Software Requirements Specification

JUNK-BROKER APPLICATION

CSE-682 SOFTWARE ENGINEERING

PROF. HESHAM SAADAWI

3/24/2023

Anthony Redamonti
Michael Peña
Clayton Colson
Kenneth Chalupa
SYRACUSE UNIVERSITY

Table of Contents

Introduction

Purpose

The junk-broker system is an interactive transaction-based web application tailored toward the junkyard industry. Junkyards, otherwise known as salvage or scrap yards, are yards containing large quantities of salvaged spare parts or scrap metal. Many junkyards specialize in storing nonfunctional or damaged vehicles, such as cars, trucks, SUV's, motorcycles, boats, ATV's, and other commercial types. One of the most common reasons for visiting a junkyard is to search for functional vehicular replacement parts because they are sold at a discounted rate compared to a new or original manufacturer part, or for parts that are no longer manufactured.

Scope

The junk-broker system offers an easier method of buying and selling vehicular parts through an organized interactive application. Buyers and sellers access the system via a device web browser. The seller uploads detailed listings of their inventory to their profile. Buyers can search active listings and filter based on price, location, vehicle type, year, make, and model. Buyers and sellers can send messages to each other, and purchases can be finalized through a third-party credit service (PayPal, Venmo, etc.). All profiles and listings are stored in a database hosted by several secure servers. The system is available at all times to buyers and sellers worldwide.

User Requirements (Team)

UR1. The Junk-Broker system shall allow anonymous users to create an account with the following roles to later be used securely:

- UR1.1: Buyer Role
- UR1.2: Seller Role

UR2. The Junk-Broker shall allow users to log-in, authorizing the option to perform ‘Buyer’ or ‘Seller’ specific role actions.

UR3. The system shall list available listings to the buyer and include filters that can filter merchandise from multiple sellers. The filters shall be as follows:

- UR3.1: Price
- UR3.2: Location
- UR3.3: Vehicle Type
- UR3.4: Year
- UR3.5: Make
- UR3.6: Model

UR4. The junk-broker system shall be accessible from a device web-browser. Devices shall include desktop and mobile (phone/tablet) systems.

UR5. The junk-broker system shall follow industry-based design standards and guidelines.

UR6. Sellers shall be able to create a listing after providing the following required information regarding the listing:

- UR6.1: Vehicle Type
- UR6.2: Year
- UR6.3: Make
- UR6.4: Model
- UR6.5: Number of previous owners
- UR6.6: Location
- UR6.7: Price
- UR6.8: Description
- UR6.9: Photos of vehicle

UR6.10: The system may present the seller with optional fields to append to the new listing. The optional fields are related to personal contact information and are as follows:

- Email Address
- Phone Number

UR7. The system shall provide a performance page that is private to the seller. It shall include the following features:

- UR7.1: The seller shall be able to see his performance rating given by the sellers. The scale shall be from zero to five stars in half star increments.
- UR7.2: The seller shall be able to see the total number of followers of their profile.
- UR7.3: The total number of views for each listing in the past seven days shall be shown.

UR8. The system shall provide the following private features on the profiles of both the buyer and seller:

- UR8.1: A complete transaction history
- UR8.2: A messaging feature of received and sent messages

UR9. All active listings shall be presented to the seller. The seller shall be able to edit and change the status of active listings.

UR10: A user shall be able to mark a listing as a 'favorite' that will produce a list for them to view at all items.

- UR10.1: view all 'favorites' from a favorites page
- UR10.2: filter 'favorites' listings

UR11: A user shall receive notifications if a favorited listing changes in state.

- UR11.1: Change in Listing status
- UR11.2: Change in Listing Price
- UR11.3: Change in Location

UR12: A user shall be able to track the shipping status of the order.

- UR12.1: The system shall display the current tracking information to the user.
- UR12.2: The system shall notify the user to any changes or delays to the order status.

UR13: A user shall be able to enter search text on the screen.

- UR13.1: A user shall be able to enter multiple keywords in the search text.
- UR13.2: The system shall return results matching the keywords entered by the user.
- UR13.3: The system shall limit the matching results displayed on the screen to a default value of 10 results.
    - UR13.3.1: The user shall be able to increase the number of matching results displayed on the screen to 50 results.
- UR13.4: The system shall notify the user when no matching product is found based on the user's search.

UR14: A buyer shall be able to add an item to a cart and initiate a Point of Sale of item on listing.

UR15. The junk-broker database shall securely store customers' private data.

UR16. The junk-broker system shall be available to customers at all times.

System Requirements (Team)

SR1: When a user creates an account in the Junk-Broker system, their information and role shall be securely stored in an encrypted database.

SR2.1: A unique user Id shall be mappable to the user's information and role.

SR2.2: Users shall not have the ability to change their roles or have any access to their role assignment.

SR2.3: Each user shall be only allowed 1 role assignment: either 'Buyer' or 'Seller'.

SR3.1: When no filters are selected, the listings should be sorted by location within 20 miles of the buyer by default.

SR3.2: There shall be a filter feature, that when activated, provides accessibility to all the filters described in UR3. The settings for each filter shall be accessible from the filter feature. The filter feature shall have the ability to clear all filters in use and the ability to apply the filters to the listings. The feature shall also allow the user to return to viewing active listings.

SR3.3: Each filter setting in the filter feature shall have three features:

- SR3.3.1: The ability to return to the main filter feature without saving the changes to the filter.
- SR3.3.2: The ability to reset the specified filter setting.
- SR3.3.3: The ability to save the specified filter setting.

SR3.4: The vehicle type filter setting shall allow the user to select one or more of the following vehicle type options: Cars, Trucks, SUV's, Sedans, Vans, Motorcycles, Boats, Commercial Vehicles, ATV's, and Bicycles. There will also be an option "any" to select all of these vehicle types.

SR3.5: The price filter setting shall prompt the user to enter the minimum and maximum values of the search.

- SR3.5.1: The user shall not be able to enter a minimum or maximum price less than zero.
- SR3.5.2: The user shall not be able to enter a minimum price greater than the maximum price or a maximum price less than the minimum price.

SR3.6: The location filter setting shall prompt the user to enter a zip code and a search radius in miles originating from the zip code in which to search.

- SR3.6.1: The user shall not be able to enter an unknown or improperly formatted zip code.
- SR3.6.2: The user shall not be able to enter a search radius less than zero miles.

SR3.7: The Make filter feature shall provide the user with the ability to choose one or more vehicle makes listed among the top 25 most popular vehicle make types. There will also be an option "any" to select all make types.

SR3.8: The Model filter feature shall provide the user with the ability to choose from all models offered by the vehicle manufacturers selected in the Make filter feature. The user must select at least one vehicle manufacturer in the Make filter feature before using the Model filter feature. There will also be an option "any" to select all Model types.

SR3.9: The Year filter feature shall prompt the user to enter the minimum and maximum years between which the manufacturing date is specified.

