COLLECTIFY

Group - 1


# Software Engineering

## BLG 411E

**Members**

Tanay Bensu Yurttürk - 150220766

İrem Çağın Yurttürk - 150220765

Bilgehan Altıntaş - 150180019

Seyfullah Sait Şahin - 150180097

Mustafa Deveci - 150190109

**Course given by**

Ahmet Cüneyd Tantuğ

December 29, 2023

# CONTENTS

# 1. INTRODUCTION

The primary objective of this document is to provide a detailed exploration and analysis of the project at hand. By doing so, it aims to offer a comprehensive understanding of the application from both a developer and customer perspective. The overarching goal is to guide the development process effectively, ensuring that the end product aligns with the expectations and requirements of all stakeholders involved. Through the examination of various use case scenarios, functionalities, and system models, this document aims to set a solid foundation for the project, enabling a clear road map for its successful implementation.

The document is structured to cover critical aspects of the project, beginning with an outlining the essential system requirements, providing a foundational understanding of the technical specifications necessary for the successful development of the project.The document then engages in an exploration of use case scenarios. This includes an examination of different user types, their accessibility's, and the functionalities they can perform within the application. Following this, each use case diagram discussed in the previous section is dissected to provide a detailed view of the system's functionality.

In summary, document provides a thorough exploration of the project, guiding both developers and stakeholders.

**Table 1.1:** Changelog

| | |
|---|---|
| **Use Case Diagrams** | Specific cases' diagrams added |
| **User Interface Model** | App's important screens added with their use cases |
| **User Interface Model** | Admin panel screens added |
| **General Data Model** | Entity Relation Diagram added |
| **Flow Diagrams** | Data Flow Diagrams added with their explanations |

# 2. SYSTEM REQUIREMENTS

List of functional and non-functional requirements in natural language form.

## 2.1. FUNCTIONAL REQUIREMENTS

### 2.1.1. User Profile Management (Deliverable: 3.1)

- Password can be changed by user.
- Email cannot be changed by user.
- Users can customize their profile by adding a profile picture.
- A bio or description field allows users to share information about themselves or their collecting interests.

### 2.1.2. Home Page (Deliverable: 4)

- Users can explore card collections.
- Users can display of card details.

### 2.1.3. Auction System (Deliverable: 5.2)

- The amount bid during the auction must be deducted from the bidder’s balance and cannot be used until the auction ends.
- Once a bid is placed it cannot be withdrawn.
- If a bid has been made, the card placed in the auction cannot be withdrawn.
- Users can not place bids that are lower than the current highest bid.

### 2.1.4. Card Collection System (Deliverable: 5.3)

- The card placed on the sales list can be withdrawn.
- If a card from a collection that is not owned is collected, that collection must be added to user’s collection list.
- If the last card of the collection is sold, that collection must be removed from user’s collection list.
- A card can be collected multiple times.
- Users can create virtual showcases for their favorite items.

### 2.1.5. Wallet Integration (Deliverable: 5.4)

- Once the card is sold, the fee must be sent to the former owner’s balance.
- The fee for the card must be deducted from the purchaser’s balance.

### 2.1.6. In-Game Coin Rewards System (Deliverable: 5.5)

- If the user earns money from the daily spin, it is added to the user’s balance.
- The daily wheel should be turned only once a day.
- Users can earn in-game coins through completing the collections.

### 2.1.7. Search Functionality (Deliverable: 6)

- User can view the related cards when searched.

### 2.1.8. Filtering Functionality (Deliverable: 6.1)

- User can categorize items based on type, rarity, or other custom attributes.

## 2.2. NON-FUNCTIONAL REQUIREMENTS

1. User’s password should be encrypted when it’s stored .
2. Application presents a user-friendly interface for users.
3. Ensuring consistency across different devices (IOS, Android).
4. The application must be responsive and users must be able to complete their transactions without waiting.
5. Server capacity must be increased when necessary.
6. Users will be able to access their accounts only when when they enter the correct username and password.

# 3. USE CASES

## 3.1. USER TYPES

List of user profiles of the software.

1. Visitor: Visitors represent individuals who can explore the application without the need for registration. They have the privilege to view collections, cards, and events available within the app. Additionally, visitors have the option to sign up or sign in, transforming into registered users to unlock further features and personalized experiences. This accessibility encourages potential users to explore the platform before committing to a registered account.

2. Card Collector (User): Users can navigate through all pages, providing them with a comprehensive experience unlike visitors. They gain access to a dynamic leaderboard, where they can track their progress and compete with fellow users. The power to buy and sell cards adds a layer of excitement and strategy to their journey, allowing them to build a unique collection. Engaging in auctions becomes a thrilling opportunity for users to collect valuable cards and enhance their portfolios. Achievements rewards users for their accomplishments within the app. Additionally, the daily spin feature adds an element of surprise and rewards, keeping users eagerly returning each day. The ability to review collected cards enhances the sense of accomplishment, providing a personalized showcase of their in-app journey.

3. App Administrators: App Administrators have comprehensive control over Collectify, ensuring the smooth operation and growth of the platform. Their authority extends across various crucial functions, empowering them to add new collections and cards, edit existing card details, and remove cards when necessary. Administrators hold the responsibility of overseeing user management, enabling them to view user profiles and take appropriate actions to maintain community standards. Moreover, they have the privilege to grant administrator roles to users, fostering a collaborative approach to system management.

