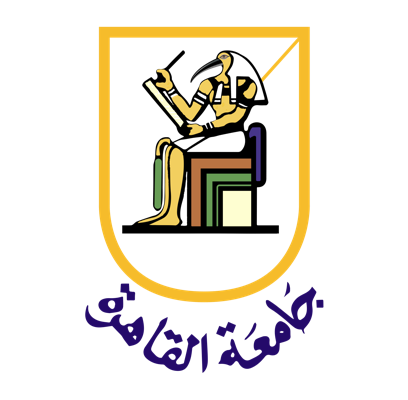
**Cairo University**

**Faculty of Computers and Artificial intelligence Computer Science Department**

Shop Bot (MaWard)

Supervised by

*Prof. Khaled Tawfik*

*TA. Ahmed Ehab*

Implemented by

|  |  |
| --- | --- |
| 20200298 | Abdulsalam Saeed Abdulsalam |
| 20200050 | Ahmed Mohamed Hany |
| 20200205 | Zaid Mahmoud Badr |
| 20200110 | Basel Mohamed |
| 20201115 | Abdulla Mohamed Abdulrhem |

Graduation Project Academic Year 2023-2024 Final Year Documentation

**Table of Content**

[Chapter 1: Introduction 6](#_Toc170828905)

[1.1 Motivation 6](#_Toc170828906)

[1.2 Problem definition 7](#_Toc170828907)

[1.3 Project Objective (suggested solution) 7](#_Toc170828908)

[1.4 Gantt chart of project time plan 9](#_Toc170828909)

[1.5 Project development methodology 10](#_Toc170828910)

[1.6 The used tools in the project (SW and HW) 11](#_Toc170828911)

[1.7 Report Organization (summary of the rest of the report) 12](#_Toc170828912)

[Chapter 2: Related Work 13](#_Toc170828913)

[2.1 The closest examples 13](#_Toc170828914)

[2.2 Main Differences 16](#_Toc170828915)

[Chapter 3: System analysis 18](#_Toc170828916)

[3.1 Functional requirements 18](#_Toc170828917)

[3.2 Non-functional requirements 19](#_Toc170828918)

[3.3 Use Case Diagram 20](#_Toc170828919)

[Chapter 4: System Design 21](#_Toc170828920)

[4.1 System Component Diagram 21](#_Toc170828921)

[4.2 System Class Diagram 22](#_Toc170828922)

[4.3 Sequence Diagrams 23](#_Toc170828923)

[4.4 Project ERD 24](#_Toc170828924)

[4.5 System GUI Design 26](#_Toc170828926)

[Chapter 5: Implementation and testing 28](#_Toc170828927)

[5.1 Challenges 28](#_Toc170828928)

[5.2 Solution 29](#_Toc170828929)

[5.3 Implementation 31](#_Toc170828930)

[5.3.1 Data preparation **Error! Bookmark not defined.**](#_Toc170828931)

[5.3.2 Arabic Dialect detection **Error! Bookmark not defined.**](#_Toc170828932)

[5.3.3 Cleaning phase in Arabic language **Error! Bookmark not defined.**](#_Toc170828933)

[5.3.4 Cleaning phase in English language **Error! Bookmark not defined.**](#_Toc170828934)

[5.4 Algorithms 31](#_Toc170828935)

[5.5 Summarization 38](#_Toc170828936)

[5.6 deployment for each model 40](#_Toc170828937)

[5.7 APIs Implementation 42](#_Toc170828938)

[Reference 43](#_Toc170828939)

**Table of Figures**

FIGURE 1 GANTT CHART PART 1 9

FIGURE 2: USE CASE 18

FIGURE 3: SYSTEM COMPONENT 19

FIGURE 4: CLASS DIAGRAM 20

FIGURE 5: SEQUENCE DIAGRAM FOR WRITE COMMENT 21

FIGURE 6 : SEQUENCE DIAGRAM FOR MAKE SUMMARY 21

FIGURE 7 : SEQUENCE FOR PROFESSOR SHOW REVIEWS 22

FIGURE 8: ERD 22

FIGURE 9: SYSTEM MONOLITHIC ARCHITECTURE 23

FIGURE 10: SIGN UP GUI 24

FIGURE 11: LOG IN GUI 24

[FIGURE 12: HOME PAGE GUI 25](#_bookmark26)

FIGURE 13: TOP RATED GUI 25

FIGURE 14: USER PROFILE GUI 26

FIGURE 15: CREATE COURSE GUI 26

FIGURE 16: UPDATE MATERIAL IN COURSE GUI 27

FIGURE 17: COURSE DETAILS PAGE GUI 27

FIGURE 18: WRITE REVIEW 28

FIGURE 19: UNBALANCED BRAD DATASET 29

FIGURE 20: UNBALANCED AMAZON DATASET 29

FIGURE 21: BALANCED DATASET 30

[FIGURE 22: TREE STRUCTURE MODEL 30](#_bookmark30)

[FIGURE 23: CODE FOR TRANSFORM LABELS TO OBJECTIVE AND SUBJECTIVE 31](#_bookmark33)

[FIGURE 24: CLASSES DISTRIBUTION FOR POSITIVE AND NEGATIVE MODEL 31](#_bookmark35)

[FIGURE 25: CODE FOR ARABIC DIALECT DETECTION 32](#_bookmark38)

