Code Appendix

ClearBtn.java

```
/*
 * Class which creates the clear overlays on meal section and recipe buttons
 * includes specific actions for on click in both buttons
 */

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class ClearBtn extends JButton implements ActionListener {
	/*
	 * type differentiates between the clear overlay button used to open a certain
	 * meal section(on first window after login) and the other overlay used to open
	 * a recipe (on top of recipe thumbnail panels)
	 */

	private int type;
	// name refers to either Meal Section name or recipe name
	private String name;

	// if a new KitchenSpace is to be created in the case of a meal section click
	public static RecipesUnderMS ks;

	// rawName is the specific reference in the database to a recipe and is needed
	// for a recipe click
	private String rawName;

	public ClearBtn(String name, String rawName, int n) {
		this.name = name;
		this.type = n;
		this.rawName = rawName;

		setBorder(null);
		setBorderPainted(false);
		setContentAreaFilled(false);
		setOpaque(false);
		addActionListener(this);

	}

	@Override
	public void actionPerformed(ActionEvent e) {
		// if meal section is clicked
		if (e.getSource() == this && type == 1) {
			// name is meal section name
			ks = new RecipesUnderMS(name);
			// dispose of the previous frame so as to allow for only the updated Main
			// kitchen to display which will be created at a later point
			MealSectionsUnderAccount.frame.dispose();

		}

		// if a recipe is clicked
		if (e.getSource() == this && type == 2) {
			ks.dispose();
			MSTemplate.ks.frame.dispose();
			MealSectionsUnderAccount.frame.dispose();
			RecipeTemplate selectedRecipe = new RecipeTemplate(name,
											rawName);
			selectedRecipe.populateRecipe();
			selectedRecipe.repaint();
		}

	}

}
```

DifficultyRate.java

```
/**
 * Create the difficult rate bar on the recipe template
 * has easy, medium and hard buttons and changes colour on hover and on click
 */
import java.awt.Color;
import java.awt.Font;
import java.util.ArrayList;
import javax.swing.*;

public class DifficultyRate extends JPanel {
	private JLabel diffLab;
	private LevelBtn easy;
	private LevelBtn med;
	private LevelBtn hard;
	public static ArrayList<LevelBtn> btnList;
	public DifficultyRate() {

		btnList = new ArrayList<>();
		// reset static variables that track selected options
		LevelBtn.lvlTotal = 0;
		LevelBtn.selectedBtn = -1;
		setSize(400, 40);
		setBackground(Color.white);
		diffLab = new JLabel("Level: ");
		diffLab.setFont(new Font("Arial", Font.BOLD, 20));
		diffLab.setForeground(Color.decode("#9B9B9B"));
		add(diffLab);

		easy = new LevelBtn("Easy");
		med = new LevelBtn("Medium");
		hard = new LevelBtn("Hard");

		add(easy);
		add(med);
		add(hard);

		btnList.add(easy);
		btnList.add(med);
		btnList.add(hard);
	}

	public int getSelectedDiff() {
		return LevelBtn.selectedBtn;
	}

	/**
	 * method for updating the colour of the buttons so only the selected is
	 */
	// coloured in
	public void repaintSelectedDiff() {
		for (int i = 0; i < 3; i++) {
			if (i != getSelectedDiff()) {
				DifficultyRate.btnList.get(i).setBackground(Color.white);
			} else {

DifficultyRate.btnList.get(i).setBackground(Color.decode("#97EDA5"));

			}
		}
	}

}
```

Dragger.java

```
/**
 * Class for creating image drag and drop field
 * Allows for custom sizes for Recipe and Meal Section form
 * Connects to DragListener
 */
import java.awt.Color;
import java.awt.dnd.DropTarget;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.BorderFactory;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.border.Border;

public class Dragger extends JPanel {
	private JLabel imageLabel;
	private JLabel pathLabel;
	private int width;
	private int height;
	public String loadedImg = "";
	private DragListener d;

	public Dragger() {
		Border border = BorderFactory.createDashedBorder(Color.decode("#9B9B9B"),
										2, 1, 3, true);
		setBorder(border);
		imageLabel = new JLabel();
		pathLabel = new JLabel();
		add(imageLabel);
		connectToDragDrop();
	}

	// allows for custom sizes for either Recipe or Meal Section
	public Dragger(int width, int height) {
		this.width = width;
		this.height = height;
```

```
        Border border = BorderFactory.createDashedBorder(Color.decode("#9B9B9B"),
                                                                      2, 1, 3, true);
        setBorder(border);
        imageLabel = new JLabel();
        pathLabel = new JLabel();
        add(imageLabel);
        // establishes constant connection to DragListener class
        connectToDragDrop(width, height);
}

// method to show the image after it has been dropped
public void loadImg(String path) {
        d.displayImage(path);
}

// methods for connecting to DragListener
private void connectToDragDrop() {

        d = new DragListener(imageLabel, pathLabel);

        new DropTarget(this, d);
}

private void connectToDragDrop(int width, int height) {

        d = new DragListener(imageLabel, pathLabel, width, height);

        new DropTarget(this, d);
}

// returns the image path of the image copied to the local "images" file. NOT
// THE ABSOLUTE PATH
public String getImagePath() {
        return DragListener.newImagePath;


}

/**
 * method for adding local image path to database
 * @param nameMS is concatenated with the imgPath
 * @param table specifies the table specific to the user
 * @param col is the recipe column
 */
	public void addImgPathToDB(String nameMS, String table, String col) {
		String imgPath = nameMS + "|" + "imgPath:" + getImagePath();


		// for no image
		if (getImagePath() == "") {
			imgPath = nameMS + "|" + "imgPath:" + "empty";

		}

		// change double backslash \\ to forward slash / for the purposes of retrieval
		// and display from database
		imgPath = imgPath.replaceAll((char) 92 + "" + (char) 92, (char) 47 + "");

		try {
			String url = "jdbc:sqlite:db.db";
			try {
				Class.forName("org.sqlite.JDBC");
			} catch (ClassNotFoundException e) {
				e.printStackTrace();
			}
			Connection con = DriverManager.getConnection(url);
			Statement stmt = con.createStatement();

			String query = "INSERT INTO '" + table + "' ('" + col + "') VALUES ('" +
											imgPath + "')";

			stmt.execute(query);
			con.close();
		} catch (SQLException e) {
			e.printStackTrace();
		}

		// reset strings to empty in case of another loaded image after
		DragListener.newImagePath = "";

	}

}
```

DragListener.java

```
/*
 * DragListener which connects to Dragger panel
 * implements DropTargetListener interface
 */
import java.awt.Image;
import java.awt.datatransfer.DataFlavor;
import java.awt.datatransfer.Transferable;
import java.awt.dnd.DnDConstants;
import java.awt.dnd.DropTargetDragEvent;
import java.awt.dnd.DropTargetDropEvent;
import java.awt.dnd.DropTargetEvent;
import java.awt.dnd.DropTargetListener;
import java.awt.image.BufferedImage;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JLabel;

public class DragListener implements DropTargetListener {
	JLabel imageLabel = new JLabel();
	JLabel pathLabel = new JLabel();
	// stores absolute path
	public static String newImagePath="";

	private int width;
	private int height;

	// DragListener sized to recipe
	public DragListener(JLabel image, JLabel path) {
		newImagePath = "";
		width = 230;
		height = 230;
		imageLabel = image;
		pathLabel = path;

	}

	// DragListener sized to meal section
	public DragListener(JLabel image, JLabel path, int width, int height) {
		this.width = width;
		this.height = height;
		imageLabel = image;
		pathLabel = path;
	}
	/**
	 * method for displaying the image on the field
	 * @param path path of inputted image
	 */
	public void showImg(String path) {
		displayImage(pathLabel.getText());
	}

	@Override
	public void dragEnter(DropTargetDragEvent dtde) {
		// TODO Auto-generated method stub

	}

	@Override
	public void dragOver(DropTargetDragEvent dtde) {
		// TODO Auto-generated method stub

	}

	@Override
	public void dropActionChanged(DropTargetDragEvent dtde) {
		// TODO Auto-generated method stub

	}

	@Override
	public void dragExit(DropTargetEvent dte) {
		// TODO Auto-generated method stub

	}

	@Override

	/**
	 * Copies dropped images to internal folder
	 */
	public void drop(DropTargetDropEvent ev) {
		ev.acceptDrop(DnDConstants.ACTION_COPY);
```

```
Transferable t = ev.getTransferable();

DataFlavor[] df = t.getTransferDataFlavors();

for (DataFlavor f : df) {
    try {
        if (f.isFlavorJavaFileListType()) {

            @SuppressWarnings("unchecked")
            List<File> files = (List<File>) t.getTransferData(f);

            for (File file : files) {

                // copying file to local images folder using file input
                                and output streams

                FileInputStream in = new
                    FileInputStream(file.getAbsolutePath());
                Path path = Paths.get(file.getPath());
                Path fileName = path.getFileName();
                String s = fileName.toString();

                // change path name
                FileOutputStream ou = new
                        FileOutputStream("images\\" + s);

                BufferedInputStream bin = new
                            BufferedInputStream(in);

                BufferedOutputStream bou = new
                            BufferedOutputStream(ou);
                int b = 0;
                while (b != -1) {
                    b = bin.read();
                    bou.write(b);
                }
                bin.close();
                bou.close();

                // update string name
                newImagePath = "images\\" + s;

                displayImage(newImagePath);
            }
```

```
                    }

                } catch (Exception e) {
                    e.printStackTrace();

                }
            }
        }

        /**
         * method for displaying the image in the field
         * reads images, scales them, and resets path strings
         * @param path is the path of the string to display
         */
        public void displayImage(String path) {
            BufferedImage img = null;
            try {
                img = ImageIO.read(new File(path));
                ImageIcon icon = new ImageIcon(img);
                Image image = icon.getImage();

                // size of recipe image field (230,230)
                Image newImage = image.getScaledInstance(width, height,
                            java.awt.Image.SCALE_SMOOTH);

                icon = new ImageIcon(newImage);

                imageLabel.setIcon(icon);

                newImagePath = path;
                pathLabel.setText(path);
            } catch (Exception e) {

            }

        }

}
```

LevelBtn.java

```
/**
 * Individual difficulty button on DifficultyRate bar
 */
```

```
import java.awt.Color;
import java.awt.Font;
import java.awt.event.*;
import javax.swing.*;

public class LevelBtn extends JButton implements ActionListener, MouseListener {
	// numeric ID of 0, 1, 2 for each button
	private int lvlID;
	// lvlTotal used to assign a number of 0, 1, 2 to difficulty rate buttons in
	// order to ID them
	public static int lvlTotal = 0;
	// set none selected at first
	public static int selectedBtn = -1;

	public LevelBtn(String text) {

		lvlID = lvlTotal;
		lvlTotal++;

		setText(text);
		setFont(new Font("Arial", Font.PLAIN, 15));
		setBackground(Color.WHITE);
		setForeground(Color.decode("#55A630"));
		setFocusable(false);
		setBorderPainted(false);
		addActionListener(this);
		addMouseListener(this);

	}

	@Override
	/**
	 * Detects a mouseclick on the level button and changes the colour of the selected
	                                                                   button
	 * resets the colour of non selected buttons
	 */
	public void mouseClicked(MouseEvent e) {
		// ensures colour persists only on selected button
		selectedBtn = lvlID;
		setBackground(Color.decode("#97EDA5"));
		for (int i = 0; i < 3; i++) {
			if (i != lvlID) {
				DifficultyRate.btnList.get(i).setBackground(Color.white);
			}
		}
		repaint();
	}

	@Override
	public void mousePressed(MouseEvent e) {
		// TODO Auto-generated method stub

	}

	@Override
	public void mouseReleased(MouseEvent e) {
		// TODO Auto-generated method stub

	}

	@Override
	public void mouseEntered(MouseEvent e) {
		// changes colour on hover
		setBackground(Color.decode("#97EDA5"));

	}

	@Override
	public void mouseExited(MouseEvent e) {
		// deselects colour on exit
		if (selectedBtn == lvlID) {

		} else {
			setBackground(Color.WHITE);
		}
		repaint();

	}

	@Override
	public void actionPerformed(ActionEvent e) {
		// TODO Auto-generated method stub

	}

}
```