## 3.2. USE CASE DIAGRAM


**Figure 3.1:** Use Case Diagram for whole system

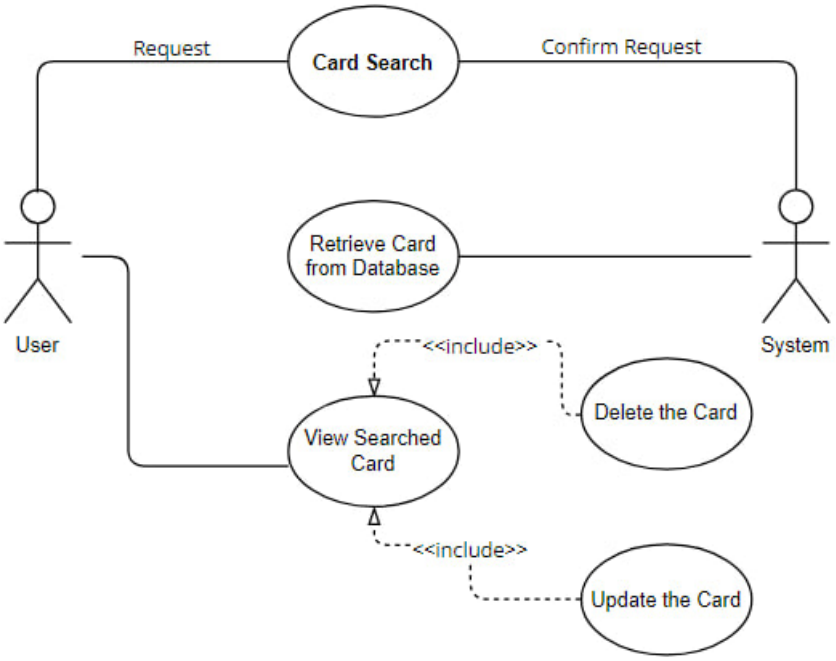


**Figure 3.2:** Use Case Diagram for Card Search

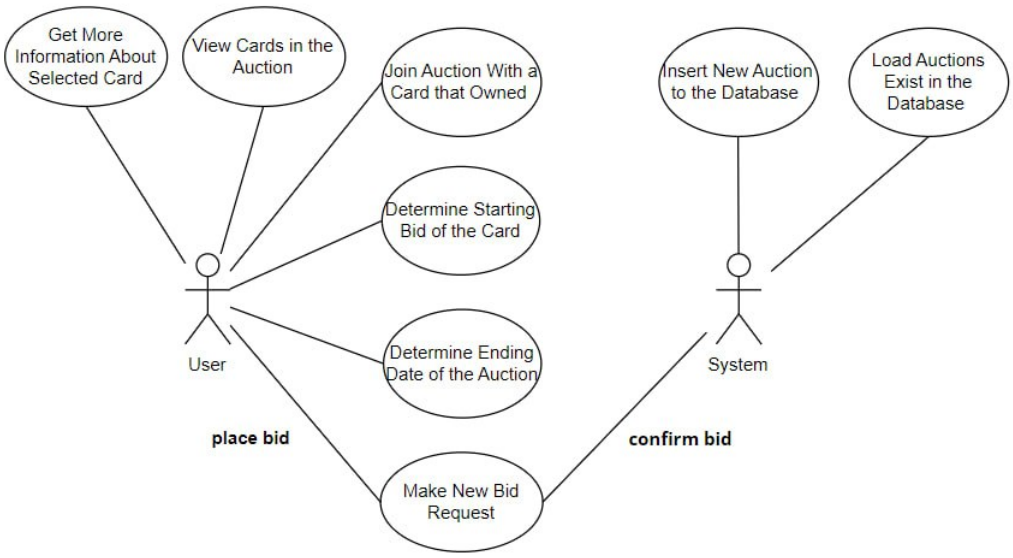


**Figure 3.3:** Use Case Diagram for Auction

## 3.3. USE CASES

A use case for each feature in project.

**Table 3.1:** Use-Case Descriptions

| | |
|---|---|
| **Use Case:** | Add New Collections |
| **Primary actor:** | Administrator |
| **Purpose:** | To add new collections to the Collectify application |
| **Overview:** | The use case begins when an admin, with the necessary permissions, intends to add a new collection to the Collectify application. The system provides an interface where the admin can enter details for the new collection, such as a name. Upon submitting the form, the system validates the input, creates the new collection, and updates the collection list. |
| **Pre Condition:** | 1. The administrator is logged into the system.<br>2. The system is running. |
| **Post Condition:** | A new collection is added to the system, and users can view and interact with it. |
| **Alternative Flow:** | 1. Admin cancels the "Add Collection" action. The process ends.<br>2. System cannot validate the input if there is a collection with the same name. Prompt the admin to correct the information. |

**Table 3.2:** Use-Case Descriptions

| | |
|---|---|
| **Use Case:** | View Users |
| **Primary actor:** | Administrator |
| **Purpose:** | To allow an admin to view users and details |
| **Overview:** | The use case begins when an admin, with the necessary permissions, wants to view details about a specific user. The system provides an interface where the admin can search for and select a user. Once a user is selected, the system displays relevant details such as username, email, and other pertinent information. |
| **Pre Condition:** | 1. The administrator is logged into the system.<br>2. The system is running. |
| **Post Condition:** | The admin can see the users or view detailed information about the selected user. |
| **Alternative Flow:** | Admin cancels the "View User" action. The process ends.<br>2. System cannot find the specified user. Prompt the admin to check the entered information. |

**Table 3.3:** Use-Case Descriptions

| | |
|---|---|
| **Use Case:** | Add New Cards |
| **Primary actor:** | Administrator |
| **Purpose:** | To add new cards to the app |
| **Overview:** | The use case begins when the administrator decides to enrich the application's content by adding new cards. The administrator provides details for the new cards, such as card name, description, and image. Card name should not exist in the database, if it exist alternative flow is executed. If not exists, the system crates specific image URL from provided details and saves this URL to the database. |
| **Pre Condition:** | 1. The administrator is logged into the system.<br>2. The system is running. 3. Cards information must be provided. |
| **Post Condition:** | 1. New cards are successfully added to the market.<br>2. New cards are presented in the main page. |
| **Alternative Flow:** | 1. Admin cancels the "Add Card" action. The process ends.<br>2. System cannot validate the input if there is a card with the same name. Prompt the admin to correct the information. |

**Table 3.4:** Use-Case Descriptions

| | |
|---|---|
| **Use Case:** | Give Administrator to User |
| **Primary actor:** | Administrator |
| **Purpose:** | To grant administrative privileges to a user |
| **Overview:** | The use case begins when an admin, with the necessary permissions, decides to promote a user to an administrator role. The system provides an interface where the admin can select a user and grant them administrator privileges. |
| **Pre Condition:** | 1. The administrator is logged into the system.<br>2. The system is running.<br>3. Admin listed the users and searched for specific account. |
| **Post Condition:** | The selected user is now granted administrator status. |
| **Alternative Flow:** | Admin cancels the "Give Administrator" action and the process ends.<br>2. System cannot find the specified user and prompt the admin to check the entered information. |

**Table 3.5:** Use-Case Descriptions

| | |
|---|---|
| **Use Case:** | Sign Up |
| **Primary actor:** | Visitor |
| **Purpose:** | To access the all functionalities |
| **Overview:** | The use case begins when the user entered the app as a visitor. If the user wants to access the all functionalities he/she can sign up to the app. The user enters email and password. The system checks whether email format is valid and password is longer than 6 chars. If there is no problem user is navigated to the page to choose username. User must choose username that does not exist in the database and if wants can choose image to display in profile. If chosen username is valid, sign up is completed successfully. But if there is a problem in any step, alternative flow is executed. |
| **Pre Condition:** | 1. The user must download the app.<br>2. The system must be running.<br>3.The user must enter valid email that does not exist in database.<br>4.The user must enter password longer than 6 characters.<br>5. The user must choose username that does not exist in the database. |
| **Post Condition:** | Visitor now becomes a user of the app, gains access to all functionalities. |
| **Alternative Flow:** | 1. The email has not valid format. Display an error message. Wait for new entry.<br>2. The password is shorter than 6 characters. Display an error message. Wait for new entry.<br>3. Username exist in database. Display an error message. Wait for new entry. |

