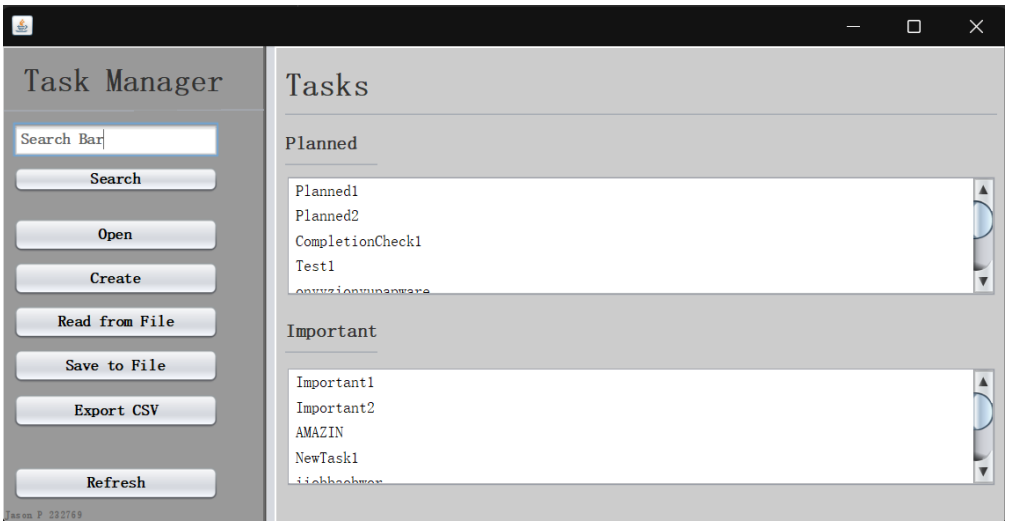
Jason Prinsloo
JD522 FA2
20232769

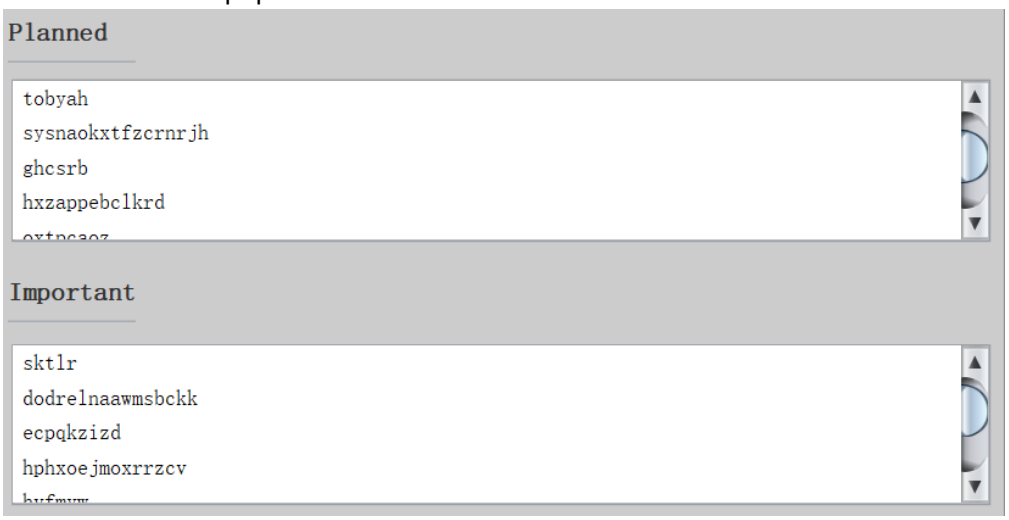
Screenshots of Project execution:

Home screen:

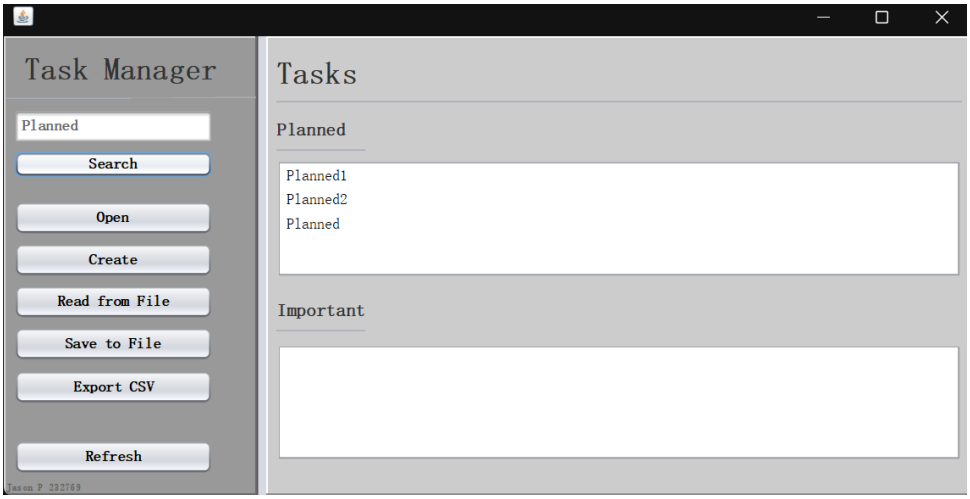


The Task Manager splits the tasks on the left side into Planned and Important.
On the right side is the all the commands.

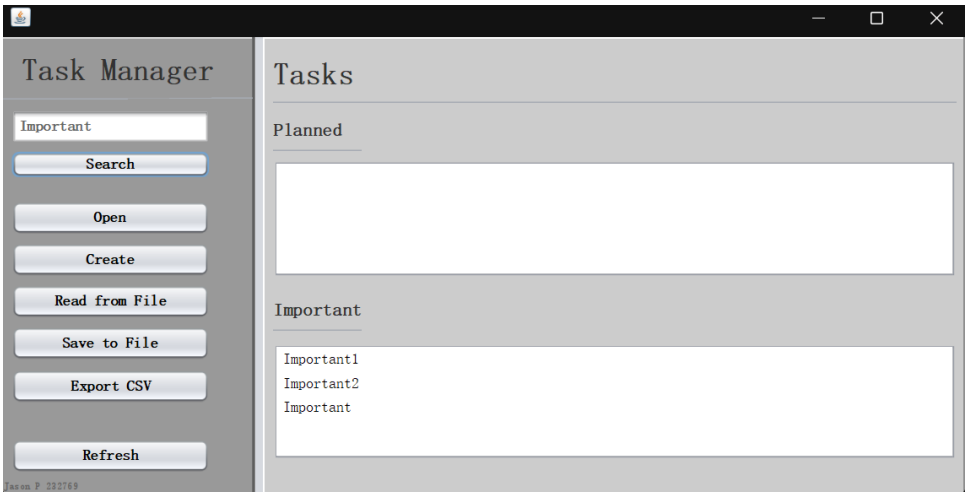
The Database was populated with Test values and then with random values.

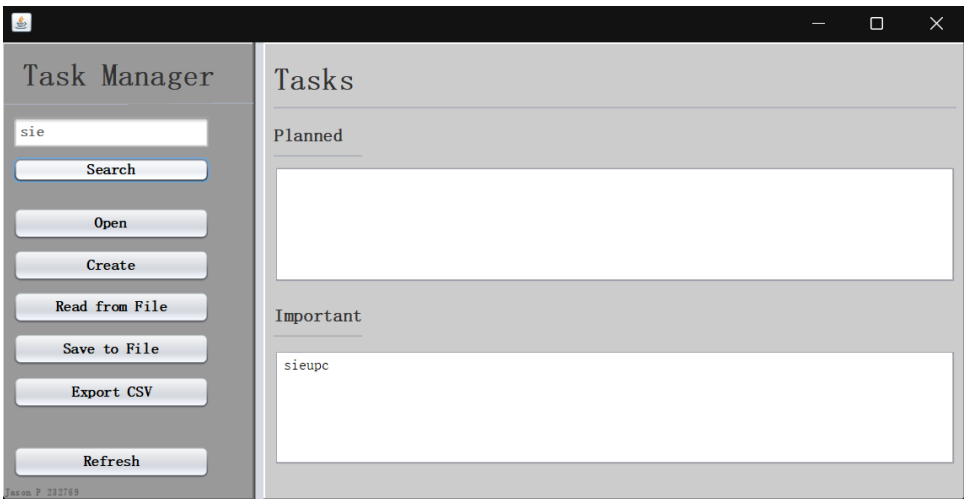


The Search Bar works.