ListForm.java

```
/**
 * Template for the extendable lists built jframes with a green button present on Meal section
and Recipe form
 * Handles spacing, storage of textfield values
 * Responsible for sending text info to the db
 */
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import javax.swing.*;

class ListForm extends JPanel implements ActionListener {

        private ArrayList<JTextField> elmntList;
        private JButton elmntAdd;
        private JLabel label;
        private int spacing;
        public String header;
        private static int totalSpacingRecip;
        private static int totalSpacingMS;
        private static int scrollCountRecip;
        private static int scrollCountMS;

        private int formID;

        // counting listforms created
        public static int totalRecip = 0;
        public static int totalMS = 0;

        // keeping track of the type (either Recipe or Meal Section)
        private static int track = 0;

        public ListForm(String header, int n) {

                totalSpacingRecip = 0;
                totalSpacingMS = 0;
                scrollCountRecip = 0;
                scrollCountMS = 0;
```

```
    }

        // when n = 1 indicates Recipe
        // when n = 2 indicates Meal Section
        track = n;
        if (n == 1) {
            formID = totalRecip;
            totalRecip++;
        }

        else if (n == 2) {
            formID = totalMS;
            totalMS++;
        }

        this.header = header;
        setLayout(null);
        setBackground(Color.WHITE);
        spacing = 0;
        elmntList = new ArrayList<>();
        label = new JLabel(header);
        label.setFont(new Font("Arial", Font.BOLD, 25));
        label.setSize(300, 50);
        label.setLocation(10, 0);
        label.setForeground(Color.decode("#9B9B9B"));
        add(label);

        elmntAdd = new JButton();
        ImageIcon btn = new ImageIcon("systemImages/btn.png");
        elmntAdd.setIcon(btn);
        elmntAdd.setSize(70, 80);
        elmntAdd.setBackground(null);
        elmntAdd.setBorder(null);
        elmntAdd.addActionListener(this);
        elmntAdd.setLocation(60, 60);
        elmntAdd.setBorderPainted(false);
        elmntAdd.setFocusPainted(false);
        add(elmntAdd);

    }

    public int getSpacing() {
        return spacing;
    }
```

```
public void actionPerformed(ActionEvent e) {

        // for ListForms on a Recipe template
        if (e.getSource() == elmntAdd && track == 1) {
                int tfLength = 770;
                int tfWidth = 900;

                // Make ingrediant and substitution text fields smaller
                if (formID == 0 || formID == 1) {
                        tfLength = 300;
                        tfWidth = 400;
                }
                // subForm spacing not taken into consideration as it is never the case
                                                                                        that
                // substitutions outnumber actual ingrediants

                // size and position the text field
                TxtField field = new TxtField(tfLength);
                field.getTF().setLocation(50, 50 + spacing);
                elmntList.add(field.getTF());
                add(field.getTF());

                spacing += 60;

                // increase height of the panel with each addition of a textfied element
                setSize(tfWidth, 150 + spacing);

                // increase spacing of the panel
                totalSpacingRecip += 60;

                // move the create button down
                int yBtn = RecipeTemplate.yCreate + totalSpacingRecip;

                // increase scroll size of JScrollPane
                scrollCountRecip = 1000 + totalSpacingRecip;

                // change dimension of recipe template to accomodate for increasing
                                                                                  number of
                // textfields
                RecipeTemplate.container.setPreferredSize(new Dimension(0,
                                                                   scrollCountRecip));

                // move create button down
                RecipeTemplate.create.setLocation(405, yBtn);
```

```
		// move green add button lower
		elmntAdd.setLocation(60, 80 + spacing);

		// if the first list form is extended, move the 2 below it down
		if (formID == 0) {
			int insY = RecipeTemplate.insY +
						RecipeTemplate.ingForm.getSpacing();
			RecipeTemplate.allForms.get(2).setLocation(20, insY);

			int noteY = RecipeTemplate.noteY +
					RecipeTemplate.instructForm.getSpacing() +
				RecipeTemplate.ingForm.getSpacing();
			RecipeTemplate.allForms.get(3).setLocation(20, noteY);
		}

		// if the second list form is extended, only move the last down
		else if (formID == 2) {
			int noteY = RecipeTemplate.noteY +
			RecipeTemplate.instructForm.getSpacing() +
			RecipeTemplate.ingForm.getSpacing();
			RecipeTemplate.allForms.get(3).setLocation(20, noteY);
		}

	}

	// For listForms on Meal Section template
	else if (e.getSource() == elmntAdd && track == 2) {
		int tfLength = 770;
		int tfWidth = 900;

		TxtField field = new TxtField(tfLength);
		field.getTF().setLocation(50, 50 + spacing);
		elmntList.add(field.getTF());
		add(field.getTF());

		spacing += 60;

		// with the addition of another textfield, increase vertical spacing
		setSize(tfWidth, 150 + spacing);

		// increase overall spacing
		totalSpacingMS += 60;
```

```
                // move create button down
                int yBtn = MSTemplate.yCreate + totalSpacingMS;
                scrollCountMS = 1150 + totalSpacingMS;
                MSTemplate.container.setPreferredSize(new Dimension(0,
                                                        scrollCountMS));
                MSTemplate.create.setLocation(405, yBtn);
                elmntAdd.setLocation(60, 80 + spacing);

                // if first list form is extended, move both of the following sections down
                if (formID == 0) {
                    int primFY = MSTemplate.primFY +
                                    MSTemplate.descForm.getSpacing();
                    MSTemplate.allForms.get(1).setLocation(20, primFY);

                    int culnGY = MSTemplate.culnGY +
                                    MSTemplate.primFForm.getSpacing()
                                + MSTemplate.descForm.getSpacing();
                    MSTemplate.allForms.get(2).setLocation(20, culnGY);
                }

                // if second list form is extended, only move the one directly under it down
                else if (formID == 1) {
                    int culnGY = MSTemplate.culnGY +
MSTemplate.primFForm.getSpacing()
                                + MSTemplate.descForm.getSpacing();
                    MSTemplate.allForms.get(2).setLocation(20, culnGY);
                }
            }
        }

        /**
         * method for populating a recipe list so that previous recipes are loaded from
         * the db and displayed
         * @param addStr string that will be put into the listform text field
         */
        public void populateRecipeListForm(String addStr) {
            int tfLength = 770;
            int tfWidth = 900;
            if (formID == 0 || formID == 1) {
                tfLength = 300;
                tfWidth = 400;
            }

            TxtField field = new TxtField(tfLength);
        // adding the provided text read from db
        field.getTF().setText(addStr);
        field.getTF().setLocation(50, 50 + spacing);
        elmntList.add(field.getTF());
        add(field.getTF());

        spacing += 60;

        setSize(tfWidth, 150 + spacing);

        totalSpacingRecip += 60;
        int yBtn = RecipeTemplate.yCreate + totalSpacingRecip;
        scrollCountRecip = 1000 + totalSpacingRecip;
        RecipeTemplate.container.setPreferredSize(new Dimension(0,
                                                        scrollCountRecip));
        RecipeTemplate.create.setLocation(405, yBtn);
        elmntAdd.setLocation(60, 80 + spacing);

        if (formID == 0) {
                int insY = RecipeTemplate.insY + RecipeTemplate.ingForm.getSpacing();
                RecipeTemplate.allForms.get(2).setLocation(20, insY);

                int noteY = RecipeTemplate.noteY +
                RecipeTemplate.instructForm.getSpacing() +
                RecipeTemplate.ingForm.getSpacing();
                RecipeTemplate.allForms.get(3).setLocation(20, noteY);
        } else if (formID == 2) {
                int noteY = RecipeTemplate.noteY +
                RecipeTemplate.instructForm.getSpacing() +
                RecipeTemplate.ingForm.getSpacing();
                RecipeTemplate.allForms.get(3).setLocation(20, noteY);
        }

}

/**
 * method for populating meal section list forms
 * @param addStr is string to be added to the listform
 */
public void populateMSListForm(String addStr) {
        int tfLength = 770;
        int tfWidth = 900;
```

```
		TxtField field = new TxtField(tfLength);

		// add inputted text
		field.getTF().setText(addStr);
		field.getTF().setLocation(50, 50 + spacing);
		elmntList.add(field.getTF());
		add(field.getTF());

		spacing += 60;

		setSize(tfWidth, 150 + spacing);

		totalSpacingMS += 60;
		int yBtn = MSTemplate.yCreate + totalSpacingMS;
		scrollCountMS = 1150 + totalSpacingMS;
		MSTemplate.container.setPreferredSize(new Dimension(0, scrollCountMS));
		MSTemplate.create.setLocation(405, yBtn);
		elmntAdd.setLocation(60, 80 + spacing);

		if (formID == 0) {
			int primFY = MSTemplate.primFY + MSTemplate.descForm.getSpacing();
			MSTemplate.allForms.get(1).setLocation(20, primFY);

			int culnGY = MSTemplate.culnGY + MSTemplate.primFForm.getSpacing()
					+ MSTemplate.descForm.getSpacing();
			MSTemplate.allForms.get(2).setLocation(20, culnGY);
		} else if (formID == 1) {
			int culnGY = MSTemplate.culnGY + MSTemplate.primFForm.getSpacing()
					+ MSTemplate.descForm.getSpacing();
			MSTemplate.allForms.get(2).setLocation(20, culnGY);
		}

	}

	/**
	 * method for formatting data and adding to the db
	 * I want data to enter the db as such:
	 * [MealSectionName]: ing1,ing2,ing3
	 */
	public void addInfoToDB() {
		String s = header + ":";
		if (elmntList.size() > 0) {
			for (JTextField t : elmntList) {
				if (t.getText() != "") {
					s += t.getText() + ",";
				}

			}
		} else {
			s += "empty";
		}
		elmntList.clear();

		try {
			String url = "jdbc:sqlite:db.db";
			try {
				Class.forName("org.sqlite.JDBC");
			} catch (ClassNotFoundException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			Connection con = DriverManager.getConnection(url);
			Statement stmt = con.createStatement();
			stmt.execute(formatQuery(s));
			con.close();
		} catch (SQLException e1) {
			JOptionPane.showMessageDialog(null, "Recipe already defined");
			e1.printStackTrace();
		}
	}

	/**
	 * follow same string format for meal section items
	 * but instead of adding directly to db, return the value
	 */
	public String formatMSList() {
		String s = header + ":";
		if (elmntList.size() > 0) {
			for (JTextField t : elmntList) {
				if (t.getText() != "") {
					s += t.getText() + ",";
				}

			}
		}

		else {
			s += "empty";
		}
		elmntList.clear();
		return s;
	}

	/**
	 * method for adding data from the block of star rating, image, difficulty
	 * rating, times, and yield
	 */
	public static void addOptsInfo() {
		String starRate = "Star Rating:" + ((StarRate)
		OtherOptions.optArr.get(0)).getSelectedStar() + ",";

		String imgPath = "imgPath:" + ((Dragger)
		OtherOptions.optArr.get(1)).getImagePath() + ",";

		// replace \\ with /
		imgPath = imgPath.replaceAll((char) 92 + "" + (char) 92, (char) 47 + "");

		String difRate = "dif Rating:" + ((DifficultyRate)
		OtherOptions.optArr.get(2)).getSelectedDiff() + ",";

		String prepTime = "prep Time:" + ((TimePanel)
		OtherOptions.optArr.get(3)).getHours() + ","
				+ ((TimePanel) OtherOptions.optArr.get(3)).getMins() + ",";

		String cookTime = "cook Time:" + ((TimePanel)
		OtherOptions.optArr.get(4)).getHours() + ","
				+ ((TimePanel) OtherOptions.optArr.get(4)).getMins() + ",";

		// make so that yield and all other textfields cannot add commas (unnecessary
		// and messes up retrieval from db)

		String yield = "yield:" + ((TimePanel) OtherOptions.optArr.get(5)).getYield() + ",";

		// add data to db
		try {
			String url = "jdbc:sqlite:db.db";
			try {
				Class.forName("org.sqlite.JDBC");
			} catch (ClassNotFoundException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			Connection con = DriverManager.getConnection(url);
			Statement stmt = con.createStatement();

			stmt.execute(formatQuery(starRate));
			stmt.execute(formatQuery(imgPath));
			stmt.execute(formatQuery(difRate));
			stmt.execute(formatQuery(prepTime));
			stmt.execute(formatQuery(cookTime));
			stmt.execute(formatQuery(yield));

			con.close();
		} catch (SQLException e1) {
			JOptionPane.showMessageDialog(null, "Could not connect to database,
								something went wrong");
			e1.printStackTrace();
		}

	}

	/**
	 * method for preparing the query that sends info to db
	 * @param s text to be sent to database
	 * @return formatted query
	 */
	public static String formatQuery(String s) {

		// for empty values
		if (s.charAt(s.length() - 1) == ':' || s.charAt(s.length() - 1) == ',') {
			s.substring(0, s.length() - 2);
			s += "empty";
		}
		// making updates to a recipe, just insert the data into the same recipe
		if (RecipeTemplate.editingRecip) {
			return "INSERT INTO '" + CreateLoginForm.currUser + "' ('" +
				RecipeTemplate.newColName + "') VALUES ('" + s + "')";

		}

		// if making a new recipe,
		if (track == 1) {
			return "INSERT INTO '" + CreateLoginForm.currUser + "' ('" +
					RecipeTemplate.colName + "') VALUES ('" + s + "')";
		}
```

```
			// if making a new meal section
			String temp = MSTemplate.msName + "|";
			temp += s;
			s = temp;
			return "INSERT INTO '" + CreateLoginForm.currUser + "' (MealSection) VALUES
('" + s + "')";

		}

}
```

