NEW YORK INSTITUTE OF TECHNOLOGY

Old Westbury Campus

SENIOR PROJECT FINAL REPORT

(CSCI 455 - W01 - Dr. Frank Lee)

COOKBOOK

Project Leader:

1. Aloysius Arno Wiputra #1244139

Members

1. Ruchira Bunga #1262634
2. Hetul Patel #1250935

New York Institute of Technology

Northern Boulevard at Valentines Lane

Old Westbury, New York 11568

United States of America

ABSTRACT

One of the essential aspects of cooking is the recipes which depict how certain ingredients can be combined and processed to form an edible end result. These recipes, especially in the era of globalization, often come from the internet. However, in the process of padding out content and slowing click through rate, websites often hide their recipes behind layers of text which tell the backstory instead of describing just the recipes themselves. This application seeks to solve that by using methods that we have learned throughout our time in New York Tech and show proficiency in the various skills that we have been taught. As we wanted the focus to be on the recipes themselves, we tried to focus our UX and UI design to be simple and minimalist, coding it in HTML and CSS. We wanted the experience to be smooth, breaking it down to three simple stages. The backend uses Java, which is the language taught in most of the curriculum. We wanted to take full advantage of Java's Object-Oriented Programming principle, using things like encapsulation and polymorphism. We designed it to be modular, with various functions such as the web scraping, database connection, and account control in its own package with different classes and methods. We also wanted integration to a database, hosting it in a local MySQL instance with input/output access in application. To this end, we separated the tasks into front end, back end, and research/database design to try and make the project as effective as possible. Ultimately, due to issues in research and our own limitations, we were unable to fully integrate the CSS/HTML and decided to run a driver program through a terminal. The project is essentially then functionally complete, but aesthetically lacking. We learned a variety of things including leadership, time management, and certainly skills to take out into the world.

TABLE OF CONTENTS

ABSTRACT **1**

TABLE OF CONTENTS **2**

CHAPTER 1
PROBLEM STATEMENT **4**

1.1 Motivation 4

1.2 Goal 4

1.2.1 Must Have 5

1.2.2 Could Have 5

CHAPTER 2
ANALYSIS AND DESIGN **6**

2.1 Analysis 6

2.2 Design 7

2.2.1 Program Flow 8

2.2.3 Program Structure 10

CHAPTER 3
DEVELOPMENT PROCESS **12**

3.1 Task Division 12

3.2 Front End Development 13

3.3 Back End Development 15

3.3.1 cookbook.webscraper.CookbookScraper 15

3.3.2 cookbook.dbconnection.DatabaseControl 22

3.3.3.1 cookbook.accountcontrol.AccountLogin 24

3.3.3.2 cookbook.accountcontrol.AccountRegister 26

3.3.3.3 cookbook.accountcontrol.PasswordControl 28

3.3.4 cookbook.Demo 30

3.4 Database Development and Hosting 48

CHAPTER 4
TESTING **50**

4.1 Sample Output 50

Chapter 5
SHORTCOMINGS, WHAT WE LEARNED, AND FUTURE WORK **55**

5.1 Shortcomings 55

5.2 What We Learned 55

5.2.1 From Aloysius Arno Wiputra's Perspective 55

5.2.2 From Hetul Patel's Perspective 56

5.3.3 From Ruchira Bunga's Perspective 56

5.4 Future Work 57
5.5 Conclusion 57

CHAPTER 1

PROBLEM STATEMENT

1.1 Motivation

The motivation to create this software arose when a person decides to cook something new or even wants to learn to cook basic dishes but can't find a website which provides a clear recipe without a layer of information or a backstory. As cooking is one of the most valuable as well as essential skills that can be learned through easy steps instruction with the recipes provided. The need for this application is to provide the recipes with the best flowing experience and without the filler data which is irrelevant to the users, so that they can enjoy cooking / learning how to cook.

1.2 Goal

The main goal of this project is to demonstrate the skills that we learned throughout New York Tech's curriculum which includes:

1. Proficiency in Java
    a. CSCI 125 - Programming I
    b. CSCI 185 - Programming II
    c. CSCI 260 - Data Structures
2. Proficiency in data evaluation + web scraping
    a. CSCI 436 - Big Data Analytics
    b. CSCI 415 - Intro to Data Mining
    c. CSCI 426 - Information Retrieval
3. Proficiency in designing and creating algorithms
    a. CSCI 235 - Discrete Structures
    b. CSCI 335 - Design and Analysis of Algorithms
4. Proficiency in database design
    a. CSCI 360 - Database Management
5. Ability to work according to SDLC
    a. CSCI 380 - Introduction to Software Engineering

Along with demonstrating the skills, our goal for this project is to create a functional system which represents our skillset and will be helpful for us when we interview for jobs and present this project as a summary of the skills for reference.

1.2.1 Must Have

The must-haves of this project include a functional web scraper that works on more than 3 different websites. Also a functional database for recipe storage with unique ID for each recipe, a driver program that allows read and write control for the database, with input sanitation, version control integration of the whole design and cloud database hosting.

1.2.2 Could Have

The could-haves of this project include an additional optional module to help with meal prepping, with hardware integration with Arduino/Rpi, account system with password encryption and hashing for security purposes, a QR scanning system from the unique recipe ID such that we can easily look up the entries and allow easy access to the recipe that the user is searching for, a machine learning for recipe and web scraping, to further expand the usable sites in the future, and a additional database features such as grocery list with online lookup and inventory management.

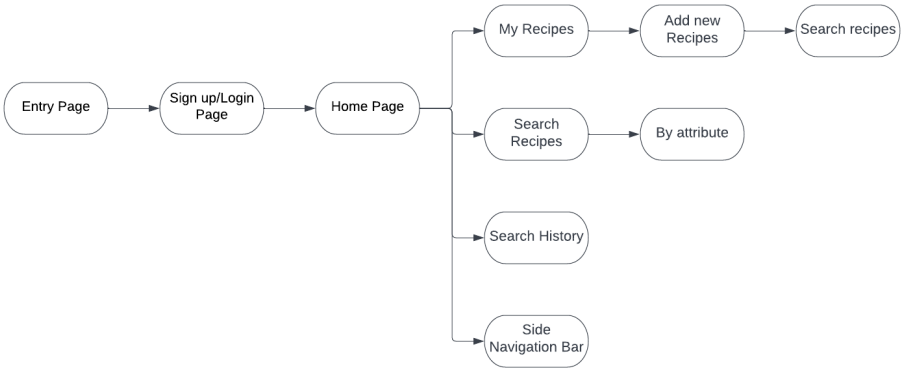
CHAPTER 2
ANALYSIS AND DESIGN

2.1 Analysis

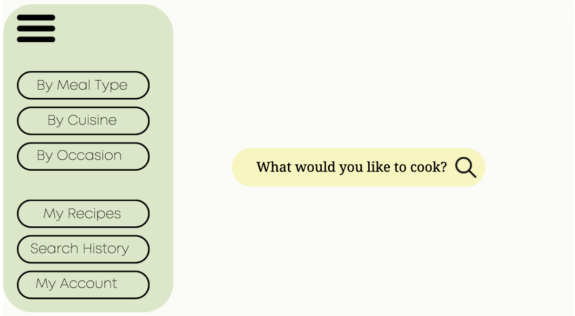
The primary objective of our application is based on a simplistic model and providing users with the ability to attain the information they require in a quick and effortless manner. Our design language reflects this ideology. We primarily focused on pastel colors as a palette, with two fonts to accent certain titles yet remain fundamental.

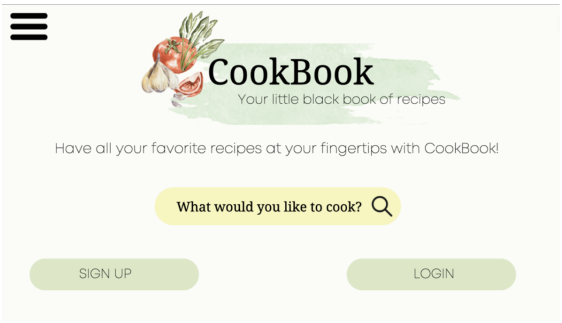
In accordance with our design language, our logo presents the paramount ideology behind the project at first glance while maintaining the intention of remaining simplistic.

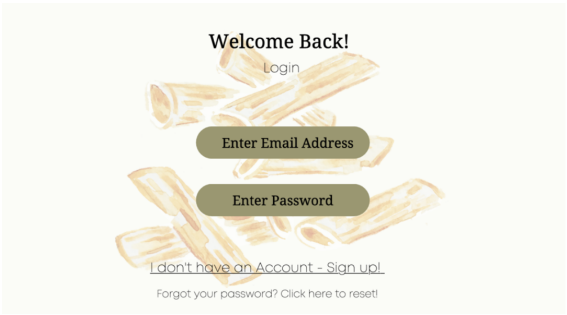
The flow of the webpage was initially designed with the layout described below -



The images below show each page design -







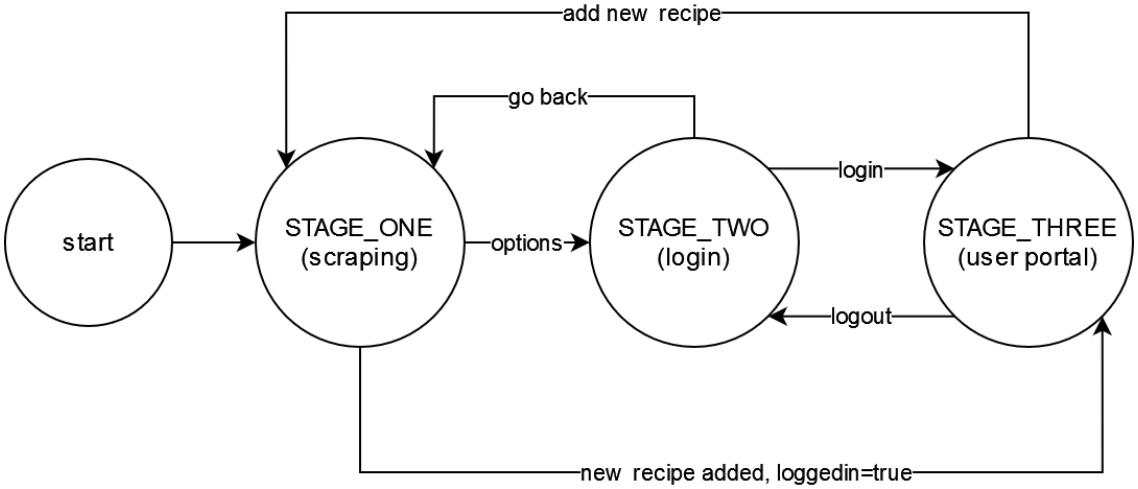
2.2 Design

As the main goal of this project was to create an app that could reliably retrieve recipes from websites, the core problem to be solved was in parsing the websites themselves. However, the app also needed to have additional functions to make it a complete system which themselves required certain design. This will be explained better in the following chapters. We also wanted to implement a bit of security practice, which included things such as password hashing such that the database is not stored in plaintext.

