Final Report : CSCE 606 Kayak for Masks Project

Team Name: CSCEProject

Team Member	Roles
Jingtun Zhang	• Team Leader • Developer
Keigo Ma	• Developer • Feature Tester
Yingying Wang	• Developer
Yuncheng (Max) Yu	• Scrum Master • Developer
Siqi Fan	• Developer
Keqiang Yan	• Developer
Hanzhou Liu	• Developer

Client: Tito Choudhary

Github Link: https://github.com/OrdinaryCrazy/Kayak_for_Masks

Heroku Production Links: http://masklink.herokuapp.com/masklink/

Youtube Video and Poster Demo Links:

https://www.youtube.com/watch?v=R0NPQBpocII&t=54s

Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2535902

Brief Summary of the Project:

In the post-pandemic period, although around 60% of the American population have been fully vaccinated, it's still indispensable to wear masks in the public area to prevent any known or unknown virus variants (Mathieu et al.). This is especially critical for kids because of much lower children vaccination rates compared to adults. Many parents report that they struggle to buy high-quality, reliable, and affordable face masks or respirators for their children. There is no doubt that an application which is able to collect real-time mask information including manufactures, prices, sizes, availability, and purchase links can help those anxious parents a lot. That's the target of the *Kayak-for-Mask* website application.

First, the customer requests that the user should be able to sort masks in order of the selected attribute. The application provides three sorting options, named as *size*, *filtration efficiency* and *names*. The specification of these filtering designs is self-explanatory, which is implemented via python numpy sorting functions. Second, the customer requests that the user should be able to filter masks based on their choices. Therefore, apart from single choice fields, the Django multiple choice fields must be added in the form file to support multiple selections. Besides, the python numpy drop functions and a tricky algorithm are used to enforce filterings. Third, the customer requests that the price data should be updated in real time. This is done by embedding a web crawler into the project codes. After running the application, the latest price data from requester manufactures will be loaded into the numpy dataframe. In sum, the final-version *Kayak-for-Mask* meets all the customer's requirements. The process of development is described in the next section.

Accomplishment in Each Iteration:

- **Iteration 0**:

  The team leader created a Pivotal Tracker project and a Github repository for the whole team, and ensured that every team member had been added as a contributor. The scrum master and the team leader have the roles as the project owners to review the entire code structure and responsible for the main branch of the project functionalities and maintenance. The project leader organized the customer meetings and the scrum master submitted all the submissions and communicated with TAs. Later, the team discussed the user stories and drew corresponding story cards, as attached in the iteration report. The skeleton of this from-scratch website application was decided in this iteration.

- **Iteration 1**:

  According to the customer's requirements, Django was selected to construct the initial software prototype, populated with two user stories , e.g. displaying mask information in separate columns and showing mask availability. Although the team met some cross-platform issues caused by the compatibility between libraries, the application was able to be run on both Windows 10 and Ubuntu OS before the end of this iteration. The team leader and scrum master who have development experience with Django web framework contributed a lot to software prototyping. The other team members were getting familiarized with how codes work in this python-based web framework.

- **Iteration 2**:

  The required specifications are fully added into the current-version Kayak-for-Mask application. Two main functions were realized, sorting and filtering. Within the top half part of this website, there were two single-selection fields *Sort* and *Availability*, and two multi-choice fields *Brand/Manufacture* and *Size*. After checking sorting and filtering conditions and clicking on the search button, the mask information including brands, sizes, prices, availability, filtration efficiency and purchasing links sorted and filtered by selected options will be displayed. To help developers find potential errors in the current-version application, the feature tester implemented a special python testing method, which is approved by Dr.Walker and the customer.. The regular unit testing approach cannot be used considering the expected project size. Also, the colorless default

background was replaced with a lovely image through implementing CSS, Javascript, and Bootstrap to improve the UI.

- **Iteration 3**:

  A python-based web crawler function was embedded to retrieve real-time price data from the required mask websites. Although two websites use some unknown anti-crawling systems, the needed data can be scraped from most of the mask websites listed by the customer. The corresponding test cases were added to the test file by the feature tester. However, 3M Vflex, a renowned corporation in the field of health care, changes its route addresses frequently, which leads to much trouble to get the valid product url. More handlers should be considered to enhance the web-crawling functionality.

Major Client meetings and feedback:

- **Nov 11th**, customers describe their requirements, such as platform, deployment, user stories, and confirm the project topic, which is designing masks web crawling sites for kids. The customers and the team reach an agreement with all the requirements and demonstrate the capabilities.