FIGURE 27 ARABIC DIALECT PERCENTAGE 32

[FIGURE 28 ARABIC CLEANING AND PREPROCESSING FUNCTION 33](#_bookmark40)

[FIGURE 29 ENGLISH CLEANING AND PREPROCESSING FUNCTION 34](#_bookmark42)

[FIGURE 30 SVM AND VECTORIZATION CODE 35](#_bookmark44)

[FIGURE 31 FITTING DATA SET 35](#_bookmark45)

[FIGURE 32 PREDICT CODE FOR SVM. 35](#_bookmark46)

[FIGURE 33 CODE FOR KNN MODEL AND PIPELINE 36](#_bookmark48)

[FIGURE 34 CODE FOR LOGISTIC REGRESSION MODEL AND PIPELINE 37](#_bookmark50)

[FIGURE 35 CODE FOR DECISION TREE MODEL AND PIPELINE 37](#_bookmark52)

[FIGURE 36 CODE FOR NAIVE MODEL AND PIPELINE 38](#_bookmark54)

[FIGURE 37 CODE FOR LSTM MODEL USING TENSORFLOW 39](#_bookmark56)

[FIGURE 38 CODE FOR CNN MODEL USING TENSORFLOW 40](#_bookmark58)

[FIGURE 39 CODE FOR CNN-LSTM MODEL USING PYTORCH 43](#_bookmark60)

[FIGURE 40 TRANSFORMER PREDICT FUNCTION 44](#_bookmark62)

[FIGURE 41 DATA PREPARATION FOR TRANSFORMER AND MODEL CREATION FOR ENGLISH 45](#_bookmark63)

[FIGURE 42 TRAINER OBJECT AND TRAINING PHASE AND EVALUATION 46](#_bookmark64)

[FIGURE 43 DATASET CLASS TO HANDLE DATASET AND ITEMS 48](#_bookmark66)

[FIGURE 44 DATA PREPARATION FOR TRANSFORMER AND MODEL CREATION FOR ARABIC 50](#_bookmark67)

[FIGURE 45 TRAINER OBJECT AND TRAINING PHASE AND EVALUATION 50](#_bookmark68)

[FIGURE 46 METRICS OF TRANSFORMER ARABIC MODEL 51](#_bookmark70)

FIGURE 47 TXT LENGTH HISTOGRAM 52

FIGURE 48 LENGTH OF SUMMARY HISTOGRAM 52

[FIGURE 49 LANGUAGE DETECTION 53](#_bookmark73)

[FIGURE 50 ENGLISH SPELLING CORRECTION 53](#_bookmark74)

[FIGURE 51 ARABIC SPELLING CORRECTION 53](#_bookmark75)

[FIGURE 52 SENTIMENT ANALYSIS API USING FLASK 55](#_bookmark77)

**Table of Lists**

[1: TABLE OF TOOLS THAT USED 8](#_bookmark4)

[TABLE 2 CLASSES FOR EACH LABEL DISTRIBUTION 31](#_bookmark34)

[TABLE 3 CLASSES FOR EACH LABEL DISTRIBUTION 31](#_bookmark36)

[TABLE 4 METRICS OF SVM MODEL 36](#_bookmark47)

[TABLE 5 METRICS OF KNN MODEL 36](#_bookmark49)

[TABLE 6 METRICS OF LOGISTIC REGRESSION MODEL 37](#_bookmark51)

[TABLE 7 METRICS OF DECISION TREE MODEL 38](#_bookmark53)

[TABLE 8 METRICS OF NAIVE MODEL 38](#_bookmark55)

[TABLE 9 METRICS OF LSTM MODEL 40](#_bookmark57)

[TABLE 10 METRICS OF CNN MODEL 41](#_bookmark59)

[TABLE11 METRICS OF CNN-LSTM MODEL 43](#_bookmark61)

[TABLE 12 METRICS OF333333333 TRANSFORMER ENGLISH MODEL 47](#_bookmark65)

[TABLE 13:SUMMARY TESTING RESULTS 52](#_bookmark71)

**Table of Abbreviations**

|  |  |  |  |
| --- | --- | --- | --- |
| MSA | Middle East and South Asia | OM | Oman |
| KW | Kuwait | YE | Yemen |
| BH | Bahrain | DZ | Algeria |
| EG | Egypt | MA | Morocco |
| SA | Saudi Arabia | PL | Poland |
| SD | Sudan | JQ | Jordan |
| AE | United Arab Emirates | TN | Tunisia |
| IQ | Iraq | SY | Syria |
| LY | Libya | QA | Qatar |
| LB | Lebanon | LSTM | Long short-term memory |
| CNN | Convolutional Neural Networks |  |  |

# Chapter 1: Introduction

## Motivation

A gift selling application with a chatbot to assist customers in finding the perfect gift is a great way to make the gift-giving process more convenient and personalized. With the help of the chatbot, users can receive recommendations based on their preferences, budget, and the recipient's interests. This interactive and helpful feature can enhance the overall shopping experience and increase customer satisfaction.

The chatbot can provide suggestions for various occasions such as birthdays, anniversaries, holidays, or special events. It can ask users relevant questions to gather information about the recipient's age, gender, hobbies, and interests. Based on this input, the chatbot can offer tailored recommendations from the available gift options on the Maward application.

