OBJECT-ORIENTED PROGRAMMING

FINAL PROJECT REPORT

LECTURER: Jude Joseph Lamug Martinez MCS

NAME: Viona Djunaidy

STUDENT ID NUMBER: 2702337602

CLASS: L2BC

I. PROJECT SPECIFICATION

PROJECT OVERVIEW

EasySplit is a desktop application designed to help users easily divide bills among a number of participants. It allows users to upload an image of the bill, crop the image, enter the names of participants, customize the distribution of expenses, and obtain the total amount every participant owes. The application will automatically extract the item details and calculate the total amount each participant must pay according to the allocation of items set by the user.

OBJECTIVES

- Create a user-centric application that can simplify the process of splitting bills among people, reducing manual effort and potential errors associated with manual calculations.
- Provide convenience for the users as calculations for each participant's share are automated by the application.
- Encourage financial transparency among participants by providing a clear breakdown of expenses and each participant's contribution to the bill.

FEATURES

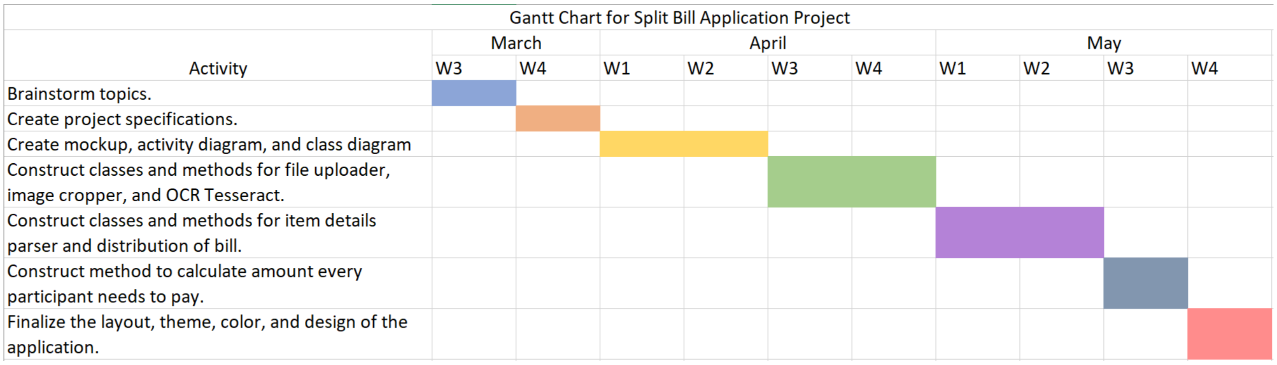
1. Upload Bill Image:
    - Upload an image of the bill through the file uploader.
2. Crop Bill Image:
    - Crop the uploaded image to the relevant part of the bill, including the list of item details including item names, quantities, prices, service charges, and tax.
3. Display the list of item details:
    - The program will scan the text in the bill and extract the item details, service charge, and tax from the bill image, and present them in the application.
4. Input Names of Participants:
    - The user can only enter a minimum of two and a maximum of ten participants to contribute to the split bill.
5. Custom Distribution:

    - The program will present checkboxes of all the items in the bill for every participant. The user can assign items to participants by ticking the checkboxes.
    - The program requires the user to allocate every item to at least one participant to ensure that all items are paid for by at least one person.
    - The program requires the user to assign every participant to at least one item to ensure that all participants contribute to the split bill.
6. Total Amount Calculation:
    - The program can calculate the total amount every participant needs to pay according to the assigned items.
7. Equal Distribution of Bill:
    - Instead of ticking checkboxes for each participant, the user can simply click the “Select All” button to assign all items to all participants, which will ease the process of splitting the bill equally among all participants.
8. Recalculation of Total Amount:
    - For cases when the user changes the distribution of the bill after calculation, the program can recalculate the total amount according to the new distribution.

TECHNOLOGY STACK

- Java FX: A Java library and GUI toolkit used to develop desktop and web applications.
- FXML: A XML-based user interface markup language used to construct the user interface of a JavaFX application.
- OCR Tesseract: An open-source optical character recognition engine used to extract text from images.
- CSS: A rule-based language used to define the styles of specific parts of the application.
- Java Collections Framework: A unified architecture used to represent and manipulate groups of individual objects.

TIMELINE

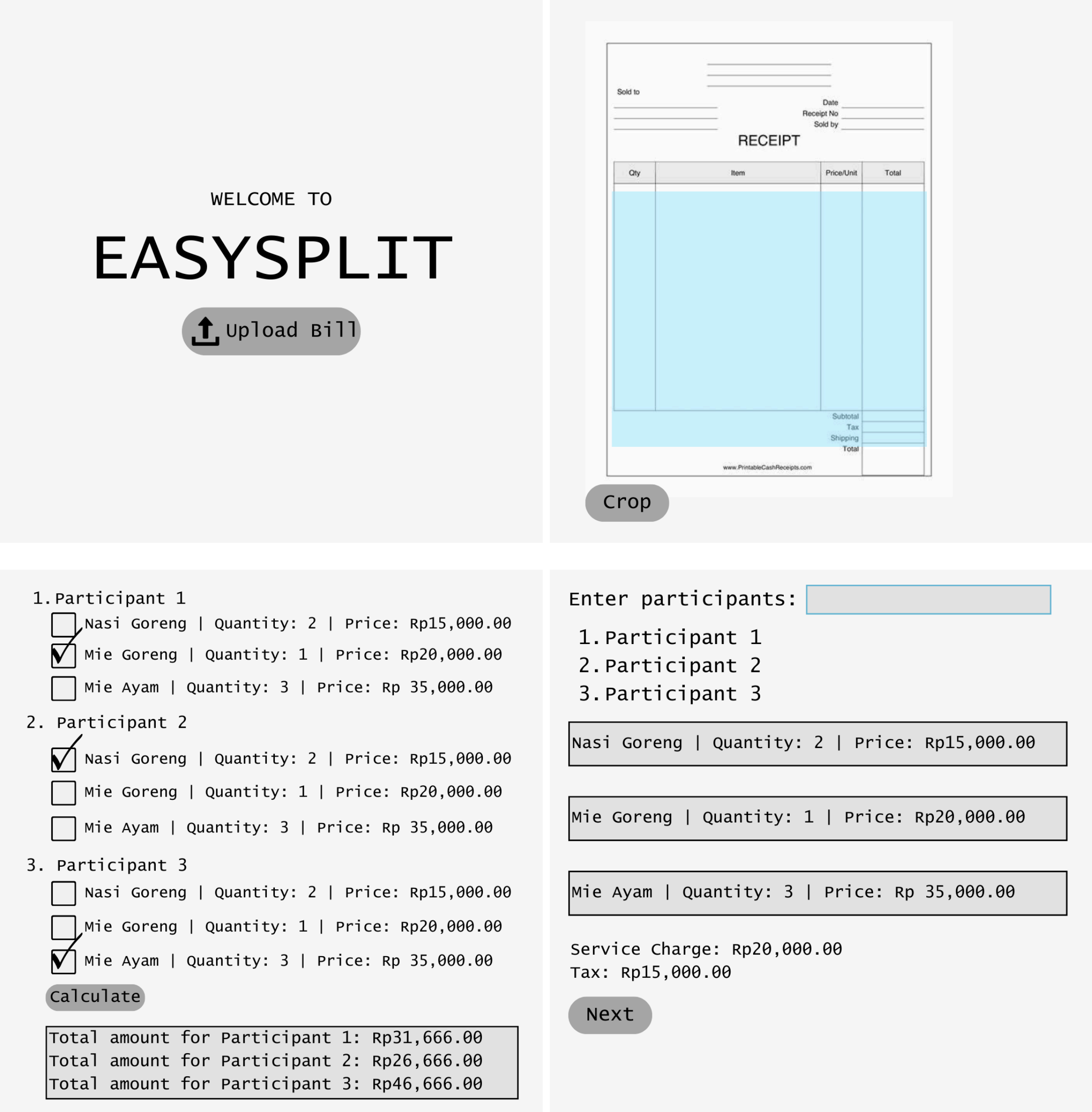


Gantt Chart for Split Bill Application Project

Activity	March		April				May			
	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
Brainstorm topics.	🟦									
Create project specifications.		🟧								
Create mockup, activity diagram, and class diagram			🟨	🟨						
Construct classes and methods for file uploader, image cropper, and OCR Tesseract.					🟩	🟩				
Construct classes and methods for item details parser and distribution of bill.							🟪	🟪		
Construct method to calculate amount every participant needs to pay.									▨	
Finalize the layout, theme, color, and design of the application.										🟥

II. SOLUTION DESIGN

MOCKUP DESIGN



WELCOME TO

EASYSPLIT

Upload Bill

Crop

1. Participant 1
   - □ Nasi Goreng | Quantity: 2 | Price: Rp15,000.00
   - ☑ Mie Goreng | Quantity: 1 | Price: Rp20,000.00
   - □ Mie Ayam | Quantity: 3 | Price: Rp 35,000.00
2. Participant 2
   - ☑ Nasi Goreng | Quantity: 2 | Price: Rp15,000.00
   - □ Mie Goreng | Quantity: 1 | Price: Rp20,000.00
   - □ Mie Ayam | Quantity: 3 | Price: Rp 35,000.00
3. Participant 3
   - □ Nasi Goreng | Quantity: 2 | Price: Rp15,000.00
   - □ Mie Goreng | Quantity: 1 | Price: Rp20,000.00
   - ☑ Mie Ayam | Quantity: 3 | Price: Rp 35,000.00