**Table 3.6:** Use-Case Descriptions

| | |
|---|---|
| **Use Case:** | Explore the Application |
| **Primary actor:** | Visitor |
| **Purpose:** | To access the all functionalities |
| **Overview:** | The use case begins when the application is opened as a visitor. Visitor can view the main page, events on the main page, cards in the market or leaderboard and search for cards. But can not purchase or sell cards and can not join the auction. Besides that, can not view the profil page and my collections page. If tries to do one of them, alternative flow is executed. |
| **Pre Condition:** | 1. The user must download the app.<br>2. The system must be running.<br>3. The user should continue as a visitor. |
| **Post Condition:** | Cards, market page, main page and achievements are uploaded but not able to perform any operation. |
| **Alternative Flow:** | If unauthorized functions are tried to be done, an error message is displayed and the user is expected to register. |

**Table 3.7:** Use-Case Descriptions

| | |
|---|---|
| **Use Case:** | Log in to the system |
| **Primary actor:** | Card Collector (user) |
| **Purpose:** | To gain access to the System |
| **Overview:** | The use case begins when the application is opened. The user initiates the login process to access the system. The user enters email and password. The system checks whether email and password matches in the database. If user provided correct credentials, user is navigated to app. But if not, alternative flow is executed. |
| **Pre Condition:** | 1. The user must download the app.<br>2. The system must be running.<br>3. The user provided valid credentials to successfully login to the system. |
| **Post Condition:** | 1. The user attempted to log in successfully, and as a result, the user's account is uploaded successfully. |
| **Alternative Flow:** | 1. The email and password does not match. Display an error message. Wait for the new entries. |

**Table 3.8:** Use-Case Descriptions

| | |
|---|---|
| **Use case:** | View Cards |
| **Primary actor:** | Card Collector (user) |
| **Purpose:** | To allow a user to view all available cards. |
| **Overview:** | The use case begins when a user logs into the Collectify application and expresses the intention to view the available cards. The system prompts the user to navigate to the "All Cards" section or cards based on the collections. Upon selection, the system retrieves related card data from the Web Server and displays the cards on the screen. |
| **Pre condition:** | 1.The user must be logged into the Collectify application.<br>2. User navigate to the home page or collections navbar. |
| **Post condition:** | 1. The user can view all available cards on the screen. |
| **Alternative Flows:** | 1. The user cancels the action. The process ends.<br>2. System cannot retrieve card data. Display an error message, and the process ends. |
| **Exceptions:** | Internet connection failure during the process. Display an error message, and the user is prompted to try again. |

**Table 3.9:** Use-Case Descriptions

| | |
|---|---|
| **Use case:** | Search Card |
| **Primary actor:** | Card Collector (user) |
| **Purpose:** | To enable a user to search for specific cards |
| **Overview:** | The use case begins when a user wants to find a specific card within the Collectify application. The system provides a search functionality where the user can enter keywords, such as card names. Upon entering the search query, the system retrieves matching cards from the Web Server and displays them on the screen. |
| **Pre condition:** | 1.The user must be logged into the Collectify application.<br>2. User navigate to the search bar. |
| **Post condition:** | The user can view a list of cards that match the search query. |
| **Alternative Flows:** | 1. The user cancels the search action. User can continue to use the app.<br>2. System cannot retrieve matching cards. Display an error message, and the process ends.<br>3. No cards match the search query. Display a message indicating no results. |

**Table 3.10:** Use-Case Descriptions

| | |
|---|---|
| **Use Case:** | Purchase a Card |
| **Primary actor:** | Card Collector (user) |
| **Purpose:** | To add a new card to the collection |
| **Overview:** | The use case begins when a user navigates to the market page.The user expresses the intention to purchase the card.The system prompts the user to confirm the purchase. Upon confirmation, the system checks the user's balance and the card's price. If the user has sufficient funds, the system deducts the purchase amount, adds the card to the user's collection. In case the user lacks funds or the card is no longer available, alternative flows are executed. |
| **Pre Condition:** | 1. The user must be logged into the Collectify application.<br>2. The user must have sufficient funds to make the purchase.<br>3.The card must be available for purchase. |
| **Post Condition:** | 1. The purchased card is added to the user's collection.<br>2. The user's balance is reduced by the purchase amount. |
| **Alternative Flow:** | 1. User cancels the purchase. Display a cancellation message, and the transaction ends.<br>2. User has insufficient funds. Display an error message, and the transaction ends.<br>3. The card is no longer available. Display an error message, and the transaction ends. |

**Table 3.11:** Use-Case Descriptions

| | |
|---|---|
| **Use case:** | View Own Card Collection |
| **Primary actor:** | Card Collector (user) |
| **Purpose:** | To view the card collection. |
| **Overview:** | The use case begins when a user logs into the Collectify application and expresses the intention to view their card collection. The system prompts the user to navigate to the "My Cards" section. Upon selection, the system requests the user's ID, retrieves the collection data from the Web Server, and displays the user's collection on the screen. |
| **Pre condition:** | 1.The user must be logged into the Collectify application.<br>2. User navigate to the "My Cards" section. |
| **Post condition:** | 1. The user can view his/her card collection on the screen. |
| **Alternative Flows:** | 1. The user cancels the action. The process ends.<br>2. System cannot retrieve collection data. Display an error message, and the process ends.<br>3. No cards exist in the collection. Display a message indicating an empty collection. |
| **Exceptions:** | Internet connection failure during the process. Display an error message, and the user is prompted to try again. |

## 3.4. USER SCENARIOS

Bilgehan wants to buy one of the cards sold at auction. If the current price of the card he will bid on is more than his own balance, he cannot bid. If his balance is sufficient, he can make an offer. If someone else bids more than Bilgehan’s bid and Bilgehan’s balance is not enough to bid again, he cannot bid. When the auction ends, if Bilgehan is the person with the highest bid, he wins the auction, the amount of bid is deducted from his balance and the card is added to his collection.

İrem wants to register to the application. On the registration page, she sees that he needs to fill in the Email and password sections. When she uses an email that does not comply with the specified email format, she receives a "Please enter a valid email address" warning. When İrem uses an email that fits the format and wants to set a password with fewer than 6 characters, she gets "Password must be longer than six characters” warning. When İrem enter a password with more than 6 characters, she can register to the application. After registration she need to choose a username and if wants a profile image, if username is taken she have to choose another name. After that she can navigate to the app.

Mustafa opens the application. He presses the "daily spin" button on the home page. He spins a wheel with various cards and different amounts of game coins in its sections. If he wins a card as a reward, that card is added to Mustafa’s own collection. If he wins game coins, the amount won is added to his balance. When Mustafa wants to spin the wheel again, he receives "the wheel can only be turned once a day" warning. He can only spin the wheel the next day.

