



COMSATS University, Islamabad Pakistan

Snap Translator

By

Syeda Rida Batool CIIT/SP17-BSE-046/ISB

Supervisor

Mr. Amir Shabir Pare

Bachelor of Science in Software Engineering (2017-2023)

The candidate confirms that the work submitted is their own and appropriate credit has been given where reference has been made to the work of others.



COMSATS University, Islamabad Pakistan

Snap Translator

**A project presented to
COMSATS University, Islamabad**

**In partial fulfillment
of the requirement for the degree of**

Bachelors Software Engineering (2017-2022)

By

Syeda Rida Batool

CIIT/SP17-BSE-046/ISB

DECLARATION

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other, we will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Syeda Rida Batool

CERTIFICATE OF APPROVAL

It is to certify that the final year project of BS (SE) “Snap Translator” was developed by **Syeda Rida Batool (CIIT/SP17-BSE-046)** under the supervision of “Prof Amir Shabir Pare” and that in his opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Software Engineering.

Supervisor

External Examiner

Head of Department
(Department of Computer Science)

Executive Summary

Many students around the globe for higher studies choose Pakistan in the same way travelers visit due to its cultural and landscape diversity. With this language barrier problem arise for many as many locals in Pakistan still do not know English, neither do foreigners speak Urdu especially overseas students. Especially in Comsats many overseas students are from backgrounds with countries like UAE, Qatar, Egypt, Nigeria and China. For example, a foreign student asks his class fellow to share notes but to only see he has written them in Urdu. Now what either his fellow will translate each and every line for him (but he may not be available all the time) or he has to do it himself using simple translator, where he will type the whole thing to get it translated. This takes a lot of time as well as human effort. Keeping this in mind gave an idea to develop a hand written image scanning language translator.

To prevent all these problems from prevailing, the Snap Translator is developed. It is a translating system, which provides solution to many of these problems. This is an AI based web application and requires computer and internet to operate. It uses image processing to scan images and extract text then translate it.

Snap Translator is an artificial intelligence-based web application that can be assigned with images to get them scanned, extract text from them and then translate them into them into desired language. It offers English to Urdu and Urdu to English translations and uses internet to scan images online.

.

Acknowledgement

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “_Amir Shabir Paare_”. Without their personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are grateful to them for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us with the values of honesty & hard work.

Syeda Rida Batool

Abbreviations

SRS	Software Require Specification
PC	Personal Computer

Table of Contents

1	Introduction.....	12	
1.1	Vision Statement.....	12	
1.2	Related System Analysis/Literature Review	12	
1.3	Project Deliverables.....	12	
1.4	System Limitations/Constraints.....	12	
1.5	Tools and Technologies.....	13	
1.6	Relevance to Course Modules	13	
2	Problem Definition.....	13	
2.1	Problem Statement.....	13	
2.2	Problem Solution	14	
2.3	Objectives of the Proposed System	14	
2.4	Scope.....	14	
2.5	Modules	14	
2.5.1	Module 1: Module Name.....	14	
3	Requirement Analysis	16	
3.1	User classes and characteristics	16	
3.2	Requirement Identifying Technique	16	
3.3	Functional Requirements	16	
3.3.1	Functional Requirement X.....	15	
3.4	Non-Functional Requirements.....	21	
3.4.1	Reliability.....	21	
3.4.2	Usability.....	21	
3.4.3	Performance	21	
3.4.4	Security	17	
3.5	External Interface Requirements	22	
3.5.1	User Interfaces Requirements.....	17	
3.5.2	Software interfaces.....	22	
3.5.3	Hardware interfaces	18	
3.5.4	Communications interfaces.....	22	
4	Design and Architecture	23	
4.1	Architectural Design.....	23	
4.2	Design Models.....	23	
4.3	Data Design	20	
4.3.1	Data Dictionary.....	20	
4.4	Human Interface Design	24	
4.4.1	Screen Images	24	
4.4.2	Screen Objects and Actions	27	
5	Implementation	28	
5.1	Algorithm.....	28	
5.2	External APIs/SDKs	34	
5.3	User Interface.....	35	
5.3.1	Login Screen	5.3.2 Home Screen	35
5.3.3	Assignee Dashboard.....		

5.3.4	New Complaint	25
5.4	Deployment.....	36
6	Testing and Evaluation	36
6.1	Unit Testing	36
6.2	Functional Testing	27
6.3	Business Rules Testing.....	38
6.4	Integration Testing.....	38
7	Conclusion and Future Work	38
7.1	Conclusion	38
7.2	Future Work.....	38
8.	References.....	39

List of Figures

Figure 1 Home Screen	24
Figure 2 Login Screen.....	24
Figure 3 Assignee Dashboard	25
Figure 4 New Complaint.....	25

1 Introduction

1.1 Vision Statement

*For people **who** want to get instant translations **the** Snap Translator is an internet-based web application **that** will allow users to add images both typed and handwritten, which will further translate images into desired language in the form of text scripts and save them for later reviews. **Unlike** current translators available in market, **our product** users who will use snap translator will not only be able to do text and image translations but also hand written image translations. This will save a lot of time especially for international students who can translate notes and can easily view sign boards and banners in Pakistan.*

1.2 Related System Analysis/Literature Review

Table 1 Related System Analysis with proposed project solution

Application Name	Weakness	Proposed Project Solution
Google Docs	This feature was introduced by google and it requires to first convert image into pdf and then translates into text, which is a little time consuming.	Whereas Snap Translator supports direct jpg image processing.

1.3 Project Deliverables

A web application.

A report.

1.4 System Limitations/Constraints

LI-1: Sometimes translated text in case of hand written image may not be 100% accurate, these are minor errors like some grammatical mistakes but overall text is well understood.

LI-2: System is internet-based.

1.5 Tools and Technologies

Table 2 Tools and Technologies for Proposed Project

Tools And Technologies	Tools	Version	Rationale
	MS Visual Studio	2022	IDE
	TensorFlow	2.11.0	DBMS
	Figma	9.0	Design Work
	Technology	Version	Rationale
	Html	1.74	Front-end Development
	CSS	1.74	Front-end Development
	Python	3.11	Back-end Development

1.4 Relevance to Course Modules

Course of Data structures and algorithm has helped me to design useful algorithm, Artificial Intelligence helped in understanding the basics of machine learning, Computer vision course taught the technique of image processing.

2 Problem Definition

This chapter discusses the precise problem to be solved. It extends to include the outcome.

2.1 Problem Statement

As tourism is increasing in Pakistan due to its diverse culture and beautiful landscapes, so is the need for foreigners to understand its local language increasing. Many students among Gulf countries (UAE, Qatar), China, Afghanistan, Nigeria and Egypt choose Pakistan for higher studies therefore, this provided an idea to design a system that will waste international students as well as travelers to understand local language easily. The system will allow them to take pictures of hand written notes, banners, billboards, roadside signs, market boards, restaurant menus and even beautiful and fun truck art poetry and convert it into readable English text. In this way it will save a lot of time in translating sentence by sentence by writing it every time online.

2.2 Problem Solution

Many students face the problem when they need to translate a paragraph or maybe a document as it is time consuming so they needed a less time consuming easy all-in-one solution for translating scripts by just taking a picture. System is also helpful for those who are learning Urdu or English as it allows user to save translated files into a separate folder which can be accessed anytime later.

2.3 Objectives of the Proposed System

BO-1: Reduce the time consumed up to half the time it takes for normal translations.

BO-2: Increases effectiveness by translating hand written images.

BO-3: Saving translated text scripts for later access saves time.

2.4 Scope

There is a huge number of audiences who visit different countries for work purposes, travelling and most importantly students overseas who are not familiar with regional languages and are trying to learn one.

Although there are several translator software and apps are developed in market most of them focus on text or voice translations. For example, in case of voice recognition it is sometimes impossible for user to properly speak a language in regional accent. If you are travelling abroad, it might become a problem for one to actually figure out the sign boards at roads, hotel names, restroom boards, instructions, shop boards, shopping tags, grocery price tags, café and restaurant menus etc.

A huge number of audiences visit Pakistan for work purposes, travelling and study overseas

‘Snap Translator’ is a web-based system which mainly focuses on translating text from images as sometimes it is hard to pronounce a phrase like on some applications that use voice recognition to translate, therefore user can simply take a picture of any sign board, price tags, documents, product ingredient list, restaurant menus, grocery packaging etc. User can also search simple phrases or words by typing text. Along with image scanning system also enable users to upload a document or image from device and system translates them as well. User can also download these translated results so they can be viewed anytime while remaining offline.

2.5 Modules

2.5.1 Module 1: Language detection

FE-1: Is responsible for recognizing text of entered image whether it is text or hand written.

2.5.2 Module 2: Urdu to English model

FE-1: Views, analyses Urdu text from translations directory.

FE-2: Converts Urdu text/handwritten script into English.

2.5.3 Module 3: English to Urdu model

FE-1: Views, analyses English text from translations directory.

FE-2: Converts English text/handwritten script into Urdu.

2.5.4 Module 4: Image to text model

FE-1: Processes image text/script.

FE-2: Converts and shows into text translation.

3 Requirement Analysis

3.1 User classes and characteristics

User Class	Description
Person	Users are universal for this system as they add images to get them translated.

3.2 Requirement Identifying Technique

Event	System State	Response
Person selects eng to urdu/ urdu to eng translation option.	Eng to urdu/urdu to eng is highlighted.	A call is created to system. Upload image dialogue box appears.
Upload image box appears.	Box shows options to upload images from device.	An image is selected for translation. Image is uploaded.
Image is uploaded.	Image is updated on dashboard.	Image is ready to get scanned.
Image scanner is activated.	Scanning box is available to move on image text.	Image is scanned by confirming and provides urdu/english text translation.
Save translated text.	Text is saved in a separate folder.	Translation saved successfully.

3.2 Functional Requirements

This section describes the functional requirements of the system expressed in the natural language style. This section is typically organized by feature as a system feature name and specific functional requirements associated with this feature. It is just one possible way to arrange them.

3.2.1 Import Language

Identifier	FR-1
Title	Import language
Requirement	System should allow user to import languages from API without having to go outside system.
Source	Client
Rationale	The motivation behind the requirement
Business Rule (if required)	No restriction of language.
Dependencies	N/A
Priority	High

3.2.2 Updating new translations

Identifier	FR-2
Title	Updating new translations
Requirement	This will allow system to keep a record of which languages are getting translated and what language is translated into so user does not need to add/select a specific language over and over.
Source	System
Rationale	The motivation behind the requirement
Business Rule (if required)	System will generate usage reports and save it back in a repository.
Dependencies	FR-1
Priority	Medium

3.2.3 Save translations

Identifier	FR-3
Title	Save translations
Requirement	User will save translated files on system in order to keep notes of phrases or translations which need to be accessed repeatedly.
Source	Client
Rationale	The motivation behind the requirement
Business Rule (if required)	Downloaded translations can be accessed while remaining offline.
Dependencies	N/A
Priority	High

3.2.4 Scan Image

Identifier	FR-4
Title	Scan image
Requirement	This will allow system to recognize uploaded file as a single image or document then will send it to image processor tool to scan written text and return image to language repository.
Source	Client
Rationale	The motivation behind the requirement
Business Rule (if required)	No risk and secure processing.
Dependencies	FR-1
Priority	High

3.2.5 Image Translation

Identifier	FR-5
Title	Image translation
Requirement	This is further extension of image scanning which will detect text on images and translate it into selected language.
Source	Client
Rationale	The motivation behind the requirement
Business Rule (if required)	To which language text will be translated must be selected before executing image translation.
Dependencies	FR-1, FR-4
Priority	High

3.2.6 Secure internet connection

Identifier	FR-6
Title	Secure connection
Requirement	This can be done to ensure network connection is secure or is device connected to internet.
Source	System
Rationale	The motivation behind the requirement
Business Rule (if required)	No restriction and risk
Dependencies	N/A
Priority	High

3.2.7 Testing correct translation

Identifier	FR-7
Title	Testing correct translation
Requirement	This can be done to ensure whether translations are performed according to selected languages.
Source	Client
Rationale	The motivation behind the requirement
Business Rule (if required)	Risk is managed.
Dependencies	FR-1
Priority	Low

3.2.8 Data Privacy

Identifier	FR-8
Title	Data privacy
Requirement	Here system will ensure the authenticity and privacy of saved data.
Source	System
Rationale	The motivation behind the requirement
Business Rule (if required)	No risk.
Dependencies	FR-5
Priority	High

3.2.9 Delete Translations

Identifier	FR-9
Title	Delete translations
Requirement	An option for saved translations is that they can be deleted anytime from system downloads.
Source	Client
Rationale	The motivation behind the requirement
Business Rule (if required)	None.
Dependencies	FR-3
Priority	Low

3.3 Non-Functional Requirements

3.3.1 Reliability

REL-1: Images files and text will be translated successfully with right translations.