We had also utilized version control through GitHub for this program. This allowed a certain amount of safety net in reworking the code base with the use of rollback and branches which allowed safe development of features.

2.2.1 Program Flow

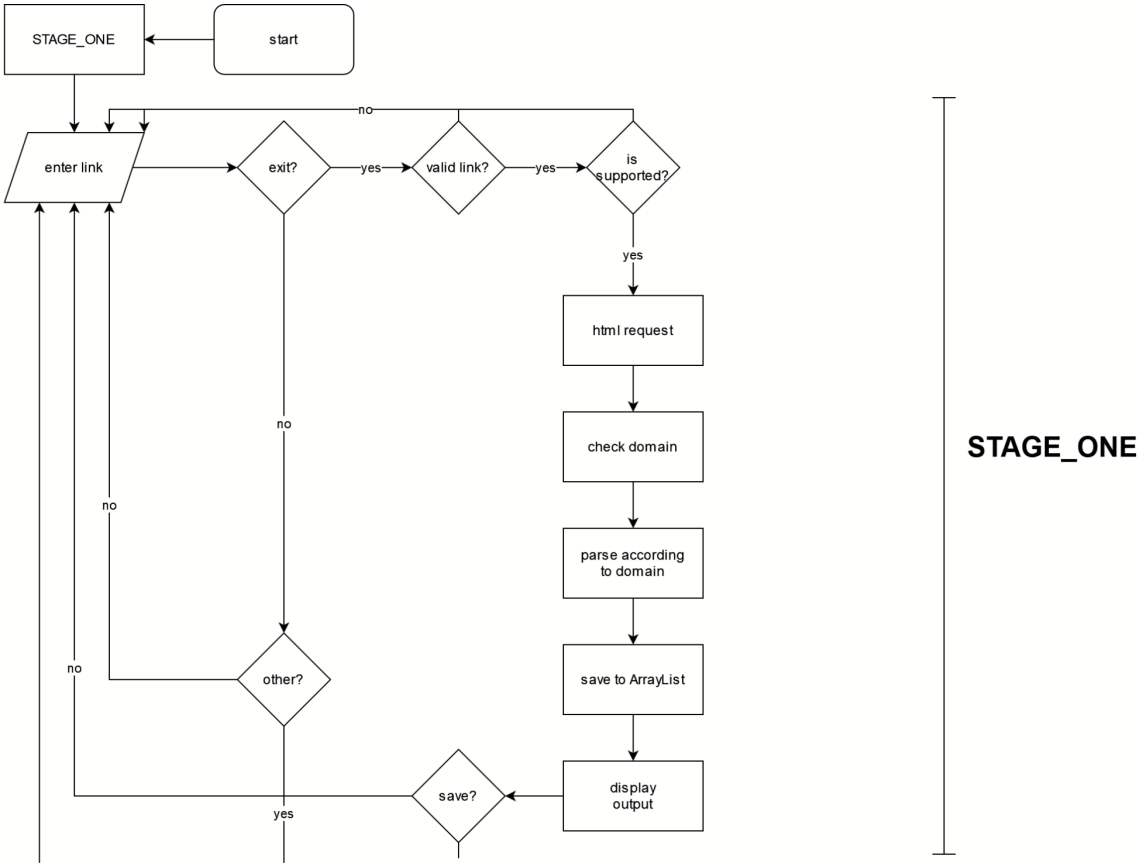
The program was designed with the basic idea of having stages in mind. The stages represent what mode the user is in and what they can do and each stage can transition to the next depending on what activity is done. Using a pseudo state machine chart, the flow in its simplest form can be depicted this way:

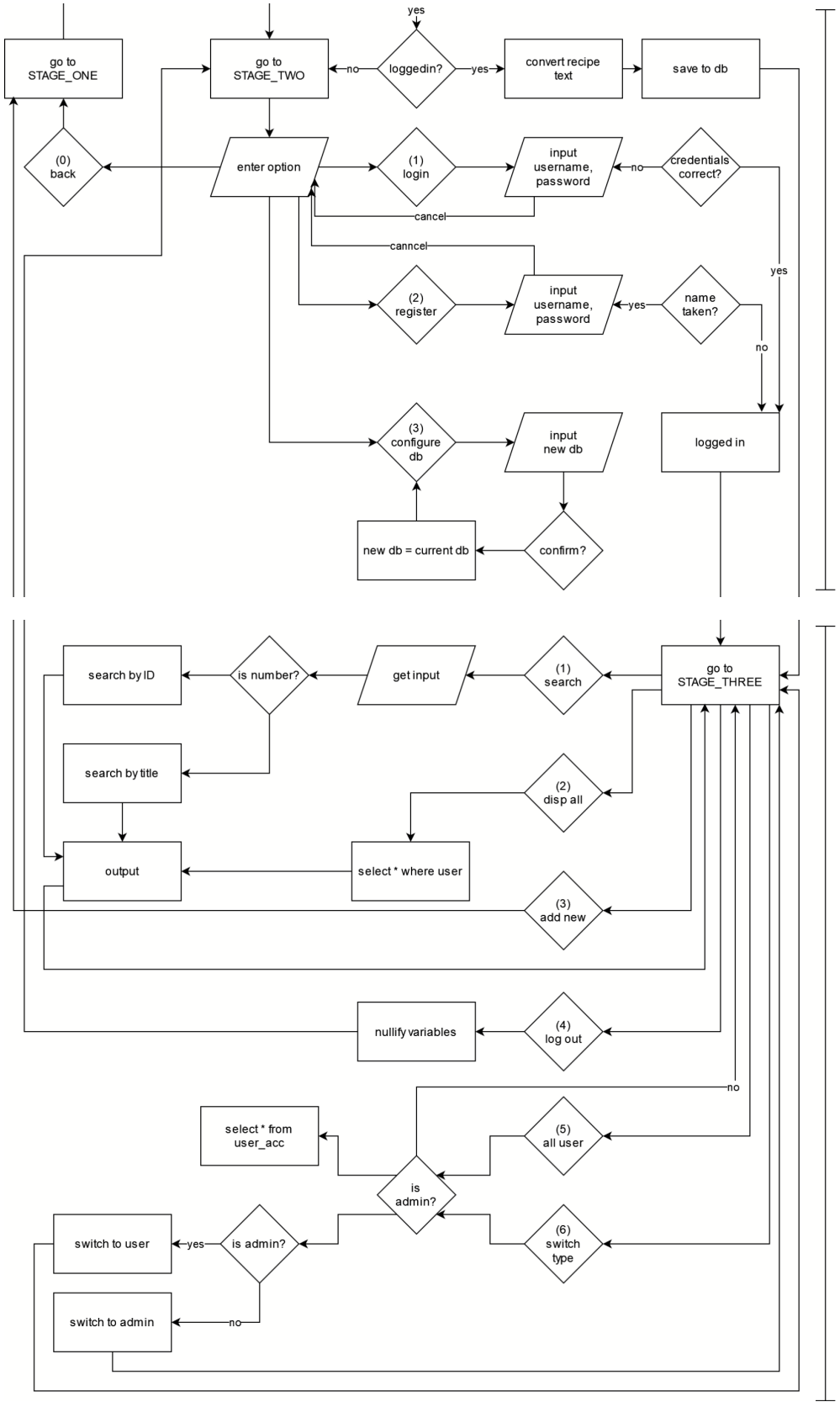


The default state of the program is to start with the scraping. This is because we wanted to put an emphasis on the recipe retrieval as the core aspect. If we wanted to then access the additional features, we can access the second stage which would be the portal for logging in and signing up. Database configuration is also available in-app in case it

needs to be changed. The user can also go back to the previous stage where the link input takes place. After logging in or signing up, the session is then moved to STAGE_THREE where users can retrieve the recipes that have been saved to their accounts. Users can also add new recipes which send them back to STAGE_ONE but will send them back to STAGE_THREE as they have been classified as logged in. If they want to go back to STAGE_TWO then all they need to do is to log out.

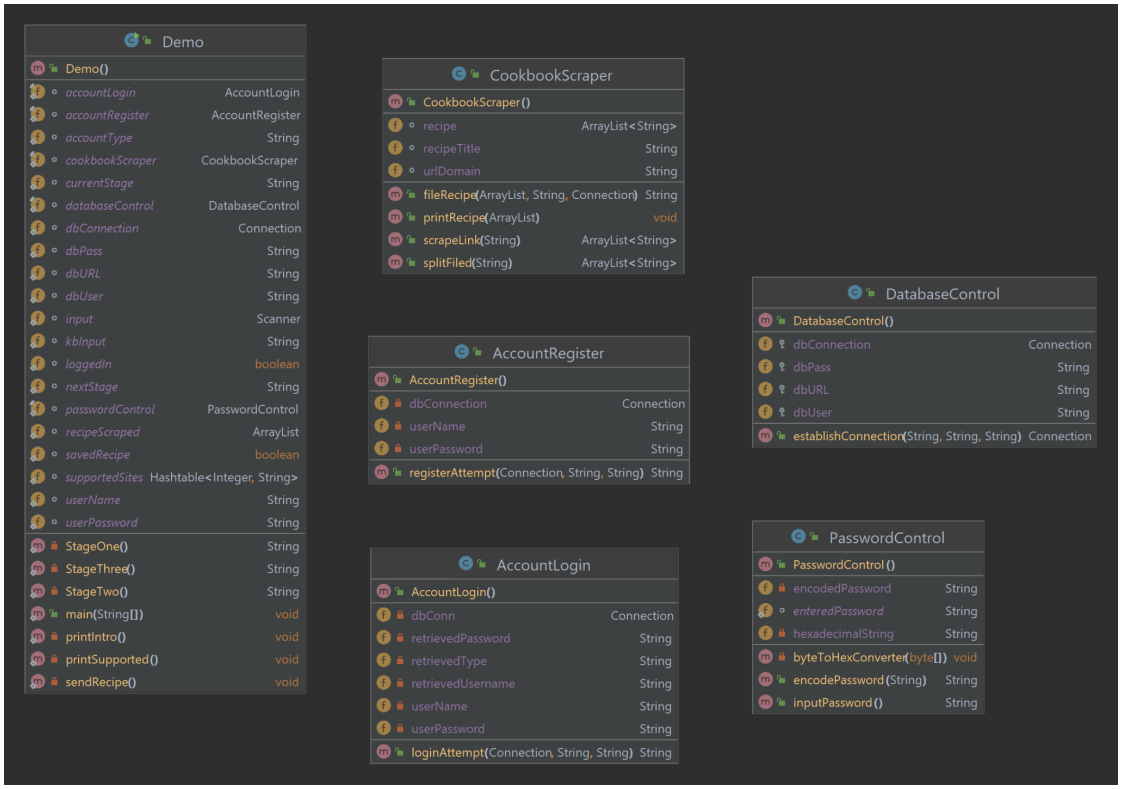
The following is a more detailed and complete program flowchart, broken down in parts:




go to STAGE_ONE
go to STAGE_TWO
yes
loggedin?
no
yes
convert recipe text
save to db
(0) back
enter option
(1) login
input username, password
no
credentials correct?
cancel
canncel
(2) register
input username, password
yes
name taken?
yes
no
(3) configure db
input new db
logged in
new db = current db
confirm?
STAGE_TWO
go to STAGE_THREE
(1) search
get input
is number?
search by ID
search by title
(2) disp all
output
select * where user
(3) add new
nullify variables
(4) log out
no
(5) all user
select * from user_acc
is admin?
(6) switch type
switch to user
yes
is admin?
switch to admin
no
STAGE_THREE

2.2.3 Program Structure

The program itself was also designed to take advantage of Java's OOP principles, which means the program was written to be as modular as possible and reusable. This means that a variety of classes and methods were made in order to keep things from tangling too much and allow for. The following diagram depicts the classes and methods that were written for this project.