Calculate

```
Total amount for Participant 1: Rp31,666.00
Total amount for Participant 2: Rp26,666.00
Total amount for Participant 3: Rp46,666.00
```

Enter participants:

1. Participant 1
2. Participant 2
3. Participant 3

Nasi Goreng | Quantity: 2 | Price: Rp15,000.00

Mie Goreng | Quantity: 1 | Price: Rp20,000.00

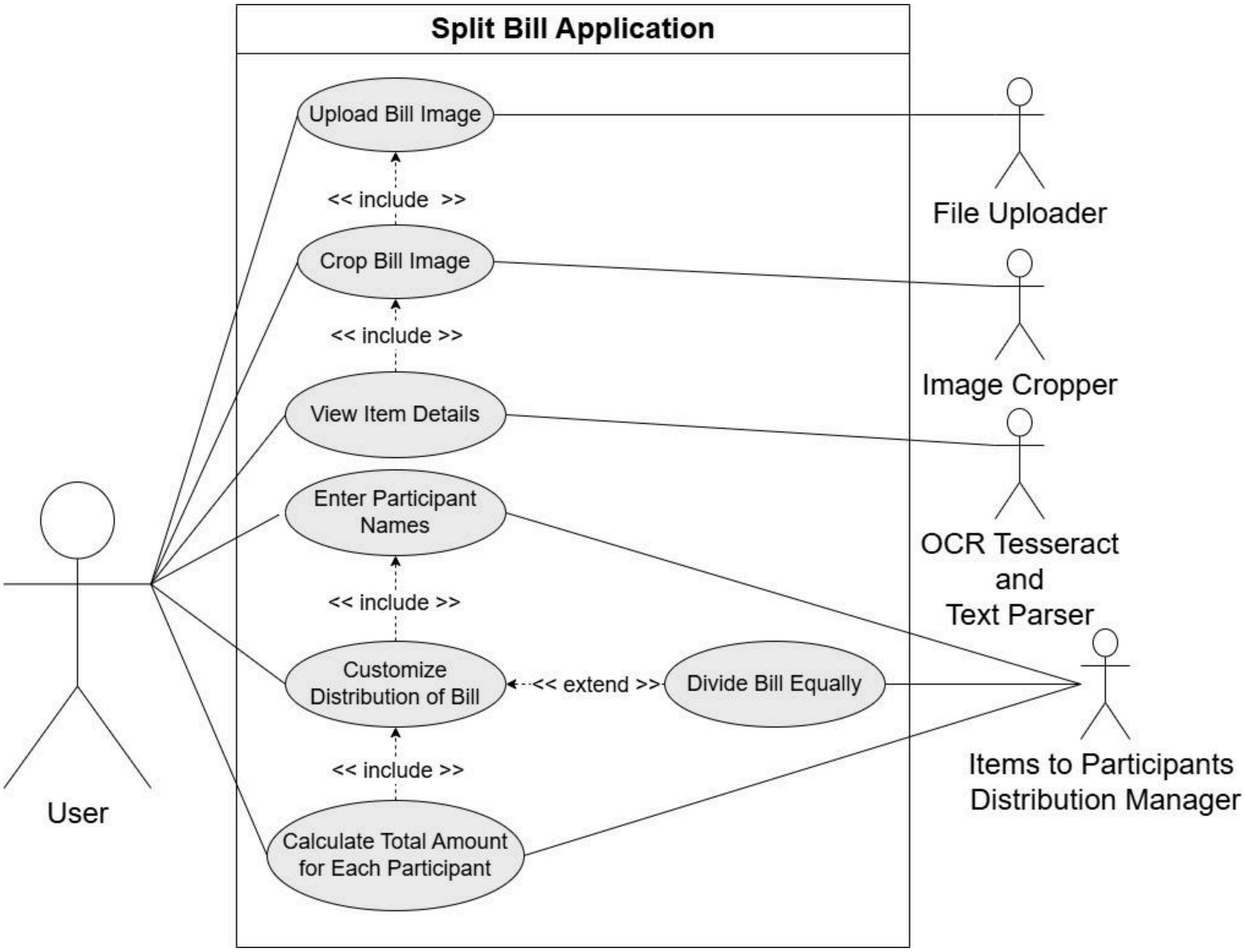
Mie Ayam | Quantity: 3 | Price: Rp 35,000.00

Service Charge: Rp20,000.00
Tax: Rp15,000.00

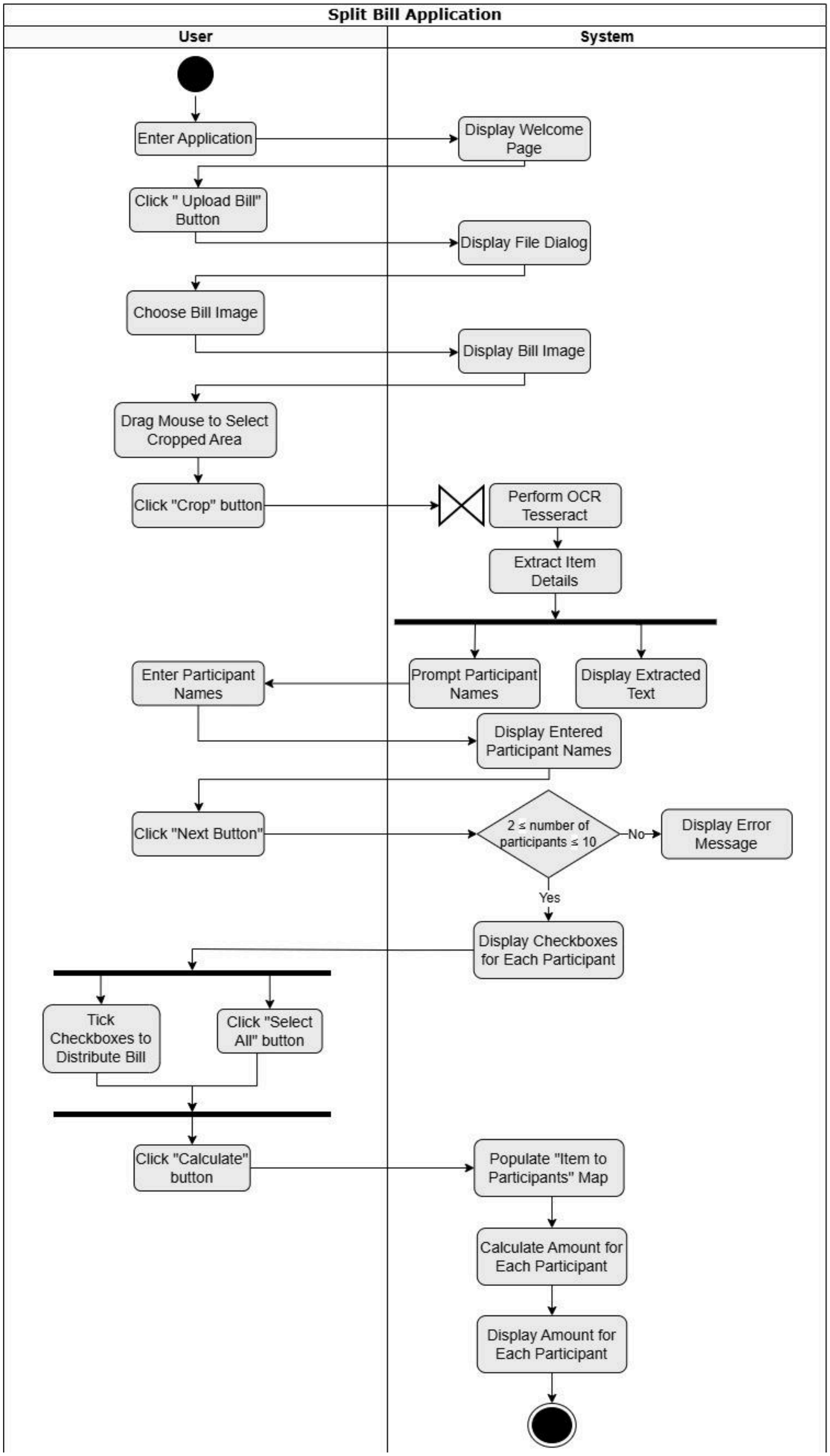
Next

Before creating the application, mockup designs were created using Canva to plan the visual properties and layout of the application. The chosen color scheme is grey and white, which imitates the typical features of receipts or bills as they often have a minimalistic and monochromatic color scheme. In addition, the font that is used is “Lucida Console” as it also represents a typical font of old-fashioned receipts. The overall visuals of the application are designed to enhance a connection with the application’s main function of splitting bills.