REL-2: There should be no disruption of the internet connection during image scanning.

3.3.2 Usability

USE-1: User must have a windows PC or laptop.

USE-2: User cannot disconnect the internet connection throughout translation process.

3.3.3 Performance

PER-1: Total text translation time should be less than 5 secs.

PER-2: Image translation time shall be less than 10 secs.

PER-3: One image can be scanned for translation at a time.

PER-4: .doc and .jpg files will take 20 secs to 60 secs depending upon file length.

PER-5: GUI response time should be from 0.1 seconds to 0.5 seconds

3.5 External Interface Requirements

This section provides information to ensure that the system will communicate properly with users and with external hardware or software elements. A complex system with multiple subcomponents should create a separate interface specification or system architecture specification. The interface documentation could incorporate material from other documents by reference. For instance, it could point to a hardware device manual that lists the error codes that the device could send to the software.

3.5.1 Software interfaces

SI-1: Image Processing System

SI-1.1: It is responsible to process text written in image and transmit to Google API for language translation.

3.5.2 Communications interfaces

CI-1: System uses programmatic web browser to host image translations.

4 Design and Architecture

The following parts of Software Design Description (SDD) report are included in this.**Architectural Design**

Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high-level overview of how the system's modules collaborate with each other in order to achieve the desired functionality.

Don't go into too much detail about the individual subsystems. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together.

Provide a diagram showing the major subsystems and their connections.

- In initial design stage create Box and Line Diagram for simpler representation of the systems
- After finalizing architecture style/pattern diagram (MVC, Client-Server, Layered, Multi-tiered) create a detailed mapping modules/components to each part of the architecture

To view example of box and line diagram and architecture styles, see Appendix B.

4.2 Design Models

Create design models as are applicable to your system. Provide detailed descriptions with each of the models that you add. Also ensure visibility of all diagrams.

Design Models for Object Oriented Development Approach

The applicable models for the project using object-oriented development approach may include:

- Activity Diagram
- Class Diagram
- Sequence Diagram
- State Transition Diagram (for the projects which include event handling and backend processes)

Design Models for Procedural Approach

The applicable models for the project using procedural approach may include:

- Activity Diagram
- Data Flow Diagram (data flow diagram should be extended to 2-3 levels. It should clearly list all processes, their sources/sinks and data stores.)
- State Transition Diagram (for the projects which include event handling and backend processes)

To view examples of all above models, see Appendix C

4.3 Human Interface Design

- Users can select options from English and Urdu, which language image is going to get translated.
- They can select an image to be uploaded using browse image option.
- Uploaded image appears on dashboard which can be scanned using a scanning box by adjusting it to image text.
- Select Done Crop option to extract text from image and then translate it into another language.

4.1.1 Screen Images

Urdu - English OCR + translation

Choose Language

Language :

en

ur

English images



Choose Image

Uploading image



Fig-1 shows selecting the language which needs to be translated.



Fig-2 shows after selecting browse image option show in fig-1 image is uploaded and ready to get scanned.



Fig-3 shows text extracted from scanned image.

رکٹ ایشیا کپ ہاکی فٹبال ٹیم

preprocessed img

Preparing the reader...

Original text extracted from the picture :

نجاتن کپسز ایسیٹس سر جے ۲ ہائی ووڈ سینڈا گورنگار رقص توفیق مشہور عالمی القادسہ انکم ٹیکس ٹرینڈ اینڈا کپ ہاکی فٹبال ٹیم

Translated text from Urdu to English:

Nizam Cancer Diabetes Surgery De Bollywood Cinema singer Dance Extraordinary World Account Income Income Tax Customs Customs Asia Cup Hockey Football Team

Fig-4 shows English translation of Urdu text.

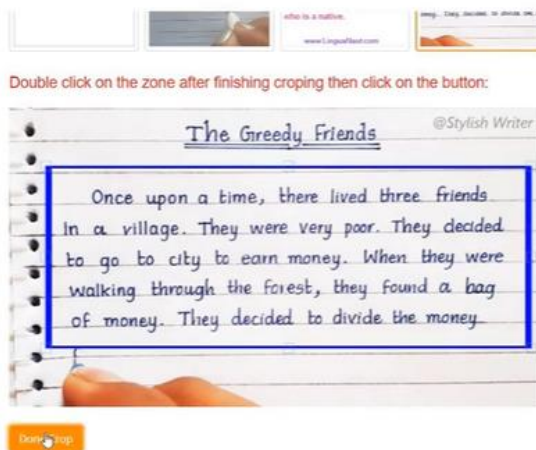


Fig-5 shows an English text image being cropped for scanning.

to go to city to earn money. When they were walking through the forest, they found a bag of money. They decided to divide the money.

preprocessed img

Preparing the reader...

Original text extracted from the picture :

Once upon time , there lived Three friend in CI village . They were very poor They decided bo to clay to earn money When go they were walking Through the Forest , they Found 1 bag of money They decided to divide the money

Translated text from English to Urdu:

ایک بار وقت پر ، وہاں سے تین دوست رہتے تھے وہ بہت غریب تھے انوں نے جنگل میں گھومنے پھرتے ہوئے پیسوں کے
ٹکڑے پائے تھے وہ ان ٹکڑوں کا فیصلہ کیا ، انیس 1 پیسہ ملا جس سے اس رات کو ان کا فیصلہ کیا تھا

Fig-6 shows English to Urdu translation.

4.1.2 Screen Objects and Actions

- At home screen two options can be seen 'eng' and 'ur' when one is selected then 'Browse Image' option occurs which allows to upload images from the present device or from cloud.
- After selecting desired image confirm and upload.
- Now the screen shows uploaded image, with a scanning box ready to scan image.
- Adjust the scanning box on the parts of text to get translated and click 'Done Crop'.
- Now the screen shows extracted text from the image first and in second box it shows its translation.

5 Implementation

5.1 Algorithm

Mention the algorithm(s) used in your project to get the work done with regards to major modules. Provide a pseudocode explanation regarding the functioning of the core features. Be sure to use the correct syntax and semantics for algorithm representations. Following are few examples of algorithms/pseudocode:

Table 3 Example of Algorithm

Algorithm 1 Main
<pre>Input: from PIL import Image, ImageDraw import numpy as np import tensorflow as tf import easyocr from googletrans import Translator from autocorrect import Speller import re import streamlit as st from streamlit_cropper import st_cropper import cv2 from streamlit_image_select import image_select from imutils.perspective import four_point_transform def preprocess(im): gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY) # threshold the image using Otsu's thresholding method _, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV cv2.THRESH_OTSU) return thresh def draw_boxes(image, bounds, color='yellow', width=2): draw = ImageDraw.Draw(image) for bound in bounds: p0, p1, p2, p3 = bound[0] draw.line([*p0, *p1, *p2, *p3, *p0], fill=color, width=width) return image</pre>

```

st.title(" Snap Translator")

st.markdown('<p      style="font-family:sans-serif;      color:Green;      font-size:
32px;">Choose Language</p>', unsafe_allow_html=True)
lang = st.radio('Language :',
('en', 'ur'))
if lang == 'ur':
    im      =      image_select("Urdu      images",      ["" ,"img/sample_img9.jpg",
"img/sample_img10.jpg", "img/sample_img_11.jpg"])
    st.write("img:", im)
elif lang == 'en':
    im      =      image_select("English      images",      ["" ,"img/sample_img_eng.jpg",
"img/sample_img_eng4.jpg", "img/sample_img_eng3.jpg"])

if im == "":
    st.markdown('<p      style="font-family:sans-serif;      color:Green;      font-size:
32px;">Choose Image</p>', unsafe_allow_html=True)
    uploaded_file = st.file_uploader("Uploading image", type="jpg")
else :
    uploaded_file = im
if uploaded_file is not None:

    if lang == 'ur':
        lang_target = 'en'
        im = Image.open(uploaded_file)
        st.markdown(f'<p      style="font-family:sans-serif;      color:Red;      font-size:
20px;">Double click on the zone after finishing cropping then click on the
button:</p>', unsafe_allow_html=True)
        cropped_img = st_cropper(im, realtime_update=False, aspect_ratio=None)
        if st.button(' Done Crop'):
            st.image(cropped_img, caption = 'cropped img', use_column_width='auto')
            cropped_img.save("cropped_img.jpg")
            cropped_img = Image.open("cropped_img.jpg")
            opencvImage = cv2.cvtColor(np.array(cropped_img), cv2.COLOR_RGB2BGR)
            preproc_im = preprocess(opencvImage)
            st.image(preproc_im,      caption      =      'preprocessed      img',
use_column_width=True)
            st.write("")

```

```

        st.markdown('<p style="font-family:sans-serif; color:Red; font-size:
32px;">Preparing the reader...</p>', unsafe_allow_html=True)
        reader = easyocr.Reader(['ur'])
        bounds = reader.readtext(cropped_img, text_threshold = 0.5, paragraph =
True)

        #draw_boxes(im, bounds)
        text = []
        for i in range(len(bounds)):
            text.append(bounds[i][-1])
            #print(bounds[i][-1])
        text = '\n'.join(text)
        st.markdown('<p style="font-family:sans-serif; color:Blue; font-size:
25px;"> Original text extracted from the picture :</p>', unsafe_allow_html=True)
        st.write(text)
        translator = Translator()
        trans_ur = translator.translate(text,src = 'ur', dest = 'en').text
        st.write('')
        st.markdown(f'<p style="font-family:sans-serif; color:Blue; font-size:
25px;">Translated text from Urdu to English:</p>', unsafe_allow_html=True)
        st.write(trans_ur)

    elif lang == 'en':
        lang_target = 'ur'
        im = Image.open(uploaded_file)
        st.markdown(f'<p style="font-family:sans-serif; color:Red; font-size:
20px;">Double click on the zone after finishing cropping then click on the
button:</p>', unsafe_allow_html=True)
        cropped_img = st_cropper(im, realtime_update=False, aspect_ratio=(1,1))
        if st.button(' Done Crop'):
            st.image(cropped_img, caption = 'cropped img', use_column_width=True)
            cropped_img.save("cropped_img.jpg")
            cropped_img = Image.open("cropped_img.jpg")
            opencvImage = cv2.cvtColor(np.array(cropped_img), cv2.COLOR_RGB2BGR)
            preproc_im = preprocess(opencvImage)
            st.image(preproc_im, caption = 'preprocessed img',
use_column_width=True)
            st.markdown('<p style="font-family:sans-serif; color:Red; font-size:
32px;">Preparing the reader...</p>', unsafe_allow_html=True)
            reader = easyocr.Reader(['en'])
            bounds = reader.readtext(cropped_img, text_threshold = 0.5, paragraph =
True)

            #draw_boxes(im, bounds)
            text = []
            for i in range(len(bounds)):

```

```

        text.append(bounds[i][-1])
        #print(bounds[i][-1])
    text = '\n'.join(text)
    spell = Speller(lang='en')
    text = re.sub('[^A-Za-z0-9\.,,]+', ' ', text)
    text_corrected = spell(text)
    st.markdown('<p style="font-family:sans-serif; color:Blue; font-size: 25px;"> Original text extracted from the picture :</p>', unsafe_allow_html=True)
    st.write(text_corrected)
    translator = Translator()
    trans_en = translator.translate(text_corrected,src =lang, dest = 'ur').text
    st.write('')
    st.markdown(f'<p style="font-family:sans-serif; color:Blue; font-size: 25px;">Translated text from English to Urdu:</p>', unsafe_allow_html=True)
    st.write(trans_en)

```

Algorithm 2 utils

```

Input: import base64
import io
import os
from pathlib import Path

import numpy as np
import streamlit as st
import streamlit.components.v1 as components
from PIL import Image
from utils import image_select

_RELEASE = True

if not _RELEASE:
    _component_func = components.declare_component(
        "image_select", url="http://localhost:3001"
    )
else:
    path = (Path(__file__).parent / "frontend" / "build").resolve()
    _component_func = components.declare_component("image_select", path=path)

```

```

@st.experimental_memo
def _encode_file(img):
    with open(img, "rb") as img_file:
        encoded = base64.b64encode(img_file.read()).decode()
    return f"data:image/jpeg;base64, {encoded}"

@st.experimental_memo
def _encode_numpy(img):
    pil_img = Image.fromarray(img)
    buffer = io.BytesIO()
    pil_img.save(buffer, format="JPEG")
    encoded = base64.b64encode(buffer.getvalue()).decode()
    return f"data:image/jpeg;base64, {encoded}"

def image_select(
    label: str,
    images: list,
    captions: list = None,
    index: int = 0,
    *,
    use_container_width: bool = True,
    return_value: str = "original",
    key: str = None,
):
    """Shows several images and returns the image selected by the user.
    Args:
        label (str): The label shown above the images.
        images (list): The images to show. Allowed image formats are paths to local
            files, URLs, PIL images, and numpy arrays.
        captions (list of str): The captions to show below the images. Defaults to
            None, in which case no captions are shown.
        index (int, optional): The index of the image that is selected by default.
            Defaults to 0.
        use_container_width (bool, optional): Whether to stretch the images to the
            width of the surrounding container. Defaults to True.
        return_value ("original" or "index", optional): Whether to return the
            original object passed into `images` or the index of the selected
            image.

