Project Manual: The Graphical Traveling Tool System (GTTS)

GTTS

By: 2131928, Anaïs Perron, 420-203-RE (Section 2)

INTRODUCTION

While being an enriching experience, traveling often presents us with a variety of challenges when it comes to management. How much will the journey cost? Which path will I take? What activities will I do? These are all concerns of travelers before starting an expedition. The lack of traveling management tools or the cost of travel agencies often compel travelers to embark on journeys for which they are not prepared. Consequently, what should be a relaxing experience transforms into a source of anxiety and concerns. Prudent preparation for any eventuality before commencing a new journey is a far preferable approach to ensure an optimized experience for the traveler. To help building these preparations, the "Graphical Traveling Tool System" is here.

DESCRIPTION

The Graphical Traveling Tool System is a GUI application that serves as a tool for travelers to manage their expenses and activities. In it, the user will be able to compare the different choices of expenses that they are being offered during a journey as well as tracking their plans. The user will be able to add lists of expenses for each journey that they choose to add to their profile, which can be accessed when inputting a username and a password (the user's account will be recorded in a serialized file as soon as the user creates the account). The idea of this program is to allow travelers and tourists to choose the most optimal way to spend their money during a journey without the need for a traveling agency or complex thinking.

FUNCTIONALITY

The Graphical Traveling Tool System is intended to be simple and easy to use, while still using complex mathematical ideas. Each journey will have a list of destinations, and each destination will have a list of expenses and a TODO list for the user to carefully plan their expedition. The application will also include a simulation that uses the mathematical concept of **weighted graphs** to find the shortest path and lowest price of the journey that the user will choose to take, as well as display all the other possibilities to allow the user to carefully consider their options. For example, the lowest price to get from Los Angeles to Boston would be 2300$, (figure 1), and the lowest distance would be 2596 km (figure 2). In this specific case, both the shortest distance and the lowest price would be by taking the path: Los Angeles – Chicago – Boston, but the price and the distance are not always related to one another. For example, taking a taxi might cover less distance while still being more expansive than taking the bus.

Figure 1

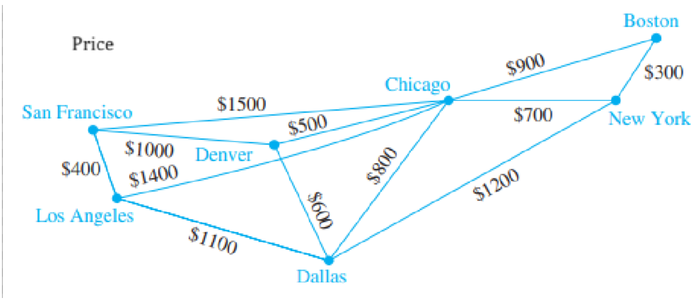
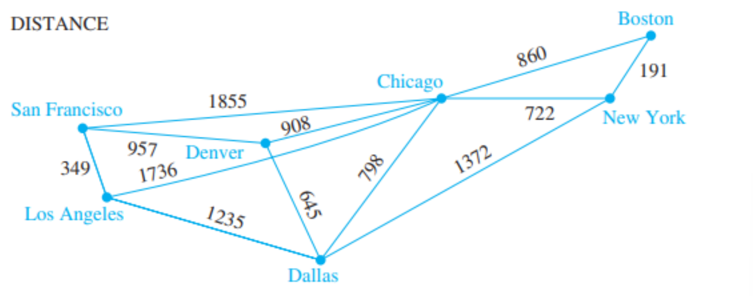


Figure 2



The user will also be able to delete, modify, and add elements to their account using a graphical environment.

FEATURES

The Graphical Traveling Tool System enhances the user's satisfaction by containing a variety of different features, allowing them to manage their journey with ease. These features are:

User-Friendly Interface: The application is designed to ensure efficiency, simplicity, and optimized visual representation to the user. They will be able to manage their expenses and activities with ease, without being overwhelmed by complexity.

User Profile Creation: The application enables the user to create an account, which can be accessed when inputting a username and a password (the user's account will be recorded in a serialized file as soon as the user creates the account). This account will contain the user's personalized planning.

Serialized File: A serialized file's purpose is to record the state of an object into a form that can be transported. In simple terms, the serialized file is a way for the application to access the data from the accounts by deserializing the file. The file is not accessible to the user, and it is unintelligible even to the programmer.

Comprehensive Itinerary Simulation: The application will include a simulation that uses the mathematical concept of weighted graphs to find the shortest path and lowest price of the journey that the user will choose to take, as well as display all the other possibilities to allow the user to carefully consider their options. The price of the itinerary can later be added to the list of expenses of the journey.

List of Journeys: The user will be able to add or modify a list of journeys into their account. Each journey will have a list of destination and each destination will incorporate personalized features such as the following:

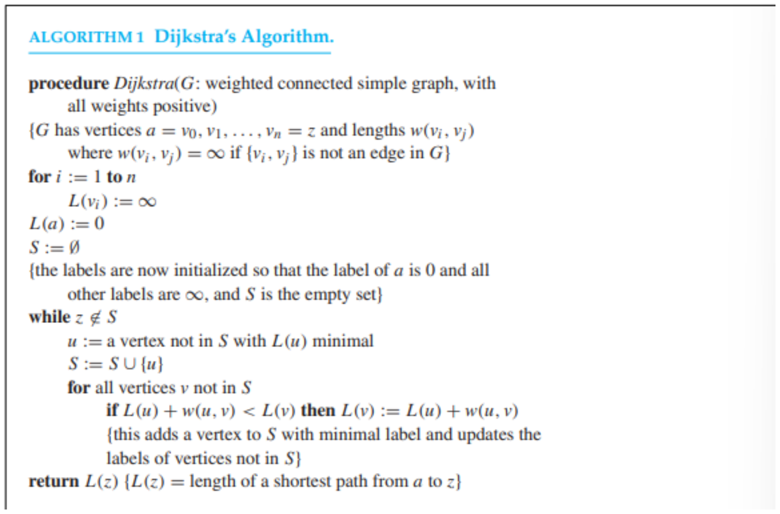
> **-Expenses and Budget Management Tool**: For each journey, the user will be able to set a budget and allocate expenses to track their spendings. The sum of all their spendings in a journey will be displayed and the application will let the user know if their spendings exceed their budget.

WHO IS THE APPLICATION FOR?

The Graphical Travel Tool System is designed for individuals who wish to travel and seek a tool to efficiently manage their travel expenses as well as their activities. It is intended to enhance a traveler's experience by providing them greater financial tranquility and activity management. Whilst the primary focus of the application is directed towards tourists, its utility can be extended to any travel-oriented scenarios, such as business trips or family road trips.

ADDITIONAL INFORMATION

As stated, the application will use the mathematical concept of weighted graphs to build a simulation that finds the shortest path and the lowest price of the user's journey. Weighted graphs are weights assigned to their edges. To build an application that utilizes this concept, I will use ***Dijkstra's Algorithm.*** Here is a summary of how the algorithm works:



ALGORITHM 1 Dijkstra's Algorithm.

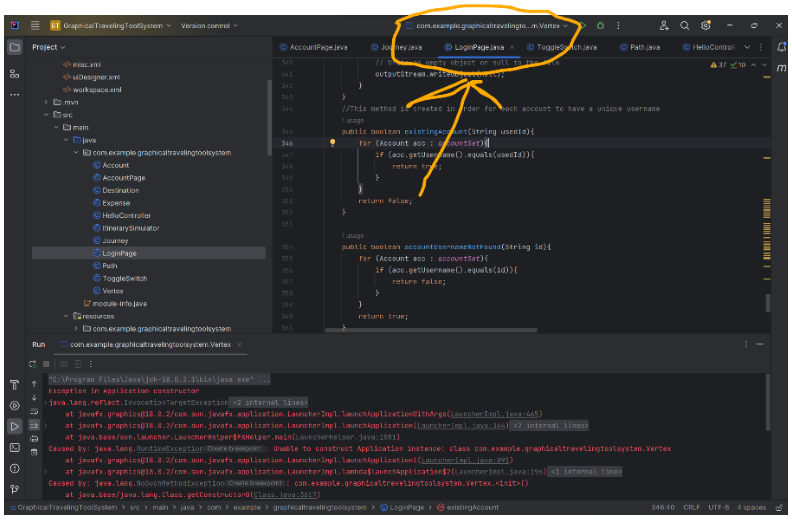
```
procedure Dijkstra(G: weighted connected simple graph, with
    all weights positive)
{G has vertices $a = v_0, v_1, \ldots, v_n = z$ and lengths $w(v_i, v_j)$
    where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}
for $i := 1$ to $n$
    $L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of $a$ is 0 and all
    other labels are $\infty$, and $S$ is the empty set}
while $z \notin S$
    $u :=$ a vertex not in $S$ with $L(u)$ minimal
    $S := S \cup \{u\}$
    for all vertices $v$ not in $S$
        if $L(u) + w(u, v) < L(v)$ then $L(v) := L(u) + w(u, v)$
        {this adds a vertex to $S$ with minimal label and updates the
        labels of vertices not in $S$}
return $L(z)$ {$L(z) =$ length of a shortest path from $a$ to $z$}
```

Source: "Discrete Mathematics and Its Applications" (Rosen, 2011, p.712)

SOME THINGS TO KNOW BEFORE RUNNING THE APPLICATION

1- The application contains three classes with the start(Stage primaryStage) method. However, **only the ItinerarySimulator class and the LoginPage class** can be run without throwing an error. There will be an InvocationTargetException error thrown if the user tries to run the application via the **AccountPage class**, since there needs to be a logged in user before accessing the page.