Additionally, the chatbot can assist with other aspects of gift selection, such as providing gift wrapping options, suggesting complementary items, or even offering personalized messages or notes to accompany the gifts. This level of customization can make the gift-giving experience more meaningful and memorable.

Furthermore, the chatbot can leverage customer feedback and past purchase data to continuously improve its recommendations. By learning from user preferences and adapting to individual tastes, the chatbot can refine its suggestions over time, ensuring a more accurate and satisfying gift selection process.

Overall, Maward, with its chatbot-powered gift assistance, can be a valuable platform for customers seeking an effortless and enjoyable gift-buying experience. By combining convenience, personalization, and the joy of giving, Maward has the potential to become a go-to destination for those in search of the perfect gift.

#### Main technique and application:

in Front-End we use:

* Flutter

in Back-End, we use:

* Larval
* My SQL

in Machine learning algorithms, we use:

* Vgg16

## Problem definition

The problem at hand involves two key challenges. Firstly, it is the difficulty in selecting an appropriate gift for a specific occasion, exacerbated by time constraints that hinder the ability to physically purchase the gift. With limited time available, individuals often struggle to identify the most suitable gift that aligns with the occasion and meets the recipient's preferences and expectations. This challenge necessitates a solution that streamlines the gift selection process, offers a range of suitable options, and enables swift acquisition without requiring the physical visitation of stores or extensive time investment.

Secondly, the need to determine the type of a specific flower through a picture when the person is unfamiliar with that particular flower. The person aims to incorporate the flower into a specific gift but lacks the knowledge or expertise to identify the flower type accurately. When encountering an unknown flower, it becomes challenging to understand its characteristics, symbolism, and suitability for the intended gift. Without proper identification, the person may struggle to select a suitable gift that complements the flower's appearance, meaning, or the recipient's preferences. The solution required should offer a mechanism to analyze the provided picture of the flower and provide accurate identification and relevant information about its type.

## Project Objective (suggested solution)

The objective of this project is to develop an application equipped with a chatbot that assists users in efficiently purchasing appropriate gifts. The chatbot will serve two main purposes: firstly, it will aid users in determining the most suitable gift for a given occasion, considering factors such as the recipient's preferences and the nature of the event. Secondly, the chatbot will help users identify and learn about specific types of flowers that they are unfamiliar with and suggest products that incorporate those flowers.

The application aims to streamline the gift-buying process by leveraging the capabilities of the chatbot. It will provide users with a user-friendly interface where they can interact with the chatbot and receive personalized recommendations for gifts based on their input and the chatbot's analysis of the occasion and recipient preferences.

Additionally, when users encounter a flower, they are unfamiliar with, they can upload a picture to the chatbot, which will then identify the flower and suggest relevant products that include that specific flower.

The project's ultimate goal is to create an efficient and user-friendly solution that helps individuals select appropriate gifts quickly, even when time is limited. By incorporating the flower identification feature, the application aims to enhance the personalization and creativity of gift-giving by suggesting products that include specific flowers the user may not have been aware of.

Implementing this application needs tools like:

|  |  |
| --- | --- |
| PHP Laravel for back-end | Pillow |
| Flutter for Mobile Application | Uvicorn |
| TensorFlow | Numpy |
| sklearn |  |
| Resnet 50 |  |
| VGG-16 |  |
| Fast API |  |

*1: Table of tools that used.*

## Gantt chart of project time plan

**A screenshot of a computer

Description automatically generated**

*Figure 1 Gantt Chart part*

## Project development methodology

In the software development life cycle, we use the waterfall model which uses the logical sequence of SDLC steps for a project. Waterfall is an appropriate method because it aims to achieve this goal. It enables early system design and specification changes to be made easily and clearly sets milestones and deadlines. The entire software development process is divided into separate phases. Usually, the result of one stage serves as the input for the next stage sequentially. Stages: **Requirements Gathering**: The first phase in the waterfall methodology is requirements gathering. In this phase, we determined user needs to gather all the functional and non-functional requirements for the project. The requirements

should include the main functionality of the project, which selling gifts and flowers and helping the user choose the appropriate gift as quickly as possible and specifying a type of flower that he does not know to use for a specific gift.

**Analysis and Design**: In this phase, we take the requirements collected in the first phase and create a detailed analysis and design plan. This plan includes all technical specifications and design documents for the application. We also identified potential risks or issues that could arise during the development process.

**Implementation**: The implementation phase involves the actual development and coding of the application. We followed the design documents and coding standards to build the project. The chatbot and the model used for you determine the type of flower through the image and application backend and frontend are implemented at this stage.

**Testing**: Once the implementation phase is completed, we conducted thorough testing of the application to ensure that it meets all the requirements and functions as intended. chatbot and the model used for you determine the type of flower through the image are tested extensively to ensure their accuracy and reliability. application is tested using automation testing tools and quality assurance techniques.

**Maintenance**: Once the application is deployed, it will require ongoing maintenance and updates to ensure that it continues to function correctly. We develop a maintenance schedule and plan to ensure that the application remains up-to-date and relevant. Also issues or bugs that arise should be addressed in a timely manner to maintain the functionality of the application.

## The used tools in the project (SW and HW)

#### Front-end:

* Flutter framework
* Android studio
* VSCode

#### Back-end:

* Php
* Laravel
* MySQL as the database

#### Others:

* Github for Version Control
* Postman for testing API’s

#### Machine learning:

#### Numpy

#### Tensorflow

#### Sklearn

#### Resnet 50

#### Vgg16

## Report Organization (summary of the rest of the report)

The rest of the document discusses the different phases that describe and illustrate the characteristics of the project.