```



```

        Defaults to "original".
        key (str, optional): The key of the component. Defaults to None.
Returns:
    (any): The image selected by the user (same object and type as passed to
        `images`).
    """

    # Do some checks to verify the input.
    if len(images) < 1:
        raise ValueError("At least one image must be passed but `images` is
empty.")
    if captions is not None and len(images) != len(captions):
        raise ValueError(
            "The number of images and captions must be equal but `captions` has "
            f"{len(captions)} elements and `images` has {len(images)} elements."
        )
    if index >= len(images):
        raise ValueError(
            f"`index` must be smaller than the number of images ({len(images)}) "
            f"but it is {index}."
        )

    # Encode local images/numpy arrays/PIL images to base64.
    encoded_images = []
    for img in images:
        if isinstance(img, (np.ndarray, Image.Image)): # numpy array or PIL image
            encoded_images.append(_encode_numpy(np.asarray(img)))
        elif os.path.exists(img): # local file
            encoded_images.append(_encode_file(img))
        else: # url, use directly
            encoded_images.append(img)

    # Pass everything to the frontend.
    component_value = _component_func(
        label=label,
        images=encoded_images,
        captions=captions,
        index=index,
        use_container_width=use_container_width,
        key=key,
        default=index,
    )

```

```

# The frontend component returns the index of the selected image but we want to
# return the actual image.
if return_value == "original":
    return images[component_value]
elif return_value == "index":
    return component_value
else:
    raise ValueError(
        "`return_value` must be either 'original' or 'index' "
        f"but is '{return_value}'."
    )

```

5.2 External APIs/SDKs

Describe the third-party APIs/SDKs used in the project implementation in the following table. Few examples of APIs are provided in the table.

Table 5 Details of APIs used in the project

Name of API and version	Description of API	Purpose of usage	List down the API endpoint/function/class in which it is used
Google Cloud translation API (version v3beta1)	Auto ML models for translating text,	Used for translating processed images.	Recogniselanguage_

5.2 User Interface

5.2.1 Home Screen

Urdu - English OCR + translation

Choose Language

Language :



English images



Choose Image

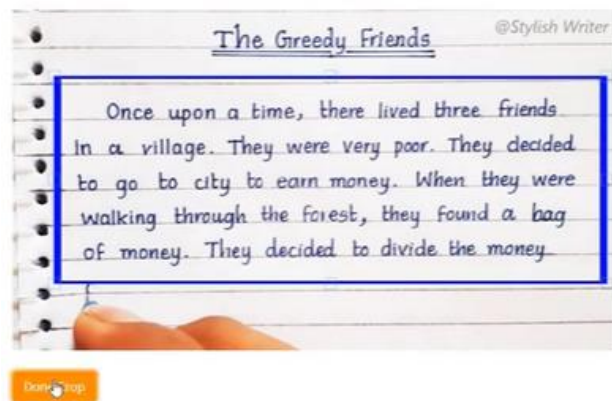
Uploading image



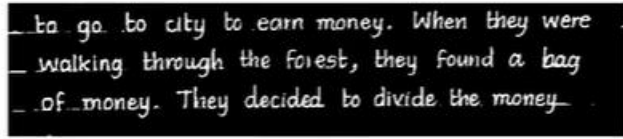
5.2.2 Uploading Image



Double click on the zone after finishing cropping then click on the button:



5.2.3 Translating Image



preprocessed img

Preparing the reader...

Original text extracted from the picture :

Once upon time , there lived Three friend in CI village . They were very poor They decided bo to clay to earn money When go they were walking Through the Forest , they Found 1 bag of money They decided to divide the money

Translated text from English to Urdu:

ایک بار وقت پر . . . وہیں سے اہل و عیال میں تین دوست رہتے تھے وہ بہت غریب تھے انہوں نے جنگ میں لڑتے ہوئے پانچ سو روپے کیے
کھانے کے لئے پانچ سو روپے کیے ، انہوں نے 1 بیگ ملا جس میں اس رقم کا تقسیم کر کے کا فیصلہ کیا تھا

5.3 Deployment

This is web application created using MS Visual Studio. HTML and CSS were used for front-end development while back-end models were created in Python. System is created and hosted on web app development framework Streamlit.

6 Testing and Evaluation

6.1 Unit Testing

Unit Testing 1:

Testing Objective: To ensure English images upload successfully.

No.	Test case/Test script	Attribute and value	Expected result	Result
1	Check English option for browsing only english images for translation	English Image	Validates successful uploading of image.	Pass
2	Check whether an error occurs by uploading urdu picture while	Urdu Image	Displays error message and asks to choose another	Pass

	selecting english option.		file.	
--	---------------------------	--	-------	--

Unit Testing 2:

Testing Objective: To ensure Urdu images upload successfully.

No.	Test case/Test script	Attribute and value	Expected result	Result
1	Check Urdu option for browsing only urdu images for translation	Urdu Image	Validates successful uploading of image.	Pass
2	Check whether an error occurs by uploading english picture while selecting urdu option.	English Image	Displays error message and asks to choose another file.	Pass

Unit Testing 3:

Testing Objective: To ensure image scanning crop box works successfully.

No.	Test case/Test script	Attribute and value	Expected result	Result
1	Crop a part of text to see if only that part is translated.	Cropped image	Validates successful translation.	Pass
2	Crop to check it does not translate the non-selected part.	Cropped image	Displays only selected part.	Pass

Unit Testing 4:

Testing Objective: To ensure English to Urdu translation.

No.	Test case/Test script	Attribute and value	Expected result	Result
1	Check that english images generate urdu translation.	English image and option selected	Validates successful translation of image in urdu.	Pass
2	Check that urdu images generate english translation.	Urdu image and option selected	Validates successful translation of image in english.	Pass

6.2 Business Rules Testing

There are no business rules for this application.

6.3 Integration Testing

Integration Testing 1: Image scanning and translation

Testing Objective: To ensure the scanning, recognizing natural language from API and translating images work cordially well together.

No.	Test case/Test script	Attribute and value	Expected result	Actual result	Result
1.	Scan Image	Uploaded image, selected preferred language to get translated.	Successfully scans and crops the selected part.	Cropped script generated successfully.	Pass
2.	Translate Image	Image, cropped text from image and selected language preference.	Successfully translates cropped text from image into desired language.	Translates successfully	Pass

7 Conclusion and Future Work

7.1 Conclusion

In conclusion to the above document each chapter has been detailed thoroughly, from requirements to design to architecture to testing each phase has been examined properly. The idea of this project was to help students and travelers who find it difficult to understand our local language and translate it quickly.

7.2 Future Work

Right now, this system supports only two languages, but in future it is intended to expand language recognition models for multiple languages.

8. References

World Wide Web

<https://docs.opencv.org/4.x/>

<https://docs.streamlit.io/>

Appendix A

Example: Context Diagram

The Cafeteria Ordering System is a new software system that replaces the current manual and telephone processes for ordering and picking up meals in the Process Impact cafeteria. The context diagram in Figure 1 illustrates the external entities and system interfaces.

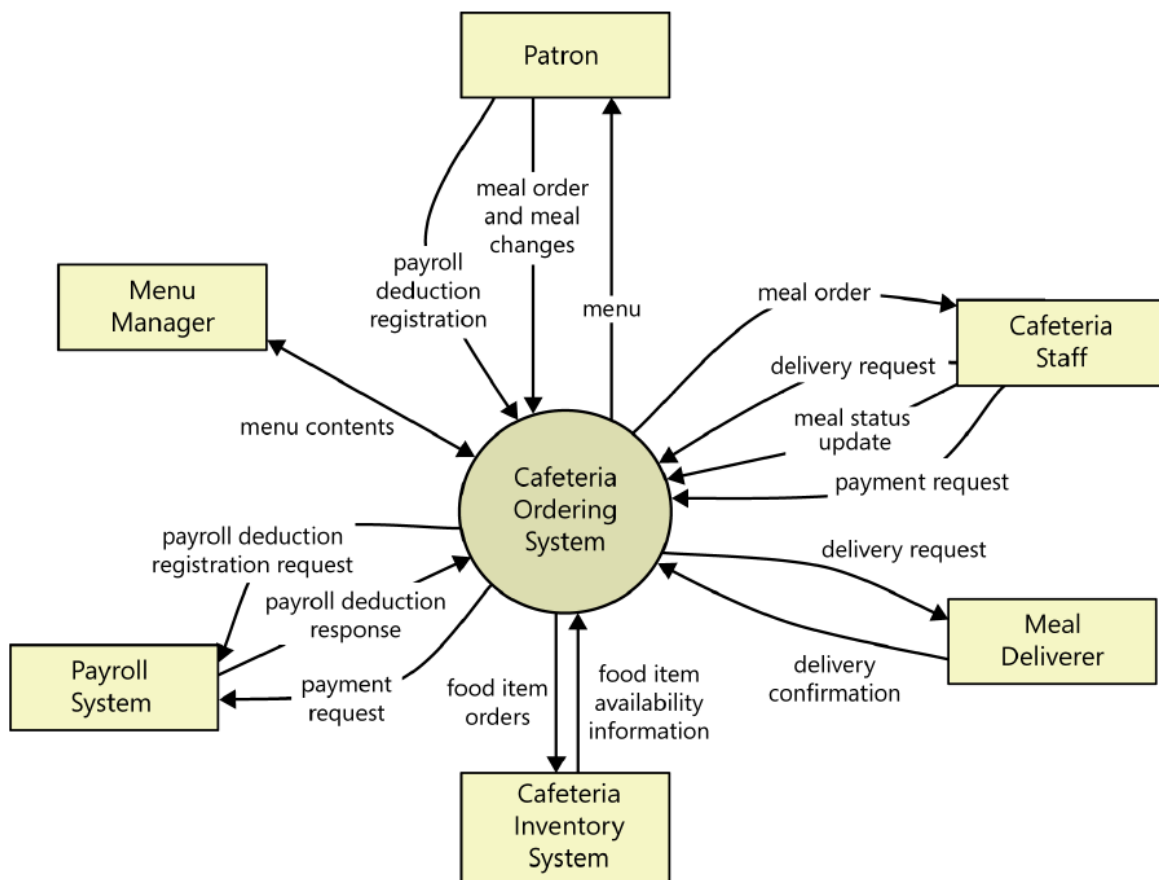


Figure A-1: Context diagram of the Cafeteria Ordering System.