LoginPage.java

```
/**
 * Class that creates the login form and begins the program
 * Includes general login page and create new account form
 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

class CreateLoginForm extends JFrame implements ActionListener {
		private Container c;
		private JButton cont;
		private JButton newAcc;
		private JButton create;
		private JLabel title, desc, noAcc, userLab, passLab, confirmPassLab;
		private final JTextField txt1;
		private final JTextField txt2;
		private JTextField txt3;
		public static String currUser;
		public static String userName;

		public static MealSectionsUnderAccount ms;

		public CreateLoginForm(String s) {
			setTitle("Login");
			setBounds(300, 90, 900, 650);
			setDefaultCloseOperation(EXIT_ON_CLOSE);
			setResizable(false);
			c = getContentPane();
			c.setLayout(null);
			c.setBackground(Color.white);
```

```
title = new JLabel("The Pantry");
title.setFont(new Font("Arial", Font.BOLD, 60));
title.setForeground(Color.decode("#1E1E1E"));
title.setSize(400, 100);
title.setLocation(295, 40);
c.add(title);

desc = new JLabel("Cook, eat, enjoy.");
desc.setFont(new Font("Arial", Font.PLAIN, 27));
desc.setForeground(Color.decode("#55A630"));
desc.setSize(400, 100);
desc.setLocation(360, 110);
c.add(desc);

userLab = new JLabel("Username");
userLab.setFont(new Font("Arial", Font.PLAIN, 18));
userLab.setSize(100, 15);
userLab.setLocation(300, 225);
c.add(userLab);

txt1 = new JTextField();
txt1.setFont(new Font("Arial", Font.PLAIN, 30));
txt1.setSize(300, 50);
txt1.setLocation(300, 250);

c.add(txt1);

passLab = new JLabel("Passcode");
passLab.setFont(new Font("Arial", Font.PLAIN, 18));
passLab.setSize(150, 20);
passLab.setLocation(300, 325);
c.add(passLab);

txt2 = new JPasswordField();
txt2.setFont(new Font("Arial", Font.PLAIN, 30));
txt2.setSize(300, 50);
txt2.setLocation(300, 350);
c.add(txt2);

setVisible(true);

// Login option display
if (s.equals("Login")) {
        cont = new JButton("Continue");
        cont.setFont(new Font("Arial", Font.PLAIN, 18));
        cont.setSize(130, 40);
        cont.setForeground(Color.white);
        cont.setBackground(Color.decode("#55A630"));
        cont.setLocation(380, 420);
        c.add(cont);

        noAcc = new JLabel("Don't have an account?");
        noAcc.setFont(new Font("Arial", Font.PLAIN, 18));
        noAcc.setSize(300, 100);
        noAcc.setLocation(350, 440);
        c.add(noAcc);

        newAcc = new JButton("Create a new Account");
        newAcc.setFont(new Font("Arial", Font.PLAIN, 15));
        newAcc.setSize(190, 40);
        newAcc.setLocation(350, 515);
        newAcc.setBackground(Color.WHITE);
        newAcc.setForeground(Color.decode("#55A630"));
        c.add(newAcc);

        cont.addActionListener(this);
        newAcc.addActionListener(this);

}

// creating new acc display
else if (s.equals("New acc")) {

        setTitle("Create an account");

        title.setText("Welcome to the Pantry");
        title.setFont(new Font("Arial", Font.BOLD, 50));
        title.setSize(600, 150);
        title.setLocation(190, 30);

        desc.setText("Let's Get Cooking!");
        desc.setFont(new Font("Arial", Font.PLAIN, 27));
        desc.setSize(450, 100);
        desc.setLocation(340, 120);
        c.add(desc);
```

```
            txt3 = new JPasswordField();
            txt3.setFont(new Font("Arial", Font.PLAIN, 30));
            txt3.setSize(300, 50);
            txt3.setLocation(300, 445);
            c.add(txt3);

            confirmPassLab = new JLabel("Confirm Passcode");
            confirmPassLab.setFont(new Font("Arial", Font.PLAIN, 18));
            confirmPassLab.setSize(150, 20);
            confirmPassLab.setLocation(300, 420);
            c.add(confirmPassLab);

            create = new JButton("Create");
            create.setFont(new Font("Arial", Font.PLAIN, 15));
            create.setSize(100, 40);
            create.setLocation(405, 515);
            create.setForeground(Color.white);
            create.setBackground(Color.decode("#55A630"));

            c.add(create);

            create.addActionListener(this);
        }

}

public void actionPerformed(ActionEvent e) {

        // if pressed continue on login page
        if (e.getSource() == cont) {
            String username = txt1.getText();
            String passcode = txt2.getText();
            currUser = passcode + username;

            // if both passcode and username fields are empty
            if (username.equals("") && passcode.equals("")) {
                JOptionPane.showMessageDialog(this, "Enter Username and
                                                                Passcode");
            }

            // only username empty
            else if (username.equals("")) {
                JOptionPane.showMessageDialog(this, "Enter Username");
            }
```

```
// only passcode empty
else if (passcode.equals("")) {
        JOptionPane.showMessageDialog(this, "Enter Passcode");
}

// check if passcode and username exist in the db
else {
        try {
                String url = "jdbc:sqlite:db.db";
                Class.forName("org.sqlite.JDBC");
                Connection con = DriverManager.getConnection(url);

                // s is the userpasskey which is a concatenation of the
                                                passcode and username
                String s = passcode + username;
                PreparedStatement st = con.prepareStatement("SELECT *
                        from users WHERE userpasskey = ?");

                st.setString(1, s);
                ResultSet rs = st.executeQuery();

                // if found, go to the main kitchen
                if (rs.next()) {
                        userName = username;
                        ms = new MealSectionsUnderAccount();
                        dispose();
                }

                // userpasskey not found in db
                else {
                        JOptionPane.showMessageDialog(this, "Either
                                username or password is incorrect");
                }

                con.close();
        } catch (SQLException e1) {
                JOptionPane.showMessageDialog(this, "Something went
                                                                wrong");

        } catch (ClassNotFoundException e1) {
                e1.printStackTrace();
        }
}
```

```
}

// when creating a new acc
if (e.getSource() == newAcc) {
        CreateLoginForm createAccForm = new CreateLoginForm("New acc");
} else if (e.getSource() == create) {
        String username = txt1.getText();
        String passcode = txt2.getText();
        String passcodeCheck = txt3.getText();
        Boolean validName = false;

        // if atleast 1 character in the username is non-numeric, allow the
                                                                username
        for (int i = 0; i < username.length(); i++) {
                if (Character.isLetter(username.charAt(i))) {
                        validName = true;
                        break;
                }
        }

        // empty username
        if (username.equals("")) {
                JOptionPane.showMessageDialog(this, "Enter Username");
        }

        // if there are only numbers in username
        else if (!validName) {
                JOptionPane.showMessageDialog(this, "Username must include a
                                                        non-numeric character");
        }

        // empty passcode
        else if (passcode.equals("") || passcodeCheck.equals("")) {
                JOptionPane.showMessageDialog(this, "Enter Passcode");
        }

        // mismatched passcodes
        else if (!passcode.equals(passcodeCheck)) {
                JOptionPane.showMessageDialog(this, "Not the same
                                                                passcode");
        }

        // valid passcode and username
        // open up the new account which sends passcode and username to the
```

```
                                                                        database
                        else {
                                NewAccount newKitchen = new NewAccount(txt1.getText(),
                                                                        txt2.getText());
                                dispose();
                        }
                }

        }
}

class LoginPage {
        public static void main(String[] args) {
                try {
                        CreateLoginForm login = new CreateLoginForm("Login");

                } catch (Exception e) {
                        JOptionPane.showMessageDialog(null, e.getMessage());
                }
        }
}
```

MealSection.java

```
/**
 * Creates the main kitchen which houses all meal sections buttons
 * Has button to create new meal section
 */

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class MealSectionsUnderAccount extends JFrame implements ActionListener {

        private JPanel c;
        public static JFrame frame;
        private JScrollPane jsp;
        private JLabel sectName;
        private JButton sort;
        private JButton makeMS;
        private MSDataRetriever ms;
        private String sectText;
```

```
public static boolean sortSelected;

public MealSectionsUnderAccount() {

        frame = new JFrame();
        frame.setBounds(300, 90, 900, 650);
        frame.setResizable(false);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("Meal Sections");

        ms = new MSDataRetriever();

        c = new JPanel();
        c.setBackground(Color.white);

        jsp = new JScrollPane(c);
        c.setPreferredSize(new Dimension(0, 300 + (ms.yDim)));
        c.setLayout(new FlowLayout(FlowLayout.CENTER));
        c.setBackground(Color.white);

        sort = new JButton("Sort alphabetically");
        sort.setFont(new Font("Arial", Font.PLAIN, 30));
        sort.setSize(230, 60);
        sort.setBackground(Color.WHITE);
  sort.setForeground(Color.decode("#55A630"));
        sort.setFocusable(false);

        c.add(sort);

        sectText = CreateLoginForm.userName + "'s Pantry";
        sectName = new JLabel(sectText);
        sectName.setFont(new Font("Arial", Font.BOLD, 80));
        sectName.setForeground(Color.white);
        sectName.setBackground(Color.decode("#BAF2BB"));
        sectName.setPreferredSize(new Dimension(sectText.length() * 40, 130));
        sectName.setOpaque(true);

        c.add(sectName);

        c.add(ms);

        makeMS = new JButton("Add a meal section");
        makeMS.setFont(new Font("Arial", Font.PLAIN, 30));
        makeMS.setSize(230, 60);
```

```
			makeMS.setBackground(Color.WHITE);
			makeMS.setForeground(Color.decode("#55A630"));
			c.add(makeMS);

			makeMS.addActionListener(this);
			sort.addActionListener(this);

			frame.getContentPane().add(jsp);

			frame.setVisible(true);
		}

		@Override
		public void actionPerformed(ActionEvent e) {
			// to create a new meal section, dispose of current frame and load the meal
			// section template
			if (e.getSource() == makeMS) {
				MSTemplate newMS = new MSTemplate();
				frame.dispose();
			}

			if (e.getSource() == sort) {
				sortSelected = true;
				frame.dispose();
				CreateLoginForm.ms= new MealSectionsUnderAccount();

			}

		}

	}
```

MSDataRetriever.java

```
/**
 * Class which fetches all information about meal sections from db and places info in arraylist
 * Then, using the arraylist, the icons of each meal section are generated and organized onto a
panel
 */
import java.awt.Color;
import java.awt.Dimension;
import java.awt.FlowLayout;
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 javax.swing.JPanel;

public class MSDataRetriever extends JPanel {
	public static ArrayList<MSThumbnail> msArr;
	public static int count = 0;
	public int yDim;

	public MSDataRetriever() {

		count = 0;
		yDim = 0;
		setBackground(Color.white);
		setLayout(new FlowLayout(FlowLayout.CENTER));
		msArr = new ArrayList<>();

		try {
			ArrayList<String> strList = new ArrayList<>();
			String url = "jdbc:sqlite:db.db";
			try {
				Class.forName("org.sqlite.JDBC");
			} catch (ClassNotFoundException e) {
				e.printStackTrace();
			}
			Connection con = DriverManager.getConnection(url);
			String query = "SELECT MealSection FROM '" +
							CreateLoginForm.currUser + "'";
			Statement stmt = con.createStatement();
			ResultSet rs = stmt.executeQuery(query);

			// collecting Meal Section info from db
			while (rs.next()) {
				if (rs.getString(1) != null) {
					String name = "";
					String path = "";
					int count = -1;
					if (rs.getString(1).contains("imgPath:")) {
						// keep track of how many meal sections are loaded
									so as to change sizing of
						// panel
						this.count++;
```

```
                                        for (int i = 0; i < rs.getString(1).length(); i++) {
                                                // get the name of the meal section which
                                                                        preceeds the |
                                                if (rs.getString(1).charAt(i) != '|') {
                                                        name += rs.getString(1).charAt(i);

                                                        // count length of just the name
                                                        count++;
                                                }
                                                if (rs.getString(1).charAt(i) == '|') {
                                                        break;
                                                }
                                        }
                                        strList.add(name);

                                        // retrieve image path
                                        for (int i = count + 2; i < rs.getString(1).length(); i++)
                                        {
                                                path += rs.getString(1).charAt(i);
                                        }

                                        // create its own meal section thumbnail (image,
                                                                        descriptors)
                                        MSThumbnail newMs = new MSThumbnail(name,
                                                                                path);

                                        // add to current panel
                                        msArr.add(newMs);
                                }

                        }

                }

                rs.close();
                con.close();

        } catch (SQLException e) {
                e.printStackTrace();
        }

        if (count >= 2) {

                // if there is an even number of panels, set the height to be amount of
```

```
                                                                rows *
        // 250 pixels
        if (count % 2 == 0) {
                yDim = (count / 2) * 250;
        }

        // odd number of meal sections, there will be another row to house the
                                                                remaining
        // odd ones
        else {
                yDim = (count / 2 + 1) * 250;
        }
}

// if less than 2 meal sections, only create 1 row
else {
        yDim = 250;
}

setPreferredSize(new Dimension(900, yDim));

// sort alphabetically

MSThumbnail[] arr = new MSThumbnail[msArr.size()];
for (int i = 0; i<msArr.size(); i++) {
        arr[i] = msArr.get(i);
}

if (MealSectionsUnderAccount.sortSelected) {
        for (int i = 1; i<msArr.size(); i++) {
                MSThumbnail var = arr[i];
                int j = i-1;
                while (j>=0 &&
                var.getName().compareToIgnoreCase(arr[j].getName())<0) {
                        arr[j+1] = arr[j];
                        j--;

                }
                arr[j+1] = var;

        }
        MealSectionsUnderAccount.sortSelected = false;
}
```

```
			for (int i = 0; i<msArr.size(); i++) {
				add(arr[i]);
			}

		}
}
```

