Software Design Documentation

Software Development

Final Project - Baggage Recovery App

Wei Liu
Ziwei Dai
Lixin Chen

TABLE OF CONTENTS

1. System Architecture

The Baggage Manager Application is a three-tier web application which applies the Model-View-Controller (MVC). MVC is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller.

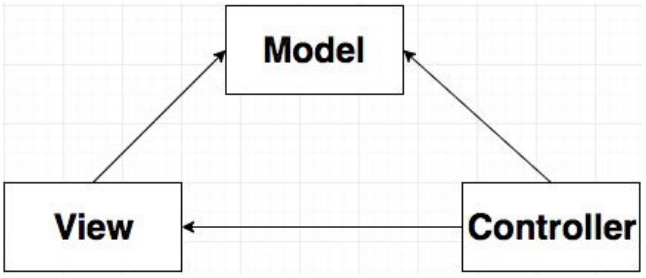

Figure 1: MVC diagram

However, the development framework used by our team to develop this application is Django which is a python framework. Therefore, it is necessary to clarify the correspondence between the components in Django and the MVC structure. The following diagram shows the correspondence.

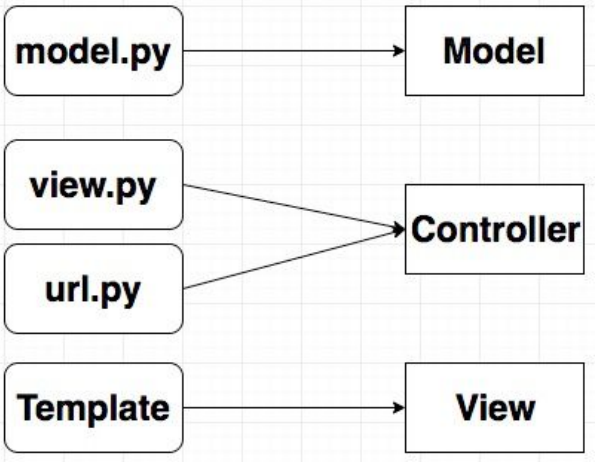

Figure 2

model.py defines all the python class which can be mapped to the database through the Django ORM mechanism. url.py defines which method handle which URL and view.py are where we actually implement those methods. The template is a folder for all HTML file, therefore, it is a view layer for front-end presentation.

2. Component diagram

This application is developed in Python using a Django server to support the image matching functionality as it uses the OpenCV Python library. The data itself will be stored in a PostgreSQL database and the application will be rendered via the user's browser (for the prototype of the application, not the eventual mobile application). A diagram of these components is shown in more detail below in Figure 3.

