReq3.1:** 200 simultaneous users shall reach all services.

Dynamic Requirements

**NReq4.1:** Web user interface documents should be rendered within 2 seconds under normal workload conditions 95% of the time.

**NReq5.1:** The Web API shall respond to text-based requests within 700ms under normal workload conditions, 95% of the time.

## Software system attributes

**a) Reliability**

**NReq6.1:** System shall experience no more than 2 level-one faults per month.

**NReq6.2:** System shall experience no more than 5 level two faults per month.

**b) Availability**

**NReq7.1:** The system architecture shall replace failed system modules with healthy ones automatically.

**c) Security**

**NReq8.1:** Data transactions using tokens should be secured with the HS256 algorithm for integrity.

**NReq9.1:** Server service ports shall not be exposed to the public.

**NReq10.1:** Web services should be secured with TLS and RSA encryption, utilizing SHA-256 to protect against attacks such as man-in-the-middle.

**NReq11.1:** System shall use a salting technique as SHA-256 to keep user passwords secure.

**d) Maintainability**

**NReq12.1:** The average time for a maintenance engineer to repair a severity-2 defect shall be no greater than 8 person-hours.

**e) Portability**

**NReq13.1:** Web software modules shall be designed to be run on Windows, MacOS, Linux operating systems, ensuring a high degree of flexibility when porting to alternative environments.

**NReq14.1:** The application shall be developed in a proven portable programming language to enhance compatibility across various operating systems.

**NReq15.1:** Web applications shall be deployed using containerization so that it achieves consistent performance across various environments, enhances scalability by allowing for rapid deployment and scaling of application instances, isolates components to prevent failures from affecting the entire system, and simplifies maintenance and management through integration with container orchestration tools.

## Constraints

**NReq16.1:** Web applications shall be implemented using TypeScript.

**NReq17.1:** The map shall be displayed using the Google Maps API.

## Error Handling Requirements.

User Errors

**NReq18.1:** System shall warn user to non-critical user errors. Such as input text value to numeric input field.

External System Errors

**NReq19.1:** System shall log noncritical external system errors. Such as cannot display map.

**NReq20.1:** System shall log critical external system errors. Such as monetary transaction fail.

Developer Errors

**NReq21.1:** System shall log errors and aim to avoid crashing the entire system.

## Other Non-Functional Requirements

**External Interfaces**

**NReq22.1:** System shall communicate with Bank API to be able to perform monetary transactions.

# Logical Database Requirements

**a) Types of information used by various functions**

1. User Data: Stores data for each user. Such data would include things like the user’s role (regular user, stock firm, validator and validator), their basic profile information like username, real name, email, image and such.
2. Image Data: Stores data of the uploaded images. Such data would include things like name of the image, directory where that image is stored in (both watermarked and non-watermarked versions), which photographer uploaded it, metadata of the image including its location and with which device it was taken on and such.
3. Category Data: Stores the category (landmark) options that will be used for categorizing images for their relevant auctions.
4. Auction Data: Stores all the history of both ongoing and finished auctions with references to which image the auction is linked to, and once finalized which stock photo company bought it, and the closing bid.
5. Vote Data: Keeps track of which user liked which image. This is a separate table to make it easier to keep track of data and possibly make it easier to generate analytical data as well.
6. Bidding Data: Keeps tracks of bids placed on ongoing auctions. This way we can keep track of the bid history on auctions.
7. Internal Financial Data: Keeps track of **finalized** internal financial actions like bid data, commission paid, how much money photographer made from the bid and such.
8. External Financial Data: Keeps track of all external financial transactions, including failed transfers. Such data would include things like the status of the transaction, where the money is requested to be withdrawn or where it is deposited from, balance change, fees paid to the external banks and such.

**b) Frequency of use**

The following are the expected frequencies of usage of the data listed above. These are just the “general” expected frequencies and do not include external things like “system admin accessing all data from database” or “data analysts accessing data through data warehouse” and such.