**Chapter Two**: Related Works, here we compare other ecommerce application with chat bot, determine user problems with current related application and similar solutions to that problem.

**Chapter Three**: System Analysis, including functional requirement of the system according to what we found in related works, non-functional requirement that indicate the performance of the system, use case diagram that shows stakeholders and also use case scenarios.

**Chapter Four**: System Design, include all diagrams that describe relations between classes and component in system and how they stored/retrieved in database.

* Component Diagram. - Class Diagram.
* Sequence Diagrams. - Entity relation Diagram.
* GUI design.

**Chapter Five**: Implementation and Testing, include reports about machine learning models and Back-end and testing scenarios results.

#### Summary

The application being developed is designed to assist users in efficiently purchasing suitable gifts with the help of a chatbot. The chatbot within the application serves two main functions: firstly, it aids users in determining the most appropriate gift for a given occasion, considering factors such as the recipient's preferences and the nature of the event. Secondly, it helps users identify unfamiliar types of flowers by analyzing uploaded images and suggests products that incorporate those specific flowers.

The primary goal of the application is to provide an efficient solution for selecting appropriate gifts, even when time is limited. With the chatbot's assistance, users can quickly find the ideal gift and discover new possibilities by incorporating specific flowers they may not have been aware of previously. Overall, the application aims to enhance the gift-giving experience by combining convenience, personalization, and creativity.

# Chapter 2: Related Work

## The closest examples

#### GifterGo.

#### GifterGo is an application that aims to simplify the gift-buying process by providing personalized gift recommendations. The application utilizes chatbot technology to assist users in finding the perfect gift for any occasion quickly and efficiently.

#### With GifterGo, users can interact with the chatbot interface, which gathers information about the recipient, the occasion, and the user's budget and preferences. Based on these inputs, the chatbot employs artificial intelligence algorithms to generate tailored gift recommendations that align with the user's requirements.

#### GiftWizard.

GiftWizard is an application designed to assist users in finding the ideal gifts for various occasions. Powered by artificial intelligence and chatbot technology, GiftWizard simplifies the gift selection process by providing personalized recommendations based on user preferences and recipient information.

When using GiftWizard, users can interact with the chatbot interface, which prompts them to input details such as the occasion, the recipient's age, gender, interests, and relationship to the user. Utilizing machine learning algorithms, the chatbot analyzes this information to generate tailored gift suggestions that align with the recipient's preferences and the user's budget.

#### Winni.

Winni is an application designed to simplify the process of sending gifts and cakes to loved ones. With Winni, users can easily browse through a wide range of gift options, including cakes, flowers, chocolates, personalized items, and more.

In addition to gift selection, Winni also offers a chatbot feature to assist users throughout the process. The chatbot can provide recommendations based on the occasion and recipient's preferences, helping users choose the perfect gift. It also offers support for queries and provides updates on order status, ensuring a smooth and transparent experience.

**Gifts.com.**

The Gifts.com app helps you find and personalize gifts for any occasion. It boasts a wide selection of items categorized by recipient (him, her, kids) and event (birthday, anniversary, graduation). The app focuses on offering unique, high-quality gifts that can be customized to make them extra special. They also highlight the ease of use of their app, making the personalization process a breeze.

In the machine learning model also, there are similar implementations we found on other papers to our model as follow:

### Flower classification using deep convolutional neural networks – IET CVI 2017.0155

In this model they utilized 3 different datasets to train their model on. As they used Oxford 102 dataset which contains 8189 images across 102 flower categories, Oxford 17 with 1360 images and 17 category and Zou-Nagy with 612 images from 102 categories.

In this model they used Fully Convolutional Network (FCN) to segment the flower region within the image. The FCN is initialized using VGG-16 model pretrained on ImageNet.

The segmentation process involves a binary classification task to distinguish between the flower and the background.

In the end of the training, they mentioned training accuracy on the datasets as follows:

- Zou–Nagy Dataset:

Achieved a classification accuracy (CA) of 99.0%.

- Oxford 17 Dataset:

Achieved a classification accuracy of 98.5%.

- Oxford 102 Dataset:

Achieved a classification accuracy of 97.1%

### Flower classification via convolutional neural network – IEEE-7818296

In this dataset they built their own dataset with 79 categories. In this model they utilized convolutional neural network to learn the features for flower classification.

Their neural network consists of five convolutional layers to pick the image features followed by max-pooling layers and three fully connected layers with final 79-way SoftMax.

With this approach they achieved 76.54% classification accuracy on their collected dataset. They tested their model on Oxford dataset, and they achieved with it 84.02% accuracy.

### Efficient deep features selections and classification for flower species recognition

In this model the used a combination of pretrained models Alex Net and VGG16 for features extraction to solve the problem of the similarities between the different types of flowers and they employed support vector machine SVM to classify the flower species.

In this model they focused on extracting the features of the type of flowers in the training, they trained their model in 2 datasets.

One dataset was Flower17 which consists of 17 categories with 80 images for each they said in the paper the accuracy was 96.39% in this model.

Other was Flower102 dataset which mentioned here many times already with 95.70% accuracy in this model on the training set.

In other search papers the methods are almost the same, they focused on feature extraction, and they train their model in the end. In next section we will discuss what we did differently.

