¬ App: Appliance Comparison Program

A basic application to compare various home appliances, their energy usage and cost.

Created as a team project to develop our skills in UX and software development.

This program was created through the use of an **in-memory H2Database** to facilitate our dataset as well as **JavaFX** to create our user-interface.

Created by Team ¬Dev:

[Michelle](https://github.com/michellesuchang)

[Tim](https://github.com/podoodoo)

[Brandon](https://github.com/Ciyon)

[Devon](https://github.com/DevonS3)

Table of Contents

- **¬ App: Appliance Comparison Program**
  - Table of Contents
  - Features and Functionality
    - Search Page
    - Comparison Page
    - User Profile Page
  - Development Procedure
  - Front-End: JavaFX
  - Back-End: H2Database

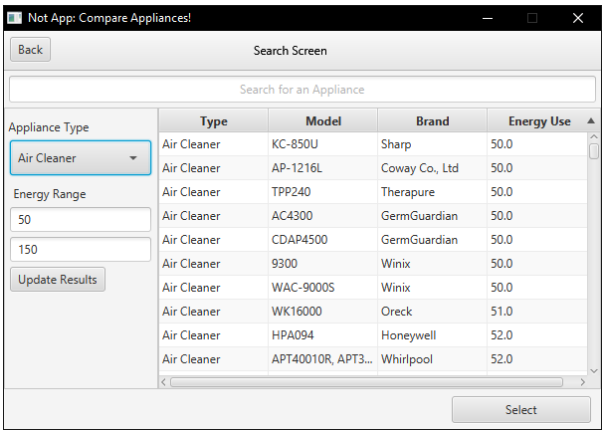
Features and Functionality

Search Page

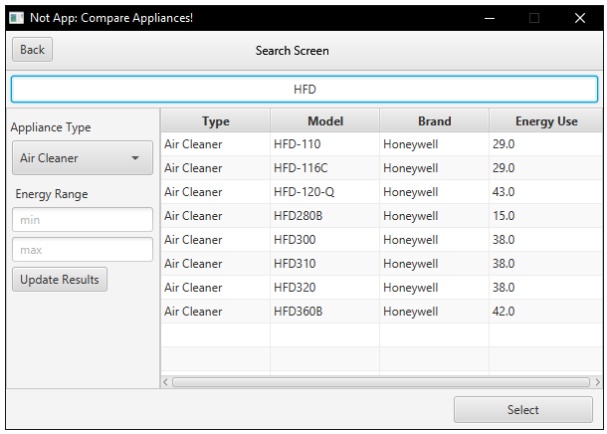
The Search Page provides basic functionality to search through multiple appliance data sources.

Namely, the U.S. Department of Energy's Database and Energy Star's Certified Product Data Sets. Additionally, the user is given the option to apply filters to the DB query for: the Appliance Type, Energy Usage, and Model.