Example: User Classes and Characteristics

Table A-1 Shows user classes and character

User class	Description
Patron	A Patron is a Process Impact employee who wants to order meals to be delivered from the company cafeteria. There are about 600 potential Patrons, of which 300 are expected to use the COS an average of 5 times per week each. Patrons will sometimes order multiple meals for group events or guests. An estimated 60 percent of orders will be placed using the corporate intranet, with 40 percent of orders being placed from home or by smartphone or tablet apps.
Cafeteria Staff	The Process Impact cafeteria employs about 20 Cafeteria Staff who will receive orders from the COS, prepare meals, package them for delivery, and request delivery. Most of the Cafeteria Staff will need training in the use of the hardware and software for the COS.
Menu Manager	The Menu Manager is a cafeteria employee who establishes and maintains daily menus of the food items available from the cafeteria. Some menu items may not be available for delivery. The Menu Manager will also define the cafeteria's daily specials. The Menu Manager will need to edit existing menus periodically.
Meal Deliverer	As the Cafeteria Staff prepare orders for delivery, they will issue delivery requests to a Meal Deliverer's smartphone. The Meal Deliverer will pick up the food and deliver it to the Patron. A Meal Deliverer's other interactions with the COS will be to confirm that a meal was (or was not) delivered.

Example: Use case Diagram

Following use case diagram is of an appointment system in which all the use case diagram relationships are presented. In further in **Table A-2** to the detail of use case diagram syntax is provided.

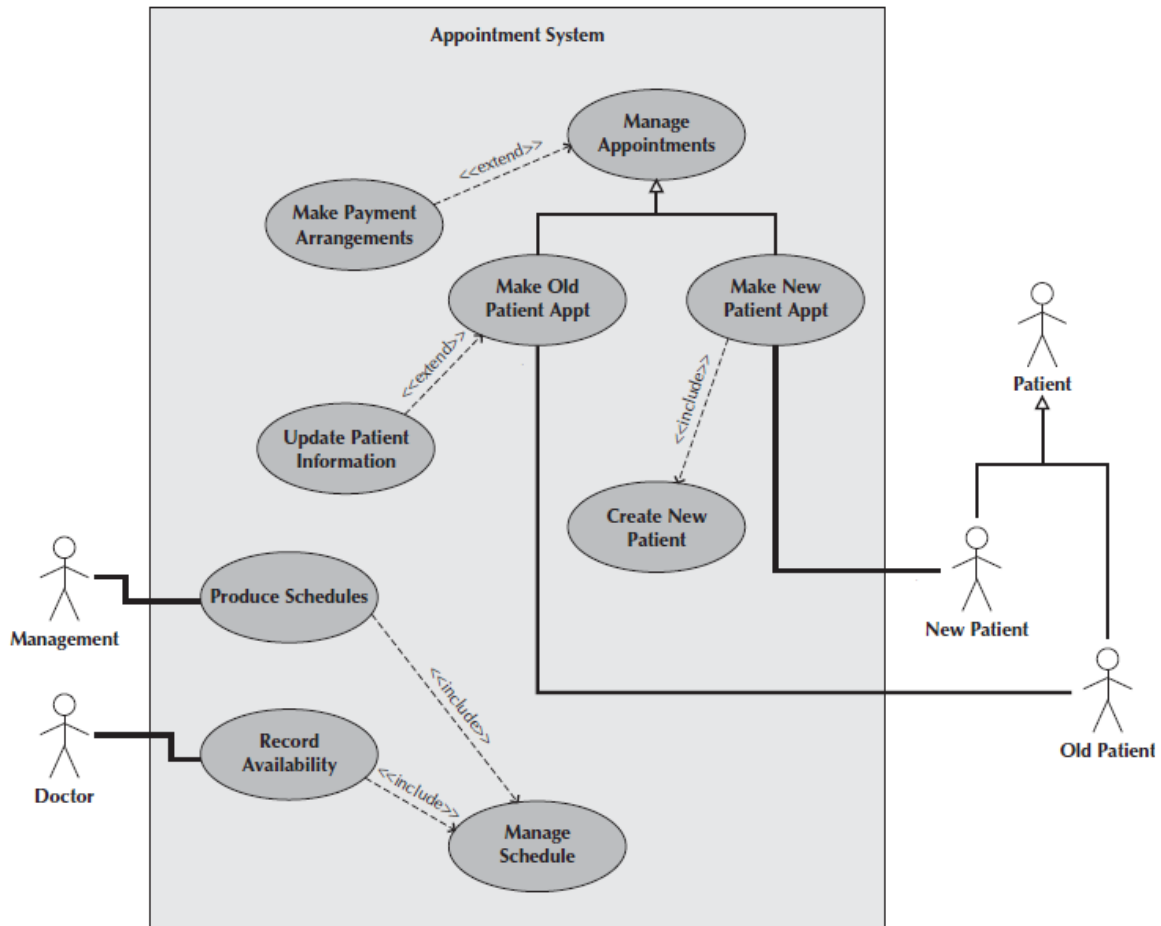

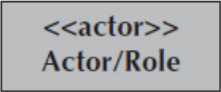
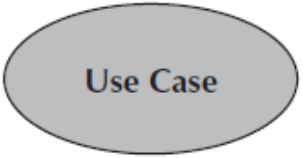
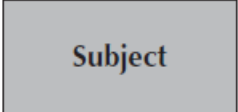

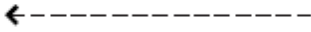
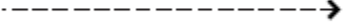



Figure A-2: Use Case Diagram of an Appointment System

Syntax for Use case Diagram

Table A-2 Syntax for Use case Diagram

<p>An actor:</p> <ul style="list-style-type: none"> ■ Is a person or system that derives benefit from and is external to the subject. ■ Is depicted as either a stick figure (default) or, if a nonhuman actor is involved, a rectangle with <<actor>> in it (alternative). ■ Is labeled with its role. ■ Can be associated with other actors using a specialization/superclass association, denoted by an arrow with a hollow arrowhead. ■ Is placed outside the subject boundary. 	 <p>Actor/Role</p> 
<p>A use case:</p> <ul style="list-style-type: none"> ■ Represents a major piece of system functionality. ■ Can extend another use case. ■ Can include another use case. ■ Is placed inside the system boundary. ■ Is labeled with a descriptive verb–noun phrase. 	
<p>subject boundary:</p> <ul style="list-style-type: none"> ■ Includes the name of the subject inside or on top. ■ Represents the scope of the subject, e.g., a system or an individual business process. 	
<p>An association relationship:</p> <ul style="list-style-type: none"> ■ Links an actor with the use case(s) with which it interacts. 	
<p>An include relationship:</p> <ul style="list-style-type: none"> ■ Represents the inclusion of the functionality of one use case within another. ■ Has an arrow drawn from the base use case to the used use case. 	<p><<include>></p> 
<p>An extend relationship:</p> <ul style="list-style-type: none"> ■ Represents the extension of the use case to include optional behavior. ■ Has an arrow drawn from the extension use case to the base use case. 	<p><<extend>></p> 
<p>A generalization relationship:</p> <ul style="list-style-type: none"> ■ Represents a specialized use case to a more generalized one. ■ Has an arrow drawn from the specialized use case to the base use case. 	

Detail Use Case Example

The **Table A-3** below indicates a comprehensive use case template filled in with an example drawn from the Cafeteria ordering system (COS).

Table A-3 Show the detail use case template and example

Use Case ID:	Enter a unique numeric identifier for the Use Case. e.g. UC-1
Use Case Name:	Enter a short name for the Use Case using an active verb phrase. e.g. Order a Meal
Actors:	[An actor is a person or other entity external to the software system being specified who interacts with the system and performs use cases to accomplish tasks.] e.g. Primary Actor: Patron Secondary Actors: Cafeteria Inventory System
Description:	[Provide a brief description of the reason for and outcome of this use case.] e.g. A Patron accesses the Cafeteria Ordering System from either the corporate intranet or external Internet, views the menu for a specific date, selects food items, and places an order for a meal to be picked up in the cafeteria or delivered to a specified location within a specified 15-minute time window.
Trigger:	[Identify the event that initiates the use case.]e.g. A Patron indicates that he wants to order a meal.
Preconditions:	[List any activities that must take place, or any conditions that must be true, before the use case can be started. PRE-1. Patron is logged into COS. PRE-2. Patron is registered for meal payments by payroll deduction.
Postconditions:	[Describe the state of the system at the conclusion of the use case execution. POST-1. Meal order is stored in COS with a status of "Accepted." POST-2. Inventory of available food items is updated to reflect items in this order. POST-3. Remaining delivery capacity for the requested time window is updated.
Normal Flow:	[Provide a detailed description of the user actions and system responses that will take place during execution of the use case under normal, expected conditions. 1.0 Order a Single Meal 1. Patron asks to view menu for a specific date. (see 1.0. E1, 1.0.E2) 2. COS displays menu of available food items and the daily special. 3. Patron selects one or more food items from menu. (see 1.1) 4. Patron indicates that meal order is complete. (see 1.2) 5. COS displays ordered menu items, individual prices, and total price, including taxes and delivery charge. 6. Patron either confirms meal order (continue normal flow) or requests to modify meal order (return to step 2). 7. COS displays available delivery times for the delivery date. 8. Patron selects a delivery time and specifies the delivery location. 9. Patron specifies payment method. 10. COS confirms acceptance of the order. 11. COS sends Patron an email message confirming order details, price, and delivery instructions. 12. COS stores order, sends food item information to Cafeteria Inventory System, and updates available delivery times.

Alternative Flows:	<p>[Document legitimate branches from the main flow to handle special conditions (also known as extensions). For each alternative flow reference the branching step number of the normal flow and the condition which must be true for this extension to be executed. e.g.</p> <p>1.1 Order multiple identical meals</p> <p>1. Patron requests a specified number of identical meals. (see 1.1. E1)</p> <p>2. Return to step 4 of normal flow.</p> <p>1.2 Order multiple meals</p> <p>1. Patron asks to order another meal.</p> <p>2. Return to step 1 of normal flow.</p> <p>Note: Insert a new row for each distinctive alternative flow.]</p>
Exceptions:	<p>1.0. E1 Requested date is today and current time is after today's order cutoff time</p> <p>1. COS informs Patron that it's too late to place an order for today.</p> <p>2a. If Patron cancels the meal ordering process, then COS terminates use case.</p> <p>2b. Else if Patron requests another date, then COS restarts use case.</p> <p>1.0. E2 No delivery times left</p> <p>1. COS informs Patron that no delivery times are available for the meal date.</p> <p>2a. If Patron cancels the meal ordering process, then COS terminates use case.</p> <p>2b. Else if Patron requests to pick the order up at the cafeteria, then continue with normal flow, but skip steps 7 and 8.</p> <p>1.1. E1 Insufficient inventory to fulfill multiple meal order</p> <p>1. COS informs Patron of the maximum number of identical meals he can order, based on current available inventory.</p> <p>2a. If Patron modifies number of meals ordered, then return to step 4 of normal flow.</p> <p>2b. Else if Patron cancels the meal ordering process, then COS terminates use case.</p>
Business Rules	<p>Use cases and business rules are intertwined. Some business rules constrain which roles can perform all or parts of a use case. Perhaps only users who have certain privilege levels can perform specific alternative flows. That is, the rule might impose preconditions that the system must test before letting the user proceed. Business rules can influence specific steps in the normal flow by defining valid input values or dictating how computations are to be performed e.g.</p> <p>BR-1 Delivery time windows are 15 minutes, beginning on each quarter hour.</p> <p>BR-2 Deliveries must be completed between 11:00 A.M. and 2:00 P.M. local time, inclusive.</p> <p>Note: If you are maintaining the business rule in a separate table in SRS then only mention here their IDs.</p>
Assumptions:	<p>[List any assumptions.</p> <p>1. e.g. Assume that 15 percent of Patrons will order the daily special (Source: previous 6 months of cafeteria data).</p>

Event-Response Tables

In order to develop Event Response table first it is required to identify all possible events. Following is an example of events list and event-response table of highway intersection system.

The highway intersection system described earlier has to deal with various events, including these:

- A sensor detects a car approaching in one of the through lanes.
- A sensor detects a car approaching in a left-turn lane.
- A pedestrian presses a button to request to cross a street.
- One of many timers counts down to zero.