- SR3.9.1: The user shall not be able to enter a minimum or maximum year less than zero.
- SR3.9.2: The user shall not be able to enter a minimum year greater than the maximum year or a maximum year less than the minimum year.

SR4.1: The junk-broker system shall be a web-based application, accessible from a device browser.

SR4.2: The front-end Graphical User Interface (GUI) shall be optimized for full functionality when used on a desktop or tablet (in landscape mode) browser window. The desktop view shall be the primary means of control and interaction with the application and shall offer the “Seller” and “Buyer” views.

SR4.3: Mobile device browsers shall make use of portrait orientation and have a reduced feature set, restricted to the “Buyer” view.

SR5.1: The junk-broker front-end GUI shall incorporate W3C’s Web Accessibility Initiative - Web Content Accessibility Guidelines (WCAG) design principles.

SR5.2: The GUI shall be perceivable:

- 5.2.1: Non-text buttons/controls shall be named according to their purpose.
- 5.2.2: The GUI shall present content in different ways without losing information or structure.
- 5.2.3: The GUI shall be distinguishable, making it easier for users to see by including appropriate use of color and contrast, or differentiating between foreground and background tasks.

SR5.3: The GUI shall be operable.

- 5.3.1: All functionality shall incorporate text input controls and provide functionality beyond standard keyboard input.
- 5.3.2: Time-dependent features shall be adjustable.
- 5.3.3: Content shall not be designed in a way that is known to cause seizures or physical reactions.
- 5.3.4: The GUI shall provide users clear ways to navigate, find content, and determine their current location.

SR5.4: The GUI shall be understandable.

- 5.4.1: Content shall be readable and predictable.
- 5.4.2: Input fields shall provide assistance and suggestions.

SR5.5: The GUI shall conform to Level A standards of the WCAG 2.1 standard.

SR6.1: The system shall not allow the user to create a listing without populating all the fields specified in UR6 with a valid entry. If the user attempts to create a listing that is missing information from one or more of these fields, an error message shall be displayed specifying the fields that require attention.

SR6.2: The system shall immediately upload all images and related listing content to the database upon the creation of a listing.

SR6.3: The system shall not allow the entry of invalid data into a given field. Examples of invalid data include improper formatting of a phone number or email address or uploading unsupported file formats into the photos section.

SR7.1: The buyer shall be able to rate a seller only after purchasing an item from the seller.

SR7.2: In addition to being displayed on the seller's private performance page, the average rating shall be publicly displayed on the seller's profile for buyers to review.

SR8.1: All messages and transactional data shall be stored in the junk-broker database. Only listings that are sold shall be considered transactional history.

SR8.2: The messaging feature shall include an inbox/outbox that utilizes Simple Mail Transfer Protocol (SMTP).

SR8.3: The messaging feature shall include functionality to create a new message, send a message, respond to an existing message, and delete selected message(s).

SR9.1: Editing an existing listing shall be subject to the same requirements specified in SR6.1, SR6.2, and SR6.3.

SR9.2: When the status of a listing is marked as missing, stolen, or sold, or if the listing is deleted by the seller, it shall be removed from the list of active listings.

SR10.1: The system shall allow the buyer to mark a listing as a favorite.

SR10.2: Each listing marked by the buyer as a 'favorite' shall be added to a favorite listings collection for further use and metrics.

SR11.1: Each favorited listing shall register a hook that shall trigger upon any listing change.

SR11.2: The changed status of the listing shall be sent to the potential buyer's mailbox.

SR 11.3: Use of any basic pub/sub framework (such as Kafka, Redis Message Broker, Azure Service-Bus) shall publish the listing state change events to all buyers/subscribers who 'favorited' that listing.

SR12.1: The system shall present a final detailed invoice to the current order once it is confirmed by the user.

SR12.2: The system shall provide the option to send the invoice to the user's profile email.

SR13.1 The system shall provide a shopping cart to facilitate online purchases.

SR13.2 The system shall allow the user to edit the shopping cart by adding or removing products located in the user's specific shopping cart instance.

SR14.1: The system shall display current open orders that are available to change.

SR14.2: The system shall allow the user to select an order to be changed.

SR14.3: The system shall provide an option to cancel an open order.

SR14.4: The system shall notify the user about any changes that have been made to the order.

SR15.1: The junk-broker database shall use a secure server that uses a Firewall to protect against attackers. Customers and sellers shall rely on the security of the server to protect their private information in the database.

SR15.2: All network traffic shall be encrypted to prevent hijacking and injection attacks.

SR15.3: If a data breach is detected, the following events shall take place:

- SR15.3.1: Log-in's, passwords, and PIN's shall be secured.
- SR15.3.2: The sensitive data that was stolen shall be identified.
- SR15.3.3: A claim shall be filed with the Federal Trade Commission.
- SR15.3.4: All buyers and sellers shall be notified of the breach via email.

SR16.1: The junk-broker database shall be hosted on a backup server in case the main server goes offline. If the main server crashes, the backup server shall be used to keep the database online while the main server is under repair.

SR16.2: The maximum allowable system downtime shall not exceed 1 hour.

SR16.3: The backup server shall be housed in a secure and separate location than the main server. This shall provide Geo-Replication and proper redundancy.

SR16.4: In case of large amounts of pending requests, the main server shall offload a percentage of its workload to the backup server to maintain system responsiveness. The percentage shall be calculated based on the number of pending requests and the resources available to the main server.

SR16.5: Use of Container-Orchestration Services such as (Kubernetes, Rancher etc.) may be utilized to scale backend services to provide elastic scaling to match increasing or decreasing demand.

Priorities (Team)

Priority 1: Users access Junk-Broker System Web Browser or Mobile Device

- UR4 -> SR4.1, SR4.2, SR4.3 (web page/mobile app available)

Priority 2: Allow Users to Create Account/Log-in as 'Buyer' or 'Seller'

- UR1 -> SR1.1, SR1.2, SR1.3 (users can create an account)
- UR2 -> SR2.1, SR2.2, SR2.3 (users can login to account)

Priority 3: Allow Users to perform role specific functions within Junk-Broker

- UR6 -> SR6.1, SR6.2, SR6.3 (sellers can create listings)
- UR9 -> SR9.1, SR9.2 (sellers can edit listings)
- UR3 -> SR3.1, SR3.2, SR3.3, SR3.4, SR3.5, SR3.6 (buyers can view/filter listings)

Priority 4: Transactions

- UR14 -> SR12.1, SR12.2, SR13.1, SR13.2, SR14.1, SR14.2, SR14.3, SR14.4 (Sale of listed vehicles)

Priority 5: User to User interactions

- UR8.1 -> SR8.1 (Transaction history available to users)
- UR8.2 -> SR8.2, SR8.3 (Users can message other users)

Priority 6: Metrics Collection and Notifications

- UR7 -> SR7.1, SR7.2 (Buyers can access interest level metrics of listings)
- UR10 -> SR10.1, SR10.2 (Users can mark listings as 'Favorite')
- UR11 -> SR11.1, SR11.2, SR11.3 (Listing change notifications)