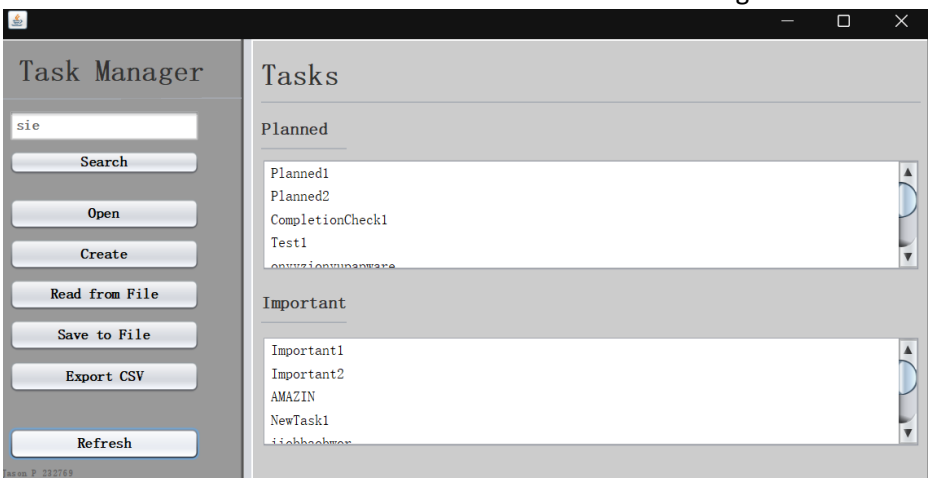


When searching for Tasks named Planned, all the planned ones will show in Planned. Same with Important, and when searching for others it will show.

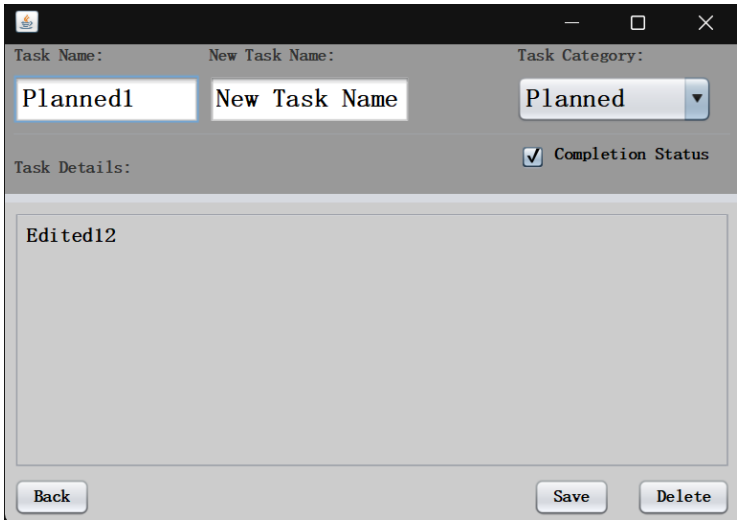


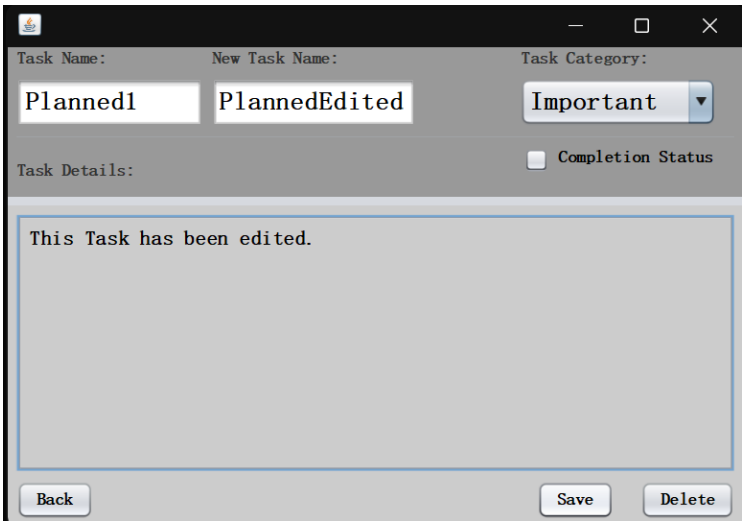


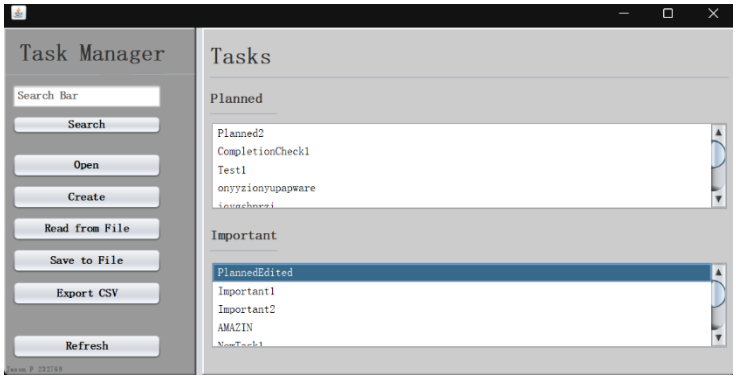
The Refresh button works and refreshes the whole Task Manager.

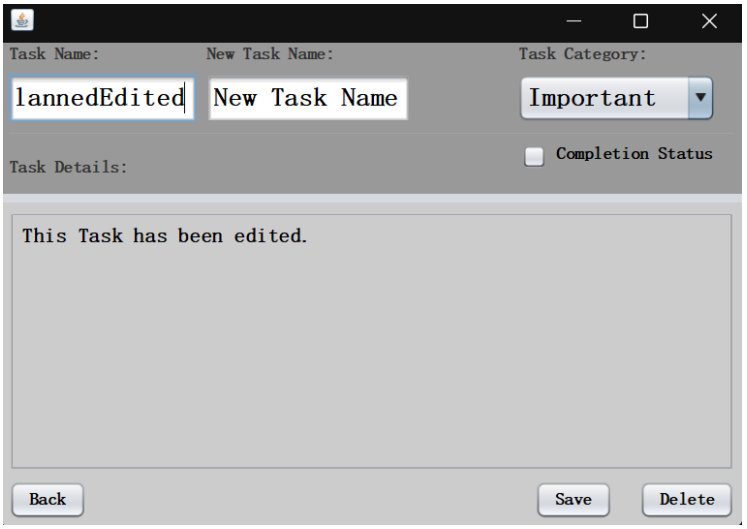


The Open button opens a new form allowing the user to view the Task they entered in the search bar's details and to edit it.