Bensu wants to sell one of the cards she owns. Chooses how the sale will be made from the card’s own page. If Bensu wants it to be sold at a fixed price, she sets a price for the card and waits for the card to be purchased by someone else. After the sale is made, the determined amount is added to the her balance. If Bensu wants to sell it at auction, she determines a starting price for the card and the duration of the auction, and the card is added to the market to be sold. After the auction ends, the amount of highest bid is added to her balance. In both cases the card is removed from Bensu’s own collection.

Seyfullah is someone who visits the application and opens it. He can view events in the application. He can view cards being sold both at fixed price and by auction. He can view the prices and features of cards. But he cannot buy a card sold at a fixed price or bid on a card at auction. Since he is not logged in to the application, he cannot view his

account information. To perform these actions, it is necessary to register or log in to the application.

Tanay loves collecting cards and wanted to add a special one to her collection. To make this process easy, she can use the search tool in the app. She can type the name of the specific card she’s looking for and click on the search button. If there is one, the app presents her with the card she desires, along with detailed information about it.

Çağın loves collecting cards and earning achievements, which is like a game within a game! If she gathers a set of cards, she can get cool rewards. If the reward is game coins, the amount won is added to her balance. If she wins a card as a reward, that card is added to her own collection. İrem can earn rewards another way also. If she actively participate in auctions, buying, and selling cards, she might find herself at the top of the leaderboard. Being number one comes with rewards.

Sait, as one of the administrators in Collectify, wants to enhance the app’s content by adding collections and cards. When he decides to introduce a new collection, he simply opens the collection creation form and gives the collection a unique name. If the chosen name isn’t already in use, the new collection is successfully created. Once the collection is set up, Sait can start adding unique cards to it. Each card needs a distinct title to be added. Once the collection is complete, users can explore these new cards, buy them, and sell them.