MSTemplate.java

```
/**
 * Format the new meal section template
 * includes the template for a new meal section and for editing an existing one
 * includes method for populating an existing meal section
 */
import java.awt.*;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import java.io.*;
import java.sql.*;
import java.util.*;
import javax.imageio.ImageIO;
import javax.swing.*;

public class MSTemplate extends JFrame implements ActionListener, FocusListener {

	private JFrame frame;
	public static JPanel container;
	private JLabel nameLab;
	private final JTextField name;
	public static int spacingY;
	private final JTextField dum;
	public static String msName = "";

	private JScrollPane jsp;

	// spacing variables
	public static int descY;
	public static int primFY;
	public static int culnGY;

	public static ListForm descForm;
	public static ListForm primFForm;
	public static ListForm culnGForm;
```

```
private Dragger imgField;
public static JButton create;
public static JButton deleteMS;

public static int yCreate;

public static boolean editingMS;

private String editableMSName;

// arrays for storing text field values
private ArrayList<String> descList;
private ArrayList<String> primFList;
private ArrayList<String> culnGoalsList;
private String imgPath;

public static RecipesUnderMS ks;

public static ArrayList<ListForm> allForms;

public MSTemplate() {

        // upon creation of a meal section form, dispose of the previous screen since it
        // will need to be updated anyway to display the new information
        MealSectionsUnderAccount.frame.dispose();
        imgPath = "";

        allForms = new ArrayList<ListForm>();
        editingMS = false;
        descList = new ArrayList<>();
        primFList = new ArrayList<>();
        culnGoalsList = new ArrayList<>();

        spacingY = 0;
        descY = 0;
        primFY = 0;
        culnGY = 0;
        yCreate = 0;

        frame = new JFrame();
        frame.setBounds(300, 90, 900, 900);
        frame.setResizable(false);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

```
container = new JPanel();
container.setBackground(Color.white);
frame.setTitle("New Meal Section");

jsp = new JScrollPane(container);

container.setPreferredSize(new Dimension(0, 1150));
container.setLayout(null);

nameLab = new JLabel("New Meal Section");
nameLab.setFont(new Font("Arial", Font.BOLD, 20));
nameLab.setSize(300, 50);
nameLab.setLocation(370, spacingY);
nameLab.setForeground(Color.decode("#9B9B9B"));
container.add(nameLab);

try {
        BufferedImage bar = ImageIO.read(new
                        File("systemImages/greenRec.png"));
        Image imgBar = bar.getScaledInstance(900, 50,
                                Image.SCALE_DEFAULT);
        JLabel barLabel = new JLabel(new ImageIcon(imgBar));
        barLabel.setSize(900, 50);
        barLabel.setLocation(-5, spacingY);
        container.add(barLabel);

} catch (IOException e1) {
        e1.printStackTrace();
}

// creates a dummy textfield that is not visible to the user so that the mouse
// focus is initally set on this dummy
// allows for mouse focus to be sent elsewhere so that correct default text is
// displayed on the title bar
dum = new JTextField();
dum.grabFocus();
dum.setFont(new Font("Arial", Font.PLAIN, 1));
dum.setSize(1, 1);
dum.setLocation(0, spacingY);
container.add(dum);

spacingY += 80;
name = new JTextField("Meal Section Name");
name.setHorizontalAlignment(JTextField.CENTER);
name.setFont(new Font("Arial", Font.PLAIN, 30));
name.setForeground(Color.decode("#9B9B9B"));
name.setSize(300, 50);
name.setLocation(290, spacingY);
container.add(name);

name.addFocusListener(new FocusListener() {
        // when focus is gained, and the field is either empty or set to default text,
        // clear the field
        public void focusGained(FocusEvent e) {
                if (name.getText().equals("") || name.getText().equals("Meal
                                                        Section Name")) {
                        name.setText("");
                }
        }

        public void focusLost(FocusEvent e) {

                // when focus is lost and there is no text, set to default text
                if (name.getText().equals("")) {
                        name.setText("Meal Section Name");
                }

                // if there is text, set the meal section name to be that text after the
                                                                              mouse
                // leaves the field
                msName = name.getText();
        }

});

spacingY += 70;

imgField = new Dragger(620, 280);
imgField.setSize(620, 280);
imgField.setLocation(120, spacingY + 30);

container.add(imgField);

spacingY += 330;
descY = spacingY;
descForm = new ListForm("Description", 2);
descForm.setSize(400, 150);
descForm.setLocation(20, descY);
```

```
        container.add(descForm);

        spacingY += 160;
        primFY = spacingY;
        primFForm = new ListForm("Primary Flavours", 2);
        primFForm.setSize(900, 150);
        primFForm.setLocation(20, primFY);
        container.add(primFForm);

        spacingY += 160;
        culnGY = spacingY;
        culnGForm = new ListForm("Culinary Goals", 2);
        culnGForm.setSize(900, 150);
        culnGForm.setLocation(20, culnGY);
        container.add(culnGForm);

        spacingY += 160;
        yCreate = spacingY;
        create = new JButton("Create");
        create.setFont(new Font("Arial", Font.PLAIN, 15));
        create.setSize(100, 40);
        create.setLocation(405, yCreate);
        create.setForeground(Color.white);
        create.setBackground(Color.decode("#55A630"));
        create.addActionListener(this);
        container.add(create);

        allForms.add(descForm);
        allForms.add(primFForm);
        allForms.add(culnGForm);

        frame.getContentPane().add(jsp);
        frame.setVisible(true);

}

// class for an existing meal section

public MSTemplate(String editableMSName) {

        editingMS = true;
        allForms = new ArrayList<ListForm>();

        this.editableMSName = editableMSName;
```

```
imgPath = "";

descList = new ArrayList<>();
primFList = new ArrayList<>();
culnGoalsList = new ArrayList<>();

spacingY = 0;
descY = 0;
primFY = 0;
culnGY = 0;
yCreate = 0;

frame = new JFrame();
frame.setBounds(300, 90, 900, 900);
frame.setResizable(false);
frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

container = new JPanel();
container.setBackground(Color.white);
frame.setTitle("Editing " + editableMSName + " Section");

jsp = new JScrollPane(container);

container.setPreferredSize(new Dimension(0, 1150));
container.setLayout(null);

nameLab = new JLabel("Editing " + editableMSName + " Section");
nameLab.setFont(new Font("Arial", Font.BOLD, 20));
nameLab.setSize(300, 50);
nameLab.setLocation(370, spacingY);
nameLab.setForeground(Color.decode("#9B9B9B"));
container.add(nameLab);

try {

        // add green bar to top of template
        BufferedImage bar = ImageIO.read(new
                        File("systemImages/greenRec.png"));
        Image imgBar = bar.getScaledInstance(900, 50,
                                Image.SCALE_DEFAULT);
        JLabel barLabel = new JLabel(new ImageIcon(imgBar));
        barLabel.setSize(900, 50);
        barLabel.setLocation(-5, spacingY);
        container.add(barLabel);
} catch (IOException e1) {
        e1.printStackTrace();
}

// dummy textfield to first grab mouse focus
dum = new JTextField();
dum.grabFocus();
dum.setFont(new Font("Arial", Font.PLAIN, 1));
dum.setSize(1, 1);
dum.setLocation(0, spacingY);
container.add(dum);

spacingY += 80;
name = new JTextField(editableMSName);
name.setEditable(false);
name.setHorizontalAlignment(JTextField.CENTER);
name.setFont(new Font("Arial", Font.PLAIN, 30));
name.setForeground(Color.decode("#9B9B9B"));
name.setSize(300, 50);
name.setLocation(290, spacingY);
container.add(name);

spacingY += 70;

imgField = new Dragger(620, 280);
imgField.setSize(620, 280);
imgField.setLocation(120, spacingY + 30);

container.add(imgField);

spacingY += 330;
descY = spacingY;
descForm = new ListForm("Description", 2);
descForm.setSize(400, 150);
descForm.setLocation(20, descY);
container.add(descForm);

spacingY += 160;
primFY = spacingY;
primFForm = new ListForm("Primary Flavours", 2);
primFForm.setSize(900, 150);
primFForm.setLocation(20, primFY);
container.add(primFForm);
```

```
		spacingY += 160;
		culnGY = spacingY;
		culnGForm = new ListForm("Culinary Goals", 2);
		culnGForm.setSize(900, 150);
		culnGForm.setLocation(20, culnGY);
		container.add(culnGForm);

		deleteMS = new JButton("Delete meal section");
		deleteMS.setFont(new Font("Arial", Font.PLAIN, 15));
		deleteMS.setSize(180, 40);
		deleteMS.setLocation(30, 50);
		deleteMS.setForeground(Color.DARK_GRAY);
		deleteMS.setBackground(Color.decode("#55A630"));
		deleteMS.addActionListener(this);
		container.add(deleteMS);

		spacingY += 160;
		yCreate = spacingY;
		create = new JButton("Save Changes");
		create.setFont(new Font("Arial", Font.PLAIN, 15));
		create.setSize(180, 40);
		create.setLocation(405, yCreate);
		create.setForeground(Color.white);
		create.setBackground(Color.decode("#55A630"));
		create.addActionListener(this);
		container.add(create);

		allForms.add(descForm);
		allForms.add(primFForm);
		allForms.add(culnGForm);

		frame.getContentPane().add(jsp);
		frame.setVisible(true);

	}

	@Override
	public void focusGained(FocusEvent e) {
		// TODO Auto-generated method stub

	}

	@Override
```

```
public void focusLost(FocusEvent e) {
	// TODO Auto-generated method stub

}

@Override
public void actionPerformed(ActionEvent e) {

	// when create button is pressed,
	if (e.getSource() == create) {

		// first check if name is valid,
		boolean validMS = true;

		// trim white space
		msName = msName.trim();
		// since the meal section name will be in the column name, it must
		                                                                exclude
		// certain characters
		for (int i = 0; i < msName.length(); i++) {
			if (!Character.isLetterOrDigit(msName.charAt(i)) &&
					msName.charAt(i) != ' ' || msName.contains("  ")) {
				JOptionPane.showMessageDialog(null,
						"Meal Section name is not in correct format.
						                                                    "
						+ "It must only contain letters or digts, single
						spaces, and no other special characters.");
				validMS = false;
				break;
			}
		}

		// check is MS name already exists only if new meal section is being
		                                                                made since
		// the name cannot be changed after creation

		if (!editingMS) {
			try {
				String url = "jdbc:sqlite:db.db";
				try {
					Class.forName("org.sqlite.JDBC");
				} catch (ClassNotFoundException e1) {
					e1.printStackTrace();
				}
```

```
        Connection con = DriverManager.getConnection(url);
        String query = "SELECT MealSection FROM '" +
                    CreateLoginForm.currUser + "'";
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
            if (rs.getString(1) != null) {
                if (rs.getString(1).contains(msName)) {

                JOptionPane.showMessageDialog(null,
                            "Meal Section name
already exists. Either delete the old one or give this one a unique name.");
                    validMS = false;
                    break;
                }

            }

        }

        rs.close();
        con.close();

    } catch (SQLException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
}

// if the name is valid and its a new meal section, add the info to the db
if (validMS && !editingMS) {
    msName.replaceAll(" ", "_");
    imgField.addImgPathToDB(msName, CreateLoginForm.currUser,
                            "MealSection");
    descForm.addInfoToDB();
    primFForm.addInfoToDB();
    cuInGForm.addInfoToDB();

    // clear the listform arraylists upon creation to maintain spacing
                                    and
    // organization for the sections using listform afterwards
    ListForm.totalMS = 0;
    allForms.clear();
```

```
			// create a new main kitchen screen
			MealSectionsUnderAccount ks = new
			MealSectionsUnderAccount();

			// dispose of this screen
			frame.dispose();

		}

		// if editing the meal section
		else if (editingMS) {

			try {
				String url = "jdbc:sqlite:db.db";
				try {
					Class.forName("org.sqlite.JDBC");
				} catch (ClassNotFoundException e1) {
					// TODO Auto-generated catch block
					e1.printStackTrace();
				}
				Connection con = DriverManager.getConnection(url);
				Statement stmt = con.createStatement();

				// delete those rows with the meal section name under
								MealSection column
				String query = "DELETE FROM '" +
			CreateLoginForm.currUser + "' WHERE MealSection LIKE '%"
						+ editableMSName + "%'";
				stmt.execute(query);

				String path = imgField.getImagePath().replaceAll((char) 92
							+ "" + (char) 92, (char) 47 + "");

				// add the updated info in its place
				String newMSPath = name.getText() + "|" + "imgPath:" +
										path;
				String newDescrip = name.getText() + "|" + "Description:"
						+
				descForm.formatMSList().replaceAll("Description:", "");
				String newPrimF = name.getText() + "|" + "Primary
									Flavours:"
						+
			primFForm.formatMSList().replaceAll("Primary Flavours:", "");
				String newCulnG = name.getText() + "|" + "Culinary Goals:"
```

```
                              +
                    culnGForm.formatMSList().replaceAll("Culinary Goals:", "");

                            query = "INSERT INTO '" + CreateLoginForm.currUser + "'
                            (MealSection) VALUES ('" + newMSPath + "')";
                            stmt.execute(query);

                            query = "INSERT INTO '" + CreateLoginForm.currUser + "'
                            (MealSection) VALUES ('" + newDescrip
                                          + "')";
                            stmt.execute(query);

                            query = "INSERT INTO '" + CreateLoginForm.currUser + "'
                            (MealSection) VALUES ('" + newPrimF + "')";
                            stmt.execute(query);