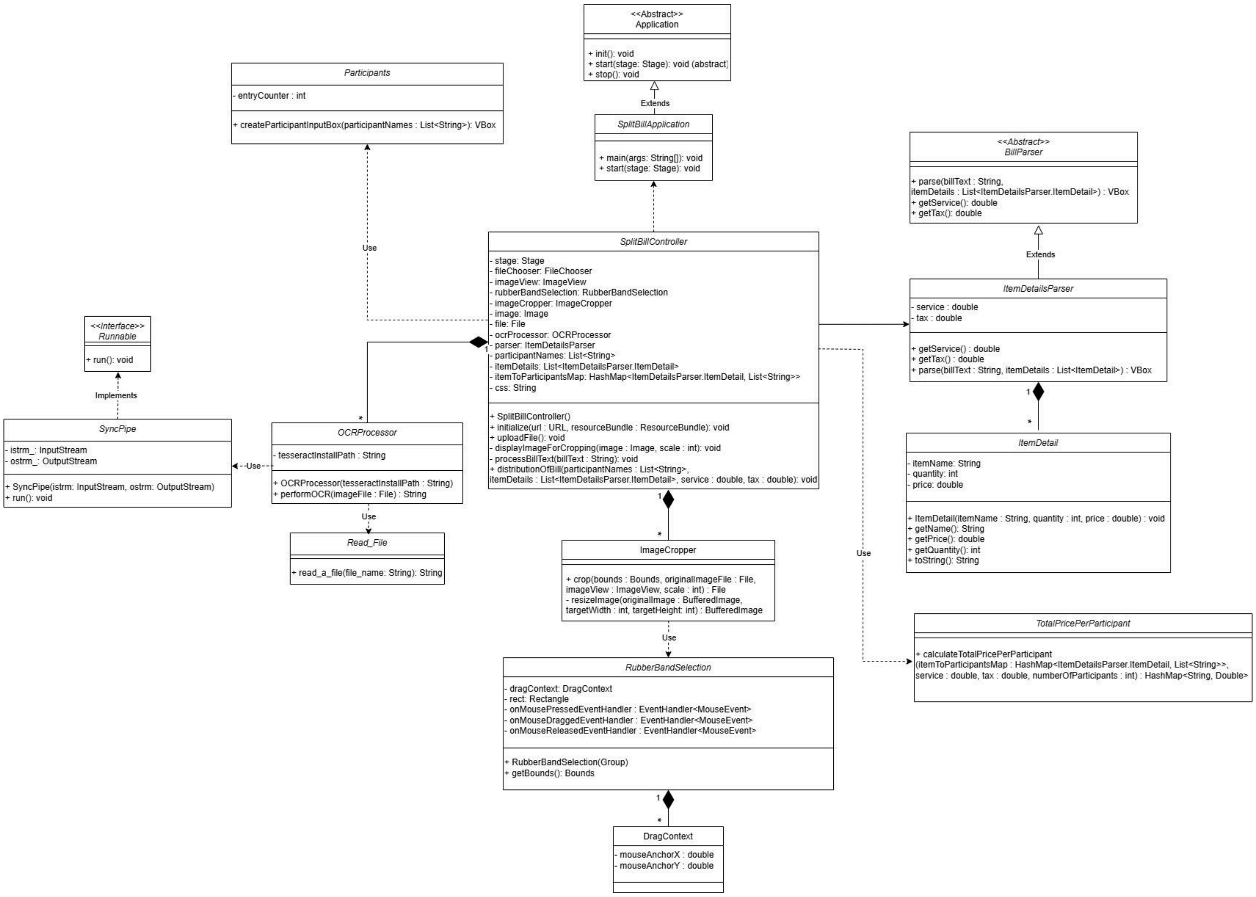
USE CASE DIAGRAM



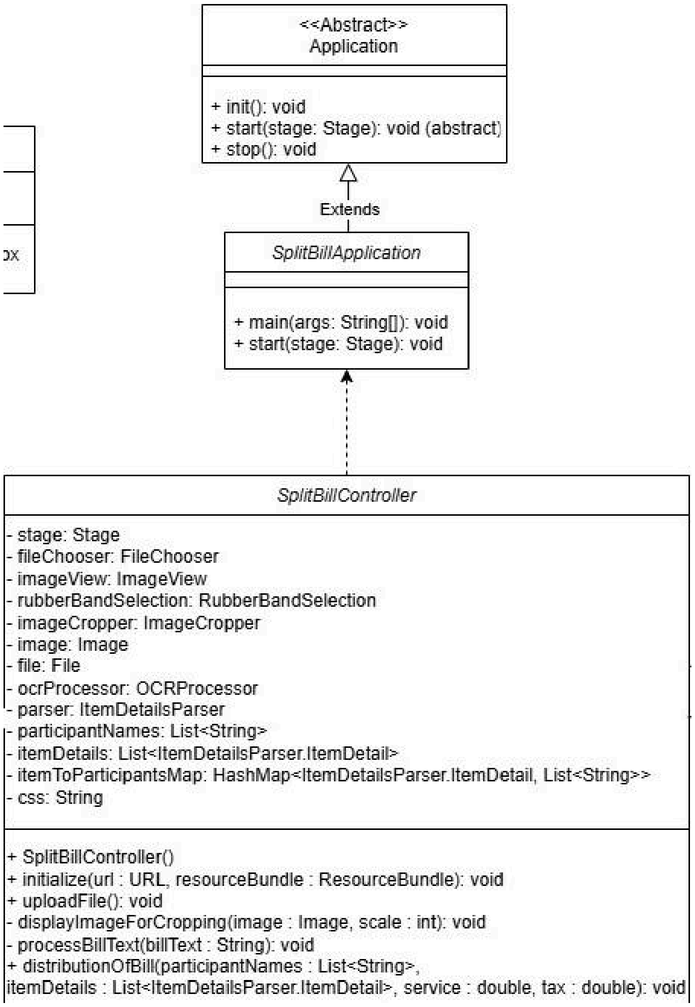
ACTIVITY DIAGRAM

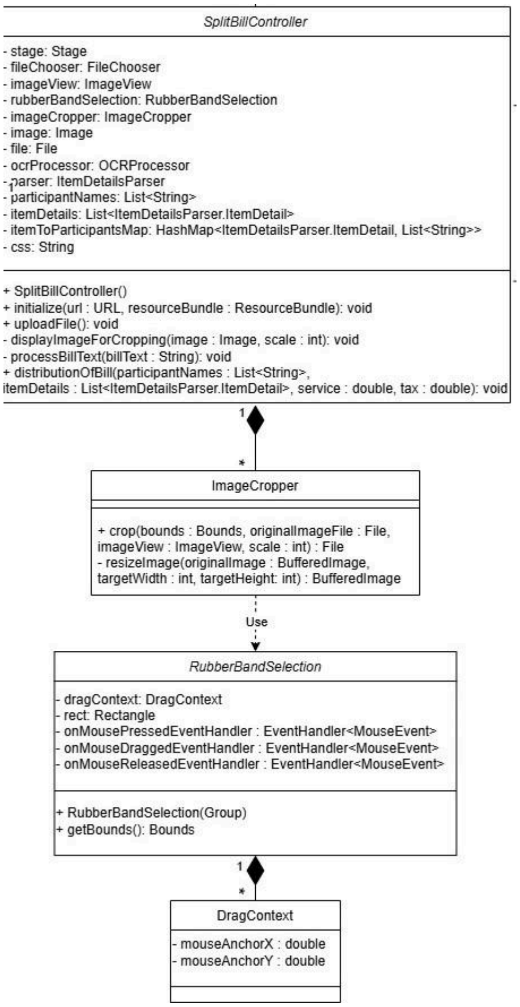

Split Bill Application
User
System
Enter Application
Display Welcome Page
Click " Upload Bill" Button
Display File Dialog
Choose Bill Image
Display Bill Image
Drag Mouse to Select Cropped Area
Click "Crop" button
Perform OCR Tesseract
Extract Item Details
Enter Participant Names
Prompt Participant Names
Display Extracted Text
Display Entered Participant Names
Click "Next Button"
2 ≤ number of participants ≤ 10
No
Display Error Message
Yes
Display Checkboxes for Each Participant
Tick Checkboxes to Distribute Bill
Click "Select All" button
Click "Calculate" button
Populate "Item to Participants" Map
Calculate Amount for Each Participant
Display Amount for Each Participant

