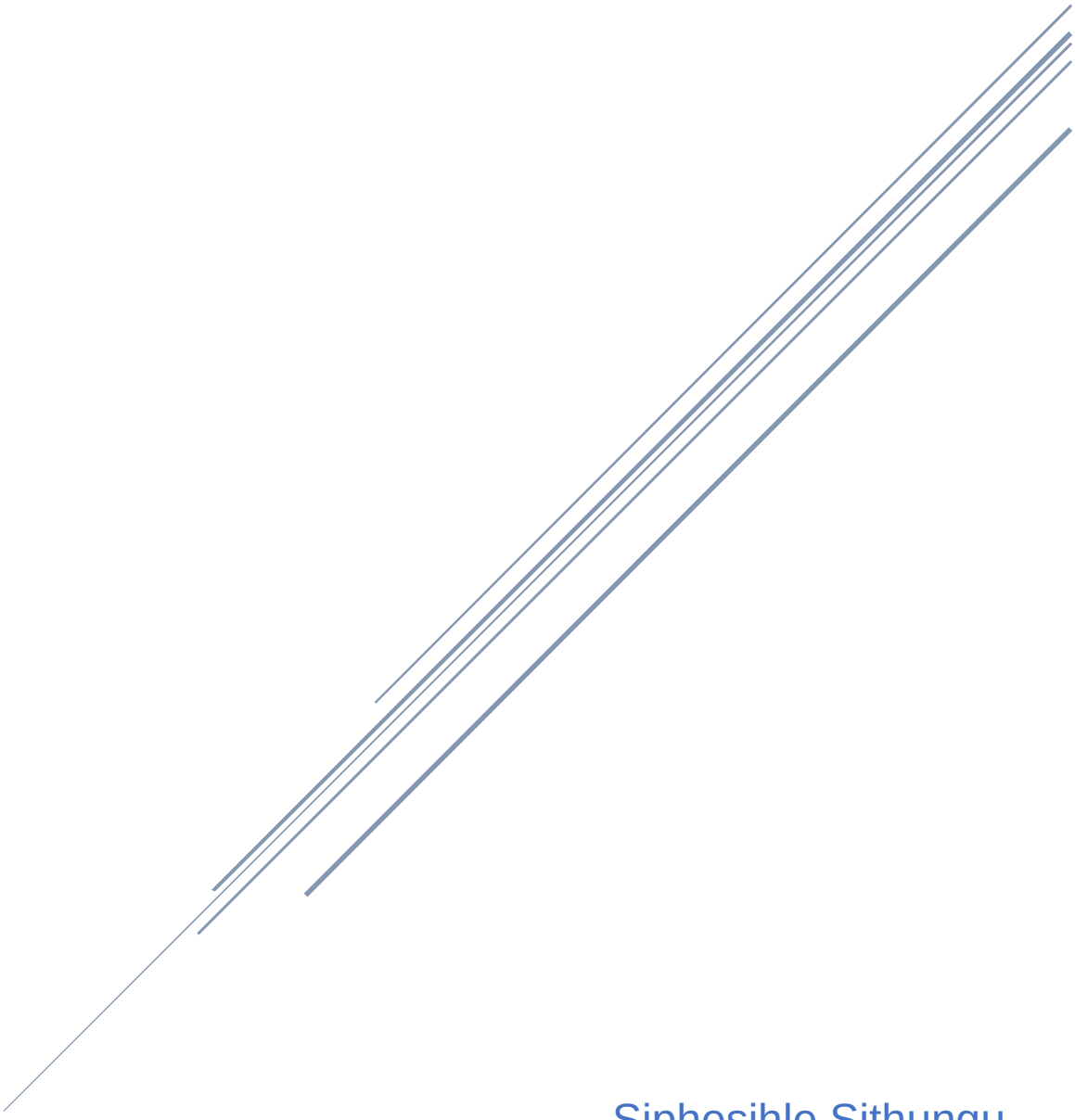
ARTIFICIAL INTELLIGENCE

Deliverable 02



Siphesihle Sithungu
201205819

Contents

1. INTRODUCTION 2
2. CLASS DIAGRAM 3
3. HIGH-LEVEL SYSTEM OVERVIEW 4
4. SOURCE CODE 4
    - 4.1. Main Class 4
    - 4.2. GraphNode Class 6
    - 4.3. SearchGraph Class 9
    - 4.4. DBClient Class 15
    - 4.5. DBConnection Class 23
    - 4.6. MaxDepthReachedException Class 24
    - 4.7. GroceryItem Class 24
    - 4.8. Merchant Class 27
    - 4.9. Session Class 30
    - 4.10. State Class 32
    - 4.11. User Class 33
    - 4.12. UIHelper Class 35
    - 4.13. Controller Class 37
    - 4.14. CreateProfileController Class 39
    - 4.15. GroceryItemController Class 40
    - 4.16. HomePageController Class 42
    - 4.17. LoginController Class 44
    - 4.18. Online View Controller Class 46
    - 4.19. PrefMerchController Class 46
    - 4.20. SettingsController Class 48
    - 4.21. ShoppingController Class 57

1. INTRODUCTION

GroceryPal serves the purpose of helping its user find groceries on the Web. In addition, GroceryPal has the capability of searching for the required grocery items on the web without the help of its user. To achieve this autonomy, GroceryPal's implementation follows that of a basic web crawler.

GroceryPal adopts this basic functionality and leverages it to create an intelligent agent that can search for grocery items on the web by using sophisticated search algorithms to find the required items fast and efficiently.

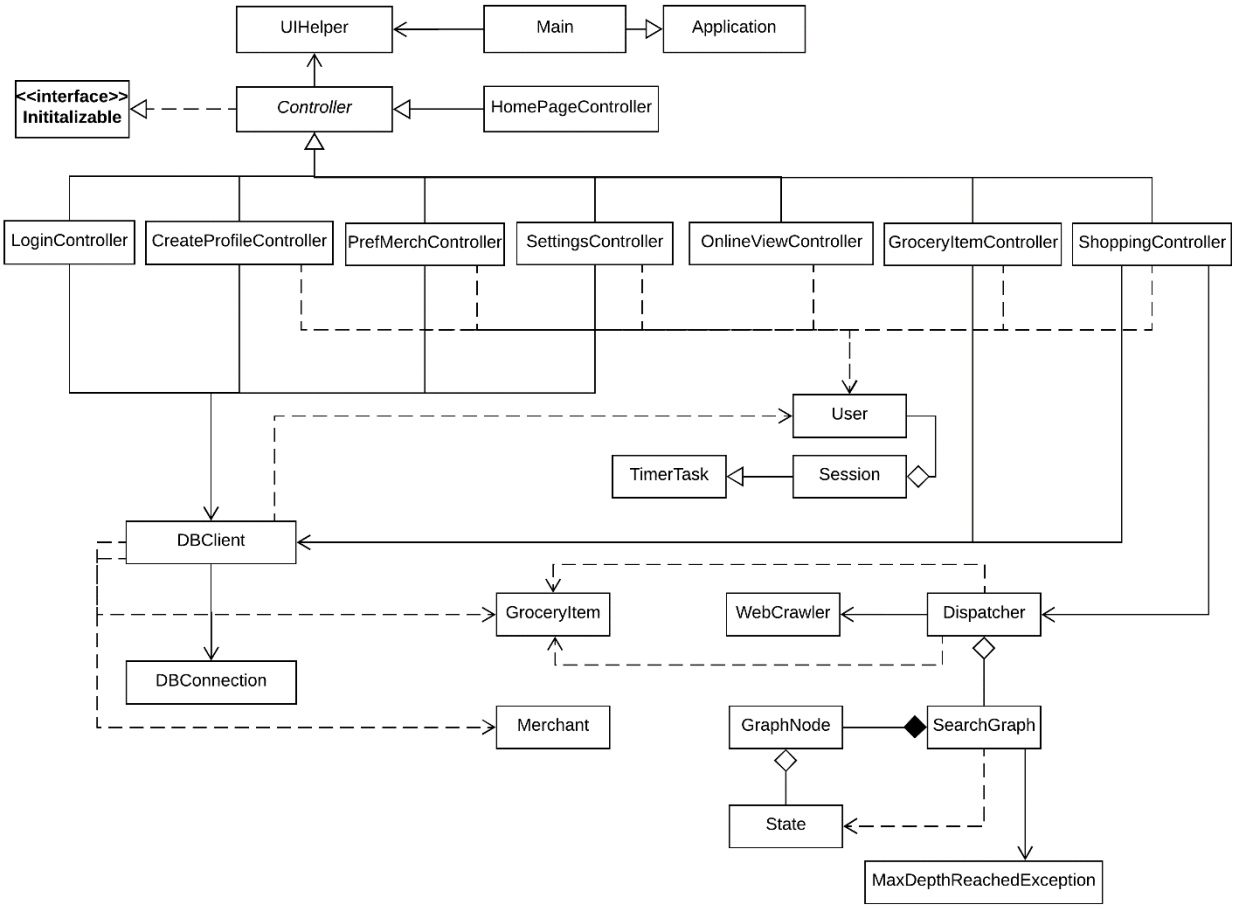
Please refer to Section 2 to view the class diagrams, Section 3 to view the high-level overview of the source code, and Section 4 to view the source code.