Table A-4 presents a fragment of what an event-response table might look like for such a system. Each expected system behavior consists of a combination of event, system state, and response.

Note: You may add more information to event-response table in order to perform detail requirement analysis which includes:

- The event frequency (how many times the event takes place in a given time period, or a limit to how many times it can occur).
- Data elements that are needed to process the event.
- The state of the system after the event responses is executed.

Table A-4 Partial Event-Response Table for a Highway Intersection

Event	System State	Response
Road sensor detects vehicle entering left-turn lane.	Left-turn signal is red. Cross-traffic signal is green.	Start green-to-amber countdown timer for cross-traffic signal.
Green-to-amber countdown timer reaches zero.	Cross-traffic signal is green.	1. Turn cross-traffic signal amber. 2. Start amber-to-red countdown timer.
Amber-to-red countdown timer reaches zero.	Cross-traffic signal is amber.	1. Turn cross-traffic signal red. 2. Wait 1 second. 3. Turn left-turn signal green. 4. Start left-turn-signal countdown timer.
Pedestrian presses a specific walk-request button.	Pedestrian sign is solid Don't Walk. Walk-request countdown timer is not activated.	Start walk-request countdown timer.
Pedestrian presses walk-request button.	Pedestrian sign is solid Don't Walk. Walk-request countdown timer is activated.	Do nothing.
Walk-request countdown timer reaches	Traffic signal in walk direction is green.	Change all green traffic signals to amber.

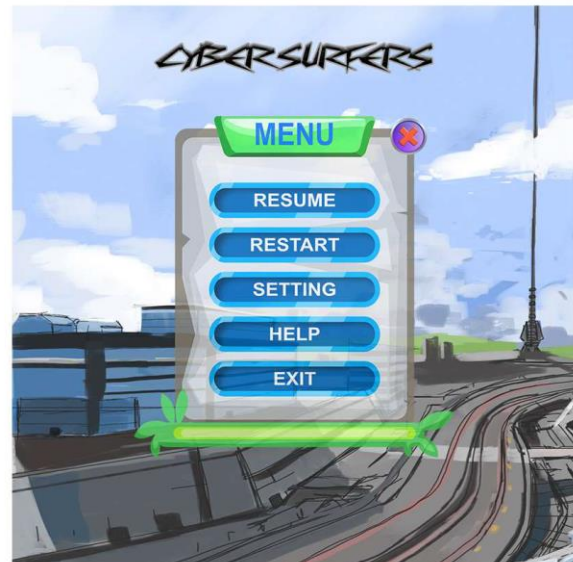
zero plus the amber display time.		
Walk-request countdown timer reaches zero.	Traffic signal in walk direction is amber.	<ol style="list-style-type: none"> 1. Change all amber traffic signals to red. 2. Wait 1 second. 3. Set pedestrian sign to Walk. 4. Start don't-walk countdown timer.

Story Boarding

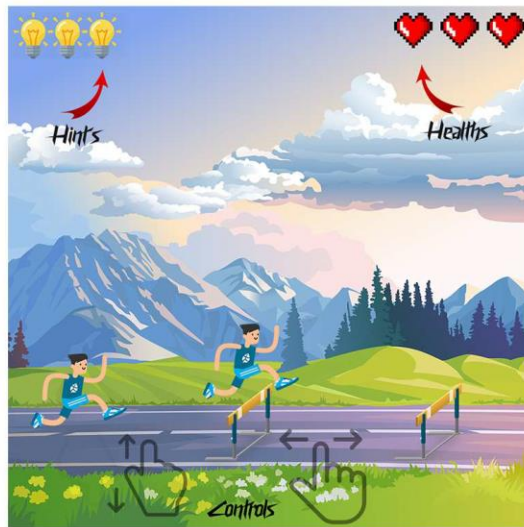
Following is an example of story boarding which is use to identify and analyze graphically intensive applications.



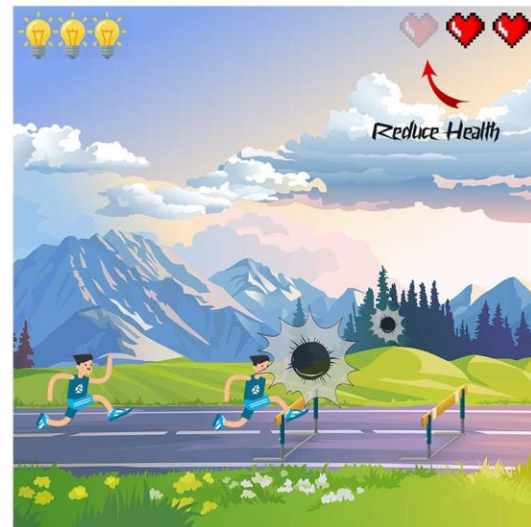
1- The main intro animation text of the game will be played



2- The main menu will appear once all the assets are load. The player can start the game, see the settings and leaderboard.



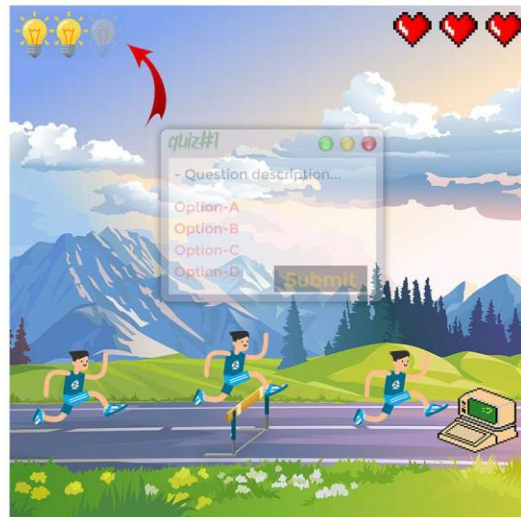
3- Gameplay consists of a running character and hurdles on the way, the player movement will be controlled using swipes on the screen.



4- If the player collides with a static hurdle, the main health will be reduced.



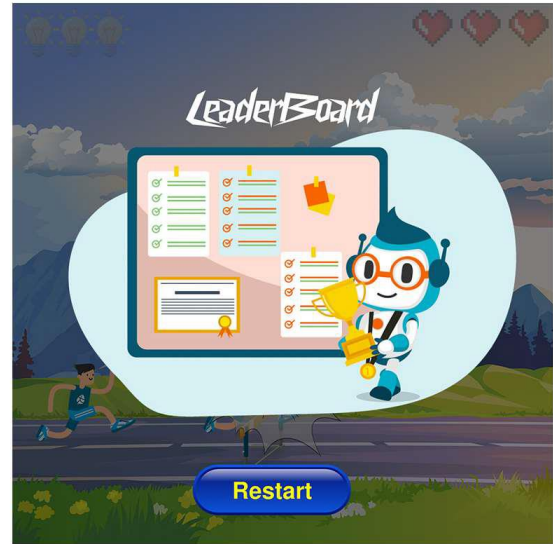
5- There will be a quiz system that will pop up once the player hits a special hurdle in the game.



6- If the player failed to solve the quiz, the hint power will be reduced, all hint powers gone means one health reduced.



6- Once there are no healths remaining and the player hits an hurdle, the dying animation will be played and the game will be over.



7- The leaderboard will be shown at the end that will provide the summary of the quizzes attempted.

Appendix B

Box-and-line diagram

Box-and-line diagrams are often used to describe the business concepts and processes during the analysis phase of the software development lifecycle. These diagrams come with descriptions of components and connectors, as well as other descriptions that provide common inherent interpretations.

Example:

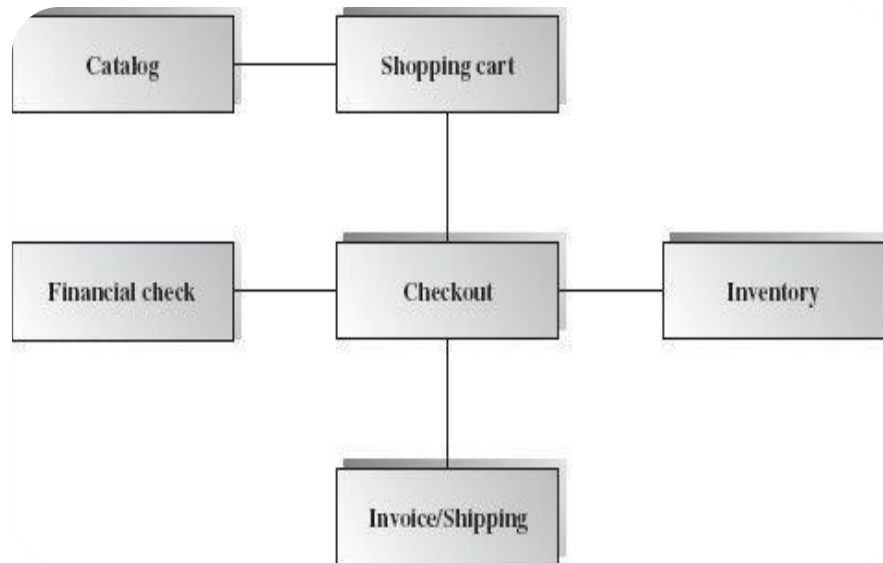


Figure B-3 Box-and-Line Diagram for an Online Shopping Business

- Lines in the box-and-line diagrams indicate the relationship among components
- The semantic of lines may refer dependency, control flow, data flow, etc
- Lines may be associated with arrows to indicate the process direction and sequence.
- A box-and-line diagram can be used as a business concept diagram describing its application domain and process concepts

Example of Architecture Pattern:

The **figure B-2** shows an example of the logical package organization of the layered architecture. The top level deals with user interface, the next level is for utilities, and the one below utility provides core services. Each layer gets support from its lower adjacent layer by an interface implementation and from the related classes in the same layer.

A simple software system may consist of two layers: an interaction layer and a processing layer:

- The interaction layer provides user interfaces to clients, takes requests, validates and forwards requests to the processing layer for processing, and responds to clients.
- The processing layer receives the forwarded requests and performs the business logic process, accesses the database, returns the results to its upper layer, and lets the upper layer respond to clients since the upper layer has the GUI interface responsibility.

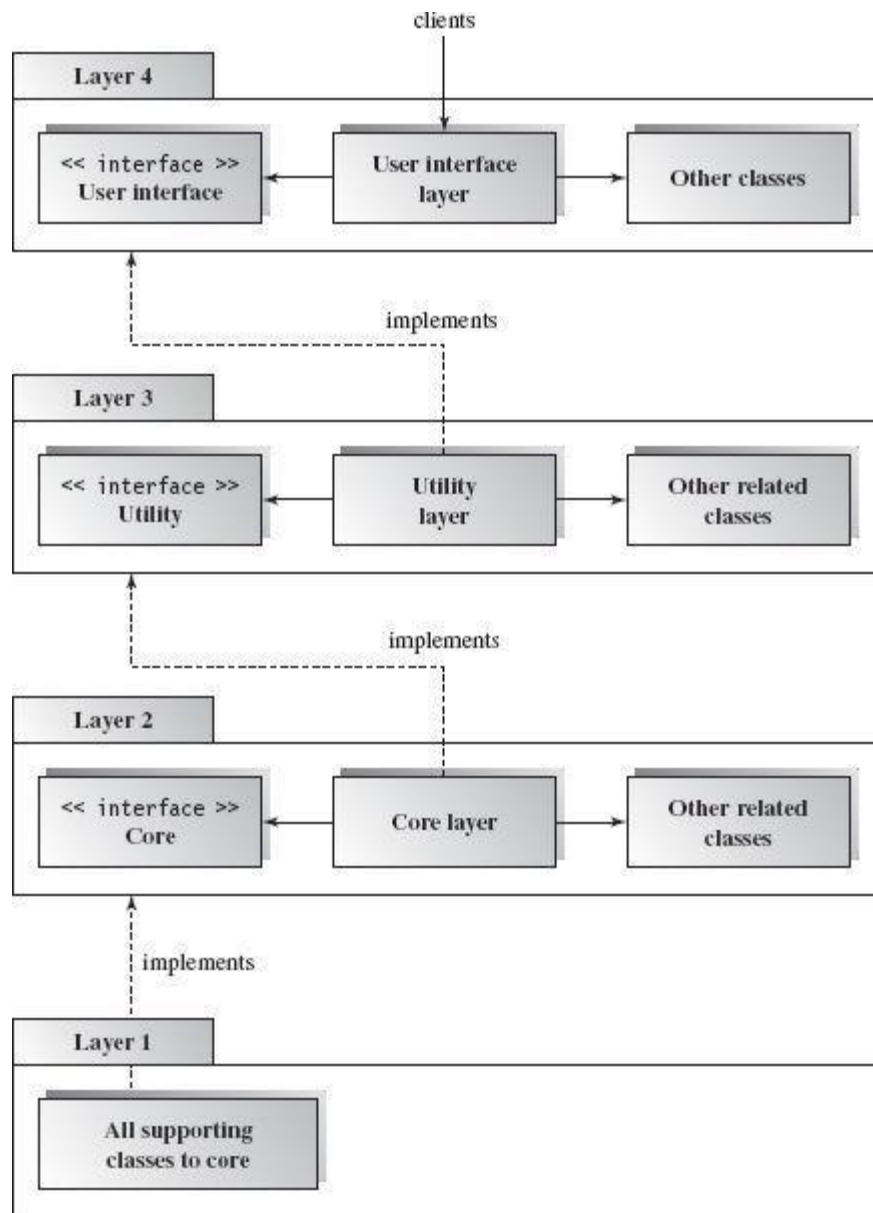


Figure B-4 Component-based Layered Architecture

Note: The Architecture pattern shall be selected according to the targeted system's requirements and quality attributes. Above example is provided to demonstrate that how the system architecture is required to be presented.