                            query = "INSERT INTO '" + CreateLoginForm.currUser + "'
                            (MealSection) VALUES ('" + newCulnG + "')";
                            stmt.execute(query);

                            this.msName = name.getText();

                            con.close();

                            // reset variables and clear arraylists
                            editingMS = false;
                            ListForm.totalMS = 0;
                            allForms.clear();

                            ClearBtn.ks.deleteKS();
                            MealSectionsUnderAccount.frame.dispose();
                            // go back to the screen which stores the recipes
                            ks = new RecipesUnderMS(msName);

                            frame.dispose();

                    } catch (SQLException e1) {
                            e1.printStackTrace();
                    }
            }
    }

    // if deleting mealsection,
    else if (e.getSource() == deleteMS) {
```

```
                // display caution message
                int result = JOptionPane.showConfirmDialog(this,
                            "Are you sure you want to delete this section? " + "All
                                                recipes will be lost forever.");

                if (result == JOptionPane.YES_OPTION) {

                    String url = "jdbc:sqlite:db.db";
                    try {
                        Class.forName("org.sqlite.JDBC");
                    } catch (ClassNotFoundException E) {
                        // TODO Auto-generated catch block
                        E.printStackTrace();
                    }

                    try {
                        Connection con = DriverManager.getConnection(url);
                        Statement stmt = con.createStatement();

                        // go to database and delete all rows under MealSection
                                                            with same name
                        String query = "DELETE FROM '" +
                    CreateLoginForm.currUser + "' WHERE MealSection LIKE '%"
                                + editableMSName + "%'";
                        stmt.execute(query);

                    } catch (SQLException e1) {
                        e1.printStackTrace();
                    }

                    // delete the frame and load the frame with all meal sections
                    ListForm.totalMS = 0;
                    frame.dispose();
                    MealSectionsUnderAccount ms = new
                    MealSectionsUnderAccount();

                }

            }

        }

        /**
```

```
 * code for populating a meal section template with data from an existing meal section
 */
public void populateMS() {
        editingMS = true;
        try {
                String url = "jdbc:sqlite:db.db";
                try {
                        Class.forName("org.sqlite.JDBC");
                } catch (ClassNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                Connection con = DriverManager.getConnection(url);
                String query = "SELECT MealSection FROM '" +
                CreateLoginForm.currUser + "'";
                Statement stmt = con.createStatement();
                ResultSet rs = stmt.executeQuery(query);

                // find rows under MealSection with the target name
                while (rs.next() && rs.getString(1) == null ||
                !rs.getString(1).contains(editableMSName)) {

                }
                String s = rs.getString(1);
                if (s.contains("imgPath:")) {
                        imgPath = s.substring(s.indexOf("imgPath:") + 8);
                }

                // break out of loops once the value is found
                while (rs.next() && rs.getString(1) != null) {

                        // storing data found under the same name
                        s = rs.getString(1);
                        if (s.contains(editableMSName)) {

                                if (s.contains("imgPath:")) {
                                        imgPath = s.substring(s.indexOf("imgPath:") + 8);
                                }

                                if (s.contains("Description:")) {

                                        // call to method which adds the comma seperated
                                        values retrieved from db into
                                        // an array
```

```
                    addValueToArr(descList, s, descForm.header);

                } else if (s.contains("Primary Flavours:")) {
                    addValueToArr(primFList, s, primFForm.header);

                } else if (s.contains("Culinary Goals:")) {
                        addValueToArr(culnGoalsList, s,
                            culnGForm.header);
                    break;
                }

            }
        }

        // for every element collected in the arraylist, display that in the MS
                                                                template
        // using the ListForm method: populateMSListForm()
        for (String str : descList) {
            descForm.populateMSListForm(str);
        }

        for (String str : primFList) {
            primFForm.populateMSListForm(str);
        }

        for (String str : culnGoalsList) {
            culnGForm.populateMSListForm(str);
        }

        imgField.loadImg(imgPath);

        con.close();

    } catch (SQLException e) {
        e.printStackTrace();
    }

}

/**
 * takes in arraylist which stores individual values, the comma seperated string with all
the values, and name of meal section
 * @param arr arraylist in which to insert the values
 * @param s meal section and recipe name concatenated string
```

```
         * @param header title of section
         */
        public void addValueToArr(ArrayList<String> arr, String s, String header) {

                // remove the MS name and | from the string
                s = s.replaceFirst(editableMSName + "|" + header, "");

                // example of a string stored in db:
                // MSName|Culinary Goals:fry, boil

                // msut also remove the :
                boolean passedColon = false;
                String value = "";
                for (int i = 0; i < s.length(); i++) {
                        if (s.charAt(i) == ':' && !passedColon) {
                                passedColon = true;
                        }

                        // once the colon has been passed, begin retrieving individual values
                                                                                seperated
                        // by commas
                        else if (passedColon) {
                                if (s.charAt(i) != ',') {
                                        value += s.charAt(i);
                                } else {
                                        if (!value.equals("empty")) {
                                                arr.add(value);
                                        }
                                        value = "";
                                }
                        }
                }
        }
}
```

MSThumbnail.java

```
/**
 * Creates panel which acts as thumbnail for each meal section
 * Uses the clear button overlay made in ClearBtn class
 */
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;
import javax.swing.*;

public class MSThumbnail extends JPanel {

	private JLabel nameLabel;
	private JLabel imgLabel;
	private BufferedImage bImg;
	private ImageIcon icon;
	private Image scaledImg;
	private Image newImg;
	private ClearBtn btn;
	private JLabel imgLabelBackground;
	private BufferedImage bImgBackground;
	private ImageIcon iconBackground;
	private Image scaledBackgroundImg;
	private Image newBackgroundImg;
	private String name;

	// input name of meal section and image path
	public MSThumbnail(String name, String imgPath) {

		this.name = name;
		setLayout(null);
		setPreferredSize(new Dimension(400, 250));
		setBackground(Color.WHITE);

		nameLabel = new JLabel(name);
		nameLabel.setFont(new Font("Arial", Font.BOLD, 30));
		nameLabel.setSize(230, 40);
		nameLabel.setLocation(60, 150);
		nameLabel.setHorizontalAlignment(SwingConstants.CENTER);
		nameLabel.setForeground(Color.white);
		add(nameLabel);

		try {

			// read image and resize
			bImg = ImageIO.read(new File(imgPath.substring(8)));
			icon = new ImageIcon(bImg);
			scaledImg = icon.getImage();
}

            newImg = scaledImg.getScaledInstance(290, 120,
                        java.awt.Image.SCALE_SMOOTH);
            icon = new ImageIcon(newImg);

            imgLabel = new JLabel();
            imgLabel.setIcon(icon);
            imgLabel.setSize(290, 120);
            imgLabel.setLocation(27, 20);

            add(imgLabel);
        } catch (Exception e) {
            // if theres an error, nothing will be displayed as the thumb nail image
        }
        try {

            // using inputted image of green rectangle as the background of the
                                                            thumbnail
            bImgBackground = ImageIO.read(new
            File("systemImages/msGreenRec.png"));
        } catch (Exception e) {
            e.printStackTrace();
        }

        iconBackground = new ImageIcon(bImgBackground);
        scaledBackgroundImg = iconBackground.getImage();

        newBackgroundImg = scaledBackgroundImg.getScaledInstance(350, 200,
        java.awt.Image.SCALE_SMOOTH);
        iconBackground = new ImageIcon(newBackgroundImg);

        imgLabelBackground = new JLabel();
        imgLabelBackground.setIcon(iconBackground);
        imgLabelBackground.setSize(350, 200);
        imgLabelBackground.setLocation(0, 0);

        add(imgLabelBackground);

        // add clear button overlay - > keep size and location same as iconBackground
        btn = new ClearBtn(name, "no raw", 1);
        btn.setSize(350, 200);
        btn.setLocation(0, 0);
        add(btn);
    }
```

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

NewAccount.java

```
/**
 * Class for making new account in the database
 * Stores username, passcode, userpasskey (concatentated string of username and passcode
to create consistently unique reference to a user)
 *
 */
import javax.swing.JOptionPane;
import java.sql.Statement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class NewAccount {

	public NewAccount(String owner, String passcode) {
		try {
			// Connect to database
			String url = "jdbc:sqlite:db.db";
			Class.forName("org.sqlite.JDBC");
			Connection con = DriverManager.getConnection(url);

			Statement stmt = con.createStatement();
			String s = "INSERT INTO USERS VALUES ('" + passcode + "','" + owner
								+ "','" + passcode + owner + "')";
			stmt.execute(s);
			s = "CREATE TABLE IF NOT EXISTS '" + passcode + owner + "' (\n"

					+ "	MealSection text DEFAULT NULL\n" + ");";

			stmt.execute(s);
			JOptionPane.showMessageDialog(null, "Success! Account created.");
			con.close();
		}
		// error message for showing if username already exists in database
		catch (SQLException e1) {
			JOptionPane.showMessageDialog(null, "Username already exists");
		} catch (ClassNotFoundException e) {
				e.printStackTrace();
			}
		}
}
```

OtherOptions.java

```
/**
 * creates panel which houses star rating bar, difficulty level bar, prep and cook time, and yield
 * includes method for populating Recipe template with existing option panel selections
 */
import java.util.ArrayList;
import javax.swing.*;

public class OtherOptions extends JPanel {

	private StarRate rateBar;
	private DifficultyRate difBar;
	private JPanel level;
	private TimePanel prep;
	private TimePanel cook;
	private TimePanel yield;
	public static ArrayList optArr;

	private Dragger imgField;
	private JPanel total;
	int startYcord;

	public OtherOptions() {

		optArr = new ArrayList<>();
		setLayout(null);
		startYcord = 20;
		rateBar = new StarRate();
		rateBar.setLocation(20, startYcord);
		add(rateBar);

		optArr.add(rateBar);

		imgField = new Dragger();
		imgField.setSize(230, 230);
		imgField.setLocation(530, startYcord + 30);
		add(imgField);
```

```
        optArr.add(imgField);

        startYcord += 60;
        difBar = new DifficultyRate();
        difBar.setLocation(-15, startYcord);
        add(difBar);

        optArr.add(difBar);

        prep = new TimePanel("Prep Time: ");
        prep.setSize(400, 60);
        startYcord += 55;
        prep.setLocation(10, startYcord);
        cook = new TimePanel("Cook Time: ");

        startYcord += 60;

        cook.setSize(400, 60);
        cook.setLocation(10, startYcord);

        startYcord += 60;
        yield = new TimePanel("Yield: ", "yieldTF");
        yield.setSize(400, 60);
        yield.setLocation(-100, startYcord);

        add(prep);
        add(cook);
        add(yield);

        optArr.add(prep);
        optArr.add(cook);
        optArr.add(yield);

    }

    /**
     * method for populating a recipe template with existing recipe option selections
     * @param starRate the string for the star rating panel
     * @param imgPath path of image
     * @param difRate rating of difficulty
     * @param pTimeHours hours time prep
     * @param pTimeMins minutes time prep
     * @param cTimeHours hours time cook
     * @param cTimeMins minutes time cook
```

```
         * @param yield quantity to be produced from recipe
         */
        public void populateRecipeListForm(String starRate, String imgPath, String difRate,
String pTimeHours,
                        String pTimeMins, String cTimeHours, String cTimeMins, String yield) {
                StarBtn.selectedStar = Integer.parseInt(starRate);
                rateBar.repaintStarBar();

                imgField.loadImg(imgPath);

                LevelBtn.selectedBtn = Integer.parseInt(difRate);
                difBar.repaintSelectedDiff();

                prep.setTime(pTimeHours, pTimeMins);
                cook.setTime(cTimeHours, cTimeMins);
                this.yield.setYield(yield);
        }

        public String getImgPath() {
                return imgField.getImagePath();
        }
}
```

RecipeDataRetrieval.java

```
/**
 * create the jpanel which houses all the recipes made
 * fetches info from the db, loads onto a panel
 */

import java.awt.Color;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.*;

import javax.swing.*;

public class RecipeDataRetrieval extends JPanel {
```

```
ArrayList<RecipeThumbnail> recipArr;
public static int count = 0;
public int yDim;

// input name of meal section
public RecipeDataRetrieval(String name) {

        // character __ is used in database so we must replace them with spaces
        name = name.replaceAll(" ", "_");
        // add name to query so that only recipes of that meal section are called

        yDim = 0;
        count = 0;
        setBackground(Color.white);
        setLayout(new FlowLayout(FlowLayout.CENTER));
        recipArr = new ArrayList<>();

        ArrayList<String> list = new ArrayList<>();

        try {
                String url = "jdbc:sqlite:db.db";
                try {
                        Class.forName("org.sqlite.JDBC");
                } catch (ClassNotFoundException e1) {
                        e1.printStackTrace();
                }
                Connection con = DriverManager.getConnection(url);

                // pragma_table_info selects only column names, meaning it just goes
                                                                        along the
                // top row of column names
                // which consists of MealSection column, followed by column names in the
                                                                            form of
                // [MealSectionName]__[RecipeName]
                String query = "select c.name from pragma_table_info('" +
                                                CreateLoginForm.currUser + "') c";

                // statement for previous query
                Statement stmt = con.createStatement();
                ResultSet rs = stmt.executeQuery(query);

                // statement for executing query which collects recipe info
                Statement stmt2 = con.createStatement();
                ResultSet rs2;
```