Please Note:

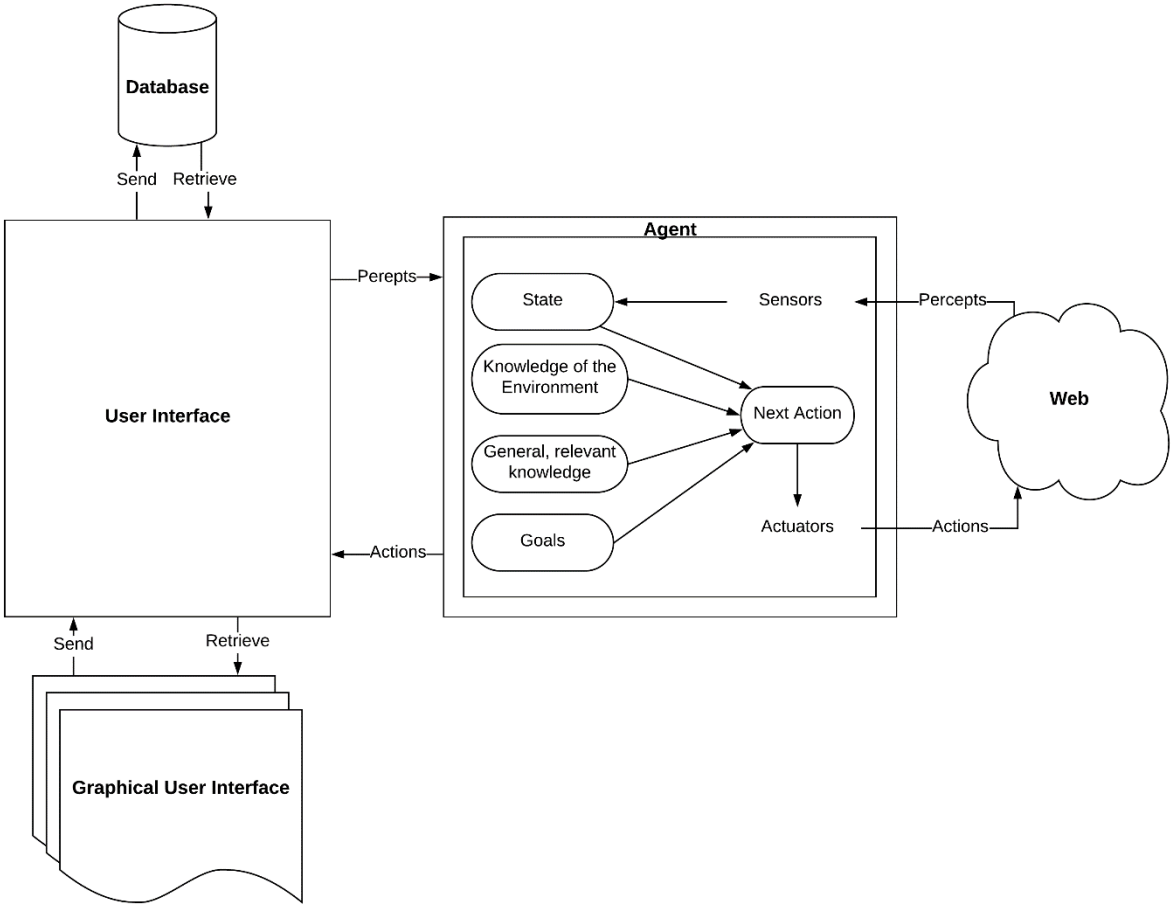
In order to run the application, the following libraries are required:

- Java FX 8 installed on eclipse IDE.
- Mysqlconnector jar file.
- Jsoup jar file.

2. CLASS DIAGRAM



3. HIGH-LEVEL SYSTEM OVERVIEW



4. SOURCE CODE

4.1. Main Class

```
import java.io.IOException;

import java.util.Optional;


import grocery_pal.ui.UIHelper;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.stage.Stage;

import javafx.scene.Parent;

import javafx.scene.Scene;
```

```
import javafx.scene.control.Alert;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Alert.AlertType;

public class Main extends Application {
	private Stage window;

	@Override
	public void start(Stage primaryStage) {
		this.window = primaryStage;
		loadFXML();
	}

	private void loadFXML() {
		try {
			Parent parent =
FXMLLoader.load(getClass().getResource("grocery_pal/ui/Design.fxml"));
			window.setScene(new Scene(parent));
			window.setTitle("GroceryPal - At Your Service");
			window.centerOnScreen();
			window.setOnCloseRequest(e -> {
				e.consume(); //Consume the event so we can handle it
manually.
				closeWindow();
			});
			window.show();
		}
		catch (IOException e) {
			e.printStackTrace();
```

```
            }
        }

        private void closeWindow() {
            Alert alert = new Alert(AlertType.CONFIRMATION);
            alert.setTitle("Close GroceryPal");
            alert.setHeaderText("You are about to exit GroceryPal.");
            alert.setContentText("Are you sure you want to exit?");
            UIHelper.findCenter(alert, window);
            Optional<ButtonType> result = alert.showAndWait();
            if(result.get() == ButtonType.OK)
                window.close();
        }

        public static void main(String[] args) {
            launch(args);
        }
}
```

4.2. GraphNode Class

```
package grocery_pal.adt;

import java.util.ArrayList;
import grocery_pal.model.State;

public class GraphNode {
        private GraphNode parent;
        private ArrayList<GraphNode> children;
        private State state;
```

```
public GraphNode(GraphNode parent, State state) {
        this.parent = parent;
        this.children = new ArrayList<>();
        this.state = state;
}

public GraphNode getParent() {
        return parent;
}

public void setParent(GraphNode parent) {
        this.parent = parent;
}

public ArrayList<GraphNode> getChildren() {
        return children;
}

public void setChildren(ArrayList<GraphNode> children) {
        this.children = children;
}

public State getState() {
        return state;
}

public void setState(State state) {
```

```
            this.state = state;
        }
}

package grocery_pal.adt;

import java.util.ArrayList;
import grocery_pal.model.State;

public class GraphNode {
        private GraphNode parent;
        private ArrayList<GraphNode> children;
        private State state;

        public GraphNode(GraphNode parent, State state) {
            this.parent = parent;
            this.children = new ArrayList<>();
            this.state = state;
        }

        public GraphNode getParent() {
            return parent;
        }

        public void setParent(GraphNode parent) {
            this.parent = parent;
        }

        public ArrayList<GraphNode> getChildren() {
```

```
		return children;
	}

	public void setChildren(ArrayList<GraphNode> children) {
		this.children = children;
	}

	public State getState() {
		return state;
	}

	public void setState(State state) {
		this.state = state;
	}
}
```

4.3. SearchGraph Class

```
package grocery_pal.adt;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Queue;
import grocery_pal.exception.MaxDepthReachedException;
import grocery_pal.model.GroceryItem;
import grocery_pal.model.State;
import grocery_pal.web.WebCrawler;

public class SearchGraph {
	private GraphNode root;
```

```
	private GroceryItem item;
	private static final int MAX_DEPTH_LEVEL = 3;

	private Queue<GraphNode> frontier;
	private ArrayList<GraphNode> explored;

	public SearchGraph(String URL, GroceryItem item) {
		State initState = new State(URL, 0, new ArrayList<>());
		GraphNode root = new GraphNode(null, initState);
		this.setRoot(root);
		this.item = item;
		frontier = new LinkedList<>();
		explored = new ArrayList<>();
		frontier.add(root);
	}

	public GroceryItem BFSSearch() {
		try {
			return BFSSearch(root);
		} catch (MaxDepthReachedException e) {
			return null;
		}
	}

	private GroceryItem BFSSearch(GraphNode node) throws
MaxDepthReachedException {
		State s = node.getState();
		if(s.getDepthLevel() == MAX_DEPTH_LEVEL) {
			throw new MaxDepthReachedException(MAX_DEPTH_LEVEL);
```

```
        }
        if(frontier.isEmpty()) {
                return null;
        }
        frontier.remove();
        explored.add(node);
        if(goalTest(s)) {
                frontier.clear();
                return IDFSSearch(node);
        }
        expandNode(node);
        if(!frontier.isEmpty())
                return BFSSearch(frontier.remove());

        return null;
}

private boolean isNewURL(String url) {
        for(GraphNode n : frontier) {
                if(n.getState().getURL().equals(url))
                        return false;
        }
        for(GraphNode n : explored) {
                if(n.getState().getURL().equals(url))
                        return false;
        }
        return true;
}
```

```
	private void expandNode(GraphNode node) throws MaxDepthReachedException
{
		Queue<String> links =  WebCrawler.getLinks();
		if(links.isEmpty()) {
			return;
		}
		while(!links.isEmpty()) {
			String url = links.remove();
			if(!isNewURL(url)) {
				continue;
			}
			makeNewNode(url, node);
		}
	}

	private void makeNewNode(String url, GraphNode node) throws
MaxDepthReachedException {
		ArrayList<String> text = WebCrawler.getFilteredText(item.getName());
		State state = new State(url, node.getState().getDepthLevel() + 1, text);

		if(state.getDepthLevel() == MAX_DEPTH_LEVEL)
			throw new MaxDepthReachedException(MAX_DEPTH_LEVEL);

		GraphNode child = new GraphNode(node, state);
		addChild(node, child);
		frontier.add(child);
	}
```

```
	private void addChild(GraphNode node, GraphNode child) {
		node.getChildren().add(child);
		//System.out.println("Child [URL: " + child.getState().getURL() + ", Depth: "
+ child.getState().getDepthLevel() + "] added to: " +
child.getParent().getState().getURL());
	}

	private GroceryItem IDFSSearch(GraphNode node) throws
MaxDepthReachedException {
		Queue<String> fLinks;
		fLinks = filterLinks(WebCrawler.getLinks());

		if(fLinks.isEmpty()) {
			return constructSolution(node);
		}
		for(String url : fLinks) {
			if(isNewURL(url)) {
				makeNewNode(url, node);
			}
		}
		if(!frontier.isEmpty())
			return IDFSSearch(frontier.remove());

		return null;
	}

	private Queue<String> filterLinks(Queue<String> links) {
		String word = item.getName();
		String[] tokens = item.getDescription().split("\\.\\s+");
```

```
        Queue<String> fLinks = new LinkedList<>();
        while(!links.isEmpty()) {
                String url = links.remove();
                if(!url.contains("#")) {
                        if(url.contains(word)) {
                                fLinks.add(url);
                        }
                        for(String s : tokens) {
                                if(url.contains(s))
                                        fLinks.add(url);
                        }
                }
        }
        return fLinks;
}

private GroceryItem constructSolution(GraphNode node) {
        GroceryItem newItem = new GroceryItem();
        newItem.setStoreURL(node.getState().getURL());
        newItem.setName(item.getName());
        newItem.setDescription(node.getState().getText().toString());
        return newItem;
}

private boolean goalTest(State state) {
        if(WebCrawler.crawl(state.getURL())) {
                String[] tokens = item.getDescription().split("\\.\\s+");
                if(WebCrawler.searchForWord(item.getName()))
```

```
                    return true;
                else{
                    for(String s : tokens) {
                        if(WebCrawler.searchForWord(s))
                            return true;
                    }
                }
            }
            return false;
        }

        public GraphNode getRoot() {
            return root;
        }

        public void setRoot(GraphNode root) {
            this.root = root;
        }
}
```

4.4. DBClient Class

```
package grocery_pal.db;

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

```
import grocery_pal.model.GroceryItem;
import grocery_pal.model.Merchant;
import grocery_pal.model.User;

public class DBClient{
	private static Connection con;
	public static int logIn(String username, String password){
		try {
			return validateUser(username, password);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return - 1;
	}

	private static int validateUser(String username, String password)throws
SQLException{
		String query = "select * from user where Username = '" + username + "'
and Password = '" + password + "' and isActive = 1";
		ResultSet rs = getResults(query);
		int id = -1;
		if(!rs.isBeforeFirst())
			return id;

		while(rs.next())
			id = (rs.getInt("Id"));

		if(!con.isClosed())
```

```
                con.close();


        return id;
    }


    public static ArrayList<GroceryItem> getGrocItems(int userId){
        ArrayList<GroceryItem> data = new ArrayList<>();
        try {
            ResultSet rs = getResults("select * from grocery_item where
User_Id = " + userId + " and isActive = 1 and Status = 'current'");
            if(!rs.isBeforeFirst())
                return null;