Priority 7: Availability and Accessibility

- UR16 -> SR16.1, SR16.2, SR16.3, SR16.4 (Functionality must match increasing demand)
- UR5 -> SR5.1, SR5.2, SR5.3, SR5.4, SR5.5 (W3c Web Accessibility Standards to user)

System Architecture

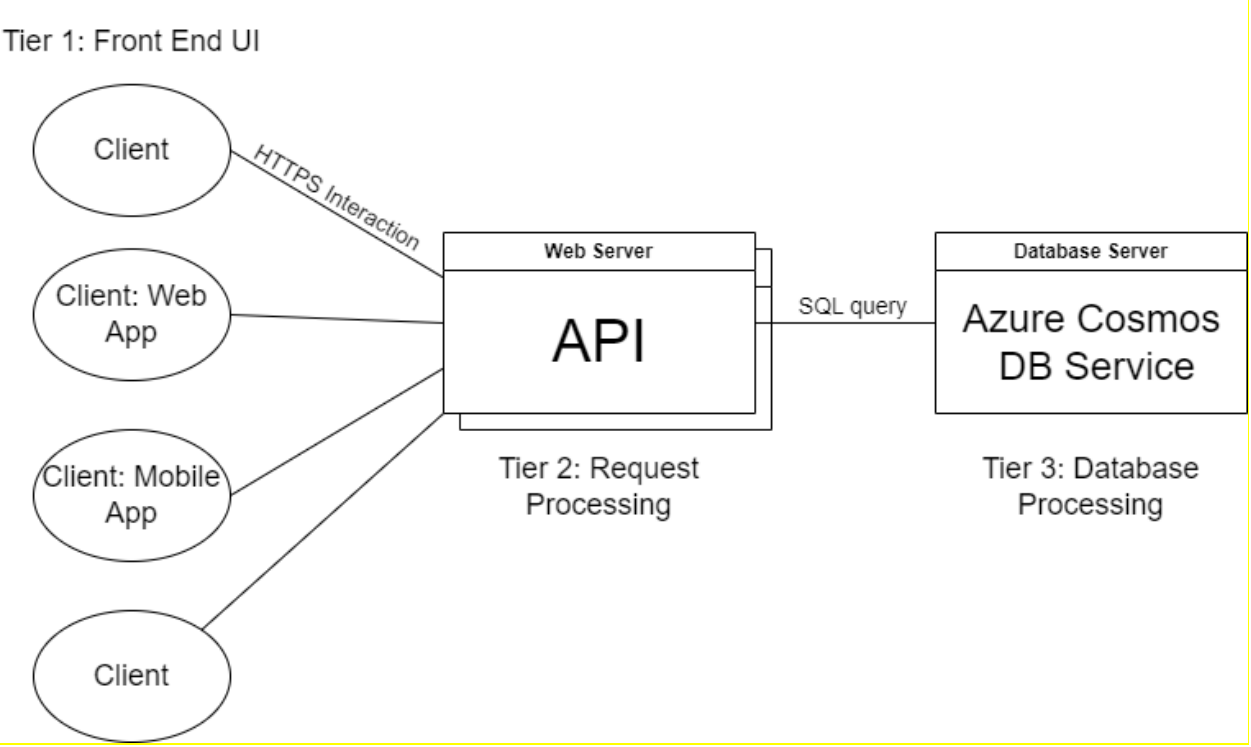
The Junk-Broker system is a transaction-processing information system. It is composed of a 3-tier structure defined in the table below.

Tier	Name	Functionality
1	Front End UI	- Provides an interface to the client - Sends user-generated asynchronous events to the API in the form of HTTPS messages
2	API	- A web server that processes HTTPS messages from the front end and sends read/write SQL queries to the database
3	Database	- A database server hosted on Azure Cosmos DB Service - Stores all junk-broker system data

The transaction-processing information system is most suitable for this type of application. The clients can communicate with the API web servers through multiple methods: Web or Mobile App. The API web server is containerized meaning that there can be more than one. If one API web server is down, another can take its place. Also, in the case of many pending HTTPS requests, the work can be shared between multiple API servers.

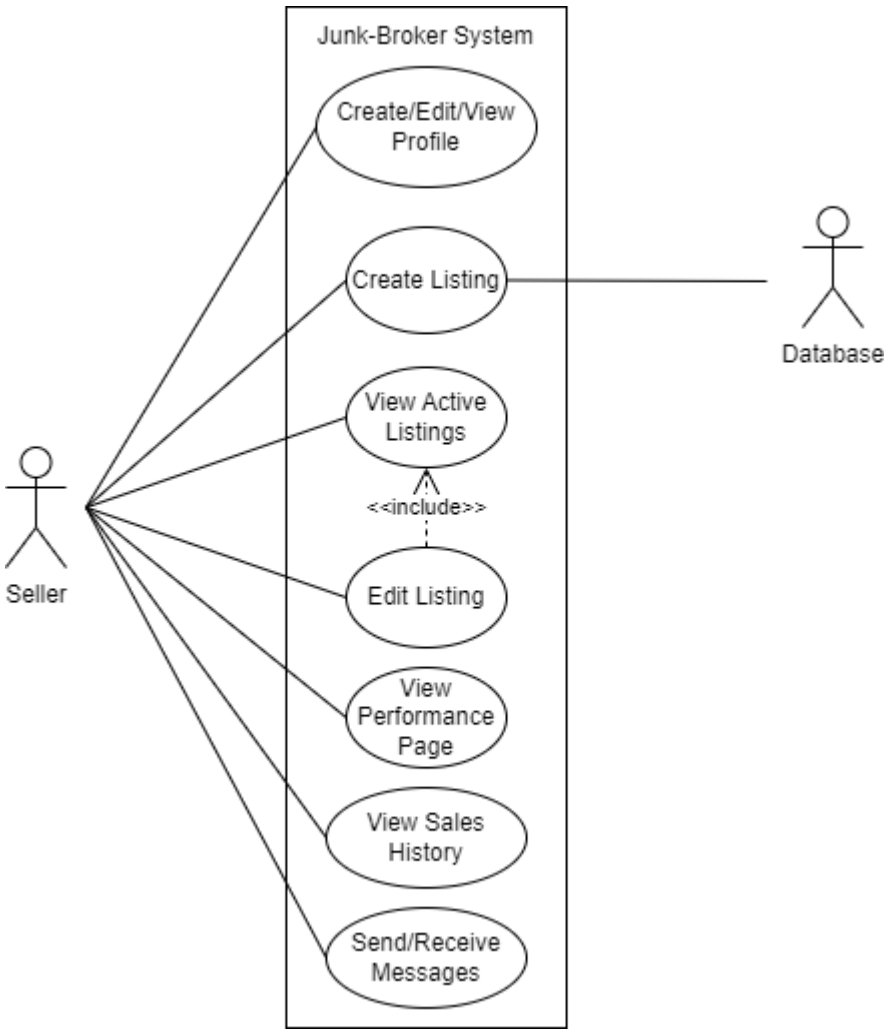
The API web server is used to communicate to the Azure Cosmos DB Server (database server). It will send read/write commands to the database in the form of SQL queries. Updating the database is handled through asynchronous tasks that are non-blocking to the application. Essentially, a separate thread is used to process each query. Using non-blocking tasks (multi-threading) was essential to the performance of the application, as it prevented the system from freezing up during the event handling process.

