System Configuration Documentation

Software Development

Final Project - Baggage Recovery App

Wei Liu
Ziwei Dai
Lixin Chen

TABLE OF CONTENTS

Introduction

This document contains the installation instructions for the Baggage Recovery Application developed by the Rensselaer Polytechnic Institute Software Development group. These instructions go through the installation of the required software components as well as how to set up the local server to run the application. There is also some additional information about what certain files are required for and the page structure of the application. As a reference, Figure 1 contains the general use case diagram for the customer and warehouse components of the application. This diagram covers all of the major actions that each kind of user can execute through the application.

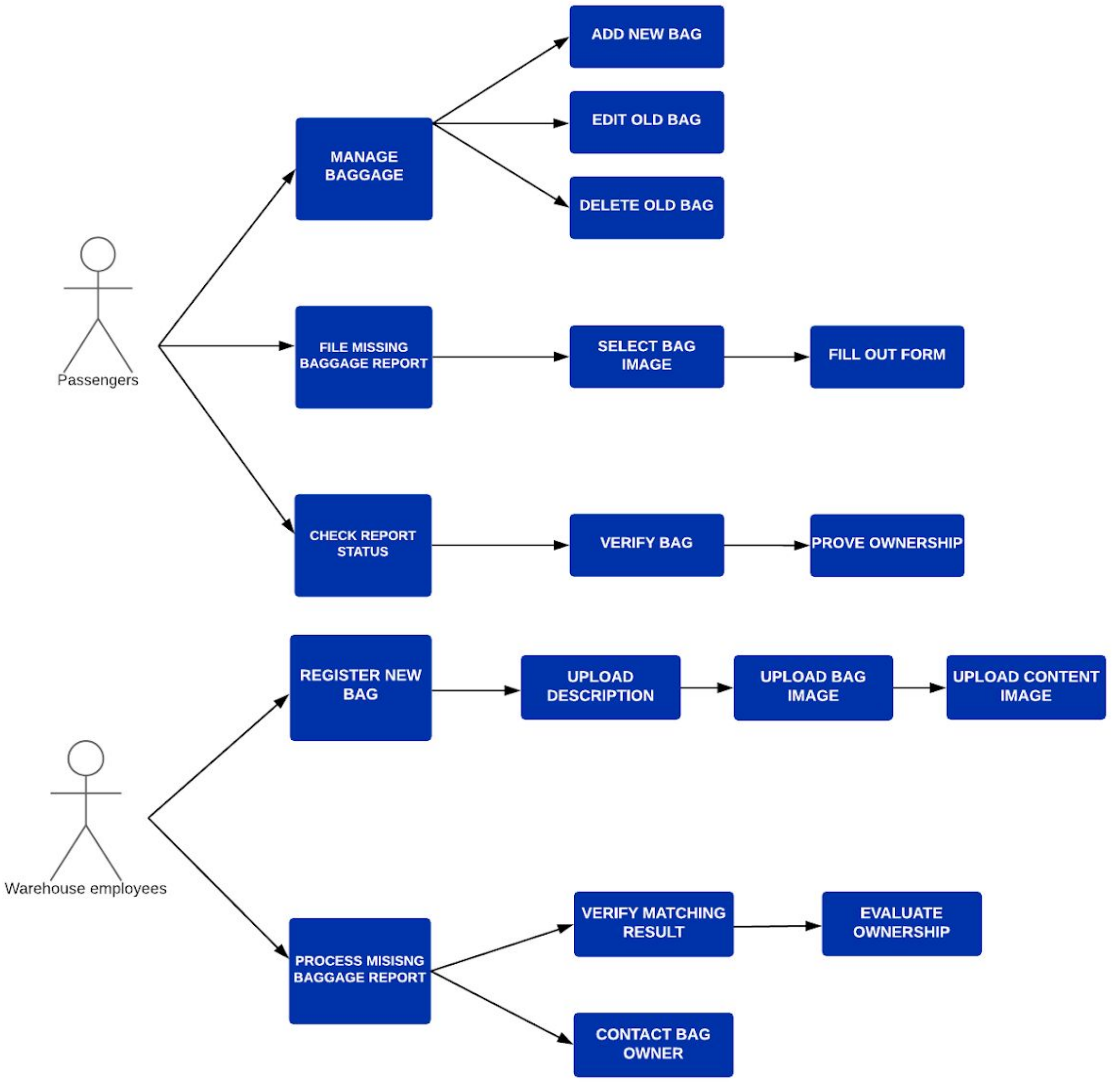

Figure 1: General use case diagram for a passenger (top) and warehouse employees (bottom)

1. Hardware Information and Deployment environment

- Linux / MacOS operating system (local test)
- Amazon AWS Instance (Online deployment)
- PostgresSQL (later than 10.0.0)

2. Software Information

There are several pieces of software that are necessary in order to successfully run the Baggage Recovery application. These software and their installation instructions are details below.

2.1 Git (Version control)

Git is an open source distributed version control system that was used by the development team to maintain the source code. In order to download, run, and edit the source code, Git must be installed. The installation instructions for Git can be found here.

2.2 Python3 (including package: Pillow, imageHash, openCV, numpy)

This project was built using the Python programming language and so Python is required for the application to run successfully. To download Python, follow the instructions in this installation guide. Any stable release after Python 3.0 will be sufficient for the application. The development team used Python 3.6.

1) Open Command line on the system:

   For Windows:

   - Click the **Start** menu, and then click **Run**.
   - In the Run dialog box, type **cmd**.
   - The Command Prompt window opens.

   For Mac OS:

   - Click the **Spotlight** icon.
   - Type **terminal** into the search box.
   - Double-click **Terminal.**

For Linux:

- Click on **Applications** in the main menu on the panel.