            //rs.beforeFirst(); //Make sure the reader always starts before the
first element of the set.
            while(rs.next())
            {
                int id = Integer.parseInt(rs.getString("Id"));
                String name = rs.getString("Name");
                String description = rs.getString("Description");
                String status = rs.getString("Status");
                String storeURL = rs.getString("Store_URL");
                GroceryItem item = new GroceryItem(id, userId, name,
description, status, storeURL);
                data.add(item);
            }
            return data;
        }
        catch (SQLException e)
```

```
            {
                e.printStackTrace();
            }
            return null;
        }

        public static ArrayList<GroceryItem> getHGrocItems(int userId){
            ArrayList<GroceryItem> data = new ArrayList<>();
            try {
                ResultSet rs = getResults("select * from grocery_item where
User_Id = " + userId + " and isActive = 1 and Status != 'current'");
                if(!rs.isBeforeFirst()) //Make sure the reader did not return an empty
list.
                    return null;

                while(rs.next())
                {
                    int id = Integer.parseInt(rs.getString("Id"));
                    String name = rs.getString("Name");
                    String description = rs.getString("Description");
                    String status = rs.getString("Status");
                    String storeURL = rs.getString("Store_URL");
                    GroceryItem item = new GroceryItem(id, userId, name,
description, status, storeURL);
                    data.add(item);
                }
                return data;
            }
            catch (SQLException e)
```

```
		{
			e.printStackTrace();
		}
		return null;
	}


	public static ArrayList<Merchant> getPrefMerchs(int userId)
	{
		ArrayList<Merchant> data = new ArrayList<>();
		try {
			ResultSet rs = getResults("select * from merchant where User_Id =
" + userId + " and isActive = 1 and Status = 'preferred'");
			if(!rs.isBeforeFirst())
				return null;


			while(rs.next())
			{
				int id = Integer.parseInt(rs.getString("Id"));
				String name = rs.getString("Name");
				String URL = rs.getString("URL");
				String rank = rs.getString("Rank");
				String status = rs.getString("Status");
				Merchant m = new Merchant(id, userId, name, URL, rank,
status);
				data.add(m);
			}
			return data;
		} catch (SQLException e) {
```

```
                e.printStackTrace();
            }
            return data;
        }

        public static void updateUser(User user) {
            String query = "update user set Username = '" + user.getUsername() + "',
Password = '" + user.getPassword() + "' where Id = " + user.getId();
            makeUpdate(query);
        }

        public static void updateGrocItem(GroceryItem item) {
            String query = "update grocery_item set Name = '" + item.getName() + "',
Description = '" + item.getDescription() + "', Status = '" + item.getStatus() + "',
Store_URL = '" + item.getStoreURL() + "' where Id = " + item.getId();
            makeUpdate(query);
        }

        public static void updatePrefMerch(Merchant merchant) {
            String query = "update merchant set Name = '" + merchant.getName() + "',
URL = '" + merchant.getURL() + "', Rank = '" + merchant.getRank() + "', Status = '" +
merchant.getStatus() + "' where Id = " + merchant.getId();
            makeUpdate(query);
        }

        public static void addUser(User user)
        {
            String query = "insert into user (Username, Password) values ('" +
user.getUsername() + "', '" + user.getPassword() + "')";
            makeUpdate(query);
```

```
	}

	public static void addGrocItem(GroceryItem item)
	{
		String query = "insert into grocery_item (Name, Description, User_Id, Status, Store_URL) values ('" + item.getName() + "', '" + item.getDescription() + "', '" + item.getUID() + "', '" + item.getStatus() + "', '" + item.getStoreURL() + "')";
		makeUpdate(query);
	}

	public static void addPrefMerch(Merchant merchant)
	{
		String query = "insert into merchant (User_Id, URL, Name, Rank, Status) values ('" + merchant.getUID() + "', '" + merchant.getURL() + "', '" + merchant.getName() + "', '" + merchant.getRank() + "', '" + merchant.getStatus() + "')";
		makeUpdate(query);
	}

	public static void removeUser(User user) {
		String query = "update user set isActive = 0 where Id = " + user.getId();
		makeUpdate(query);
	}

	public static void removeGrocItem(GroceryItem item) {
		String query = "update grocery_item set isActive = 0 where Id = " + item.getId();
		makeUpdate(query);
	}

	public static void removePrefMerchant(Merchant merchant) {
```

```
		String query = "update merchant set isActive = 0 where Id = " +
merchant.getId();
		makeUpdate(query);
	}

	public static void addHGrocItem(GroceryItem item) {
		String query = "update grocery_item set Store_URL = '" +
item.getStoreURL() + "', Status = '" + item.getStatus() + "' where Name = '" +
item.getName() + "'";
		makeUpdate(query);
	}

	private static void makeUpdate(String query)
	{
		Connection con;
		try
		{
			con =
DriverManager.getConnection(DBConnection.connectionString(),
DBConnection.username(), DBConnection.password());
			Statement stmnt = con.createStatement();
			stmnt.executeUpdate(query);
			con.close();
		}
		catch (SQLException e)
		{
			e.printStackTrace();
		}
	}
```

```
	private static ResultSet getResults(String query) throws SQLException
	{
			con =
DriverManager.getConnection(DBConnection.connectionString(),
DBConnection.username(), DBConnection.password());
			Statement stmnt = con.createStatement();
			ResultSet rs = stmnt.executeQuery(query);
			return rs;
	}
}
```

4.5. DBConnection Class

```
package grocery_pal.db;


public final class DBConnection {
	private static final String CONNECTION_STRING =
"jdbc:mysql://localhost:3306/gp_db?useSSL=false";
	private static final String USER_NAME = "root";
	private static final String PASSWORD = "jesusjesusjesus";


	public static String username()
	{
		return USER_NAME;
	}


	public static String password()
	{
		return PASSWORD;
	}
```

```
	public static String connectionString()
	{
		return CONNECTION_STRING;
	}
}
```

4.6. MaxDepthReachedException Class

```
package grocery_pal.exception;

public class MaxDepthReachedException extends Throwable {
	private static final long serialVersionUID = 1L;
	private int maxDepth;
	public MaxDepthReachedException(int maxDepth) {
		this.maxDepth = maxDepth;
	}
	public String getMessage() {
		return "Maximum depth of search graph reached: " + maxDepth;
	}
}
```

4.7. GroceryItem Class

```
package grocery_pal.model;

public class GroceryItem {
	private int id;
	private int UID;
	private String name;
	private String description;
	private String status;
```

```
	private String storeURL;

	public GroceryItem(int UID, String name, String description, String status, String storeURL) {
		this.UID = UID;
		this.name = name;
		this.description = description;
		this.storeURL = storeURL;
		this.status = status;
	}

	public GroceryItem(int id, int UID, String name, String description, String status, String storeURL) {
		this.id = id;
		this.UID = UID;
		this.name = name;
		this.description = description;
		this.storeURL = storeURL;
		this.status = status;
	}

	public GroceryItem() {}

	public int getUID() {
		return UID;
	}

	public void setUID(int uID) {
		UID = uID;
```

```
}
public String getName() {
	return name;
}

public void setName(String name) {
	this.name = name;
}

public String getDescription() {
	return description;
}

public void setDescription(String description) {
	this.description = description;
}

public int getId() {
	return id;
}

public void setId(int id) {
	this.id = id;
}

public String getStatus() {
	return status;
}
```

```
	public void setStatus(String status) {
		this.status = status;
	}

	public String getStoreURL() {
		return storeURL;
	}

	public void setStoreURL(String storeURL) {
		this.storeURL = storeURL;
	}

	@Override
	public String toString() {
		return "GroceryItem [id=" + id + ", UID=" + UID + ", name=" + name + ",
description=" + description
				+ ", status=" + status + ", storeURL=" + storeURL + "]";
	}

}
```

4.8. Merchant Class

```
package grocery_pal.model;

public class Merchant{
	private int id;
	private int UID;
	private String name;
```

```
	private String URL;
	private String status;
	private String rank;

	public Merchant(int UID, String name, String URL, String rank, String status) {
		this.UID = UID;
		this.name = name;
		this.URL = URL;
		this.rank = rank;
		this.status = status;
	}

	public Merchant(int id, int UID, String name, String URL, String rank, String
status) {
		this.id = id;
		this.UID = UID;
		this.name = name;
		this.URL = URL;
		this.rank = rank;
		this.status = status;
	}

	public int getUID() {
		return UID;
	}

	public void setUID(int uID) {
		UID = uID;
	}
```

```
public String getName() {
	return name;
}

public void setName(String name) {
	this.name = name;
}

public String getURL() {
	return URL;
}

public void setURL(String URL) {
	this.URL = URL;
}


public int getId() {
	return id;
}

public void setId(int id) {
	this.id = id;
}

@Override
public String toString() {
```

```
		return "Merchant [id=" + id + ", UID=" + UID + ", name=" + name + ",
URL=" + URL + "]";
	}

	public String getStatus() {
		return status;
	}

	public void setStatus(String status) {
		this.status = status;
	}

	public String getRank() {
		return rank;
	}

	public void setRank(String rank) {
		this.rank = rank;
	}
}
```

4.9. Session Class

```
package grocery_pal.model;

import java.util.Hashtable;
import java.util.TimerTask;

public class Session extends TimerTask {
	private static final long MAX_SESSION_TIMEOUT = 3600000;
```

```
private long startTime;
private static User user;
private static Hashtable<String, Object> variables;
public Session(User u) {
        user = u;
        variables = new Hashtable<>();
        start();
}

public void start(){
        startTime = System.currentTimeMillis();
        System.out.println("Session started.");
}

public static void destroy() {
        user = null;
        System.out.println("Session destroyed.");
}

public static boolean isActive() {
        return user != null;
}

public static User User() {
        return user;
}

public static void add(String key, Object value) {
```

```
            variables.put(key, value);
        }

        public static Object get(String key) {
            return variables.get(key);
        }

        @Override
        public void run() {
            long currentTime = System.currentTimeMillis();
            if((currentTime - startTime) >= MAX_SESSION_TIMEOUT) {
                destroy();
                System.out.println("Session timed out.");
            }
        }
}
```

4.10. State Class