Appendix C

Design Models

Activity Diagram

Following activity diagram is of an appointment system presenting **make an appointment** process in which all diagram's elements are presented. In further in **Table C-1** to the detail of activity diagram syntax is provided.

Example

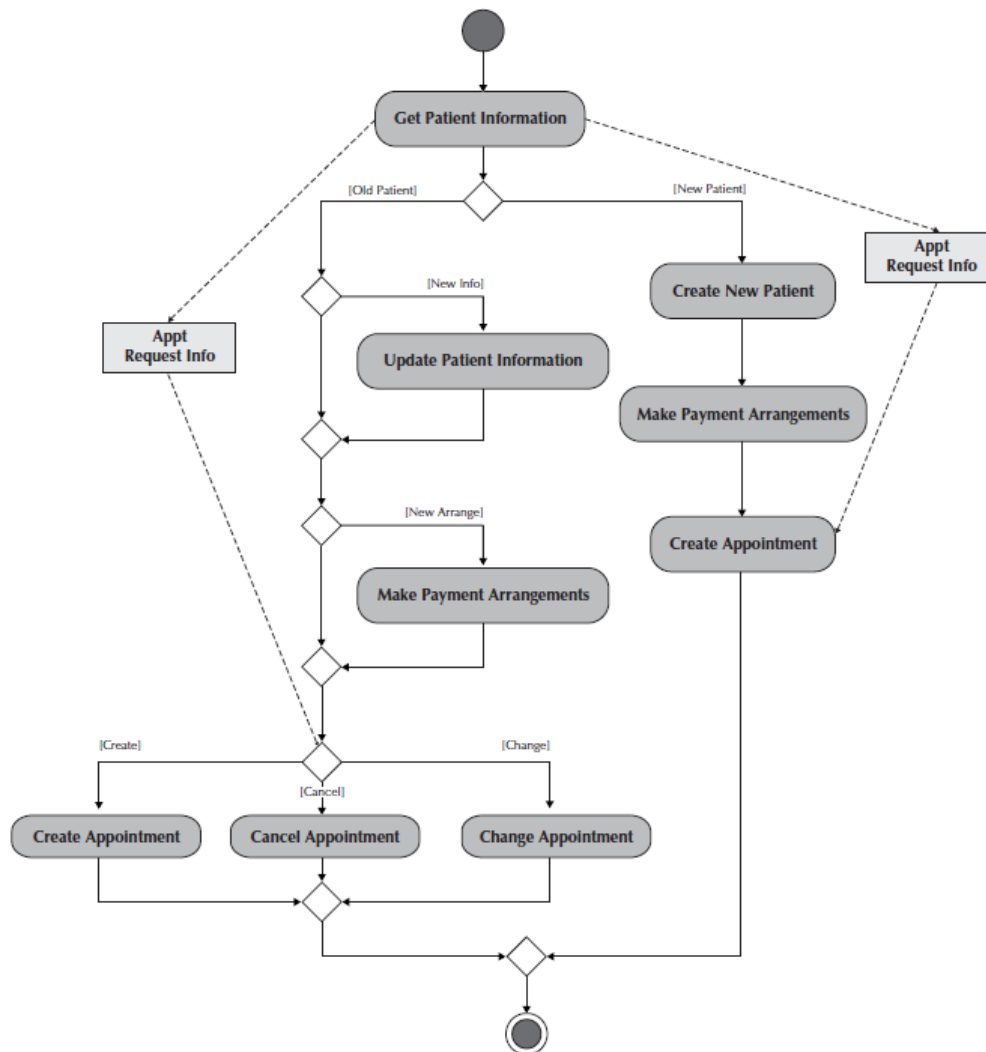

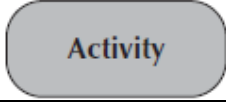
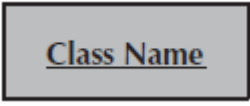

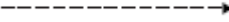



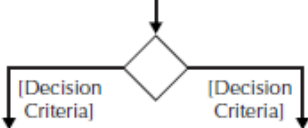
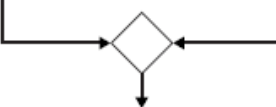


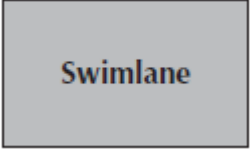


Figure C-1 Activity Diagram for Make an Appointment Process

Activity Diagram Syntax

Table C-1 Activity Diagram Syntax

Term and definition	Symbol
An action: <ul style="list-style-type: none"> Is a simple, non-decomposable piece of behavior. Is labeled by its name. 	
An activity: <ul style="list-style-type: none"> Is used to represent a set of actions. Is labeled by its name. 	
An object node: <ul style="list-style-type: none"> Is used to represent an object that is connected to a set of object flows. Is labeled by its class name. 	
A control flow: <ul style="list-style-type: none"> Shows the sequence of execution. 	
An object flow: <ul style="list-style-type: none"> Shows the flow of an object from one activity (or action) to another activity (or action). 	
An initial node: <ul style="list-style-type: none"> Portrays the beginning of a set of actions or activities. 	
A final-activity node: <ul style="list-style-type: none"> Is used to stop all control flows and object flows in an activity (or action). 	
A final-flow node: <ul style="list-style-type: none"> Is used to stop a specific control flow or object flow. 	
A decision node: <ul style="list-style-type: none"> Is used to represent a test condition to ensure that the control flow or object flow only goes down one path. Is labeled with the decision criteria to continue down the specific path. 	
A merge node: <ul style="list-style-type: none"> Is used to bring back together different decision paths that were created using a decision node. 	
A fork node: <ul style="list-style-type: none"> Is used to split behavior into a set of parallel or concurrent flows of activities (or action) 	
A join node: <ul style="list-style-type: none"> Is used to bring back together a set of parallel or concurrent flows of activities (or action) 	
A swimlane: <ul style="list-style-type: none"> Is used to break up an activity diagram into rows and columns to assign the individual activities (or actions) to the individuals or objects that are responsible for executing the activity (or action) Is labeled with the name of the individual or object responsible 	

Class Diagram

Following class diagram is of an appointment system in which all class diagrams elements are presented. In further in **Table C-2** to the detail of class diagram syntax is provided.

Example

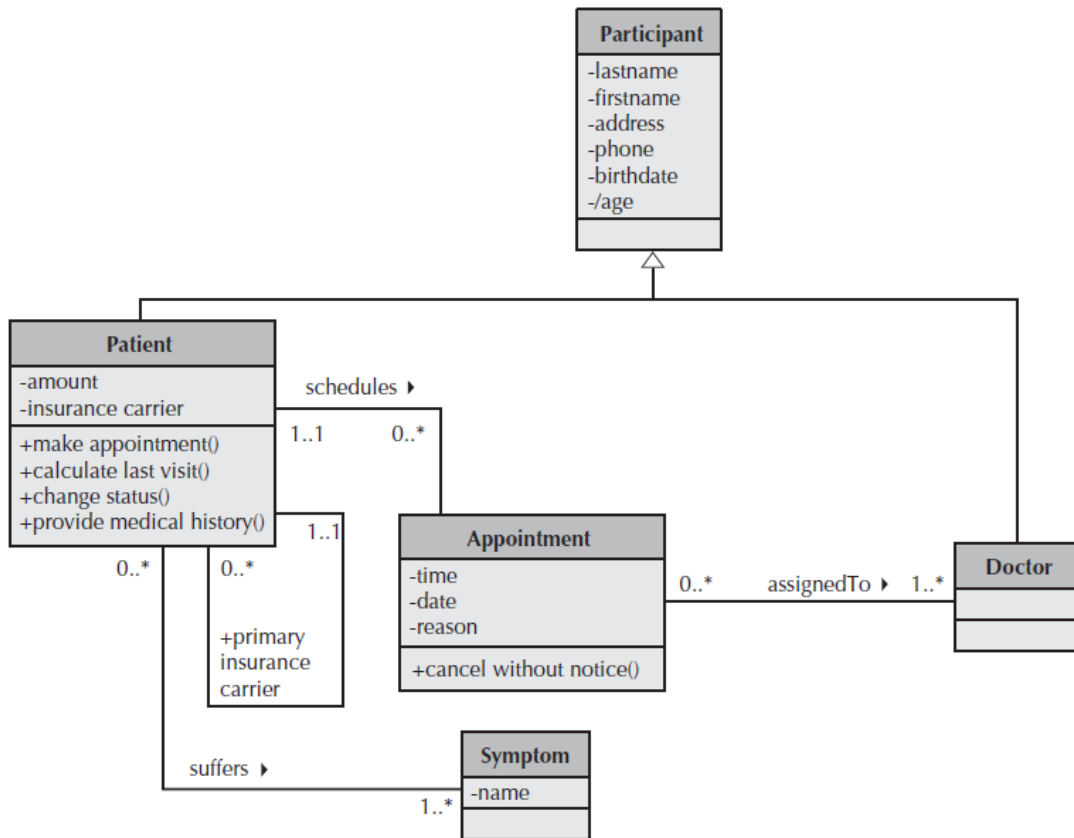
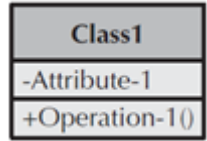
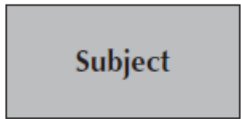






Figure C-2 Class Diagram for an Appointment System

Class Diagram Syntax

Table C- 2 Class Diagram Syntax

Term and definition	Symbol
A class: <ul style="list-style-type: none"> Has a name typed in bold and centered in its top compartment. Has a list of attributes in its middle compartment. Represents a kind of person, place, or thing about which the system will need to capture and store information. Has a list of operations in its bottom compartment. Does not explicitly show operations that are available to all classes. 	
An attribute: <ul style="list-style-type: none"> Represents properties that describe the state of an object. Can be derived from other attributes, shown by placing a slash before the attribute's name. 	
An operation: <ul style="list-style-type: none"> Represents the actions or functions that a class can perform. Can be classified as a constructor, query, or update operation. Includes parentheses that may contain parameters or information needed to perform the operation. 	<p>operation name ()</p>
An association: <ul style="list-style-type: none"> Represents a relationship between multiple classes or a class and itself. Is labeled using a verb phrase or a role name, whichever better represents the relationship. Can exist between one or more classes. Contains multiplicity symbols, which represent the minimum and maximum times a class instance can be associated with the related class instance. 	
A generalization: <ul style="list-style-type: none"> Represents a-kind-of relationship between multiple classes. 	
An aggregation: <ul style="list-style-type: none"> Represents a logical a-part-of relationship between multiple classes or a class and itself. Is a special form of an association. 	
A composition: <ul style="list-style-type: none"> Represents a physical a-part-of relationship between multiple classes or a class and itself. Is a special form of an association. 	

