CSCI-455 Senior Project

Final Report

Aloysius Arno Wiputra	1244139
Hetul Patel	1250935
Ruchira Bunga	1262634

CookBook

Your little black book of recipes

summary of the report

- PROBLEM STATEMENT
  - Motivation
  - Goals
- ANALYSIS AND DESIGN
  - Analysis
  - Design
    - Program Flow
    - Structure
- DEVELOPMENT
  - TASK DIVISION
  - FRONT END
  - BACK END
  - DATA BASE
- TESTING AND RESULTS
  - SAMPLE OUTPUT
  - DEMO
- WHAT WE LEARNED
  - SHORT COMINGS
  - WHAT WE LEARNED
  - FUTURE WORK

CHAPTER 1

PROBLEM STATEMENT

1.1 Motivation

- Many websites do not provide a clear recipe **without a layer of information** or **a backstory**

- The need for this application is to **provide the recipes with the best flowing experience** and **without the filler data**

1.2 Goals

Demonstrate skills learned throughout New York Tech:

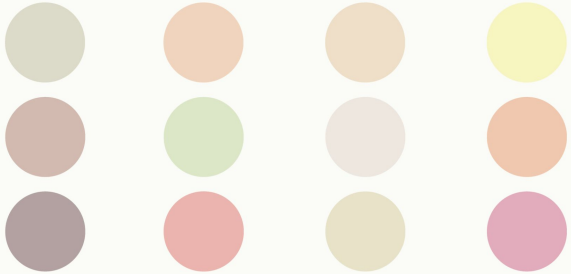
- **Proficiency** in **Java**
  (CSCI 125, CSCI 185, CSCI 260)
- **Proficiency** in **data evaluation** + **web scraping**
  (CSCI 436, CSCI 415, CSCI 426)
- **Proficiency** in **designing** and **creating algorithms**
  (CSCI 235, CSCI 335)
- **Proficiency** in **database design**
  (CSCI 360)
- **Ability** to **work** according to **SDLC**
  (CSCI 380)
- **Adaptability** in implementing **external libraries** and **new technology**

CHAPTER 2

ANALYSIS AND DESIGN

2.1 Analysis

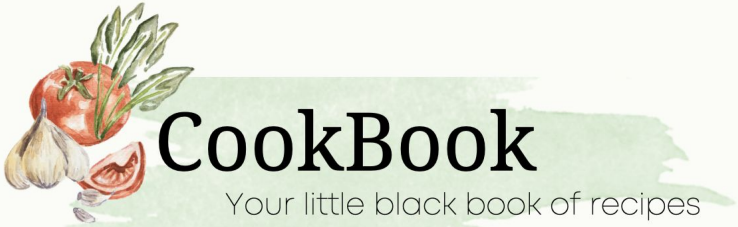
Color Palette



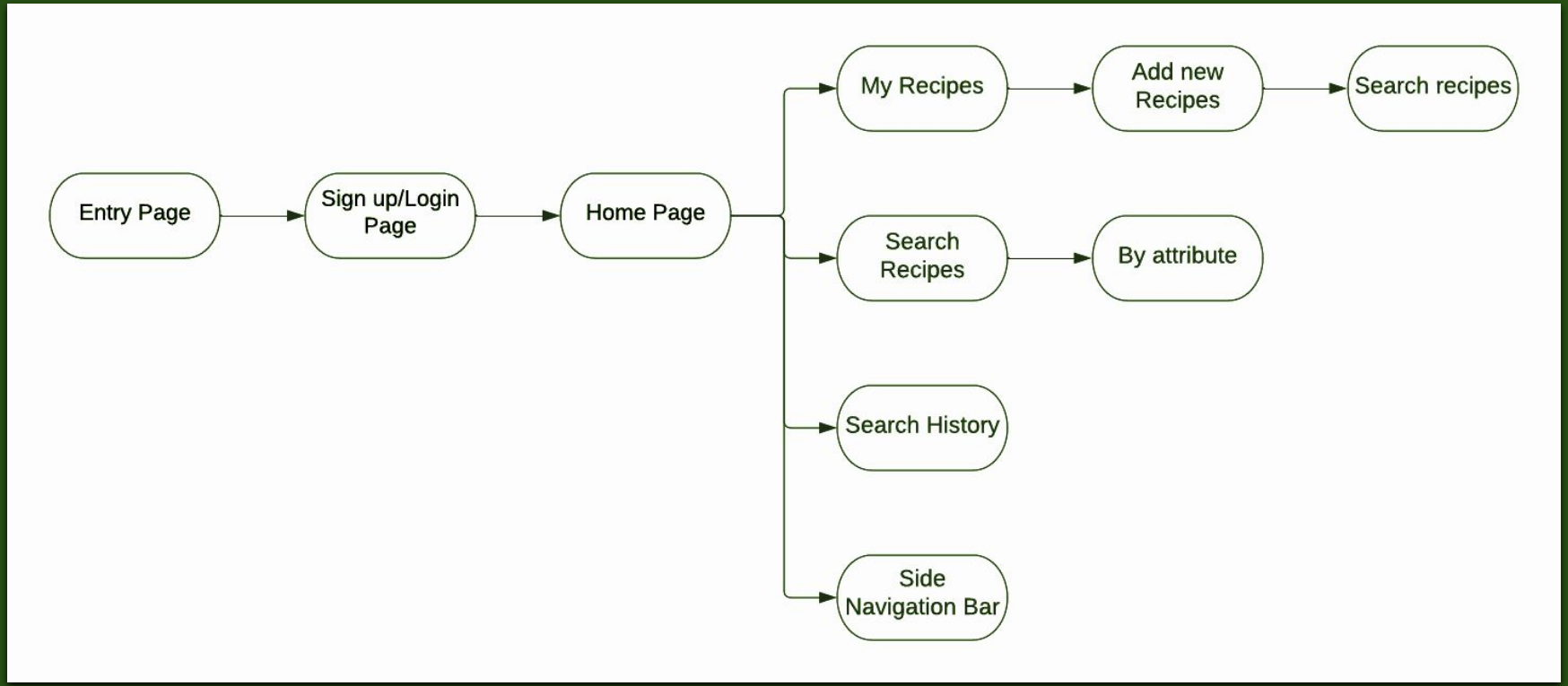
Fonts

Mont Thin

Noto Serif



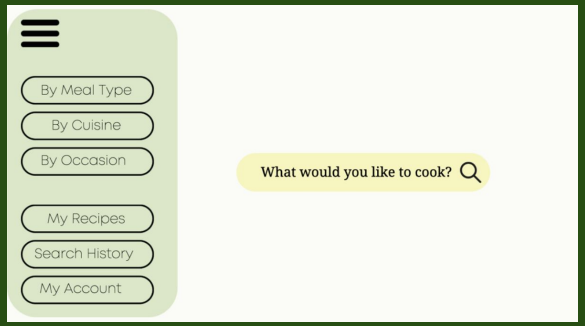
2.1 Analysis

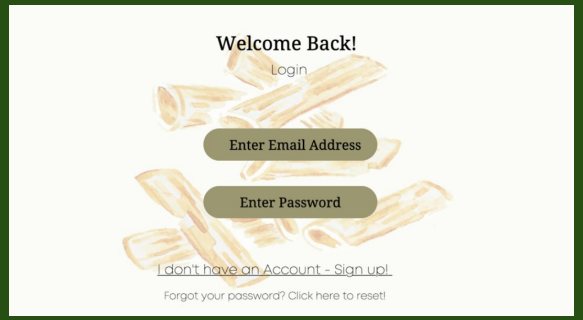

Entry Page
Sign up/Login Page
Home Page
My Recipes
Add new Recipes
Search recipes
Search Recipes
By attribute
Search History
Side Navigation Bar