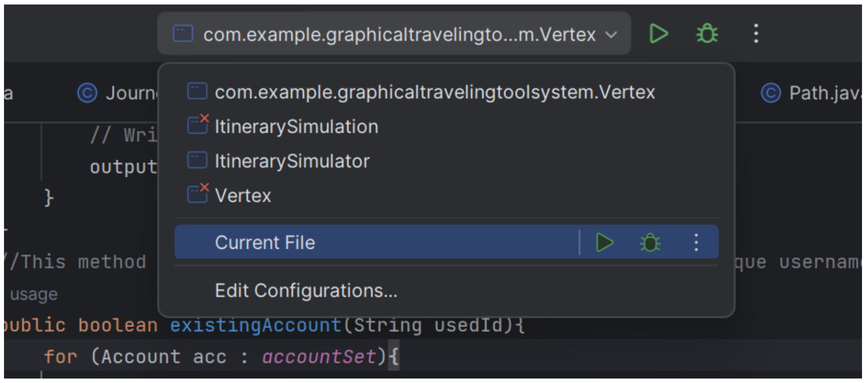
2- If the application throws an error when ran, it might be because the application is running via the **com.example.graphicaltravelingtoolsystem.Vertex by default even if the current file opened is the right one.**



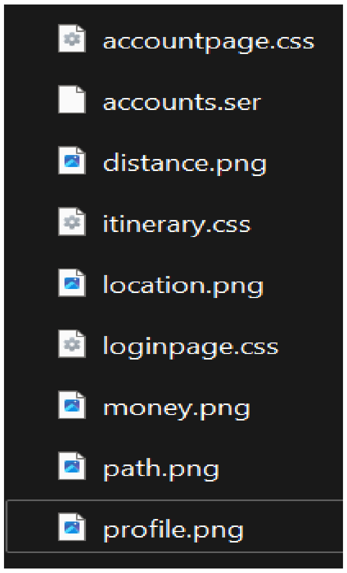
```java
        // Write an empty object or null to the file
        outputStream.writeObject(null);
    }
}
//This method is created in order for each account to have a unique username
public boolean existingAccount(String usedId){
    for (Account acc : accountSet){
        if (acc.getUsername().equals(usedId)){
            return true;
        }
    }
    return false;
}

public boolean accountUsernameNotFound(String id){
    for (Account acc : accountSet){
        if (acc.getUsername().equals(id)){
            return false;
        }
    }
    return true;
}
```

```
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
Exception in Application constructor
java.lang.reflect.InvocationTargetException <2 internal lines>
    at javafx.graphics@18.0.2/com.sun.javafx.application.LauncherImpl.launchApplicationWithArgs(LauncherImpl.java:465)
    at javafx.graphics@18.0.2/com.sun.javafx.application.LauncherImpl.launchApplication(LauncherImpl.java:364) <2 internal lines>
    at java.base/sun.launcher.LauncherHelper$FXHelper.main(LauncherHelper.java:1081)
Caused by: java.lang.RuntimeException Create breakpoint : Unable to construct Application instance: class com.example.graphicaltravelingtoolsystem.Vertex
    at javafx.graphics@18.0.2/com.sun.javafx.application.LauncherImpl.launchApplication1(LauncherImpl.java:891)
    at javafx.graphics@18.0.2/com.sun.javafx.application.LauncherImpl.lambda$launchApplication$2(LauncherImpl.java:196) <1 internal line>
Caused by: java.lang.NoSuchMethodException Create breakpoint : com.example.graphicaltravelingtoolsystem.Vertex.<init>()
    at java.base/java.lang.Class.getConstructor0(Class.java:3617)
```

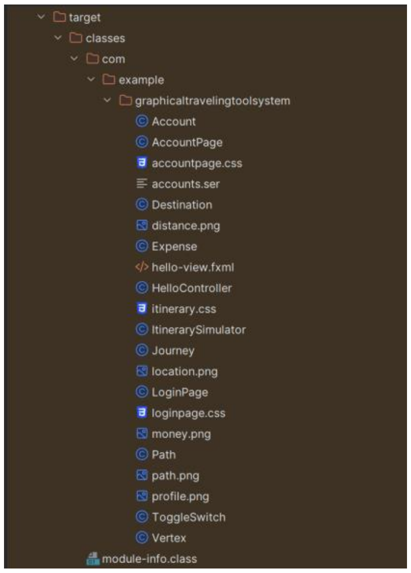
To remediate this situation, one can simply press on **Current File.** The right file will run and no InvocationTargetException will be met.


com.example.graphicaltravelingto...m.Vertex
com.example.graphicaltravelingtoolsystem.Vertex
ItinerarySimulation
ItinerarySimulator
Vertex
Current File
Edit Configurations...

3- The application must contain the following files and **must be placed int the “target” folder of the project** (if not, the application will not be able to find the files)**.** These files are contained in the IMPORTANT folder of the project zip file in case they do not appear when opening the project.

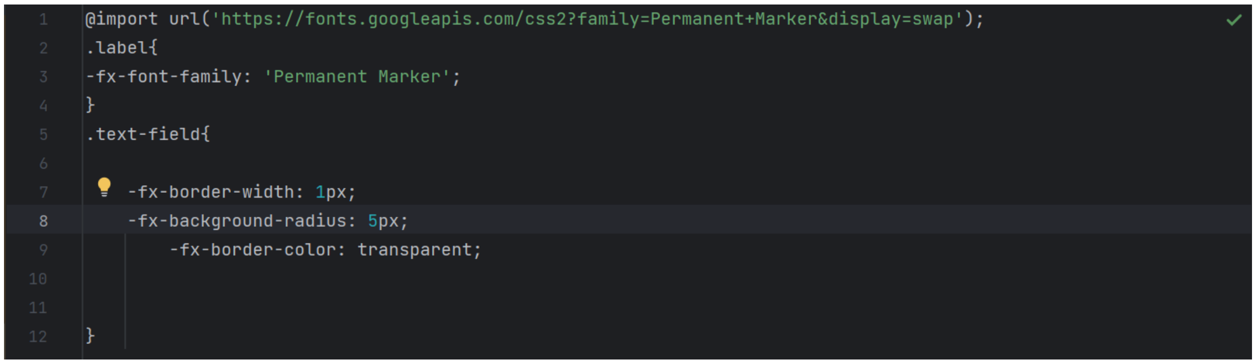

accountpage.css
accounts.ser
distance.png
itinerary.css
location.png
loginpage.css
money.png
path.png
profile.png

The files contains: 3 different css files for each each class that extend Application (itinerary.css is to style the ItinerarySimulator class, accountpage.css is to style the AccountPage class, and loginpage.css is to style the LoginPage class), a serialized file that contains the existent accounts, and numerous png images that have been inserted in AccountPage and ItinerarySimulator.


```
target
  classes
    com
      example
        graphicaltravelingtoolsystem
          Account
          AccountPage
          accountpage.css
          accounts.ser
          Destination
          distance.png
          Expense
          hello-view.fxml
          HelloController
          itinerary.css
          ItinerarySimulator
          Journey
          location.png
          LoginPage
          loginpage.css
          money.png
          Path
          path.png
          profile.png
          ToggleSwitch
          Vertex
    module-info.class
```

CSS FILES

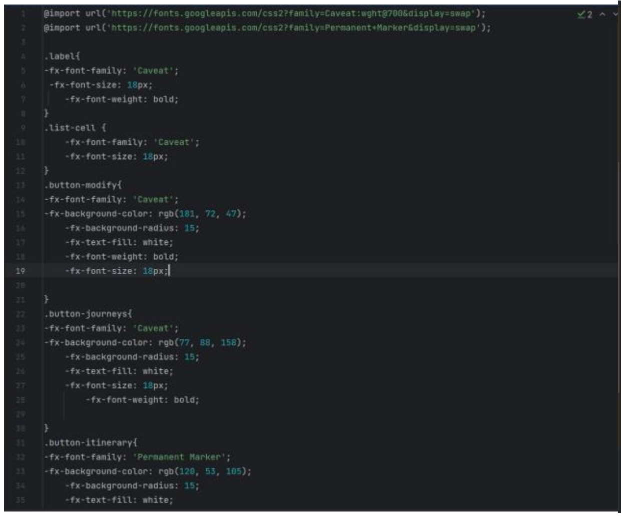
The first css file styles some items of the LoginPage class. It imports the "Permanent Marker" google font.


```css
@import url('https://fonts.googleapis.com/css2?family=Permanent+Marker&display=swap');
.label{
-fx-font-family: 'Permanent Marker';
}
.text-field{

    -fx-border-width: 1px;
    -fx-background-radius: 5px;
        -fx-border-color: transparent;


}
```