```
while (rs.next()) {
        // skip meal section column and only choose columns part of the
                                                requested meal
        // section
        if (!rs.getString(1).equals("MealSection") &&
                        rs.getString(1).startsWith(name)) {
                // Keep track of number of recipes for spacing
                count++;

                ArrayList<String> tempList = new ArrayList<>();
                String recipName = rs.getString(1).replace(' ', '_');
                list.add(recipName);

                // use that column name and select all the rows directly
                                                        below it
                String queryInfo = "SELECT " + recipName + " FROM '" +
                                CreateLoginForm.currUser + "'";
                rs2 = stmt2.executeQuery(queryInfo);

                // skip null values
                while (rs2.next() && rs2.getString(1) == null) {
                }
                // start adding the values
                while (rs2.next() && rs2.getString(1) != null) {
                        list.add(rs2.getString(1));
                }

                // store the elements of the recipe in tempList
                for (int i = 0; i < list.size(); i++) {
                        if (i == 0 || i == 5 || i == 6 || i == 7 || i == 8 | i == 9) {
                                tempList.add(list.get(i));

                        }
                }

                // assign values from tempList
                String recipTitle = "";
                String recipPath = "";
                String recipRate = "";
                try {
                        recipTitle = tempList.get(0);
                        recipPath = tempList.get(1);
                        recipRate = "Unrated";
        // retrieving numeric rating from the string
        String num = "" + tempList.get(2).charAt(11);
        if (Integer.parseInt(num) == 0) {
                recipRate = "Easy";
        } else if (Integer.parseInt(num) == 1) {
                recipRate = "Medium";
        } else {
                recipRate = "Hard";
        }
} catch (Exception e) {

}

// adding up prep and cook time
// 2d arrays, 1 column dedicated to prep time, the other
                                                    cook time
int[] hourList = new int[2];
int[] minList = new int[2];

boolean afterCol = false;

int count = 0;

try {
        for (int m = 3; m <= 4; m++) {
                // skip until after the :
                for (int i = 0; i < tempList.get(m).length();
                                                                i++) {
                        if (tempList.get(m).charAt(i) == ':') {
                                afterCol = true;
                        } else if (afterCol) {
                                int k = i;
                                String s = "";

                                // in the db, times are stored
        as 4,3 for example, where the first digit in this
                                // case 4 is hours and the
                                digit after the, is the minute
                                while
                        (tempList.get(m).charAt(k) != ',') {
                                        // concatenating the
                                                hour string
                                        s +=
                                tempList.get(m).charAt(k);
                                        k++;
                                }

                                // adding the integer value to
                                            the list
                                hourList[count] =
                                        Integer.parseInt(s);
                                s = "";

                                // skip over ,
                                k++;

                                // concatenating the minute
                                            string
                                while
                        (tempList.get(m).charAt(k) != ',') {
                                        s +=
                                tempList.get(m).charAt(k);
                                        k++;
                                }

                                // adding integer value to list
                                minList[count] =
                                Integer.parseInt(s);
                                break;
                        }
                }
                count++;
                afterCol = false;
        }
        int totalMin = 0;
        int totalHour = 0;

        // summing up times
        for (int i = 0; i < 2; i++) {
                totalMin += minList[i];
                totalHour += hourList[i];
        }

        // formating times to appropriate values
        if (totalMin > 59) {
                totalHour += totalMin / 60;
                totalMin %= 60;
```

```
                                        }

                                        String recipTime;

                                        recipTime = totalHour + " h " + totalMin + " m";

                                        String recipYield = tempList.get(5).substring(6);
                                        String temp = "";

                                        // retrieving yield value
                                        for (int i = 0; i < recipYield.length(); i++) {
                                                if (recipYield.charAt(i) != ',') {
                                                        temp += recipYield.charAt(i);
                                                } else {
                                                        break;
                                                }
                                        }
                                        recipYield = temp;
                                        try {

                                                // retrieving recipe name from db
                                                temp = "";

                                                boolean afterMSName = false;
                                                for (int i = 0; i < recipTitle.length(); i++) {
                                                        if (i < recipTitle.length() - 1 &&
                                                                recipTitle.charAt(i) == '_'
                                                                        && recipTitle.charAt(i
                                                        + 1) == '_' && !afterMSName) {
                                                                afterMSName = true;
                                                                i++;
                                                        }

                                                        // only start concatenating recipe
                                                        name if passed meal section name
                                                        else if (afterMSName) {
                                                                temp += recipTitle.charAt(i);
                                                        }

                                                }
                                                temp = temp.replaceAll("_", " ");

                                                RecipeThumbnail recip = new
                        RecipeThumbnail(temp, recipPath, recipTime, recipRate, recipYield,
                                                            recipTitle);
                                                    recipArr.add(recip);

                                            } catch (Exception e) {
                                                    e.printStackTrace();
                                            }

                                            list.clear();
                                            rs2.close();
                                    } catch (Exception e) {
                                            e.printStackTrace();
                                    }

                            }

                    }

                    con.close();
            } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }

            // formatting the recipe thumbnails

            if (count >= 3) {

                    // if there are an odd amount of panels, divide total by 3 and multiply by
                                                                                        200
                    // pixels for the height
                    if (count % 3 == 0) {
                            yDim = (count / 3) * 200;
                    } else {

                            // divide by 3 and add extra panel to house remaining thumbnails
                            yDim = (count / 3 + 1) * 200;
                    }
            }

            // less than 3, just use 200 pixels for height of container
            else {
                    yDim = 200;
            }
```

```
		setPreferredSize(new Dimension(900, yDim));

		RecipeThumbnail[] arr = new RecipeThumbnail[recipArr.size()];
		// sort alphabetically

		for (int i = 0; i<recipArr.size(); i++) {
			arr[i] = recipArr.get(i);
		}

		if (RecipesUnderMS.sortSelected) {
			for (int i = 1; i<recipArr.size(); i++) {
				RecipeThumbnail var = arr[i];
				int j = i-1;
				while (j>=0 &&
				var.getName().compareToIgnoreCase(arr[j].getName())<0) {
					arr[j+1] = arr[j];
					j--;

				}
				arr[j+1] = var;

			}
			RecipesUnderMS.sortSelected = false;
		}

		// add each thumbnail to container
		for (int i = 0; i<recipArr.size(); i++) {
			add(arr[i]);
		}
	}
}
```

RecipesUnderMS.java

```
/**
 * Creates the window where all recipes under a meal section are stored
 */
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

class RecipesUnderMS extends JFrame implements ActionListener {
```

```
private JPanel c;
public static JFrame frame;
private JLabel sectName;
private JScrollPane jsp;
private JButton makeRecipe;
private JButton backToKitchen;
private JButton seeMealSection;
private JButton sort;
private RecipeDataRetrieval ms;
public static String msName;
public static boolean sortSelected;

public RecipesUnderMS(String sectText) {

        msName = sectText;
        frame = new JFrame();
        frame.setBounds(300, 90, 900, 650);
        frame.setResizable(false);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("Kitchen Space");

        // making instance of meal section by supplying meal section name
        ms = new RecipeDataRetrieval(sectText);
        c = new JPanel();
        c.setBackground(Color.white);

        jsp = new JScrollPane(c);
        c.setPreferredSize(new Dimension(0, 300 + ms.yDim));
        c.setLayout(new FlowLayout(FlowLayout.RIGHT));
        c.setBackground(Color.white);

        backToKitchen = new JButton("Back to kitchen");
        backToKitchen.setFont(new Font("Arial", Font.PLAIN, 30));
        backToKitchen.setSize(230, 60);
        backToKitchen.setBackground(Color.WHITE);
        backToKitchen.setForeground(Color.decode("#55A630"));
        c.add(backToKitchen);

        seeMealSection = new JButton("See Meal Section");
        seeMealSection.setFont(new Font("Arial", Font.PLAIN, 30));
        seeMealSection.setSize(230, 60);
        seeMealSection.setBackground(Color.WHITE);
        seeMealSection.setForeground(Color.decode("#55A630"));
}


        seeMealSection.setFocusable(false);

        c.add(seeMealSection);

        sort = new JButton("Sort alphabetically");
        sort.setFont(new Font("Arial", Font.PLAIN, 30));
        sort.setSize(230, 60);
        sort.setBackground(Color.WHITE);
   sort.setForeground(Color.decode("#55A630"));
        sort.setFocusable(false);

        c.add(sort);

        sectName = new JLabel(sectText);
        sectName.setFont(new Font("Arial", Font.BOLD, 80));
        sectName.setForeground(Color.white);
        sectName.setBackground(Color.decode("#BAF2BB"));
        sectName.setPreferredSize(new Dimension(sectText.length() * 60, 130));
        sectName.setHorizontalAlignment(SwingConstants.LEFT);
        sectName.setOpaque(true);
        c.add(sectName);

        c.add(ms);

        makeRecipe = new JButton("Make a recipe");
        makeRecipe.setFont(new Font("Arial", Font.PLAIN, 30));
        makeRecipe.setSize(230, 60);
        makeRecipe.setBackground(Color.WHITE);
        makeRecipe.setForeground(Color.decode("#55A630"));
        c.add(makeRecipe);

        makeRecipe.addActionListener(this);
        seeMealSection.addActionListener(this);
        backToKitchen.addActionListener(this);
        sort.addActionListener(this);

        frame.getContentPane().add(jsp);
        frame.setVisible(true);

}

public void deleteKS() {
        frame.dispose();
}
```

```
	public void actionPerformed(ActionEvent e) {
		// if a recipe is made, dispose of the current frame and create frame of recipe
		// template
		if (e.getSource() == makeRecipe) {
			RecipeTemplate recip = new RecipeTemplate(msName);
			frame.dispose();
		}

		if (e.getSource() == sort) {
			sortSelected = true;
			frame.dispose();
			ClearBtn.ks= new RecipesUnderMS(msName);

		}

		// If viewing meal section, dispose of current frame and create Meal Section
		// template populated with corresponding data
		else if (e.getSource() == seeMealSection) {
			frame.dispose();
			MSTemplate ms = new MSTemplate(msName);
			ms.populateMS();
			ms.repaint();

		}
		// Going back to KitchenSpace, dispose of current frame and reload Main Kitchen
		// screen
		else if (e.getSource() == backToKitchen) {
			MealSectionsUnderAccount ms = new MealSectionsUnderAccount();
			frame.dispose();
		}
	}

}
```

RecipeTemplate.java

```
/**
 * template for the recipe fill out form
 * includes template for existing recipe and corresponding method for populating fields
 */
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
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 javax.imageio.ImageIO;
import javax.swing.*;

public class RecipeTemplate extends JFrame implements ActionListener, FocusListener {

	public static boolean editingRecip;
	private JFrame frame;
	public static JPanel container;
	private JLabel nameLab;
	private final JTextField name;
	private final JTextField dum;
	public static int ingSpacing;
	public static JButton create;

	public static JButton deleteRecip;
	public static ListForm ingForm;
	public static ListForm instructForm;
	public static ListForm subForm;
	public static ListForm noteForm;
	public static ArrayList<ListForm> allForms;
	public static String recipName = "";
	public static int ingY;
	public static int insY;
	public static int noteY;
	public static int yCreate;
	private OtherOptions opts;
	public static int spacingY;
	private String nameStr;
	private String msName;
	public static String colName;
	private String raw;
	public static String newColName;
	private int type;
```

```
// template for new recipe, type = 1
public RecipeTemplate(String msName) {

        editingRecip = false;
        type = 1;
        allForms = new ArrayList<>();

        this.msName = msName;
        spacingY = 0;
        frame = new JFrame();
        frame.setBounds(300, 90, 900, 650);
        frame.setResizable(false);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        container = new JPanel();
        container.setBackground(Color.white);
        frame.setTitle("New Recipe");

        JScrollPane jsp = new JScrollPane(container);

        container.setPreferredSize(new Dimension(0, 650));
        container.setLayout(null);

        deleteRecip = new JButton("Delete Recipe");
        deleteRecip.setFont(new Font("Arial", Font.PLAIN, 15));
        deleteRecip.setSize(180, 40);
        deleteRecip.setLocation(30, 50);
        deleteRecip.setForeground(Color.DARK_GRAY);
        deleteRecip.setBackground(Color.decode("#55A630"));
        deleteRecip.addActionListener(this);
        container.add(deleteRecip);

        nameLab = new JLabel("New " + msName + " Recipe");
        nameLab.setFont(new Font("Arial", Font.BOLD, 20));
        nameLab.setSize(300, 50);
        nameLab.setLocation(370, spacingY);
        nameLab.setForeground(Color.decode("#9B9B9B"));
        container.add(nameLab);

        try {
                BufferedImage bar = ImageIO.read(new
                                                        File("systemImages/greenRec.png"))
                                                ;
```

```
			Image imgBar = bar.getScaledInstance(900, 50,
							Image.SCALE_DEFAULT);
			JLabel barLabel = new JLabel(new ImageIcon(imgBar));
			barLabel.setSize(900, 50);
			barLabel.setLocation(-5, spacingY);
			container.add(barLabel);