2.1 Analysis


Search History
Chicken Salad
Dim Sum
Sphagetti
Fried Rice
Lasagna
Thai Curry


My Recipes
Chicken Salad
Dim Sum
Sphagetti

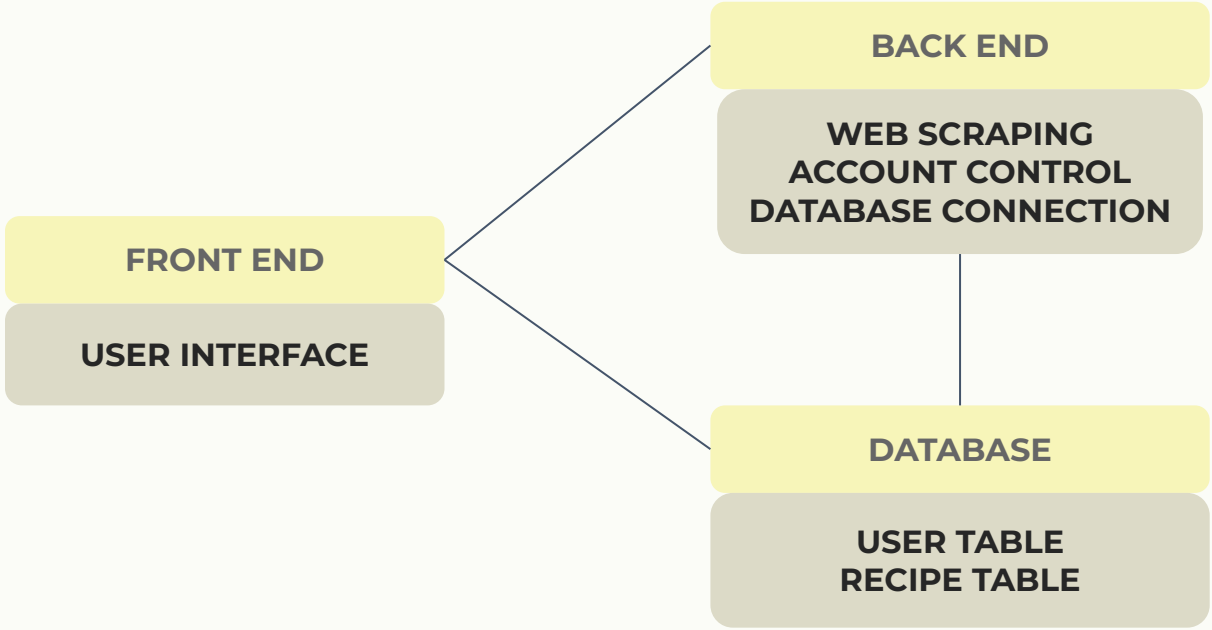

By Meal Type
By Cuisine
By Occasion
My Recipes
Search History
My Account
What would you like to cook?


Welcome Back!
Login
Enter Email Address
Enter Password
I don't have an Account - Sign up!
Forgot your password? Click here to reset!


CookBook
Your little black book of recipes
Have all your favorite recipes at your fingertips with CookBook!
What would you like to cook?
SIGN UP
LOGIN

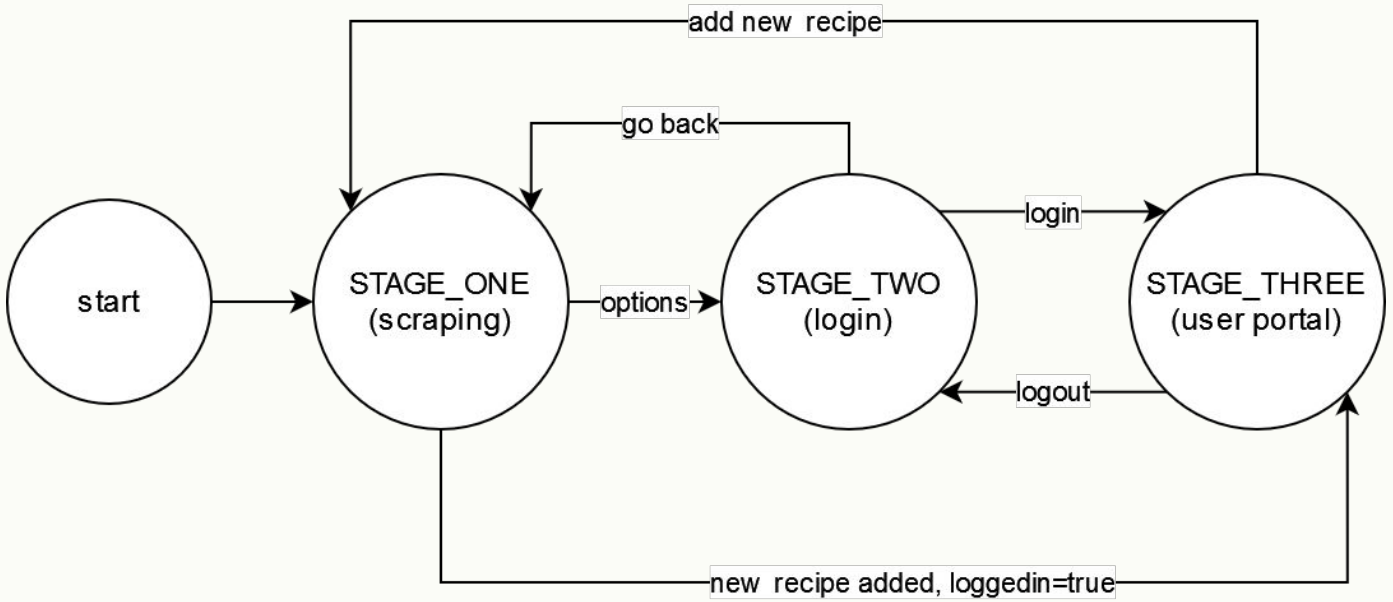

Welcome to CookBook!
Create your Account
Enter Email Address
Enter New Password
Enter Phone Number
Re-enter Password
I already Have an Account
By signing up I accept the terms of use
and the data privacy policy.