System Architecture Diagram (Team)

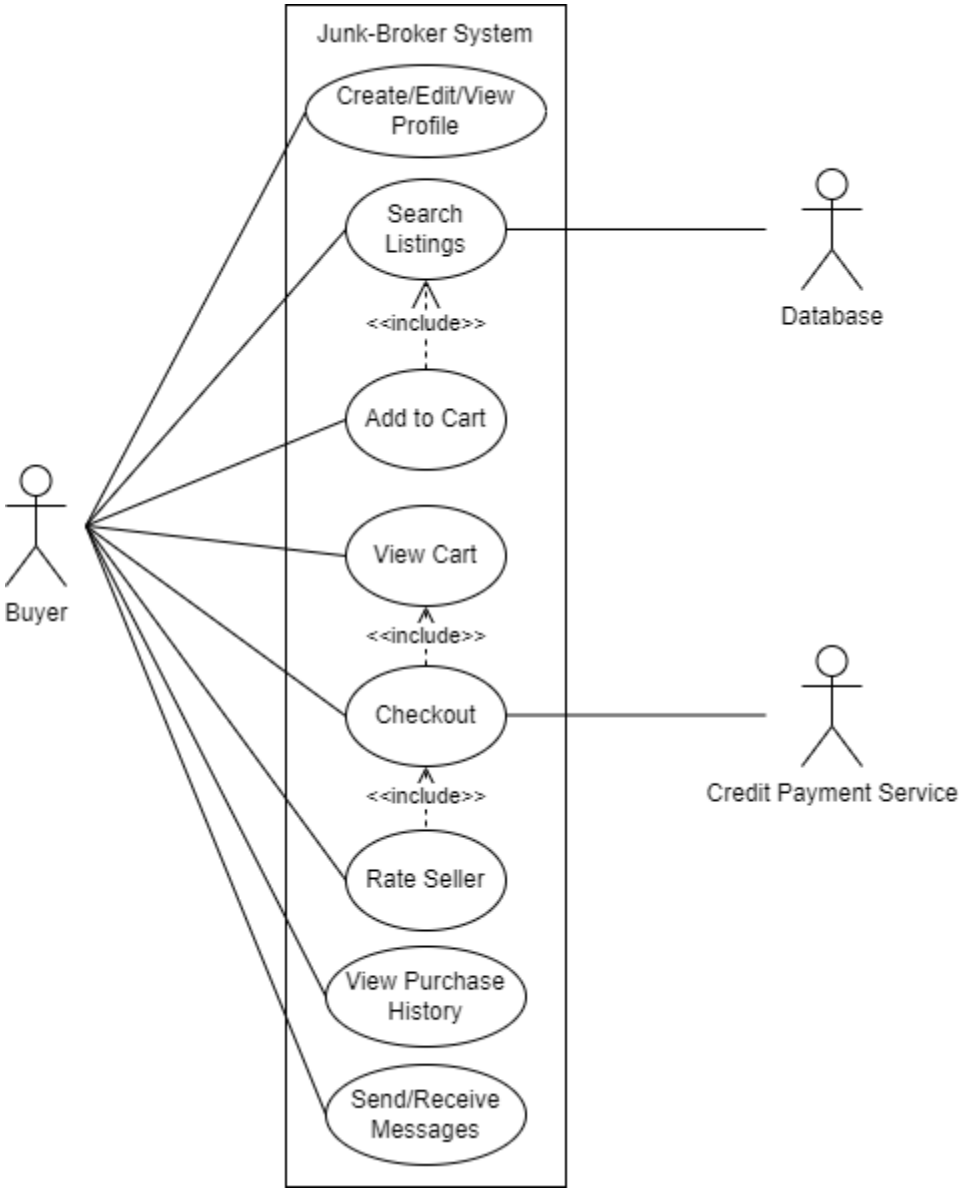


Diagrams

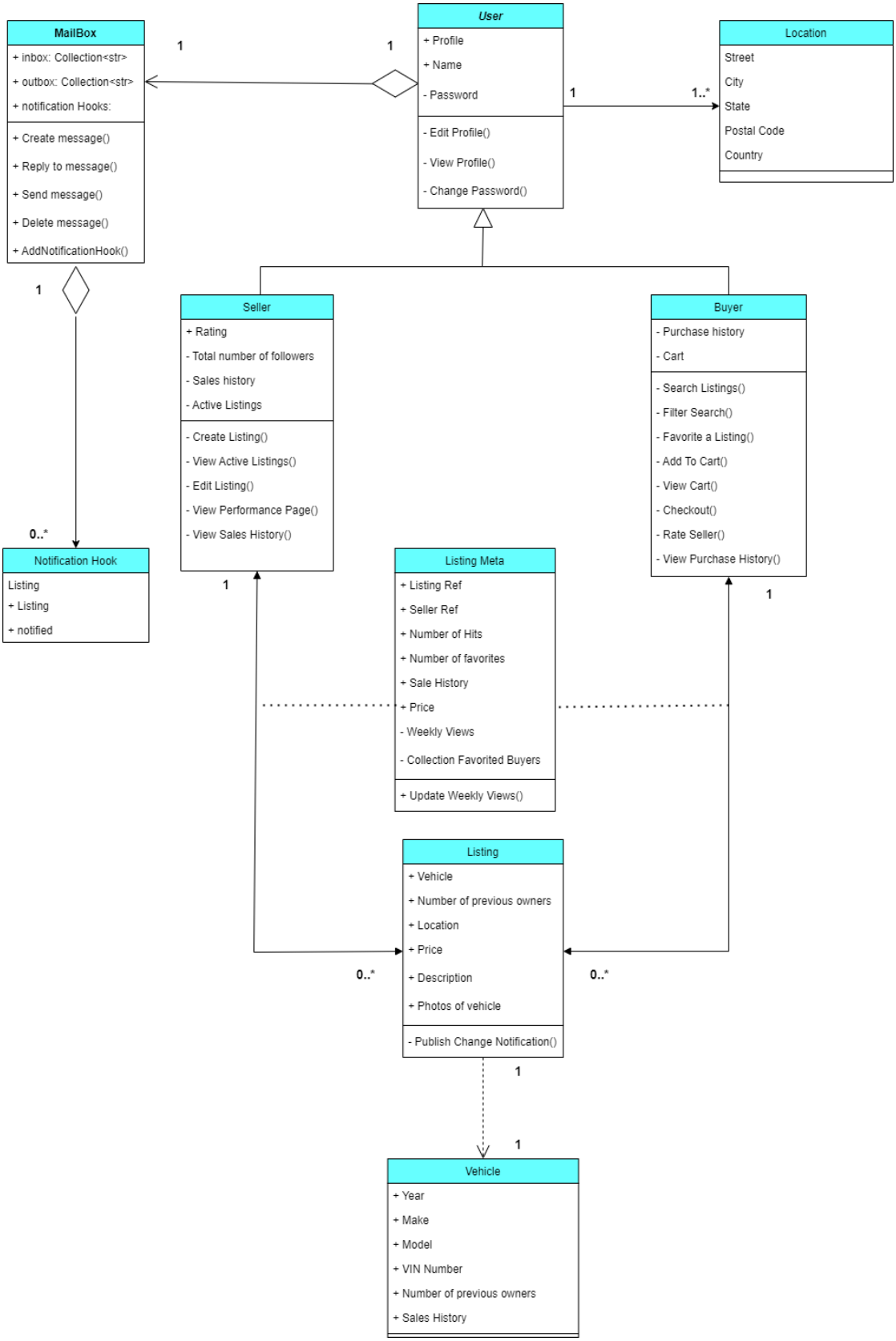
Seller Use-Case Diagram (Team)