Each class outside of the demo is instantiated by the Demo class. Each of them are independent but are often used together, for example when logging in or registering, the password that is typed by the user is then hashed into SHA-256 by the PasswordControl class and passed onto the database. The details of each class and methods will be explained in further detail in the next section.

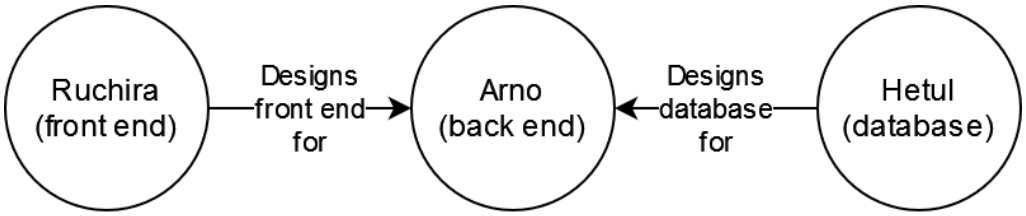
CHAPTER 3
DEVELOPMENT PROCESS

3.1 Task Division

The original task division was created with the idea of maximizing each team member's potential.

- Hetul Patel was stronger academically, but weaker in coding and so was tasked with theory based research and designs.
- Ruchira Bunga was experienced in making designs and UI/UX development based on her internship experience and was thus given the front end portion.
- Aloysius Arno Wiputra had served as the main programmer in a few projects before this, but had no experience in front end coding, thus was best suited for back end engineering.

The plan was to combine all the work into one based on this chart:



Each week there were goals to be met, with the initial division being as follows:

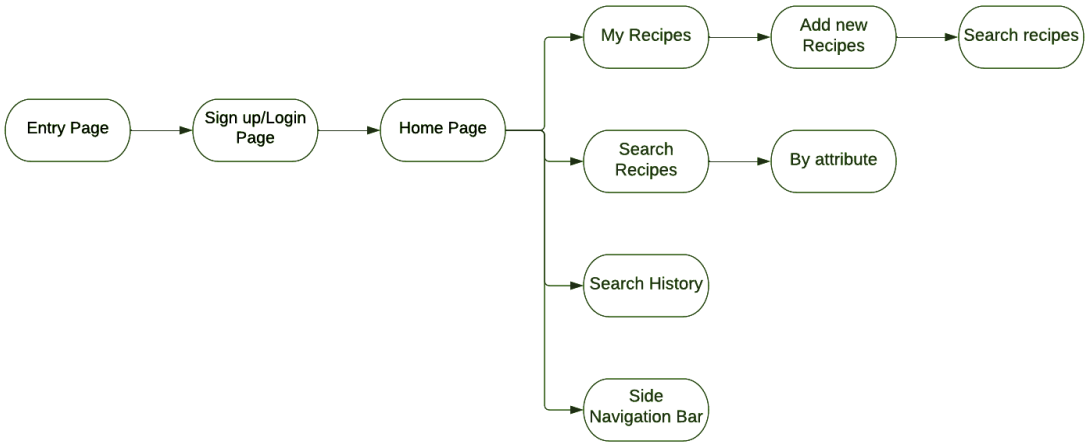
	Date		Goal						Notes
	Start	End		Arno		Hetul		Ruchira	
Week 0	2022/10/31	2022/11/06	Preparation	Finish task division	☐				
Week 1	2022/11/07	2022/11/13	Sign-in	Setup version control solutions	☐	Start user database design	☐	Start making general design language	☐	
				Research web scraping methods	☐	Research encryption methods	☐	Research front end solutions	☐	
								Start design on sign in and sign up page	☐	
Week 2	2022/11/14	2022/11/20	Home & DB	Research cloud database solutions	☐	Start general database design	☐	Start implementing front end solutions	☐	
				Setup remote hosting	☐	Research input sanitation methods	☐	Start work on landing and home page	☐	
				Start work on scraper	☐	Research search engine & recommendation systems	☐			
				Start work on user account control	☐					
Week 3	2022/11/21	2022/11/27	User center	Start work on search engine	☐	Test user sign in+sign up	☐	Start work on history/input/search page	☐	
						Test input sanitation and security features	☐		☐	
Week 4	2022/11/28	2022/12/04	Thanksgiving	Further testing on scraper	☐	Test search engine	☐		☐	
Week 5	2022/12/05	2022/12/11	Additional		☐		☐		☐	

However, not all went according to plan and ultimately it deviated from the original course, factoring things such as other course works and personal lives as well as holidays. Still, a weekly report was filed every week and this table was updated weekly to reflect on the progress made each week:

Task Description	Start Date	Target Date	%Complete	Task Status
Testing database connection	2022/12/5	2022/12/11	95%	Connection and testing through local instance has been made, class made in a way that switching to cloud is feasible
Testing test queries			95%	Putting dummy data in the database and accessing them through the driver program has been successful
Password hashing method and function			100%	Using SHA-256 to hash and encode the data was successful, result came out in 64 digit hexadecimal which was irreversible
Scraper testing			70%	Foundation has been laid, function for one has been done and implementation on the rest should come quickly
Revision on the plan moving forward			50%	In rush mode trying to solve the issues faced and make it in time for the deadline
Redo research on connection method			50%	Looking at GWT, implementation using HTML5 object, and HTML through JSwing
More pages has been made			80%	More revisions might be needed as connection is made and testing is done
Focus shifted to finishing database design			60%	Member is behind on finishing the designs
Revision on front end design, to simplify for time limit	2022/11/28	2022/12/4	50%	Revision has to be made to make sure it's doable within the remaining time
Start work on back end, need to revise for time			70%	A lot of the functionalities have been implemented foundationally, just need to be built up further and more testing
Catching up on previous research and design			50%	Catching up on past weeks
Prepare for midsemester progress report	2022/11/21	2022/11/27	100%	Presentation has been made, committed to repo, shared to members
Continue work on front end implementations			100%	Codebase for individual pages have been made in html and css
Catch up on previous research			80%	Starting to catch up on encryption research
Start work on scraper	2022/11/14	2022/11/20	70%	It is functional on one website, and a switcher + parser has been implemented, just need to repeat the process for more sites
Start work on user account control			50%	Current method of implementation may be limited user account control
Start general database design			50%	Basic design has been made, need to be validated and checked
Research input sanitation methods			50%	Basic research has been done
Research search engine & recommendation systems			50%	Basic research has been done
Start implementing front end solutions			70%	Pages have been made, connection still a big missing part
Start work on landing and home page			50%	Page designs have been made, html/css page have been done, need to implement functionality
Setup version control solutions	2022/11/07	2022/11/13	95%	Online repository has been setup, need to socialize to members
Setup online database hosting			80%	Database hosting is currently local to avoid overage, research has been done
Start user database design			70%	
Research encryption methods			100%	In the end, SHA-256 was chosen as the hashing method due to its irreversability
Start making general design language			100%	Design language document has been made, with general color palette and layouts
Research front end solutions			100%	Languages has been chosen
Start sign in and sign up pages			90%	Page designs have been made, html/css page have been done, need to implement functionality

3.2 Front End Development

With each page meticulously designed and with a structured layout, the first section of the code was the framework. We created a style.css file where we began with importing the fonts and a basic design of the home page. By using the index.html page we confirmed that the CSS code reflected the required design through any browser. The first step was to design the layout, create sections across the page and set transition timings.



The next part was done by creating the color scheme across the pages, the placement of images or buttons and text. Then went on to the specifics of each text - font size, color, if it required a background and so on. Some of the features that were used are

- border radius, padding, margins and many more. All of the images used were created on CANVA and embedded as PNG's into the CSS code.

As the layout goes for all CSS codes - we first declared the HTML element we wanted to structurally design, followed by each incremental property and property value. For instance, below is a snippet of the CSS code that shows the design features of one clickable button on a page.

```
.button {
    font-size: 20px;
    font-family: "mont-thin";
    background-color: #dce7c6;
    padding: 10px 60px;
    border-radius: 25px;
    text-transform: uppercase;
    color: #0000009a;
    font-weight: bold;
    text-align: center;
}
```

In addition to that, we used functions like .button:hover to make each page volatile and user friendly along with setting transitions between each page.

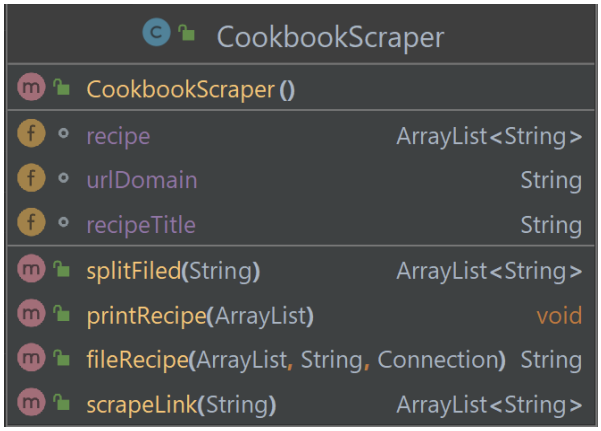
Since we used a set framework for all pages, it was relatively simple to create each page. Furthermore, given that all of the designs were premade, the coding aspect of it was uncomplicated. They all fit the specifications of the design language

Ultimately, however, due to inexperience and failed research we were unable to connect the HTML/CSS framework with Java and thus the program was run through a terminal interface.

3.3 Back End Development

The back end was coded purely in Java. The files were developed following different packages to ensure easier maintenance, and so the breakdown will also follow the packages culminating in the Demo class at the end.

3.3.1 cookbook.webscraper.CookbookScraper



This package consists of only one class, CookbookScraper.java and serves as the class that primarily does the web scraping.

```
package cookbook.webscraper;

import com.gargoylesoftware.htmlunit.*;
import com.gargoylesoftware.htmlunit.html.*;
import org.apache.commons.lang3.StringUtils;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.sql.Connection;
```

The scraping itself is done with the help of a library called htmlunit. It was published under Apache License 2.0 and its core idea is the simulation of web browsers in a Java code. Aside from that, multiple java classes are also used. Notably, the SQL elements which allow storing the scraped information into a database, but not before storing them in an ArrayList. ArrayList was chosen due to its innate property of being faster to parse but slower to modify, as well as its non-reliance on a fixed length unlike regular Arrays.