2.2 Design



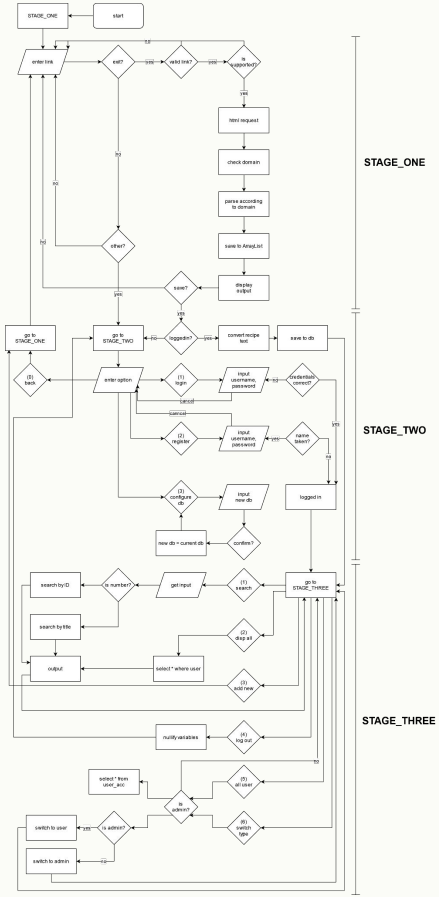
- Idea is to have the back end functions be written as **modules**
- Modules is then **connected** to the **front end**
- Each can interact with the database

2.2 Design - Program Flow



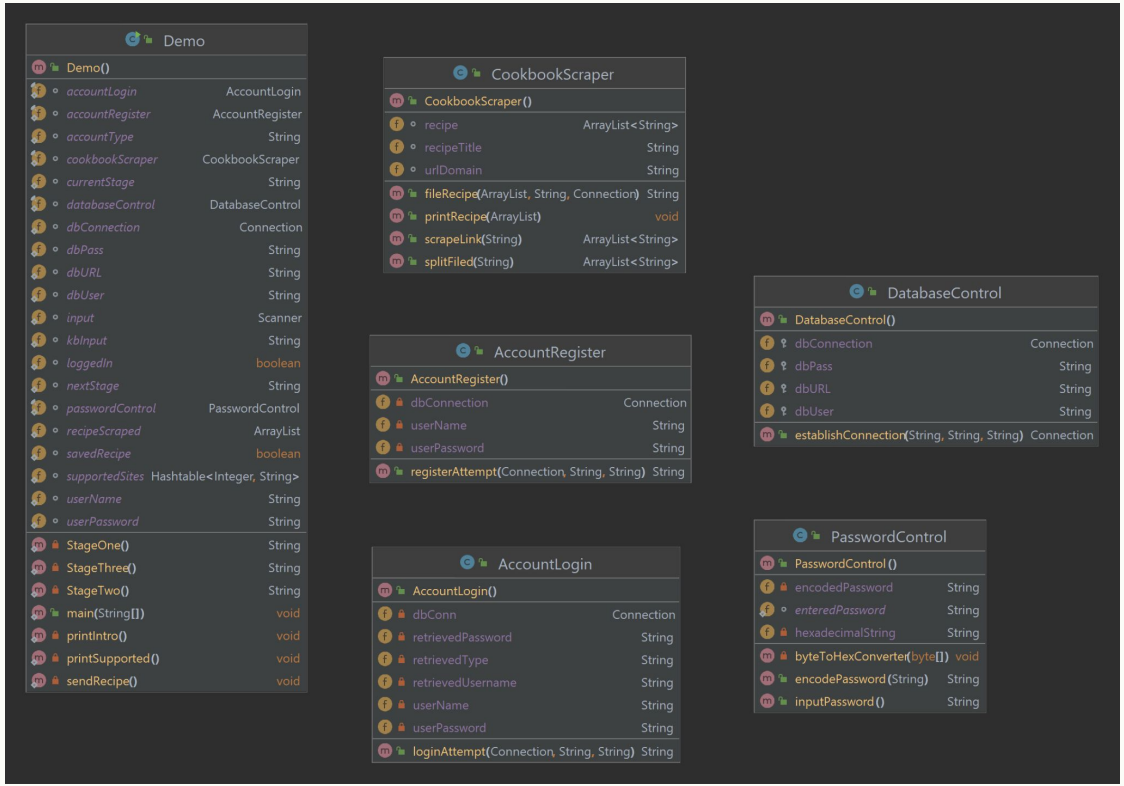
- Simplified final iteration of the **program flow**
- Each stage represents **the layers of the program**

2.2 Design - Program Flow



- Each layer has multiple different **back end functions**
  - **STAGE_ONE** : web scraping
  - **STAGE_TWO** : login/logout/db connection
  - **STAGE_THREE** : user portal
- Due to reliance on user input, **switch-case function is favored** over if-statements
- More detail in the **development section**

2.2 Design - Program Structure

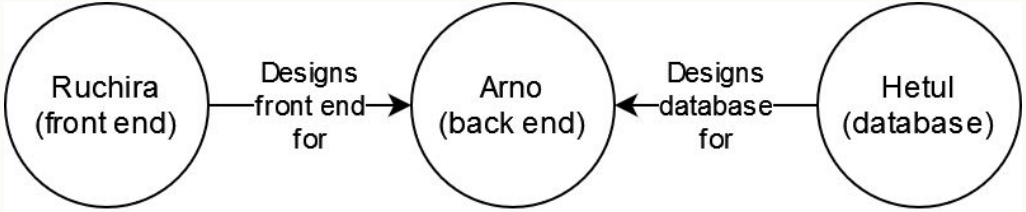


- Multiple classes were written **under different packages** with **their own methods**
- Each classes have **their own variables**
- The classes are **instantiated** in the main Demo class
- Methods **pass in variables** to perform their functions

CHAPTER 3

DEVELOPMENT

3.1 Task Division



- Tasks were divided in accordance **to each individual skills** to maximize effectiveness
- Each week **there is a goal** to be reported back for each members
- All branches **ultimately merge** in the end

3.1 Task Division

	Date		Goal						Notes
	Start	End		Amo		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		☐		☐		☐	

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

- Example of the **initial task division** and **weekly breakdown**
- Final division changes because of **scheduling** and **other issues**

3.2 Front End

Steps -

- Create framework based on designed layout : style.css file
- index.html page we confirmed that the CSS code reflected the required design
- create sections across the page and set transition timings
- creating the color scheme across the pages, the placement of images or buttons and text.
- specifics of each text - font size, color, background
- Specifics of each aspect/image of page - border radius, padding, margins
- Declared the HTML element we wanted to structurally design, followed by each incremental property and property value.