Buyer Use-Case Diagram (Team)



System Class Diagram (Team)



Use-Case Detailed Descriptions with Corresponding Sequence & Class Diagrams

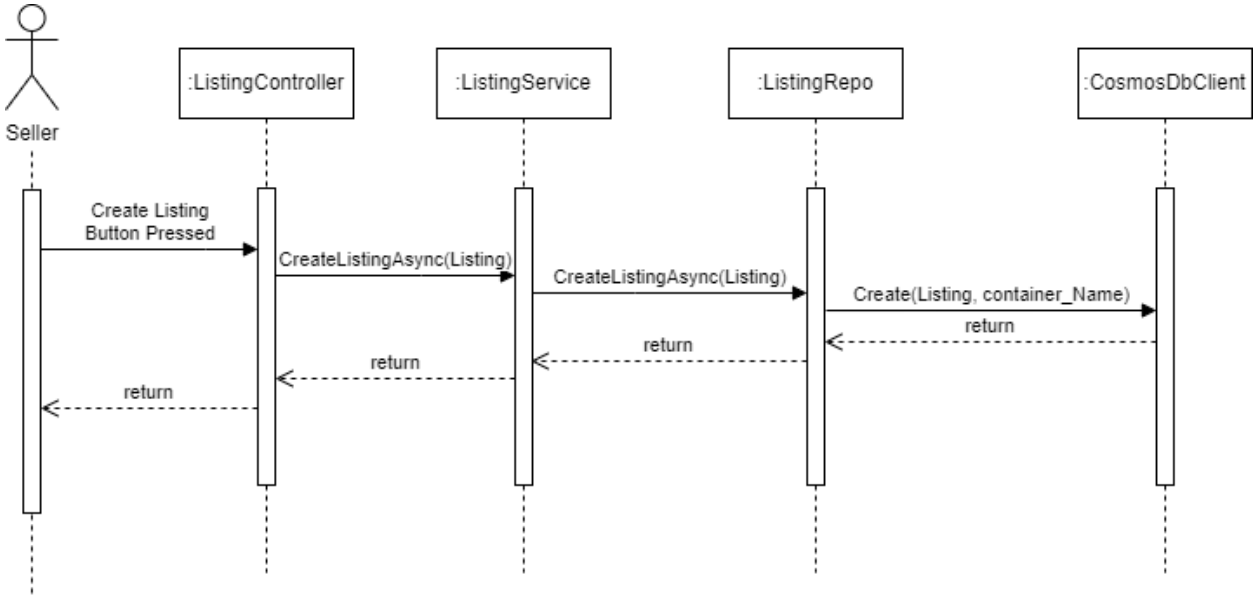
Use-Case Detailed Description: Create a Listing (Anthony Redamonti)

Team Member:	Anthony Redamonti
Use Case:	Create a Listing
Actors:	Seller
Goal:	To create a listing of a vehicle in the Junk-Broker system
Overview:	
The seller will create a listing by pressing the "Create a Listing" tab in the seller's main page. The tab will open a menu of unpopulated fields that the seller must populate in order to create a listing. After entering all the required information in the fields, the seller will press the Create Listing button. After pressing the button, the listing will be posted in the Junk-Broker database, available for buyers to view and purchase.	
Cross-reference:	
R1, R6	

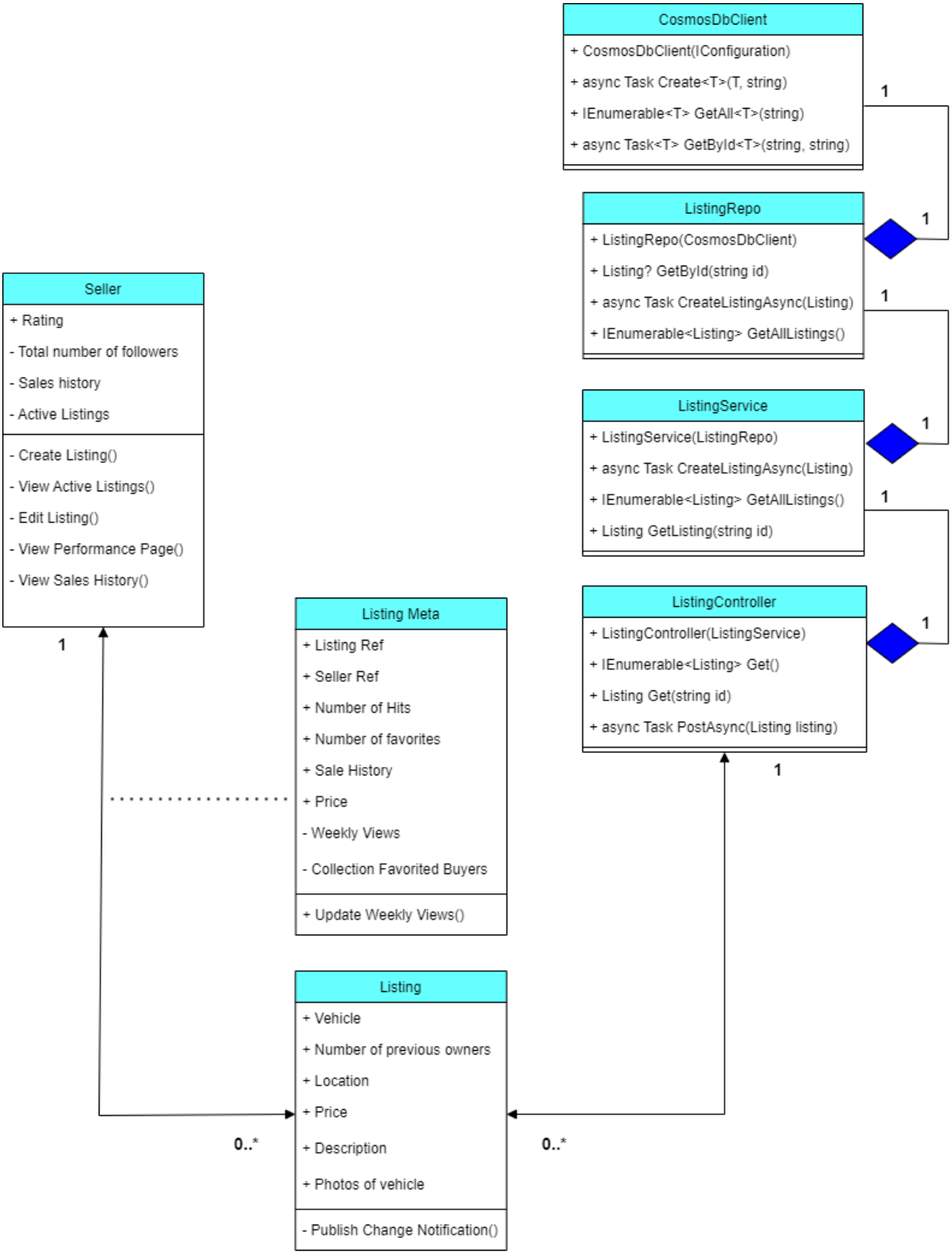
Typical Course of Events:	
Actor Action	**System Response**
1. The seller populates all required fields under the Create Listing tab.	
2. The seller presses the create listing button.	
	3. A listing object is created by the listing controller.
	Steps 4-6 occur in the form of asynchronous events.
	4. The listing controller passes the listing object to the listing service.
	5. The listing service passes the listing object to the listing repository.
	6. The listing repository passes the listing object to the CosmosDbClient database.
7. The listing appears in the seller's active listings page and is available for buyers to view and purchase.	