- Click on **System Tools**.
- Click on **Terminal**.

2) In the command window, type the instructions below one at a time (hit enter between each command):

```
pip3 install Pillow
pip3 install imageHash
pip3 install OpenCV-Python
pip3 install numpy
```

2.3 Django2

Django, a Python Framework, is used as the server for this application. Django can be downloaded here.

2.4 PostgreSQL (Version 10.0)

The database used to store information for the application is a PostgreSQL database. Click here to visit the download page for PostgreSQL. Download and Run the **setup.exe** file and follow the instructions to install PostgreSQL on your computer. We also recommend installing **pgAdmin 4** as a development platform for PostgreSQL. This installation can be found here.

3. Application Import and Settings

In order to set up the environment for the application, follow the below steps.

Step 1: Cloning the GitHub repository

Use the following command to clone the Github repository into your local system:

```
git clone https://github.com/baggage-recovery-app/United_Baggage_Manager
```

Step 2: Setting Up the Database

- Open **pgAdmin 4** application and connect to your **PostgreSQL** DBMS.
- Create a new database named "United_Baggage_Manager" (Database name should be same as Django project/application name).
- In any text editor or IDE, navigate to **settings.py** file in the **United_Baggage_Manager** folder in the **United_Baggage_Manager** project.
- Find the below code snippet and update the information with your PostgreSQL database username and password:

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.postgresql',
      'NAME': 'United_Baggage_Manager',
      'USER': 'postgres',
      'PASSWORD': your password,
      'HOST': 'localhost',
      'PORT': '5432',
   }
}
```

Step 3: Making migrations (Synchronizing Application Code and Database)

After making changes in the **settings.py** file in Step 2, we need to synchronize the Python code and database. To do this

- Navigate into the **United_Baggage_Manager** directory (project folder) through the command line.
- Execute the following two commands:

```
python3 manage.py makemigrations
python3 manage.py migrate
```

Step 4: Importing Data

To import the data to the application, execute the instruction below in the command prompt after navigating inside the **United_Baggage_Manager** project directory/folder:

```
python3 import_data.py
```

Step 5: Running Server/Application

Once, all the settings are completed, the code is updated, and the data is imported, you can run the server to start the application. To run the server and start the application in your default browser, run following instruction in the command prompt after navigating inside the **United_Baggage_Manager** project directory/folder:

```
python3 manage.py runserver
```

4. Code Structure

Once the server is running, navigating to http://127.0.0.1:8000/Baggage_Recovery_App/ loads the customer end of the application. Navigating to http://127.0.0.1:8000/Baggage_Recovery_App/warehouse.html loads the warehouse end of the application. The following section describes the overall structure of the application, as well as certain relevant files.

4.1 Application Structure

The project has two main subfolders: Baggage_Recovery_App and United_Baggage_Manager.

The Baggage_Recovery_App is the folder for the Python application that holds this project. This application folder contains the static assets for the application in a subfolder called static/Baggage_Recovery_App. These static assets consist of the CSS and Javascript files for the application as well as any image assets that were used in the user interface. Another subfolder is templates/Baggage_Recovery_App which contains the HTML files for the application. The Baggage_Recovery_App folder also contains the Python file for the image recognition algorithm, as well as other supporting Python files.

The United_Baggage_Manager subfolder contains the main settings for the application. This is where **settings.py** is located, which is the file contains the settings for the Django server and PostgreSQL database.

Another subfolder in the main application is image-import-data. This folder contains the team's testing data, including customer images, warehouse images, and content images. This test data was uploaded as a part of the project just so that the team had the same data to test with. Any images that are uploaded through the application are stored somewhere else, not in this folder. Other testing data is located in **report_data.txt** and **warehouse_data.txt**.

4.2 Login Credentials

When the customer application is loaded, there is a login page that requires the customer's mileage plus number. We have provided multiple sample mileage plus logins in the file **customer_data.txt** that correspond to multiple different accounts. For example, if you log in with the number "123" this corresponds to the account for one of our team members. These credentials can be changed in this file which is located in the

main United_Baggage_Manager folder. Currently, this file only contains the mileage plus number and the customer name, but the intention is that developers could connect United's mileage plus information to the application and pull that data in the future. There currently are no login credentials for the warehouse interface.

4.3 Content Items

In both the customer and warehouse sides of the application, the user is asked to identify whether their bag contains one of ten possible categories of items. These items are currently the following:

- Books
- Currency
- Optics
- ID
- Tobacco
- Footwear
- Electronics
- Dresses
- Shirts
- Medication

We selected the following as they are a combination of common and less common items found in bags. However, if these are not sufficient or need to be changed over time, they can be updated in the **content_data.txt** file. Any changes made will require the data to be re-imported into the application using the import_data.py command mentioned in section 2.3 step 4.

4.4 Image Matching

The image matching algorithm used for the project is located in the **matching.py** file in the Baggage_Recovery_App folder. This algorithm works by extracting the bag itself from the background and then comparing the bags themselves without the backgrounds. An example of this is shown in Figure 2 below. GrabCut is a function of OpenCV that was used for extracting the bag from the background. It will identify the target baggage from the background according to the color change in the boundary. Even though the algorithm is able to extract the bag from the background, the less background there is in the image the better the matching will be. Since this is the case, when a user is uploading a bag there is a message and sample picture explaining to them how to take the picture in a way that works best for the algorithm. Perceptual Hashing is a function from the ImageHash Python library which provides the algorithm used for image similarity. This algorithm generates values between two images where the more similar those two images are, the lower value will be. We rank these values to determine the top three most similar images. The algorithm also takes into account the

textual descriptions of the bag and its contents. This serves as a way to narrow down the results so that fewer images need to be compared against each other.

Figure 2: Customer image (top) and warehouse image (bottom) before and after the bag is extracted from the background of the image.