		} catch (IOException e1) {
			e1.printStackTrace();
		}

		// create dummy text field to grab mouse focus so that default text is displayed
		// as the title
		dum = new JTextField();
		dum.grabFocus();
		dum.setFont(new Font("Arial", Font.PLAIN, 1));
		dum.setSize(1, 1);
		dum.setLocation(0, spacingY);
		container.add(dum);

		spacingY += 80;
		name = new JTextField("Recipe Name");
		name.setHorizontalAlignment(JTextField.CENTER);
		name.setFont(new Font("Arial", Font.PLAIN, 30));
		name.setForeground(Color.decode("#9B9B9B"));
		name.setSize(300, 50);
		name.setLocation(290, spacingY);
		container.add(name);

		name.addFocusListener(new FocusListener() {
			public void focusGained(FocusEvent e) {

				// if focus is gained and text field is empty or has default text, clear
it
				if (name.getText().equals("") || name.getText().equals("Recipe
Name")) {
					name.setText("");
				}
			}

			public void focusLost(FocusEvent e) {

				// if focus is lost and field is empty, set text to default text
				if (name.getText().equals("")) {
```

```
                        name.setText("Recipe Name");
                    }

                    // else set the text to whatever was entered
                    recipName = name.getText();
            }

    });

    spacingY += 50;
    opts = new OtherOptions();
    opts.setSize(900, 310);
    opts.setBackground(Color.white);
    opts.setLocation(0, spacingY);
    container.add(opts);

    spacingY += 330;
    ingY = spacingY;
    ingForm = new ListForm("Ingrediants", 1);
    ingForm.setSize(400, 150);
    ingForm.setLocation(20, ingY);
    container.add(ingForm);

    subForm = new ListForm("Substitutions", 1);
    subForm.setSize(400, 150);
    subForm.setLocation(450, ingY);
    container.add(subForm);

    spacingY += 160;
    insY = spacingY;
    instructForm = new ListForm("Instructions", 1);
    instructForm.setSize(900, 150);
    instructForm.setLocation(20, insY);
    container.add(instructForm);

    spacingY += 160;
    noteY = spacingY;
    noteForm = new ListForm("Notes", 1);
    noteForm.setSize(900, 150);
    noteForm.setLocation(20, noteY);
    container.add(noteForm);

    spacingY += 160;
    yCreate = spacingY;
```

```
        create = new JButton("Create");
        create.setFont(new Font("Arial", Font.PLAIN, 15));
        create.setSize(100, 40);
        create.setLocation(405, yCreate);
        create.setForeground(Color.white);
        create.setBackground(Color.decode("#55A630"));
        create.addActionListener(this);
        container.add(create);

        allForms.add(ingForm);
        allForms.add(subForm);
        allForms.add(instructForm);
        allForms.add(noteForm);

        frame.getContentPane().add(jsp);
        frame.setVisible(true);

    }

    public RecipeTemplate(String nameStr, String raw) {

        editingRecip = true;
        type = 2;
        allForms = new ArrayList<>();

        this.nameStr = nameStr;

        // raw is a concatenation of the meal section name and recipe name
        // it is the exact title of a recipe column in the db
        this.raw = raw;

        spacingY = 0;
        frame = new JFrame();
        frame.setBounds(300, 90, 900, 650);
        frame.setResizable(false);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        container = new JPanel();
        container.setBackground(Color.white);
        frame.setTitle("Editing Recipe");

        JScrollPane jsp = new JScrollPane(container);

        container.setPreferredSize(new Dimension(0, 650));
```

```
container.setLayout(null);

deleteRecip = new JButton("Delete Recipe");
deleteRecip.setFont(new Font("Arial", Font.PLAIN, 15));
deleteRecip.setSize(180, 40);
deleteRecip.setLocation(30, 50);
deleteRecip.setForeground(Color.DARK_GRAY);
deleteRecip.setBackground(Color.decode("#55A630"));
deleteRecip.addActionListener(this);
container.add(deleteRecip);

nameLab = new JLabel("Editing " + nameStr + " Recipe");
nameLab.setFont(new Font("Arial", Font.BOLD, 20));
nameLab.setSize(300, 50);
nameLab.setLocation(370, spacingY);
nameLab.setForeground(Color.decode("#9B9B9B"));
container.add(nameLab);

try {
        BufferedImage bar = ImageIO.read(new
                                        File("systemImages/greenRec.png"));
        Image imgBar = bar.getScaledInstance(900, 50,
                                                        Image.SCALE_DEFAULT);
        JLabel barLabel = new JLabel(new ImageIcon(imgBar));
        barLabel.setSize(900, 50);
        barLabel.setLocation(-5, spacingY);
        container.add(barLabel);

} catch (IOException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
}

dum = new JTextField();
dum.grabFocus();
dum.setFont(new Font("Arial", Font.PLAIN, 1));
dum.setSize(1, 1);
dum.setLocation(0, spacingY);
container.add(dum);

spacingY += 80;
name = new JTextField(nameStr);
name.setHorizontalAlignment(JTextField.CENTER);
name.setFont(new Font("Arial", Font.PLAIN, 30));
name.setForeground(Color.decode("#9B9B9B"));
name.setSize(300, 50);
name.setLocation(290, spacingY);
container.add(name);

name.addFocusListener(new FocusListener() {
        public void focusGained(FocusEvent e) {
                if (name.getText().equals("") || name.getText().equals(nameStr)) {
                        name.setText("");
                }
        }

        public void focusLost(FocusEvent e) {
                if (name.getText().equals("")) {
                        name.setText(nameStr);
                }
                recipName = name.getText();
        }

});

spacingY += 50;
opts = new OtherOptions();
opts.setSize(900, 310);
opts.setBackground(Color.white);
opts.setLocation(0, spacingY);
container.add(opts);

spacingY += 330;
ingY = spacingY;
ingForm = new ListForm("Ingrediants", 1);
ingForm.setSize(400, 150);
ingForm.setLocation(20, ingY);
container.add(ingForm);

subForm = new ListForm("Substitutions", 1);
subForm.setSize(400, 150);
subForm.setLocation(450, ingY);
container.add(subForm);

spacingY += 160;
insY = spacingY;
instructForm = new ListForm("Instructions", 1);
instructForm.setSize(900, 150);
```

```
        instructForm.setLocation(20, insY);
        container.add(instructForm);

        spacingY += 160;
        noteY = spacingY;
        noteForm = new ListForm("Notes", 1);
        noteForm.setSize(900, 150);
        noteForm.setLocation(20, noteY);
        container.add(noteForm);

        spacingY += 160;
        yCreate = spacingY;
        create = new JButton("Save Changes");
        create.setFont(new Font("Arial", Font.PLAIN, 15));
        create.setSize(160, 40);
        create.setLocation(405, yCreate);
        create.setForeground(Color.white);
        create.setBackground(Color.decode("#55A630"));
        create.addActionListener(this);
        container.add(create);

        allForms.add(ingForm);
        allForms.add(subForm);
        allForms.add(instructForm);
        allForms.add(noteForm);

        frame.getContentPane().add(jsp);
        frame.setVisible(true);

    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == create) {
            boolean numCheck = false;

            // check if only numbers are entered into the prep and cook time fields
            try {
                int n = 0;

                // if one of these returns an error, then catch it by showing a
                                                        JOptionPane
                // dialog message
                n = Integer.parseInt(((TimePanel)
                OtherOptions.optArr.get(3)).getHours());
				n = Integer.parseInt(((TimePanel)
				OtherOptions.optArr.get(3)).getMins());
				n = Integer.parseInt(((TimePanel)
				OtherOptions.optArr.get(4)).getHours());
				n = Integer.parseInt(((TimePanel)
				OtherOptions.optArr.get(4)).getMins());

				// if all assignments to n are integer and no errors are caused, then
											all the
				// values are integers
				numCheck = true;
			} catch (Exception e1) {
				JOptionPane.showMessageDialog(null, "Enter numbers for Cook
										and Prep time.");
			}

			boolean validRecipName = true;

			// remove leading and trailing white space
			recipName = recipName.trim();
			// since the recipe name will be in the column name, it must exclude
											certain
			// characters
			for (int i = 0; i < recipName.length(); i++) {
				if (!Character.isLetterOrDigit(recipName.charAt(i)) &&
								recipName.charAt(i) != ' '
						|| recipName.contains("  ")) {
					JOptionPane.showMessageDialog(null,
							"Recipe name is not in correct format. It
must only contain letters or digts, single spaces, and no other special characters.");
					validRecipName = false;
					break;
				}
			}

			// if all numeric required values are correct and the name is valid, then
			// proceed by sending data to db
			if (numCheck && validRecipName) {
				try {

					String url = "jdbc:sqlite:db.db";
					try {
						Class.forName("org.sqlite.JDBC");
					} catch (ClassNotFoundException e1) {
```

```
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
            Connection con = DriverManager.getConnection(url);
            Statement stmt = con.createStatement();
            // __ distinguishes meal section name from recipe name

            if (type == 1) {
                colName = msName.replaceAll(" ", "_") + "__" +
            RecipeTemplate.recipName.replaceAll(" ", "_");

                String query = "ALTER TABLE '" +
        CreateLoginForm.currUser + "' ADD '" + colName + "' text NULL";
                stmt.execute(query);

                ingForm.addInfoToDB();
                subForm.addInfoToDB();
                instructForm.addInfoToDB();
                noteForm.addInfoToDB();
                ListForm.addOptsInfo();

                // when a new recipe is created after one has
                                already been created in the same
                // session,
                // static counter totalRecip is to be reset and the
                                        arrayList is cleared so at
                // to preserve the spacing
                // and reformatting system
                ListForm.totalRecip = 0;

                ClearBtn.ks.deleteKS();

                ClearBtn.ks = new RecipesUnderMS(msName);
                frame.dispose();

            } else if (type == 2) {
                // when updating a recipe, you just delete the old
                                        one and add the new data
                // under the same column name

                String query = "ALTER TABLE '" +
        CreateLoginForm.currUser + "' DROP COLUMN '" + raw + "'";
                stmt.execute(query);
```

```
}


                                String msName = "";
                                for (int i = 0; i < raw.length(); i++) {
                                        if (!(raw.charAt(i) == '_' && raw.charAt(i + 1)
                                                                            == '_')) {
                                                msName += raw.charAt(i);
                                        } else {
                                                break;
                                        }
                                }
                                newColName = msName + "__" +
                                        name.getText().replaceAll(" ", "_");
                                query = "ALTER TABLE `" +
                CreateLoginForm.currUser + "` ADD `" + newColName + "` text NULL";

                                stmt.execute(query);

                                // add the rest of the info to the column
                                ingForm.addInfoToDB();
                                subForm.addInfoToDB();
                                instructForm.addInfoToDB();
                                noteForm.addInfoToDB();
                                ListForm.addOptsInfo();

                                editingRecip = false;

                                ListForm.totalRecip = 0;

                                ClearBtn.ks.deleteKS();
                                ClearBtn.ks = new RecipesUnderMS(msName);

                                frame.dispose();
                        }

                        con.close();
                } catch (SQLException e1) {

                        // if recipe with same name already exists under that meal
                                                                    section, show error
                        JOptionPane.showMessageDialog(null, "Recipe already
                                                                            defined.");
                        e1.printStackTrace();
                }
        }
}
```

```
// if deleting recipe
if (e.getSource() == deleteRecip) {
        int result = JOptionPane.showConfirmDialog(this,
                        "Are you sure you want to delete this recipe? " + "All data
                                                        will be lost forever.");

        if (result == JOptionPane.YES_OPTION) {

                // if deleting a recipe that is currently being written,
                if (type == 1) {
                        ListForm.totalRecip = 0;
                        RecipesUnderMS ks = new RecipesUnderMS(msName);
                        frame.dispose();
                }

                // deleting a recipe that had been written previously and has now
                                                                been loaded
                else {
                        String url = "jdbc:sqlite:db.db";
                        try {
                                Class.forName("org.sqlite.JDBC");
                        } catch (ClassNotFoundException E) {
                                E.printStackTrace();
                        }

                        try {
                                Connection con =
                                                DriverManager.getConnection(url);
                                Statement stmt = con.createStatement();

                                // drop the column with that meal section and recipe
                                                name which are concatenated
                                // in a single string called raw
                                String query = "ALTER TABLE '" +
                CreateLoginForm.currUser + "' DROP COLUMN '" + raw + "'";
                                stmt.execute(query);

                        } catch (SQLException e1) {
                                e1.printStackTrace();
                        }

                        // getting the meal section name from raw
                        String s = "";
```

```
                                        int i = 0;
                                        while (raw.charAt(i) != '_') {
                                                s += raw.charAt(i);
                                                i++;
                                        }
                                        // ListForm.totalRecip = 0 resets the page so that
                                                                                formatting of elements is
                                        // normal
                                        ListForm.totalRecip = 0;
                                        ClearBtn.ks.deleteKS();
                                        ClearBtn.ks = new RecipesUnderMS(s);
                                        frame.dispose();

                                }
                        }

                }

        }

        @Override
        public void focusGained(FocusEvent e) {
                // TODO Auto-generated method stub

        }

        @Override
        public void focusLost(FocusEvent e) {
                // TODO Auto-generated method stub

        }

        /**
         *  method for populating a recipe template with existing data from db
         */
        public void populateRecipe() {

                try {
                        String url = "jdbc:sqlite:db.db";
                        try {
                                Class.forName("org.sqlite.JDBC");
                        } catch (ClassNotFoundException e) {
                                e.printStackTrace();
                        }
                        Connection con = DriverManager.getConnection(url);

                        // using raw select that recipe column from db
                        String query = "SELECT " + raw + " FROM '" + CreateLoginForm.currUser
        + "'";