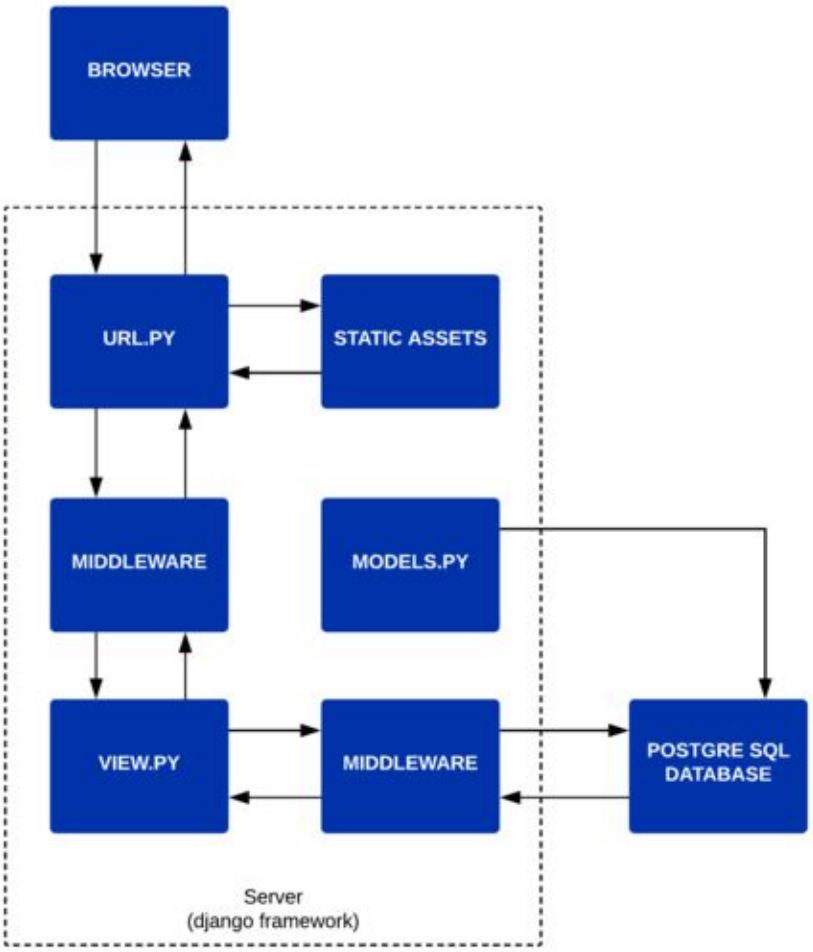

Figure 3: Component Diagram of the Baggage Recovery Application

3. Data model diagram

The overall design of the database is illustrated by the Entity-Relation Diagram is shown below.

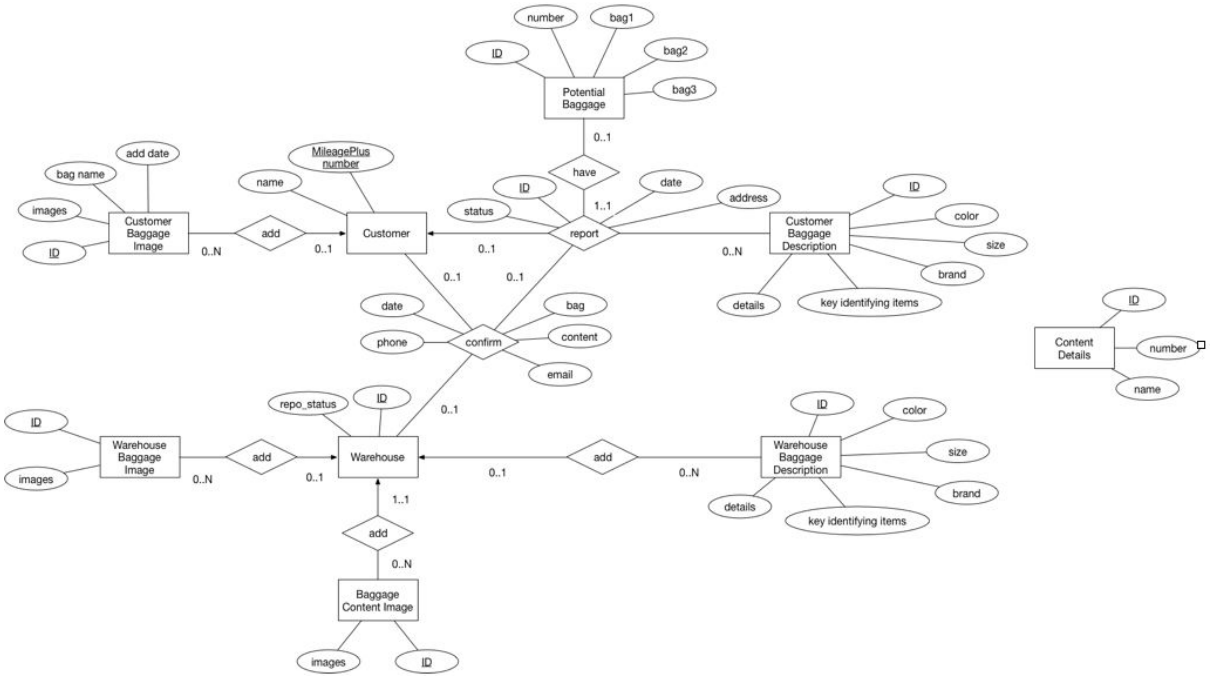

Figure 4: Entity-Relation Diagram for Baggage Recovery Application Database

There are five entities in the database model: customer, warehouse, baggage image, baggage description, and baggage content image. Each of these entities has several attributes where the underlined attributes are the corresponding primary key for that entity.

The detailed relational model for the database illustrated in Figure 4 is shown in Figure 5 below. There are six relations (tables) for the database that come from the five original entities plus the “report” relationship which must also be stored in a table. Each table has a primary key and a set of attributes; some tables also have a foreign key. Note that not all of the attributes listed are the actual database attributes that will be stored. For example, the “key_identifying_item” attribute is actually a collection of attributes that will be stored to represent the key identifying content in a bag.