* User Data – Very Frequently: Accessed very frequently in order to validate and authenticate users and their access levels, mainly during register and login actions.
* Image Data - Frequently: Accessed whenever an image is uploaded and viewed.
* Category Data – Somewhat Frequently: Accessed whenever an admin or photographer wishes to see why an image was rejected.
* Auction Data – Frequently: Accessed whenever an image with an ongoing or finished auction is shown.
* Vote Data – Frequently: Accessed whenever an image is viewed, additionally when user performs a vote action, wants to view on which images they voted on, and when sorting all auctions and images by their votes.
* Bidding Data – Somewhat Frequently: Accessed whenever a stock photo company bids in an auction, when auction details are shown and when bid history is shown.
* Internal Financial Data - Rarely: Accessed whenever the internal transactions of the user are shown.
* External Financial Data – Rarely: Accessed whenever the external transactions of the user are shown.

**c) Accessing capabilities**

There will be a monolithic database for the beginning. For this database, the system will use only a singular account with full read/write access to the **existing** tables, which means that the system user will not be able to create/drop any tables or modify any of the schematics and only will be able to perform CRUD operations on the database. This is to reduce the possible impact of an attack that might occur if something goes wrong within the system.

There will also be another user with “superuser” permissions which will be used solely by the system administrator when modifying the schematics or performing general maintenance on the system.

**d) Data entities and their relationships**

The following are written based on the tables written in the a. section of this part. Therefore, “User” does not mean a regular user but rather the “users” table and so on.

Table 3: Relationships Between Entities

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| To   From | User | Image | Category | Auction | Vote | Bid | Internal Finance | External Finance |
| User | X | 1-M | X | M-N | 1-M | 1-M | 1-M | 1-M |
| Image | M-1 | X | M-1 | 1-1 | 1-M | X | X | X |
| Category | X | 1-M | 1-1 | X | X | X | X | X |
| Auction | M-N | 1-1 | X | X | X | 1-M | 1-M | 1-M |
| Vote | M-1 | M-1 | X | X | X | X | X | X |
| Bid | M-1 | X | X | M-1 | X | X | X | X |
| Internal Finance | M-1 | X | X | M-1 | X | X | X | X |
| External Finance | M-1 | X | X | M-1 | X | X | X | X |

Legend:

X: No relation

1-M: One-to-Many

M-1: Many-to-One

M-N: Many-to-Many

**e) Integrity constraints**

* Users: Each User may have relations with Images, Votes, Bids, Internal and External transactions. Since we will not actually delete any users from the system (for bookkeeping reasons and to keep historical data consistent, explained more in next “Data retention requirements” section), the only deletion we would do in this table would be anonymizing the user’s personal details, which we will achieve by setting the values of columns that identify that person to `NULL`.
* Images: Each Image has a relation with a User and a Category; and may have relations with Auctions and Votes. To keep historical data consistent and User actions still traceable in case of need to do so in the future, for reasons such as a federal investigation, the image records will not be deleted. However, on such cases for Image deletion as explained in more detail in the next “Data retention requirements” section, image itself will be deleted from the storage itself, and the path pointers that point to Image’s stored location will be set to `NULL`.
* Categories: Each Category may have relations with Images, and Categories due to parent-child category relationship. If a Category is deleted, the related Image entries’ category reference will be set to `NULL` to make it possible to re-categorize them again. Also, any child category that points to the deleted category will also no longer point to the deleted Category by setting their related column value to `NULL` as well, and they will become a parent Category themselves.
* Auctions: Each Category has a relation with an Image, and may have relations with Bids, Internal and External transactions. Similarly, as in Users case, to keep proper bookkeeping, auction data will never be deleted, and rather their status would be updated accordingly instead.
* Votes: Each Vote has a relation with both a User and an Image. In case of a deletion of a Vote, since no entity references to the Votes, but rather Votes reference to them, there will not be any issues upon deletion of Votes.
* Bids: Similarly, as in Votes, each Bid has a relation with both a User and an Image, and once again, since no entity references to the Bids, but rather Bids reference to them, there will not be any issues upon deletion of Bids.
* Internal and External Financial Data: Bookkeeping is one of the most important jobs in a business. It is not important only for the company itself to see how they are doing financially, but also due to tax reasons and settling balances and keeping records of them to make sure they support each other. Because of this, none of the entries in any financial data tables will be deleted.