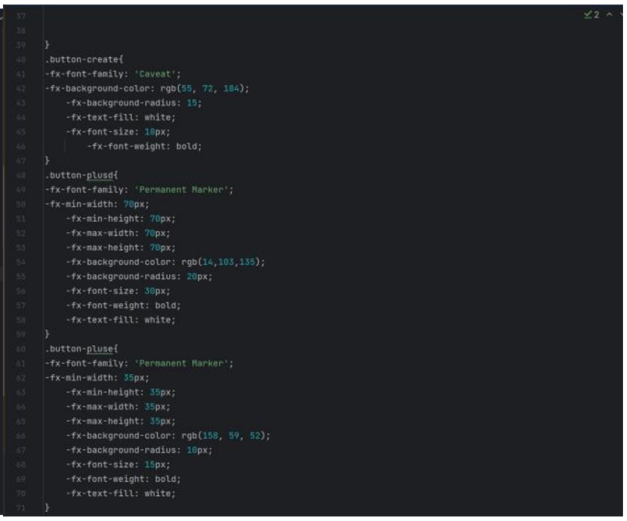
.label: sets all labels to have a "Permanent Marker" google font.

.text-field: sets the TextField and the PasswordField to have thin transparent borders with rounded edges.

The second css file styles some items of the AccountPage class. It imports the "Caveat" and "Permanent Marker" google fonts.



```css
@import url('https://fonts.googleapis.com/css2?family=Caveat:wght@700&display=swap');
@import url('https://fonts.googleapis.com/css2?family=Permanent+Marker&display=swap');

.label{
-fx-font-family: 'Caveat';
 -fx-font-size: 18px;
    -fx-font-weight: bold;
}
.list-cell {
    -fx-font-family: 'Caveat';
    -fx-font-size: 18px;
}
.button-modify{
-fx-font-family: 'Caveat';
-fx-background-color: rgb(181, 72, 47);
    -fx-background-radius: 15;
    -fx-text-fill: white;
    -fx-font-weight: bold;
    -fx-font-size: 18px;

}
.button-journeys{
-fx-font-family: 'Caveat';
-fx-background-color: rgb(77, 88, 158);
    -fx-background-radius: 15;
    -fx-text-fill: white;
    -fx-font-size: 18px;
        -fx-font-weight: bold;

}
.button-itinerary{
-fx-font-family: 'Permanent Marker';
-fx-background-color: rgb(120, 53, 105);
    -fx-background-radius: 15;
    -fx-text-fill: white;
```



```css


}
.button-create{
-fx-font-family: 'Caveat';
-fx-background-color: rgb(55, 72, 184);
    -fx-background-radius: 15;
    -fx-text-fill: white;
    -fx-font-size: 18px;
        -fx-font-weight: bold;
}
.button-plusd{
-fx-font-family: 'Permanent Marker';
-fx-min-width: 70px;
    -fx-min-height: 70px;
    -fx-max-width: 70px;
    -fx-max-height: 70px;
    -fx-background-color: rgb(14,103,135);
    -fx-background-radius: 20px;
    -fx-font-size: 30px;
    -fx-font-weight: bold;
    -fx-text-fill: white;
}
.button-pluse{
-fx-font-family: 'Permanent Marker';
-fx-min-width: 35px;
    -fx-min-height: 35px;
    -fx-max-width: 35px;
    -fx-max-height: 35px;
    -fx-background-color: rgb(158, 59, 52);
    -fx-background-radius: 10px;
    -fx-font-size: 15px;
    -fx-font-weight: bold;
    -fx-text-fill: white;
}
```

.label: Sets all label sizes to 18px, their font to the imported google font “Caveat”, and makes them bold so that they can be a little more noticeable.

.list-cell: sets the text values inside the ListView to the “Caveat” font and to a size of 18px.

.button-modify: sets the “modify” button to be a dark shade of red, with white bolded text, with a “Caveat font” and a radius of 15 to make the edges more round.

.button-journeys: sets the buttons with the journey names to have a dark blue color with white bolded text, with a “Caveat font”, and a radius of 15 to make the edges more round.

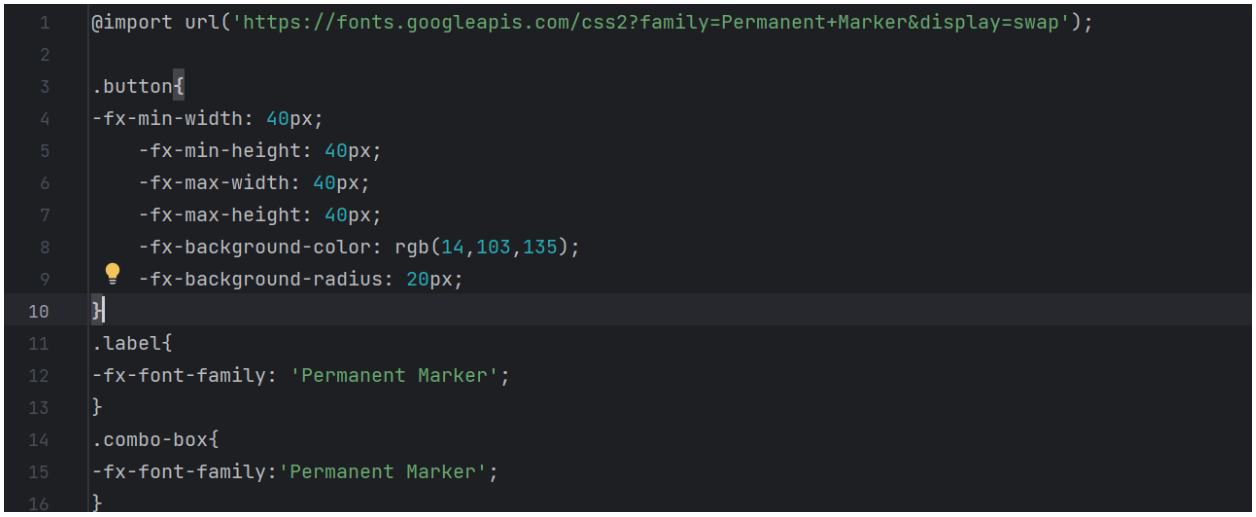
.button-itinerary: sets the Itinerary Simulator button to have a dark purple color, with a “Permanent Marker”, and a radius of 15 to make the edges more round.

.button-create: sets the +Create Journey button to have a dark shade of blue, , with white bolded text, with a “Caveat font” and a radius of 15 to make the edges more round.

.button-plusd: sets the style of the button meant to create a destination to be a dark shade of blue, with a font bolded white of “Permanent Marker” and 30px, and rounded square shape.

.button-pluse: sets the style of the button meant to create an expanse to be a dark shade of red, with a font bolded white of “Permanent Marker” and 15px (because it is half the size of the “add destination” button), and rounded square shape.

The third css file styles some items of the ItinerarySimulator class. It imports the “Permanent Marker” google font.



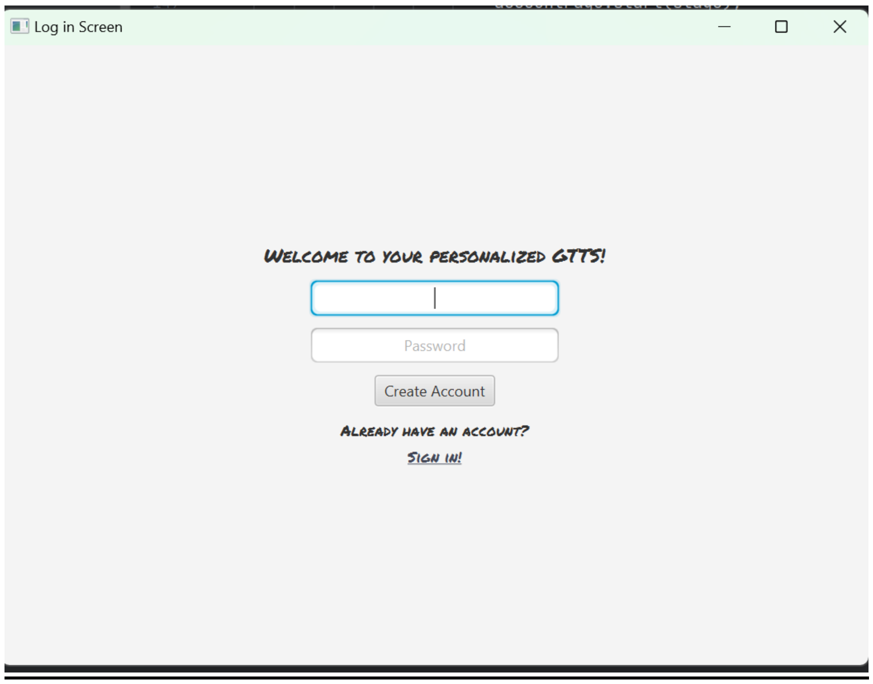
```css
@import url('https://fonts.googleapis.com/css2?family=Permanent+Marker&display=swap');

.button{
-fx-min-width: 40px;
    -fx-min-height: 40px;
    -fx-max-width: 40px;
    -fx-max-height: 40px;
    -fx-background-color: rgb(14,103,135);
    -fx-background-radius: 20px;
}
.label{
-fx-font-family: 'Permanent Marker';
}
.combo-box{
-fx-font-family:'Permanent Marker';
}
```