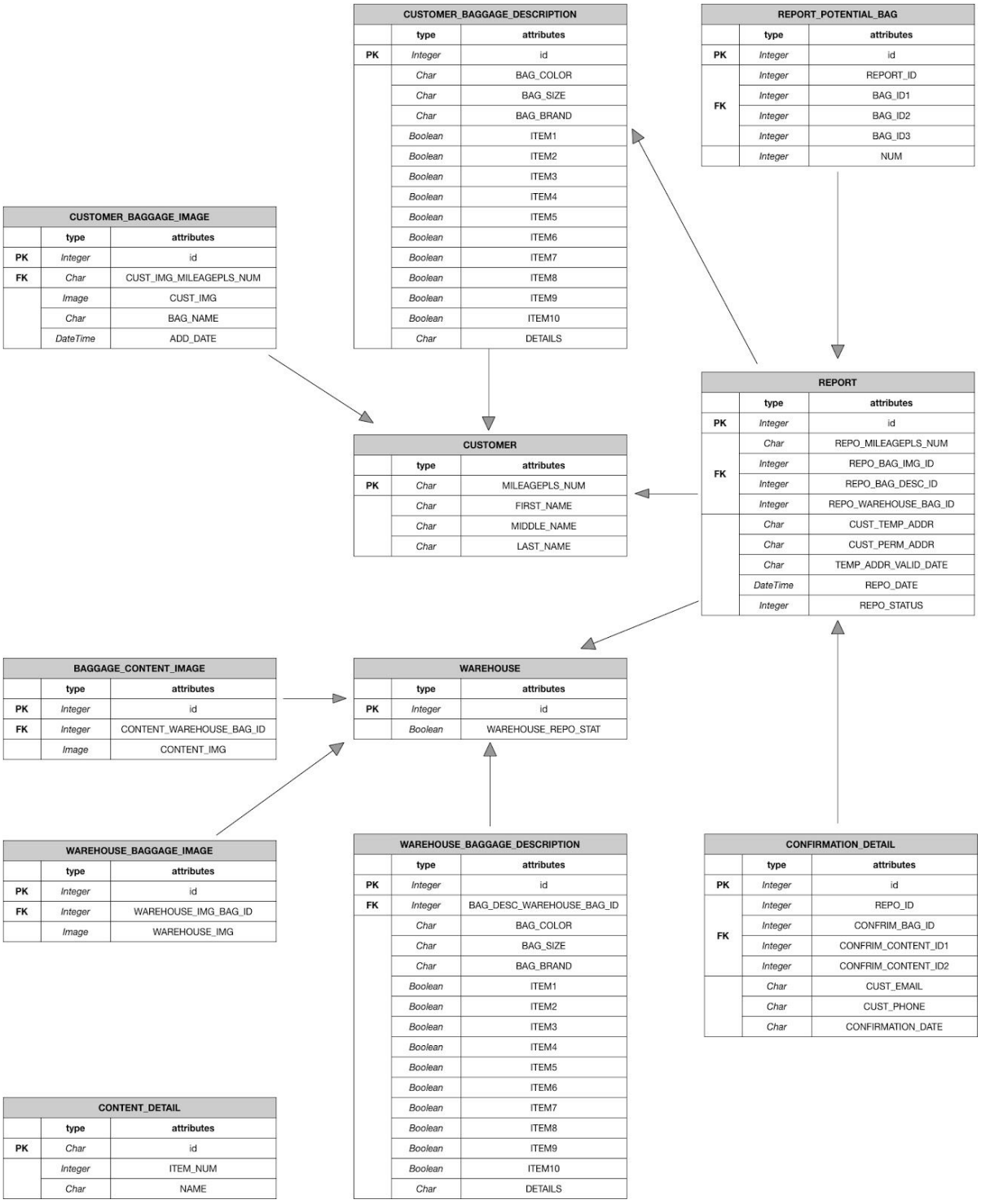

Figure 5: the Detailed relational model for Baggage Recovery Application Database

The cardinalities of the associations are as shown below:

- customer to baggage description (1: N)
- customer to baggage image (1: N)
- customer to report (1: N)
- report to baggage description (1:1)
- warehouse to baggage description (1: N)

- warehouse to baggage image (1: N)
- warehouse to baggage content image (1: N)

4. Use Case Diagram and Class Diagram

4.1 Use Case Diagrams

In order to get a better understanding of the design of the user interface, the team developed several use cases that encompassed the major functionalities of the application. These use cases are detailed below and summarized in Figures 4 and 5.