Sequence Diagram

Following example shows an instance sequence diagram that depicts the objects and messages for the Make Old Patient Appt use case, which describes the process by which an existing patient creates a new appointment or cancels or reschedules an appointment. In further in **Table C-3** to the detail of class diagram syntax is provided.

Example

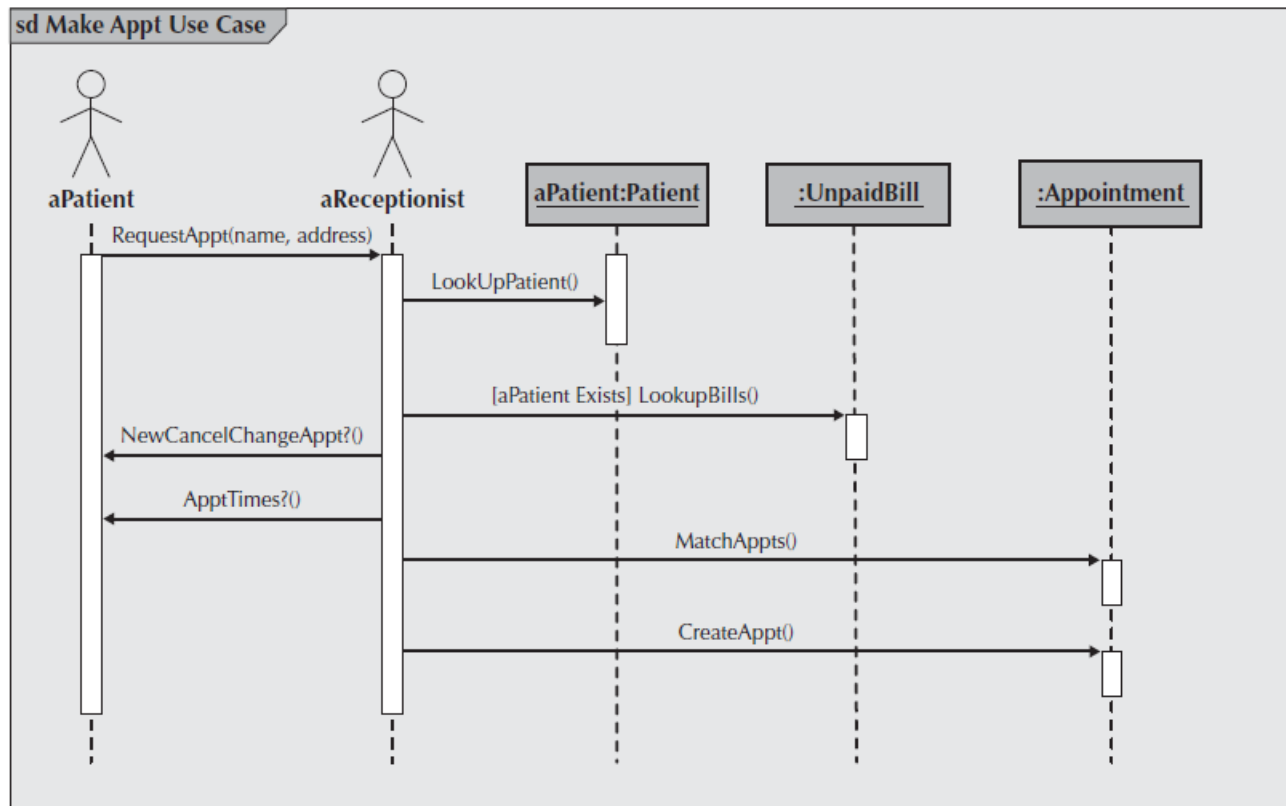

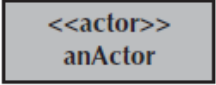
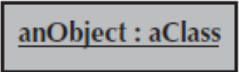



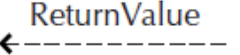


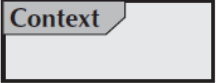


Figure C-3 Example Sequence Diagram

Sequence diagram Syntax

Table C-3 Sequence Diagram Syntax

Term and definition	Symbol
An actor: <ul style="list-style-type: none"> Is a person or system that derives benefit from and is external to the system. Participates in a sequence by sending and/or receiving messages. Is placed across the top of the diagram. Is depicted either as a stick figure (default) or, if a nonhuman actor is involved, as a rectangle with <<actor>> in it (alternative). 	 anActor 
An object: <ul style="list-style-type: none"> Participates in a sequence by sending and/or receiving messages. Is placed across the top of the diagram. 	
A lifeline: <ul style="list-style-type: none"> Denotes the life of an object during a sequence. Contains an X at the point at which the class no longer interacts. 	
An execution occurrence: <ul style="list-style-type: none"> Is a long narrow rectangle placed atop a lifeline. Denotes when an object is sending or receiving messages. 	
A message: <ul style="list-style-type: none"> Conveys information from one object to another one. An operation call is labeled with the message being sent and a solid arrow, whereas a return is labeled with the value being returned and shown as a dashed arrow. 	 
A guard condition: <ul style="list-style-type: none"> Represents a test that must be met for the message to be sent. 	
For object destruction: <ul style="list-style-type: none"> An X is placed at the end of an object's lifeline to show that it is going out of existence. 	
A frame: <ul style="list-style-type: none"> Indicates the context of the sequence diagram. 	

Behavioral State Machine Diagram

Following example of a behavioral state machine representing the patient class in the context of a hospital environment. From this diagram, we can tell that a patient enters a hospital and is admitted after checking in. If a doctor finds the patient to be healthy, he or she is released and is no longer considered a patient after two weeks elapse. If a patient is found to be unhealthy, he or she remains under observation until the diagnosis changes.

Example

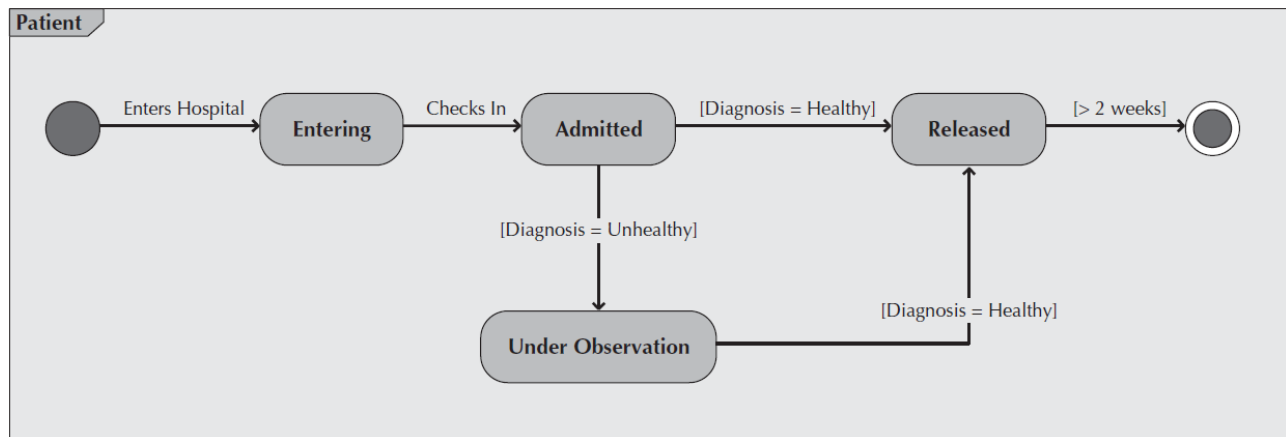

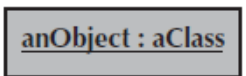



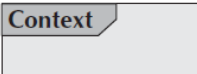


Figure C-4 Sample Behavioral State Machine Diagram

Behavioral State Machine Diagram Syntax

Table C-4 Behavioral State Machine Diagram Syntax

Term and definition	Symbol
A state: <ul style="list-style-type: none"> Is shown as a rectangle with rounded corners. Has a name that represents the state of an object. 	
An initial state: <ul style="list-style-type: none"> Is shown as a small, filled-in circle. Represents the point at which an object begins to exist. 	
A final state: <ul style="list-style-type: none"> Is shown as a circle surrounding a small, filled-in circle (bull's-eye). Represents the completion of activity. 	
An event: <ul style="list-style-type: none"> Is a noteworthy occurrence that triggers a change in state. Can be a designated condition becoming true, the receipt of an explicit signal from one object to another, or the passage of a designated period of time. Is used to label a transition. 	
A transition: <ul style="list-style-type: none"> Indicates that an object in the first state will enter the second state. Is triggered by the occurrence of the event labeling the transition. Is shown as a solid arrow from one state to another, labeled by the event name. 	
A frame: <ul style="list-style-type: none"> Indicates the context of the behavioral state machine. 	

Data Flow Diagram

Data flow diagram symbols, symbol names, and examples

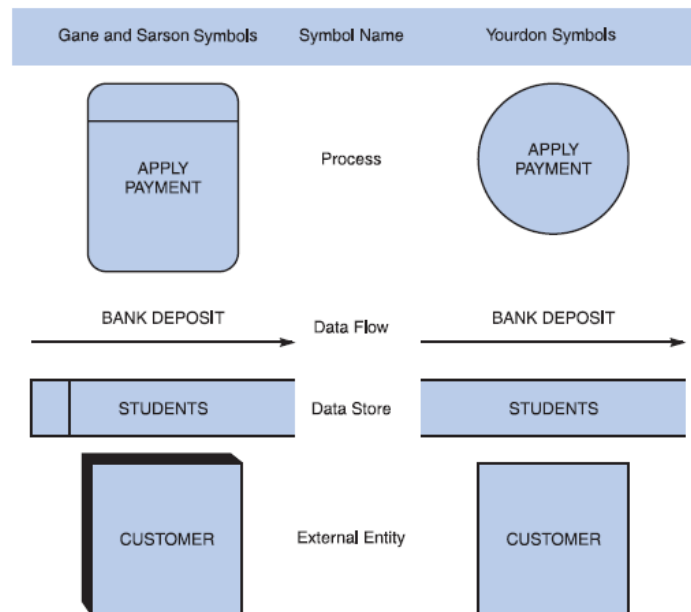


Figure C-5 Data flow diagram symbols, symbol names, and examples of the Gane and Sarson and Yourdon symbol sets.

Guidelines for Drawing DFDs

Step 1: Draw a Context Diagram: The first step in constructing a set of DFDs is to draw a context diagram. A **context diagram** is a top-level view of an information system that shows the system's boundaries and scope. Data stores are not shown in the context diagram because they are contained within the system and remain hidden until more detailed diagrams are created.

Example

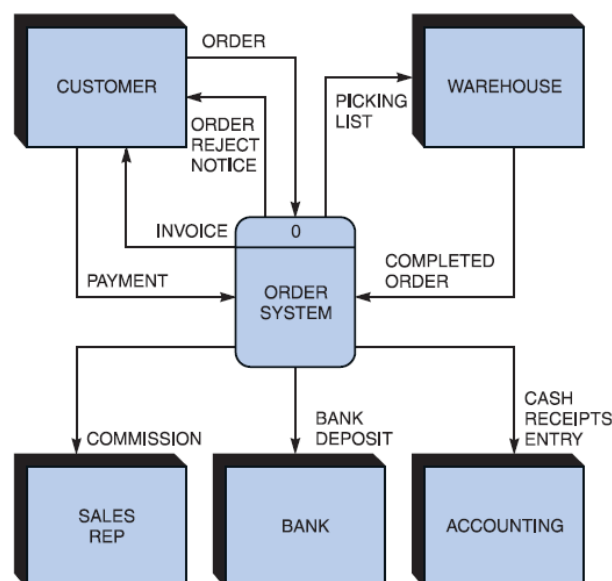


Figure C-6 Context diagram DFD for an order system.

Step 2: Draw a Diagram 0 DFD: To show the detail inside the black box, you create DFD diagram 0. **Diagram 0** zooms in on the system and shows major internal processes, data flows, and data stores. Diagram 0 also repeats the entities and data flows that appear in the context diagram. When you expand the context diagram into DFD diagram 0, you must retain all the connections that flow into and out of process 0.