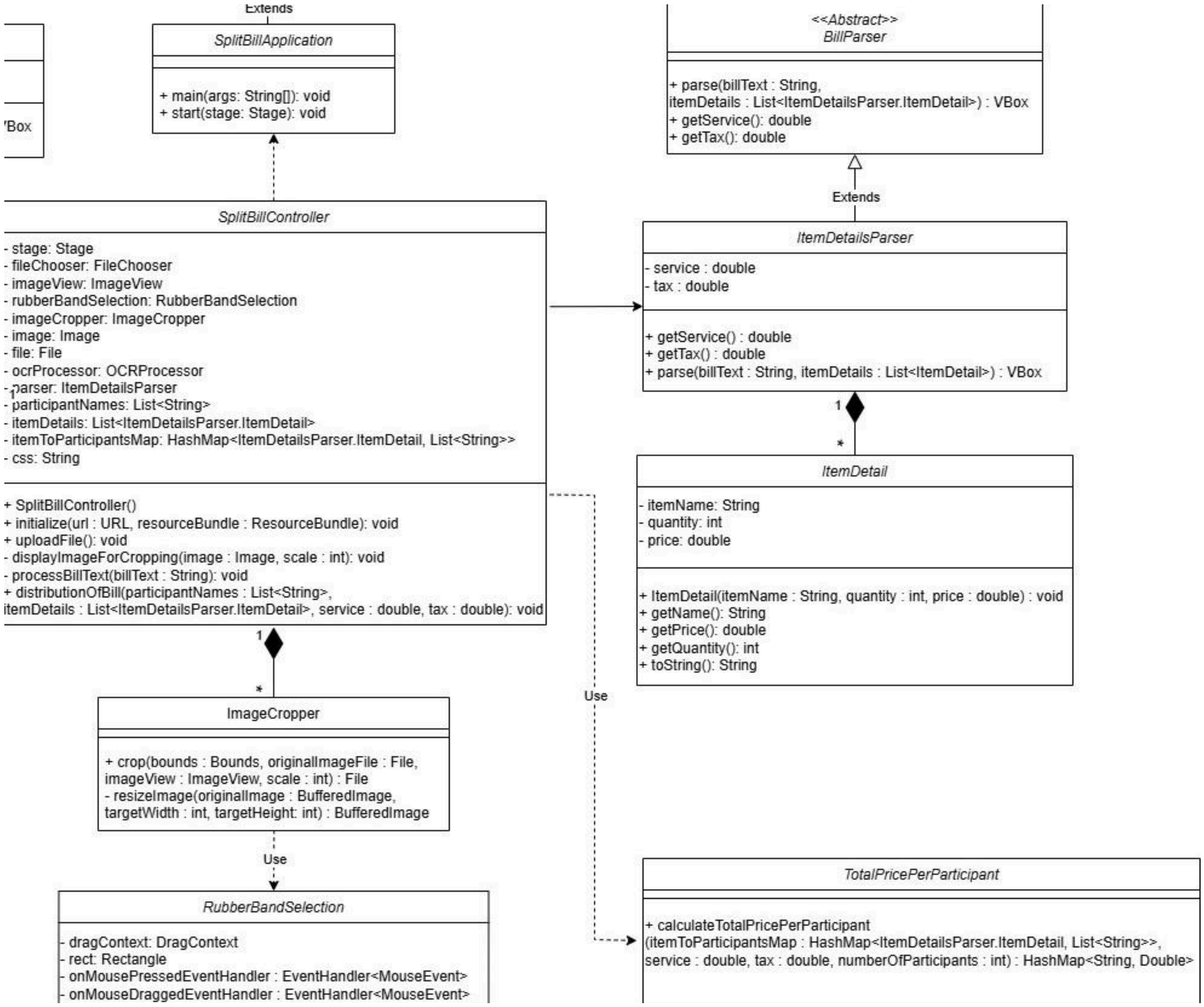
CLASS DIAGRAM

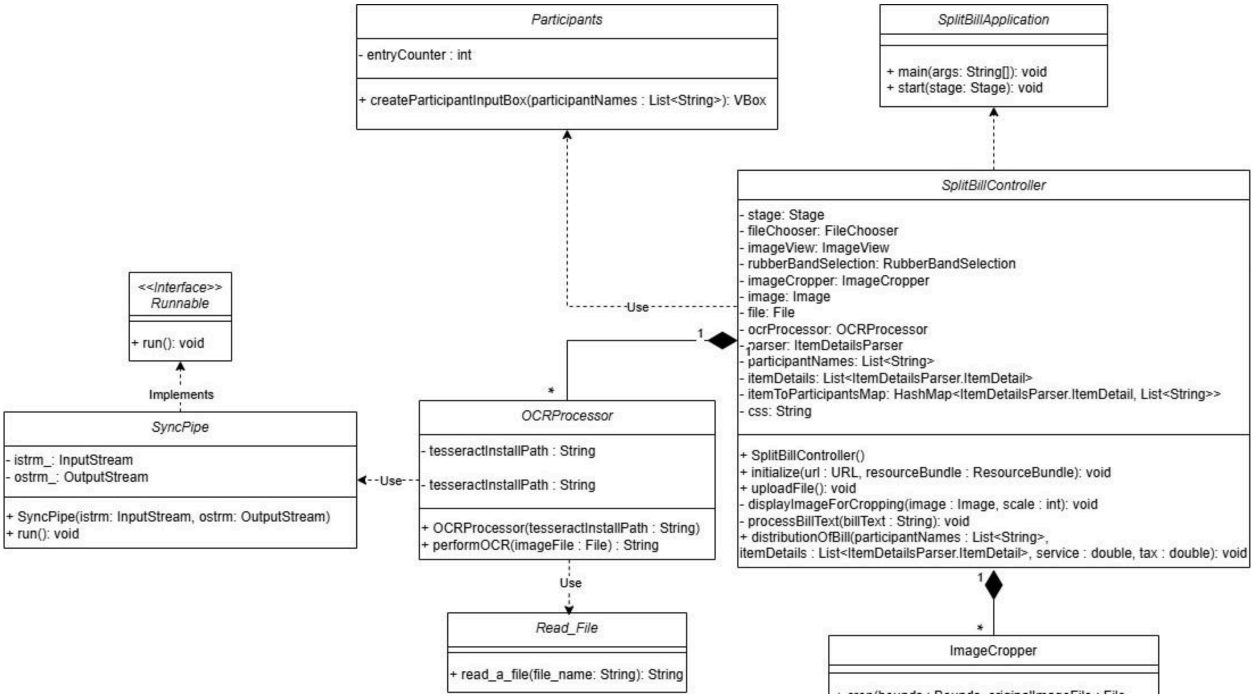

<<Abstract>>
Application
+ init(): void
+ start(stage: Stage): void (abstract)
+ stop(): void
Extends
SplitBillApplication
+ main(args: String[]): void
+ start(stage: Stage): void
Participants
- entryCounter : int
+ createParticipantInputBox(participantNames : List<String>): VBox
Use
<<Interface>>
Runnable
+ run(): void
Implements
SyncPipe
- istrm_: InputStream
- ostrm_: OutputStream
+ SyncPipe(istrm: InputStream, ostrm: OutputStream)
+ run(): void
Use
*
OCRProcessor
- tesseractInstallPath : String
+ OCRProcessor(tesseractInstallPath : String)
+ performOCR(imageFile : File) : String
Use
Read_File
+ read_a_file(file_name: String): String
1
SplitBillController
- stage: Stage
- fileChooser: FileChooser
- imageView: ImageView
- rubberBandSelection: RubberBandSelection
- imageCropper: ImageCropper
- image: Image
- file: File
- ocrProcessor: OCRProcessor
- parser: ItemDetailsParser
- participantNames: List<String>
- itemDetails: List<ItemDetailsParser.ItemDetail>
- itemToParticipantsMap: HashMap<ItemDetailsParser.ItemDetail, List<String>>
- css: String
+ SplitBillController()
+ initialize(url : URL, resourceBundle : ResourceBundle): void
+ uploadFile(): void
- displayImageForCropping(image : Image, scale : int): void
- processBillText(billText : String): void
+ distributionOfBill(participantNames : List<String>,
itemDetails : List<ItemDetailsParser.ItemDetail>, service : double, tax : double): void
<<Abstract>>
BillParser
+ parse(billText : String,
itemDetails : List<ItemDetailsParser.ItemDetail>) : VBox
+ getService(): double
+ getTax(): double
Extends
ItemDetailsParser
- service : double
- tax : double
+ getService() : double
+ getTax() : double
+ parse(billText : String, itemDetails : List<ItemDetail>) : VBox
1
*
ItemDetail
- itemName: String
- quantity: int
- price: double
+ ItemDetail(itemName : String, quantity : int, price : double) : void
+ getName(): String
+ getPrice(): double
+ getQuantity(): int
+ toString(): String
Use
1
*
ImageCropper
+ crop(bounds : Bounds, originalImageFile : File,
imageView : ImageView, scale : int) : File
- resizeImage(originalImage : BufferedImage,
targetWidth : int, targetHeight: int) : BufferedImage
Use
RubberBandSelection
- dragContext: DragContext
- rect: Rectangle
- onMousePressedEventHandler : EventHandler<MouseEvent>
- onMouseDraggedEventHandler : EventHandler<MouseEvent>
- onMouseReleasedEventHandler : EventHandler<MouseEvent>
+ RubberBandSelection(Group)
+ getBounds(): Bounds
1
*
DragContext
- mouseAnchorX : double
- mouseAnchorY : double
TotalPricePerParticipant
+ calculateTotalPricePerParticipant
(itemToParticipantsMap : HashMap<ItemDetailsParser.ItemDetail, List<String>>,
service : double, tax : double, numberOfParticipants : int) : HashMap<String, Double>