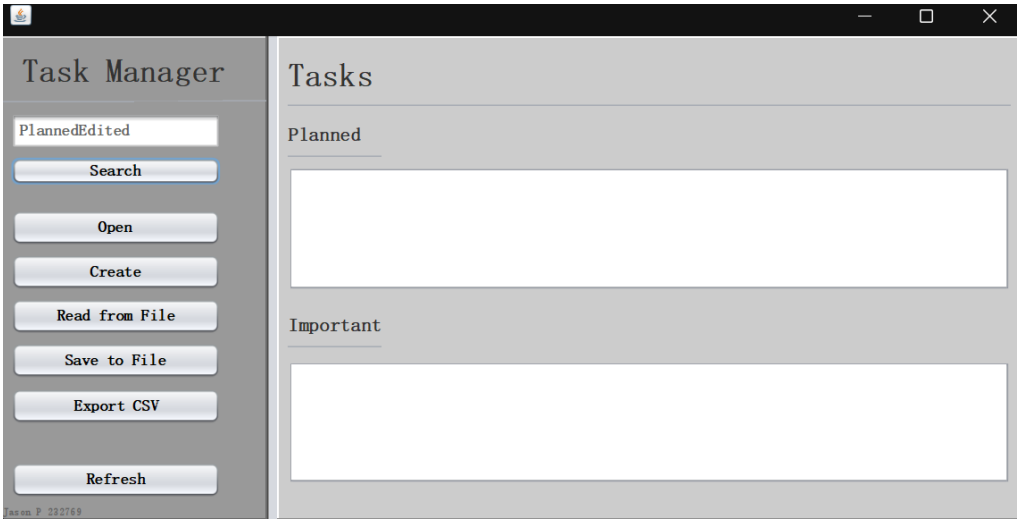




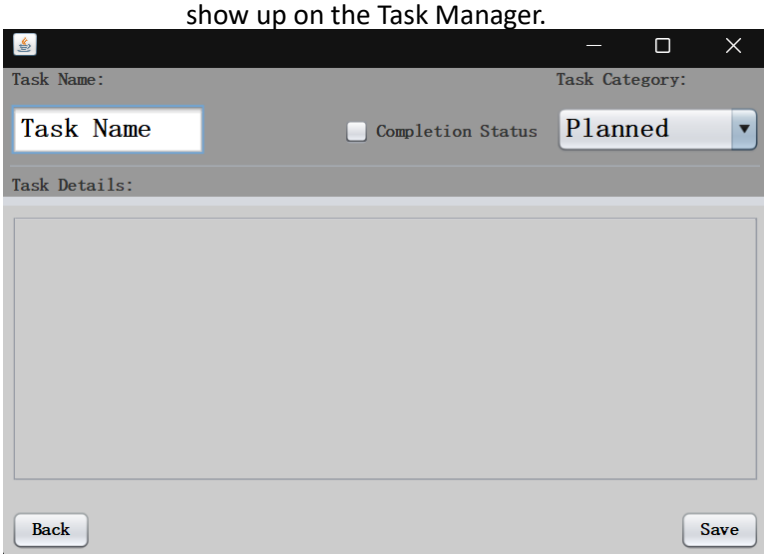


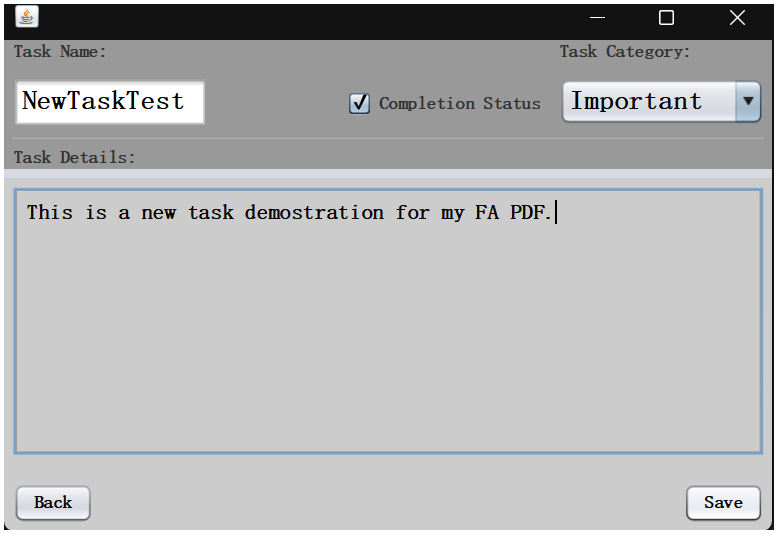


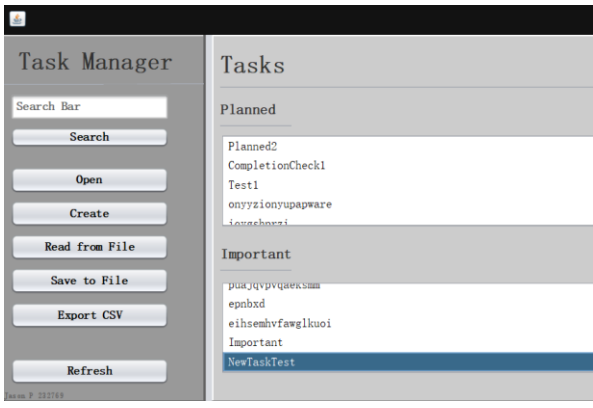
The Delete button also deletes the entry from the database and the Task Manager.

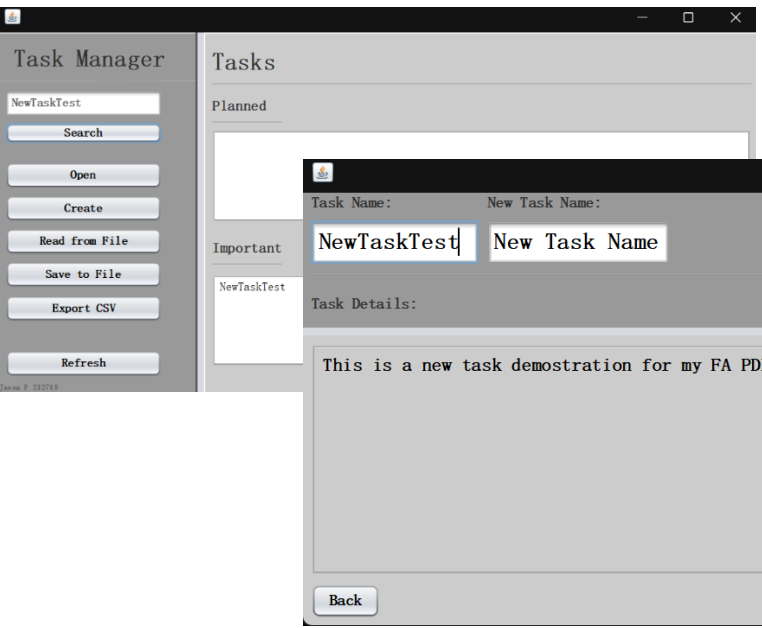


The Create button adds a new entry to the database and allows the user to create a new task that will show up on the Task Manager.