.button: sets the button to be a dark blue circle

.label: sets all labels to have the “Permanent Marker” font

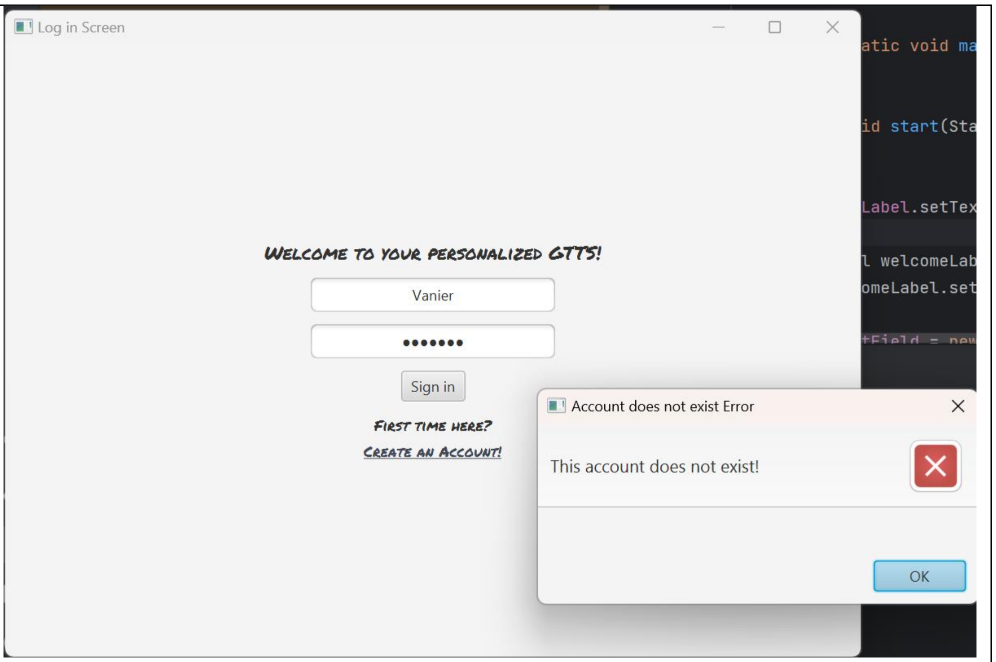
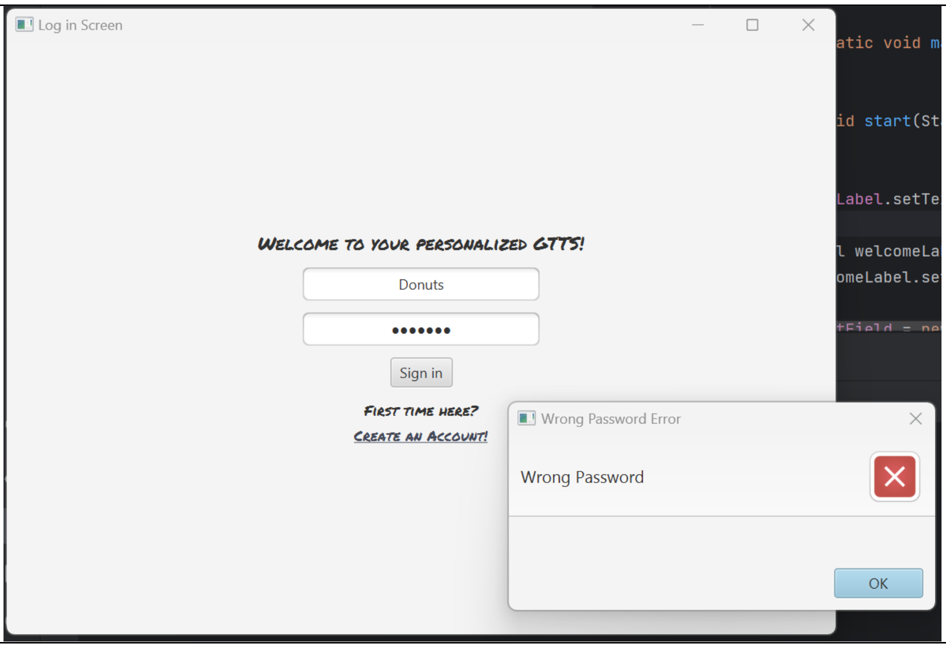
.combo-box: sets the text values of the combo boxes to have the “Permanent Marker” font

Login Page



The Log In Page is used to access or to create an account. The class contains the main() method and should be the first stage to be seen when running the application. The first text field is meant for the user to type his or her username and the second field is a PasswordField meant for the user to input his or her password related to the username. An account can be accessed only if the inputted username and the password match an account in the serialized file "accounts.ser**".**

EXCEPTION HANDLING IN THE LOGIN PAGE

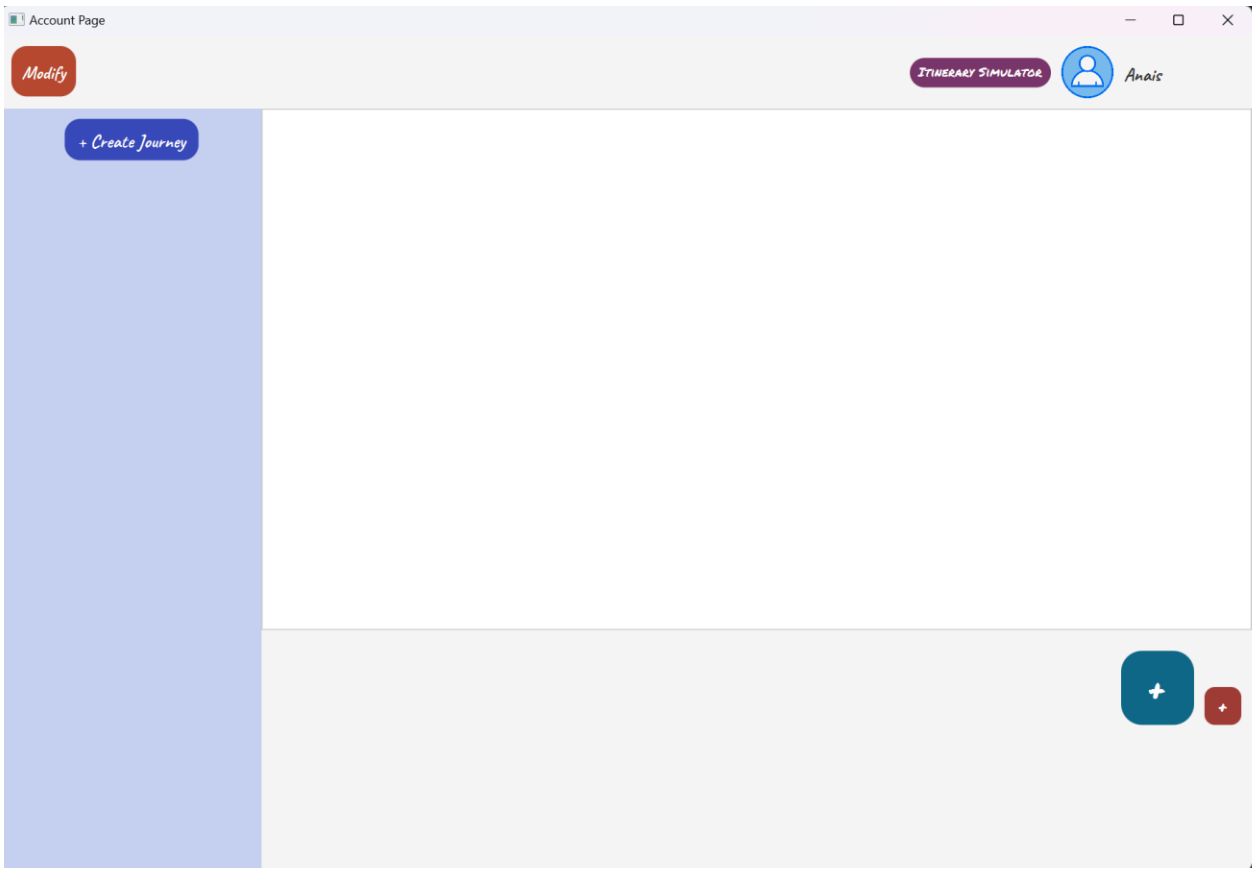
If the user tries to access an account that does not exist in the file, an Alert of type "Error" appears.	 Log in Screen WELCOME TO YOUR PERSONALIZED GTTS! Vanier ••••••• Sign in FIRST TIME HERE? CREATE AN ACCOUNT! Account does not exist Error This account does not exist! OK
If the user tries to sign in to an existing account but inputs the wrong password, an Alert of type "Error" appears.	 Log in Screen WELCOME TO YOUR PERSONALIZED GTTS! Donuts ••••••• Sign in FIRST TIME HERE? CREATE AN ACCOUNT! Wrong Password Error Wrong Password OK

Account Page

Once the user has either created or accessed their account, their personalized features are displayed on a new stage.