Details of the Class Diagram

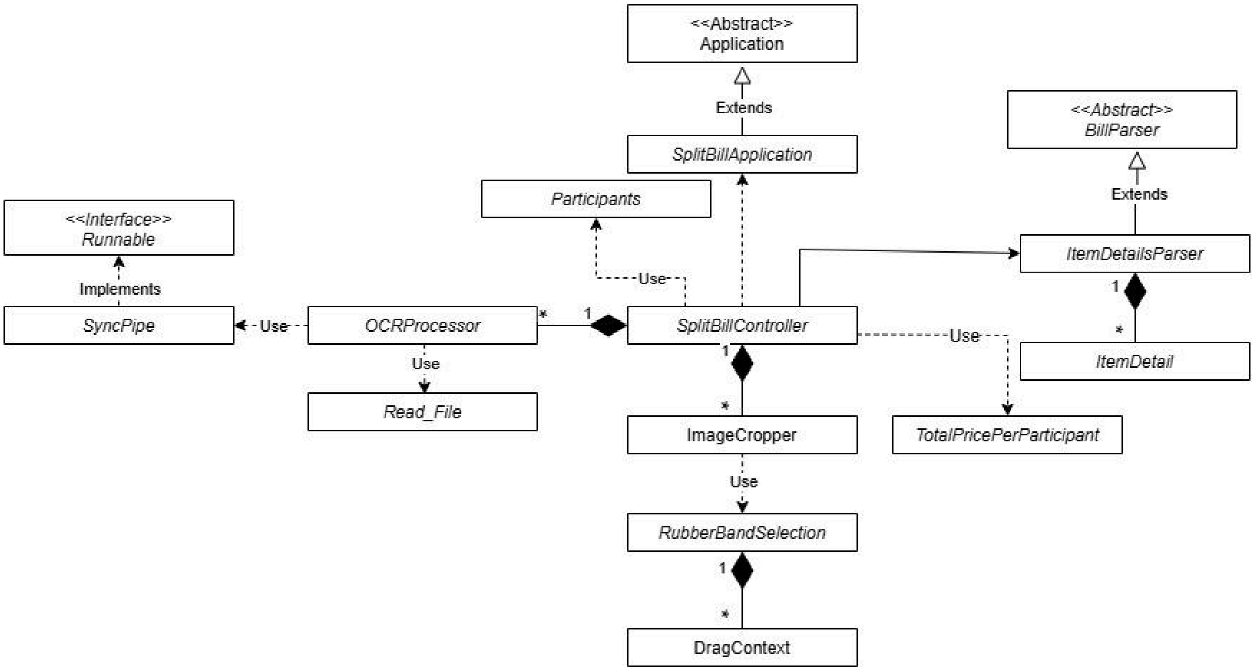

ox
<<Abstract>>
Application
+ init(): void
+ start(stage: Stage): void (abstract)
+ stop(): void
Extends
SplitBillApplication
+ main(args: String[]): void
+ start(stage: Stage): void
SplitBillController
- stage: Stage
- fileChooser: FileChooser
- imageView: ImageView
- rubberBandSelection: RubberBandSelection
- imageCropper: ImageCropper
- image: Image
- file: File
- ocrProcessor: OCRProcessor
- parser: ItemDetailsParser
- participantNames: List<String>
- itemDetails: List<ItemDetailsParser.ItemDetail>
- itemToParticipantsMap: HashMap<ItemDetailsParser.ItemDetail, List<String>>
- css: String
+ SplitBillController()
+ initialize(url : URL, resourceBundle : ResourceBundle): void
+ uploadFile(): void
- displayImageForCropping(image : Image, scale : int): void
- processBillText(billText : String): void
+ distributionOfBill(participantNames : List<String>,
itemDetails : List<ItemDetailsParser.ItemDetail>, service : double, tax : double): void


SplitBillController
- stage: Stage
- fileChooser: FileChooser
- imageView: ImageView
- rubberBandSelection: RubberBandSelection
- imageCropper: ImageCropper
- image: Image
- file: File
- ocrProcessor: OCRProcessor
- parser: ItemDetailsParser
- participantNames: List<String>
- itemDetails: List<ItemDetailsParser.ItemDetail>
- itemToParticipantsMap: HashMap<ItemDetailsParser.ItemDetail, List<String>>
- css: String
+ SplitBillController()
+ initialize(url : URL, resourceBundle : ResourceBundle): void
+ uploadFile(): void
- displayImageForCropping(image : Image, scale : int): void
- processBillText(billText : String): void
+ distributionOfBill(participantNames : List<String>,
itemDetails : List<ItemDetailsParser.ItemDetail>, service : double, tax : double): void
1
*
ImageCropper
+ crop(bounds : Bounds, originalImageFile : File,
imageView : ImageView, scale : int) : File
- resizeImage(originalImage : BufferedImage,
targetWidth : int, targetHeight: int) : BufferedImage
Use
RubberBandSelection
- dragContext: DragContext
- rect: Rectangle
- onMousePressedEventHandler : EventHandler<MouseEvent>
- onMouseDraggedEventHandler : EventHandler<MouseEvent>
- onMouseReleasedEventHandler : EventHandler<MouseEvent>
+ RubberBandSelection(Group)
+ getBounds(): Bounds
1
*
DragContext
- mouseAnchorX : double
- mouseAnchorY : double


Extends
'Box
SplitBillApplication
+ main(args: String[]): void
+ start(stage: Stage): void
<<Abstract>>
BillParser
+ parse(billText : String,
itemDetails : List<ItemDetailsParser.ItemDetail>) : VBox
+ getService(): double
+ getTax(): double
Extends
SplitBillController
- stage: Stage
- fileChooser: FileChooser
- imageView: ImageView
- rubberBandSelection: RubberBandSelection
- imageCropper: ImageCropper
- image: Image
- file: File
- ocrProcessor: OCRProcessor
- parser: ItemDetailsParser
- participantNames: List<String>
- itemDetails: List<ItemDetailsParser.ItemDetail>
- itemToParticipantsMap: HashMap<ItemDetailsParser.ItemDetail, List<String>>
- css: String
+ SplitBillController()
+ initialize(url : URL, resourceBundle : ResourceBundle): void
+ uploadFile(): void
- displayImageForCropping(image : Image, scale : int): void
- processBillText(billText : String): void
+ distributionOfBill(participantNames : List<String>,
itemDetails : List<ItemDetailsParser.ItemDetail>, service : double, tax : double): void
ItemDetailsParser
- service : double
- tax : double
+ getService() : double
+ getTax() : double
+ parse(billText : String, itemDetails : List<ItemDetail>) : VBox
1
*
ItemDetail
- itemName: String
- quantity: int
- price: double
+ ItemDetail(itemName : String, quantity : int, price : double) : void
+ getName(): String
+ getPrice(): double
+ getQuantity(): int
+ toString(): String
1
*
Use
ImageCropper
+ crop(bounds : Bounds, originalImageFile : File,
imageView : ImageView, scale : int) : File
- resizeImage(originalImage : BufferedImage,
targetWidth : int, targetHeight: int) : BufferedImage
Use
RubberBandSelection
- dragContext: DragContext
- rect: Rectangle
- onMousePressedEventHandler : EventHandler<MouseEvent>
- onMouseDraggedEventHandler : EventHandler<MouseEvent>
TotalPricePerParticipant
+ calculateTotalPricePerParticipant
(itemToParticipantsMap : HashMap<ItemDetailsParser.ItemDetail, List<String>>,
service : double, tax : double, numberOfParticipants : int) : HashMap<String, Double>


Participants
- entryCounter : int
+ createParticipantInputBox(participantNames : List<String>): VBox
SplitBillApplication
+ main(args: String[]): void
+ start(stage: Stage): void
SplitBillController
- stage: Stage
- fileChooser: FileChooser
- imageView: ImageView
- rubberBandSelection: RubberBandSelection
- imageCropper: ImageCropper
- image: Image
- file: File
- ocrProcessor: OCRProcessor
- parser: ItemDetailsParser
- participantNames: List<String>
- itemDetails: List<ItemDetailsParser.ItemDetail>
- itemToParticipantsMap: HashMap<ItemDetailsParser.ItemDetail, List<String>>
- css: String
+ SplitBillController()
+ initialize(url : URL, resourceBundle : ResourceBundle): void
+ uploadFile(): void
- displayImageForCropping(image : Image, scale : int): void
- processBillText(billText : String): void
+ distributionOfBill(participantNames : List<String>,
itemDetails : List<ItemDetailsParser.ItemDetail>, service : double, tax : double): void
Use
1
1
*
<<Interface>>
Runnable
+ run(): void
Implements
SyncPipe
- istrm_: InputStream
- ostrm_: OutputStream
+ SyncPipe(istrm: InputStream, ostrm: OutputStream)
+ run(): void
Use
OCRProcessor
- tesseractInstallPath : String
- tesseractInstallPath : String
+ OCRProcessor(tesseractInstallPath : String)
+ performOCR(imageFile : File) : String
Use
Read_File
+ read_a_file(file_name: String): String
1
*
ImageCropper

Minimized Class Diagram



III. IMPLEMENTATION

CLASSES AND METHODS

‘SplitBillApplication’ Class

This is the main class that initializes and starts the application, which extends the java.fx.application.Application class. It contains the ‘start(Stage primaryStage)’ method which overrides an abstract method from the JavaFX Application class. This method provides the application’s entry point as the ‘Stage’ object represents the application's main window. While the ‘primaryStage’ argument specifies a container for the user interface. The ‘start’ method loads the FXML and CSS files and sets the dimensions and title of the stage. In addition, this method is enclosed with a try-catch block to handle potential errors that may occur. When an error occurs, it is caught and printed to the console to prevent the application from crashing. On the other hand, the ‘main’ method is used to call the ‘launch(args)’ that launches the application.