Example

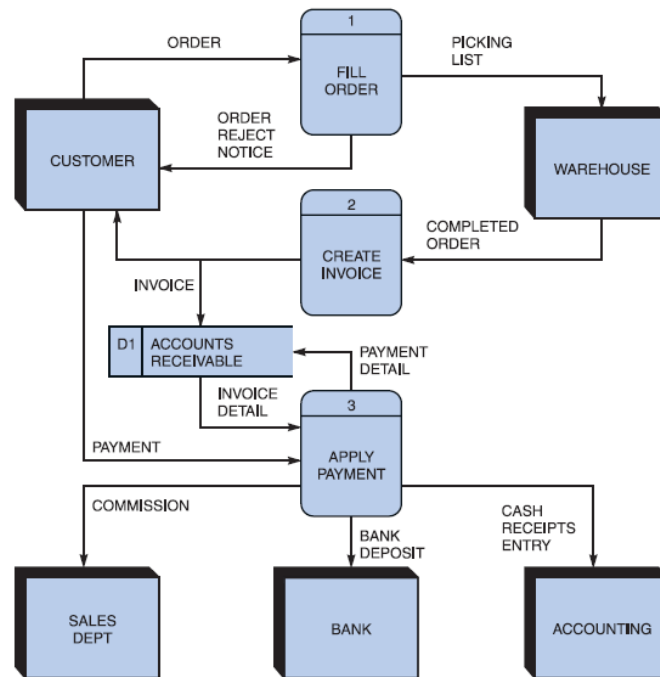


Figure C-7 Diagram 0 DFD for the order system.

Step 3: Draw the Lower-Level Diagrams:

To create lower-level diagrams, you must use leveling and balancing techniques. **Leveling** is the process of drawing a series of increasingly detailed diagrams, until all functional primitives are identified.

Leveling Example

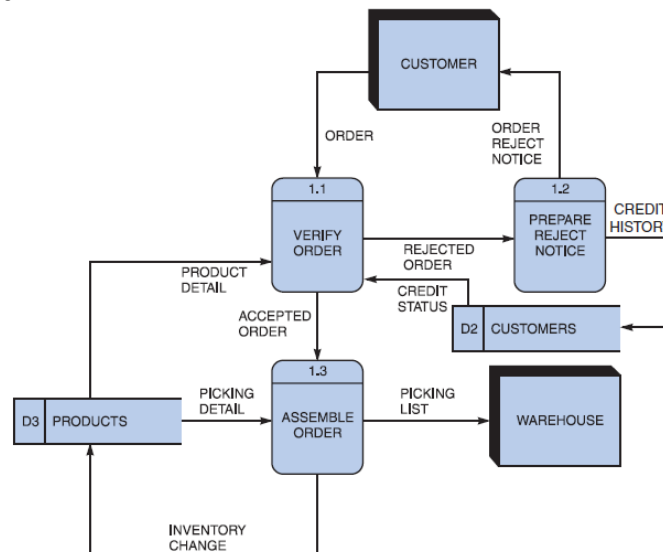
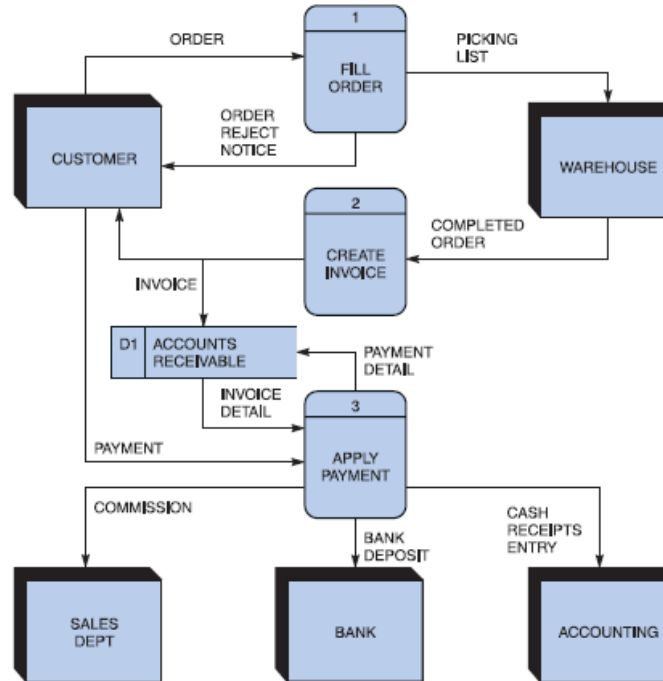


Figure C-8 Diagram 1 DFD shows details of the FILL ORDER process in the order system.

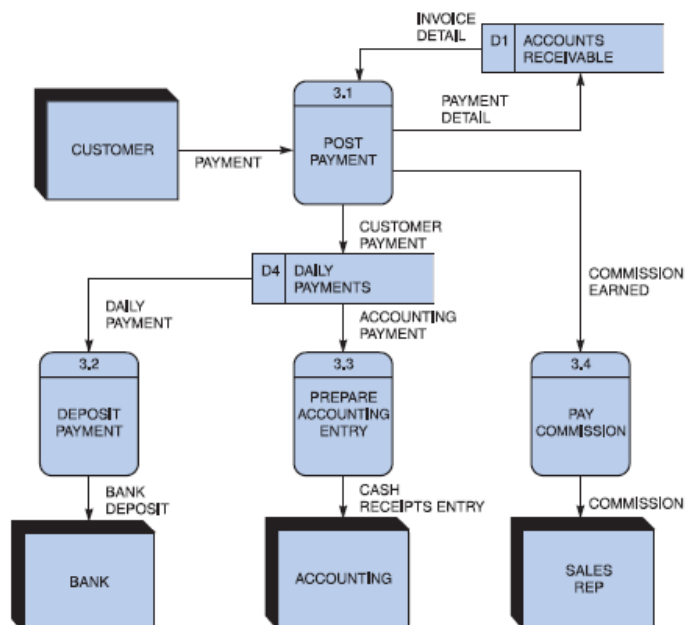
Balancing maintains consistency among a set of DFDs by ensuring that input and output data flows align properly.

Balancing Example

Order System Diagram 0 DFD



Order System Diagram 3 DFD



The order system diagram 0 is shown at the top of the figure and exploded diagram 3 DFD (for the APPLY PAYMENT process) is shown at the bottom. The two DFDs are balanced, because child diagram at the bottom has the same input and output flows as the parent process 3 shown at the top.

Appendix D: General Coding Standards & Guidelines

1. Follow a consistent variable naming convention throughout the code. E.g.
 - Snakecase (words are delimited by “_” like: variable_one)
 - Pascalcase (words are delimited by capital letters like: VariableOne)
 - Camelcase (words are delimited by capital letters except the initial word like: variableOne)
 - Hungarian Notation (describes the variable type or purpose at the start of the variable name like: arrDistributeGroup)
2. Use naming that visually describes scope like privateField, Const etc
3. Use read only/immutable when a field’s value should not be changed after initialization
4. Use only get, for properties that should not be updated from outside
5. Name functions according to their functionalities.
6. Insert appropriate comments to make the code understandable to any reader. Additionally follow a consistent style to do so. E.g.

```
/* the below function will be used for the addition of two variables*/  
int Add(){  
    //logic of the function  
}
```

Avoid commenting on obvious things
7. Make use of indentation for indicating the start and end of the control structures along with a clear specification of where the code is between them.
8. Follow consistent naming convention for files and folders.
9. Follow proper structure for classes
10. Group code entities logically into projects/packages/modules/folders
 - a. Separate logical layers of application into different modules/services/utilities etc.
 - b. User separate files for each class, struct, interface, enum etc. Name of the file and the enclosing entity must be same. E.g., class Employee in Employee.cs/Employee.java
11. Define and use everything within the minimum scope possible
12. Use proper access modifier for all code entities if required
13. Code entities should have maximum cohesion and least coupling possible.
14. Follow DRY law.
 - a. Do not repeat code.
 - b. A piece of knowledge should exist only in one place within the codebase/application
 - c. Reuse code as much as possible
 - d. Always write short methods
 - e. Single method should not have too many logic, long conditional flow or too many parameters
15. Strictly follow Single Responsibility Principle (SRP) when writing methods, classes, modules, projects, packages, or any other code entities.
16. Write classes and other code entities that are easy to extend without modification.
17. Handle exceptions
18. Log exception and other significant event details
19. Follow a consistent convention for logging all over the application

Appendix E

Business rules testing

Methodology for creating decision table:

1. Identify Conditions & Values	Find the data attribute each condition tests and all of the attribute's values.
2. Identify Possible Actions	Determine each independent action to be taken for the decision or policy.
3. Compute Max Number of Rules	Multiply the number of values for each condition data attribute by each other.
4. Enter All Possible Rules	Fill in the values of the condition data attributes in each numbered rule column.
5. Define Actions for each Rule	For each rule, mark the appropriate actions with an X in the decision table.
6. Verify the Policy	Review completed decision table with end-users.
7. Simplify the Table	Eliminate and/or consolidate rules to reduce the number of columns.

Example:

The provided example is of a super store.

SUPERSTORES MARKETING POLICY #1:

- Repeat customers get free shipping.
- New customers pay regular shipping on their first order.

Decision table:

Condition	Rule 1	Rule 2
Repeat customer?	Y	N
Free shipping	Y	N

SUPERSTORES MARKETING POLICY #2:

- Repeat customers get free shipping.
- New customers pay regular shipping on their first order.
- Any customer who uses the SuperStore (SS) charge card gets a 5% discount.

Decision table:

Condition	Rule 1	Rule 2	Rule 3	Rule 4
Repeat customer?	Y	Y	N	N
Used SS card?	Y	N	Y	N
Free shipping				
5% discount				

SUPERSTORES MARKETING POLICY #2:

- Repeat customers get free shipping.
- New customers pay regular shipping on their first order.
- Any customer who uses the SuperStore (SS) charge card gets a 5% discount.

Decision table:

Condition	Rule 1	Rule 2	Rule 3	Rule 4
Repeat customer?	Y	Y	N	N
Used SS card?	Y	N	Y	N
Free shipping	Y	Y	N	N
5% discount	Y	N	Y	N

SUPERSTORES MARKETING POLICY #3:

- Repeat customers get free shipping.
- New customers pay regular shipping on their first order. However, if a new customer's first order is over \$100, shipping is free.
- Any customer who uses the SuperStore (SS) charge card gets a 5% discount.

Decision table:

Condition	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
Repeat customer?	Y	Y	Y	Y	N	N	N	N
Used SS card?	Y	Y	N	N	Y	Y	N	N
Order over \$100?	Y	N	Y	N	Y	N	Y	N
Free shipping								
5% discount								

SUPERSTORES MARKETING POLICY #3:

- Repeat customers get free shipping.
- New customers pay regular shipping on their first order. However, if a new customer's first order is over \$100, shipping is free.
- Any customer who uses the SuperStore (SS) charge card gets a 5% discount.

Decision table:

Condition	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
Repeat customer?	Y	Y	Y	Y	N	N	N	N
Used SS card?	Y	Y	N	N	Y	Y	N	N
Order over \$100?	Y	N	Y	N	Y	N	Y	N
Free shipping	Y	Y	Y	Y	Y	N	Y	N
5% discount	Y	Y	N	N	Y	Y	N	N

- Now Combine rules where it is apparent that an alternative does not make a difference in the outcome

Original Rules

Condition	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
Repeat customer?	Y	Y	Y	Y	N	N	N	N
Used SS card?	Y	Y	N	N	Y	Y	N	N
Order over \$100?	-	-	-	-	Y	N	Y	N
Free shipping	Y	Y	Y	Y	Y	N	Y	N
5% discount	Y	Y	N	N	Y	Y	N	N

Original Rules 1 & 2

Original Rules 3 & 4

Combined Rules

Condition	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
Repeat customer?	Y	Y	N	N	N	N
Used SS card?	Y	N	Y	Y	N	N
Order over \$100?	-	-	Y	N	Y	N
Free shipping	Y	Y	Y	N	Y	N
5% discount	Y	N	Y	Y	N	N

This is the final table and now you have to create test cases on every rule. In above example there are 6 rules so there shall be 6 test cases.