More notably, the recipes come out as ArrayLists in the main scrapeLink method. This is unsuitable for storage in a database. There are then three complementary methods written. The main one is fileRecipe, which combines the contents of the recipe and separates them using a symbol to then be sent over to the database. If the reverse is desired, that is to read the data from a website, then splitRecipe was written to split the content by the aforementioned symbol. printRecipe then lastly is what was written to output the recipe to a console.

```
public class CookbookScraper
{
    ArrayList<String> recipe = new ArrayList<>(); //chose ArrayList because
faster to parse, only needs to be modified once.
   String recipeTitle;
   String urlDomain;

   public ArrayList<String> scrapeLink(String url)
   {
       int elem;
       /*todo:
           1. parse domain (ok)
           2. make individual ways to scrape (3/3) (ok)
           3. more and more testing
        */
       this.recipe.clear(); //clean the recipe array list for every run
       try
       {
           //System.out.println("A");

                   WebClient webClient = new WebClient(BrowserVersion.CHROME);
//simulate Firefox
```

```
            webClient.getOptions().setCssEnabled(false);
            webClient.getOptions().setThrowExceptionOnFailingStatusCode(false);
            webClient.getOptions().setThrowExceptionOnScriptError(false);
            webClient.getOptions().setPrintContentOnFailingStatusCode(false);
            webClient.getOptions().setJavaScriptEnabled(false);

                                                                    //test      url:
https://www.thespruceeats.com/classic-southern-fried-chicken-3056867
            HtmlPage htmlPage = webClient.getPage(url); //send request to link

            webClient.getCurrentWindow().getJobManager().removeAllJobs();
            webClient.close();
            this.recipeTitle = htmlPage.getTitleText();
            System.out.println("\nTitle: " + this.recipeTitle);
            //System.out.println(htmlPage.asXml());
            /*
            String xPath = "//*[@id=\"mntl-sc-block_3-0-2\"]";
            DomElement element = htmlPage.getFirstByXPath(xPath); //use xpath
            String test = element.getTextContent();
            //String test = element.getAttribute("innerHTML");
            System.out.println(test);
            /*
                List<HtmlAnchor> links = htmlPage.getAnchors();
                for (HtmlAnchor link : links)
                {
                    String href = link.getHrefAttribute();
                    System.out.println("C");
                    System.out.println("Link: " + href);
                }
              */
            boolean elementNotNull = true;

               this.urlDomain = StringUtils.substringBetween(url, "https://www.",
".");
            System.out.println("Domain: " + urlDomain);

            switch(urlDomain)
            {
                case "thespruceeats": //increments by 5, for some reason
                     if(htmlPage.getTitleText().toLowerCase().contains("recipe"))
//check if title contains recipe, kinda basic way of doing it
```

```
                    {
                        elem = 2;
                        while (elementNotNull)
                        {
                             String xPath = "//*[@id=\"mntl-sc-block_3-0-" + elem
+ "\"]";
                                                                DomElement  element  =
htmlPage.getFirstByXPath(xPath); //use xpath
                            if (element != null)
                            {
                                String test = element.getTextContent();
                                test = test.trim();
                                test = test.replace("\n","");
                                //System.out.println(test);
                                                                    //String  test  =
element.getAttribute("innerHTML");
                                if (!test.contains("Tips"))
                                {
                                    recipe.add(test);
                                }
                            }
                            else
                            {
                                elementNotNull = false;
                            }
                            elem = elem + 5;
                        }
                    }
                    else
                    {
                           System.out.println("Domain does not seem to contain a
recipe.");
                    }
                    break;
                case "gimmesomeoven": //most straightforward one
                    elem = 1;
                    while (elementNotNull)
                    {
                           String xPath = "//*[@id=\"instruction-step-" + elem +
"\"]";
                           DomElement element = htmlPage.getFirstByXPath(xPath);
//use xpath
```

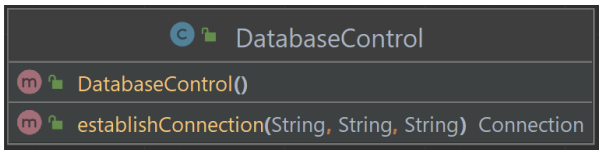
```
                        if (element != null)
                        {
                            String test = element.getTextContent();
                            test = test.trim();
                            test = test.replace("\n","");
                            this.recipe.add(test);
                        }
                        else
                        {
                            elementNotNull = false;
                        }
                        elem++;
                    }
                    break;
                case "playfulcooking": //this one has different structures for
its recipe, requires unique ID
                    elem = 1;
                    while (elementNotNull)
                    {
                                        //  //*[@id="wprm-recipe-23513-step-0-0"]
//*[@id="wprm-recipe-23545-step-0-0"]
                                                                                //
/html/body/div[1]/div/div/div/article/div[3]/div[2]/div[2]/div/div[10]/div/ul/l
i[1]/div
                                                                                //
/html/body/div[1]/div/div/div/article/div[3]/div[2]/div[2]/div/div[10]/div/ul/l
i[2]/div


                                                                                //
/html/body/div[1]/div/div/div/article/div[3]/div[4]/div/div[12]/div/ul/li[1]/di
v/span

                                                            //String   xPath   =
"/html/body/div[1]/div/div/div/article/div[3]/div[2]/div[2]/div/div[10]/div/ul/
li[" + elem + "]/div";

                        //*[@id="wprm-recipe-container-23545"]

                                                          DomElement   testElem   =
htmlPage.getFirstByXPath("//*[@data-recipe-id]");
                                                              String   valueID   =
testElem.getAttribute("data-recipe-id");
```

```
                        //System.out.println(valueID); //this took so much time
                             String xPath = "//*[@id=\"wprm-recipe-" + valueID +
"-step-0-" + elem + "\"]";
                            DomElement element = htmlPage.getFirstByXPath(xPath);
//use xpath
                        //*[@id="wprm-recipe-23545-step-0-0"]
                        //*[@id="wprm-recipe-23545-step-0-0"]/div/span/text()

                        if (element != null)
                        {
                            String test = element.getTextContent();
                            test = test.trim();
                            test = test.replace("\n","");
                            this.recipe.add(test);
                        }
                        else
                        {
                            elementNotNull = false;
                        }
                        elem++;
                    }
                    break;
                default:
                       System.out.println("Error: invalid input (domain not found
or not in the list.)");
            }
        }
        catch(Exception e)
        {
            System.out.println("Exception: " + e);
        }
        //printRecipe(recipe);
        return recipe;
    }
    public void printRecipe(ArrayList currRecipe)
    {
        if(!currRecipe.isEmpty())
        {
            System.out.println("\nRecipe:");
            int listLength = currRecipe.size();
            for (int i = 0; i < listLength; i++)
            {
```

```
                System.out.print(i + 1 + ") " + currRecipe.get(i) + "\n");
            }
        }
    }
     public String fileRecipe(ArrayList filedRecipe, String user, Connection
dbConnection)
    {
        String resultCode = "";
        try
        {
                                                        Statement     stm     =
dbConnection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);

                 String queryCheckData = "SELECT RECIPE_ID FROM RECIPES WHERE
RECIPE_TITLE='" + recipeTitle + "'";
            ResultSet queryCheckIfFilled = stm.executeQuery(queryCheckData);
            if (queryCheckIfFilled.next())
            {
                resultCode = "RECIPE_EXISTS";
            }
            else
            {
                int listLength = filedRecipe.size();
                String currentFiled = "";
                String nextFiled;
                for (int i = 0; i < listLength; i++)
                {
                    if (!currentFiled.equals(""))
                    {
                        nextFiled = filedRecipe.get(i).toString();
                        currentFiled = currentFiled + "@" + nextFiled;
                    }
                    else
                    {
                        currentFiled = filedRecipe.get(i).toString();
                    }
                }
                                            String   queryFile   =   "INSERT   INTO
RECIPES(SOURCE_USER,RECIPE_TITLE,RECIPE_SOURCE,RECIPE_CONTENT)   VALUES   ('"   +
user + "','" + recipeTitle + "','" + urlDomain + "',\"" + currentFiled + "\")";
                stm.executeUpdate(queryFile);
```

```
                resultCode = "RECIPE_ADDED";
            }
        }
        catch(SQLException e)
        {
            System.out.println(e);
        }
        return resultCode;
    }
    public ArrayList<String> splitFiled(String filedRecipe)
    {
        String splitRaw[] = filedRecipe.split("@");
        List<String> splitRecipe = new ArrayList<>(Arrays.asList(splitRaw));
        return (ArrayList<String>) splitRecipe;
    }
}
```

3.3.2 cookbook.dbconnection.DatabaseControl



The DatabaseControl class was written to make sure that there is an ease to connect the database and establish Connection parameters everytime it is needed, rather than having to call it again every time.

```
package cookbook.dbconnection;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DatabaseControl
{
    protected String dbURL;
    protected String dbUser;
    protected String dbPass;
    protected Connection dbConnection;

    //getter -> return connection
    //setter -> establish connection
    //nvm, consolidate
```

```
    public Connection establishConnection(String url, String user, String pass)
throws SQLException //setter, needs url input
   {
       try
       {
           this.dbURL = "jdbc:mysql://" + url;
           this.dbUser = "?&user=" + user;
           this.dbPass = "&pass=" + pass;
       }
       catch (Exception e)
       {
           System.out.println("Error in setting connection values: " + e);
       }
       //jdbc:mysql://localhost:3306/?user=root
       String connectionString = this.dbURL + this.dbUser + this.dbPass;
       try
       {
           Class.forName("com.mysql.cj.jdbc.Driver");
           dbConnection = DriverManager.getConnection(connectionString);
       }
       catch (Exception e)
       {
           System.out.println("Error in establishing connection: " + e);
       }
       System.out.println(dbConnection);
       if(dbConnection != null)
       {
           return dbConnection;
       }
       else
       {
           System.out.println("No connection established");
           return null;
       }
   }
   /* redundant, mixed with connection
     public Connection establishConnection() //getter, return the connection
value
   {
       if(dbConnection != null)
       {
           return dbConnection;
```

```
        }
        else
        {
            System.out.println("No connection established");
            return null;
        }
    }*/
}
```

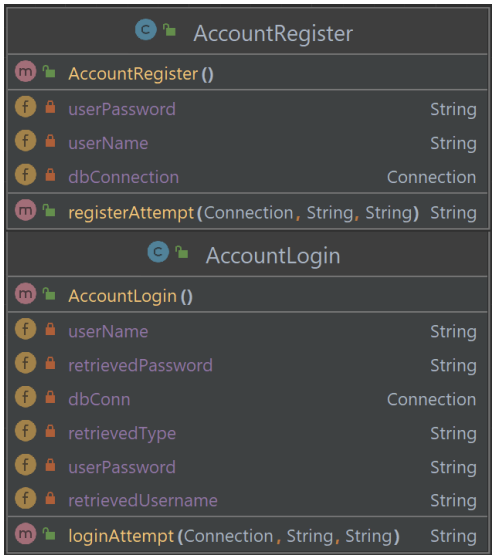
3.3.3.1 cookbook.accountcontrol.AccountLogin