3.2 Front End

Demo section of

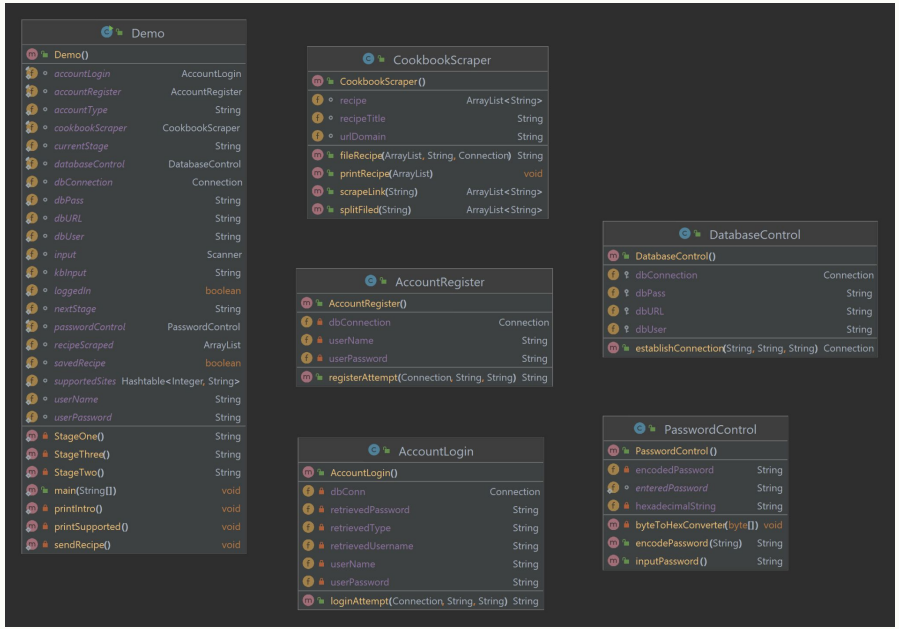
CSS Code

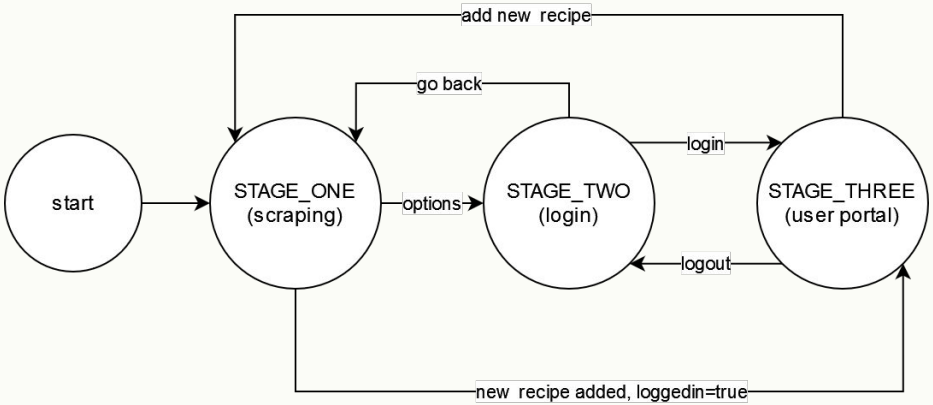
```css
/* home or index page css  */
/* .menu_btn a {
    color: black;
    position: absolute;
    font-size: 30px;
    left: 20px;
    top: 20px;
} */

main {
    text-align: center;
}

main img {
    width: 500px;
}

main p {
    font-size: 25px;
    font-family: "mont-thin";
    color: #0000009a;
    font-weight: bold;
    margin: 0;
    padding: 50px 0px;
}

main button {
    font-size: 25px;
    font-family: 'Noto Serif', serif;
    background-color: #f9f5b8;
    border: none;
    padding: 10px 15px;
    border-radius: 25px;
    cursor: pointer;
}

main button:hover {
    background-color: rgba(252, 243, 125, 0.725)
}

main button:active {
    transform: scale(0.98);
}
```

```css
.buttons_wrapper {
    padding: 50px 0px;
    text-align: center;
    display: flex;
    justify-content: center;
    gap: 150px;
}

.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;
}

.button:hover {
    background-color: #d8ebb4;
}

.button:active {
    transform: scale(0.98);
}
```

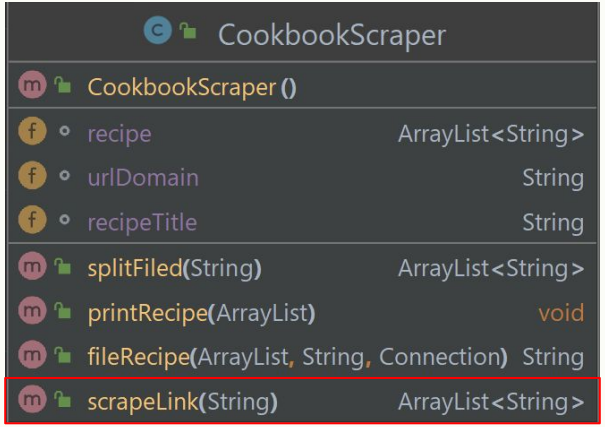
3.3 Back End

- Will be broken down to **packages** and then to **stages**





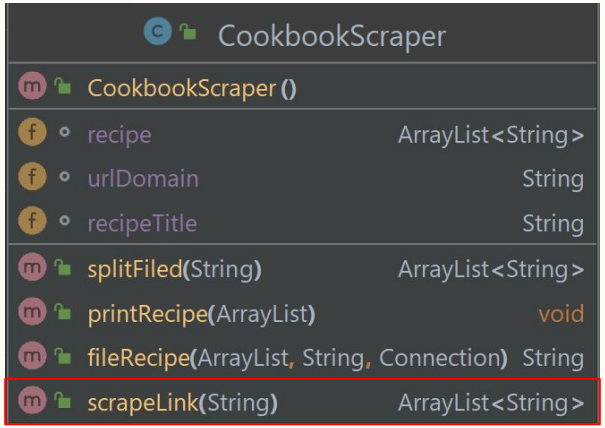
3.3 Back End



- Create HTML request
- Filter out the javascript code
- Get title

```
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);
```

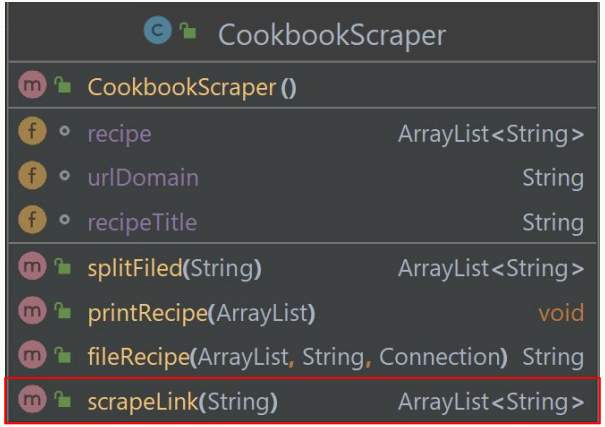
3.3 Back End




```
boolean elementNotNull = true;

this.urlDomain = StringUtils.substringBetween(url, open: "https://www.", close: ".");
if(urlDomain==null)
{
    this.urlDomain = StringUtils.substringBetween(url, open: "https://", close: ".");
}
System.out.println("Domain: " + urlDomain);
```

- Get the domain
- Catch function because some websites do not start with www

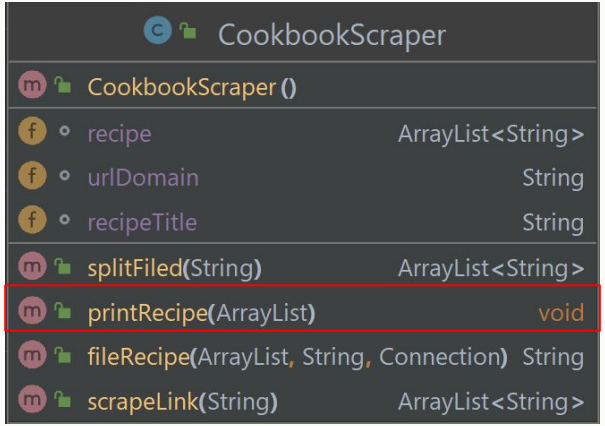
3.3 Back End



- Example of a switch case featuring one website
- Each web uses different structure, necessitating different ways of scraping

```
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/li[1]/div
        // /html/body/div[1]/div/div/div/article/div[3]/div[2]/div[2]/div/div[10]/div/ul/li[2]/div
        // /html/body/div[1]/div/div/div/article/div[3]/div[4]/div/div[12]/div/ul/li[1]/div/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( xpathExpr: "//*[@data-recipe-id]");
        String valueID = testElem.getAttribute( attributeName: "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( target: "\n", replacement: "");
            this.recipe.add(test);
        }
```