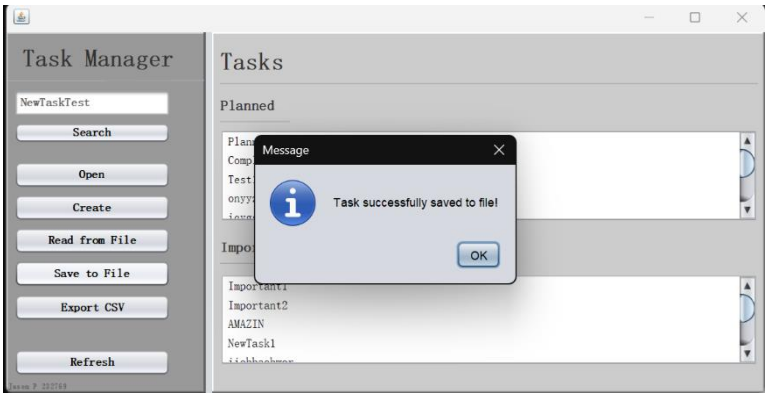


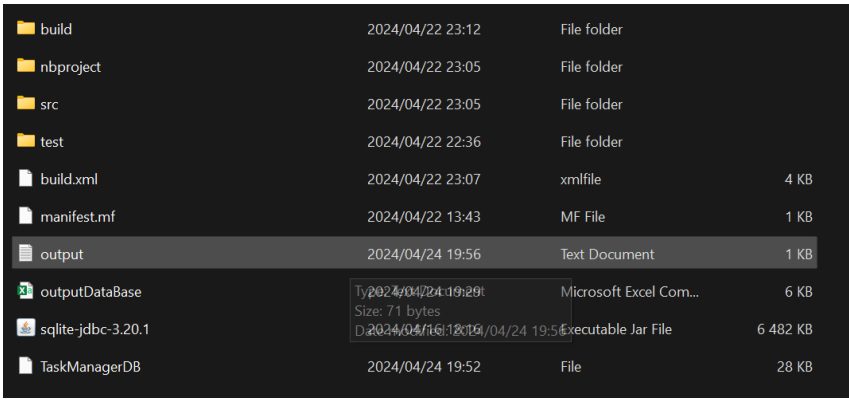


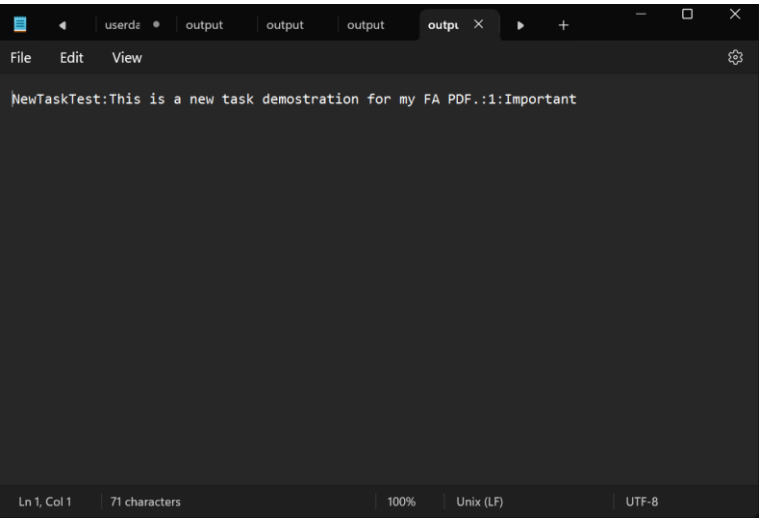




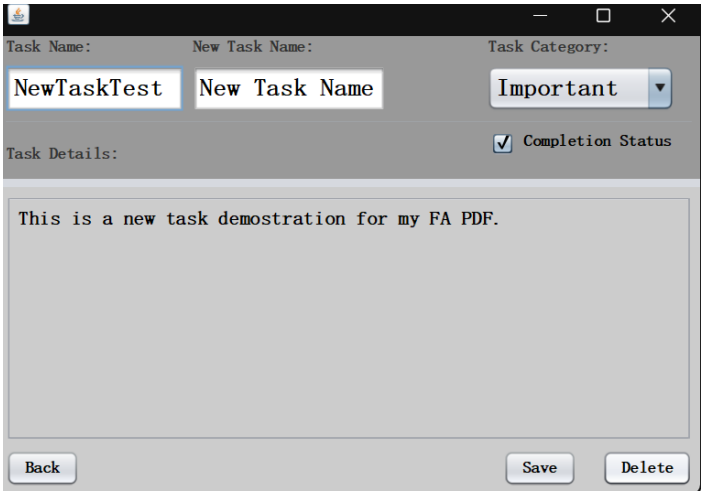
The Save To File and Read from File works.



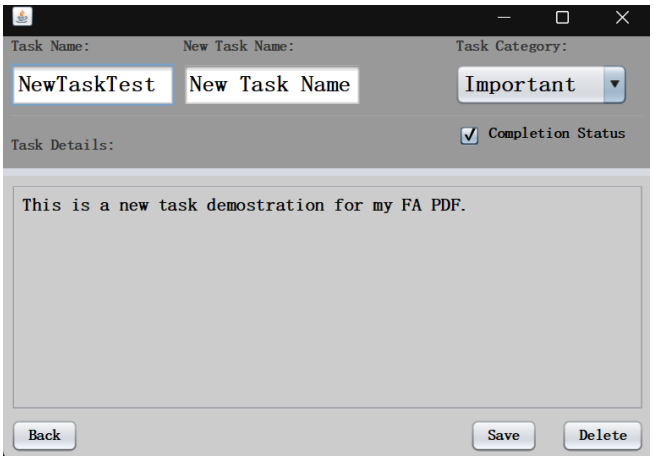


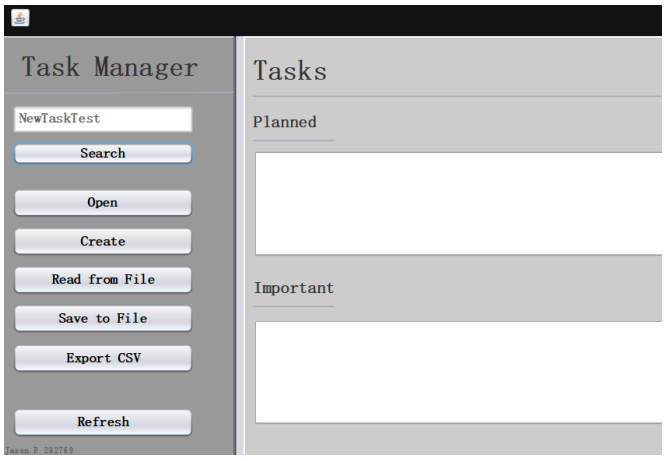


It's been saved in that format so it is easier to read from the file and add it to the Task Manager.
For this Demonstration I'm going to delete it and then re-add it by reading the saved file.