Also, in the models and the approaches we found on websites like Kaggle by the developers we found the models has the problem of low accuracy due the weaknesses in the datasets which we will mention later also.

### Efficient deep features selections and classification for flower species recognition

In this model the used a combination of pretrained models Alex Net and VGG16 for features extraction to solve the problem of the similarities between the different types of flowers and they employed support vector machine SVM in order to classify the flower species.

In this model they focused on extracting the features of the type of flowers in the training, they trained their model in 2 datasets.

One dataset was Flower17 which consists of 17 categories with 80 images for each they said in the paper the accuracy was 96.39% in this model.

Other was Flower102 dataset which mentioned here many times already with 95.70% accuracy in this model on the training set.

In other search papers the methods are almost the same, they focused on feature extraction, and they train their model in the end. In next section we will discuss what we did differently.

Also, in the models and the approaches we found on websites like Kaggle by the developers we found the models has the problem of low accuracy due the weaknesses in the datasets which we will mention later also.

## Main Differences

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | MAWARD | Gifts.com | Winnie | GifterGo | GiftWizard |
| Gifts Recommendation | yes | Does not specify products | no | yes | yes |
| Image Classification | yes | yes | no | no | no |
| Manipulate User’s Cart | yes | no | yes | no | no |
| Search Products | yes | no | no | no | no |
| Conversational Style | semi | Human like | semi | semi | semi |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Model 1 | Model 2 | Model 3 | Our Model |
| Dataset | Flower17, Flower102,  Zou–Nagy | Manually collected dataset | Flower17, Flower102 | Flower 102 + collected more data manually |
| Model used | VGG-16 | CNN | VGG-16 + SVM | VGG-16 |
| Training Accuracy | 97.1% | 84.02% | 96.39% | 95% |
| Validation Accuarcy | - | 76.54% | - | 83% |
| Improved Dataset | no | yes | no | yes |

# Chapter 3: System analysis

## Functional requirements

* + - **User Registration:** The application should allow users to create an account to access the chatbot and personalized features.
    - **Gift Recommendation:** The chatbot should assist users in selecting appropriate gifts by analyzing user input, occasion details, recipient preferences, and budget.
    - **Flower Identification**: The chatbot should be able to identify different types of flowers based on user input or uploaded images.
    - **Product Suggestions:** The chatbot should recommend products that contain specific flowers identified by the user.
    - **Product Catalog:** The application should have a comprehensive catalog of gift options, including various categories, such as electronics, fashion, home decor, books, etc.
    - **Search and Filter:** Users should be able to search for specific gifts or filter options based on criteria like price range, brand, or product features.
    - **Order Placement:** The application should allow users to add selected gifts to their cart, provide delivery details, and place orders.
    - **Payment Integration:** The application should support secure and convenient payment options for users to complete their transactions.
    - **Order Tracking:** Users should be able to track the status of their orders and receive notifications regarding delivery updates.
    - **Shopping Cart and Checkout:** The application should provide a shopping cart functionality where users can add selected gifts, review their choices, and proceed to checkout for payment.

## **User Profile Management:** The application should allow users to create and manage their profiles, including personal information, delivery addresses, and saved preferences.

* + - **Admin Dashboard:** a dashboard for admins to manage user accounts, product catalog, order fulfillment, and view analytics.
    - **Reporting and Analytics:** The admin dashboard should provide comprehensive reports and analytics on user activity, popular gifts, and sales performance.

## Non-functional requirements

## **User Interface:** The application should have an intuitive and user-friendly interface that is easy to navigate and interact with.

## **Performance:** The chatbot should respond quickly and provide instant recommendations, ensuring a smooth user experience.

## **Accuracy:** The chatbot's gift recommendations and flower identification should be accurate and reliable.

## **Security:** The application should implement appropriate security measures to protect user data, including personal information and payment details.

## **Scalability:** The application should be designed to handle a growing number of users and accommodate increased data and product catalog size.

## **Reliability:** The application should be available and accessible to users without frequent downtime or disruptions.

## **Compatibility:** The application should be compatible with various devices and operating systems to ensure widespread usability.

## **Privacy:** The application should adhere to privacy regulations and safeguard user information.

## **Error Handling and Logging:** The application should have error handling mechanisms in place to handle exceptions gracefully and log system errors for troubleshooting and debugging purposes.

* + - **Maintainability**: The system should be easy to maintain and update, with clear documentation and a modular design that allows for easy changes and upgrades.

## **Integration:** The application should integrate with external systems and APIs, such as payment gateways and delivery services, to facilitate a seamless user experience.