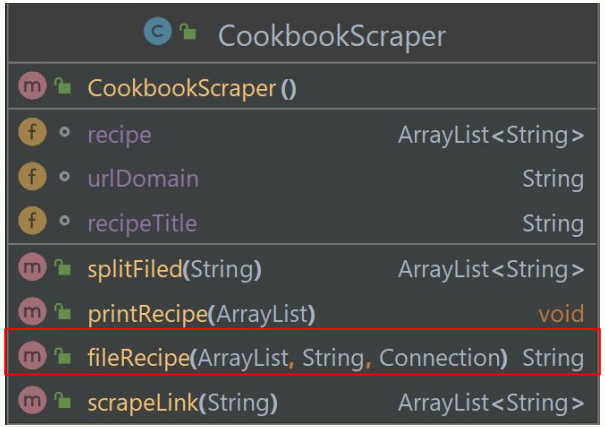
3.3 Back End



- Simple method to print out the ArrayList
- Done as a separate method because it could be called in different situations

```
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");
        }
    }
}
```

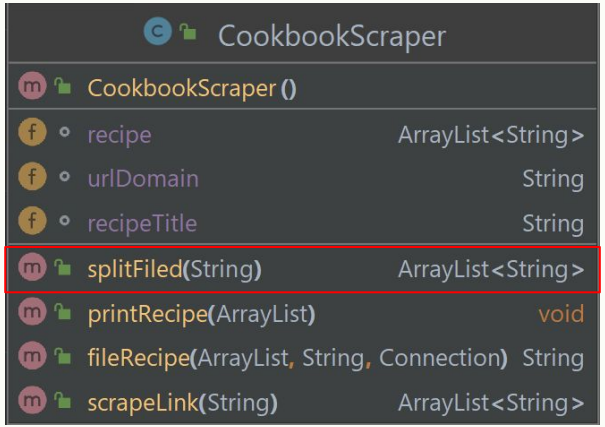
3.3 Back End



- Method to file recipe to the database
- Database cannot take ArrayList, so it needs to be converted

```
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();
```

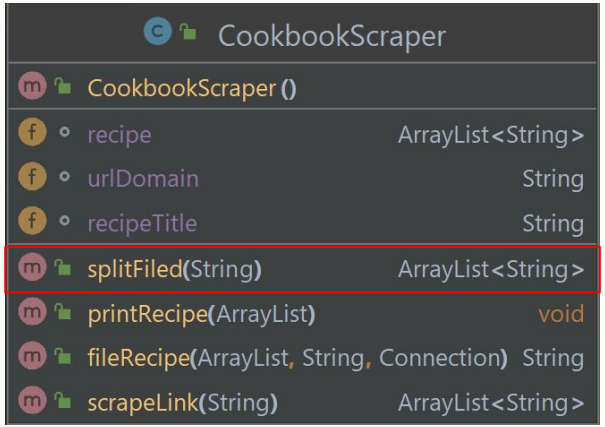
3.3 Back End




```
public ArrayList<String> splitFiled(String filedRecipe)
{
    String splitRaw[] = filedRecipe.split( regex: "@");
    List<String> splitRecipe = new ArrayList<>(Arrays.asList(splitRaw));
    return (ArrayList<String>) splitRecipe;
}
```

- Method to parse the recipe block in the database and then parse it using regex and String.split()
- Each line is then filed into an ArrayList

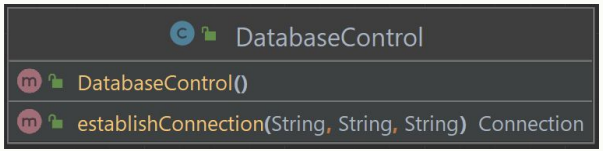
3.3 Back End




```
public ArrayList<String> splitFiled(String filedRecipe)
{
    String splitRaw[] = filedRecipe.split( regex: "@");
    List<String> splitRecipe = new ArrayList<>(Arrays.asList(splitRaw));
    return (ArrayList<String>) splitRecipe;
}
```

- Method to parse the recipe block in the database and then parse it using regex and String.split()
- Each line is then filed into an ArrayList

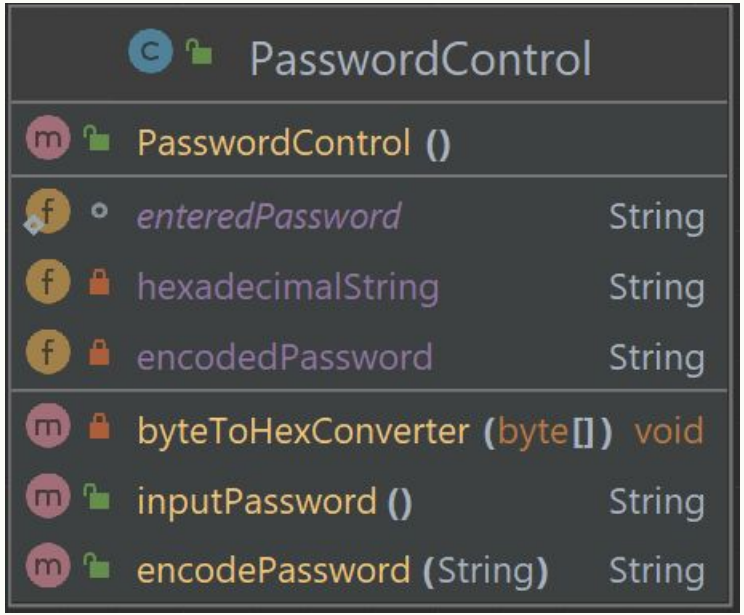
3.3 Back End



- Method to connect to a MySQL server and return a Connection data so that each function going forward only has to use that Connection variable instead of reopening connection
- Passes arguments so that connection string can easily be changed in the driver program

```
4 usages  Aloysius Arno Wiputra
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( className: "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
```