**f) Data retention requirements**

All the data that goes into the server are planned to stay there as long as possible. Only exceptions for this are as the following:

* All the rejected images by validators to keep the server clean of things like personal data and criminally offensive images.
* Images of the canceled / unbought auctions will be deleted after 2 years to optimize storage.
* User data in case a user uses “Forget Me” setting under “Right to be Forgotten” to comply with GDPR / CCPA / TDPA regulations based on the user’s country. In such case, user’s personal data will be replaced with anonymous data to keep historical data intact and correct for financial reports.

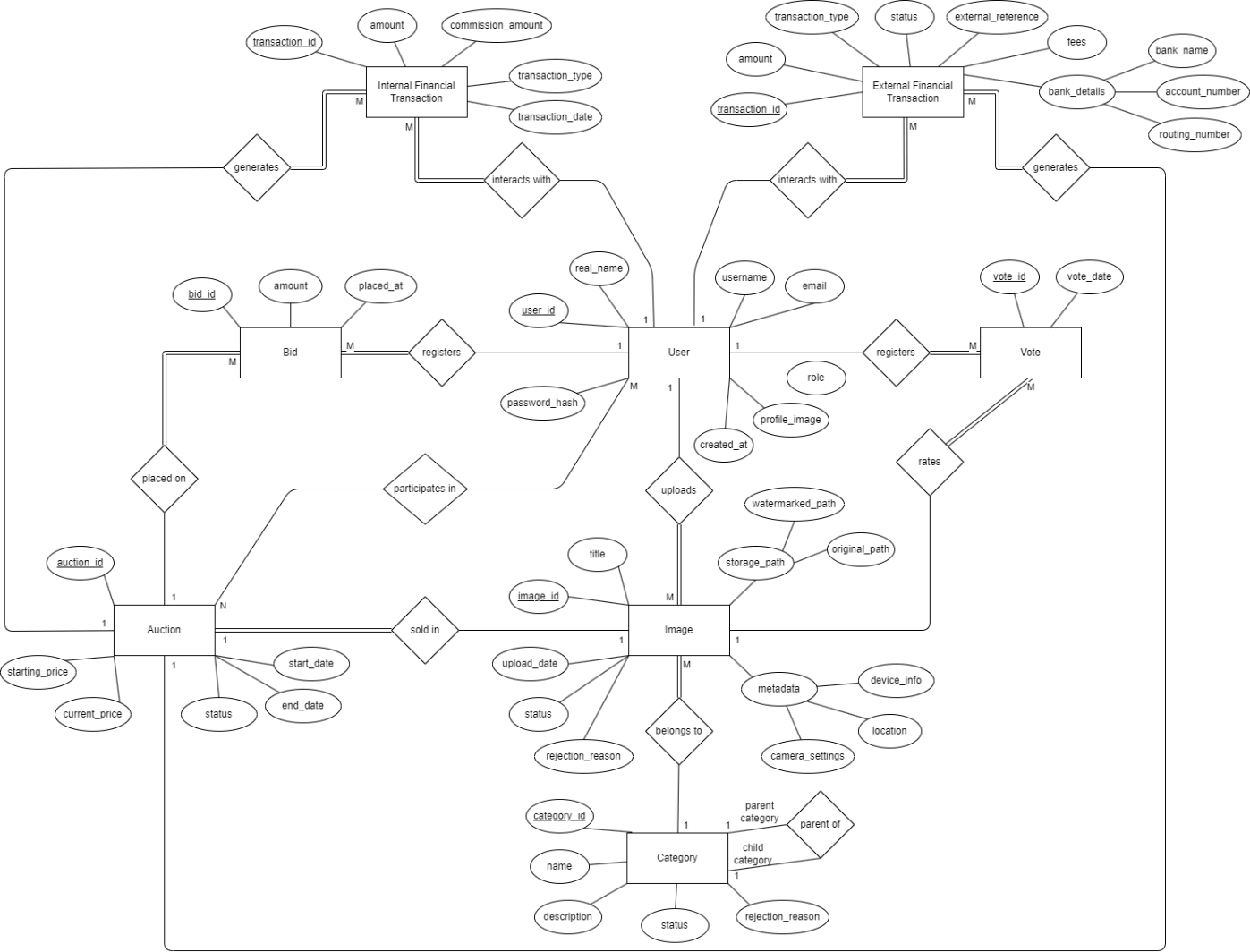
****

Figure 12: EER Diagram of St{au}cktion’s Database Using Chen’s Notation

# Verification

-Unit Testing

This is a type of testing done to test individual units or components of a software. Unit testing will be utilized on the code side to test the functionality of different functions, methods, objects, etc. during development and make sure they operate the way they are supposed to, and to check that they fulfill the functional and non-functional requirements of the project. Examples of how we can use this can be checking the functions used to fulfill the functional requirement 2.1.0, which describes that the system shall allow guest users to register by entering their information and clicking the register button. We can test this by requirement by separating it into mini requirements, like the validation of user input, the saving of the user data to the database, and the verification of CRUD operations. Then write test codes that check if these individual “mini requirements” work as they should.  As for how the testing will be done. Depending on the nature of the function, and the requirement, manual testing can be useful but considering the scope of the project using automated testing tools and manually testing key requirements/functions would be for the benefit of the team, time management, and the project.

-Integration Testing:

This is a type of testing where modules, or components of the software are tested together as a unified system. This form of testing will be conducted to test components like the login and registration systems to see if the integration of frontend, backend, and database elements work as intended. Additionally, the components of our project such as taking a photo and the system watermarking it will be tested using this testing format. As for the automation level of the tests, doing manual testing for the integration would be more beneficial so the tests will be done manually.

-System Testing:

This is a type of testing that focuses on verifying the entire system as a whole. One that ensures the complete application functions as intended. Meaning that this testing will be done near the end of development once the whole system is integrated to make sure it functions as intended, and that it implements all functional and non-functional requirements we have defined in project documentation. Because of this conducting these tests manually will be for the best. One approach being considered is getting family members, friends, and members of other senior project groups we are familiar with to use our system and get manual testing results that way.

-Acceptance Testing:

This is a form of testing that ensures the software meets the business and end users’ requirements. Because of this, similarly to system testing, we will be getting the help of outside users to manually test the product. Specifically, to see if it meets end users’ requirements. To test if it meets the business requirements, we will have our project advisor, and product owner, CTIS department senior lecturer Cüneyt Sevgi manually test the system.

# Discussions

The preparation of the SRS faced several challenges, including time and resource limitations. With effective team working, SRS is prepared, even if the time limitation was so stressful on the team.

## Limitations and Constraints

Because of early due date and midterm period we suffer to meet deadline.

## Health and Safety Issues

In the mental health extent, group members experienced huge stress for preparing the SRS before the deadline. Because of cold allergies some team members suffer.

## Legal Issues

We did not experience any legal issues while preparing the SRS.

## Economic Issues and Constraints

Team members need to use paid software such as Microsoft Word.

## Sustainability

We held our meetings on Zoom to avoid in-person gatherings, which would require everyone to use transportation that contributes excess carbon emissions to the environment. We also used digital tools like Notion and GitHub to minimize paper usage by keeping all documents and collaboration online.

## Ethical Issues

We did not experience any ethical issues while preparing the SRS.

## Multidisciplinary Collaboration

We did not collaborate with anyone from other departments, so this section is not applicable for us.

1. GenAI tool: ChatGPT 4

   Prompt: Please make this text sound more formal and concise: [text here]

   Rationale: To improve the language used and make it more formal while keeping it concise.   [↑](#footnote-ref-1)
2. GenAI tool: ChatGPT 3.5

   Prompt: Make a table about these users

   Rationale: To make this table better. [↑](#footnote-ref-2)