The AccountLogin class checks with the SQL database to make sure that the data entered is correct. It also provides its own reader for password using console so that password data is obscured.

```
package cookbook.accountcontrol;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class AccountLogin //might be obsolete if we simplify the flow
{
    private String userName;
    private String userPassword;
    private Connection dbConn;

    private String retrievedPassword;
    private String retrievedUsername;
    private String retrievedType;

    public String loginAttempt(Connection dbConn, String name, String password)
    {
        this.userName = name;
        this.userPassword = password;
        try
        {
                                                    Statement     stm     =
dbConn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
            String queryPass = "SELECT * FROM USER_ACCOUNT WHERE USER_NAME = '"
+ name + "'";
```

```
            ResultSet queryResult = stm.executeQuery(queryPass);
            if(queryResult.next()) //check if entry is empty, if it is then exit
            {
                queryResult.first();
                this.retrievedUsername = queryResult.getString("USER_NAME");
                this.retrievedPassword = queryResult.getString("USER_PASSWORD");
                this.retrievedType = queryResult.getString("USER_TYPE");
                //System.out.println(password);
                //System.out.println(retrievedPassword);
            }
        }
        catch (SQLException e)
        {
            System.out.println(e);
        }
        if(retrievedPassword==null)
        {
            return "USER_NOT_FOUND";
        }
        else if (!userPassword.equals(retrievedPassword))
        {
            return "PASSWORD_INCORRECT";
        }
                            else    if(userPassword.equals(retrievedPassword)    &&
userName.equalsIgnoreCase(retrievedUsername))
        {
            return retrievedType;
        }
        else
        {
            return "UNKNOWN";
        }
    }
}
```

3.3.3.2 cookbook.accountcontrol.AccountRegister



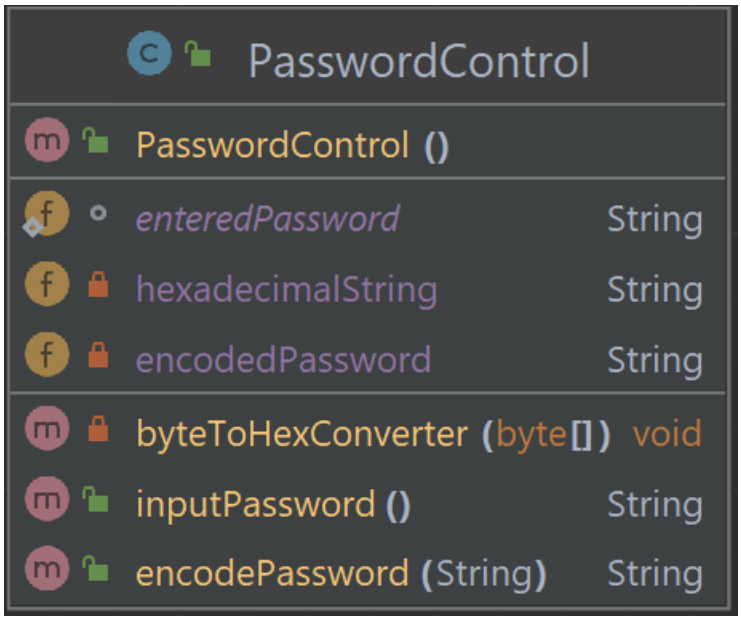
AccountRegister works similarly to AccountLogin, except it checks for whether the username has been taken or not.

```
package cookbook.accountcontrol;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

public class AccountRegister
{
   private String userName;
   private String userPassword;
   private Connection dbConnection;

      public String registerAttempt(Connection dbConn, String name, String
password)
   {
       String result = "UNKNOWN";
       this.userName = name;
       this.userPassword = password;
       this.dbConnection = dbConn;
       try
       {
```

```
                                                    Statement     stm     =
dbConnection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
              String queryCheckData = "SELECT USER_NAME FROM USER_ACCOUNT WHERE
USER_NAME='" + name + "'";
            ResultSet queryCDResult = stm.executeQuery(queryCheckData);
            //System.out.println("A");
            if(!queryCDResult.next())
            {
                        System.out.println("Username  available.  Confirm  account
creation? (y/n)");
                System.out.print("Input: ");
                Scanner input = new Scanner(System.in);
                String regInput = input.next();
                if(regInput.equalsIgnoreCase("y"))
                {
                                        String  queryInsertData  =  "INSERT  INTO
USER_ACCOUNT(USER_NAME,USER_PASSWORD,USER_TYPE)  VALUES('"  +  userName  +  "','"  +
userPassword + "','USER')"; //create account
                    //System.out.println("B");
                    stm.executeUpdate(queryInsertData);
                    //System.out.println("C");
                                                ResultSet   queryCheckAgain   =
stm.executeQuery(queryCheckData); //check if user exists
                      if (queryCheckAgain.next()) //if it exists, then it's true
that next value exist
                    {
                        result = "ACCOUNT_CREATION_SUCCESS";
                    }
                    else
                    {
                        System.out.println("An error occurred.");
                    }
                }
                else
                {
                    result="REGISTRATION_CANCELLED";
                }
            }
            else
            {
                System.out.println("Username already taken.");
```

```
                result = "USERNAME_TAKEN";
            }
        }
        catch (SQLException e)
        {
            System.out.println(e);
        }
        return result;
    }
}
```

3.3.3.3 cookbook.accountcontrol.PasswordControl



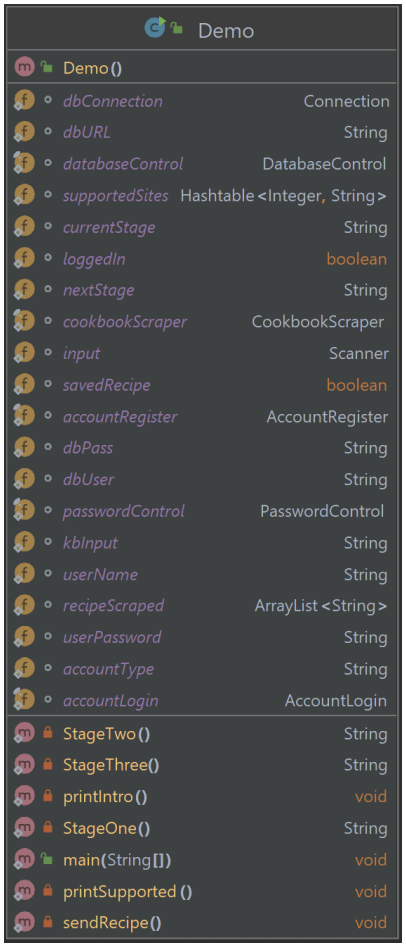
This class is where the passwords are returned to be hashed using a built-in Java function of MessageDigest. The class initially returns it in a hex form, which is then converted into hexadecimal for storage. This allows for password storage where even if someone gains access to the database, it will be nearly impossible to reverse the hash into plaintext.

```
package cookbook.accountcontrol;

import java.io.Console;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.nio.charset.StandardCharsets;
import java.util.Scanner;
```

```
public class PasswordControl
{
   private String hexadecimalString;
   static String enteredPassword;
   private String encodedPassword;
   public String encodePassword(String userPassword)
   {
       MessageDigest digest = null;
       try
       {
             digest = MessageDigest.getInstance("SHA-256"); //built in class to
hash string into SHA-256 hash
       }
       catch (NoSuchAlgorithmException e)
       {
           System.out.println(e);
       }
                                            byte[]      encodedHashString      =
digest.digest(userPassword.getBytes(StandardCharsets.UTF_8));
       byteToHexConverter(encodedHashString);
       return hexadecimalString;
   }
    private void byteToHexConverter(byte[] hash) //convert hash into string to
be able to be stored
   {
       StringBuilder hexString = new StringBuilder(2 * hash.length);
       for (int i = 0; i < hash.length; i++)
       {
           String hex = Integer.toHexString(0xff & hash[i]);
           if(hex.length() == 1) {
               hexString.append('0');
           }
           hexString.append(hex);
       }
       this.hexadecimalString = hexString.toString();
   }
   public String inputPassword()
   {
       //todo: switch the commented out block when building artifacts
       //Console doesn't work for some reason in IDE
```

```
        Console console = System.console();
        System.out.println("\nPlease input your password.\n");
        enteredPassword = new String(console.readPassword("Input: "));

        /*
        Scanner passInput = new Scanner(System.in);
        System.out.print("\nPlease input your password.\nInput: ");
        enteredPassword = passInput.next();
        */

        encodedPassword = encodePassword(enteredPassword);
        return encodedPassword;
    }
}
```

3.3.4 cookbook.Demo



The Demo program was written as a demonstration of the features and is a substitute terminal interface for our failure to connect the front end and establish it as a web app. The software is at least still usable in a terminal interface. The main method simply acts as a handler for the stage methods, which are separated to ensure it can be developed isolated from another, ensuring code readability and easier maintenance.