‘SplitBillController’ Class

This class is responsible for handling user interactions and the application's logic. It controls the flow of the application, including uploading the image of the bill, processing the bill, and presenting the total amount for each participant. The ‘uploadFile’ method allows the user to choose an image from their file explorer and upload the image of a bill. Then, the ‘displayImageForCropping’ method displays the image in a new scene and allows the user to crop it. With the ‘rubberBandSelection’ class, the user can drag their mouse to select the area where the bill mentions the item details, including item name, quantity, and price, as well as the service charge and tax. This is done to ensure that the OCR Tesseract can accurately extract the text.

Next, the text of the cropped image is extracted by the ‘OCRprocessor’ class. The extracted text is then parsed with an instance of the ‘itemDetailsParser’ class, referred to as the ‘parser’ object. It identifies the item names, quantity, price, service charge, and tax. The item details are then presented in a new scene. In addition, the program allows the user to input the participant names. As the user inputs the correct number of participants, the program continues to a new scene where the user can customize the distribution of items using checkboxes. There is a validation mechanism that requires the user to allocate all items to at least one participant and assign all participants to at least one item. If these exceptions are not fulfilled, an alert will pop out. The alert is created using the Alert class from the JavaFX API.

As the user customizes the allocation of items for every participant, the program populates a HashMap named ‘itemToParticipantsMap’ to store information about the distribution of the bill. Then, it calculates and displays the total amount each participant owes. In addition, there is also a “Select All” button, which automatically ticks all checkboxes of all participants. This eases the user when they want to divide the bill equally.

‘ImageCropper’ Class

It handles the cropping bill image feature. It consists of the ‘crop’ method which takes the bounds of the cropped area, the original image, the ‘ImageView’ that displays the original image, and a scale factor as its parameters. The method begins by defining the directory and file name of the cropped image. The cropped image will be saved in the same directory as the original image. While the file name will be the original name followed by “_cropped.png”. Next, it maps the selection bounds from the ‘ImageView’ coordinates to the coordinates on the original image. Then, it creates a buffered image to create a new image, which represents the cropped area. Lastly, the method returns the obtained cropped image. In addition, the ‘resizeImage’ method is useful to maintain the quality and readability of the cropped image. This is important because the accuracy of the OCR Tesseract is very sensitive to the quality of the image. If the image is too small or blurry, it won’t be able to accurately extract the text.

‘RubberBandSelection’ Class

This class handles the selection of cropped areas. The ‘DragContext’ inner class contains fields including ‘mouseAnchorX’ and ‘mouseAnchorY’ to represent the initial coordinates of the mouse. A ‘rect’ object is initialized to represent the selected cropped area. The constructor ‘RubberBandSelection’ sets the visual properties of the rectangle and adds it to the ‘Group’. In addition, it also adds mouse event handlers to the group. ‘onMousePressedEventHandler’ obtains the initial position of the mouse as it initializes the rectangle’s coordinates and size to zero, storing them in the ‘DragContext’ class. While ‘onMouseDraggedEventHandler’ updates the dimensions of the rectangle as the user drags their mouse by calculating the horizontal and vertical distances from the initial mouse position. Lastly, ‘onMouseReleasedEventHandler’ finalizes the selected area when the mouse is no longer pressed.

‘OCRProcessor’ Class

This class has the role of executing the OCR Tesseract to extract texts from the cropped image. It starts by defining the path to the Tesseract installation directory. Then, it is initialized with a constructor that sets the OCR Tesseract. The ‘performOCR’ method takes the image file as an input and outputs a string of the extracted text. It then generates a text file that contains the extracted text. It is saved in the same directory and file name as the processed image.

‘Read_File’ Class

After the ‘OCRProcessor’ generates the text file, the ‘Read_File’ class reads this file and returns it as a single string. It has a method named, ‘read_a_file’ that takes a file name as its parameter. ‘BufferedReader’ is used to read the file line by line. As it reads every line of the text, it appends the line to a string variable named ‘read_string’. The method then returns the concatenated string, representing the compiled text from the image.

‘SyncPipe’ Class

This class ensures the synchronization of input and output streams between processes. It implements the ‘Runnable’ interface, allowing instances of this class to be executed by a thread. The class has two fields, which are ‘ostrm_’ where data will be written, and ‘istrm_’ where data will be read. The ‘run’ method reads data from the input stream and writes the data to the output stream. This method ensures that data is efficiently transferred.

‘ItemDetailsParser’ Class

This class is responsible for parsing the extracted text to recognize the item name, quantity, price, service charge, and tax. It extends an abstract class named ‘BillParser’, which defines abstract methods needed to parse the bill text, return service charge, and tax. This abstract class allows flexibility for parsing different types of bills that have different formats or patterns in the presentation of item details.

In addition, the ‘ItemDetailsParser’ class has a nested class called ‘ItemDetail’ to represent the name, quantity, and price of items. It defines fields such as ‘itemName’ as a String, ‘quantity’ as an integer, and ‘price’ as a double. Then, it has a constructor to initialize the ‘ItemDetail’ object, along with three getters that return the item name, price, and quantity. Lastly, the ‘toString’ method is created to format how the item details are presented.

The ‘parse’ method is crucial as it includes functions to obtain the item details. It takes a String, ‘billText’, and List, ‘itemDetails’ as its parameters. It first creates a VBox container to contain the item details that are going to be obtained. Then, it replaces all existing dots in prices with commas. This is needed because some bills may use dots as the thousands separator instead of commas. Thus, unifying all the thousand separators into commas will make it easier for the program to extract the price. Then, the code uses a regular expression to match the item details in the bill text. For each match, it extracts the quantity, item name, and price, creating an ‘ItemDetail’ object and adding it to the ‘itemDetails’ list. In addition, the method also uses a regular expression to extract the service charge and text. For each match, it identifies the label, such as “SERVICE CHARGE”, “TAX”, or “PB1”, and the corresponding value. Then, it sets the ‘service’ and ‘tax’ field accordingly.

‘Participants’ Class

The class manages the collection of participant names inputted by the user. It consists of a method named ‘createparticipantInputBox’ to create a ‘TextField’ for the user to type in the names of participants. In addition, it allows the user to press enter for every participant input. As the user presses enter, the input will be displayed below the input box.

‘TotalPricePerParticipant’ Class

This class calculates the total amount that every participant contributes. It contains a method named ‘calculateTotalPricePerParticipant’, which takes the ‘itemToParticipantsMap’, ‘service’, ‘tax’, and ‘numberOfParticipants’ as its parameters. The method iterates every entry in the ‘itemToParticipantsMap’. For every item, it calculates the price that every participant owes by dividing the price by the number of participants assigned to it. Hence, if an item is assigned to more than one participants, the price will be divided by the number of participants assigned to it. Then, it distributes the result of the division among those participants. The service charge and tax are divided equally among all participants. This method returns a HashMap named ‘totalPricePerParticipant’, which maps the total amount each participant needs to pay.

DATA STRUCTURE

List

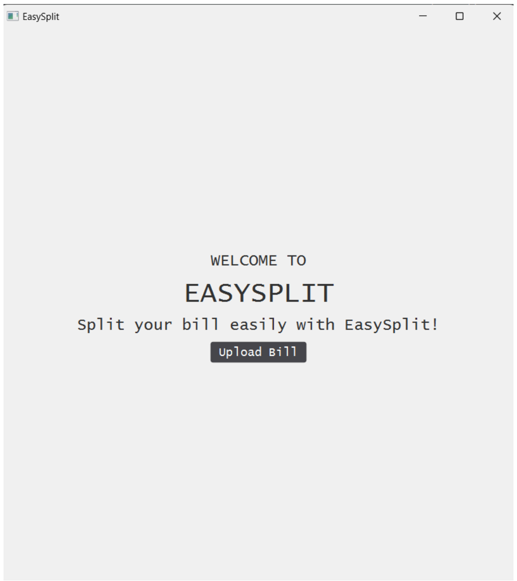
- ‘List <String> participantNames’: Holds the participant names from the user input.
- ‘List <ItemDetailsParser.ItemDetail> itemDetails’: Holds the item details extracted by the parser.

HashMap

- ‘HashMap <ItemDetailsParser.ItemDetail, List <String>> itemToParticipantsMap’: Maps each item to a list of participants who are assigned to it.
- ‘HashMap <String, Double> totalPricePerParticipant’: Maps each participant to the amount they owe.