Filter by Energy Usage



Search by Model

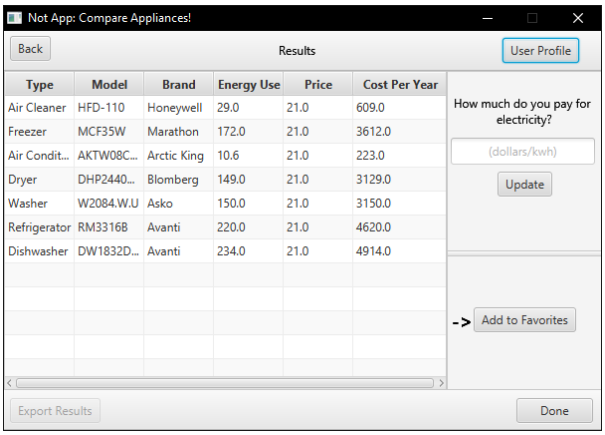


Comparison Page

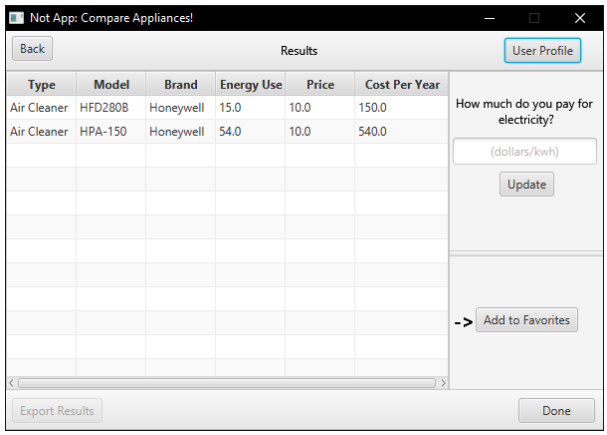
The comparison page displays the appliances selected through the search page. Furthermore, the user is given the option to input the dollar amount that they pay for each kilowatt hour (kWh).

A **use-case** for this specific page would be to estimate the cost of a collection of products that user owns. Another use case would be to compare two different appliances that the user may consider purchasing.