Note: The Application cannot be started via the account page, it would throw an InvocationTargetException.

Here is the default page visual for new users:



1_ The **LEFT** pane of the BorderPane is meant to contain the set of journeys of the account. They are in the form of **buttons.**

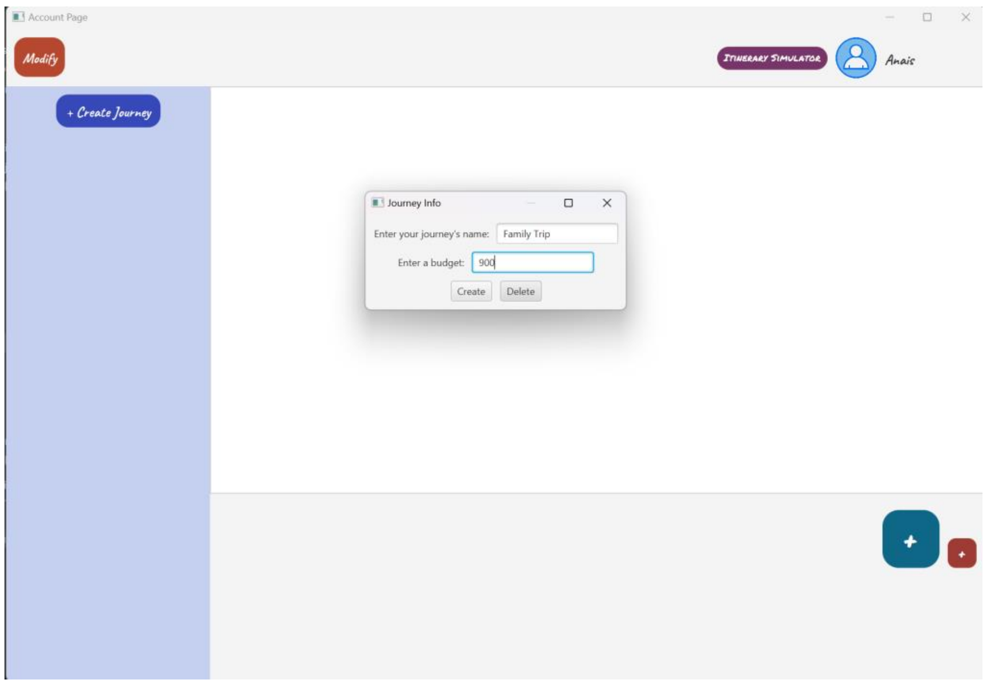
2_The **RIGHT PART OF THE TOP** pane of the BorderPane contains a button that directly calls the start(Stage primaryStage) of the ItinerarySimulator class, which I will refer to later on. The pane also contains a simple ImageView of a profile icon and the username of the account.

4_The **LEFT PART OF THE TOP** pane contains the "modify" button. Once clicked, the user can modify journeys, destinations, and expanses by pressing on the item. In the "modify" option, the user cannot create anything. Once the "modify" button is clicked, the button's text will be set to "Create", and the user will be able to create

3_The **CENTER** pane of the BorderPane contains a ListView representation of what is inside a journey. Since no journey is created when an account is, the ListView is empty and looks like a simple white rectangle. If the journey selected does not have any destination inside its arraylist of destinations, the ListView will be empty as shown in the previous figure. The center pane also contains two buttons. The big red button is used to add a destination, and the little blue one is meant to add an expense. A journey has an ArrayList of destinations, and destinations have ArrayList of expenses. An expense cannot be added without a destination to link to it.

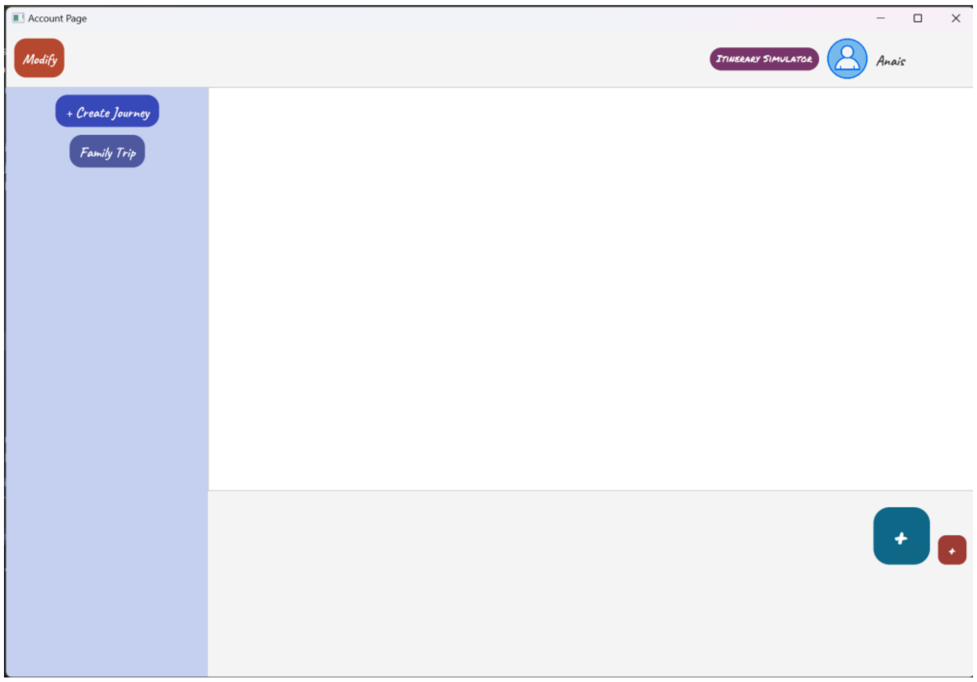
TO ADD ITEMS

Let's now look at what happens when we add a Journey in the account.

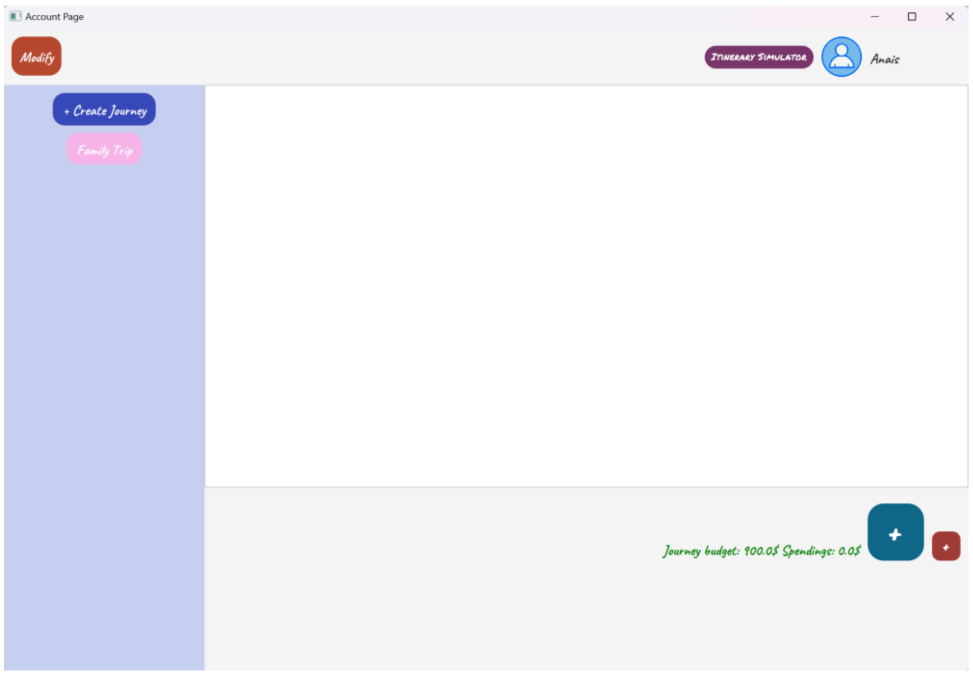


Enter your journey's name: Family Trip
Enter a budget: 900

When pressing the "+ Create Journey" button, a new Stage appears. This stage will add a journey with a specified name.