# 4. USER INTERFACE MODEL

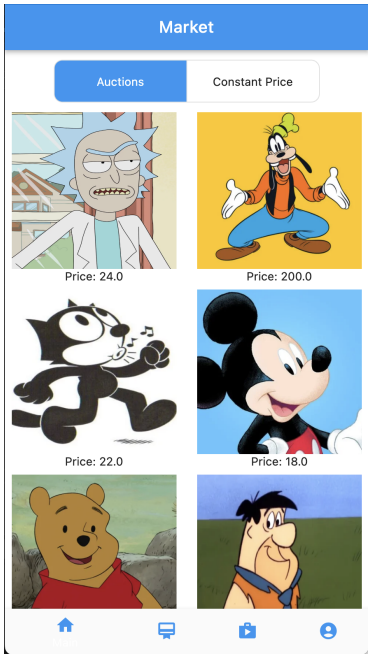


**Figure 4.1:** Market for Auctions

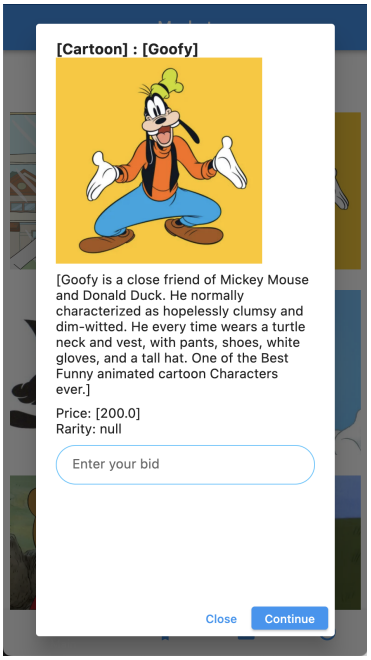


**Figure 4.2:** Bid at Auction

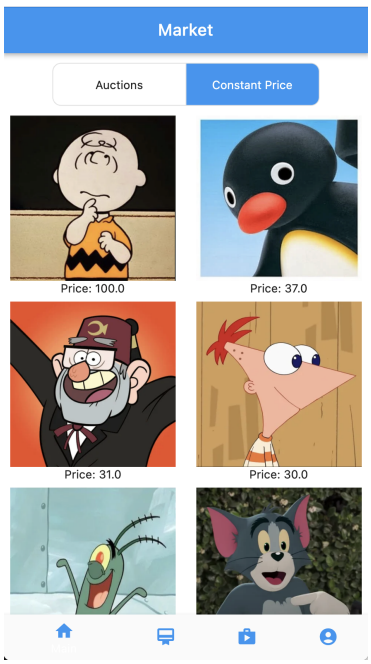


**Figure 4.3:** Market for Fixed Price

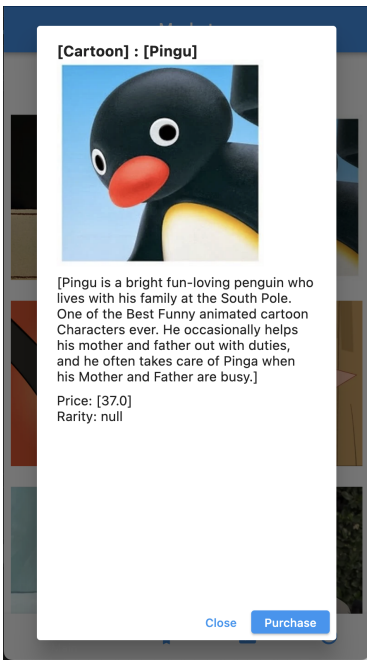


**Figure 4.4:** Purchasing Card

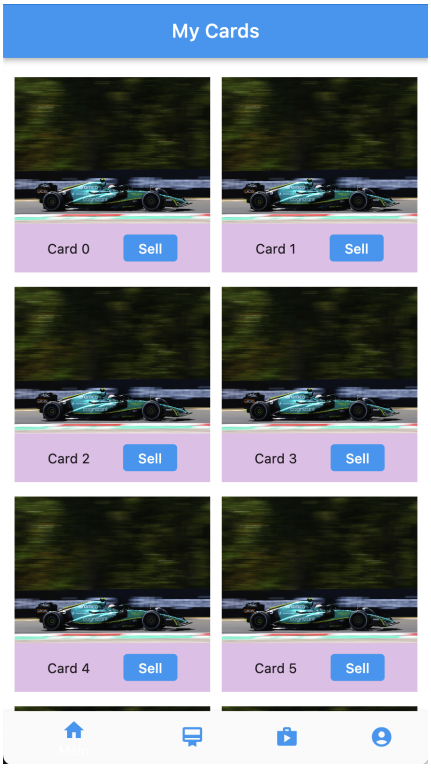


**Figure 4.5:** User's Card List

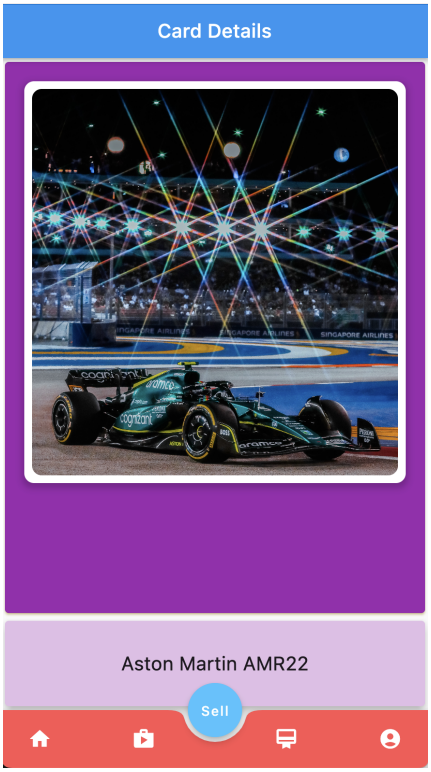


**Figure 4.6:** Card Details

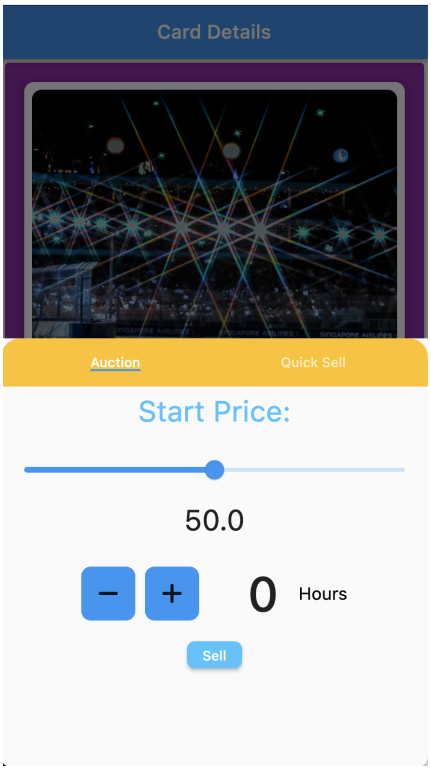


**Figure 4.7:** Selling Card in Auction

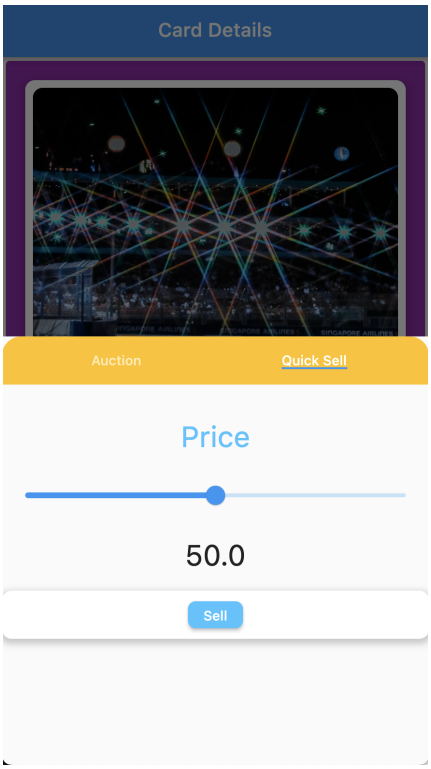


**Figure 4.8:** Selling Card with Fixed Price

One of our use cases involves a user navigating to the auction page and bidding on their preferred card, as shown in Figure 4.1 and Figure 4.2. The page displays a list of images, along with detailed descriptions and the current bid for each card. In the images, users can also place their bid if their account balance is sufficient. Additionally, in Figure 4.7 they have the option to sell their card in an auction using a sliding price bar and set a time limit. This allows them to choose the speed and competitiveness of the sale and start the timer accordingly. Users also can sell their cards with fixed price option as seen in Figure 4.8 After the transaction is successfully completed, the user's balance is updated.

Another use case is the fixed-price feature, which implements a quick buy system for users to access and choose cards more efficiently. The market page features a second bar displaying constant prices(Figure 4.3). Users can thoroughly examine their selected card and instantly add it to their collection(Figure 4.4).