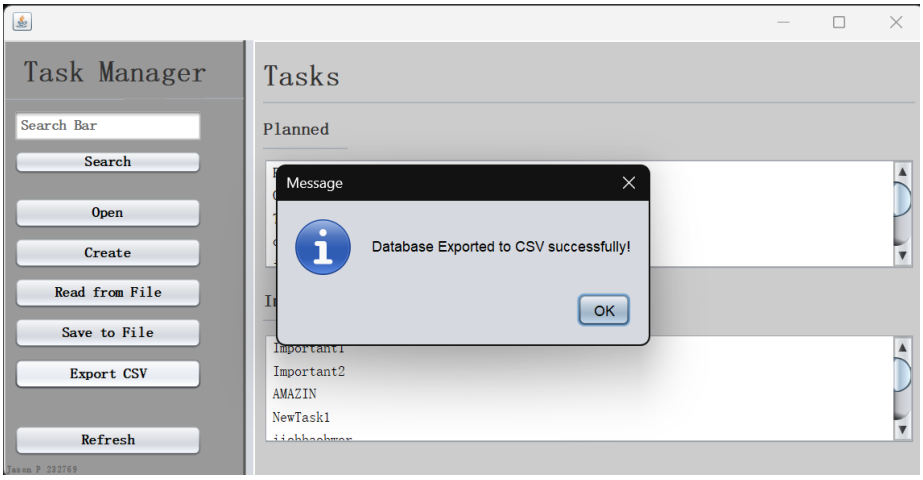


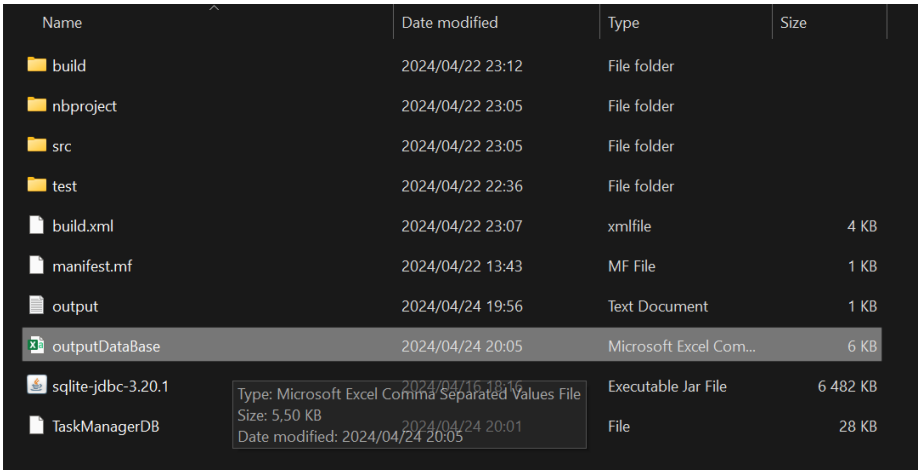
When clicking on "Read from File" the Open form will open with the data from the text file.

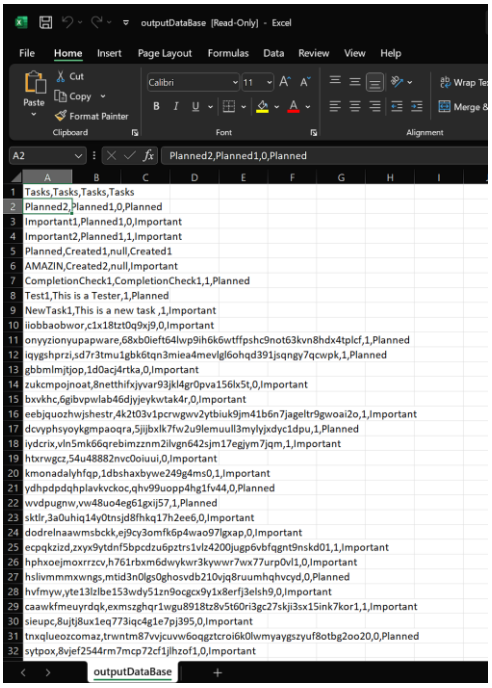




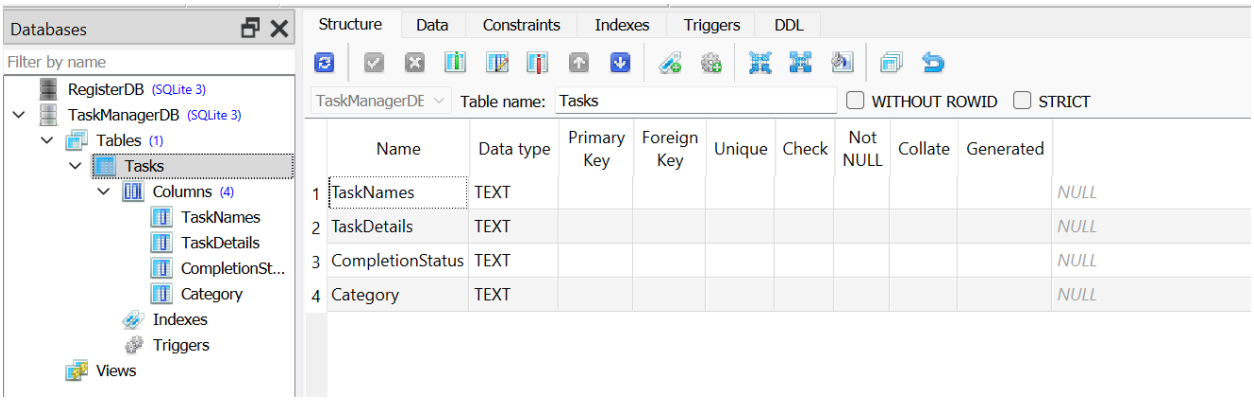
Export to CSV works.





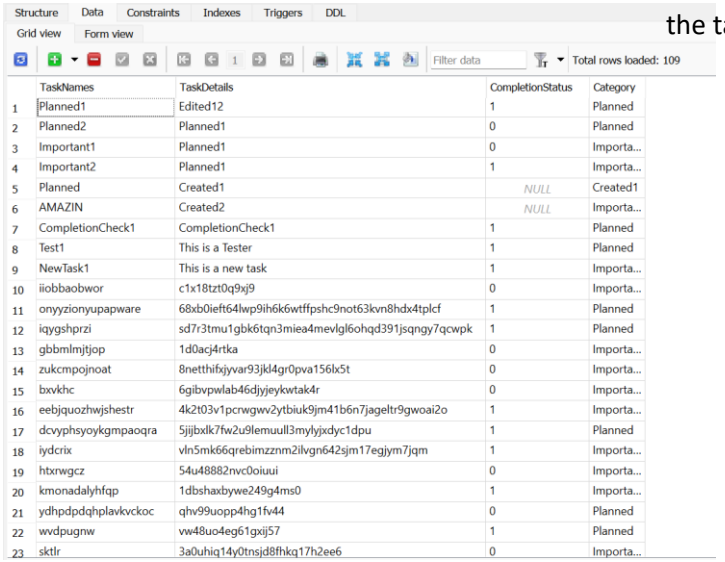


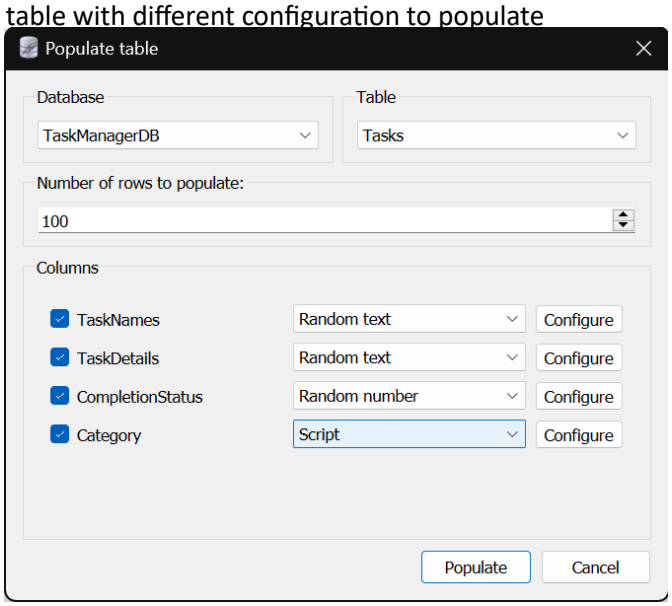
The database was made as follows:

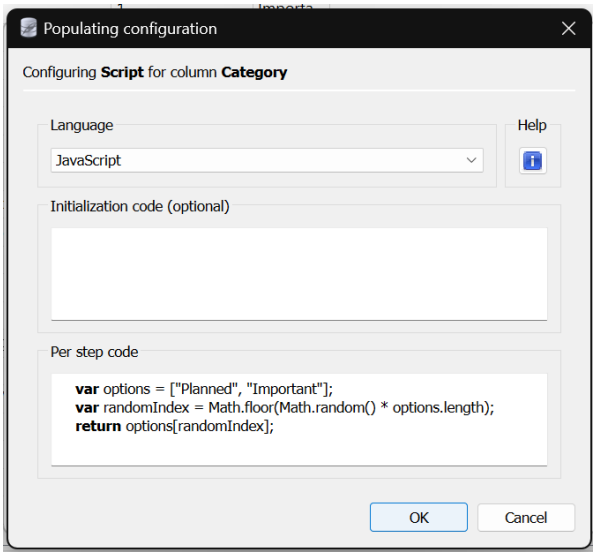


I did not add a Unique key so that there can be multiple tasks with the same name.