  In addition, we have scheduled several meetings with TA and our customer in the past weeks, and received the extension approval from the TA for this iteration 1. During the most recent meeting with our customer, which was held on 11/11/2021 2:30pm, we demonstrated two of our user stories, a working website framework, and did all our testing together with the customer for those stories. The customer's feedback is very positive and excited that the platform is up and progressing. We have discussed our confusions about the project during our working progress and explained to the customer some Programming approach to accomplish what they want.
- **Nov 19th**, We held a meeting with customers on Nov. 19th to discuss the current progress of the Kayak Mask website. The customer was really pleased with your current progress and made a few recommendations. Based on the customer's feedback, we decide to make a few adjustments to our website. In order to give a clear look at the size category, we will only display four different size options available on the website: one size(one size fits all or adjustable mask), extra small, small, and medium-size.Since the prices of the masks are unpredictable and cause misleading issues, the customer

suggested removing the price column so we won't make any confusion to the users. In the previous plan, we were trying to separate the "available to purchase masks" and "out of stock masks" so we can display two different lists of masks for the users. After discussing with the customer, we will include the masks that are out of stock on the default page, so the users will have an overview of complete lists of masks.

- **Dec 10th**. The team has accomplished the customer's needs based on the past meetings, built the web platform, enabled the web crawling abilities of the required websites, deployed the platform, solved the customer deployment issues, and as well as did the testings to provide the feasibility of the whole project. We have received very positive feedback from our customer and she loves our design. Our next and final meeting with the customer will be 12/10 2-3pm.

  On Dec 10th, the customer saw all our useful stories and web crawling functionality, as well as the deployment on Docker Heroku. The customer was very positive about the project's current stage and our final products.

Description of All User Stories (total: 12 points):

1. **User story 1 (4 points)**:

   As a customer, I want to see sizes, filtration efficiencies, manufactures, availability and purchase links of all masks, so that I can choose the mask with needed

   Corresponding features: The website should include the information of sizes, filtration efficiencies, manufacturers, availability and purchase links of all masks.

   Status: Implemented.

   UI & Screenshot:

Name	Brand	Size	Price	Availablity	Filtration Efficiency	Purchasing Link
ASTM F3502	POD	OneSize	$9.99/per item	Yes	98.9%	Purchasing
Cloth mask with filter	Happy Mask	OneSize	$24.0/per item	Yes	99.9%	Purchasing
Cloth mask with filter	Wayre	OneSize	$21.36/per item	Yes	99.9%	Purchasing
Cloth mask with filter	Caraa Tailored Junior Mask	Medium	$25.0/per item	Yes	99.0%	Purchasing
Cloth mask with filter	Honeywell	OneSize	$55.99/per item	Yes	97.0%	Purchasing
Mask with a filter	Flomask	Small	$49.99/per item	Yes	99.8%	Purchasing

2. **User story 2 (4 points)**:

   As a customer, I want to see the real-time price of all masks so that I can compare different masks before placing the order.

   Corresponding features: The website should be able to crawl price data of each mask based on their purchase website whenever the web page is refreshed.

   Status: Implemented.

   UI & Screenshot: Shown in the screenshot in user story 1.

3. **User story 3 (2 points)**:

   As a customer, I want to sort masks based on size, filtration efficiency and name so that I can quickly find the masks I want.