## Use Case Diagram

A diagram of a diagram

Description automatically generated with medium confidence

*Figure 2: Use Case*

# Chapter 4: System Design

## System Component Diagram

A diagram of a computer server

Description automatically generated

*Figure 3: System Component*

## System Class Diagram

A blue screen with text

Description automatically generated with medium confidence

*Figure 4: Class Diagram*

## Sequence Diagrams

Chat with bot

A diagram of a user view

Description automatically generated

*Figure 5: Sequence diagram for chat with bot*

Sign up

A screenshot of a computer

Description automatically generated

*Figure 6: Sequence diagram for sign up*

Classify with image

A diagram of a user view

Description automatically generated

*Figure 7: Sequence for classify with image*

Buy product

A black screen with white text

Description automatically generated

*Figure 8: Sequence for buy product*

## ERD

A diagram of a diagram

Description automatically generated with medium confidence

*Figure 9: ERD*

## System GUI Design

A screenshot of a login form

Description automatically generated A screenshot of a login screen

Description automatically generated

*Figure 10: SignUp Figure 11: Login*

*ChatBot
* A screenshot of a phone

Description automatically generated

*Figure 12: ChatBot GUI Figure 13: Stripe Payment GUI*

A screenshot of a phone

Description automatically generated Screenshot of a phone screen

Description automatically generated

*Figure 14: Cart Figure15: Search Screen*

*A screenshot of a phone

Description automatically generated* A screenshot of a phone

Description automatically generated

*Figure 16 : Products Details Figure 17: Homepage*

# A screenshot of a login page Description automatically generated

# *Figure 18: Admin Login*

# A screenshot of a computer Description automatically generated

*Figure 19: Dashboard Home*

# A screenshot of a computer Description automatically generated

*Figure 20: Analytics*

# 

*Figure 21: Orders Management*

# A screenshot of a computer Description automatically generated

*Figure 22: Client Statistics*

# A screenshot of a computer Description automatically generated

*Figure 23: Order Details*

# Chapter 5: Implementation and testing

## Challenges

#### Machine learning model challenges

One of the most significant challenges we encountered in developing our machine learning model for flower classification, which is echoed by many researchers in the field as seen in various papers on Google Scholar, is the inherent difficulty of distinguishing between different flower types. Unlike more straightforward image classification tasks, such as differentiating between cats and dogs, flower classification is complicated by the high degree of similarity in shapes, colors, and textures among different flower species.

These subtle differences can be hard for a model to learn, leading to issues with accuracy and generalization. The variations in petal shapes, overlapping color palettes, and similar textural patterns make it a particularly complex problem. This intrinsic complexity necessitates more advanced techniques and careful dataset preparation to achieve reliable classification results. Our efforts to mitigate these challenges included reducing the number of flower types, enriching our dataset with diverse images, and carefully balancing the dataset, ultimately leading to improved model performance.

We initially faced challenges with the Oxford 102 flowers dataset due to the high variation in the number of images per category, causing overfitting. Our data augmentation efforts, aimed at balancing the dataset, resulted in overly similar images. This led to poor validation performance with ResNet50.

## Solution

#### Machine learning model solutions

To address the challenges faced with our flower classification model, we first attempted to balance the Oxford 102 flowers dataset through extensive data augmentation. Given the significant disparity in the number of images per category, we augmented the existing dataset by applying techniques such as zooming in and out, rotating, and changing contrasts. This resulted in a maximum of 1,000 images per type. However, training our ResNet50 model on this augmented dataset led to poor performance on the validation set, with the validation loss increasing significantly after each epoch. This was primarily due to the generated images being overly similar, leading to overfitting.

Recognizing the limitations of this approach, we decided to reduce the number of flower categories from 102 to 50, focusing on those that are more practical and available in the Middle East for our e-commerce system. We then manually collected additional images for each flower type from the internet, ensuring a variety of angles and conditions to enrich the dataset. This resulted in an average of 200-300 images per type.

After rebalancing the model's weights, we retrained ResNet50 on this improved dataset. The results were significantly better, with a training accuracy of 96% and a validation accuracy of 79%, indicating a more robust model with decreasing validation loss across epochs.

To further enhance performance, we opted to train our dataset on a simpler model architecture, selecting the VGG16 pre-trained on ImageNet. VGG16, known for its straightforward architecture comprising 16 weight layers, primarily uses 3x3 convolutional layers stacked on top of each other, with max-pooling layers for down-sampling. This model choice led to remarkable results, achieving a training accuracy of 95% and a validation accuracy of 83%. This outperformed the results of similar models found on platforms like Kaggle, validating our approach and solidifying VGG16 as our model of choice for deployment.

A green and blue rectangular shapes

Description automatically generated with medium confidence  *Figure 24: VGG Architecture*

## A green line graph on a white background Description automatically generated

## *Figure 25: Relation between Train/Validation accuracy, Validation loss to epochs*

## Implementation

#### Machine learning code

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

## 

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

## *Figure 26: machine learning*

## **Mobile App Code**

## **Chatbot Screen**

## **A screenshot of a computer program**

## *Figure 27: Chatbot Screen*

## This screen is responsible for viewing the chatbot which consists of a list of messages (the Messages object) then a small form which has a text form field for the user to input data and the submit button and the send image button.

## **Create Message Cubit**

## A screenshot of a computer program Description automatically generated

## *Figure 28: Create Message Cubit*

## This code defines a MessageCubit class in Flutter using the flutter\_bloc package, which is typically used for state management based on the BLoC (Business Logic Component) architecture.

## “MessageCubit” manages the state of a message creation process. It starts with an initial state (Initial()), transitions to a loading state (“MessageState.loading()”), and then transitions to either a success state (MessageState.success(response)) or a failure state (MessageState.failure(networkExceptions)) based on the outcome of the asynchronous operation performed by \_createMessageRepository. This structure helps in separating the business logic related to message creation from the UI, promoting code organization and reusability.

## **Products Model**

## A screen shot of a computer code Description automatically generated

## ***Figure 29: Products Model***

## The ProductModel class is designed to represent product data in a structured manner. Here's a brief explanation of its key features:

## Inheritance: ProductModel extends Product, inheriting properties and behaviors defined in the Product class. This allows it to reuse common attributes and methods related to products.