A majority of the variables are written as a static variable as it will only be used in this main class and allows reuse without having to create a new variable or instance every time.

```
package cookbook;

import cookbook.accountcontrol.AccountLogin;
import cookbook.accountcontrol.AccountRegister;
import cookbook.dbconnection.DatabaseControl;
import cookbook.webscraper.CookbookScraper;
import cookbook.accountcontrol.PasswordControl;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Hashtable;
import java.util.Scanner;
import java.util.ArrayList;
import java.sql.Connection;

import de.vandermeer.asciitable.AsciiTable;

public class Demo
{
   /*todo (must haves):
       1. Functional web scraper that works on >=3 different websites (ok)
       2. Database for recipe storage and unique ID for each recipe (ok)
         3. Driver program with read and write control for the database, with
input sanitation (ok, partially)
       4. Version control integration (ok)
       5. Cloud database hosting
       6. Configure db in app (ok)
       7. Admin access (ok)
    */
```

```
final static CookbookScraper cookbookScraper = new CookbookScraper();
final static PasswordControl passwordControl = new PasswordControl();
final static DatabaseControl databaseControl = new DatabaseControl();

final static AccountLogin accountLogin = new AccountLogin();
final static AccountRegister accountRegister = new AccountRegister();

static Scanner input = new Scanner(System.in);
static String kbInput;

static String dbURL = "localhost:3306/cookbook";
static String dbUser = "root";
static String dbPass = "pass";


static boolean loggedIn = false;

static Hashtable<Integer, String> supportedSites = new Hashtable<>();

static ArrayList recipeScraped = new ArrayList<>();
static boolean savedRecipe = false;

static Connection dbConnection;

static String userName;
static String userPassword;
static String accountType;

static String nextStage;
static String currentStage = "STAGE_ONE";

public static void main(String[]args)
{
    Demo.printIntro();
    boolean appRunning = true;
    do
    {
        if(nextStage!=null)
        {
            currentStage=nextStage;
        }
        switch(currentStage)
```

```
        {
            case("STAGE_ONE"):
            {
                Demo.StageOne();
                break;
            }
            case("STAGE_TWO"):
            {
                Demo.StageTwo();
                break;
            }
            case("STAGE_THREE"):
            {
                Demo.StageThree();
                break;
            }
            case("STAGE_EXIT"):
            {
                appRunning = false;
                break;
            }
        }

    } while(appRunning);
}
private static void printIntro()
{
    System.out.println("""
                 *@@,                          ,@@    \s
                 *@@@@@@@@@@         @@@@@@@@@@    \s
             @@@ *@@@@@@@@@@@@@   @@@@@@@@@@@@@  @@@
            @@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
            @@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
            @@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
            @@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
            @@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
            @@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
            @@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
            @@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
            @@@@    %@@@@@@@@@@/ @@@@@@@@@@@#    @@@
            @@@@@@@@@.    @@@@@/ @@@@@%    /@@@@@@@@
            """);
```

```
        System.out.println("""
                Welcome to Cookbook.
                This project was brought to you by:

                Team leader:
                \t1. Aloysius Arno Wiputra
                Team members:
                \t1. Ruchira Bunga
                \t2. Hetul Patel

                Instructor\t: Dr. Frank Lee
                Class\t\t: CSCI 455 - Senior Project
                """);

        printSupported();

        System.out.println("\n-----------------------------------------");
    }
    private static void printSupported()
    {
           supportedSites.put(0, "gimmesomeoven"); //fully supported, such a nice easy site
            supportedSites.put(1, "playfulcooking"); //seems okay, after some tinkering
           supportedSites.put(2, "thespruceeats"); //not terribly effective, only works with some links

        System.out.println("Current supported sites: ");
        for(int i=0; i<supportedSites.size(); i++)
        {
            System.out.println(i+1 + ") " + supportedSites.get(i));
        }
    }
    private static String StageOne()
    {
           System.out.println("\nPlease input the recipe's link.\nFor additional options, please input 'other'.");
        System.out.print("\nInput: ");
        savedRecipe = false;
        kbInput = input.next();
        if(kbInput.equalsIgnoreCase("exit"))
```

```
        {
            nextStage = "STAGE_EXIT";
        }
        else if(kbInput.contains("http"))
        {
            boolean isSupported = false;
            for(int i=0; i<supportedSites.size(); i++)
            {
                if(kbInput.contains(supportedSites.get(i)))
                {
                    isSupported = true;
                }
            }
            if(isSupported)
            {
                recipeScraped = cookbookScraper.scrapeLink(kbInput);
                cookbookScraper.printRecipe(recipeScraped);
                      System.out.println("\nWould you like to save this recipe?
(y/n)");
                System.out.print("Input: ");
                String saveInput = input.next();
                if (saveInput.equalsIgnoreCase("y"))
                {
                    savedRecipe = true;
                    if(loggedIn)
                    {
                        sendRecipe();
                    }
                    //System.out.println(filedRecipeRaw);
                    else
                    {
                        System.out.println("\nYou are not logged in.");
                        nextStage = "STAGE_TWO";
                    }
                }
            }
            else
            {
                System.out.println("Site isn't supported.");
            }
        }
        else if(kbInput.equalsIgnoreCase("other") || savedRecipe)
```

```
        {
            nextStage = "STAGE_TWO";
        }
        else
        {
            System.out.println("Input is invalid. Please try again.");
        }
        if(loggedIn)
        {
            nextStage="STAGE_THREE";
        }
        return nextStage;
    }
    private static String StageTwo()
    {
            System.out.println("\nIf you would like to retrieve your existing recipes or save them, please log in. If you do not have an account, feel free to make one.\n");
           System.out.println("Options:\n(0) Back\n(1) Log in\n(2) Sign up\n(-) Configure database connection");

        System.out.print("\nInput: ");
        String optInput = input.next();
        switch (optInput)
        {
            case("exit"):
            {
                nextStage = "STAGE_EXIT";
                break;
            }
            case("0"):
                nextStage = "STAGE_ONE";
                break;
            case("1"):
                boolean attemptingLogin = true;
                   System.out.println("\nYou have opted to log in.\nInput '0' to cancel.");
                try
                {
                        dbConnection = databaseControl.establishConnection(dbURL, dbUser, dbPass);
                    while (attemptingLogin)
```

```
                    {
                          System.out.print("\nPlease input your username.\nInput:
");
                        userName = input.next();
                        if (userName.equals("0"))
                        {
                            attemptingLogin = false;
                        }
                        else
                        {
                            //System.out.println("Username: " + userName);
                            userPassword = passwordControl.inputPassword();
                            //System.out.println("Password: " + userPassword);
                                                                String  loginStatus  =
accountLogin.loginAttempt(dbConnection, userName, userPassword);
                            switch (loginStatus)
                            {
                                case ("USER_NOT_FOUND"): //user not found
                                          System.out.println("Error: username not
found.\nTry again.");
                                    break;
                                case ("PASSWORD_INCORRECT"):
                                           System.out.println("Error: password is
incorrect.\nTry again.");
                                    break;
                                case ("ADMIN"):
                                case ("USER"):
                                    accountType = loginStatus;
                                    loggedIn = true;
                                      System.out.println("Successfully logged in,
logged in as '" + userName + "'.");
                                    attemptingLogin = false;
                                    nextStage="STAGE_THREE";
                                    break;
                                default:
                                         System.out.println("Error: unknown error
occurred.\nTry again.");
                                    break;
                            }
                        }
                    }
                }
```

```
                catch (SQLException e)
                {
                    System.out.println("Error: " + e);
                }
                break;
            case("2"):
                boolean attemptingRegister = true;
                  System.out.println("\nYou have opted to register.\nInput '0' to
cancel.");
                try
                {
                        dbConnection = databaseControl.establishConnection(dbURL,
dbUser, dbPass);
                    while(attemptingRegister)
                    {
                          System.out.print("\nPlease input your username.\nInput:
");
                        userName = input.next();
                        if(userName.equals("0"))
                        {
                            attemptingRegister=false;
                        }
                        else
                        {
                            //System.out.println("Username: " + userName);
                            userPassword = passwordControl.inputPassword();
                            //System.out.println("Password: " + userPassword);
                                                    String  registerStatus  =
accountRegister.registerAttempt(dbConnection, userName, userPassword);
                            switch (registerStatus)
                            {
                                case("USERNAME_TAKEN"): //user not found
                                      System.out.println("Error: username already
taken.\nTry again.");
                                    break;
                                case("ACCOUNT_CREATION_SUCCESS"):
                                     System.out.println("Successfully registered,
logged in as '" + userName + "'.");
                                    loggedIn=true;
                                    attemptingRegister=false;
                                    accountType="USER";
                                    nextStage="STAGE_THREE";
```

```
                                    break;
                                case("REGISTRATION_CANCELLED"):
                                            System.out.println("Account creation
aborted.");
                                    break;
                                default:
                                {
                                        System.out.println("Error: unknown error
occurred.\nTry again.");
                                    break;
                                }
                            }
                        }
                    }
                catch (SQLException e)
                {
                    System.out.println("Error: " + e);
                }
                break;
            case("-"):
                  System.out.println("\nCurrent database configuration:\nURL: " +
dbURL + "\nUser: " + dbUser + "\nPass: " + dbPass);
                System.out.println("\nReconfigure? (y/n)");
                System.out.print("Input: ");
                String userReconfig = input.next();
                String dbConfig;
                if(userReconfig.equalsIgnoreCase("y"))
                {
                    System.out.println("\nInput your new configurations: ");

                    System.out.print("URL: ");
                    String newdbURL = input.next();

                    System.out.print("User: ");
                    String newdbUser = input.next();

                    System.out.print("Pass: ");
                    String newdbPass = input.next();

                      System.out.println("\nNew database configuration:\nURL: " +
newdbURL + "\nUser: " + newdbUser + "\nPass: " + newdbPass);
```

```
                    System.out.println("\nConfirm changes? (y/n)");
                    String userConfChanges = input.next();
                    if(userConfChanges.equalsIgnoreCase("y"))
                    {
                        dbURL = newdbURL;
                        dbUser = newdbUser;
                        dbPass = newdbPass;
                    }
                    else
                    {
                        System.out.println("Changes not saved.");
                    }
                }
                break;
            default:
            {
                System.out.println("Error: invalid input. Try again.\n");
                break;
            }
        }
        if(loggedIn && savedRecipe)
        {
            try
            {
                sendRecipe();
            }
            catch (Exception e)
            {
                System.out.println(e);
            }
        }
        return nextStage;
    }
    private static String StageThree()
    {
        /*todo:
                1. retrieve recipe (go through resultset, see if it contains
string), print ascii table? (ok)
            2. print all recipes saved (id, title, domain source) (ok)
            2. save recipe (go back to stage one) (ok)
            3. log out (nullify values, go back to stage two) (ok)
            4. if admin, see user tables, promote to admin
```

```
        */

       System.out.println("\nWelcome, " + userName + ".");
               System.out.println("\nAvailable commands: \n(1) Search saved
recipes\n(2) See all recipes\n(3) Add new recipes\n(4) Log out");
       if(accountType.equalsIgnoreCase("Admin"))
       {
             System.out.println("(5) See list of users\n(6) Switch user account
privilege");
       }
       System.out.print("\nInput: ");
       String s3Input = input.next();
       switch(s3Input)
       {
           case("exit"):
           {
               nextStage = "STAGE_EXIT";
               break;
           }
           case("1"): //search recipes
           {
               System.out.println("\nYou can search by title or recipe ID.");
               System.out.print("\nInput: ");
               kbInput = input.next();
                if(kbInput.matches("^[0-9]*$")) //if it's number, then search by
ID
               {
                   try
                   {
                                                                   Statement stm =
dbConnection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
                                       String queryOptOneID = "SELECT RECIPE_TITLE,
RECIPE_SOURCE, RECIPE_CONTENT FROM RECIPES WHERE SOURCE_USER = '" + userName +
"' AND RECIPE_ID = '" + kbInput + "'";
                                                             ResultSet optOneIDResult =
stm.executeQuery(queryOptOneID);
                           if(optOneIDResult.next())
                           {
                               optOneIDResult.first();
                                                              String recipeToParseRaw =
optOneIDResult.getString("RECIPE_CONTENT"); //returned file is in raw format
```

```
                                                                    recipeScraped  =
cookbookScraper.splitFiled(recipeToParseRaw); //splits the text into array list
                            cookbookScraper.printRecipe(recipeScraped);
                        }
                        else
                        {
                            System.out.println("\nNo result found.");
                        }
                    }
                    catch(SQLException e)
                    {
                        System.out.println(e);
                    }
                }
                else
                {
                    try
                    {
                                                                Statement   stm   =
dbConnection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
                                String queryOptOneTitle = "SELECT RECIPE_TITLE,
RECIPE_SOURCE,  RECIPE_CONTENT  FROM  RECIPES  WHERE  RECIPE_TITLE  LIKE  '%"  +
kbInput + "%'";
                             String queryGetCount = "SELECT COUNT * FROM RECIPES
WHERE RECIPE_TITLE LIKE '%" + kbInput + "%'";
                        ResultSet optOneCount = stm.executeQuery(queryGetCount);
                        int count = optOneCount.getInt(1);
                        System.out.println(count);
                        if(count!=0)
                        {
                                                ResultSet  optOneTitleResult  =
stm.executeQuery(queryOptOneTitle);
                            if (count == 1)
                            {
                                                System.out.println("\nTitle:  " +
optOneTitleResult.getString("RECIPE_TITLE"));
                                                System.out.println("\nSource:  " +
optOneTitleResult.getString("RECIPE_SOURCE"));
                                optOneTitleResult.first();
                                                String  recipeToParseRaw  =
optOneTitleResult.getString("RECIPE_CONTENT"); //returned file is in raw format
```

```
                                                                    recipeScraped  =
cookbookScraper.splitFiled(recipeToParseRaw); //splits the text into array list
                                cookbookScraper.printRecipe(recipeScraped);
                            }
                            else if (count > 1)
                            {
                                AsciiTable optOneTable = new AsciiTable();
                                optOneTable.addRule();
                                optOneTable.addRow("ID", "TITLE","SOURCE");
                                optOneTable.addRule();
                                while(optOneTitleResult.next())
                                {

optOneTable.addRow(optOneTitleResult.getString("RECIPE_ID"),optOneTitleResult.g
etString("RECIPE_TITLE"),optOneTitleResult.getString("RECIPE_SOURCE"));
                                    optOneTable.addRule();
                                }
                                                    String optOneTitleResultPrint =
optOneTable.render();
                                System.out.println(optOneTitleResultPrint);
                            }
                        }
                        else
                        {
                            System.out.println("\nNo result found.");
                        }
                    }
                    catch(SQLException e)
                    {
                        System.out.println(e);
                    }
                }
                break;
            }
            case("2"): //see all
            {
                try
                {
                                                                Statement   stm   =
dbConnection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
```

```
                        String queryOptTwo = "SELECT RECIPE_ID, RECIPE_TITLE,
RECIPE_SOURCE FROM RECIPES WHERE SOURCE_USER = '" + userName + "'";
                    ResultSet optTwoResult = stm.executeQuery(queryOptTwo);

                    AsciiTable optTwoTable = new AsciiTable();
                    optTwoTable.addRule();
                    optTwoTable.addRow("ID", "TITLE","SOURCE");
                    optTwoTable.addRule();
                    while(optTwoResult.next())
                    {

optTwoTable.addRow(optTwoResult.getString("RECIPE_ID"),optTwoResult.getString("
RECIPE_TITLE"),optTwoResult.getString("RECIPE_SOURCE"));
                        optTwoTable.addRule();
                    }
                    String optTwoTablePrnt = optTwoTable.render();
                    System.out.println(optTwoTablePrnt);
                }
                catch(SQLException e)
                {
                    System.out.println(e);
                }
                break;
            }
            case("3"): //add new
            {
                nextStage = "STAGE_ONE";
                break;
            }
            case("4"): //log out
            {
                System.out.println("\nConfirm log out? (y/n)");
                System.out.print("Input: ");
                kbInput = input.next();
                if(kbInput.equalsIgnoreCase("y"))
                {
                    userName=null;
                    userPassword=null;
                    loggedIn=false;
                    try
                    {
                        dbConnection.close();
```

```
                    }
                    catch(SQLException e)
                    {
                        System.out.println(e);
                    }
                    nextStage="STAGE_TWO";
                }
                break;
            }
            case("5"): //admin, see user list
            {
                if(accountType.equalsIgnoreCase("Admin"))
                {
                    try
                    {
                                                                Statement   stm   =
dbConnection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
                        String query = "SELECT * FROM USER_ACCOUNT";
                        ResultSet resultQuery = stm.executeQuery(query);

                        AsciiTable userList = new AsciiTable();
                        userList.addRule();
                        userList.addRow("USERS", "TYPE", "PASSWORD");
                        userList.addRule();
                        while (resultQuery.next())
                        {
                              userList.addRow(resultQuery.getString("USER_NAME"),
resultQuery.getString("USER_TYPE"),resultQuery.getString("USER_PASSWORD"));
                            userList.addRule();
                        }
                        String userListTable = userList.render();
                        System.out.println(userListTable);
                    }
                    catch (SQLException e)
                    {
                        System.out.println(e);
                    }
                }
                break;
            }
            case("6"): //admin, promote userf
```

```
            {
                if(accountType.equalsIgnoreCase("Admin"))
                {
                          System.out.println("\nInput the user to switch account
type.");
                    System.out.print("\nInput: ");
                    String userElevate = input.next();
                    try
                    {
                                                                    Statement   stm   =
dbConnection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);

                         String query = "SELECT USER_TYPE FROM USER_ACCOUNT WHERE
USER_NAME = '" + userElevate + "'";
                        ResultSet resultQuery = stm.executeQuery(query);
                        if(resultQuery.next())
                        {
                            resultQuery.first();
                                                                String  resultType   =
resultQuery.getString("USER_TYPE");

                             System.out.println("User '" + userElevate + "' = " +
resultType);
                            System.out.println("Switch types? (y/n)");
                            System.out.print("\nInput: ");
                            kbInput = input.next();
                            if(kbInput.equalsIgnoreCase("y"))
                            {
                                if(resultType.equalsIgnoreCase("admin"))
                                {
                                    resultType="USER";
                                }
                                else if(resultType.equalsIgnoreCase("user"))
                                {
                                    resultType="ADMIN";
                                }
                                    String querySwitch = "UPDATE USER_ACCOUNT SET
USER_TYPE='" + resultType + "' WHERE USER_NAME='" + userElevate + "'";
                                stm.executeUpdate(querySwitch);
                            }
                        }
```

```
                        else
                        {
                            System.out.println("User not found.");
                        }

                    }
                    catch (SQLException e)
                    {
                        System.out.println(e);
                    }
                }
                break;
            }
        }

        return nextStage;
    }
    private static void sendRecipe()
    {
          String resultCode = cookbookScraper.fileRecipe(recipeScraped, userName,
dbConnection);
        //System.out.println(resultCode);
        switch(resultCode)
        {
            case("RECIPE_ADDED"):
            {
                System.out.println("\nRecipe added.");
                break;
            }
            case("RECIPE_EXISTS"):
            {
                System.out.println("\nYou've already added this recipe.");
                break;
            }
            default:
            {
                System.out.println("Unknown error occurred");
                break;
            }
        }
        savedRecipe = false;
        recipeScraped = null;
```

```
    }
}
```