Alternative Courses:	
Steps 1-3	The user does not populate all the required fields under the Create Listing tab and presses the Create Listing button. The listing controller informs the user via error message that all the required fields have not been populated to create the listing.

Sequence Diagram: Create a Listing (Anthony Redamonti)



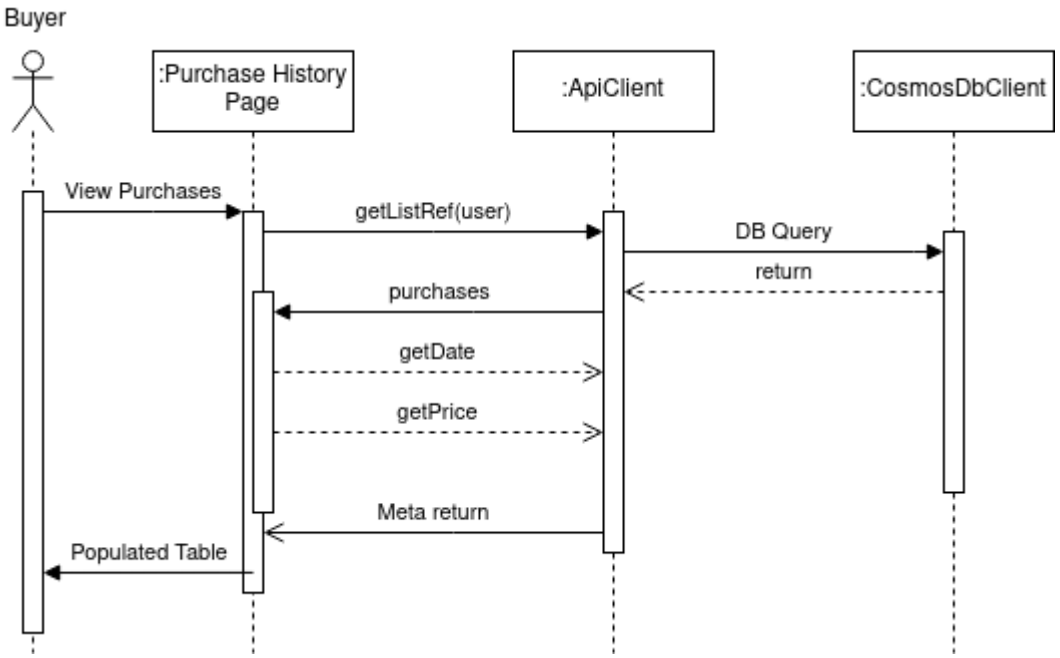
Class Diagram: Create a Listing (Anthony Redamonti)



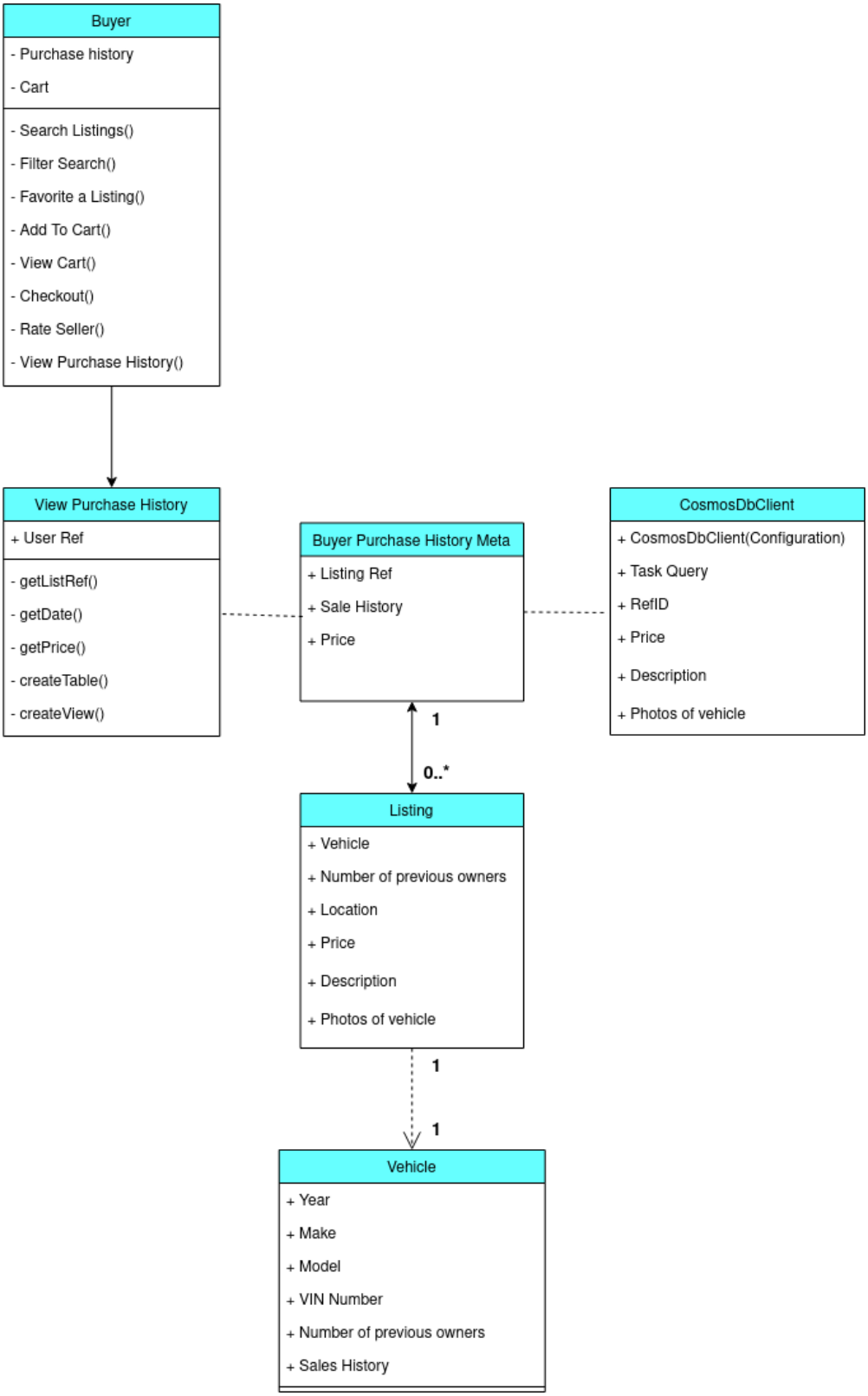
Use-Case Detailed Description: View Purchase History (Michael Peña)