I used Populate table with different configuration to populate the table.

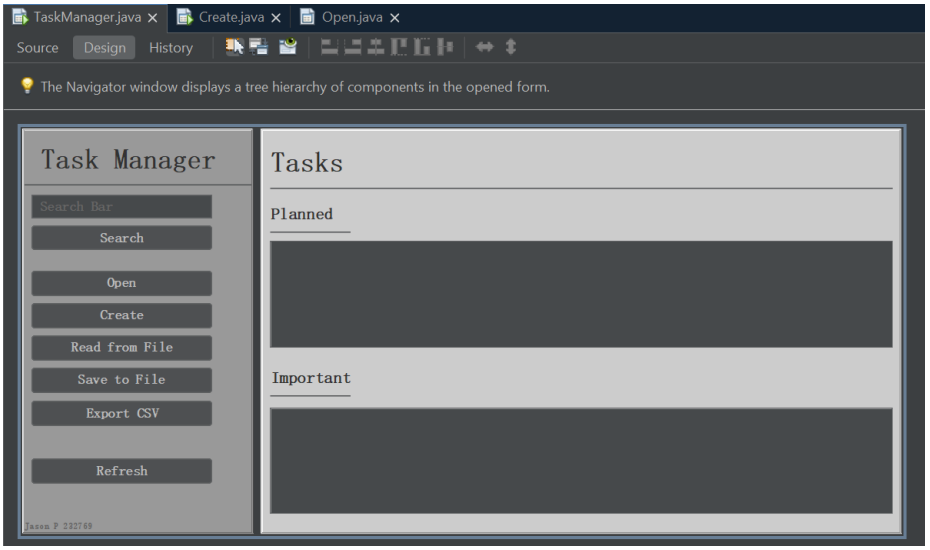


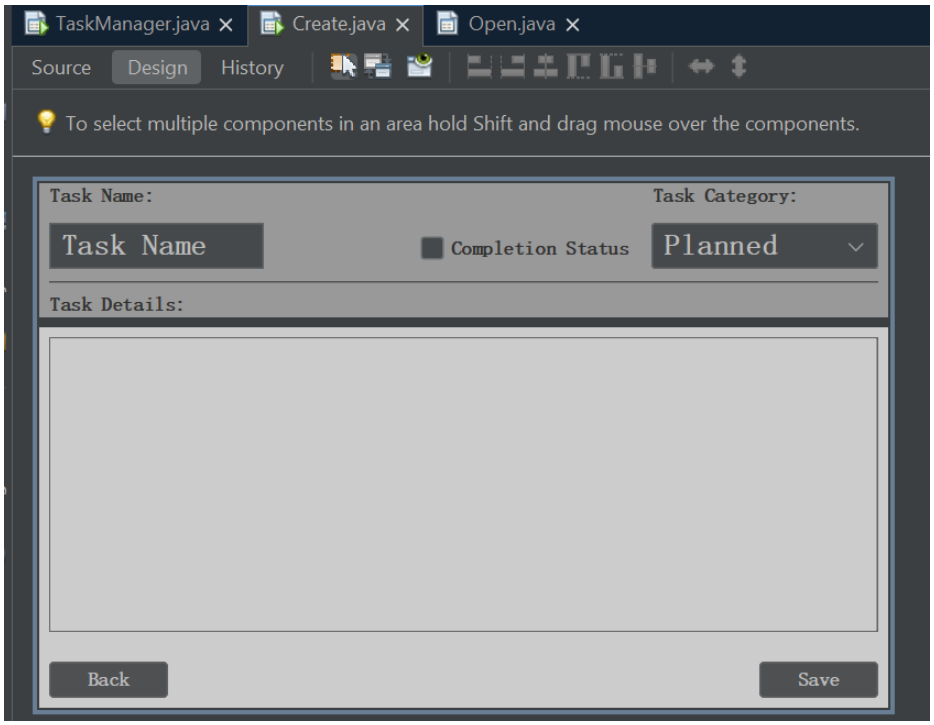


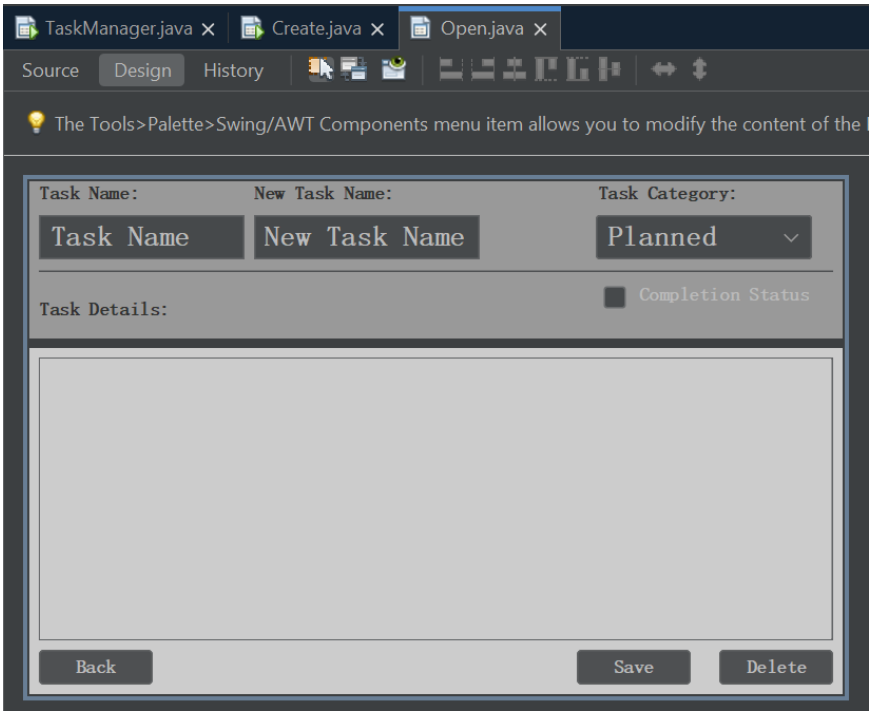


This is the JavaScript is used to generate random values between Important and Planned for Category.

The Design View for all Forms.

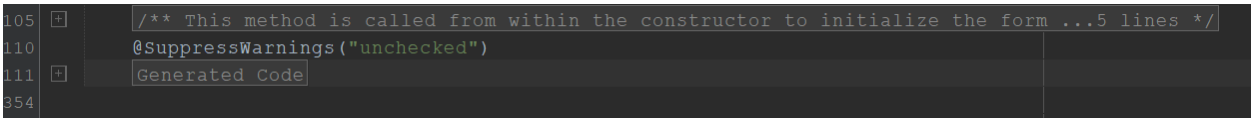






Source Code:

I’m not copying this that takes over 200 lines…



This is for TaskManager Form, the whole project consists out of TaskManager, Create and Open Forms

```
package taskmanagerfa;

import java.io.*;
import java.sql.*;
import java.awt.HeadlessException;
import java.util.Arrays;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.DefaultListModel;
import javax.swing.JOptionPane;

public class TaskManager extends javax.swing.JFrame {

   private Connection conn;
   private PreparedStatement ps;
   private ResultSet rs;


   public TaskManager() {

      this.model = new DefaultListModel();
      this.model1 = new DefaultListModel();
      initComponents();
      try
      {
      conn = DriverManager.getConnection("jdbc:sqlite:F:\\JD522 FA2\\TaskManagerFA2\\TaskManagerDB");
      //connects to the db and reloads the task manager to load all the database entries.
      Reload();
      }
      catch(SQLException error)
      {
      error.printStackTrace();
      }
   }
   DefaultListModel model;
   DefaultListModel model1;

   public void Search(String searchValue){
      String searchQuery1 = "SELECT * FROM Tasks WHERE TaskNames LIKE ? ";

      try {ps = conn.prepareStatement(searchQuery1);
         ps.setString(1, "%"+searchValue +"%");
         rs =ps.executeQuery();

         //clear whats in list
         while(model.getSize()>0){
            model.removeAllElements();
```

```
        }

        //ADD INFO To List1
        if("Planned".equals(rs.getString("Category"))){
            while(rs.next()){
                String[]row ={
                    rs.getString("TaskNames"),
                };
                //removes the [] so it looks nice
                String rowAsString = Arrays.toString(row);
                model.addElement(rowAsString.replace("[", "").replace("]",""));
            }
        }
        rs.close();
        ps.close();

    } catch (Exception ex) {
        JOptionPane.showMessageDialog(rootPane, ex);
    }

    try {
        ps = conn.prepareStatement(searchQuery1);
        ps.setString(1, "%"+searchValue +"%");
        rs =ps.executeQuery();

        //clear whats in list
        while(model1.getSize()>0){
            model1.removeAllElements();
        }

        //ADD INFO To List2
        if("Important".equals(rs.getString("Category"))){
            while(rs.next()){
                String[]row ={
                    rs.getString("TaskNames"),
                };
                //removes the [] so it looks nice
                String rowAsString = Arrays.toString(row);
                model1.addElement(rowAsString.replace("[", "").replace("]",""));
            }
        }
        rs.close();
        ps.close();

    } catch (Exception ex) {
        JOptionPane.showMessageDialog(rootPane, ex);
    }

  }

private void Reload(){
        //reloads the enitre task manager
    String importantQuery = "SELECT *FROM Tasks WHERE Category = 'Important'";
    String plannedQuery = "SELECT *FROM Tasks WHERE Category = 'Planned'";
        try{
            jList1.setModel(model);
            ps=conn.prepareStatement(plannedQuery);
            rs=ps.executeQuery();

            //clear whats in table
            while(model.getSize()>0){
```

```
            model.removeAllElements();
        }

        //ADD INFO To table
        while(rs.next()){
            String[]row ={
                rs.getString("TaskNames"),
            };
            String rowAsString = Arrays.toString(row);
            model.addElement(rowAsString.replace("[", "").replace("]",""));
        }
        rs.close();
        ps.close();

    }catch(Exception ex)
    {
        JOptionPane.showMessageDialog(rootPane, ex);
    }

    try{
        jList2.setModel(model1);
        ps=conn.prepareStatement(importantQuery);
        rs=ps.executeQuery();

        //clear whats in table
        while(model1.getSize()>0){
            model1.removeAllElements();
        }

        //ADD INFO To table
        while(rs.next()){
            String[]row ={
                rs.getString("TaskNames"),
            };
            String rowAsString = Arrays.toString(row);
            model1.addElement(rowAsString.replace("[", "").replace("]",""));
        }
        rs.close();
        ps.close();

    }catch(Exception ex)
    {
        JOptionPane.showMessageDialog(rootPane, ex);
    }

}
private void searchFieldActionPerformed(java.awt.event.ActionEvent evt) {
    //double clicked by accident on this one...
}

private void refreshBtnActionPerformed(java.awt.event.ActionEvent evt) {
    Reload();
      //calls reload to reload everything
}

private void searchBtnActionPerformed(java.awt.event.ActionEvent evt) {
    String searchValue = searchField.getText();
    Search(searchValue);
    //uses search with the searchValue to search for the users entered data.
}
```

```
    private void openBtnActionPerformed(java.awt.event.ActionEvent evt) {
        String searchValue = searchField.getText();
        //error handling for when a user enters an invalid Task Name
        try{
            ps = conn.prepareStatement("SELECT 1 FROM Tasks WHERE TaskNames = ?");
            ps.setString(1, searchValue);
            try(ResultSet rs = ps.executeQuery()){
                if(rs.next()){
                    Open open = new Open();
                    open.setVisible(true);
                    open.loadSearch(searchValue);
                    this.dispose();
                }else{
                    JOptionPane.showMessageDialog(rootPane, "Pleas eneter a valid Task Name.");
                }
            }

        }catch(Exception ex){
            JOptionPane.showMessageDialog(rootPane, ex);
        }
    }

    private void createBtnActionPerformed(java.awt.event.ActionEvent evt) {
        Create create = new Create();
        create.setVisible(true);
        this.dispose();
        //opens the Create form
    }

    private void saveToFileBtnActionPerformed(java.awt.event.ActionEvent evt) {
        String taskQuery = "SELECT * FROM Tasks WHERE TaskNames LIKE ? ";
        String searchValue = searchField.getText();
        try{
            ps = conn.prepareStatement(taskQuery);
            ps.setString(1, "%"+searchValue +"%");
            rs = ps.executeQuery();
            String taskName = rs.getString("TaskNames");
            String taskDetails = rs.getString("TaskDetails");
            String taskCompletionStatus = rs.getString("CompletionStatus");
            String taskCategory = rs.getString("Category");
            rs.close();
            ps.close();
            //fetches the current data in the fields and saves it in a format to the file.
            try{
                String data = taskName+":"+taskDetails+":"+taskCompletionStatus+":"+taskCategory+"\n";
                OutputStream output = new FileOutputStream("F:\\JD522 FA2\\TaskManagerFA2\\output.txt");
                byte[] toSaveList = data.getBytes();
                output.write(toSaveList);
                JOptionPane.showMessageDialog(rootPane,"Task successfully saved to file!");
            }catch(Exception ex){
                JOptionPane.showMessageDialog(rootPane, ex);
            }


        }catch(HeadlessException | SQLException ex){
            JOptionPane.showMessageDialog(rootPane, ex);
        }
    }
private void readFromFileBtnActionPerformed(java.awt.event.ActionEvent evt) {
        Open open = new Open();
        open.setVisible(true);
```

```
    open.setTask();
    //setTask in Open.java that fetches the data in the file and uses the format i set it to, to insert it into an Open Form
    this.dispose();
}

private void exportCSVActionPerformed(java.awt.event.ActionEvent evt) {

    String csvOutput = "outputDataBase.csv";

    try{

        ps = conn.prepareStatement("SELECT * FROM Tasks");
        rs = ps.executeQuery();

        FileWriter fw = new FileWriter(csvOutput);
        ResultSetMetaData meta = rs.getMetaData();
        int numClms = meta.getColumnCount();
        String dataHeaders = "";

        for(int i = 1; i<= numClms; i++){
            dataHeaders += meta.getCatalogName(i) + ",";
        }

        fw.append(dataHeaders.substring(0, dataHeaders.length()- 1) + "\n");

        while(rs.next()){
            String rowData = "";

            for(int i = 1; i <= numClms; i++){
                rowData += rs.getString(i) + ",";
            }
            fw.append(rowData.substring(0, rowData.length() - 1) + "\n");
        }
        JOptionPane.showMessageDialog(rootPane, "Database Exported to CSV successfully!");

    }catch(SQLException ex){
        JOptionPane.showMessageDialog(rootPane, ex);
        ex.printStackTrace();
    } catch (IOException ex) {
        Logger.getLogger(TaskManager.class.getName()).log(Level.SEVERE, null, ex);
    }
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(TaskManager.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
```

```
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(TaskManager.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(TaskManager.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(TaskManager.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new TaskManager().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton createBtn;
    private javax.swing.JButton exportCSV;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JList<String> jList1;
    private javax.swing.JList<String> jList2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JSeparator jSeparator2;
    private javax.swing.JSeparator jSeparator3;
    private javax.swing.JSeparator jSeparator4;
    private javax.swing.JButton openBtn;
    private javax.swing.JButton readFromFileBtn;
    private javax.swing.JButton refreshBtn;
    private javax.swing.JButton saveToFileBtn;
    private javax.swing.JButton searchBtn;
    private javax.swing.JTextField searchField;
    // End of variables declaration
}
```