Use-Case 1

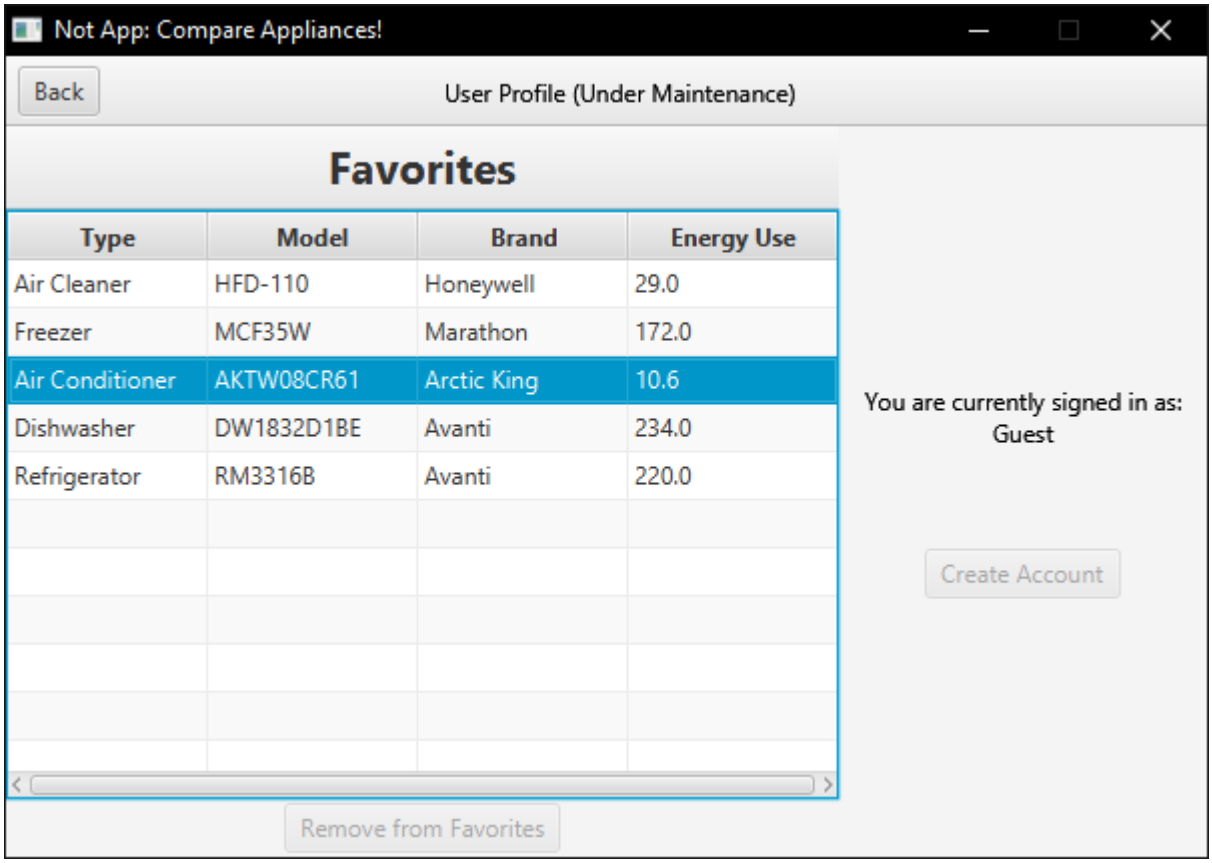


Use-Case 2



User Profile Page

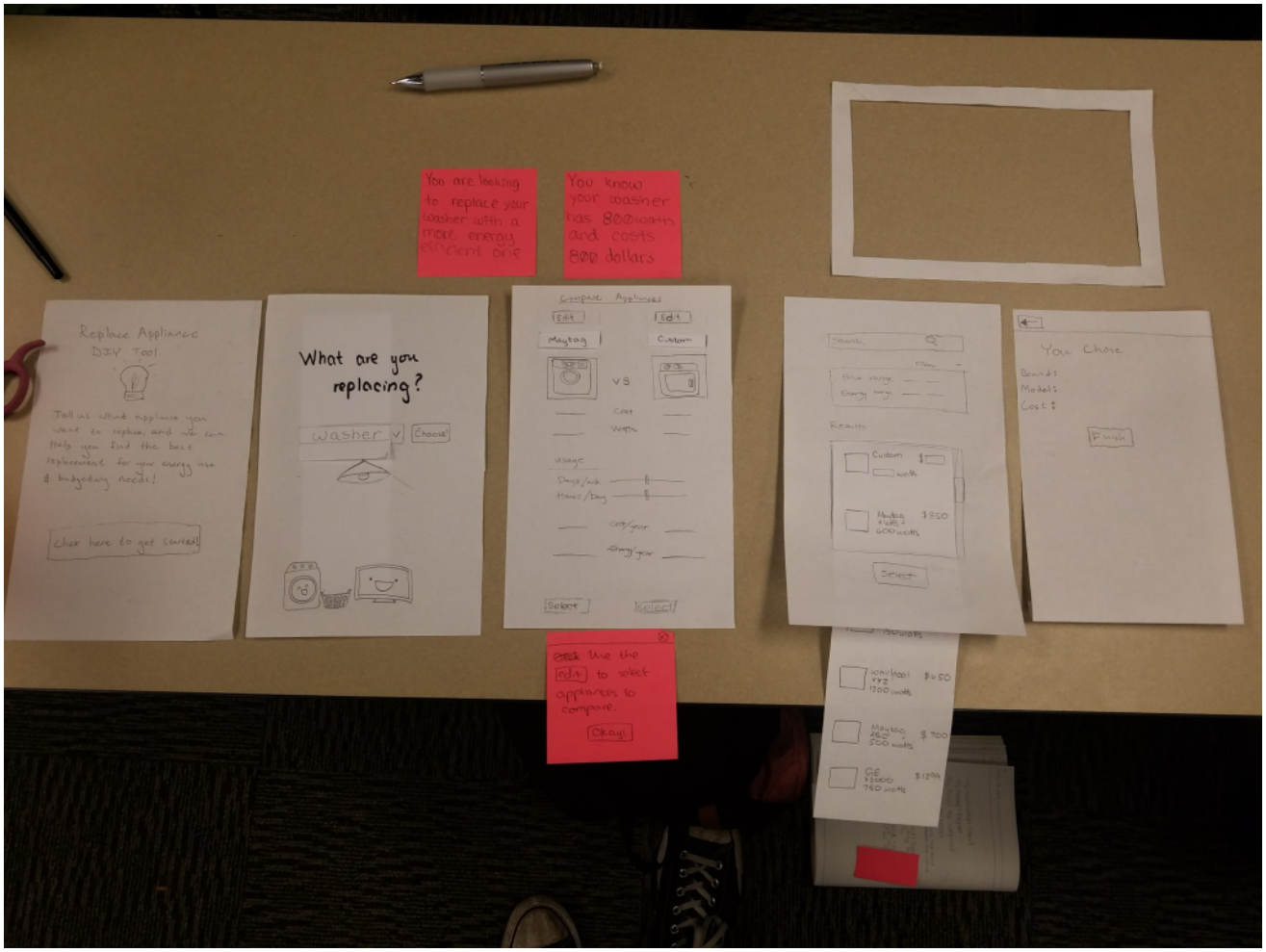
The user page is currently incomplete, although has a functioning favorites page. Which allows users to create a separate list of appliances from the comparison page.