There is one external library used, which is asciitable. As the table's values cannot be overwritten, it is instantiated for each use which may not be perfectly efficient but is suitable for our needs. It allows printing of the data in a cleaner manner as opposed to just text, which is important when the data size that is desired to be outputted gets bigger and bigger.

There is also a small administrator access function, only available for users with the user type of admin. As of now, the function only allows the printing of the USER_ACCOUNT table which demonstrates the password hashing, as well as switching the privilege level of other users in case a user is to be promoted to an administrator role as well.

3.4 Database Development and Hosting

The database was created with simple MySQL queries.

```
CREATE TABLE USER_ACCOUNT
(
    USER_NAME       VARCHAR(16) PRIMARY KEY,
    USER_PASSWORD   CHAR(64) NOT NULL,
    USER_TYPE       VARCHAR(8) NOT NULL
);
```

The user_account table consisted of three fields. USER_NAME serves as the primary key for the table, as it is the unique distinguishing factor of the user. It is marked as a variable character with a limit of 16 characters. USER_PASSWORD was made as a character with a fixed length of 64, due to the process converting it to a hexadecimal value which will always result in 64 characters. By default, USER_TYPE will be "User" and grant access to the majority of the program.

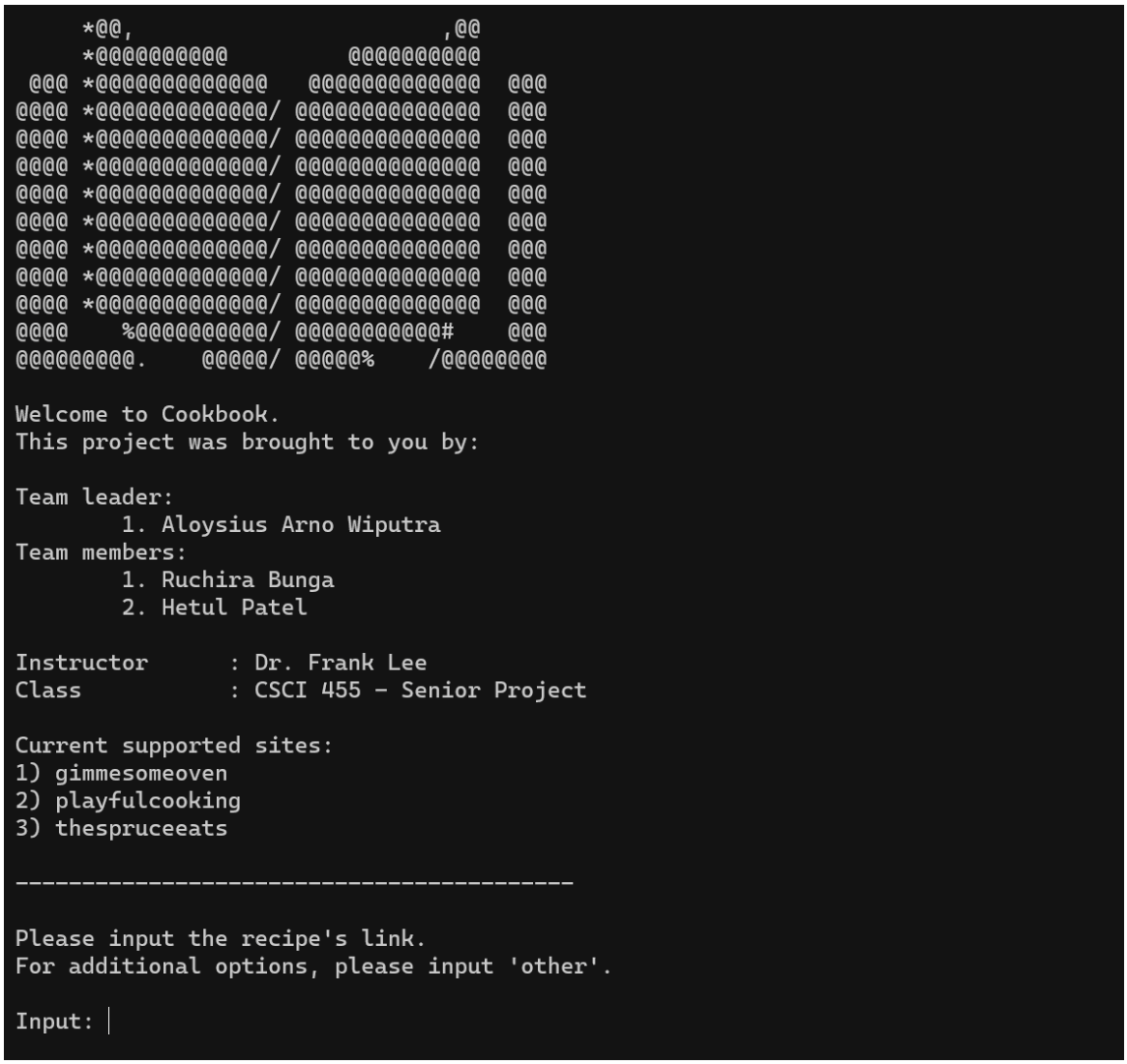
```
CREATE TABLE RECIPES
(
        RECIPE_ID       INT PRIMARY KEY AUTO_INCREMENT,
        SOURCE_USER     CHAR(16),
        FOREIGN KEY     (SOURCE_USER) REFERENCES USER_ACCOUNT(USER_NAME),
        RECIPE_TITLE    VARCHAR(64),
        RECIPE_SOURCE   VARCHAR(16),
        RECIPE_CONTENT  TEXT(1024)
);
```

The recipe table consists of 6 columns, of which one is a foreign key from the USER_ACCOUNT table. The RECIPE_ID is an auto incrementing field which allows the creation of a unique ID without further work from the driver program. The SOURCE_USER identifies which user submitted the recipe, allowing the recipes to be grabbed based on which user picked it. RECIPE_TITLE and RECIPE_SOURCE contain the webpage title and domain respectively, giving more context to recipes. RECIPE_CONTENT is a big text field with allowances up to 1024 characters. The reason the recipe is made as a big text field instead of multiple columns is for simplicity and efficiency, as each recipe will have differing amount instruction steps, and thus it was deemed better to have the program parse the text content instead of storing individual steps in the database.