Team Member:	Michael Peña	
Use Case:	View Purchase History	
Actors:	Buyer	
Goal:	Allow the buyer to view their purchase history	
Overview:		
When a Buyer wants to view their purchase history, they click the "View Purchase History" button from the application's navigation area (left side of the application). The contents of the main page container shifts to display a table listing the Buyer's purchase history. The table is rendered with information from the database, to include date purchased, the listing reference, and price. The Buyer can scroll through the history of their purchases and drill down through each purchase to see the full listing details.		
Cross-reference:		
R2, R12		
Typical Course of Events:		
Actor Action		**System Response**
1. Buyer selects "View Purchase History" from the left-side navigation area of the application.		
		2. The application main window view shifts to the "View Purchase History" page.
		3. The system pulls the Buyer's historical transaction data from the database.
		4. The system displays the data in a scrolling table format.
5. The Buyer scrolls through their purchases.		
Alternative Courses:		
Before Step 1	If the user is in "Seller" mode, they will have to switch to the "Buyer" mode and then proceed along the steps.	

Sequence Diagram: View Purchase History (Michael Peña)



Class Diagram: View Purchase History (Michael Peña)



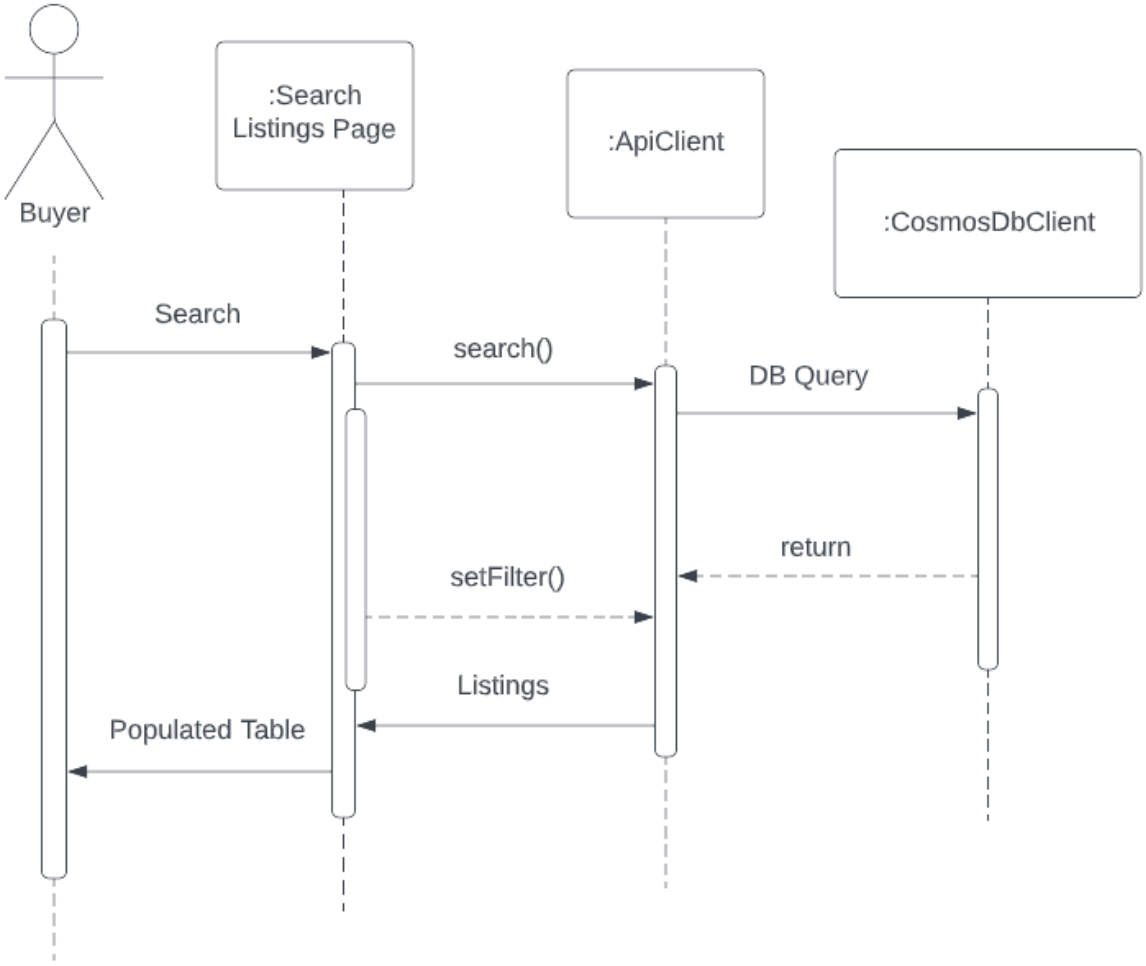
Use-Case Detailed Description: Search Listings (Clayton Colson)

Team Member:	Clayton Colson
Use Case:	Search Listings
Actors:	Buyer
Goal:	Allow the buyer to search and return specific listings
Overview:	
When a buyer would like to search for a specific listing that matches their personal requirements, they click the "Search" from the applications navigation area. The contents of the page present an editable text input that queries the listings to return the results matching keywords entered by the buyer. The results are returned from the database that match the search and filter conditions set by the buyer. The buyer can scroll through the list of eligible results matching the criteria and select one listing to display information relating to that listing.	
Cross-reference:	
R3, R13	