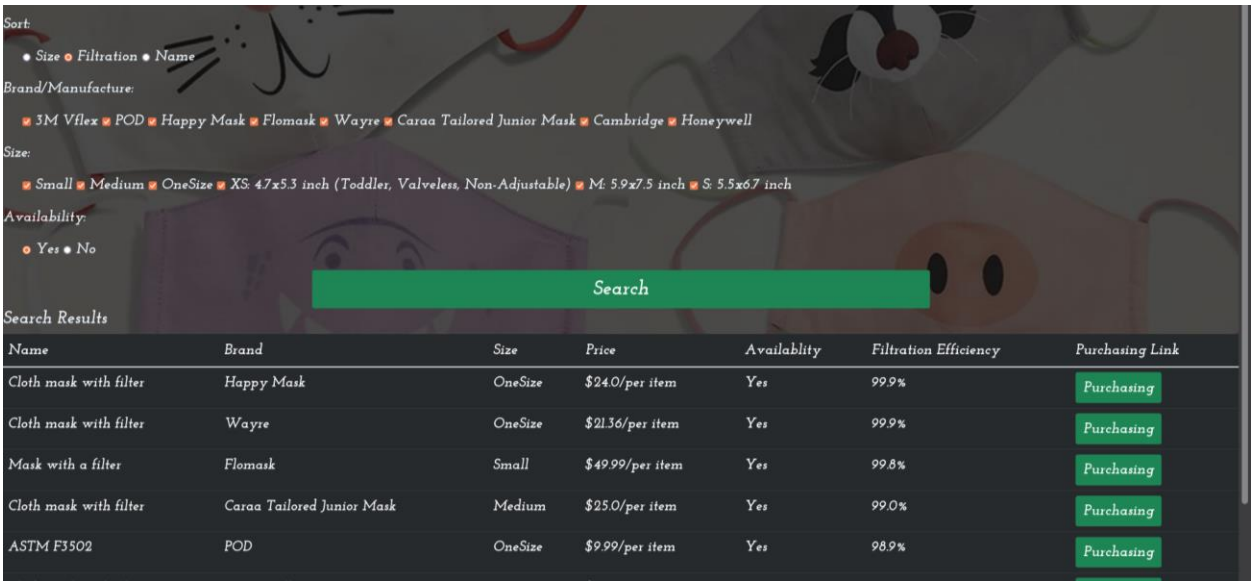
   Corresponding features: The website should allow users to sort masks based on their size, filtration efficiency and name.

   Status: Implemented.

   UI & Screenshot:

   UI:

   Example (sorted by filtration efficiency):



4. **User story 4 (2 points)**:

   As a customer, I want to filter masks based on size and manufacturers, so that I can quickly find the masks I want.

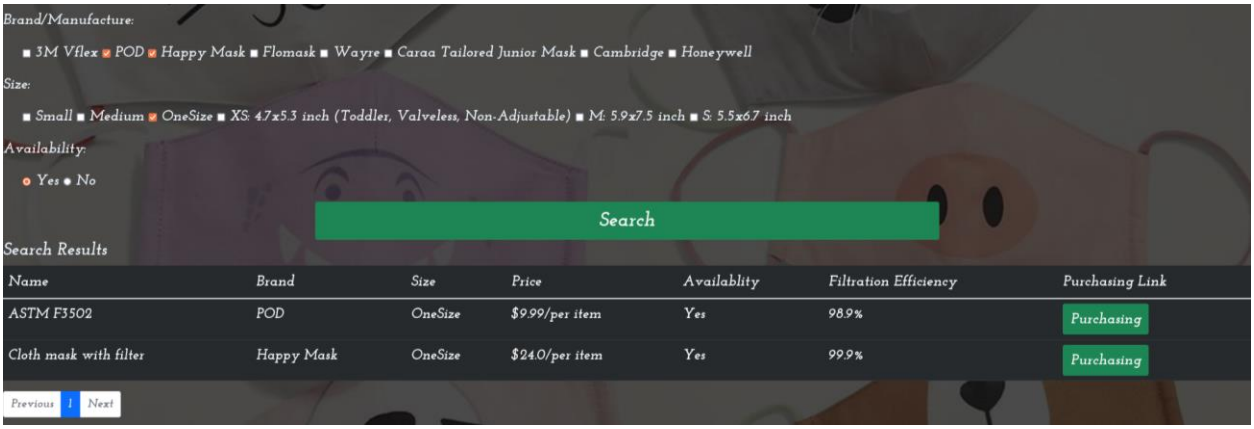
Corresponding features: The website should allow users to filter choices of masks based on their size and manufacturers.

Status: Implemented.

UI & Screenshot:

UI:

Example (Filter masks with size OneSize and Brand POD, Happy Mask):



<u>Test Process</u>:

A special python testing approach is used due to the limitation of the expected project size, which is approved by Dr.Walker and the customer.

First, the feature tester tests if the application's homepage could display correctly. Once the customer enters the home page, the mask list will automatically update to the current time status and display in the front. Second, the availability filter is checked. There are two *availability* options for customers to choose, when the users click on either, searching for masks, the search results filtered by availability will be updated. Then, more filters' functionalities were tested. Also, it's critical to ensure that the purchasing links work well. The *purchasing link* button is available for users to click on and they should be directed to the target website.