Use Case 1: New user manages bags
Jane is a frequent flyer who must travel often for business. Since Jane flies frequently, she wants to make use of their new Baggage Recovery Application. When she arrives at the airport to check in, she first navigates to the Baggage Recovery Application and signs in with her MileagePlus number. From the main page, she notices the "Manage My Baggage" menu option. When she clicks on this option, she is directed to a page containing her "Saved Baggage" which is currently empty since this is her first time using the application. She clicks to "Add New Bag" and takes a picture of the luggage she frequently checks in when flying with United. She names this bag "Large Red Bag" and sees that this image was added to her "Saved Baggage" page. She is now ready to check in her bag and continue to her flight.

Use Case 2: Returning user manages bags
A few months later, Jane is flying on another business trip with United Airlines but this time, she is using her brand new bag because her old bag broke after she returned from her last trip. Before checking in, Jane navigates to the United Airlines Baggage Recovery Application and selects "Manage My Baggage". She is brought to the "Saved Baggage" page where she sees the entry for "Large Red Bag". First, she clicks "Add New Bag" and uploads an image of her new bag, "Large Purple Bag". When she sees that this new bag was added, she scrolls to "Large Red Bag" and clicks "Delete" since she no longer has this bag to travel with. Jane is now ready to check in her purple bag and continue to her flight.

Use Case 3: Customer reports a missing bag
Jane has just landed at JFK for her business trip but is unable to find her bag at the baggage claim. She opens up the Baggage Recovery Application and clicks on the "Report Missing Baggage" option on the home page. Here, she is prompted to enter her reservation number and select the image of the bag she wants to report as lost. She is further asked to enter details about the external appearance of the bag as well as some details about the contents within the bag. Lastly, she fills in her contact information and submits the missing baggage report.

Use Case 4: Customer checks bag status
A few days after she reported her bag missing, Jane receives a notification that her bag has been found. She goes into the Baggage Recovery Application and clicks the "Check Baggage Status" option and enters her reservation number. Since her bag has been found, the next step for Jane is to confirm ownership of her bag. She is presented with a few pictures of potential bags and selects the bag that is hers. She is then presented with several images of items and must select the images of the items that were in her bag. She picks the images of her belongings and enters a phone number and preferred time to be contacted. At this time, a United Airlines employee will contact Jane for a final confirmation and send the bag back to Jane at the address she specified when she filed the report

Use Case 5: Employee resisters new bag
Michael is an employee at the Baggage Resolution Service Center who works on cataloging new bags as they come into the warehouse. When Michael goes to process a bag, he logs in to the Baggage Recovery Application Warehouse Assistant using his employee ID. Once he is logged in, he clicks the "Register New Baggage" option. Here, Michael fills in information about the bag's external appearance and uploads an image of the bag he is processing. Next, he opens the bag and analyses its contents, taking note of any uniquely identifiable items in this bag. He picks two of these items and takes pictures of them through the application. Once he is done registering this bag, he is provided with a list of baggage reports that are potential matches to this bag. He looks at each of these reports and determines whether the bag he just processed matches one of these reports. He finds the matching report and selects "Confirm Match" for this bag. On a later bag that the processes, there are no matching reports for this bag and he selects "Wrong Match" for this bag.

Use Case 6: Employee processes missing bag report
Rafael is another employee at the Baggage Resolution Service Center who works on processing the reports of missing bags after the bags have been registered. Rafael logs in to the Baggage Recovery Application Warehouse assistant and selects the "Process Missing Baggage Report" option from the menu. He scrolls through the unresolved reports, sees a report that has potential matches, and clicks "view". Rafael then looks through the possible matches that were generated for this report. When he finds the bag that matches the information in this report, he selects "the bag has been found". He goes back to the unresolved reports and selects a report that is waiting for a phone call. He clicks "view" and sees that the customer correctly identified themselves as the owner to this bag. Rafael then calls this customer to confirm their address and locates the bag to be sent back to the customer.