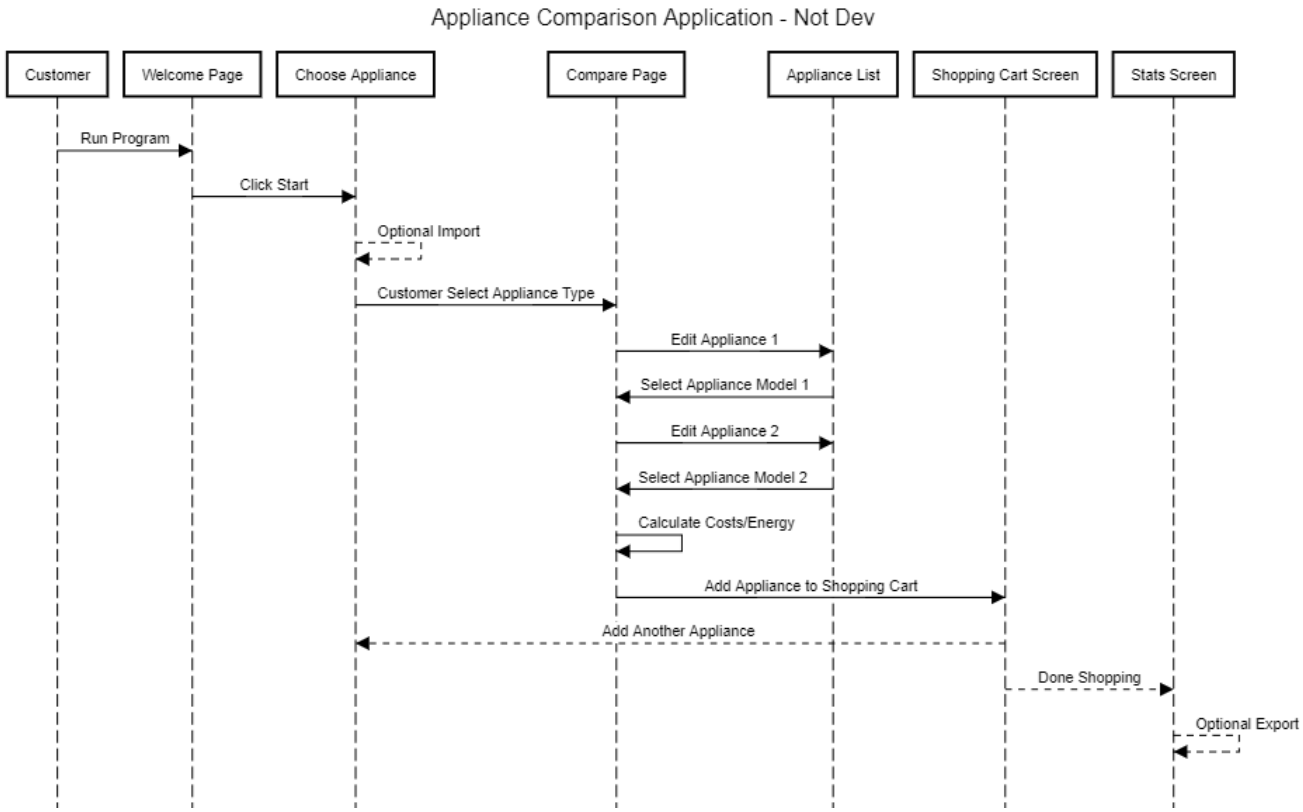


Development Procedure

This was a **Team project** where we had created and designed an application to compare a variety of different appliances and the data that relates to them; such as price, energy cost, name and manufacturer.

We had **analyzing requirements** as well as created prototypes and mockups through involved UX which had then gone through review through our peers.

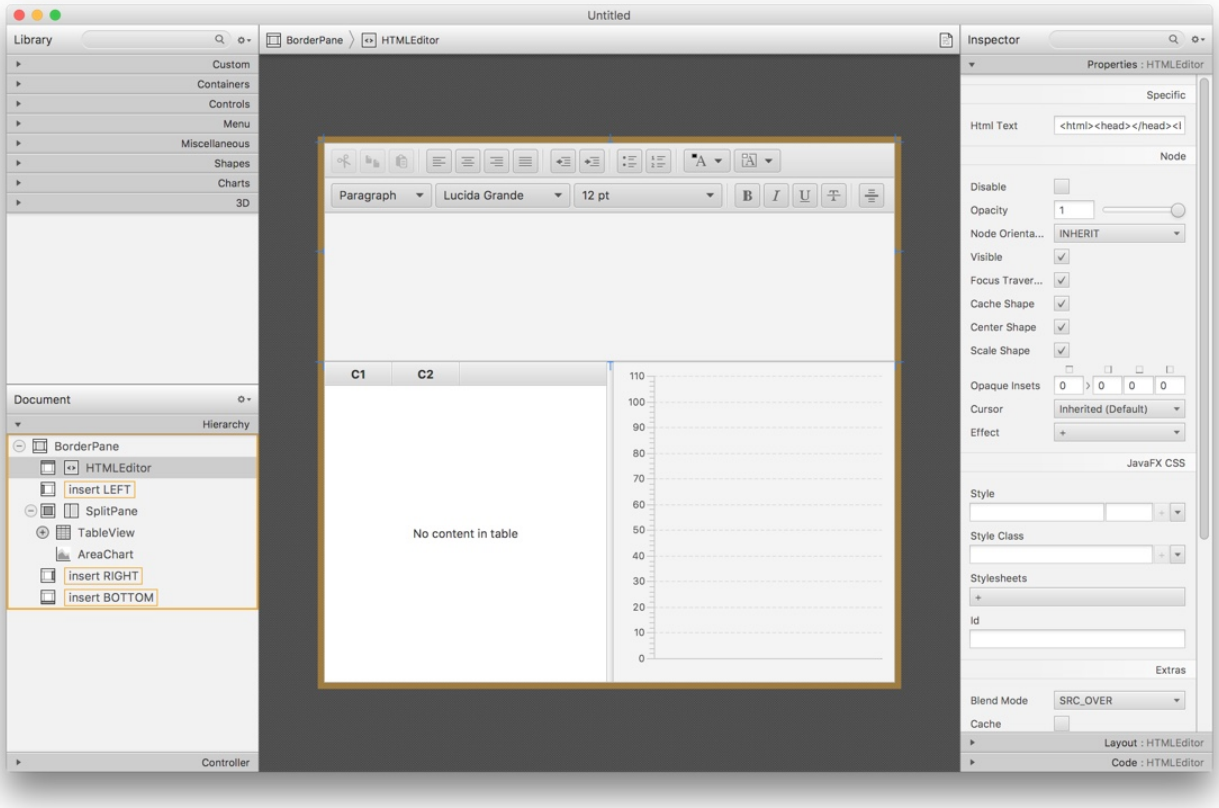

You are looking to replace your washer with a more energy efficient one
You know your washer has 800watts and costs 800 dollars
Replace Appliance DIY Tool
Click here to get started!
What are you replacing?
washer
Choose
Compare Appliances
Edit
Maytag
Custom
VS
Select
Use the edit to select appliances to compare.
Okay!
You Chose
Brand:
Model:
Cost:
Finish