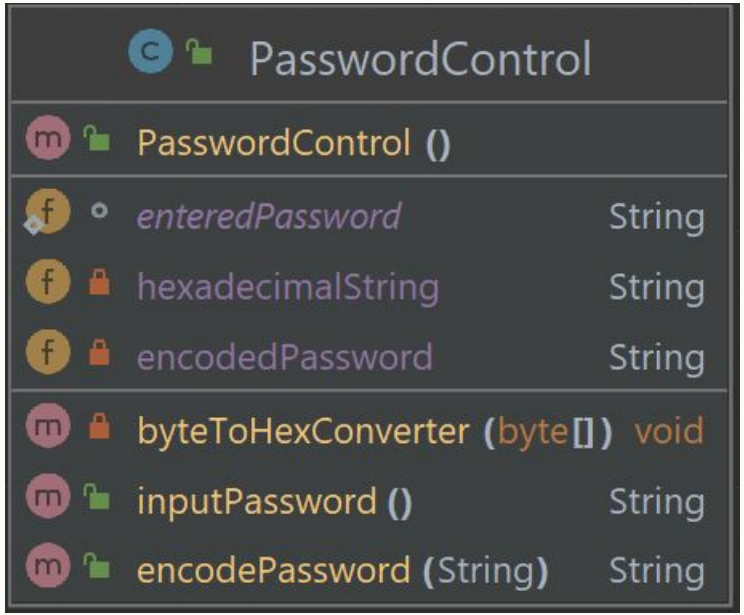
3.3 Back End



- Uses built in Java SHA-256 function to convert String to byte, and then a different function to convert it to hexadecimal

```
public String encodePassword(String userPassword)
{
    MessageDigest digest = null;
    try
    {
        digest = MessageDigest.getInstance( algorithm: "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;
}
1 usage  Aloysius Arno Wiputra
private void byteToHexConverter(byte[] hash) //convert hash into string to be able to be stored
{
    StringBuilder hexString = new StringBuilder( capacity: 2 * hash.length);
    for (int i = 0; i < hash.length; i++)
    {
        String hex = Integer.toHexString( i: 0xff & hash[i]);
        if(hex.length() == 1) {
            hexString.append('0');
        }
        hexString.append(hex);
    }
    this.hexadecimalString = hexString.toString();
}
```

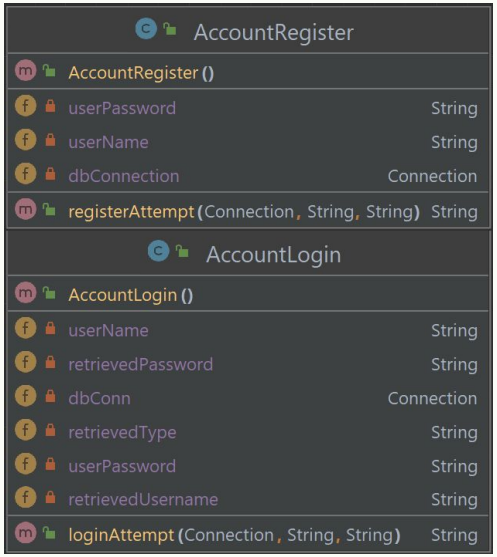
3.3 Back End



- Also has its own input function to make sure that when text is inputted, it is obscured

```
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( fmt: "Input: "));

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

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

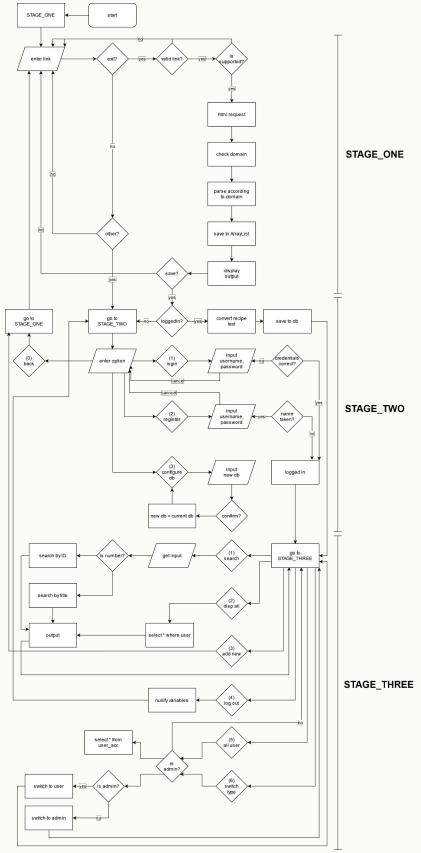
3.3 Back End



- Login and register works similarly
- Passes username and password
- Checks database for matches

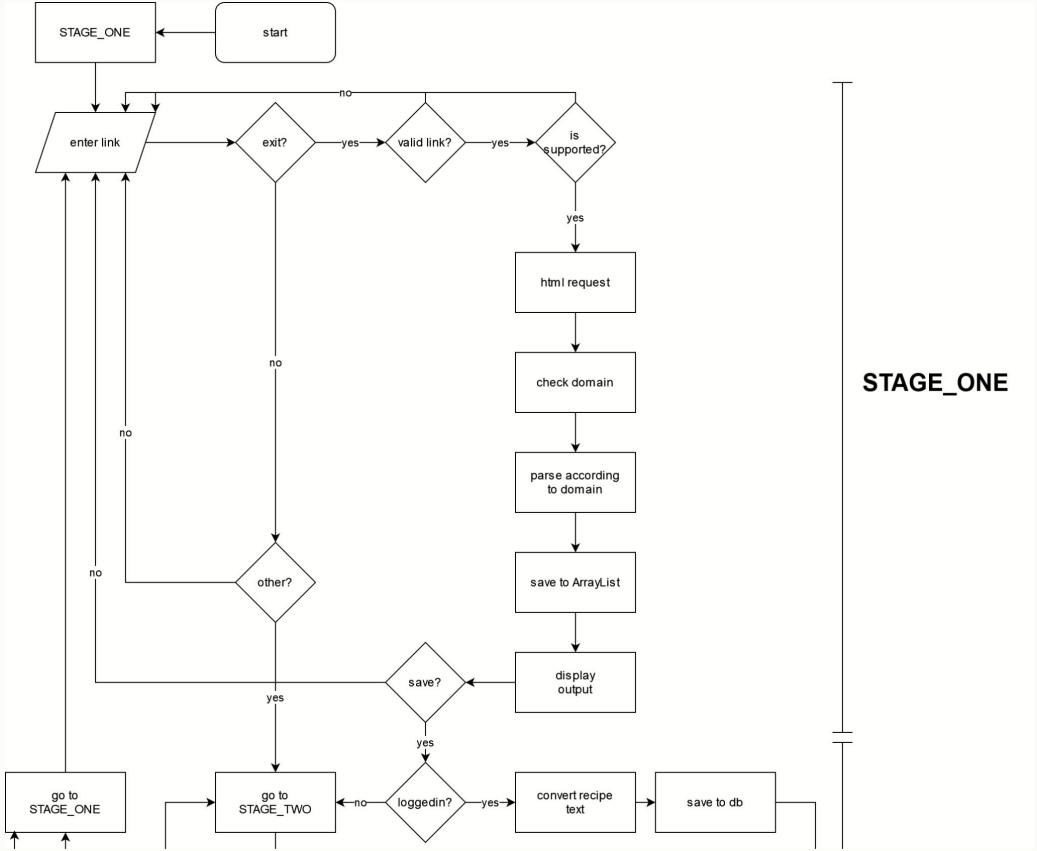
```
public String loginAttempt(Connection dbConn, String name, String password)
{
    this.userName = name;
    this.userPassword = password;
    try
    {
        Statement stm = dbConn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
        String queryPass = "SELECT * 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( columnLabel: "USER_NAME");
            this.retrievedPassword = queryResult.getString( columnLabel: "USER_PASSWORD");
            this.retrievedType = queryResult.getString( columnLabel: "USER_TYPE");
            //System.out.println(password);
            //System.out.println(retrievedPassword);
        }
    }
    catch (SQLException e)
    {
        System.out.println(e);
    }
```

3.3 Back End