## Constructor: It has a constructor that initializes a ProductModel instance with various required attributes such as id, name, description, price, quantity, color, isFavorite, specialText, specialTextPrice, specialImage, specialImagePrice, category, brand, occasion, updatedAt, createdAt, and images. These attributes collectively define the product's characteristics.

## Factory Method (fromJson): This method facilitates the creation of a ProductModel instance from a JSON object (Map<String, dynamic>). It parses JSON data to populate each attribute of the ProductModel, handling optional fields and converting nested JSON objects (like category, brand, occasion, and images) into their respective model instances (CategoryModel, BrandModel, OccasionModel, ImageModel).

## Method (toJson): The toJson method serializes a ProductModel instance back into a JSON format (Map<String, dynamic>). It collects each attribute's value and nests JSON representations of associated model instances (CategoryModel, BrandModel, OccasionModel, ImageModel) within the JSON structure of the product.

## Overall, ProductModel encapsulates product-related data and operations, leveraging inheritance and composition to efficiently manage and serialize/deserialize product information in a Flutter application.

## **Stripe Web View**

## A screen shot of a computer program Description automatically generated

## *Figure 30: Stripe Web View*

## This StripeWebViewScreen Flutter widget sets up a web view for displaying a Stripe payment checkout page and handles navigation based on specific URLs.

## This widget (StripeWebViewScreen) integrates a Stripe payment checkout page into a Flutter app using a web view (WebView from webview\_flutter). It enables interaction with the web view via a WebViewController, handles URL navigation to detect payment success or cancellation, and closes the screen accordingly using Flutter's navigation capabilities (Navigator.of(context).pop(true/false)). This setup allows for a seamless user experience when processing payments within the app.

## **HomePage Body**

## A screen shot of a computer program Description automatically generated

## *Figure 31: HomePage Body*

## The HomeBody widget in Flutter defines the structure and content of the homepage screen.

## The HomeBody widget organizes the homepage screen into different sections using ListView for vertical scrolling. It integrates BLoC state management (BlocProvider, BlocBuilder) to fetch and display slider images dynamically (GetSlidersCubit). Each section (MostSellingProductsSection, CategorySection, etc.) is separated by SizedBox widgets for spacing and follows a structured layout approach for presenting content in a cohesive manner on the screen.

## **Back-End Code**

## **Order Model**

## **A screenshot of a computer program Description automatically generated**

## *Figure 32: Order Model*

## This is Laravel Eloquent model class for an Order. The class includes:

## - Constants for statuses, payment methods, and payment statuses.

## - A protected $fillable array that lists the attributes that can be mass-assigned.

## - Relationships with the User and OrderDetail models using belongsTo and hasMany methods.

## **Order Controller**

## **A screen shot of a computer program**

## A screen shot of a computer code Description automatically generated

## *Figure 33: Order Controller*

## This is a Laravel OrderController with methods for managing orders, including listing (index), creating (store), retrieving (show), handling successful payments (success), and canceling payments (cancel). The controller relies on OrderService for order operations and StripeServices for payment processing. Responses are sent using ApiResponse::sendResponse()

## **Create Order Request**

## 

## *Figure 34: Create Order Request*

## This is a CreateOrderRequest class in Laravel, which extends FormRequest. It includes an authorize method that always returns true and a rules method that defines validation rules for creating an order, ensuring required fields are correctly formatted and validating the attachment field for specific file types.

## **Order Service**

## A screen shot of a computer program Description automatically generated

## A screen shot of a computer program Description automatically generated

## A screen shot of a computer program Description automatically generated

## *Figure 35: Order Service*

## This is an OrderService class in Laravel with methods for handling order operations:

## Dependencies:

## Uses StoreImageAction for handling image storage. Methods:

## getOrders: Retrieves orders for a specific user.

## createOrder: Creates a new order, processes the cart, and updates stock.

## handleAttachment: Handles storing attachments.

## getOrder: Retrieves a specific order by ID with its details.

## createOrderRecord: Creates an order record.

## createOrderDetails: Creates order details for each cart item.

## **Stripe Service**

## A screen shot of a computer program Description automatically generated

## *Figure 36: Stripe Service*

## The StripeServices class integrates Stripe for payment processing in Laravel:

## pay: Creates a Stripe checkout session for an order, setting up line items, payment mode, and success/cancel URLs. It returns the session URL if successful.

## success: Updates the order status to "Paid" upon successful payment, executes stock quantity update, and clears the user's cart items. Returns true if successful.

## **ChatBot Implementation**

## **Cart Flow**

## 

## *Figure 37: Cart Flow*

## This Flow is Responsible for cart management especially getting cart items and deleting any one of them. This flow is excuted after chat gpt classifies the user’s last message as either deleting from cart or getting or viewing cart items.

## A screenshot of a computer Description automatically generated

## *Figure 38: Search and Add to Cart Flow*

## In this flow the search is done after asking the user to input the search keyword then the chatbot asks the user whether he wants to add any one of the search results into his cart.