                        Statement stmt = con.createStatement();
                        ResultSet rs = stmt.executeQuery(query);

                        ArrayList<String> ingList = new ArrayList<>();
                        ArrayList<String> subList = new ArrayList<>();
                        ArrayList<String> insList = new ArrayList<>();
                        ArrayList<String> notesList = new ArrayList<>();
                        ArrayList<String> otherOpts = new ArrayList<>();

                        // start retrieving values and adding to appropriate arraylists
                        while (rs.next() && rs.getString(1) == null) {
                        }

                        String s = rs.getString(1);

                        if (s.contains("Ingrediants:")) {

                            // call to a method which adds the elements from a comma
                                                        seperated string to an
                            // arraylist
                            addValueToArr(ingList, s);
                        }

                        while (rs.next() && rs.getString(1) != null) {
                            s = rs.getString(1);

                            if (s.contains("Substitutions:")) {
                                addValueToArr(subList, s);
                            }

                            else if (s.contains("Instructions:")) {
                                addValueToArr(insList, s);
                            }

                            else if (s.contains("Notes:")) {
                                addValueToArr(notesList, s);
                            }
```

```
				else {
					addValueToArr(otherOpts, s);
				}

			}
			con.close();

			// populating elements from arraylist into the list form text field
			for (String str : ingList) {
				ingForm.populateRecipeListForm(str);
			}

			for (String str : subList) {
				subForm.populateRecipeListForm(str);
			}
			for (String str : insList) {
				instructForm.populateRecipeListForm(str);
			}
			for (String str : notesList) {
				noteForm.populateRecipeListForm(str);
			}

			opts.populateRecipeListForm(otherOpts.get(0), otherOpts.get(1),
			otherOpts.get(2), otherOpts.get(3),
			otherOpts.get(4), otherOpts.get(5), otherOpts.get(6), otherOpts.get(7));

		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	/**
	 * method which takes an arraylist and comma seperated string as input and adds
	 * each element from the string to the arraylist
	 * @param arr array which will store the seperate sections of the strings
	 * @param s the string which will be seperated
	 */
	public void addValueToArr(ArrayList<String> arr, String s) {
		boolean passedColon = false;
		String value = "";

		// must skip over : in the string to access values
		for (int i = 0; i < s.length(); i++) {
			if (s.charAt(i) == ':' && !passedColon) {
```

```
                        passedColon = true;
                    } else if (passedColon) {

                        // start adding values seperated by comma
                        if (s.charAt(i) != ',') {
                            value += s.charAt(i);
                        } else {
                            if (!value.equals("empty")) {
                                arr.add(value);
                            }
                            value = "";
                        }
                    }
                }
            }
        }
```

RecipeThumbnail.java

```
/**
 * Panel for displaying the thumbnail of a recipe on the meal section panel
 */
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.*;
import javax.swing.*;

public class RecipeThumbnail extends JPanel {

    private JLabel nameLabel;
    private JLabel imgLabel;
    private JLabel imgLabelBackground;
    private JLabel timeLabel;
    private JLabel levelLabel;
    private JLabel yieldLabel;
    private BufferedImage bImg;
    private ImageIcon icon;
    private String name;
    private Image scaledImg;
    private Image newImg;
    private BufferedImage bImgBackground;
	private ImageIcon iconBackground;
	private Image scaledBackgroundImg;
	private Image newBackgroundImg;
	private ClearBtn btn;
	private String rawName;

	// name is actual recipe name with spaces
	public RecipeThumbnail(String name, String imgPath, String time, String level, String
yield, String rawName) {

		// rawName is the name of the recipe column in the database
		this.rawName = rawName;

		this.name = name;
		setLayout(null);
		setPreferredSize(new Dimension(250, 190));

		setBackground(Color.WHITE);

		// add name
		nameLabel = new JLabel(name);
		nameLabel.setFont(new Font("Arial", Font.BOLD, 20));
		nameLabel.setSize(220, 30);
		nameLabel.setHorizontalAlignment(SwingConstants.CENTER);

		nameLabel.setForeground(Color.white);
		add(nameLabel);

		// add total time
		timeLabel = new JLabel("Total Time: " + time);
		timeLabel.setFont(new Font("Arial", Font.BOLD, 15));
		timeLabel.setSize(220, 30);
		timeLabel.setLocation(10, 90);
		timeLabel.setHorizontalAlignment(SwingConstants.LEFT);
		timeLabel.setForeground(Color.white);
		add(timeLabel);

		// add level of difficulty
		levelLabel = new JLabel("Level: " + level);
		levelLabel.setFont(new Font("Arial", Font.BOLD, 15));
		levelLabel.setSize(220, 30);
		levelLabel.setLocation(10, 105);
		levelLabel.setHorizontalAlignment(SwingConstants.LEFT);
		levelLabel.setForeground(Color.white);
```

```
add(levelLabel);

// add yield
yieldLabel = new JLabel("Yield: " + yield);
yieldLabel.setFont(new Font("Arial", Font.BOLD, 15));
yieldLabel.setSize(220, 30);
yieldLabel.setLocation(10, 120);
yieldLabel.setHorizontalAlignment(SwingConstants.LEFT);
yieldLabel.setForeground(Color.white);
add(yieldLabel);

try {
        // add image
        bImg = ImageIO.read(new File(imgPath.substring(8).replace(",empty",
                                                                        "")));
        icon = new ImageIcon(bImg);
        scaledImg = icon.getImage();

        newImg = scaledImg.getScaledInstance(180, 49,
                                        java.awt.Image.SCALE_SMOOTH);
        icon = new ImageIcon(newImg);

        imgLabel = new JLabel();
        imgLabel.setIcon(icon);
        imgLabel.setSize(180, 49);
        imgLabel.setLocation(19, 30);

        add(imgLabel);
} catch (IOException e) {
        // if there is no such image, then catch the error by simply displaying
                                                                nothing
}

try {
        bImgBackground = ImageIO.read(new
File("systemImages/recipeThumbnail.png"));
} catch (IOException e) {
        e.printStackTrace();
}

iconBackground = new ImageIcon(bImgBackground);
scaledBackgroundImg = iconBackground.getImage();

newBackgroundImg = scaledBackgroundImg.getScaledInstance(220, 150,
                                                java.awt.Image.SCALE_SMOOTH);
        iconBackground = new ImageIcon(newBackgroundImg);

        imgLabelBackground = new JLabel();
        imgLabelBackground.setIcon(iconBackground);
        imgLabelBackground.setSize(220, 150);
        imgLabelBackground.setLocation(0, 0);

        add(imgLabelBackground);

        btn = new ClearBtn(name, rawName, 2);
        btn.setSize(220, 150);
        btn.setLocation(0, 0);
        add(btn);

    }

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

StarBtn.java

```
/**
 * custom JButton for individual star button on star rate bar
 */
import java.awt.event.*;
import javax.swing.ImageIcon;
import javax.swing.JButton;

public class StarBtn extends JButton implements MouseListener {

    private ImageIcon btn;
    public static int count = 0;
    private int starID;

    // set default to no star selected
    public static int selectedStar = -1;

    public StarBtn() {

        starID = count;
        count++;
```

```
		setBackground(null);
		setBorder(null);
		setBorderPainted(false);
		setFocusPainted(false);

		// image for greyed out (unselected star)
		btn = new ImageIcon("systemImages/emptyStar.png");
		setIcon(btn);

		addMouseListener(this);

	}

	@Override
	public void mouseClicked(MouseEvent e) {

		btn = new ImageIcon("systemImages/star.png");
		selectedStar = starID;
		for (int i = 0; i <= starID; i++) {
			// colour all stars up to the selected star as yellow
			StarRate.starArr.get(i).setIcon(btn);
		}
		btn = new ImageIcon("systemImages/emptyStar.png");
		for (int i = starID + 1; i < 5; i++) {
			// colour all stars following the selected stars grey
			StarRate.starArr.get(i).setIcon(btn);
		}
	}

	@Override
	public void mousePressed(MouseEvent e) {
		// TODO Auto-generated method stub

	}

	@Override
	public void mouseReleased(MouseEvent e) {
		// TODO Auto-generated method stub

	}

	@Override
	public void mouseEntered(MouseEvent e) {
		// TODO Auto-generated method stub
	}

	@Override
	public void mouseExited(MouseEvent e) {
		// TODO Auto-generated method stub

	}
}
```

StarRate.java

```
/**
 * Creates the panel that houses 5 StarBtn 's
 * and method to repaint star bar
 */
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.util.ArrayList;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class StarRate extends JPanel {

	// starArr stores all the stars
	public static ArrayList<StarBtn> starArr;
	private JLabel tasteLab;

	public StarRate() {

		StarBtn.count = 0;
		StarBtn.selectedStar = -1;
		tasteLab = new JLabel("Taste Rating: ");
		tasteLab.setFont(new Font("Arial", Font.BOLD, 20));
		tasteLab.setForeground(Color.decode("#9B9B9B"));
		add(tasteLab);

		starArr = new ArrayList<>();

		setSize(400, 50);
		setLocation(100, 100);
		setBackground(Color.decode("#F7FFF4"));
```

```
                setLayout(new FlowLayout());

                // generate 5 StarBtns and add to starArr
                for (int i = 0; i < 5; i++) {
                        StarBtn star = new StarBtn();
                        starArr.add(star);
                        add(star);
                }
        }

        /**
         * returns the chosen star level
         * @return the id number of the clicked star
         */
        public int getSelectedStar() {
                return StarBtn.selectedStar;
        }

        /**
         * updates the star bar to display the correct orientation of selected and
         * deselected stars. Called from OtherOptions class
         */
        public void repaintStarBar() {
                ImageIcon btn = new ImageIcon("systemImages/star.png");
                for (int i = 0; i <= getSelectedStar(); i++) {
                        if (i < StarRate.starArr.size())
                                StarRate.starArr.get(i).setIcon(btn);
                }
                btn = new ImageIcon("systemImages/emptyStar.png");
                for (int i = getSelectedStar() + 1; i < 5; i++) {
                        StarRate.starArr.get(i).setIcon(btn);
                }
        }
}
```

TimePanel.java

```
/**
 * creates the panel housing prep time panels used by prep and cook time
 * also includes constructor for yield bar
 */
import java.awt.*;
import javax.swing.*;
```

```
public class TimePanel extends JPanel {
	private JLabel lab;
	private JLabel hours;
	private JLabel mins;
	private TxtField fHours;
	private TxtField fMins;

	// label specifies cook or prep time
	public TimePanel(String label) {

		setBackground(Color.white);
		setLayout(new FlowLayout());
		lab = new JLabel(label);
		lab.setFont(new Font("Arial", Font.BOLD, 20));
		lab.setForeground(Color.decode("#9B9B9B"));
		add(lab);

		fHours = new TxtField(60);
		this.fHours.getTF().setPreferredSize(new Dimension(60, 40));
		add(fHours.getTF());

		hours = new JLabel("hours");
		hours.setFont(new Font("Arial", Font.BOLD, 20));
		hours.setForeground(Color.decode("#9B9B9B"));

		add(hours);

		fMins = new TxtField(60);
		this.fMins.getTF().setPreferredSize(new Dimension(60, 40));
		add(fMins.getTF());

		mins = new JLabel("mins");
		mins.setFont(new Font("Arial", Font.BOLD, 20));
		mins.setForeground(Color.decode("#9B9B9B"));

		add(mins);
	}

	// constructor for yield bar
	public TimePanel(String label, String yield) {

		setBackground(Color.white);
		setLayout(new FlowLayout());
		lab = new JLabel(label);
		lab.setFont(new Font("Arial", Font.BOLD, 20));
		lab.setForeground(Color.decode("#9B9B9B"));
		add(lab);

		fHours = new TxtField(60);
		this.fHours.getTF().setPreferredSize(new Dimension(60, 40));
		add(fHours.getTF());

	}

	// setter methods for time and yield
	public void setTime(String hours, String mins) {
		fHours.getTF().setText(hours);
		fMins.getTF().setText(mins);
	}

	public void setYield(String yield) {
		fHours.getTF().setText(yield);
	}

	// getter methods for times and yields
	public String getHours() {
		if (fHours.getTF().getText().equals("")) {
			return "empty";
		}
		return fHours.getTF().getText();
	}

	public String getMins() {
		if (fMins.getTF().getText().equals("")) {
			return "empty";
		}
		return fMins.getTF().getText();
	}

	public String getYield() {
		if (fHours.getTF().getText().equals("")) {
			return "empty";
		}
		return fHours.getTF().getText();
	}
}
```

TxtField.java

```
/**
 * creates text field which has a dotted border and is used by Recipe and MealSection templates
 */
import java.awt.Color;
import java.awt.Font;
import javax.swing.*;
import javax.swing.border.Border;

public class TxtField {

	private final JTextField txtField;

	public TxtField(int tfLength) {

		txtField = new JTextField("");
		txtField.setFont(new Font("Arial", Font.PLAIN, 15));
		txtField.setSize(tfLength, 40);
		txtField.setForeground(Color.decode("#9B9B9B"));
		txtField.setLocation(50, 190 + RecipeTemplate.ingSpacing);
		Border border = BorderFactory.createDashedBorder(Color.decode("#9B9B9B"),
2, 1, 3, true);
		txtField.setBorder(border);

	}

	public JTextField getTF() {
		return txtField;
	}
}
```