```
package grocery_pal.model;

import java.util.ArrayList;

public class State {
    private String URL;
    private int depthLevel;
    private ArrayList<String> text;
    public State(String URL, int depthLevel, ArrayList<String> text) {
        this.URL = URL;
        this.depthLevel = depthLevel;
```

```
            this.setText(text);
        }

        public String getURL() {
            return URL;
        }

        public void setURL(String uRL) {
            URL = uRL;
        }
        public int getDepthLevel() {
            return depthLevel;
        }

        public void setDepthLevel(int depthLevel) {
            this.depthLevel = depthLevel;
        }

        public ArrayList<String> getText() {
            return text;
        }

        public void setText(ArrayList<String> text) {
            this.text = text;
        }
}
```

4.11. User Class

```
package grocery_pal.model;
```

```
public class User {
	private String username;
	private String password;
	private int id;
	public User(String username, String password)
	{
		this.username = username;
		this.password = password;
	}

	public User(int id, String username, String password)
	{
		this.id = id;
		this.username = username;
		this.password = password;
	}

	public String getUsername()
	{
		return username;
	}

	public void setUsername(String name)
	{
		this.username = name;
	}
```

```
	public String getPassword() {
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}

	public int getId()
	{
		return id;
	}

	public void setId(int id)
	{
		this.id = id;
	}
}
```

4.12. UIHelper Class

```
package grocery_pal.ui;

import java.util.Optional;

import javafx.scene.control.Alert;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Alert.AlertType;
import javafx.stage.Stage;
```

```
public class UIHelper {
	//Helper function for first confirming with the user before closing the window.
		public static void makeExitAlert(Stage stage) {
			Alert alert = new Alert(AlertType.CONFIRMATION);
			alert.setTitle("Close GroceryPal");
			alert.setHeaderText("You are about to exit GroceryPal.");
			alert.setContentText("Are you sure you want to exit?");
			findCenter(alert, stage);
			Optional<ButtonType> result = alert.showAndWait();
			if(result.get() == ButtonType.OK)
				stage.close();
		}

		public static void makeCustomAlert(Stage stage, String text) {
			Alert alert = new Alert(AlertType.CONFIRMATION);
			alert.setTitle("GroceryPal");
			alert.setHeaderText(text);
			findCenter(alert, stage);
			alert.showAndWait();
		}

		public static void findCenter(Alert alert, Stage window) {
			double x1 = window.getX();
			double y1 = window.getY();
			double x2 = x1 + window.getWidth();
			double y2 = y1 + window.getHeight();
			double xa = 0.5 * (x1 + x2);
			double ya = 0.5 * (y1 + y2);
```

```
                double xf = xa - 185;
                double yf = ya - 70;
                alert.setX(xf);
                alert.setY(yf);
            }
}
```

4.13. Controller Class

```
package grocery_pal.ui;

import java.io.IOException;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Modality;
import javafx.stage.Stage;

public abstract class Controller implements Initializable {
    //Method for handling the switching between scenes.
    protected void switchScene(String fxmlURL, Node node) {
        try {
            Parent parent =
FXMLLoader.load(getClass().getResource(fxmlURL));
            Stage stage = (Stage) node.getScene().getWindow();
            stage.setScene(new Scene(parent));
            stage.show();
        } catch (IOException ioe) {
```

```
                ioe.printStackTrace();
        }
}

protected Controller showSecondary(String fxmlURL) {
        FXMLLoader loader = new FXMLLoader();
        try {
                loader.setLocation(getClass().getResource(fxmlURL));
                Parent parent = loader.load();
                Stage secStage = new Stage();
                secStage.setScene(new Scene(parent));
                secStage.initModality(Modality.APPLICATION_MODAL);
                secStage.showAndWait();
                return loader.getController();
        } catch (IOException ioe) {
                ioe.printStackTrace();
        }
        return null;
}

//method for handling closing how the program will close.
protected void closeWindow(Node node, boolean confirmClose) {
        Stage stage = (Stage)node.getScene().getWindow();
        if(!confirmClose) {
                stage.close();
                return;
        }
        UIHelper.makeExitAlert(stage);
```

```
        }
}
```

4.14. CreateProfileController Class

```
package grocery_pal.ui;

import java.net.URL;
import java.util.ResourceBundle;

import grocery_pal.db.DBClient;
import grocery_pal.model.User;
import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;

public class CreateProfileController extends Controller {

        @FXML private BorderPane rootPane; //We are going to use this to keep track of the current scene.
        @FXML private TextField txtUName, txtPass, txtCPass;
        @FXML private Label lblErr; //for displaying error messages in exception cases.
        private boolean allowContinue = false;

        //function for handling the user's attempt to create a new profile.
        @FXML
        private void btnConfirmClick() {
                if(!(txtPass.getText().equals(txtCPass.getText())))
                        lblErr.setVisible(true);
```

```
            else {
                String username = txtUName.getText();
                String password = txtPass.getText();
                User user = new User(username, password);
                DBClient.addUser(user);
                allowContinue = true;
                closeWindow(rootPane, false);
            }
        }

        public boolean allowContinue() {
            return allowContinue;
        }

        @Override
        public void initialize(URL arg0, ResourceBundle arg1) {
        }

}
```

4.15. GroceryItemController Class

```
package grocery_pal.ui;

import java.net.URL;
import java.util.ResourceBundle;

import grocery_pal.db.DBClient;
import grocery_pal.model.GroceryItem;
import grocery_pal.model.Session;
```

```
import grocery_pal.model.User;
import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;

public class GroceryItemController extends Controller{
	@FXML BorderPane rootPane;
	@FXML TextField txtName, txtQuantity;
	@FXML TextArea txtDescription;
	@FXML Label lblErr;

	@FXML
	private void btnConfirmClick() {
		String name = txtName.getText();
		String description = txtDescription.getText();
		User user = Session.User();
		GroceryItem item = new GroceryItem(user.getId(), name, description,
"current", "Unknown");
		DBClient.addGrocItem(item);
		closeWindow(rootPane, false);
	}

	@Override
	public void initialize(URL arg0, ResourceBundle arg1) {

	}
```

```
}
```

4.16. HomePageController Class

```
package grocery_pal.ui;

import java.io.File;
import java.net.URL;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.scene.image.Image;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundImage;
import javafx.scene.layout.BackgroundPosition;
import javafx.scene.layout.BackgroundRepeat;
import javafx.scene.layout.BackgroundSize;
import javafx.scene.layout.BorderPane;

public class HomePageController extends Controller{
	@FXML private BorderPane rootPane;

	@FXML
	private void loginMenuItemClick() {
		attemptLogin();
	}

	private void attemptLogin() {
		LoginController c = (LoginController)showSecondary("LoginDesign.fxml");
		if(c.allowContinue())
```

```
                switchScene("Settings.fxml", rootPane);
        }

        @FXML
        private void cProfMenuItemClick() {
                CreateProfileController c =
(CreateProfileController)showSecondary("CProfile.fxml");
                if(c.allowContinue())
                        attemptLogin();
        }

        @FXML
        private void closeMenuItemClick() {
                closeWindow(rootPane, true);
        }

        @Override
        public void initialize(URL arg0, ResourceBundle arg1) {
                //Display the soccer field as a background image for the VBox defined in
the FXML document.
                File imageFile = new File("img/S-1.png"); //Extract image file.
                Image image = new Image(imageFile.toURI().toString()); //Get the image
using its URI.
                BackgroundSize size = new BackgroundSize(100, 100, true, true, true,
true); //Create background size;
                //Create a background image.
                BackgroundImage bImg = new BackgroundImage(image,
BackgroundRepeat.NO_REPEAT, BackgroundRepeat.NO_REPEAT,
BackgroundPosition.CENTER, size);
                rootPane.setBackground(new Background(bImg)); //Set the background
for the VBox as the background image.
```

```
        }
}
```

4.17. LoginController Class

```
package grocery_pal.ui;

import java.net.URL;
import java.util.ResourceBundle;
import java.util.Timer;

import grocery_pal.db.DBClient;
import grocery_pal.model.Session;
import grocery_pal.model.User;
import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;

public class LoginController extends Controller {

        //Variables for manipulating the controls in the current scene
        @FXML BorderPane rootPane;
        @FXML TextField txtUName;
        @FXML TextField txtPass;
        @FXML Label lblErr;
        private boolean allow = false;

        //For managing the button_click action
        @FXML
```

```
	private void btnLoginClick(){
		String username = txtUName.getText();
		String password = txtPass.getText();
		int UID = DBClient.logIn(username, password); //first verify the user against database entries.
		if(UID != -1)
		{
			Timer timer = new Timer();
			timer.schedule(new Session(new User(UID, username, password)), 10000);
			allow = true;
			closeWindow(rootPane, false);
		}
		else
			lblErr.setVisible(true);
	}

	public boolean allowContinue() {
		return allow;
	}

	//For initializing the scene
	@Override
	public void initialize(URL arg0, ResourceBundle arg1) {

	}
}
```

4.18. Online View Controller Class

```
package grocery_pal.ui;

import java.net.URL;
import java.util.ResourceBundle;

import grocery_pal.model.Session;
import javafx.fxml.FXML;
import javafx.scene.web.WebEngine;
import javafx.scene.web.WebView;

public class OnlineViewController extends Controller {
	@FXML private WebView webView;

	@Override
	public void initialize(URL arg0, ResourceBundle arg1) {
		if(Session.User() != null) {
			String url = (String)Session.get("URL");
			WebEngine engine = webView.getEngine();
			engine.load(url);
		}
	}
}
```

4.19. PrefMerchController Class

```
package grocery_pal.ui;

import java.net.URL;
import java.util.ResourceBundle;
```

```
import grocery_pal.db.DBClient;
import grocery_pal.model.Merchant;
import grocery_pal.model.Session;
import grocery_pal.model.User;
import javafx.fxml.FXML;
import javafx.scene.control.RadioButton;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;

public class PrefMerchController extends Controller{
	@FXML BorderPane rootPane;
	@FXML TextField txtName, txtURL;
	@FXML RadioButton rdCh1, rdCh2, rdCh3;

	@FXML
	private void btnConfirmClick() {
		String name = txtName.getText();
		int rank = 3;
		if(rdCh1.isSelected())
			rank = 1;
		else if(rdCh2.isSelected())
			rank = 2;

		String url = txtURL.getText();
		User user = Session.User();
		Merchant m = new Merchant(user.getId(), name, url,
Integer.toString(rank), "preferred");
		DBClient.addPrefMerch(m);
```