```
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;
        }
    }
```

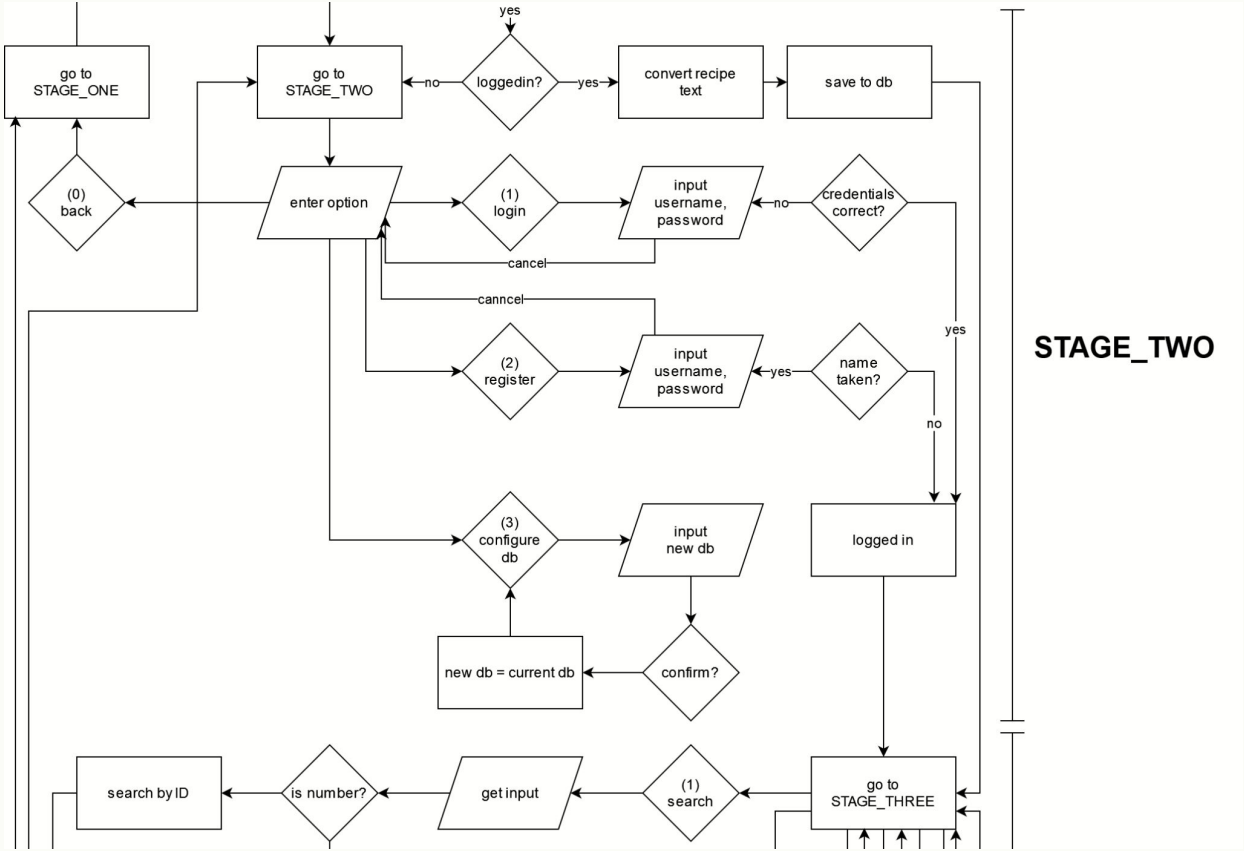
3.3 Back End




```java
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( anotherString: "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( anotherString: "y"))
            {
                savedRecipe = true;
```

```java
else if(kbInput.equalsIgnoreCase( anotherString: "other") || savedRecipe)
{
    nextStage = "STAGE_TWO";
}
else
{
    System.out.println("Input is invalid. Please try again.");
}
if(loggedIn)
{
    nextStage="STAGE_THREE";
}
return nextStage;
```

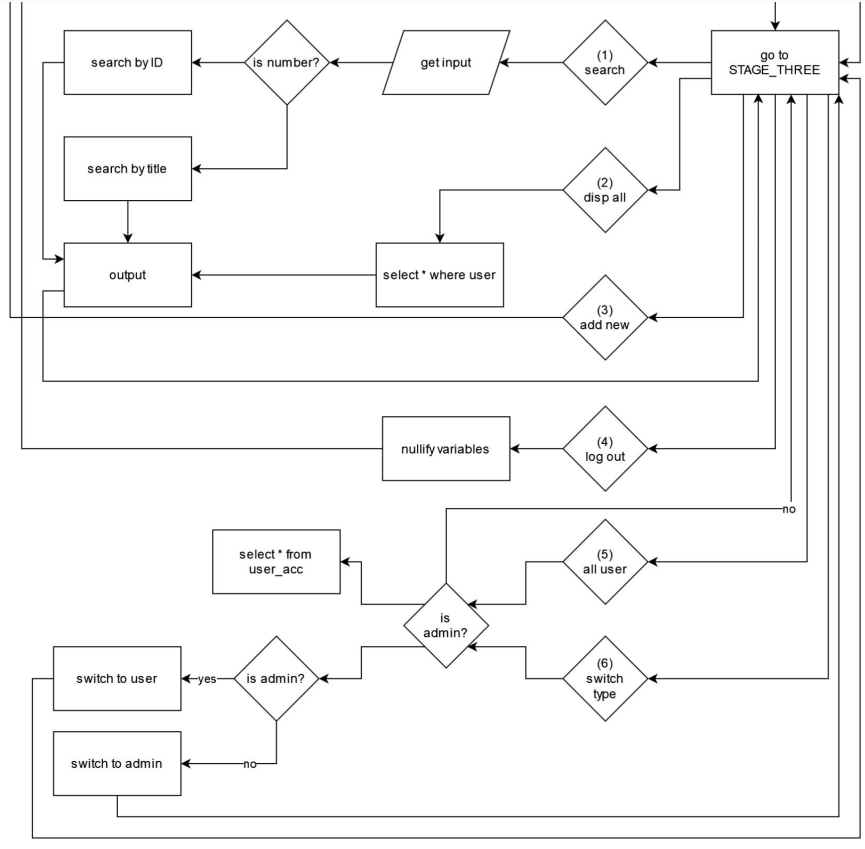
3.3 Back End




```
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"):
```

```
                //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)
```

3.3 Back End



STAGE_THREE

```
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( anotherString: "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( regex: "^[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 = '"
                ResultSet optOneIDResult = stm.executeQuery(queryOptOneID);
                if(optOneIDResult.next())
                {
```

```
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( columnIndex: 1);
        System.out.println(count);
        if(count!=0)
        {
            ResultSet optOneTitleResult = stm.executeQuery(queryOptOneTitle);
            if (count == 1)
            {
                System.out.println("\nTitle: " + optOneTitleResult.getString( columnLabel: "RECIPE_TITLE"));
                System.out.println("\nSource: " + optOneTitleResult.getString( columnLabel: "RECIPE_SOURCE"));
                optOneTitleResult.first();
                String recipeToParseRaw = optOneTitleResult.getString( columnLabel: "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( ...columns: "ID", "TITLE","SOURCE");
                optOneTable.addRule();
                while(optOneTitleResult.next())
```

3.4 Database

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

- Table for storing user account data, with user_name being the primary identifier serving as a primary key
- USER_PASSWORD is stored in a 64 character file type due to its fixed length of 64 hexadecimal charas

3.4 Database