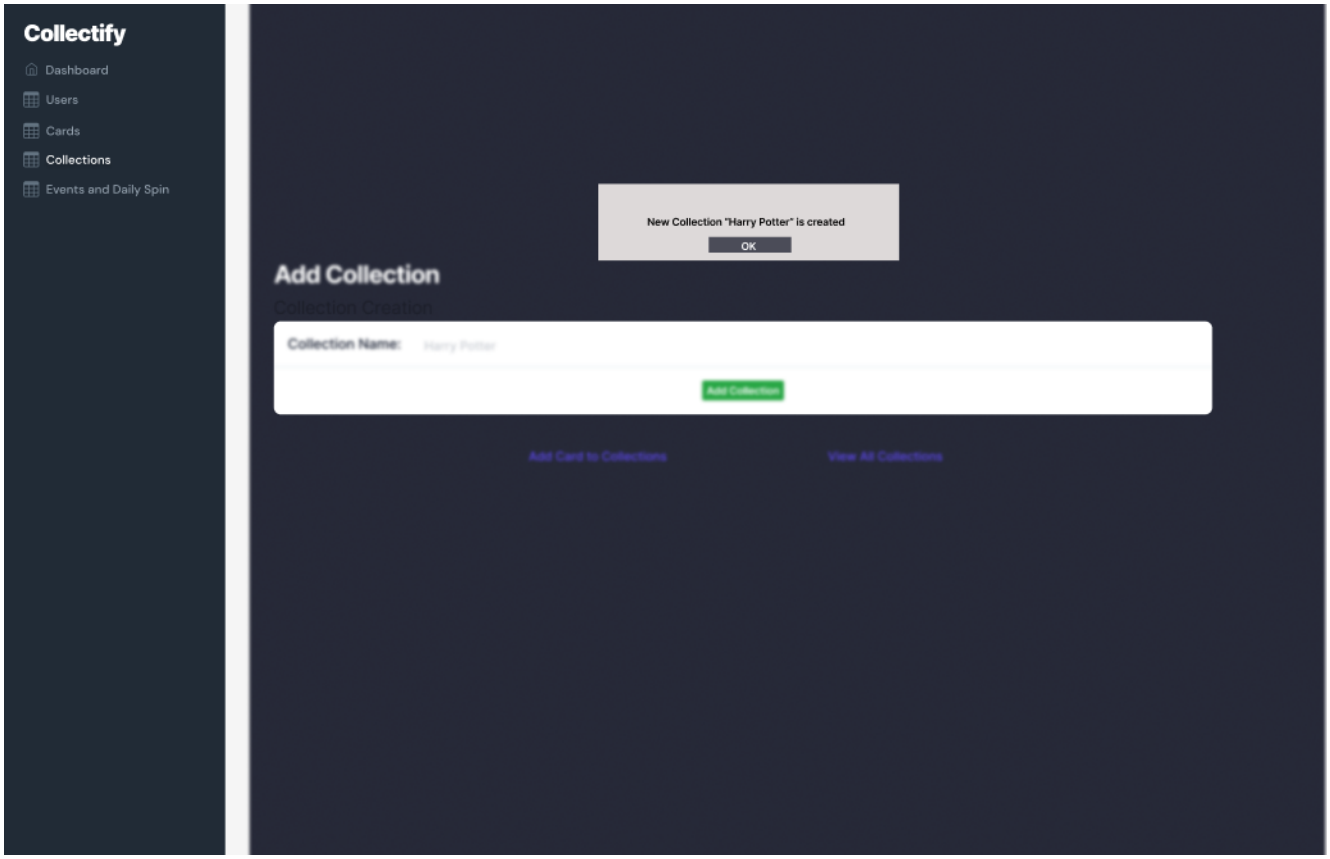


**Figure 4.9:** Adding Harry Potter Collection

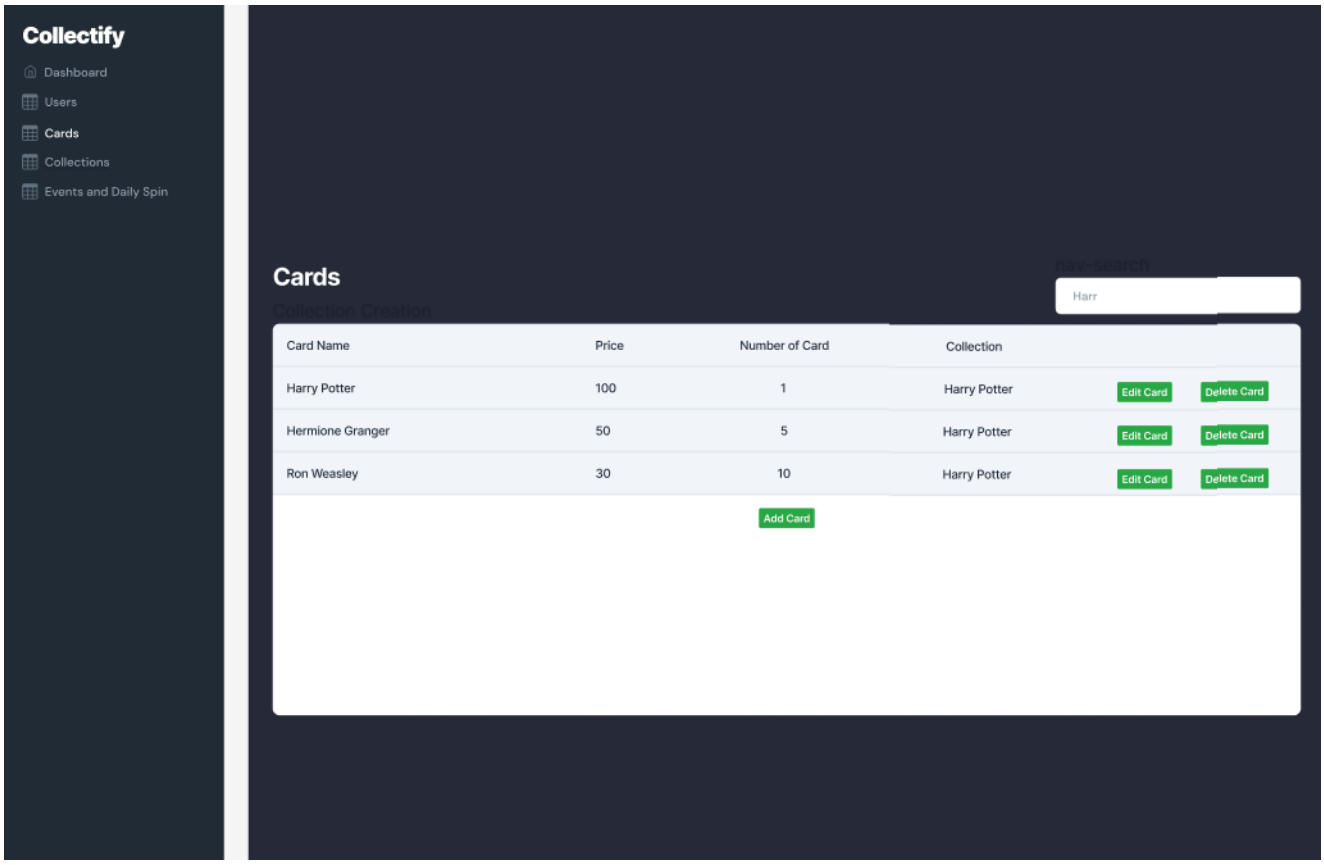


**Figure 4.10:** Using Search Function in Cards

# 5. FLOW DIAGRAMS

## 5.1. GENERAL DATA MODEL

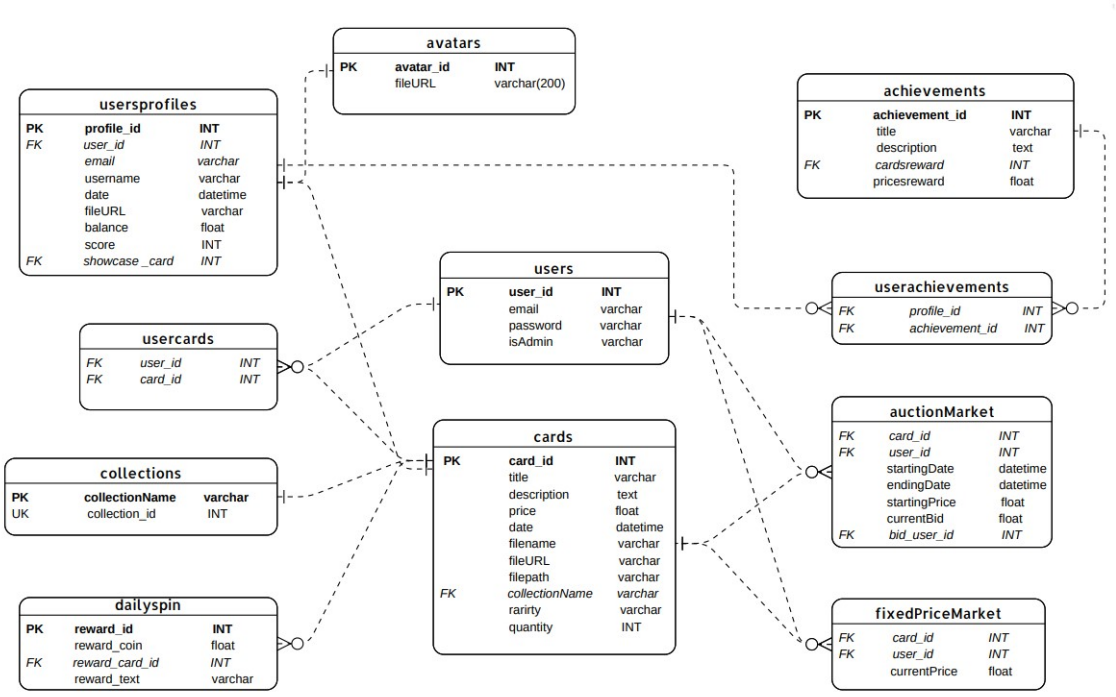


**Figure 5.1:** Entity Relationship Diagram

## 5.2. IMPORTANT DATA CONSIDERATIONS

In our backend development, we Utilize JSON as the primary data interchange format. JSON, or JavaScript Object Notation, provides a structured and lightweight approach to representing data, offering a clear and standardized means of communication between the backend and frontend components of our application. This format, comprised of key-value pairs, is highly versatile, accommodating a variety of data structures, including arrays. By adopting JSON, our backend ensures efficient and seamless interactions with the frontend, particularly beneficial when responding to client requests or handling API transactions. This approach enables our system to easily exchange and process data while promoting compatibility with frontend. As part of our backend processes, we commonly use the term "jsonify" to describe the transformation of data into the JSON format, facilitating a smooth and standardized flow of information within

our application architecture. There is a example below :

userdata =
’username’: username,
’email’: email,
’registrationdate’: date,
’fileURL’: imageURL

return jsonify(userdata)

## 5.3. DATA FLOW DIAGRAM

Data Flow Diagram (minimum 2 levels (Level 0 - Level 2)) and its explanation.