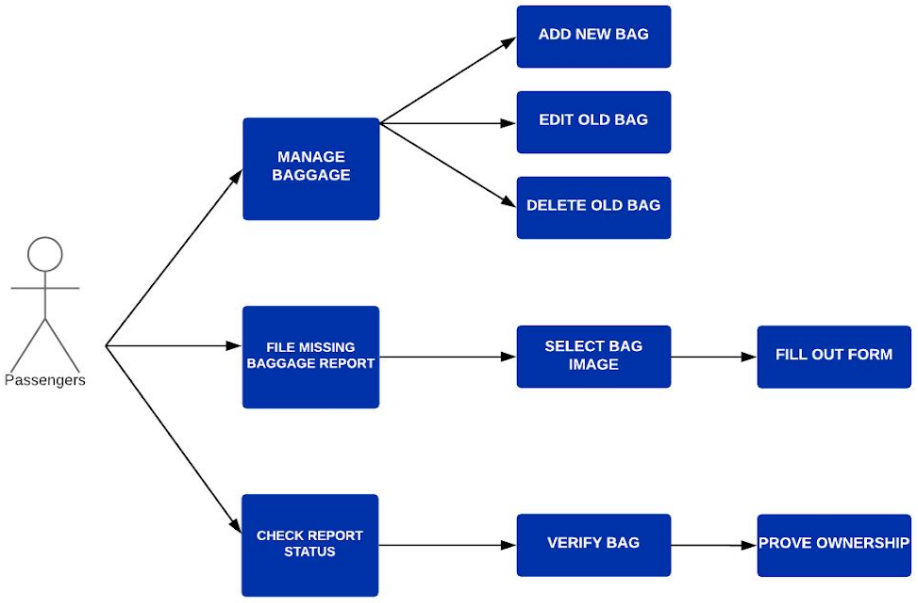

Figure 6: General use case diagram for a passenger

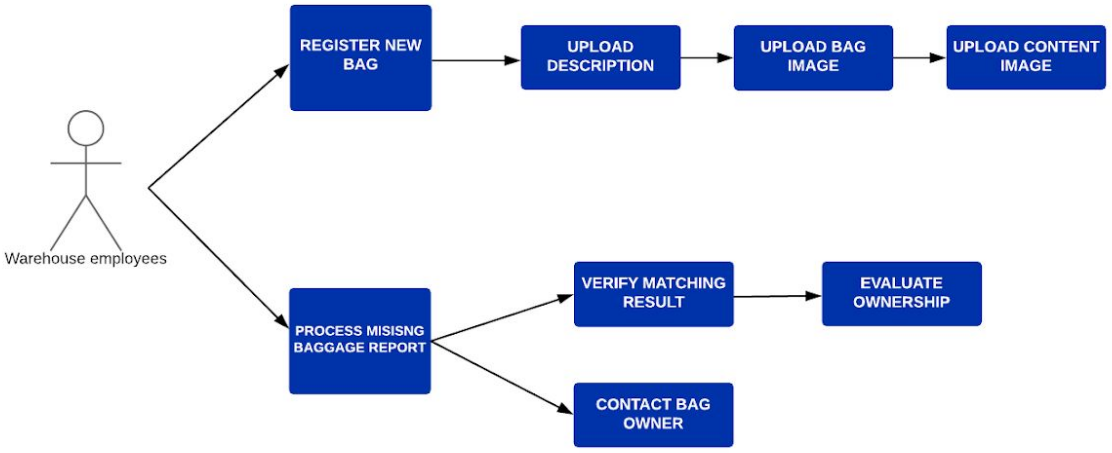

Figure 7: General use case diagram for a warehouse employee

4.2 Class Diagram

The following figure is the class diagram derived from the use case diagram.