Appliance Comparison Application - Not Dev
Customer
Welcome Page
Choose Appliance
Compare Page
Appliance List
Shopping Cart Screen
Stats Screen
Run Program
Click Start
Optional Import
Customer Select Appliance Type
Edit Appliance 1
Select Appliance Model 1
Edit Appliance 2
Select Appliance Model 2
Calculate Costs/Energy
Add Appliance to Shopping Cart
Add Another Appliance
Done Shopping
Optional Export

Front-End: JavaFX

The usage of FXML allowed for our team to quickly develop a non-intimidating, modern looking user-interface. Correspondingly, when working with FXML debugging was relatively painless considering that the syntax is relatable to XML or HTML.

The **source code for our user-interface** may be accessed through the directory: `\src\fxml\`

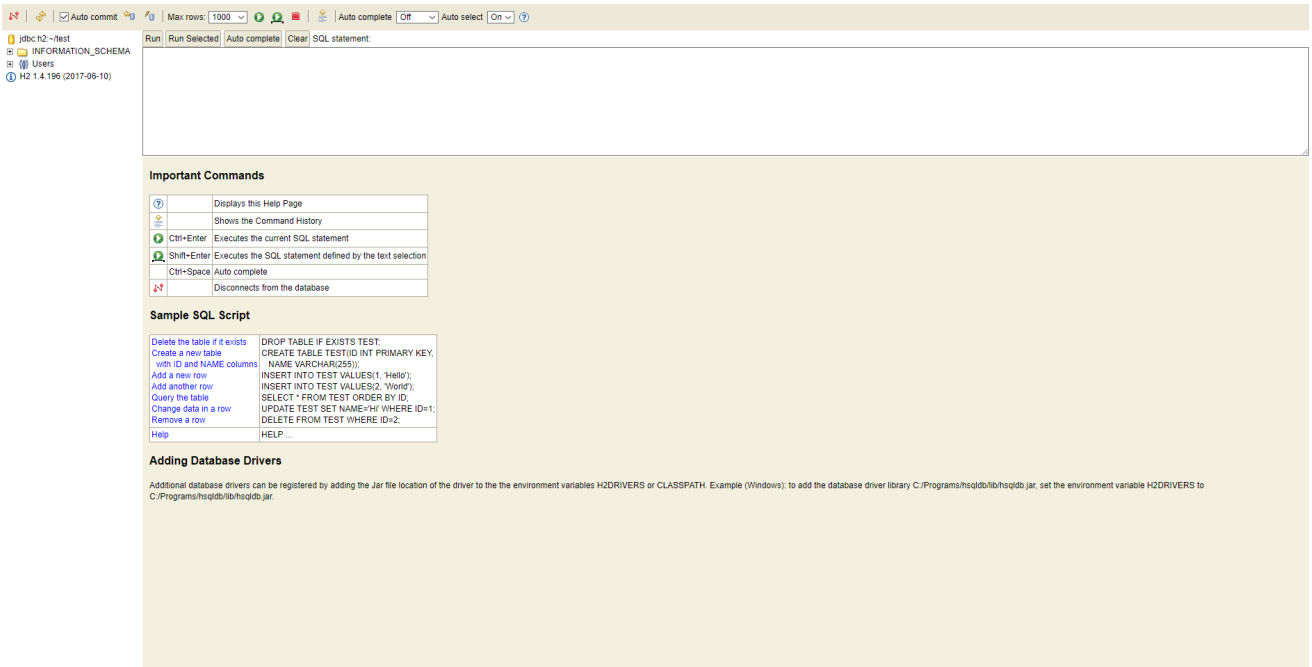


> **Example FXML Code:**

```
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHei
   <top>
      <ToolBar prefHeight="12.0" prefWidth="600.0" BorderPane.alignment="CENTER">
        <items>
         <Button mnemonicParsing="false" onAction="#backButtonClicked" text="Back" />
           <Text strokeType="OUTSIDE" strokeWidth="0.0" text="User Profile (Under Maintenance)" textAlig
        </items>
      </ToolBar>
   </top>
   <left>
      <VBox BorderPane.alignment="CENTER">
         <children>
            <ToolBar>
               <items>
                  <Label alignment="CENTER" prefHeight="32.0" prefWidth="404.0" text="Favorites" textAlig
                     <font>
                        <Font name="System Bold" size="22.0" />
                     </font>
                  </Label>
               </items>
            </ToolBar>
            <TableView fx:id="favTableView" prefHeight="329.0" prefWidth="410.0">
              <columns>
                  <TableColumn fx:id="favTypeCol" prefWidth="100.0" text="Type" />
                  <TableColumn fx:id="favModelCol" prefWidth="105.0" text="Model" />
                  <TableColumn fx:id="favBrandCol" prefWidth="105.0" text="Brand" />
                  <TableColumn fx:id="favEnergyCol" prefWidth="105.0" text="Energy Use" />
              </columns>
            </TableView>
            <Pane prefHeight="65.0" prefWidth="416.0">
               <children>
                  <Button fx:id="favRemoveButton" layoutX="139.0" layoutY="2.0" mnemonicParsing="false"