```
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)
);
```

- Recipe table, storing automatically incrementing recipe_id to serve as the identifier, with a foreign key of source_user indicating which user submitted the recipe
- Recipe content is stored in a text block due to its large nature

CHAPTER 4

TESTING

4.1 SAMPLE OUTPUTS

```
     *@@,                       ,@@
     *@@@@@@@@@@         @@@@@@@@@@
 @@@ *@@@@@@@@@@@@@   @@@@@@@@@@@@@  @@@
@@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
@@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
@@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
@@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
@@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
@@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
@@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
@@@@ *@@@@@@@@@@@@@/ @@@@@@@@@@@@@@  @@@
@@@@    %@@@@@@@@@@/ @@@@@@@@@@@#    @@@
@@@@@@@@@.    @@@@@/ @@@@@%    /@@@@@@@@

Welcome to Cookbook.
This project was brought to you by:

Team leader:
        1. Aloysius Arno Wiputra
Team members:
        1. Ruchira Bunga
        2. Hetul Patel

Instructor      : Dr. Frank Lee
Class           : CSCI 455 - Senior Project

Current supported sites:
1) gimmesomeoven
2) playfulcooking
3) thespruceeats

------------------------------------------

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

Input:
```

- Initial output on launch, logo was an ascii converted block based on the logos designed, also puts out supported sites to alleviate confusion

- Output follows the requirements as set in the syllabus

- Initial prompt is always what recipe wants to be put in, as that is the main objective of this project

4.1 SAMPLE OUTPUTS

```
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:
```

- Sample output if a recipe is deemed valid, along with the extraction of the instruction block

4.1 SAMPLE OUTPUTS

```
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:
```

- Depiction of STAGE_TWO and STAGE_THREE transition, with automatic transfer to STAGE_TWO if recipe is asked to be saved but user is not logged in

- STAGE_THREE also has additional options because the logged in user is an admin type

4.1 SAMPLE OUTPUTS

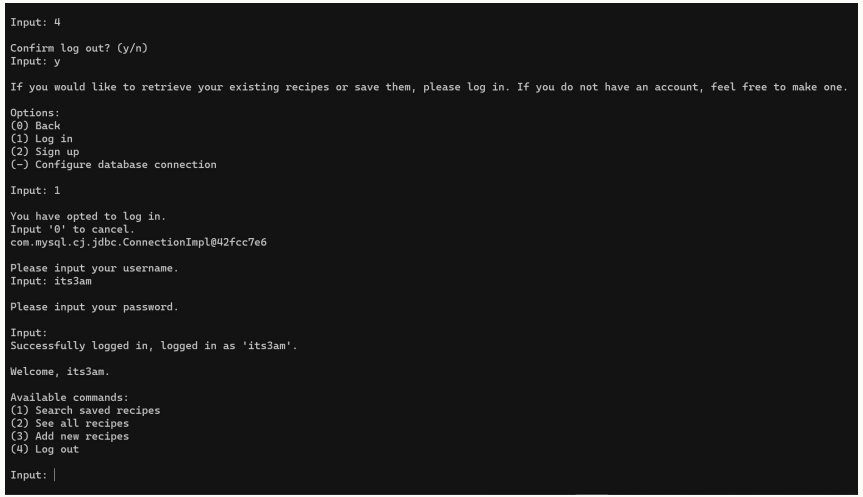
```
Input:
Successfully logged in, logged in as 'aloysius_w'.

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:
```

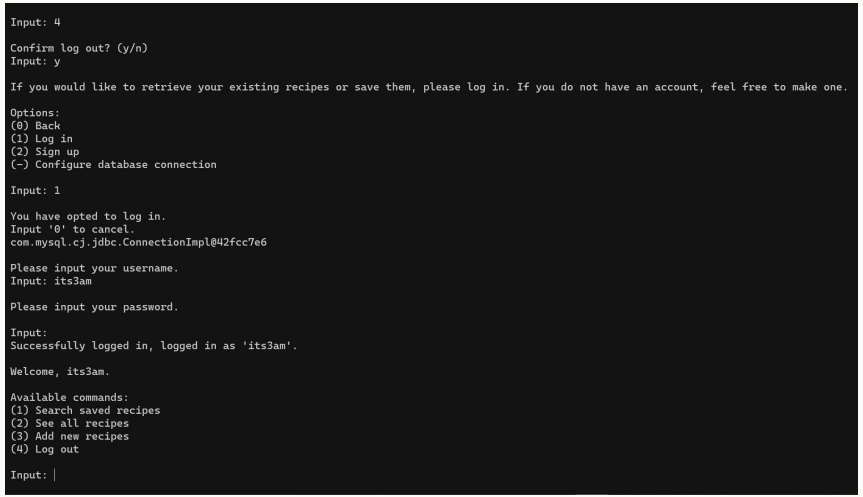
- Example output of a STAGE_THREE function and implementation of asciitable instance, showing output from database as to what recipe have been stored by the user

4.1 SAMPLE OUTPUTS



- Showcasing logging out and relogging as a different user, showing different options due to the different privilege level

4.1 SAMPLE OUTPUTS



- Showcasing logging out and relogging as a different user, showing different options due to the different privilege level

4.2 SAMPLE OUTPUTS

Input: 5

USERS	TYPE	PASSWORD
aloysius_w	ADMIN	32d14f3fc8bc869c655276b843b11d7a2f8c1f1a9d27bf85ca2d0577438e5a58
its3am	USER	05e3aaae03ac72ad5a54d54d92e87dc09c4d4bb8abcac96a752466d13a29e6cd

- Example of the user table, only accessible by admin and an example of the hashed password value

4.2 SAMPLE OUTPUTS

```
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:
```

```
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:
```

- Example of error catching, if user tries to register with a username that has already been taken and if the link is not supported

DEMO

CHAPTER 5

WHAT WE LEARNED

5.1 SHORTCOMINGS

- Lack of proper **graphical user interface**
- **Bugs**, lots of bugs
- Switching from **web app** to **purely client software** means no cloud database hosting because no API endpoint
- No ingredient support

5.2 WHAT WE LEARNED

from
ALOYSIUS ARNO WIPUTRA

(+)

- Learned a few new tools, such as Git
- Learned the hardships of team management
- Made the most out of what we've learned

(-)

- Technology is volatile and ever changing
- Team management is difficult

5.2 WHAT WE LEARNED

from
HETUL PATEL

(+)

- Implementing theoretical knowledge into practical use
- Learned working in a team

(-)

- Managing course work
- Meeting the deadlines is challenging

5.2 WHAT WE LEARNED

from RUCHIRA BUNGA

(+)

- HTML and CSS
- Understanding and Implementing the theoretical concepts

(-)

- Functioning as a Unit
- Time Management

5.2 FUTURE WORK

- More **supported websites**
- **Stronger password** mechanism
  - Use **unique string** (such as adding user and ID to the hashed password) to maximize security
  - Current approach means attackers can reference **commonly used passwords** to the hashed result
- Add **GUI**
- Use **PreparedStatement** instead of **Statement** (prevents **SQL injection attacks**)
- Add **more features** to **administrator access**
- Add **support** for **ingredients**

THANK

YOU