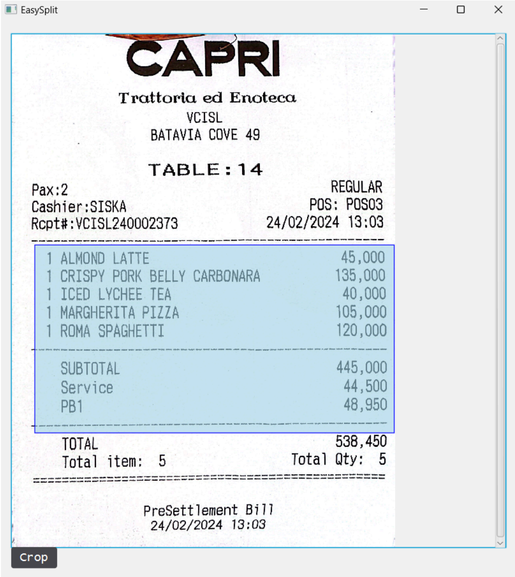
IV. EVIDENCE OF WORKING PROGRAM

Welcome Page



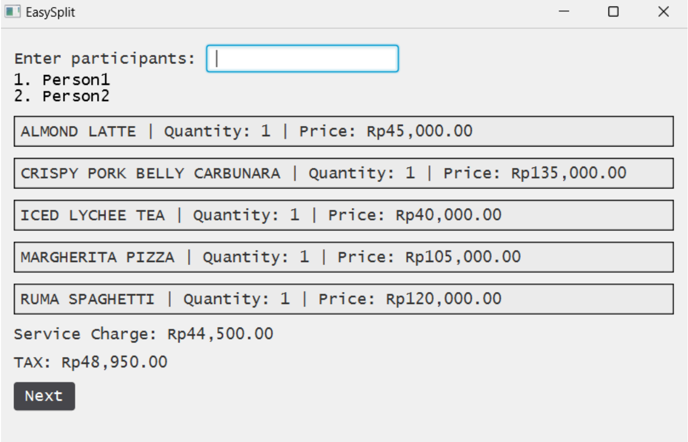
```
WELCOME TO
EASYSPLIT
Split your bill easily with EasySplit!
```

Displaying and cropping the uploaded image



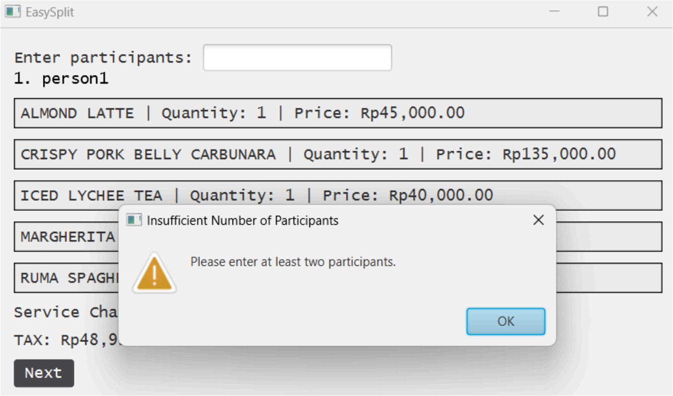
```
CAPRI
Trattoria ed Enoteca
VCISL
BATAVIA COVE 49

TABLE:14
Pax:2                                REGULAR
Cashier:SISKA                     POS: POS03
Rcpt#:VCISL240002373      24/02/2024  13:03
---------------------------------------------
 1 ALMOND LATTE                       45,000
 1 CRISPY PORK BELLY CARBONARA       135,000
 1 ICED LYCHEE TEA                    40,000
 1 MARGHERITA PIZZA                  105,000
 1 ROMA SPAGHETTI                    120,000
---------------------------------------------
   SUBTOTAL                          445,000
   Service                            44,500
   PB1                                48,950
---------------------------------------------
   TOTAL                             538,450
   Total item:  5              Total Qty:  5
=============================================

            PreSettlement Bill
             24/02/2024 13:03
```

Entering participant names and displaying extracted item details



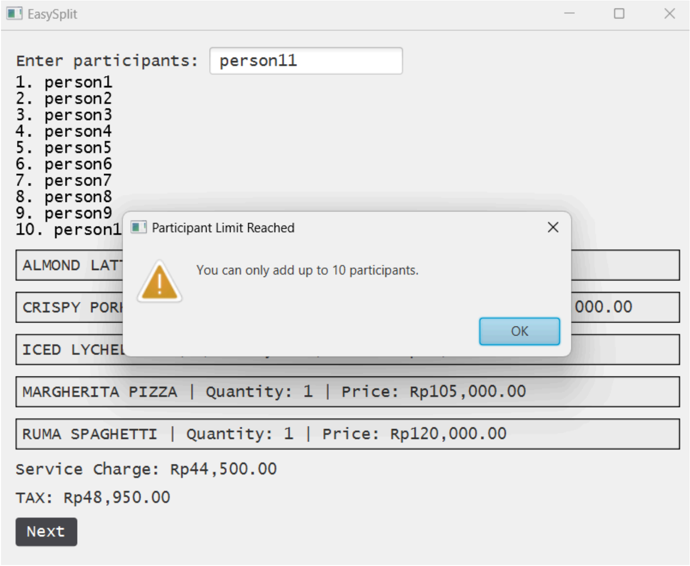
```
Enter participants: |
1. Person1
2. Person2
ALMOND LATTE | Quantity: 1 | Price: Rp45,000.00
CRISPY PORK BELLY CARBUNARA | Quantity: 1 | Price: Rp135,000.00
ICED LYCHEE TEA | Quantity: 1 | Price: Rp40,000.00
MARGHERITA PIZZA | Quantity: 1 | Price: Rp105,000.00
RUMA SPAGHETTI | Quantity: 1 | Price: Rp120,000.00
Service Charge: Rp44,500.00
TAX: Rp48,950.00
```

Alert when input is less than 2



```
Enter participants:
1. person1
ALMOND LATTE | Quantity: 1 | Price: Rp45,000.00
CRISPY PORK BELLY CARBUNARA | Quantity: 1 | Price: Rp135,000.00
ICED LYCHEE TEA | Quantity: 1 | Price: Rp40,000.00
MARGHERITA
RUMA SPAGHI
Service Cha
TAX: Rp48,9

Insufficient Number of Participants
Please enter at least two participants.
```

Alert when input is more than 10

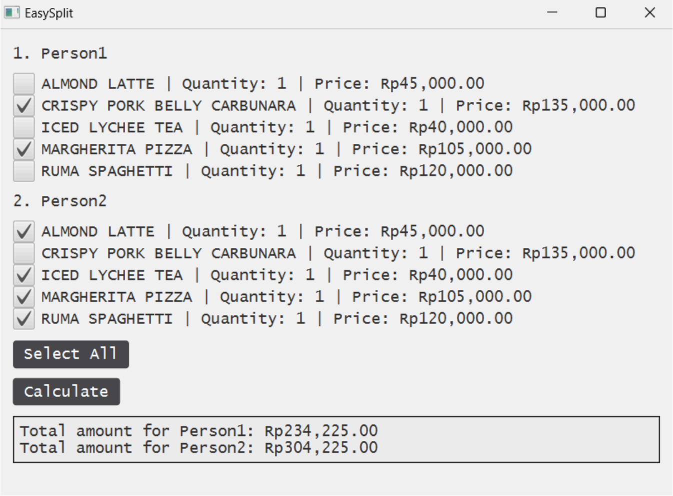


```
Enter participants: person11
1. person1
2. person2
3. person3
4. person4
5. person5
6. person6
7. person7
8. person8
9. person9
10. person1
ALMOND LAT
CRISPY PORI                                  000.00
ICED LYCHEI
MARGHERITA PIZZA | Quantity: 1 | Price: Rp105,000.00
RUMA SPAGHETTI | Quantity: 1 | Price: Rp120,000.00
Service Charge: Rp44,500.00
TAX: Rp48,950.00
```

Participant Limit Reached

You can only add up to 10 participants.

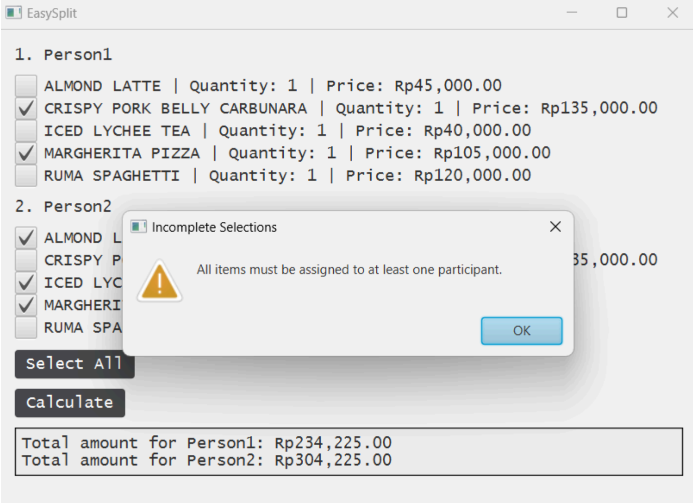
Displaying checkboxes, customizing the allocation of items, and displaying the total amount of participants



```
1. Person1
☐ ALMOND LATTE | Quantity: 1 | Price: Rp45,000.00
☑ CRISPY PORK BELLY CARBUNARA | Quantity: 1 | Price: Rp135,000.00
☐ ICED LYCHEE TEA | Quantity: 1 | Price: Rp40,000.00
☑ MARGHERITA PIZZA | Quantity: 1 | Price: Rp105,000.00
☐ RUMA SPAGHETTI | Quantity: 1 | Price: Rp120,000.00
2. Person2
☑ ALMOND LATTE | Quantity: 1 | Price: Rp45,000.00
☐ CRISPY PORK BELLY CARBUNARA | Quantity: 1 | Price: Rp135,000.00
☑ ICED LYCHEE TEA | Quantity: 1 | Price: Rp40,000.00
☑ MARGHERITA PIZZA | Quantity: 1 | Price: Rp105,000.00
☑ RUMA SPAGHETTI | Quantity: 1 | Price: Rp120,000.00
```

```
Total amount for Person1: Rp234,225.00
Total amount for Person2: Rp304,225.00
```