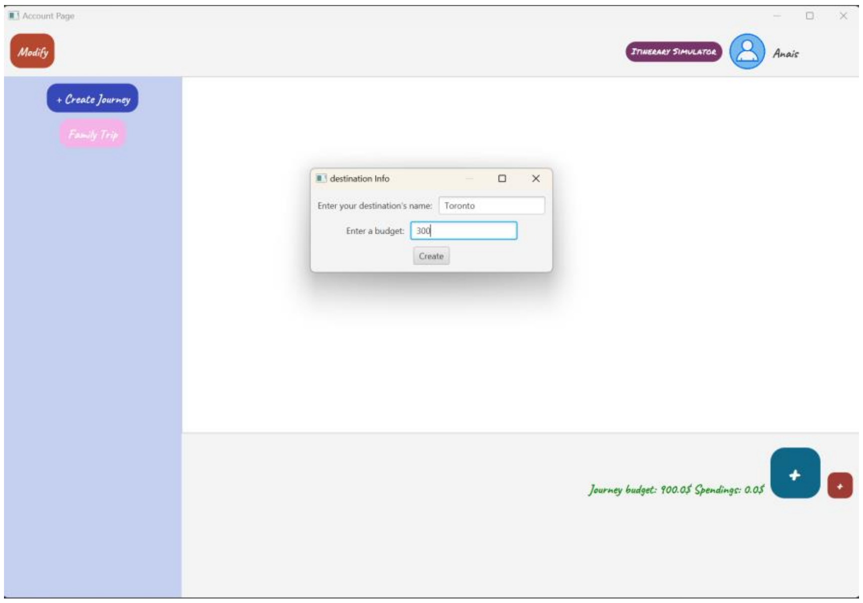
After creating the Journey, a new Button appears in the left pane of the BorderPane. The userData of the button is set to the journey object, so it can be referenced later on.



When the user presses on the journey, the button turns pink. This is to let the user know which journey has its information displayed. **If the journey budget exceeds the spendings, the label next to the plus buttons is green.**

When clicking on the journey button, the user can now add destinations and expenses.

Let's look at what happens when we add a destination.

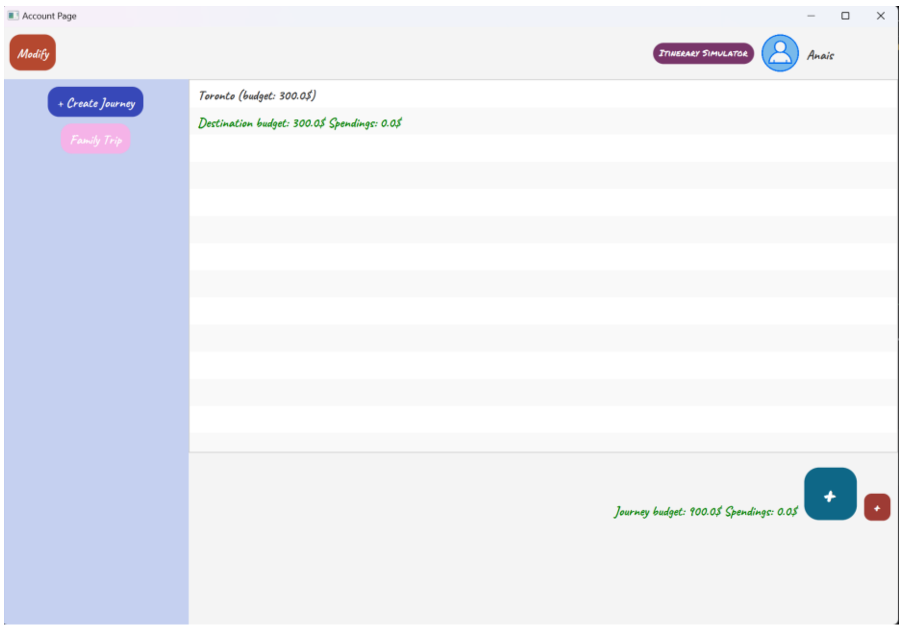


Enter your destination's name: Toronto

Enter a budget: 300

Journey budget: 900.0$ Spendings: 0.0$

When clicking on the blue button to add a destination, approximately the same scene as the one to add a journey appears. Only, the "delete" button is not present, because we are not creating a node out of a destination. We are just adding it to the journey's list of destinations.



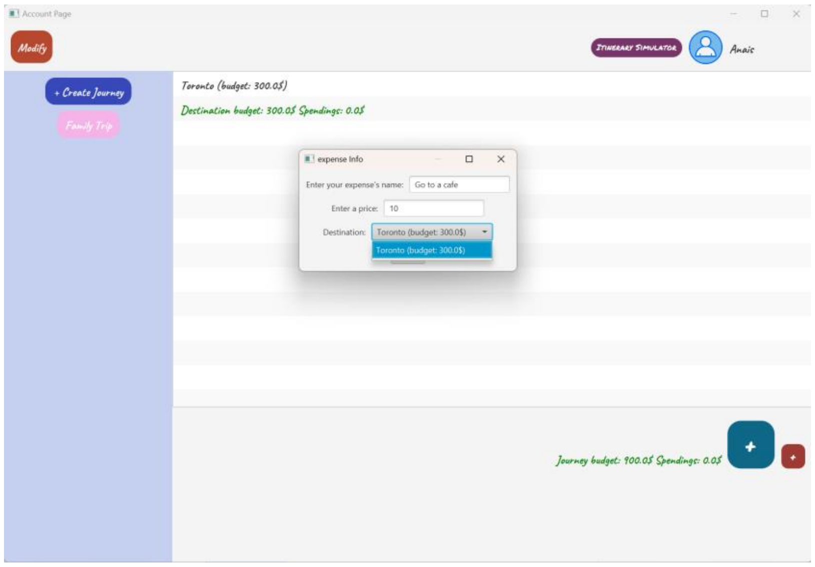
Toronto (budget: 300.0$)

Destination budget: 300.0$ Spendings: 0.0$

Journey budget: 900.0$ Spendings: 0.0$

As the user can see, the destination object has been added to the ListView as well as a Label that tracks the spendings and the destination's budget.

<u>Let's now look at what happens when we add an expense…</u>



Toronto (budget: 300.0$)

Destination budget: 300.0$ Spendings: 0.0$

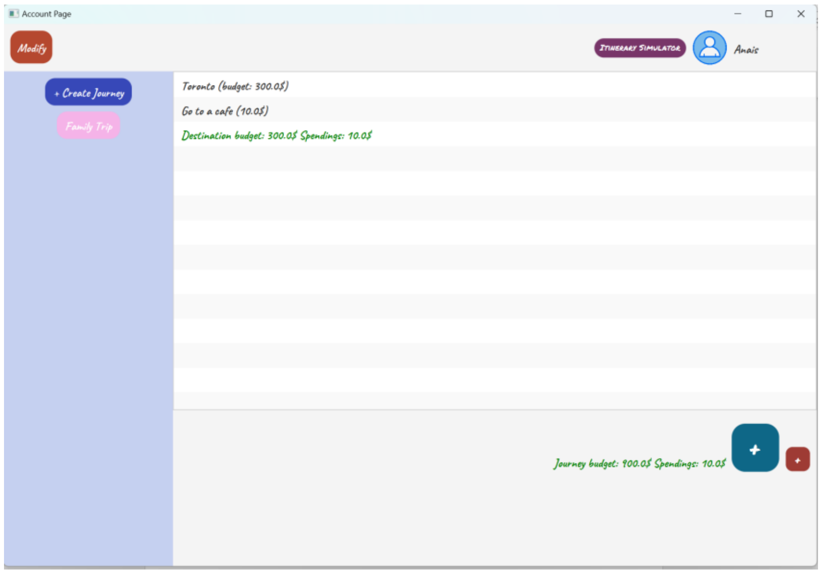
Enter your expense's name: Go to a cafe

Enter a price: 10

Destination: Toronto (budget: 300.0$)

Journey budget: 900.0$ Spendings: 0.0$

Here, the stage is roughly the same as the other options. However, we need to choose a destination in which we will add the expense.