```
            closeWindow(rootPane, false);
        }

        @Override
        public void initialize(URL arg0, ResourceBundle arg1) {

        }
}
```

4.20. SettingsController Class

```
package grocery_pal.ui;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;

import grocery_pal.db.DBClient;
import grocery_pal.model.GroceryItem;
import grocery_pal.model.Merchant;
import grocery_pal.model.Session;
import grocery_pal.model.User;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
```

```
import javafx.scene.control.Label;
import javafx.scene.control.MenuItem;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellEditEvent;
import javafx.scene.control.TableView;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.ContextMenu;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.control.cell.TextFieldTableCell;
import javafx.scene.input.MouseButton;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;

public class SettingsController extends Controller {
	@FXML private BorderPane rootPane;
	@FXML private Label lblErr;

	//All the control variables needed to use the tables.
	@FXML private TableView<GroceryItem> grocTable;
	@FXML private TableView<Merchant> merchTable;
	@FXML private TableColumn<GroceryItem, String> colName, colDescription,
colStoreURL;
	@FXML private TableColumn<Merchant, String> colMerchName, colURL,
colRank;

	@FXML
	private void closeMenuItemClick() {
		closeWindow(rootPane, true);
```

```
}

@FXML
private void logoutMenuItemClick() {
	Session.destroy();
	switchScene("Design.fxml", rootPane);
}

@FXML
private void newGrocMenuItemCick() {
	if(Session.isActive()) {
		super.showSecondary("GroceryItem.fxml");
		updateGrocTable(Session.User());
	}
	else
		showSecondary("LoginDesign.fxml");
}

@FXML
private void newMerchMenuItemCick() {
	if(Session.isActive()) {
		super.showSecondary("PrefMerchant.fxml");
		updateMerchTable(Session.User());
	}
	else
		showSecondary("LoginDesign.fxml");
}
```

```
    @FXML
    private void stMenuItemClick(){
        if(Session.isActive()) {
            switchScene("Shopping.fxml");
        }
        else
            showSecondary("Login.fxml");
    }

    private void updateGrocTable(User user) {
        //Fetch grocery items from the database.
        ArrayList<GroceryItem> grocList = DBClient.getGrocItems(user.getId());
        if(grocList != null) {
            //Define the attribute names the columns need to look for in the
GroceryItem class.
            colName.setCellValueFactory(new
PropertyValueFactory<>("name"));

    colName.setCellFactory(TextFieldTableCell.<GroceryItem>forTableColumn());
//Make the columns editable.

            colDescription.setCellValueFactory(new
PropertyValueFactory<>("description"));

    colDescription.setCellFactory(TextFieldTableCell.<GroceryItem>forTableColumn
());

            colStoreURL.setCellValueFactory(new
PropertyValueFactory<>("storeURL"));

    colStoreURL.setCellFactory(TextFieldTableCell.<GroceryItem>forTableColumn()
);
```

```
                        ObservableList<GroceryItem> grocOList =
FXCollections.observableArrayList(grocList);
                        grocTable.setItems(grocOList);

                        //Add a context menu from removing a grocery item from the list.
                        ContextMenu c = new ContextMenu();
                        grocTable.addEventHandler(MouseEvent.MOUSE_CLICKED, e ->
{
                                if(e.getButton() == MouseButton.SECONDARY) {
                                        c.show(grocTable, e.getScreenX(), e.getScreenY());
                                }
                        });

                        MenuItem rMenuItem = new MenuItem("Remove item");
                        rMenuItem.setOnAction(e -> {
                                GroceryItem item =
grocTable.getSelectionModel().getSelectedItem();
                                DBClient.removeGrocItem(item);
                                grocTable.getItems().remove(item);
                                Stage s = (Stage)rootPane.getScene().getWindow();
                                UIHelper.makeCustomAlert(s, "Changes saved.");
                        });
                        c.getItems().add(rMenuItem);
                }
        }

        private void updateMerchTable(User user) {
                //Fetch preferred merchants from database.
```

```
ArrayList<Merchant> merchList = DBClient.getPrefMerchs(user.getId());
if(merchList != null) {
	//Define the attribute names the columns need to look for in the Merchant class.
	colMerchName.setCellValueFactory(new PropertyValueFactory<>("name"));

colMerchName.setCellFactory(TextFieldTableCell.<Merchant>forTableColumn());


	colURL.setCellValueFactory(new PropertyValueFactory<>("URL"));

colURL.setCellFactory(TextFieldTableCell.<Merchant>forTableColumn());


	colRank.setCellValueFactory(new PropertyValueFactory<>("rank"));

colRank.setCellFactory(TextFieldTableCell.<Merchant>forTableColumn());


	ObservableList<Merchant> merchOLIst = FXCollections.observableArrayList(merchList);
	merchTable.setItems(merchOLIst);


	//Add a context menu from removing a preferred merchant from the list.
	ContextMenu c = new ContextMenu();
	merchTable.addEventHandler(MouseEvent.MOUSE_CLICKED, e -> {
		if(e.getButton() == MouseButton.SECONDARY) {
			c.show(merchTable, e.getScreenX(), e.getScreenY());
		}
	});
```

```
                    MenuItem rMenuItem = new MenuItem("Remove item");
                    rMenuItem.setOnAction(e -> {
                        Merchant merchant =
merchTable.getSelectionModel().getSelectedItem();
                        DBClient.removePrefMerchant(merchant); //remove
preferred merchant from database.
                        merchTable.getItems().remove(merchant); //remove grocery
item from database.
                        Stage s = (Stage)rootPane.getScene().getWindow();
                        UIHelper.makeCustomAlert(s, "Changes saved.");
                    });
                    c.getItems().add(rMenuItem);
                }
            }

            //Handling column edit events by storing the newly committed value(s) to the
database.
            @FXML
            private void grocColEditCommit(CellEditEvent<GroceryItem, String> e) {
                if(Session.isActive()) {
                    GroceryItem item = e.getRowValue();
                    String newValue = e.getNewValue();

                    @SuppressWarnings("unchecked")
                    TableColumn<GroceryItem, String> c =
(TableColumn<GroceryItem, String>)e.getSource();
                    String colName = c.getText();

                    if(colName.equals("Item Name"))
```

```
                item.setName(newValue);
            else if(colName.equals("Description"))
                item.setDescription(newValue);
            else if(colName.equals("Store Web Address"))
                item.setStoreURL(newValue);
            DBClient.updateGrocItem(item);
            updateGrocTable(Session.User());
            Stage s = (Stage)rootPane.getScene().getWindow();
            UIHelper.makeCustomAlert(s, "Changes saved.");
        }
        else
            showSecondary("LoginDesign.fxml");
    }

    @FXML
    private void merchColEditCommit(CellEditEvent<Merchant, String> e) {
        if(Session.isActive()) {
            Merchant merch = e.getRowValue();
            String newValue = e.getNewValue();

            @SuppressWarnings("unchecked")
            TableColumn<Merchant, String> c = (TableColumn<Merchant,
String>)e.getSource();
            String colName = c.getText();

            if(colName.equals("Merchant Name"))
                merch.setName(newValue);
            else if(colName.equals("URL"))
                merch.setURL(newValue);
```

```
                else if(colName.equals("Rank")) {
                        if(!newValue.matches("\\d+")){
                                alertUser("Please enter a valid number (1, 2, or 3) for
the Rank.");
                                return;
                        }
                        merch.setRank(newValue);
                }
                DBClient.updatePrefMerch(merch);
                updateGrocTable(Session.User());
                Stage s = (Stage)rootPane.getScene().getWindow();
                UIHelper.makeCustomAlert(s, "Changes saved.");
        }
        else
                showSecondary("LoginDesign.fxml");
    }

    private void alertUser(String message) {
        Alert alert = new Alert(AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText("Invalid input.");
        alert.setContentText(message);
        alert.showAndWait();
    }

    protected void switchScene(String fxmlURL) {
        try {
                Parent parent =
FXMLLoader.load(getClass().getResource(fxmlURL));
```

```
                Stage stage = (Stage) rootPane.getScene().getWindow();
                stage.setScene(new Scene(parent));
                stage.setMaximized(true);
                stage.show();
            } catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }

        @Override
        public void initialize(URL arg0, ResourceBundle arg1) {
            if(!Session.isActive()) {
                showSecondary("LoginDesign.fxml");
                return;
            }
            updateGrocTable(Session.User());
            updateMerchTable(Session.User());
        }
}
```

4.21. ShoppingController Class

```
package grocery_pal.ui;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;
import grocery_pal.db.DBClient;
import grocery_pal.model.GroceryItem;
```

```
import grocery_pal.model.Merchant;
import grocery_pal.model.Session;
import grocery_pal.model.User;
import grocery_pal.web.Dispatcher;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.ContextMenu;
import javafx.scene.control.MenuItem;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.control.cell.TextFieldTableCell;
import javafx.scene.input.MouseButton;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.BorderPane;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class ShoppingController extends Controller {
        @FXML BorderPane rootPane;
        //All the control variables needed to use the tables.
        @FXML TableView<GroceryItem> tGrocList;
        @FXML TableView<GroceryItem> tHGrocList;
        @FXML TableView<GroceryItem> tFGrocList;
```

```
//Columns for tGrocList
@FXML private TableColumn<GroceryItem, String> colName, colDescription, colQuantity, colPrice, colSName, colSURL;
//Columns for tHGrocList
@FXML private TableColumn<GroceryItem, String> colHName, colHDescription, colHQuantity, colHPrice, colHStoreName, colHStoreURL, colHStatus;
//Columns for tFGrocList
@FXML private TableColumn<GroceryItem, String> colFName, colFDescription, colFQuantity, colFPrice, colFStoreName, colFStoreURL, colFStatus;

ArrayList<GroceryItem> fItems; //List for keeping storing grocery items found on the web.

private void updateTGrocList(int userId) {
    //Fetch grocery items from the database.
    ArrayList<GroceryItem> grocList = DBClient.getGrocItems(userId);

    if(grocList != null) {
        //Define the attribute names the columns need to look for in the GroceryItem class.
        colName.setCellValueFactory(new PropertyValueFactory<>("name"));

colName.setCellFactory(TextFieldTableCell.<GroceryItem>forTableColumn()); //Make the columns editable.

        colDescription.setCellValueFactory(new PropertyValueFactory<>("description"));

colDescription.setCellFactory(TextFieldTableCell.<GroceryItem>forTableColumn());
```

```
                colSURL.setCellValueFactory(new
PropertyValueFactory<>("storeURL"));

        colSURL.setCellFactory(TextFieldTableCell.<GroceryItem>forTableColumn());