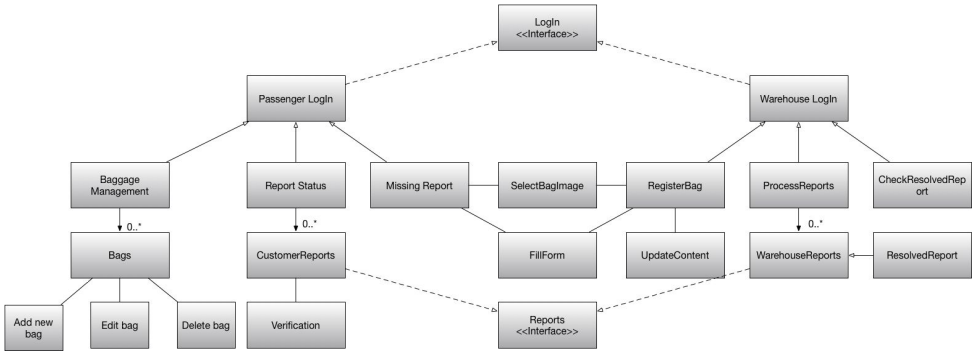

Figure 8: Class diagram

5. Sequential diagram

In order to facilitate these processes for the customers and employees, the system had several requirements to ensure seamless execution of the application. These requirements are related closely to the desired workflow for the application which is depicted in the sequence diagram in Figure 9 below.