Alert when an item is not assigned to any participants



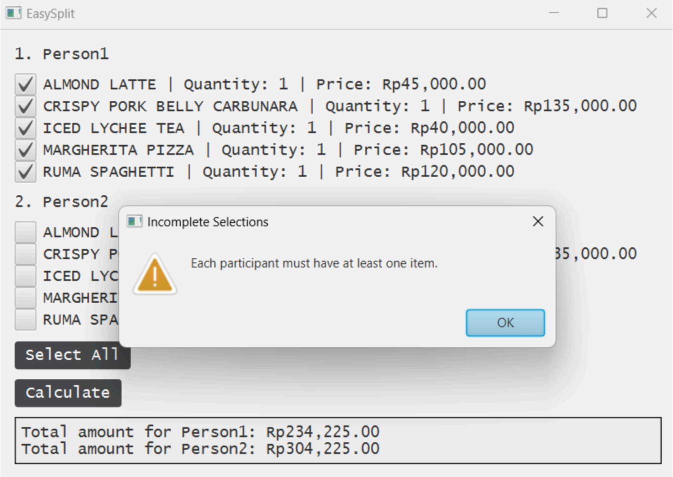
```
1. Person1
☐ ALMOND LATTE | Quantity: 1 | Price: Rp45,000.00
☑ CRISPY PORK BELLY CARBUNARA | Quantity: 1 | Price: Rp135,000.00
☐ ICED LYCHEE TEA | Quantity: 1 | Price: Rp40,000.00
☑ MARGHERITA PIZZA | Quantity: 1 | Price: Rp105,000.00
☐ RUMA SPAGHETTI | Quantity: 1 | Price: Rp120,000.00
2. Person2
☑ ALMOND L
☐ CRISPY P                                    35,000.00
☑ ICED LYC
☑ MARGHERI
☐ RUMA SPA
```

Incomplete Selections

All items must be assigned to at least one participant.

```
Total amount for Person1: Rp234,225.00
Total amount for Person2: Rp304,225.00
```

Alert a participant is not assigned to any items



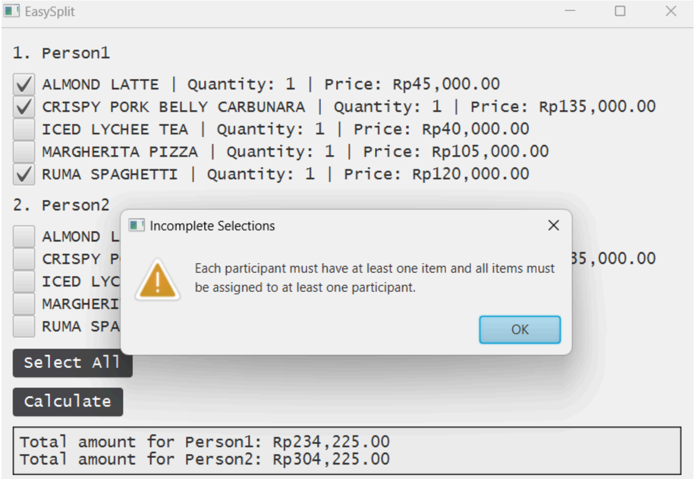
```
1. Person1
☑ ALMOND LATTE | Quantity: 1 | Price: Rp45,000.00
☑ CRISPY PORK BELLY CARBUNARA | Quantity: 1 | Price: Rp135,000.00
☑ ICED LYCHEE TEA | Quantity: 1 | Price: Rp40,000.00
☑ MARGHERITA PIZZA | Quantity: 1 | Price: Rp105,000.00
☑ RUMA SPAGHETTI | Quantity: 1 | Price: Rp120,000.00
2. Person2
☐ ALMOND L
☐ CRISPY P                                    35,000.00
☐ ICED LYC
☐ MARGHERI
☐ RUMA SPA
```

Incomplete Selections

Each participant must have at least one item.

```
Total amount for Person1: Rp234,225.00
Total amount for Person2: Rp304,225.00
```

Alert when an item is not assigned to any participants and a participant is not assigned to any items



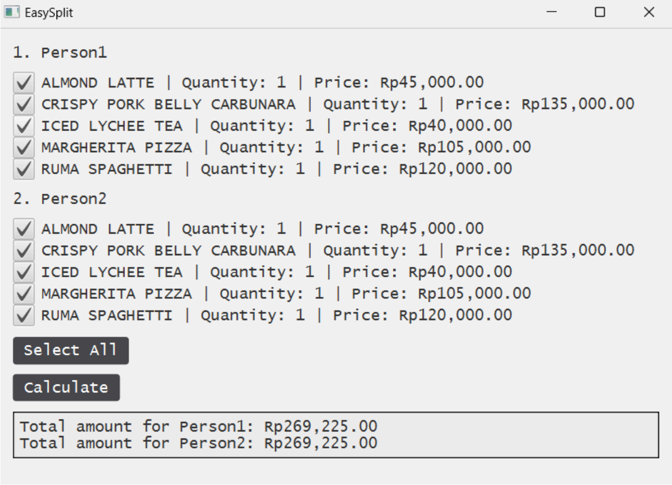
```
1. Person1
☑ ALMOND LATTE | Quantity: 1 | Price: Rp45,000.00
☑ CRISPY PORK BELLY CARBUNARA | Quantity: 1 | Price: Rp135,000.00
☐ ICED LYCHEE TEA | Quantity: 1 | Price: Rp40,000.00
☐ MARGHERITA PIZZA | Quantity: 1 | Price: Rp105,000.00
☑ RUMA SPAGHETTI | Quantity: 1 | Price: Rp120,000.00
2. Person2
☐ ALMOND L
☐ CRISPY P                                              35,000.00
☐ ICED LYC
☐ MARGHERI
☐ RUMA SPA
```

Incomplete Selections

Each participant must have at least one item and all items must be assigned to at least one participant.

```
Total amount for Person1: Rp234,225.00
Total amount for Person2: Rp304,225.00
```

“Select All” button to divide the bill equally



```
1. Person1
☑ ALMOND LATTE | Quantity: 1 | Price: Rp45,000.00
☑ CRISPY PORK BELLY CARBUNARA | Quantity: 1 | Price: Rp135,000.00
☑ ICED LYCHEE TEA | Quantity: 1 | Price: Rp40,000.00
☑ MARGHERITA PIZZA | Quantity: 1 | Price: Rp105,000.00
☑ RUMA SPAGHETTI | Quantity: 1 | Price: Rp120,000.00
2. Person2
☑ ALMOND LATTE | Quantity: 1 | Price: Rp45,000.00
☑ CRISPY PORK BELLY CARBUNARA | Quantity: 1 | Price: Rp135,000.00
☑ ICED LYCHEE TEA | Quantity: 1 | Price: Rp40,000.00
☑ MARGHERITA PIZZA | Quantity: 1 | Price: Rp105,000.00
☑ RUMA SPAGHETTI | Quantity: 1 | Price: Rp120,000.00
```

```
Total amount for Person1: Rp269,225.00
Total amount for Person2: Rp269,225.00
```

GitHub repository: vionadjunaidy/SplitBillApplication (github.com)

Video demo: Demo_OOP

V. REFLECTION

This project was highly challenging for me as most of the components of the application were not covered in class, which required me to conduct independent research. The hardest part of this project would be creating the classes and methods for cropping the image and extracting the text from the image. Initially, I didn’t plan to include the cropping feature. However, the OCR Tesseract couldn’t accurately extract the text. Thus, I decided to add the cropping feature so that the user can select the part that contains the necessary information, such as item details, service charge, and tax. This way, the image is more focused on the important information, increasing the accuracy of the OCR Tesseract. However, the OCR Tesseract still struggled to extract the text. Thus, I had to construct a new method to maintain the quality of the image. Overall, ensuring the accuracy of the OCR was one of the most challenging parts of the project, but I am glad that I was able to find effective solutions for it.

For further improvements, I would like to expand my application so that it can split bills that have different templates or layouts. In this project, I used a regular expression that only recognizes a specific pattern, where the quantity is identified as a number with one or more digits, followed by one or more whitespace, and the item name, which is recognized with one or more word characters. Then, the item name should be followed by a numeric value with a thousand separator, which represents the price. With this pattern, the program would not be able to extract item details from bills with different layouts. Thus, I would like to improve my program so that it can be used for more diverse layouts.

VI. RESOURCES

baeldung. (2018, April 9). *Introduction to JavaFX | Baeldung*. Www.baeldung.com. https://www.baeldung.com/javafx

3 Hello World, JavaFX Style (Release 8). (n.d.). Docs.oracle.com. https://docs.oracle.com/javase/8/javafx/get-started-tutorial/hello_world.htm

JavaFX Application Basic Structure By Example - Dev.java. (n.d.). Dev.java: The Destination for Java Developers. Retrieved June 9, 2024, from https://dev.java/learn/javafx/structure/

Convert image to text using CMD Command Prompt, Tesseract Optical Character Recognition (OCR). (2017, May 28). ChillyFacts. https://chillyfacts.com/convert-image-to-text-using-cmd-prompt/