                ObservableList<GroceryItem> grocOList =
FXCollections.observableArrayList(grocList);
                tGrocList.setItems(grocOList);


                //Add a context menu from removing a grocery item from the list.
                ContextMenu c = new ContextMenu();
                tGrocList.addEventHandler(MouseEvent.MOUSE_CLICKED, e -> {
                        if(e.getButton() == MouseButton.SECONDARY) {
                                c.show(tGrocList, e.getScreenX(), e.getScreenY());
                        }
                });

                MenuItem rMenuItem = new MenuItem("Remove item");
                rMenuItem.setOnAction(e -> {
                        GroceryItem item =
tGrocList.getSelectionModel().getSelectedItem();
                        DBClient.removeGrocItem(item);
                        tGrocList.getItems().remove(item);
                        Stage s = (Stage)rootPane.getScene().getWindow();
                        UIHelper.makeCustomAlert(s, "Changes saved.");
                });
                c.getItems().add(rMenuItem);
        }
    }
```

```
    private void updateTHGrocList(int userId) {
        //Fetch grocery items from the database.
        ArrayList<GroceryItem> hGrocList = DBClient.getHGrocItems(userId);

        if(hGrocList != null) {
            //Define the attribute names the columns need to look for in the GroceryItem class.
            colHName.setCellValueFactory(new PropertyValueFactory<>("name"));

    colHName.setCellFactory(TextFieldTableCell.<GroceryItem>forTableColumn()); //Make the columns editable.

            colHDescription.setCellValueFactory(new PropertyValueFactory<>("description"));

    colHDescription.setCellFactory(TextFieldTableCell.<GroceryItem>forTableColumn());

            colHStatus.setCellValueFactory(new PropertyValueFactory<>("status"));

    colHStatus.setCellFactory(TextFieldTableCell.<GroceryItem>forTableColumn());

            colHStoreURL.setCellValueFactory(new PropertyValueFactory<>("storeURL"));

    colHStoreURL.setCellFactory(TextFieldTableCell.<GroceryItem>forTableColumn());

            ObservableList<GroceryItem> fGrocOList = FXCollections.observableArrayList(hGrocList);
            tHGrocList.setItems(fGrocOList);
```

```
            }
        }

        private void updateTFGrocList(ArrayList<GroceryItem> items) {
            if(items != null) {
                //Define the attribute names the columns need to look for in the GroceryItem class.
                colFName.setCellValueFactory(new PropertyValueFactory<>("name"));

        colFName.setCellFactory(TextFieldTableCell.<GroceryItem>forTableColumn()); //Make the columns editable.

                colFStatus.setCellValueFactory(new PropertyValueFactory<>("status"));

        colFStatus.setCellFactory(TextFieldTableCell.<GroceryItem>forTableColumn());

                colFStoreURL.setCellValueFactory(new PropertyValueFactory<>("storeURL"));

        colFStoreURL.setCellFactory(TextFieldTableCell.<GroceryItem>forTableColumn());

                ObservableList<GroceryItem> oItems = FXCollections.observableArrayList(items);
                tFGrocList.setItems(oItems);

                //Add a context menu for visiting the web page that contains the selected grocery item.
                ContextMenu c = new ContextMenu();
```

```
                tFGrocList.addEventHandler(MouseEvent.MOUSE_CLICKED, e ->
{
                        if(e.getButton() == MouseButton.SECONDARY) {
                                c.show(tFGrocList, e.getScreenX(), e.getScreenY());
                        }
                });

                MenuItem visitMenuItem = new MenuItem("Visit store");
                visitMenuItem.setOnAction(e -> {
                        GroceryItem item =
tFGrocList.getSelectionModel().getSelectedItem();
                        String url = item.getStoreURL();
                        Session.add("URL", url);
                        showSecondary("OnlineView.fxml");
                });

                MenuItem acceptMenuItem = new MenuItem("Accept Item");
                acceptMenuItem.setOnAction(e -> {
                        GroceryItem item =
tFGrocList.getSelectionModel().getSelectedItem();
                        item.setStatus("Accepted");
                        DBClient.addHGrocItem(item);
                        tFGrocList.getItems().remove(item);
                        Stage s = (Stage)rootPane.getScene().getWindow();
                        UIHelper.makeCustomAlert(s, "Changes saved.");
                        updateTHGrocList(Session.User().getId());
                });

                MenuItem rejectMenuItem = new MenuItem("Reject Item");
```

```
                rejectMenuItem.setOnAction(e -> {
                        GroceryItem item =
tFGrocList.getSelectionModel().getSelectedItem();
                        item.setStatus("Rejected");
                        DBClient.addHGrocItem(item);
                        tFGrocList.getItems().remove(item);
                        Stage s = (Stage)rootPane.getScene().getWindow();
                        UIHelper.makeCustomAlert(s, "Changes saved.");
                        updateTHGrocList(Session.User().getId());
                });
                c.getItems().add(visitMenuItem);
                c.getItems().add(acceptMenuItem);
                c.getItems().add(rejectMenuItem);
        }
    }

    @FXML
    private void settingsMenuItemClick() {
        if(Session.User() != null) {
            switchScene("Settings.fxml", rootPane);
        }
    }

    protected Controller showSecondary(String fxmlURL) {
        FXMLLoader loader = new FXMLLoader();
        try {
            loader.setLocation(getClass().getResource(fxmlURL));
            Parent parent = loader.load();
            Stage secStage = new Stage();
```

```
                secStage.setScene(new Scene(parent));
                secStage.initModality(Modality.APPLICATION_MODAL);
                secStage.setMaximized(true);
                secStage.showAndWait();
        } catch (IOException ioe) {
                ioe.printStackTrace();
        }
        return null;
}

@FXML
private void closeMenuItemClick() {
        closeWindow(rootPane, true);
}

@FXML
private void logoutMenuItemClick() {
        Session.destroy();
        switchScene("Design.fxml", rootPane);
}

@FXML
private void btnStartClick() {
        if(Session.User() != null) {
                int userId = Session.User().getId();
                ArrayList<Merchant> merchList = DBClient.getPrefMerchs(userId);
                ArrayList<GroceryItem> items = DBClient.getGrocItems(userId);
                if(items == null) {
```

```
                    Stage thisStage = (Stage)rootPane.getScene().getWindow();
                    UIHelper.makeCustomAlert(thisStage, "Please add grocery
items first.");
                    return;
                }
                Dispatcher dispatcher = new Dispatcher(merchList);
                for(GroceryItem item : items) {
                    GroceryItem foundItem = dispatcher.findItem(item);
                    if(foundItem != null) {
                        fItems.add(foundItem);
                        System.out.println(foundItem.getName() + " found.");
                    }
                    else {
                        System.out.println(item.getName() + " not found.");
                    }
                }
                System.out.println("Done.");
                updateTFGrocList(fItems);
                return;
            }
            switchScene("LoginDesign.fxml", rootPane);
        }

        @Override
        public void initialize(URL arg0, ResourceBundle arg1) {
            if(Session.User() != null) {
                fItems = new ArrayList<>();
                User user = Session.User();
                updateTGrocList(user.getId());
```

```
                updateTHGrocList(user.getId());
            }

        }
}
```

4.22. Dispatcher Class

```
package grocery_pal.web;

import java.util.ArrayList;
import java.util.Comparator;
import java.util.PriorityQueue;

import grocery_pal.adt.SearchGraph;
import grocery_pal.model.GroceryItem;
import grocery_pal.model.Merchant;

public class Dispatcher {
        private SearchGraph graph;
        private ArrayList<Merchant> merchants;
        private static final String DEFAULT_URL = "http://www.pnp.co.za";

        public Dispatcher(ArrayList<Merchant> merchants) {
            this.merchants = merchants;
        }

        public GroceryItem findItem(GroceryItem item) {
            if((merchants != null) && !merchants.isEmpty()) {
                PriorityQueue<Merchant> mQueue = prepareData();
```

```
                while(!mQueue.isEmpty()) {
                        Merchant m = mQueue.remove();
                        System.out.println("Current Merchant URL: " + m.getURL());
                        graph = new SearchGraph(m.getURL(), item);
                        GroceryItem foundItem = graph.BFSSearch();
                        if(foundItem != null) {
                                foundItem.setStatus("Pending");
                                return foundItem;
                        }
                        System.out.println("Item not found.");
                }
        }
        else {
                System.out.println("Current Merchant URL: " + DEFAULT_URL);
                graph = new SearchGraph(DEFAULT_URL, item);
                GroceryItem foundItem = graph.BFSSearch();
                if(foundItem != null) {
                        foundItem.setStatus("Pending");
                        return foundItem;
                }
        }
        System.out.println("Returnin null.");
        return null;
}

private PriorityQueue<Merchant> prepareData() {
        Comparator<Merchant> c = new Comparator<Merchant>() {
                @Override
```

```
                public int compare(Merchant arg0, Merchant arg1) {
                        int r1 = Integer.parseInt(arg0.getRank());
                        int r2 = Integer.parseInt(arg1.getRank());
                        return r2 - r1;
                }
        };

        PriorityQueue<Merchant> queue = new PriorityQueue<>(c);
        for(Merchant m : merchants)
                queue.add(m);

        return queue;
    }
}
```

4.23. WebCrawler Class

```
package grocery_pal.web;

import java.io.IOException;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Queue;

import org.jsoup.Connection;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
```

```
// The following class draws inspiration from Stephen's web crawler implementation that he posted on 'Net Instructions.