In another iteration, the web crawler is tested to check if the application can retrieve real-time price data. The spider is able to receive the url as a website request so that it will crawl data from corresponding HTML elements, and then return the value. The price spider will receive relevant

data from the div web spider, so that the price information can be returned to the HTML element. Abstract price is the real-time price on the mask website. Once the spider receives the price from the crawling, it'll return the value to the HTML element. Below are our user stories and web crawling testing results: (test.py file in the Github for more information)

User Stories Results:

```
(base) C:\Users\makei\OneDrive\Documents\GitHub\Kayak_for_Masks\KayakMask>manage.py test
C:\Users\makei\OneDrive\Documents\GitHub\Kayak_for_Masks\KayakMask
Creating test database for alias 'default'...
System check identified some issues:

WARNINGS:
masklink.MaskInfo: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.
db.models.AutoField'.
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the MasklinkConfig.default_auto_field attribute to point to a
subclass of AutoField, e.g. 'django.db.models.BigAutoField'.

System check identified 1 issue (0 silenced).
.....
----------------------------------------------------------------------
Ran 5 tests in 0.004s

OK
Destroying test database for alias 'default'...
```

Web crawling Results:

```
(base) C:\Users\makei\OneDrive\Documents\GitHub\Kayak_for_Masks\KayakMask>python manage.py test
C:\Users\makei\OneDrive\Documents\GitHub\Kayak_for_Masks\KayakMask
Creating test database for alias 'default'...
System check identified some issues:

WARNINGS:
masklink.MaskInfo: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.
db.models.AutoField'.
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the MasklinkConfig.default_auto_field attribute to point to a
subclass of AutoField, e.g. 'django.db.models.BigAutoField'.

System check identified 1 issue (0 silenced).
....test_abs_price (masklink.tests.abstract_priceTestCase)
....
----------------------------------------------------------------------
Ran 8 tests in 0.005s

OK
Destroying test database for alias 'default'...
```

Software Versions:

Details of our project development and branches history:


* 88020c4 - (HEAD -> main, origin/main, origin/HEAD) Iteration 3 Report uploaded (3 days ago) <yyctamu>
* 8b5c567 - Delete CSCE-Iteration 3 Report.tar (3 days ago) <yyctamu>
* e594d41 - Iteration 3 Report Uploaded (3 days ago) <yyctamu>
* b2f1007 - modify for final deployment (3 days ago) <Zhang>
*   d7652a6 - merge (3 days ago) <Zhang>
|\
| * 13c8d31 - deploy (3 days ago) <Zhang>
| *   2cdf6d0 - Merge remote-tracking branch 'origin/hanzhou_2ndBranch' into main (3 days ago) <Zhang>
| |\
| | * f21943a - (origin/hanzhou_2ndBranch) avail filter modified (4 days ago) <Hanzhou(Marco) Liu>
| | * bf6aeb5 - Delete views.py (4 days ago) <Hanzhou(Marco) Liu>
| | * a1b5a79 - Avail filter modified (4 days ago) <Hanzhou(Marco) Liu>
| | * 76e8202 - Availability = Yes or All (4 days ago) <Hanzhou(Marco) Liu>
| * | bf0a26e - spider all complete (3 days ago) <Zhang>
| * | 3ab2fcc - Unit Test (4 days ago) <KeigoUtdMa>
| |/
| * 483f476 - environment file (4 days ago) <yyc>
| *   adf2ef8 - Web Crawling Updates (4 days ago) <yyc>
| |\
| | * 89082e3 - clean (4 days ago) <yyc>
| | * c5b588f - All Mask (4 days ago) <yyc>
| | * 5872684 - reform function + 1 price for Wayre (5 days ago) <qbsking>
| | * e654a49 - delete some print (5 days ago) <qbsking>
| | * ca76891 - merge spider with previous work (5 days ago) <qbsking>
| | * 7347726 - align with main branch (5 days ago) <qbsking>
| * | 29a1ea1 - Clean Code (4 days ago) <yyc>
| * | 9d88b07 - Merge branch 'main' of https://github.com/OrdinaryCrazy/Kayak_for_Masks (4 days ago) <yyc>
| |\ \
| | * | abf2350 - Iteration 2 Report (2 weeks ago) <yyctamu>
| * | | 6fd39d5 - Merge branch 'main' of https://github.com/OrdinaryCrazy/Kayak_for_Masks (3 weeks ago) <yyc>
| |\ \ \
| * | | | 43c3122 - Updates (3 weeks ago) <yyc>
* | | | | 9eae5b9 - merge for deploy (3 days ago) <Zhang>
| |_|/ /
|/| | |
* | | | b960b7e - Test Cases Update (3 weeks ago) <KeigoUtdMa>
* | | | 1186726 - TestCase (3 weeks ago) <KeigoUtdMa>
| |/ /
|/| |
* | | 9a0ad5c - before deploy (3 weeks ago) <Zhang>
|/ /
* | 5261f45 - merge hanzhou (3 weeks ago) <Zhang>
|\ \
| * | 617d59f - (origin/hanzhou, hanzhou) Multi-choice filtering for 'size' & 'brand' added. (3 weeks ago) <Hanzhou(Marco) Liu>
* | | 7e1cf8b - merge siqi (3 weeks ago) <Zhang>
|\ \ \
| | |/
| |/|
| * | 5dca9f8 - (siqi) sort object change (3 weeks ago) <qbsking>
* | | 61dff91 - multi choice and default value (3 weeks ago) <Zhang>
* | | 174bc5b - Update README.md (3 weeks ago) <yyctamu>
* | | 677ce72 - Iteration 2 All User Stories and Merge Conflicts Fix (3 weeks ago) <yyc>
* | | 32676b6 - Clean Code (3 weeks ago) <yyc>
|\ \ \
| | |/
| |/|
  * | ec969ca - filtering finished (4 weeks ago) <Hanzhou(Marco) Liu>
  * | 605fd23 - filtering finished (4 weeks ago) <Hanzhou(Marco) Liu>
  * | 786d2af - Delete views.py (4 weeks ago) <Hanzhou(Marco) Liu>
  * | 07589b5 - Filtering finished (4 weeks ago) <Hanzhou(Marco) Liu>
| |/
* | 1b20504 - Merge Sort Branch (3 weeks ago) <yyc>
* | ab4b551 - Merge branch 'siqi' (3 weeks ago) <yyc>
|\|
| * 8bc867a - finish sorting (4 weeks ago) <qbsking>
* | 55d0805 - all branches (3 weeks ago) <yyc>
|/
* d4b353e - Iteration 1 Report (4 weeks ago) <yyctamu>
* b1105c2 - Behave update (4 weeks ago) <KeigoUtdMa>
* ee75fd9 - BDD (4 weeks ago) <KeigoUtdMa>
* ef230b2 - Update README.md (4 weeks ago) <yyctamu>
* 20f0b2d - Update README.md (4 weeks ago) <yyctamu>
* c852b9b - Update README.md (4 weeks ago) <yyctamu>
* 8254af5 - Update README.md (4 weeks ago) <yyctamu>
* 6ea2ab5 - Update README.md (4 weeks ago) <yyctamu>
* e2128c1 - Update README.md (4 weeks ago) <yyctamu>
* 7198ea8 - Update README.md (4 weeks ago) <yyctamu>
* 4e76138 - Update README.md (4 weeks ago) <yyctamu>
* 4692dd9 - Update README.md (4 weeks ago) <yyctamu>
* 7505345 - Comment Change on Update Time (4 weeks ago) <yyc>
* 5c7843e - Format WebFrame Fix (4 weeks ago) <yyc>
* ed62bff - Format WebFrame Fix (4 weeks ago) <yyc>
* 8bf814d - Bugs Fixation and Code Cleanup (5 weeks ago) <yyc>
* f9a40e2 - Size and Availibity (5 weeks ago) <yyc>
* 00a5dcf - iteration 1 environment yml (5 weeks ago) <Zhang>
* 0381b47 - iteration 1 need spider size and avai (5 weeks ago) <Zhang>
* 14b2a92 - iteration 1 form bugs (5 weeks ago) <Zhang>
* 71845b1 - iteration 1 needs spider (5 weeks ago) <Zhang>
* 5d9317d - Delete README.md (7 weeks ago) <yyctamu>
* c72ad31 - CSCE 606 Iteration 0 Report (7 weeks ago) <yyctamu>
* 3a8f1d3 - Create the documentation folder (7 weeks ago) <yyctamu>
* fd59c2d - Delete CSCE606-Iteration 0 Report.tar (7 weeks ago) <yyctamu>
* 151db5c - CSCE606 Iteration 0 Report (7 weeks ago) <yyctamu>
* 4046fd4 - Initial commit (8 weeks ago) <JINGTUN ZHANG 张劲暾>

Note: yyc and yyctamu are belong to the same person (Yuncheng Yu).
Note: Zhang and JINGTUN ZHANG are belong to the same person (Jingtun Zhang).