Toronto (budget: 300.0$)

Go to a cafe (10.0$)

Destination budget: 300.0$ Spendings: 10.0$

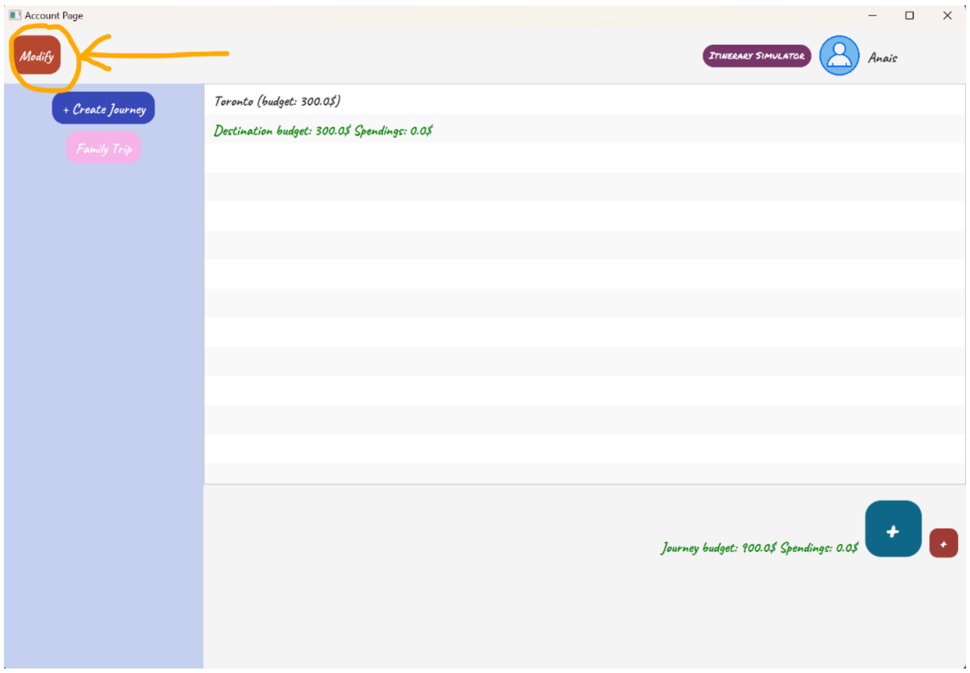
Journey budget: 900.0$ Spendings: 10.0$

Here, we added the expense to the destination. The total spending and the destination's spending is now updated.

Note: A journey cannot be added without an account, a destination cannot be added without a journey, and an expanse cannot be added without a destination.

TO MODIFY ITEMS

To modify any item, it is essential to click on the “modify” button first.

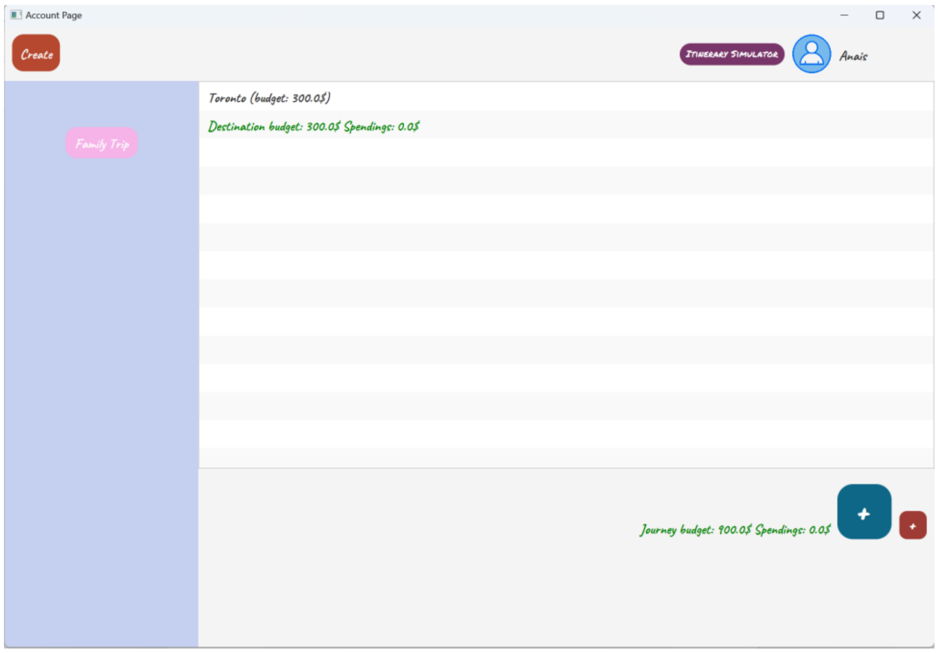


Toronto (budget: 300.0$)

Destination budget: 300.0$ Spendings: 0.0$

Journey budget: 900.0$ Spendings: 0.0$

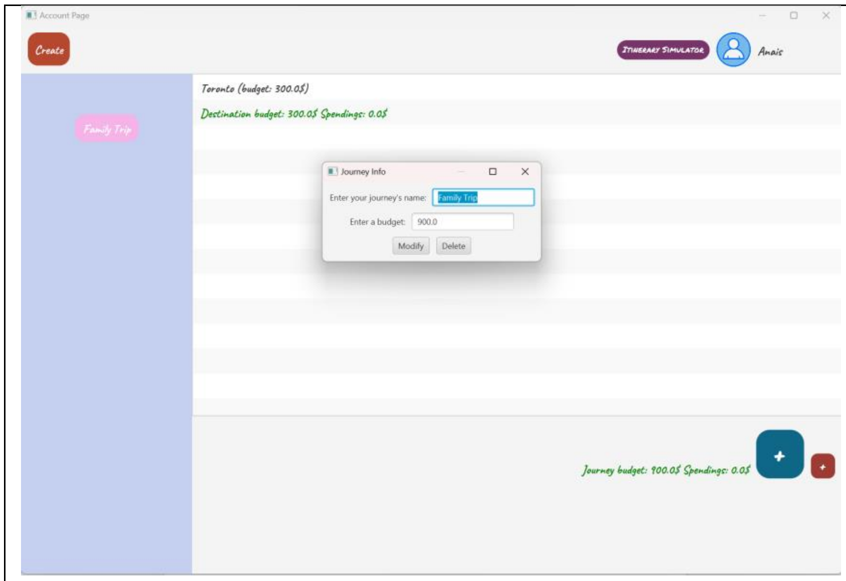
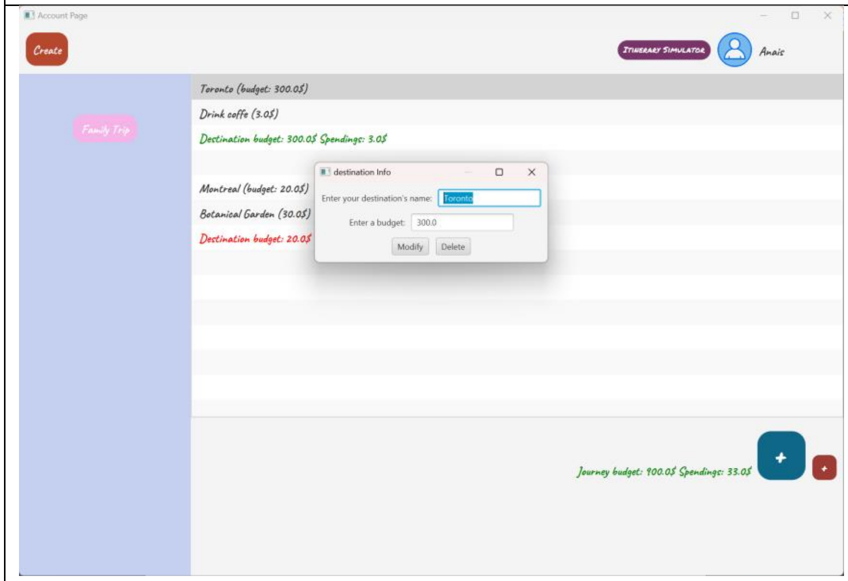
Once the button is clicked, the text is set to “create”. So, if clicked again, the user can go back to adding items rather than modifying them. Also, the “+ Create Journey” is set to invisible because it is useless in the “modifying” option.

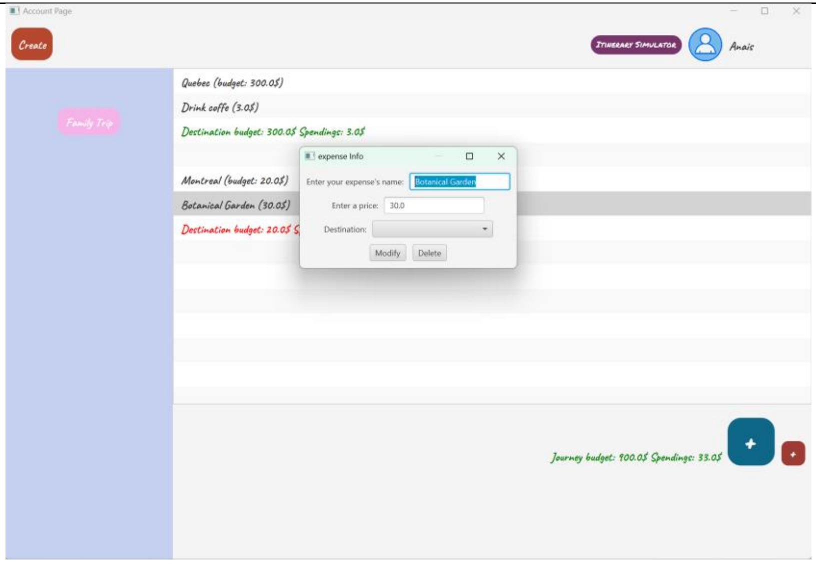


Toronto (budget: 300.0$)