// To view Stphen's implementation, please visit http://www.netinstructions.com/how-to-make-a-simple-web-crawler-in-java/.


public class WebCrawler
{
    // We'll use a fake USER_AGENT so the web server thinks the robot is a normal web browser.
    private static final String USER_AGENT =
            "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/13.0.782.112 Safari/535.1";
    private static Queue<String> links = new LinkedList<String>();
    private static Document htmlDocument;


    public static boolean crawl(String url)
    {
        try
        {
            Connection connection = Jsoup.connect(url).userAgent(USER_AGENT);
            Document htmlDoc = connection.get();
            htmlDocument = htmlDoc;
            if(connection.response().statusCode() == 200)
            {
                System.out.println("\n**Visiting** Received web page at " + url);
            }
            if(connection.response().statusCode() == 404)
            {
                    System.out.println("Resource Not Found.");
                    return false;
```

```
            }
            if(!connection.response().contentType().contains("text/html"))
            {
                System.out.println("**Failure** Retrieved something other than HTML");
                return false;
            }
            Elements linksOnPage = htmlDocument.select("a[href]");
            for(Element link : linksOnPage)
            {
                links.add(link.absUrl("href"));
            }
            return true;
        }
        catch(IOException ioe)
        {
            return false;
        }
         catch(IllegalArgumentException iae) {
            System.out.println("Illegal Argument");
            return false;
        }
    }

    public static boolean searchForWord(String searchWord)
    {
        if(htmlDocument == null)
        {
            System.out.println("ERROR! Call crawl() before performing analysis on the
document");
```

```
            return false;
        }
        String bodyText = htmlDocument.body().text();
        if(bodyText.toLowerCase().contains(searchWord.toLowerCase())) {
            return true;
        }
       return false;
    }

    public static ArrayList<String> getFilteredText(String searchWord) {
          String bodyText = htmlDocument.body().text();
                String[] tokens = bodyText.split("\\s+");

                int wordIndex = 0;
                for(int i = 0; i < tokens.length; i++) {
                        if(tokens[i].equalsIgnoreCase(searchWord))
                                wordIndex = i;
                }
                return trimCollection(wordIndex, tokens);
    }
    private static ArrayList<String> trimCollection(int wordIndex, String[] tokens) {
                ArrayList<String> result = new ArrayList<>();
                int maxBefore = 10;
                int maxAfter = 10;
                int i = wordIndex;
                int j = wordIndex;
                while((i > 0) && (j < tokens.length) && (maxBefore > 0) && (maxAfter > 0))
{
                        i--;
```

```
                    j++;
                    maxBefore--;
                    maxAfter--;
                    result.add(tokens[i]);
                    result.add(tokens[j]);
                }
                return result;
            }

            public static Queue<String> getLinks()
             {
                return links;
             }
}
```

5. FXML SOURCE CODE AUTOMATICALLY GENERATED FROM SCENE BUILDER

5.1. OnlineView.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.web.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0"
xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="grocery_pal.ui.OnlineViewController">
   <center>
```

```
      <WebView fx:id="webView" prefHeight="200.0" prefWidth="200.0"
BorderPane.alignment="CENTER" />
   </center>
</BorderPane>
```

5.2. Settings.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.canvas.*?>
<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import javafx.scene.control.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>

<BorderPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-
Infinity" minWidth="-Infinity" prefHeight="500.0" prefWidth="700.0"
xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="grocery_pal.ui.SettingsController">
   <center>
      <SplitPane dividerPositions="0.15013404825737264" orientation="VERTICAL"
BorderPane.alignment="CENTER">
        <items>
          <VBox alignment="TOP_CENTER" prefHeight="40.0" prefWidth="598.0">
            <children>
              <Label contentDisplay="CENTER" text="Settings">
                <font>
                  <Font size="36.0" />
                </font>
```

```
            </Label>
          </children>
        </VBox>
        <TabPane tabClosingPolicy="UNAVAILABLE">
          <tabs>
            <Tab text="Grocery List">
              <content>
                <TableView fx:id="grocTable" editable="true"
tableMenuButtonVisible="true">
                  <columnResizePolicy>
                    <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
                  </columnResizePolicy>
                  <columns>
                    <TableColumn fx:id="colName"
onEditCommit="#grocColEditCommit" prefWidth="75.0" text="Item Name" />
                    <TableColumn fx:id="colDescription"
onEditCommit="#grocColEditCommit" prefWidth="75.0" text="Description" />
                    <TableColumn fx:id="colStoreURL"
onEditCommit="#grocColEditCommit" prefWidth="96.0" text="Store Web Address" />
                  </columns>
                </TableView>
              </content>
            </Tab>
            <Tab text="Preferred Merchants">
              <content>
                <TableView fx:id="merchTable" editable="true"
tableMenuButtonVisible="true">
                  <columnResizePolicy>
                    <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
                  </columnResizePolicy>
```

```
                        <columns>
                            <TableColumn fx:id="colMerchName"
onEditCommit="#merchColEditCommit" prefWidth="75.0" text="Merchant Name" />
                            <TableColumn fx:id="colURL"
onEditCommit="#merchColEditCommit" prefWidth="75.0" text="URL" />
                            <TableColumn fx:id="colRank"
onEditCommit="#merchColEditCommit" prefWidth="75.0" text="Rank/Priority" />
                        </columns>
                    </TableView>
                </content>
            </Tab>
        </tabs>
    </TabPane>
  </items>
 </SplitPane>
</center>
<top>
 <MenuBar BorderPane.alignment="CENTER">
  <menus>
   <Menu mnemonicParsing="false" text="Options">
    <items>
     <MenuItem fx:id="stMenuItem" mnemonicParsing="false"
onAction="#stMenuItemClick" text="Start Shopping" />
     <SeparatorMenuItem mnemonicParsing="false" />
     <MenuItem fx:id="logoutMenuItem" mnemonicParsing="false"
onAction="#logoutMenuItemClick" text="Logout" />
     <MenuItem mnemonicParsing="false" onAction="#closeMenuItemClick"
text="Close" />
    </items>
   </Menu>
```

```
            <Menu mnemonicParsing="false" text="Help">
              <items>
                <MenuItem mnemonicParsing="false" text="About" />
              </items>
            </Menu>
            <Menu mnemonicParsing="false" text="New">
              <items>
                <MenuItem mnemonicParsing="false" onAction="#newGrocMenuItemCick"
text="Grocery Item" />
                <MenuItem mnemonicParsing="false" onAction="#newMerchMenuItemCick"
text="Preferred Merchant" />
              </items>
            </Menu>
          </menus>
        </MenuBar>
      </top>
</BorderPane>
```

5.3. Shopping.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.geometry.*?>
<?import javafx.scene.shape.*?>
<?import javafx.scene.text.*?>
<?import javafx.scene.control.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>
```

```
<BorderPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0" prefWidth="1320.0" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="grocery_pal.ui.ShoppingController">
  <top>
    <MenuBar BorderPane.alignment="CENTER">
      <menus>
        <Menu mnemonicParsing="false" text="Options">
          <items>
            <MenuItem mnemonicParsing="false" onAction="#settingsMenuItemClick" text="Settings" />
            <SeparatorMenuItem mnemonicParsing="false" />
            <MenuItem mnemonicParsing="false" onAction="#closeMenuItemClick" text="Close" />
            <MenuItem mnemonicParsing="false" onAction="#logoutMenuItemClick" text="Logout" />
          </items>
        </Menu>
        <Menu mnemonicParsing="false" text="Help">
          <items>
            <MenuItem mnemonicParsing="false" text="About" />
          </items>
        </Menu>
      </menus>
    </MenuBar>
  </top>
  <center>
    <SplitPane dividerPositions="0.11169284467713787, 0.6457242582897034" orientation="VERTICAL" BorderPane.alignment="CENTER">
      <items>
```

```
        <HBox alignment="TOP_RIGHT" minHeight="-Infinity" prefHeight="25.0"
prefWidth="1318.0" spacing="10.0">
          <children>
            <Label alignment="TOP_LEFT" contentDisplay="CENTER" text="Shopping
History" textAlignment="CENTER">
              <font>
                <Font size="18.0" />
              </font>
              <HBox.margin>
                <Insets right="470.0" />
              </HBox.margin>
            </Label>
            <Button fx:id="btnStart" mnemonicParsing="false" onAction="#btnStartClick"
text="Start Shopping">
              <HBox.margin>
                <Insets right="50.0" />
              </HBox.margin></Button>
          </children>
        </HBox>
        <GridPane alignment="TOP_CENTER" minHeight="-Infinity">
          <columnConstraints>
            <ColumnConstraints halignment="CENTER" hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />
            <ColumnConstraints halignment="CENTER" hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />
          </columnConstraints>
          <rowConstraints>
            <RowConstraints maxHeight="136.0" minHeight="10.0" prefHeight="35.0" />
            <RowConstraints maxHeight="-Infinity" minHeight="10.0" prefHeight="300.0"
vgrow="SOMETIMES" />
```

```
            </rowConstraints>
            <children>
               <Label text="Current Grocery List">
                  <font>
                     <Font size="18.0" />
                  </font>
                  <GridPane.margin>
                     <Insets bottom="5.0" top="5.0" />
                  </GridPane.margin>
               </Label>
               <TableView fx:id="tGrocList" tableMenuButtonVisible="true"
GridPane.rowIndex="1">
                  <columns>
                     <TableColumn fx:id="colName" prefWidth="75.0" text="Item Name" />
                     <TableColumn fx:id="colDescription" prefWidth="75.0" text="Description"
/>
                     <TableColumn fx:id="colSURL" prefWidth="75.0" text="Store Web
Address" />
                  </columns>
                  <columnResizePolicy>
                     <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
                  </columnResizePolicy>
               </TableView>
               <Label text="History" GridPane.columnIndex="1">
                  <font>
                     <Font size="18.0" />
                  </font>
               </Label>
               <TableView fx:id="tHGrocList" prefWidth="370.0"
tableMenuButtonVisible="true" GridPane.columnIndex="1" GridPane.rowIndex="1">
```

```
                <columns>
                  <TableColumn fx:id="colHName" prefWidth="75.0" text="Item Name" />
                  <TableColumn fx:id="colHDescription" prefWidth="75.0"
text="Description" />
                  <TableColumn fx:id="colHStoreURL" prefWidth="75.0" text="Store Web
Address" />
                  <TableColumn fx:id="colHStatus" prefWidth="75.0" text="Status" />
                </columns>
                <columnResizePolicy>
                  <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
                </columnResizePolicy>
              </TableView>
            </children>
          </GridPane>
          <VBox alignment="TOP_CENTER" maxHeight="-Infinity" minHeight="-Infinity"
prefHeight="300.0">
            <children>
              <Label text="Items Found">
                <font>
                  <Font size="18.0" />
                </font>
                <VBox.margin>
                  <Insets top="10.0" />
                </VBox.margin>
              </Label>
              <TableView fx:id="tFGrocList" editable="true"
tableMenuButtonVisible="true">
                <columns>
                  <TableColumn fx:id="colFName" prefWidth="75.0" text="Item Name" />
```

```
                    <TableColumn fx:id="colFStoreURL" prefWidth="85.0" text="Store Web
Address" />
                    <TableColumn fx:id="colFStatus" prefWidth="85.0" text="Status" />
                </columns>
                <columnResizePolicy>
                    <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
                </columnResizePolicy>
            </TableView>
        </children>
    </VBox>
  </items>
 </SplitPane>
</center>
</BorderPane>
```

5.4. CProfile.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.shape.*?>
<?import javafx.scene.paint.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.text.*?>
<?import javafx.scene.web.*?>
<?import javafx.scene.effect.*?>
<?import javafx.scene.*?>
<?import javafx.scene.media.*?>
<?import java.lang.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
```