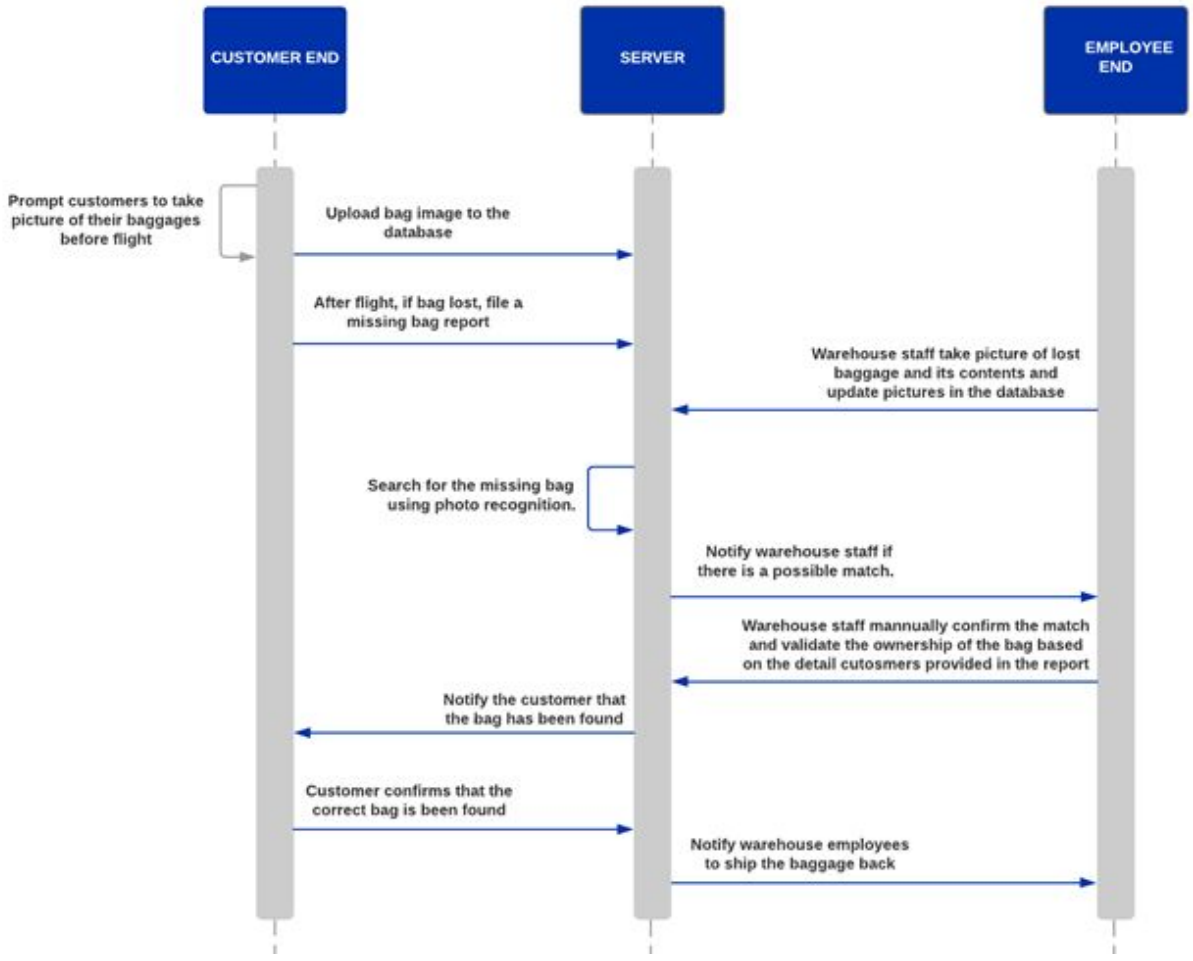

Figure 9: Sequence Diagram between the customer end, server, and warehouse end

One major requirement for the system was that it supports two different kinds of matchings: a bag to report and report to the bag. This is because some bags might arrive at the warehouse before a report is filed and some reports might be filed before their bag arrives at the warehouse. In the first case, when an employee registers a bag at the warehouse, the system looks to see if there are any currently open reports that match with the bag. If this report exists, then the bag is matched and the process is complete. If there are no matches, then the system leaves this bag as unmatched. In the second case, whenever a new bug report comes in, the system tries to match this report with any existing bags. If the bag isn't currently at the warehouse, then the

match is found when the bag arrives at the warehouse via the bag to report matching. When trying to find matches for both of these situations, the system first uses the textual descriptions uploaded by the customer and warehouse employees to narrow down the range of potential target bags. Only then does it use the image matching technology to match the pictures together by selecting the top three images with the greatest similarity.

Another major requirement had to do with confirmation of ownership. The system has to protect against fraudulent actors that intentionally try to get bags that are not theirs, and so the warehouse staff needed a way to confirm that they are returning a bag to its true owner. This is done by sending the customer several images of items from bags at the warehouse. The customer must then pick which 2 of these items were actually in their bag. This captcha-like system creates an added layer of accountability which enhances the success rate of the application. When the employee is sure of the owner, they contact the customer and solidify the shipping arrangements.

Other more minor requirements were established throughout the development process, such as the option for an employee to close a report manually if the bag was found before it reached the warehouse.

6. Deployment Diagram

While not a part of the scope of this project, the eventual goal for this application is that the client-side will be integrated into the existing United Airlines mobile application and the warehouse-side will be deployed as a separate application. This deployment structure is illustrated in Figure 10 below.

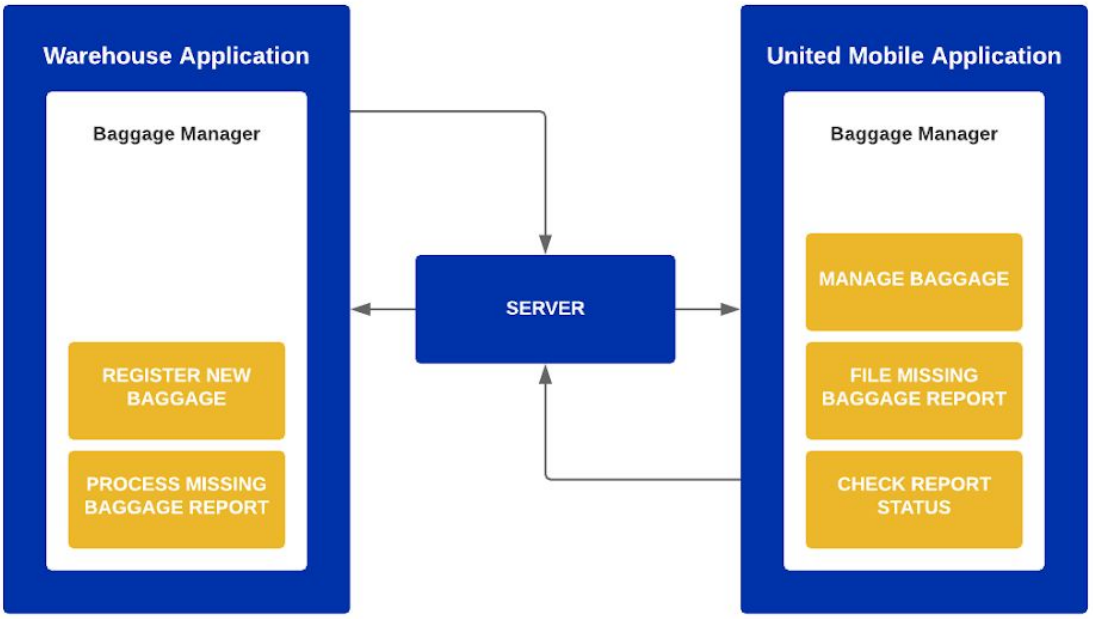

Figure 10: Deployment diagram for the United Airlines Baggage Recovery Application