Branches and Release versions:

Yuncheng (Project manager) and Jingtun (Team Leader) mainly contributed to the main branch to review the entire code and solve all the merge conflicts. Other contributors, Hanzhou and Siqi, worked their user stories on individual branches and prepared for the scrum master to approve. Keqiang Yan and Yingying Wang worked on the crawling code base and tested it before sending the code to the whole group. It is marked in the codebase but not submitted by their github account. Keigo worked on the feature tests part, his code did not conflict with the main code structure. By doing this in an organized way, the code can be well-maintenanced. We have a total of 4 branches and three released versions.

Version 0.0: Web prototype
(Jingtun and Yuncheng)
Construct the Python Django Web prototype web framework, syncing with Google Sheet.
Version 1.0: User stories
(Jingtun, Yuncheng, Siqi, HanZhou, Keigo)
Building all the user stories, such as filtering and sorting, on the front-end side and implemented the behavior tests based on constructed stories.
Version 2.0: (Jingtun, Yuncheng, Siqi, HanZhou, Keigo, Keqiang, Yingying)
Implemented the web crawling features and started to sync all the crawling websites information from all the customer's demanded web page lists. In addition, using docker to deploy on Heroku and implemented real-time spiders for products' price.

Issues and Solutions:

Issue 1: Many items in the customer's sheet are not consistent to process
Solution: Communicate with the customer to only process items with exactly one product with a shopping link.
Issue 2: The slug-in size of our project is more than 500MB thus cannot deploy on heroku

Solution: Deploying with Docker, details see **Section Deployment release on Heroku.**

Issue 3: Different Manufacture has their own shopping pages thus cannot consistently crawl the price information.

Solution: Developing spiders for each manufacturers' product. This will lead to some time cost in sacrifice of feedback speed, multi-threading might help with this problem as feature work.

Deployment release on Heroku:

As our project needs lots of libraries like the main framework Django, interfaces for google sheets, spider formatting and other data processing libs like Numpy and Pandas, directly deploying our project on Heroku (the traditional Git plus slug compiler deployments: `` `git push heroku master` ``) will cost more than 1.7G memory, which much more than the 500M limitation of Heroku. So, we choose to deploy our project with the Docker container to make use of the Heroku Container Runtime to manage our project deployment. Compared with the directly approach, Docker based deployment can solve the over size issue by its advantages:

1. **No slug limit**: for Docker based deployment, there is no 500MB slug size limitation
2. **Full control over the OS**: free installation of any package support by Docker
3. **Stronger dev/prod parity**: Docker-based deployment has stronger parity between development and production since the underlying environments are the same.
4. **Less vendor lock-in**: It is much easier to switch to a different cloud hosting provider such as AWS or GCP.

Deployment Type	Deployment Mechanism	Security Updates (who handles)	Access to Pipelines, Review, Release	Access to CLI, Addons, and Dashboard	Slug size limits
Git + Slug Compiler	Git Push	Heroku	Yes	Yes	Yes
Docker + Container Runtime	Docker Push	You	No	Yes	No
Docker + Build Manifest	Git Push	You	Yes	Yes	No

There are two ways to deploy our project with Docker to Heroku: **Container Registry** and **Build Manifest,** our project supports both ways and the final version is deployed by the Manifest way. Our configuration is:

1. For heroku.yml:

```
build:
  docker:
    web: Dockerfile
    worker: Dockerfile
```

2. For Dockerfile:

```
# pull official base image
FROM python:3.7.0

# set work directory
RUN mkdir /masklink
WORKDIR /masklink

# set environment variables
# Prevents Python from writing pyc files to disc
ENV PYTHONDONTWRITEBYTECODE 1
# Prevents Python from buffering stdout and stderr
ENV PYTHONUNBUFFERED 1
ENV DEBUG 0

# install dependencies
COPY ./requirements.txt /masklink/
RUN pip install -r /masklink/requirements.txt

# copy project
COPY . /masklink/

# run gunicorn
CMD python ./manage.py migrate masklink
CMD python manage.py runserver 0.0.0.0:$PORT
```

For other details, we referred to these tutorials:

1. Heroku: Deploying with Docker: Container Registry
2. Heroku: Deploying with Docker: Build your Docker images
3. Deploying Django to Heroku With Docker

References

Mathieu, E., Ritchie, H., Ortiz Ospins, E., Roser, M., Hasell, J., Appel, C., Giattino, C., & Rodés Guirao, L. (2021). A global database of COVID-19 vaccinations. *Nature Human Behaviour*, *5*(7), 947-953.