Hosting was ultimately done locally through MySQL. Options such as AzureSQL and Google Cloud SQL instance were looked at, however due to the nature of the program, efficient use of the cloud platform was unadvisable due to a purely client software without API access.

CHAPTER 4
TESTING

4.1 Sample Output



The initial splash screen reflects our efforts to mimic the front end that has already been designed, acting as a landing page. It was also written in accordance to the syllabus, which required adding in certain fields to the program output. The supported sites were outputted using values drawn from a hashtable, to make sure that when a support is added, there was no need to tinker with the introduction printing method.

```
Please input the recipe's link.
For additional options, please input 'other'.

Input: https://www.gimmesomeoven.com/mulled-wine-recipe/

Title: Mulled Wine Recipe | Gimme Some Oven
Domain: gimmesomeoven

Recipe:
1) Combine ingredients. Add wine, brandy, orange slices, cloves, cinnamon, star anise, and 2 tablespoons sweetener to a large saucepan.  Stir briefly to combine.
2) Simmer. Cook the mulled wine on medium-high heat until it just barely reaches a simmer.  (Avoid letting it bubble ? you don?t want to boil off the alcohol.)  Reduce heat to low
, cover, and let the wine simmer for at least 15 minutes or up to 3 hours.
3) Strain. Using a fine mesh strainer, remove and discard the orange slices, cloves, cinnamon sticks, and star anise.  Give the mulled wine a taste, and stir in extra sweetener if
 needed.
4) Serve. Serve warm in heatproof mugs, topped with your favorite garnishes.

Would you like to save this recipe? (y/n)
Input:
```

This is a sample output of when a valid link with an acceptable domain is given.

```
Would you like to save this recipe? (y/n)
Input: y

You are not logged in.

If you would like to retrieve your existing recipes or save them, please log in. If you do not have an account, feel free to make one.

Options:
(0) Back
(1) Log in
(2) Sign up
(-) Configure database connection

Input: 1

You have opted to log in.
Input '0' to cancel.
com.mysql.cj.jdbc.ConnectionImpl@543e593

Please input your username.
Input: aloysius_w

Please input your password.

Input:
Successfully logged in, logged in as 'aloysius_w'.

Recipe added.

Welcome, aloysius_w.

Available commands:
(1) Search saved recipes
(2) See all recipes
(3) Add new recipes
(4) Log out
(5) See list of users
(6) Switch user account privilege

Input:
```

When we then choose to save it, the program detects that we are not logged in and redirects us to STAGE_TWO. After logging in with valid credentials, it is then automatically saved into the database. The password input is also blank due to the obscuring function of the console.

```
Input:
Successfully logged in, logged in as 'aloysius_w'.

Recipe added.

Welcome, aloysius_w.

Available commands:
(1) Search saved recipes
(2) See all recipes
(3) Add new recipes
(4) Log out
(5) See list of users
(6) Switch user account privilege

Input: 2
```

ID	TITLE	SOURCE
1	Rugelach Recipe	thespruceeats
2	Easy Zuppa Toscana Recipe | Gimme Some Oven	gimmesomeoven
3	Easy Chocolate Pots de Crème Recipe | Gimme Some Oven	gimmesomeoven
4	Mulled Wine Recipe | Gimme Some Oven	gimmesomeoven

```
Welcome, aloysius_w.

Available commands:
(1) Search saved recipes
(2) See all recipes
(3) Add new recipes
(4) Log out
(5) See list of users
(6) Switch user account privilege

Input:
```

The following is an example of the asciitable library used, and the STAGE_THREE stage display.

```
Input: 4

Confirm log out? (y/n)
Input: y

If you would like to retrieve your existing recipes or save them, please log in. If you do not have an account, feel free to make one.

Options:
(0) Back
(1) Log in
(2) Sign up
(-) Configure database connection

Input: 1

You have opted to log in.
Input '0' to cancel.
com.mysql.cj.jdbc.ConnectionImpl@42fcc7e6

Please input your username.
Input: its3am

Please input your password.

Input:
Successfully logged in, logged in as 'its3am'.

Welcome, its3am.

Available commands:
(1) Search saved recipes
(2) See all recipes
(3) Add new recipes
(4) Log out

Input:
```

Upon logout, we are then greeted back to STAGE_TWO. When we log in to a normal "User" account, the admin functions disappear from the options.

```
Input: 5
```

USERS	TYPE	PASSWORD
aloysius_w	ADMIN	32d14f3fc8bc869c655276b843b11d7a2f8c1f1a9d27bf85ca2d0577438e5a58
its3am	USER	05e3aaae03ac72ad5a54d54d92e87dc09c4d4bb8abcac96a752466d13a29e6cd

This is an example of the user tables showing the different types as well as the hashed password.

```
Please input the recipe's link.
For additional options, please input 'other'.

Input: https://google.com
Site isn't supported.

Please input the recipe's link.
For additional options, please input 'other'.

Input:
```

This is an example of an output when the site is not supported

```
You have opted to register.
Input '0' to cancel.
com.mysql.cj.jdbc.ConnectionImpl@73017a80

Please input your username.
Input: aloysius_w

Please input your password.

Input:
Username already taken.
Error: username already taken.
Try again.

Please input your username.
Input:
```

An example of error catching when trying to register with an already existing username.

Chapter 5
SHORTCOMINGS, WHAT WE LEARNED, AND FUTURE WORK

5.1 Shortcomings

We had notably failed in achieving several objectives. One, we had failed in doing cloud database hosting. This is due to the nature of our app moving from a web app with a client-server relation to a purely client only local app. This prevented us from properly using cloud hosting because most cloud providers do not allow public network access. We had also failed in making an app with proper GUI. The GUI had been made and was ready, but none of us had sufficient knowledge to connect it to the backend

5.2 What We Learned

5.2.1 From Aloysius Arno Wiputra's Perspective

I think this project ultimately showed the gaps in my knowledge and the growth I need to have to become a better leader. I was relatively confident in suggesting this project as it seemed to strike a balance of not being too simple but also not too complex for our team of 3 people. However, these projects are never as simple as we thought.

The big hurdle we faced was connecting the front end to the back end. We had assumed it would be relatively easy, as I had experience in doing a Java project with a Jswing GUI. figured documentation for a Java based back end with a web front end would be aplenty. There certainly were many, however many were outdated. For example:

1. We started with Java embedded as applet in HTML, only to realize support had been taken off in HTML5
2. Then we saw Google had GWT, which translated Java code into JavaScript. However, that project had been abandoned and no longer recommended.
3. Then there was hope in Spring Boot or Spring MVC, but the learning curve was steep for all the projects we had to deal with this semester. Not to mention, it required refactoring our project structure which was difficult to do.

Essentially, this teaches me to be more diligent in my research and ensure our technology is up to date. Tech is moving incredibly fast, and if we fall behind a little then we'll be abandoned.

I also encountered difficulties as a leader. Filing weekly reports was relatively straightforward, however keeping up with the members and motivating yhrm was difficult. I tried to fit the tasks to suit each members' strength, however with our size of 3 there were gaps in our knowledge that we weren't able to fill, and my own shortcoming as a leader was put to the spotlight then.

5.2.2 From Hetul Patel's Perspective

Being in a team of 3 with a very responsible leader and an active teammate, I learned a lot during this project. From applying technical skills that we learned throughout New York Tech to team working skills and communication skills with each other in the team as well as with the Professor. In the technical reference, I learned about web scraping for which I didn't have any prior hands-on experience and during this project I was able to implement database design which I learned in the Database Management course. Along with that I was successfully able to implement the Scrum methodology which I learned in Introduction to Software Engineering course.

5.3.3 From Ruchira Bunga's Perspective

Overall, this course was a nice practical experience of everything we learned over the years. It tied together numerous concepts we learned in separate classes and was an optimal way of further understanding and implementing the theoretical concepts that we were familiar with. I think that we each had to pick up skill sets over the course of the class to better understand and implement our part of the project. Of course, while we did assist each other, given that learning a specific aspect of technology to support your own part was hard enough, hence we were not really able fully assist one another. From a technical perspective, HTML and CSS were not concepts that I was proficient in, they were something I was familiar with given their role in the general field of computer science. Hence I learned how to create and implement any file in the aforementioned languages from scratch and was able to deliver my part of the project in accordance with the initial plan.

Nevertheless, the overall project was not completely successful and I believe that this reflects on every team member and our ability to function as a unit. While we each did get more proficient in, we have a long way to go! In the sense of professional development, this task helped me to collaborate with people in a timely manner and further understand the scope of each individual's role in a group project.

5.4 Future Work

The nature of the code should allow more websites to be supported with relative ease following the same framework. There also needs to be more bug fixing and testing as well as strengthening our security, as the software is still vulnerable to things such as SQL injections due to its use of Statements instead of PreparedStatements. Figuring out a proper solution to the GUI would also be lucrative.

5.5 Conclusion

Our primary aim for this project was to create a webspace that allows users to seamlessly find and save recipes. Through our project, we were able to do so with a few glitches. Primarily while each segment of our projects performs the task it was supposed to,yet they don't coherently work with one another. Nevertheless, while it is regrettable that we had not been able to achieve all of the goals we had set from the start, we still tried to make the best out of what he had and delivered a program which was functionally complete. As the functions had all been laid out, if the big hurdle of Java-HTML connection can be solved then the program should be able to reach its full original goal and capable of going further beyond. The project can potentially be very useful for many people, being their own personal digital cookbook that can be sourced from different places automatically.