This is for the Create Form.

```
package taskmanagerfa;

import java.util.logging.Level;
import java.util.logging.Logger;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class Create extends javax.swing.JFrame {

   private Connection conn;
   private PreparedStatement ps;
   private ResultSet rs;

   public Create() {
      initComponents();

      try
      {
      conn = DriverManager.getConnection("jdbc:sqlite:F:\\JD522 FA2\\TaskManagerFA2\\TaskManagerDB");
      //connects to the db and reloads the task manager to load all the database entries.
      }
      catch(SQLException error)
      {
      error.printStackTrace();
      }
   }
  private void categoryComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
      //accidentally double clicked on this one also...
   }

   private void backBtnActionPerformed(java.awt.event.ActionEvent evt) {
      TaskManager tm = new TaskManager();
      tm.setVisible(true);
      this.dispose();
   }

   private void saveBtnActionPerformed(java.awt.event.ActionEvent evt) {
      try {
         if(!completionCheckBox.isSelected()){
            String taskName = taskNameTextField.getText();
            String taskDetails = descriptionTextArea.getText();
            String taskCategory = categoryComboBox.getSelectedItem().toString();
            String completionStatus = "0";
            String insertQuery = "INSERT INTO Tasks (TaskDetails, Category, TaskNames, CompletionStatus) VALUES('"+ taskDetails +"','"+
taskCategory +"','"+ taskName +"','"+ completionStatus +"');";
            PreparedStatement ps = conn.prepareStatement(insertQuery);
            ps.executeUpdate();
         }else{
            String taskName = taskNameTextField.getText();
            String taskDetails = descriptionTextArea.getText();
            String taskCategory = categoryComboBox.getSelectedItem().toString();
            String completionStatus = "1";
            String insertQuery = "INSERT INTO Tasks (TaskDetails, Category, TaskNames, CompletionStatus) VALUES('"+ taskDetails +"','"+
taskCategory +"','"+ taskName +"','"+ completionStatus +"');";
            PreparedStatement ps = conn.prepareStatement(insertQuery);
            ps.executeUpdate();
         }
```

```
        } catch (SQLException ex) {
            Logger.getLogger(Open.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Create.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Create.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(Create.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(Create.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Create().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JComboBox<String> categoryComboBox;
    private javax.swing.JCheckBox completionCheckBox;
    private javax.swing.JTextArea descriptionTextArea;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JButton saveBtn;
    private javax.swing.JTextField taskNameTextField;
    // End of variables declaration
}
```

This is for Open form.

```
package taskmanagerfa;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.io.IOException;
import java.io.FileReader;
import java.io.BufferedReader;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;


public class Open extends javax.swing.JFrame {

   private Connection conn;
   private PreparedStatement ps;
   private ResultSet rs;

   public Open() {
      initComponents();

      try{
      conn = DriverManager.getConnection("jdbc:sqlite:F:\\JD522 FA2\\TaskManagerFA2\\TaskManagerDB");
      }catch(SQLException error)
      {
         error.printStackTrace();
      }
   }
   private void saveBtnActionPerformed(java.awt.event.ActionEvent evt) {
      try {
         String updateQuery = "UPDATE Tasks SET CompletionStatus = ?, TaskDetails = ?, Category = ?, TaskNames = ? WHERE TaskNames = ?";
         PreparedStatement ps = conn.prepareStatement(updateQuery);

         if(!completionCheckBox.isSelected()){
            ps.setString(1,"0");
         }else{
            ps.setString(1, "1");
         }

         ps.setString(2,descriptionTextArea.getText());
         ps.setString(3, categoryComboBox.getSelectedItem().toString());
         ps.setString(4, newTaskNameTextField.getText());
         ps.setString(5, taskNameTextField.getText());

         ps.executeUpdate();
      } catch (SQLException ex) {
         Logger.getLogger(Open.class.getName()).log(Level.SEVERE, null, ex);
      }
   }

   private void backBtnActionPerformed(java.awt.event.ActionEvent evt) {
      TaskManager tm = new TaskManager();
      tm.setVisible(true);
      this.dispose();
   }

   private void deleteBtnActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    String deleteQuery = "DELETE FROM Tasks WHERE TaskNames = ?";
    try {
        PreparedStatement ps = conn.prepareStatement(deleteQuery);
        ps.setString(1, taskNameTextField.getText());
        ps.executeUpdate();
        TaskManager tm = new TaskManager();
        tm.setVisible(true);
        this.dispose();
    } catch (SQLException ex) {
        Logger.getLogger(Open.class.getName()).log(Level.SEVERE, null, ex);
    }
}

private void categoryComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

public void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(Open.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(Open.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(Open.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(Open.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Open().setVisible(true);
        }
    });

}

public void setTask(){
    try {
        FileReader reader = new FileReader("F:\\JD522 FA2\\TaskManagerFA2\\output.txt");
        BufferedReader breader = new BufferedReader(reader);



        String line;
        while((line = breader.readLine()) != null){
            String[] data = line.split(":");
```

```
            if(data.length >= 4){
                String value1 = data[0].trim();
                String value2 = data[1].trim();
                String value3 = data[2].trim();
                String value4 = data[3].trim();

                taskNameTextField.setText(value1);
                descriptionTextArea.setText(value2);
                if("1".equals(value3)){
                    completionCheckBox.setSelected(true);
                }else{
                    completionCheckBox.setSelected(false);
                }
                if("Important".equals(value4)){
                    categoryComboBox.setSelectedItem("Important");
                }else{
                    categoryComboBox.setSelectedItem("Planned");
                }

            }
        }
        breader.close();


    } catch (IOException ex) {
        Logger.getLogger(Open.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public void loadSearch(String searchValue){
    String searchQuery1 = "SELECT * FROM Tasks WHERE TaskNames LIKE ? ";

    //Gets search string to populate the open form with the users search value
//      TaskManager taskManager = new TaskManager();
//      SearchString searchString = taskManager.new SearchString();
//      String searchValue = searchString.getSearchString();


    try {ps = conn.prepareStatement(searchQuery1);
        ps.setString(1, "%"+searchValue +"%");
        rs =ps.executeQuery();

        taskNameTextField.setText(rs.getString("TaskNames"));
        descriptionTextArea.setText(rs.getString("TaskDetails"));

        if("1".equals(rs.getString("CompletionStatus"))){
            completionCheckBox.setSelected(true);
        }else{
            completionCheckBox.setSelected(false);
        }

        if("Important".equals(rs.getString("Category"))){
            categoryComboBox.setSelectedItem("Important");
        }else{
            categoryComboBox.setSelectedItem("Planned");
        }


        rs.close();
        ps.close();
```

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



// Variables declaration - do not modify
private javax.swing.JButton backBtn;
private javax.swing.JComboBox<String> categoryComboBox;
private javax.swing.JCheckBox completionCheckBox;
private javax.swing.JButton deleteBtn;
private javax.swing.JTextArea descriptionTextArea;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JTextField newTaskNameTextField;
private javax.swing.JButton saveBtn;
private javax.swing.JTextField taskNameTextField;
// End of variables declaration
```