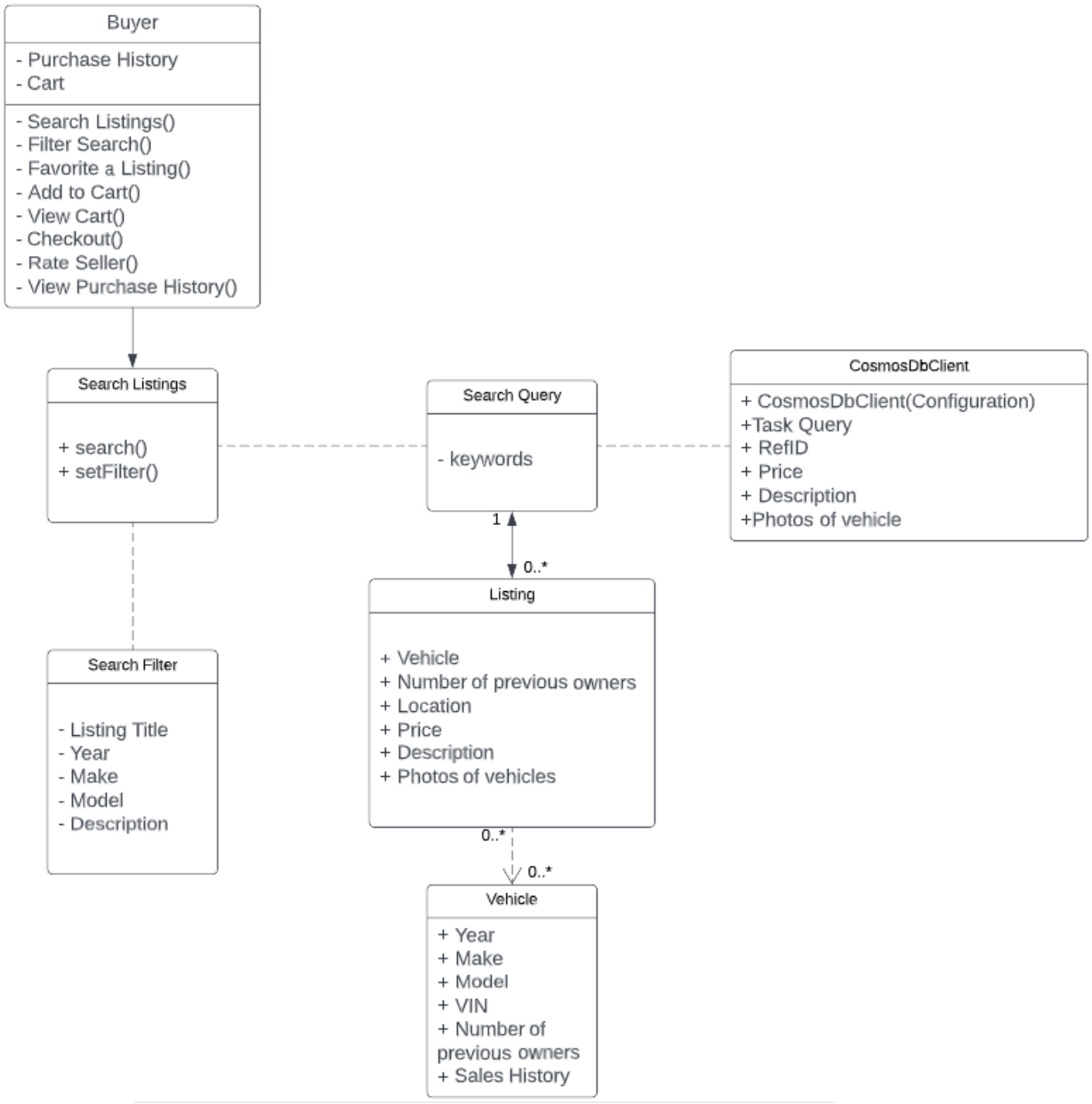
Typical Course of Events:	
Actor Action	**System Response**
1. Buyer selects "Search" from the menu navigation drawer.	
	2. The display changes to the search page and presents search criteria.
3. Buyer enters search criteria into the search bar.	
	4. The system returns eligible listings in the database that match search criteria.
5. The Buyer scrolls through a list of matching search results.	

Alternative Courses:	
Before Step 4	If no listings in the database match the search criteria, the system will return a string: "No listings found."

Sequence Diagram: Search Listings (Clayton Colson)



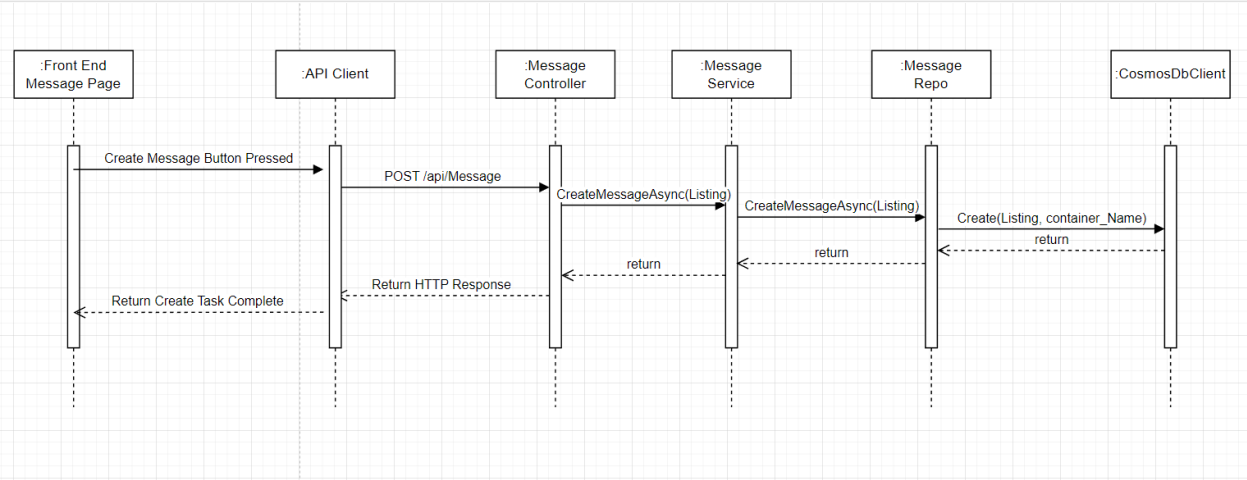
Class Diagram: Search Listings (Clayton Colson)



Use-Case Detailed Description: Message Other Users (Kenneth Chalupa)

Team Member:	Kenny Chalupa	
Use Case:	Message Other Users	
Actors:	Receiver, Sender	
Goal:	Allow users to send each other messages	
Overview:		
When a user wants to send a message to another user, they can utilize the UI to send a message to them. The sender of the original message can also be replied to by the receiver from the UI, prompting the receiver to send a response message.		
Cross-reference:		
R8		
Typical Course of Events:		
Actor Action		**System Response**
1. Sender sends a message through the UI.		
		2. API receives a new message HTTP Post Request.
3. Receiver is notified through UI of new message.		
		4. API utilized database connection to create the new message for persistence.
		5. Database persists new message.
Alternative Courses:		

Sequence Diagram: Message Other Users (Kenneth Chalupa)



Class Diagram: Message Other Users (Kenneth Chalupa)