7. Activity Diagram

The activity diagram in Figure 11 below depicts the activities done by the server and the processes it takes based on the output of certain activities. For example, the server will receive a missing bag report and bag registration information and then try to find image matches between existing reports and bags. Depending on the results of the image match, the process can either continue to the next step of notifying the customer and employee or the report and bag are marked as unmatched and will be tried again later. If the match is successful, the employee sends a confirmation request to the customer and the server will determine if the customer has correctly verified their ownership. If it has, then the employee can return the bag to the customer otherwise the employee will continue with a manual process to contact the customer.

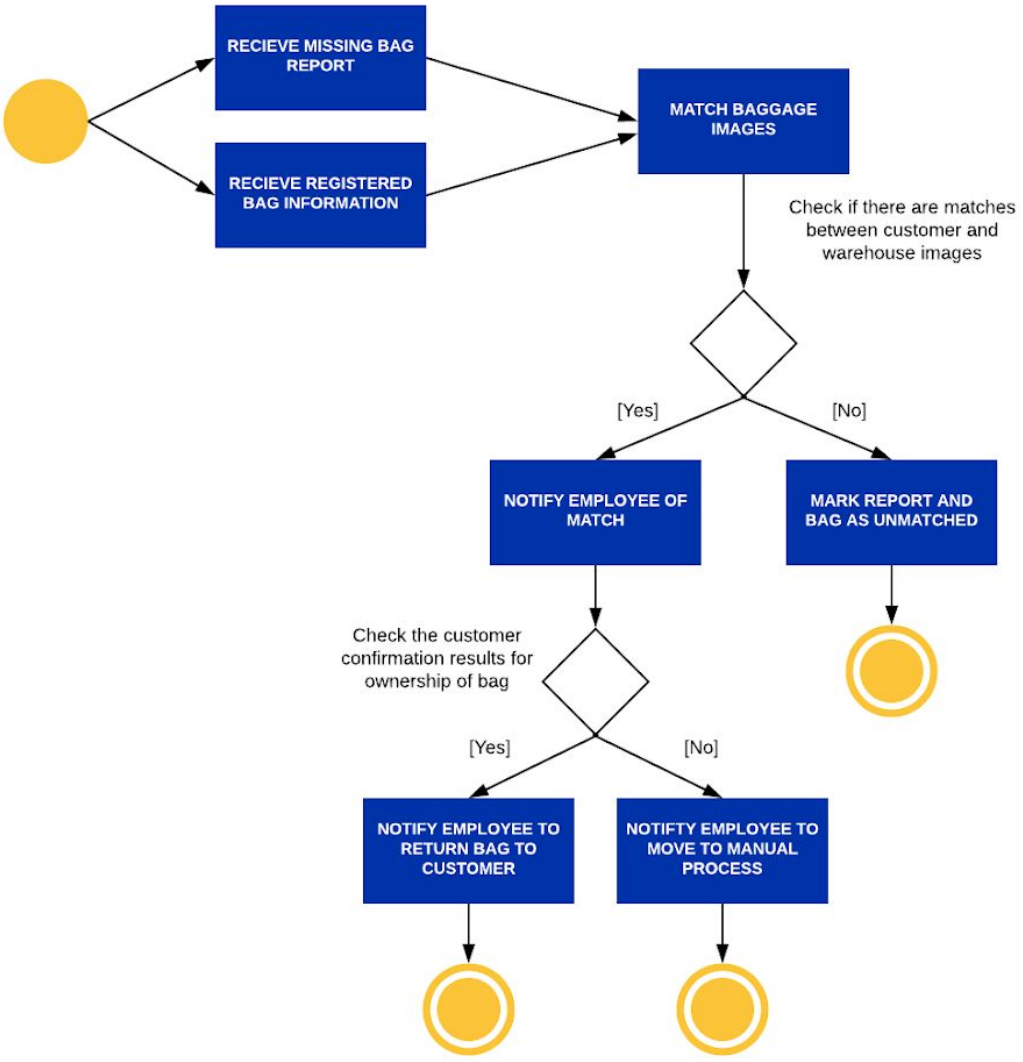

Figure 11: Activity Diagram for the United Airlines baggage recovery application