               </children>
            </Pane>
         </children>
      </VBox>
   </left>
   <center>
      <VBox alignment="CENTER" maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" minHe
         <children>
            <Text strokeType="OUTSIDE" strokeWidth="0.0" text="You are currently signed in as:" />
            <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Guest" />
            <Button disable="true" mnemonicParsing="false" text="Create Account">
               <VBox.margin>
                  <Insets top="50.0" />
               </VBox.margin>
```

Back-End: H2Database

H2Database is a **Java SQL database** to which we had opt'd to configure an in-memory database. More specifically, when the program is initially executed the data is loaded from the CSV files contained within the `./database/` directory. This is accomplished, through the `./src/driver/H2Database.java` which is used to interact with the DB server. This includes, connecting to the server, sending queries, parsing the CSVs and other various SQL and DBMS commands.


Auto commit
Max rows: 1000
Auto complete Off
Auto select On
jdbc:h2:~/test
INFORMATION_SCHEMA
Users
H2 1.4.196 (2017-06-10)
Run
Run Selected
Auto complete
Clear
SQL statement:
Important Commands
Displays this Help Page
Shows the Command History
Ctrl+Enter
Executes the current SQL statement
Shift+Enter
Executes the SQL statement defined by the text selection
Ctrl+Space
Auto complete
Disconnects from the database
Sample SQL Script
Delete the table if it exists
DROP TABLE IF EXISTS TEST;
Create a new table with ID and NAME columns
CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255));
Add a new row
INSERT INTO TEST VALUES(1, 'Hello');
Add another row
INSERT INTO TEST VALUES(2, 'World');
Query the table
SELECT * FROM TEST ORDER BY ID;
Change data in a row
UPDATE TEST SET NAME='Hi' WHERE ID=1;
Remove a row
DELETE FROM TEST WHERE ID=2;
Help
HELP ...
Adding Database Drivers
Additional database drivers can be registered by adding the Jar file location of the driver to the the environment variables H2DRIVERS or CLASSPATH. Example (Windows): to add the database driver library C:/Programs/hsqldb/lib/hsqldb.jar, set the environment variable H2DRIVERS to C:/Programs/hsqldb/lib/hsqldb.jar.