```
<?import javafx.scene.layout.AnchorPane?>


<BorderPane fx:id="rootPane" prefHeight="260.0" prefWidth="500.0"
xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="grocery_pal.ui.CreateProfileController">

   <center>

      <VBox alignment="CENTER" style="-fx-background-color: white;"
BorderPane.alignment="CENTER">

         <children>

            <Label text="Create Profile" VBox.vgrow="ALWAYS">

               <font>

                  <Font size="36.0" />

               </font>

               <padding>

                  <Insets bottom="10.0" />

               </padding>

            </Label>

            <Line endX="300.0" opacity="0.4" startX="-100.0" stroke="#006633"
strokeLineCap="ROUND" strokeLineJoin="ROUND" strokeWidth="2.0" />

            <Label fx:id="lblErr" text="Passwords do not match." textFill="#d61515"
visible="false">

               <padding>

                  <Insets bottom="20.0" />

               </padding>

            </Label>

            <GridPane>

              <columnConstraints>

                <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0"
prefWidth="100.0" />

                <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0"
prefWidth="100.0" />
```

```
        </columnConstraints>
        <rowConstraints>
          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
/>
          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
/>
          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
/>
          <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
        </rowConstraints>
        <children>
          <Label text="Username:">
            <padding>
              <Insets right="50.0" />
            </padding>
          </Label>
          <TextField fx:id="txtUName" maxWidth="200.0"
onAction="#btnConfirmClick" prefWidth="200.0" promptText="Username"
GridPane.columnIndex="1" />
          <Label text="Password:" GridPane.rowIndex="1">
            <padding>
              <Insets right="50.0" />
            </padding>
          </Label>
          <Button id="btnLogin" fx:id="btnConfirm" minWidth="100.0"
mnemonicParsing="false" onAction="#btnConfirmClick" text="Create Profile"
GridPane.rowIndex="3">
            <GridPane.margin>
              <Insets top="10.0" />
            </GridPane.margin>
```

```
                </Button>
                <Label fx:id="lblCPass" layoutX="140.0" layoutY="47.0" text="Confirm
Password:" GridPane.rowIndex="2">
                  <padding>
                    <Insets right="50.0" />
                  </padding>
                </Label>
                <PasswordField fx:id="txtPass" onAction="#btnConfirmClick"
promptText="Password" GridPane.columnIndex="1" GridPane.rowIndex="1" />
                <PasswordField fx:id="txtCPass" onAction="#btnConfirmClick"
promptText="Password" GridPane.columnIndex="1" GridPane.rowIndex="2" />
              </children>
              <padding>
                <Insets left="100.0" right="100.0" />
              </padding>
            </GridPane>
          </children>
          <BorderPane.margin>
            <Insets />
          </BorderPane.margin>
        </VBox>
      </center>
      <bottom>
        <HBox alignment="CENTER" style="-fx-background-color: #003366;"
BorderPane.alignment="CENTER">
          <children>
            <Label text="(C) Grocery Pal 2018. All Rights Reserved." textFill="WHITE" />
          </children>
        </HBox>
      </bottom>
```

```
</BorderPane>
```

5.5. Design.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.web.*?>
<?import javafx.scene.effect.*?>
<?import javafx.scene.*?>
<?import javafx.scene.media.*?>
<?import java.lang.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>

<BorderPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="500.0" prefWidth="700.0" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="grocery_pal.ui.HomePageController">
  <top>
    <MenuBar BorderPane.alignment="CENTER">
      <menus>
        <Menu mnemonicParsing="false" text="Options">
          <items>
            <MenuItem fx:id="loginMenuItem" mnemonicParsing="false" onAction="#loginMenuItemClick" text="Login" />
            <MenuItem mnemonicParsing="false" onAction="#cProfMenuItemClick" text="Create New Profile" />
            <SeparatorMenuItem mnemonicParsing="false" />
            <MenuItem mnemonicParsing="false" onAction="#closeMenuItemClick" text="Close" />
```

```
        </items>
      </Menu>
      <Menu mnemonicParsing="false" text="Help">
        <items>
          <MenuItem mnemonicParsing="false" text="About" />
        </items>
      </Menu>
    </menus>
  </MenuBar>
 </top>
</BorderPane>
```

5.6. GroceryItem.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.scene.shape.*?>
<?import javafx.scene.paint.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.text.*?>
<?import javafx.scene.web.*?>
<?import javafx.scene.effect.*?>
<?import javafx.scene.*?>
<?import javafx.scene.media.*?>
<?import java.lang.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>
```

```
<BorderPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="330.0" prefWidth="400.0" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="grocery_pal.ui.GroceryItemController">
   <center>
      <VBox alignment="TOP_CENTER" style="-fx-background-color: white;" BorderPane.alignment="CENTER">
         <children>
            <Label text="Add Grocery Item" VBox.vgrow="ALWAYS">
               <font>
                  <Font size="36.0" />
               </font>
               <padding>
                  <Insets bottom="10.0" />
               </padding>
            </Label>
            <Line endX="300.0" opacity="0.4" startX="-100.0" stroke="#006633" strokeLineCap="ROUND" strokeLineJoin="ROUND" strokeWidth="2.0" />
            <Label fx:id="lblErr" text="*Please enter a valid number for the preferred quantity." textFill="#d61515" visible="false">
               <VBox.margin>
                  <Insets top="10.0" />
               </VBox.margin>
               <padding>
                  <Insets bottom="10.0" />
               </padding>
            </Label>
            <GridPane minHeight="-Infinity" minWidth="-Infinity" prefHeight="220.0" prefWidth="300.0">
               <columnConstraints>
```

```
            <ColumnConstraints hgrow="SOMETIMES" maxWidth="173.0"
minWidth="10.0" prefWidth="149.0" />
            <ColumnConstraints hgrow="SOMETIMES" maxWidth="227.0"
minWidth="10.0" prefWidth="173.0" />
         </columnConstraints>
         <rowConstraints>
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
/>
            <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
/>
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
/>
            <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
         </rowConstraints>
          <children>
            <Label text="Item Name:">
               <padding>
                  <Insets right="50.0" />
               </padding>
            </Label>
            <TextField fx:id="txtName" maxWidth="210.0" onAction="#btnConfirmClick"
prefHeight="25.0" prefWidth="210.0" GridPane.columnIndex="1" />
            <Button id="btnLogin" fx:id="btnConfirm" minWidth="100.0"
mnemonicParsing="false" onAction="#btnConfirmClick" text="Add Item"
GridPane.rowIndex="4">
               <GridPane.margin>
                  <Insets top="10.0" />
```

```
                </GridPane.margin>
            </Button>
            <Label text="Description:" GridPane.rowIndex="1" />
            <TextArea fx:id="txtDescription" prefHeight="200.0" prefWidth="200.0"
GridPane.columnIndex="1" GridPane.rowIndex="1" GridPane.rowSpan="3" />
        </children>
        <padding>
            <Insets left="30.0" right="30.0" />
        </padding>
        <VBox.margin>
            <Insets />
        </VBox.margin>
    </GridPane>
  </children>
  <BorderPane.margin>
    <Insets />
  </BorderPane.margin>
 </VBox>
</center>
<bottom>
 <HBox alignment="CENTER" style="-fx-background-color: #003366;"
BorderPane.alignment="CENTER">
  <children>
    <Label text="(C) Grocery Pal 2018. All Rights Reserved." textFill="WHITE" />
  </children>
 </HBox>
</bottom>
</BorderPane>
```

5.7. LoginDesign.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.scene.shape.*?>

<?import javafx.scene.paint.*?>

<?import javafx.geometry.*?>

<?import javafx.scene.text.*?>

<?import javafx.scene.web.*?>

<?import javafx.scene.effect.*?>

<?import javafx.scene.*?>

<?import javafx.scene.media.*?>

<?import java.lang.*?>

<?import javafx.scene.control.*?>

<?import javafx.scene.layout.*?>

<?import javafx.scene.layout.AnchorPane?>


<BorderPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-
Infinity" minWidth="-Infinity" prefHeight="260.0" prefWidth="500.0"
xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="grocery_pal.ui.LoginController">
  <bottom>
    <HBox alignment="CENTER" style="-fx-background-color: #003366;"
BorderPane.alignment="CENTER">
      <children>
        <Label text="(C) Grocery Pal 2018. All Rights Reserved." textFill="WHITE" />
      </children>
    </HBox>
  </bottom>
  <center>
```

```
    <VBox alignment="CENTER" style="-fx-background-color: white;" BorderPane.alignment="CENTER">
        <children>
            <Label text="Login" textFill="#003366" VBox.vgrow="ALWAYS">
                <font>
                    <Font size="36.0" />
                </font>
                <padding>
                    <Insets bottom="10.0" />
                </padding>
            </Label>
            <Line endX="300.0" opacity="0.4" startX="-100.0" stroke="#006633" strokeLineCap="ROUND" strokeLineJoin="ROUND" strokeWidth="2.0" />
            <Label fx:id="lblErr" text="Incorrect username or password." textFill="#d61515" visible="false">
                <padding>
                    <Insets bottom="20.0" />
                </padding>
            </Label>
            <GridPane>
                <children>
                    <Label text="Username:">
                        <padding>
                            <Insets right="50.0" />
                        </padding>
                    </Label>
                    <TextField fx:id="txtUName" maxWidth="200.0" onAction="#btnLoginClick" prefWidth="200.0" promptText="Username" GridPane.columnIndex="1" />
                    <Label text="Password:" GridPane.rowIndex="1">
                        <padding>
```

```
                <Insets right="50.0" />
              </padding>
            </Label>
            <Button id="btnLogin" fx:id="btnLogin" minWidth="100.0"
mnemonicParsing="false" onAction="#btnLoginClick" text="Log In"
GridPane.rowIndex="2">
              <GridPane.margin>
                <Insets top="10.0" />
              </GridPane.margin>
            </Button>
            <PasswordField fx:id="txtPass" onAction="#btnLoginClick"
promptText="Password" GridPane.columnIndex="1" GridPane.rowIndex="1" />
          </children>
          <columnConstraints>
            <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0"
prefWidth="100.0" />
            <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0"
prefWidth="100.0" />
          </columnConstraints>
          <padding>
            <Insets left="100.0" right="100.0" />
          </padding>
          <rowConstraints>
            <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
          </rowConstraints>
        </GridPane>
```

```
        </children>
        <BorderPane.margin>
            <Insets />
        </BorderPane.margin>
    </VBox>
</center>
</BorderPane>
```