## Algorithms

### Machine Learning techniques

#### Data Augmentation

To address the imbalanced nature of the Oxford 102 flowers dataset, we implemented extensive data augmentation techniques to increase the diversity of our training data. These techniques included:

* **Rotation**: Random rotations within a specified range to simulate different viewing angles.
* **Width and Height Shifts**: Horizontal and vertical translations to mimic slight positional variations.
* **Shear Transformations**: Shearing operations to simulate changes in perspective.
* **Zooming**: Random zoom in and out to handle varying distances from the camera.
* **Horizontal Flipping**: Flipping images horizontally to account for bilateral symmetry.
* **Fill Mode**: Filling in newly created pixels after transformations to avoid blank spaces.

This augmentation strategy resulted in a more balanced dataset with a maximum of 1,000 images per flower type, reducing the risk of overfitting.

#### Model Selection and Training

Initially, we trained a ResNet50 model on the augmented dataset. However, this model demonstrated poor validation performance due to the overly similar nature of the generated images, leading to overfitting. To mitigate this, we took the following steps:

1. **Reduction of Categories**: We reduced the number of flower categories from 102 to 50, focusing on those more prevalent in the Middle East.
2. **Manual Data Collection**: We manually gathered additional images for each flower type from various internet sources, ensuring a diversity of angles and conditions.
3. **Rebalancing and Training**: With an enriched dataset averaging 200-300 images per type, we rebalanced the model's weights and retrained ResNet50. This resulted in improved performance with a training accuracy of 96% and validation accuracy of 79%.

#### Fine-Tuning with VGG16

To further enhance our model's performance, we opted for a simpler yet effective architecture, . VGG16, pre-trained on ImageNet. VGG16's straightforward architecture, consisting of 16 weight layers with stacked 3x3 convolutional layers and max-pooling layers for down-sampling, was well-suited for our task. We fine-tuned the model as follows:

1. **Unfreezing Layers**: We unfroze the last 8 layers of the VGG16 model for further fine-tuning.
2. **Model Compilation**: We compiled the model using the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 1e-4 and momentum of 0.9.
3. **Training with Class Weights**: We computed class weights to handle any residual class imbalance and trained the model with early stopping, learning rate reduction, and checkpoint callbacks to avoid overfitting and save the best model.

This approach led to significant improvements, achieving a training accuracy of 95% and a validation accuracy of 83%, surpassing the results of similar models and solidifying VGG16 as our model of choice for deployment.

#### Evaluation and Deployment

After fine-tuning, the final model was evaluated on the validation dataset, achieving a validation accuracy of 83%. The final model was then saved for deployment in our e-commerce system, ensuring robust and accurate flower classification suitable for practical applications in the Middle East.

## 

## 5.5 Summarization

## deployment for each model

#### Machine Learning model deployment

In our machine learning project, we needed to create a public API accessible over the internet to interface with our model and process image inputs. We developed a solution using FastAPI in Python, which loads the model, retrieves images from specified URLs, resizes them to 224x224 pixels, passes them through the model, and returns the prediction as a response.

However, we encountered challenges during the deployment phase. Many cloud service providers, such as Heroku and Render, do not offer free plans that meet our requirements. After extensive research, we discovered Railway, a cloud service provider that offers a suitable free plan for deploying our model temporarily for demonstration purposes.

To proceed with the deployment, we performed the following steps:

1. **Created a Private GitHub Repository**:
   * We set up a private repository on GitHub and uploaded our model file (model.h5).
   * We also added our FastAPI code, which handles the model inference and API requests, along with a requirements.txt file that lists all the dependencies needed to run the code.
2. **Connected to Railway**:
   * We logged in to Railway and linked it to our GitHub repository.
   * Railway seamlessly created a Docker image from our repository and initiated the deployment process.
3. **Deployment**:
   * Railway successfully built and deployed the application, providing us with a public API endpoint.

This deployment approach allowed us to efficiently demonstrate our model's capabilities using a publicly accessible API. Railway's seamless integration with GitHub and automated Docker image creation simplified the deployment process, enabling us to focus on refining our model and application.

` Overall, Railway proved to be a reliable and cost-effective solution for deploying our machine learning model, ensuring smooth and hassle-free access for our demo purpose

**Reference**

# AI

[https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-cvi.2017.0155#cvi2bf00486-bib-0005](https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-cvi.2017.0155%23cvi2bf00486-bib-0005)   
  
<https://www.sciencedirect.com/science/article/abs/pii/S0263224119300284>

<https://ieeexplore.ieee.org/abstract/document/7818296/>

<https://paperswithcode.com/paper/resnet-strikes-back-an-improved-training>

<https://geeksforgeeks.org/vgg-16-cnn-model/>

**Related Works**

[**Winni - Cake, Flowers & Gifts - التطبيقات على Google Play**](https://play.google.com/store/apps/details?id=in.winni.app&referrer=utm_source%3Dwinni-website%26utm_medium%3Dweb-link)

[**Gifts.com: Custom Gifts App - التطبيقات على Google Play**](https://play.google.com/store/apps/details?id=com.planetart.gifts)

**Articles**

[**https://medium.com/design-bootcamp/case-study-genie-app-that-helps-you-find-the-perfect-gift-for-someone-98f2c38f522d**](https://medium.com/design-bootcamp/case-study-genie-app-that-helps-you-find-the-perfect-gift-for-someone-98f2c38f522d)

[**https://www.wamda.com/ar/2021/07/floral-marketplaces-does-future-look-rosy-arabic**](https://www.wamda.com/ar/2021/07/floral-marketplaces-does-future-look-rosy-arabic)