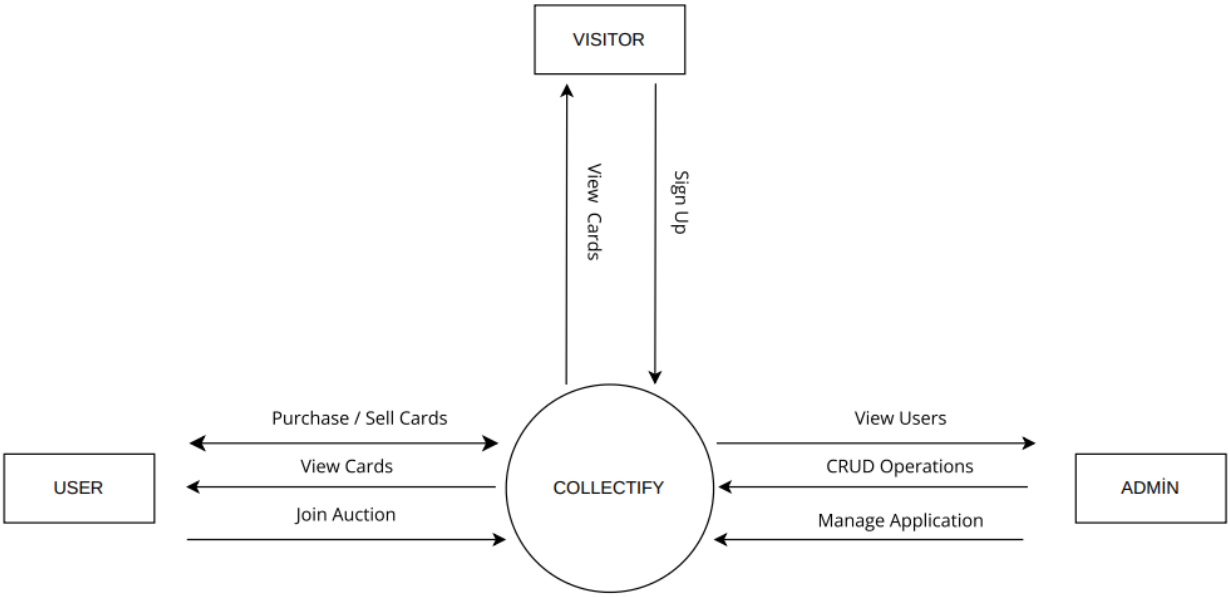


**Figure 5.2:** Data-Flow Diagram - Level 0

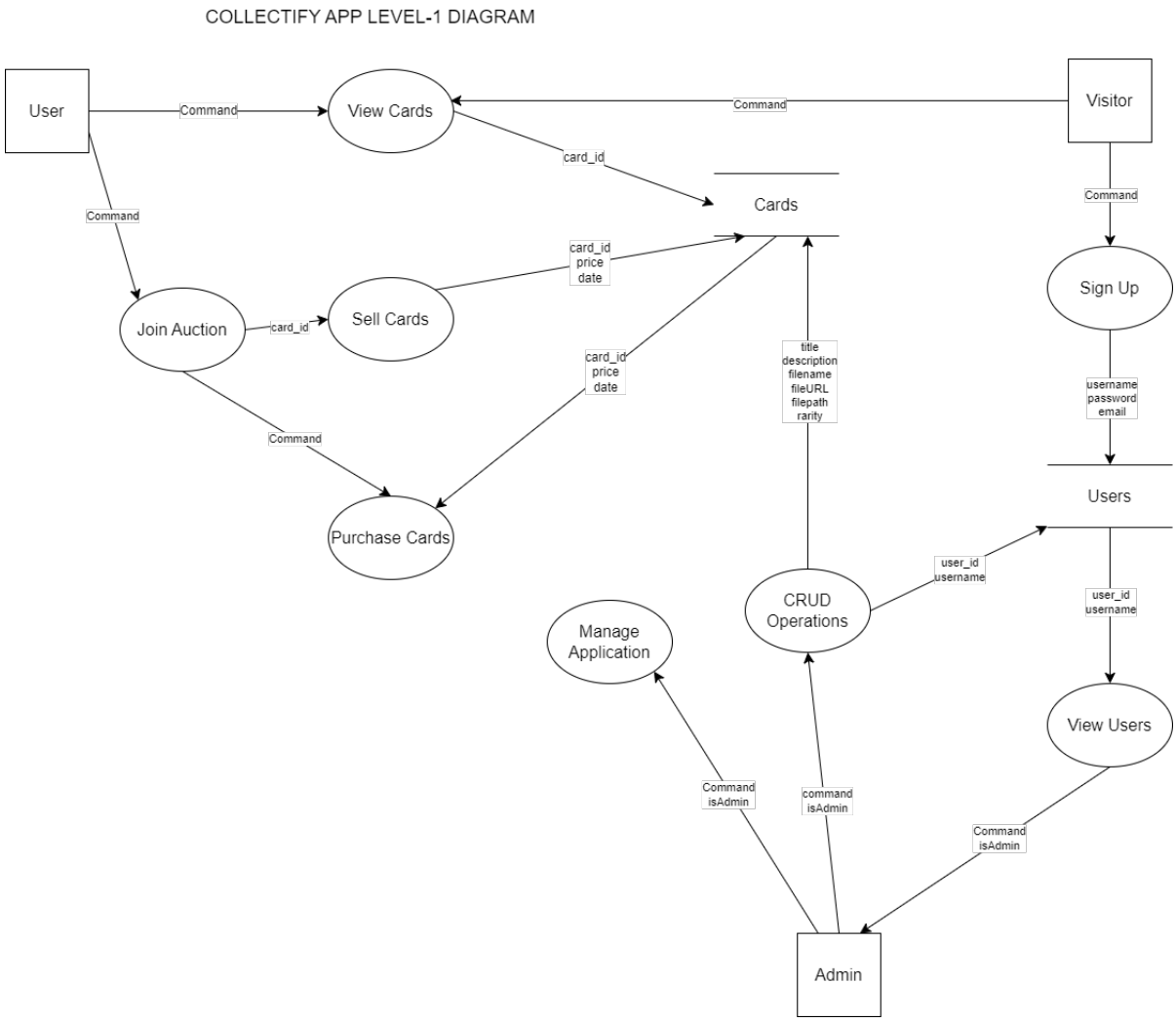


**Figure 5.3:** Data-Flow Diagram - Level 1

Level 1 chart shows general detailed structure of our project that covers the most crucial functions and tasks. By looking at the flow chart we can easily understand the basic methods of card transactions, admin controls and even a visitor that can interact with the app by signing up or just viewing what the app offers. The diagram also focuses on what data is being transfered or used. In example to sell a card the app needs the info cardId, price and date as key data. Some functions may also start by just User "Command". This means that no data transfer is needed until the necessary conditions are met.

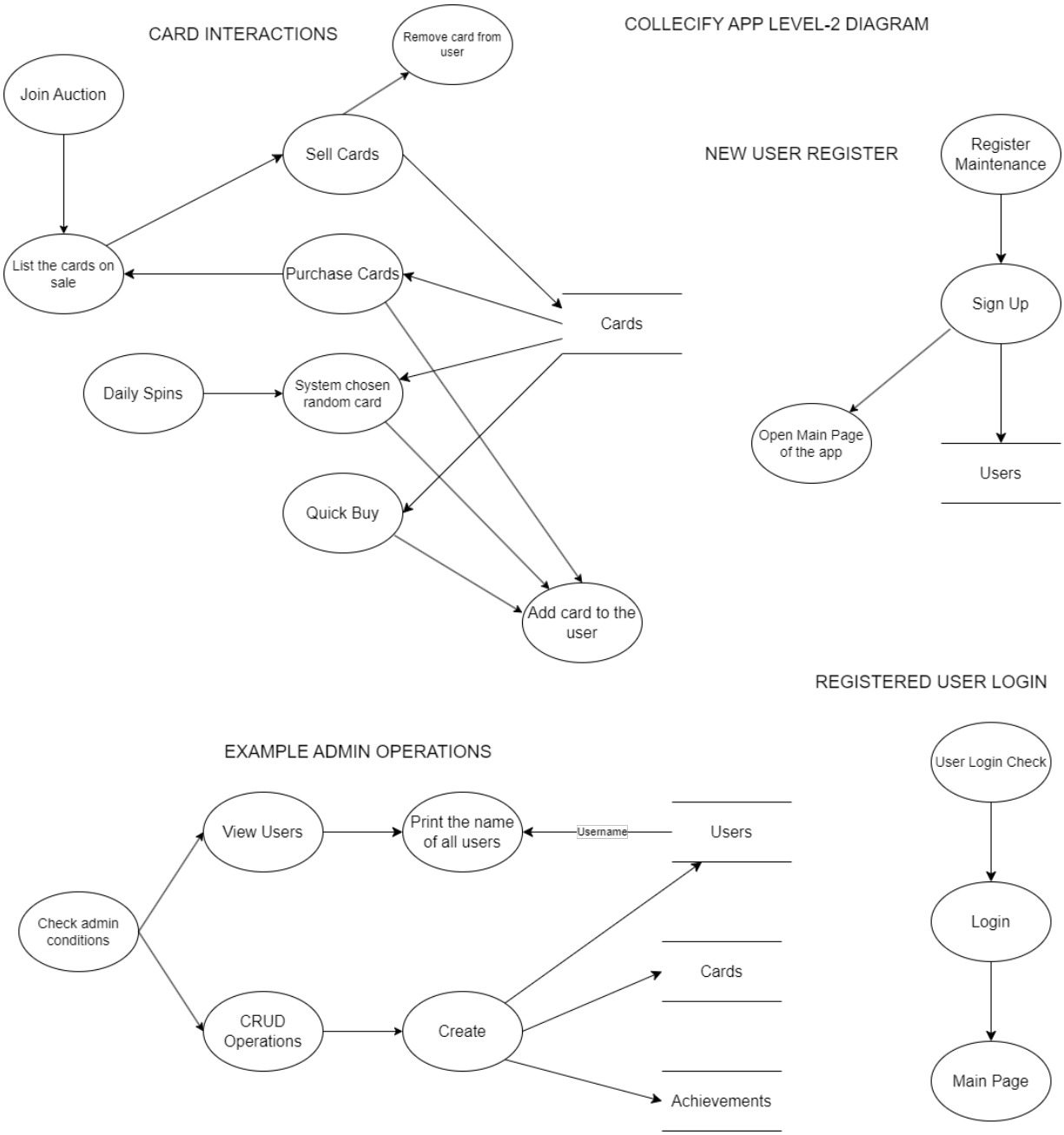


**Figure 5.4:** Data-Flow Diagram - Level 2

Level 2 chart focuses on specific details of certain tasks in more detail. Instead of looking at the whole app it provides insight on how certain important functions are implemented in the app. The card interactions shown in the table is the whole card trading system currently planned or started to be implemented in the app. There are different methods of trading or purchasing cards. First is the auction system where the users put their cards on some amount of time and other users give offer on it. The second is the daily spin where the user spins a wheel of luck and earns bonus card. The third method of getting a card is quick-buy some specific cards from the game store. The table provides context for example admin operations, first time user register and registered user login. By looking at the diagrams the flow of these functions can be understood. Some features of the app needs an admin account like CRUD operations or listing all users. These can be accessed if the admin conditions are met. The diagrams also shows the Register Maintenance for visitors that want to sign up to the app.