Destination budget: 300.0$ Spendings: 0.0$

Journey budget: 900.0$ Spendings: 0.0$

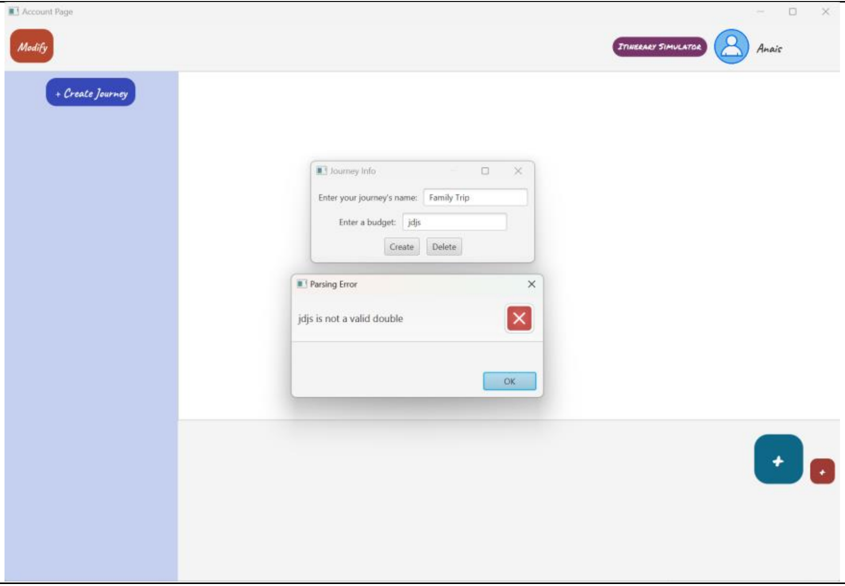
 Account Page Create Itinerary Simulator Anais Family Trip Journey Info Modify Delete Toronto (budget: 300.0$) Destination budget: 300.0$ Spendings: 0.0$ Enter your journey's name: Family Trip Enter a budget: 900.0 Journey budget: 900.0$ Spendings: 0.0$	If the user clicks on a journey, the journey's properties are displayed. The user can then change the journey's properties. If the user presses on "delete", the button will be deleted as well as the object.
 Account Page Create Itinerary Simulator Anais Family Trip destination Info Modify Delete Toronto (budget: 300.0$) Drink coffe (3.0$) Destination budget: 300.0$ Spendings: 3.0$ Montreal (budget: 20.0$) Botanical Garden (30.0$) Destination budget: 20.0$ Enter your destination's name: Toronto Enter a budget: 300.0 Journey budget: 900.0$ Spendings: 33.0$	If the user clicks on a destination, the destination's properties are displayed. The user can then change the destination's properties. If the user presses on "delete", the destination will be deleted as well as the object.

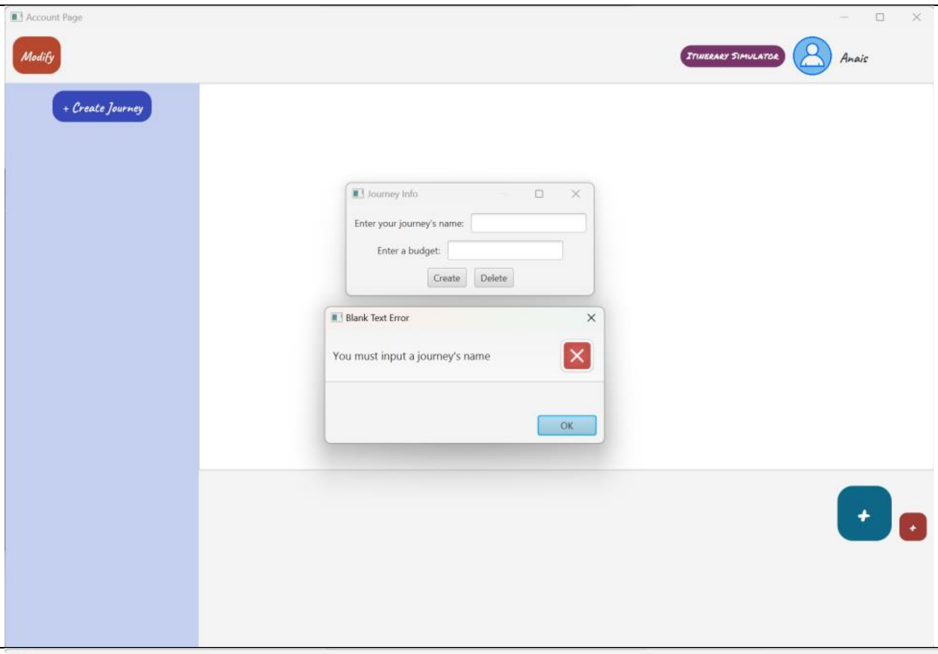
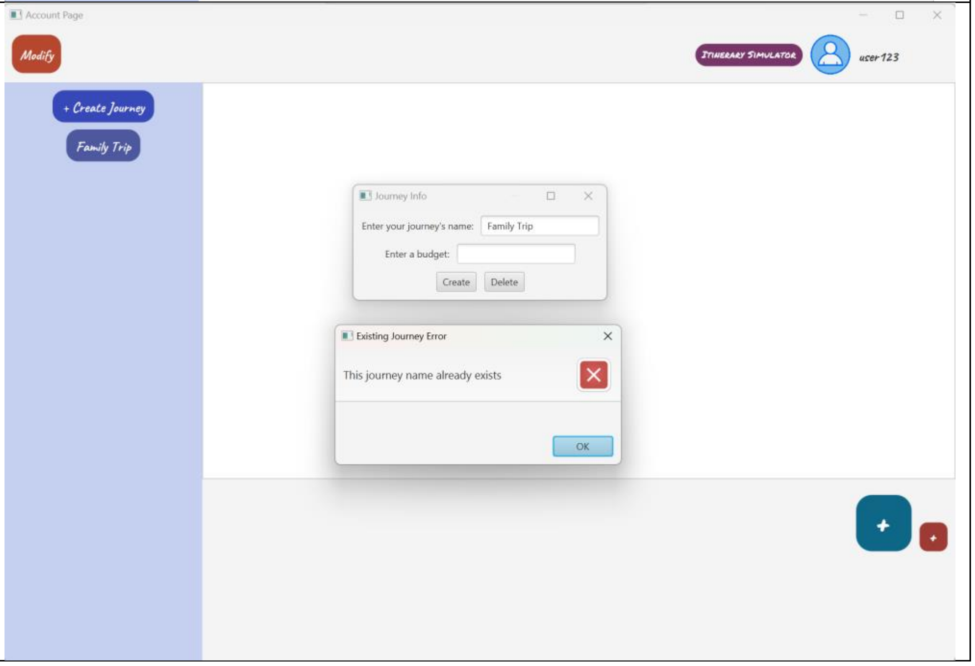
 Account Page Create Itinerary Simulator Anais Family Trip Quebec (budget: 300.0$) Drink coffe (3.0$) Destination budget: 300.0$ Spendings: 3.0$ Montreal (budget: 20.0$) Botanical Garden (30.0$) Destination budget: 20.0$ S expense Info Enter your expense's name: Botanical Garden Enter a price: 30.0 Destination: Modify Delete Journey budget: 900.0$ Spendings: 33.0$	If the user clicks on an expanse, the expanse's properties are displayed. The user can then change the expanse's properties. If the user presses on "delete", the expanse object will be deleted.

Note: The exception handling is the same as to create an item.

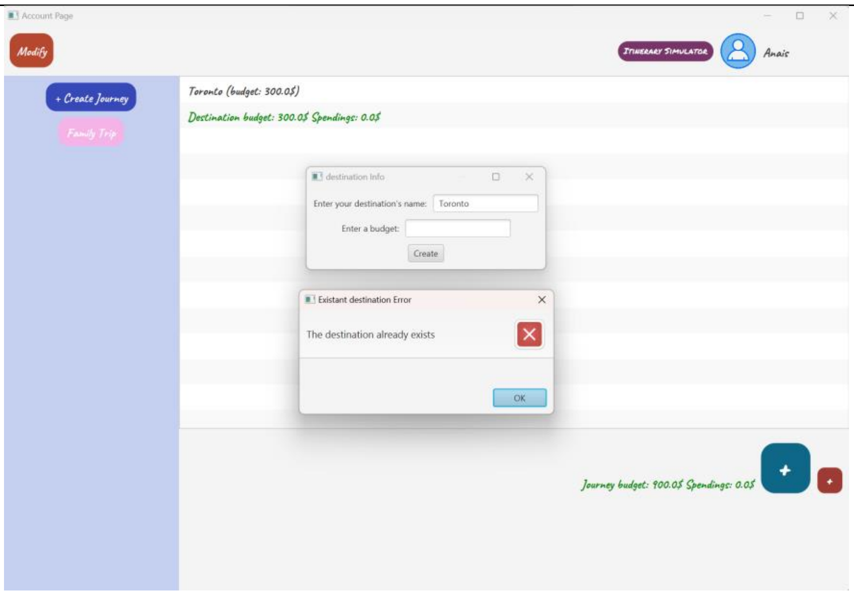
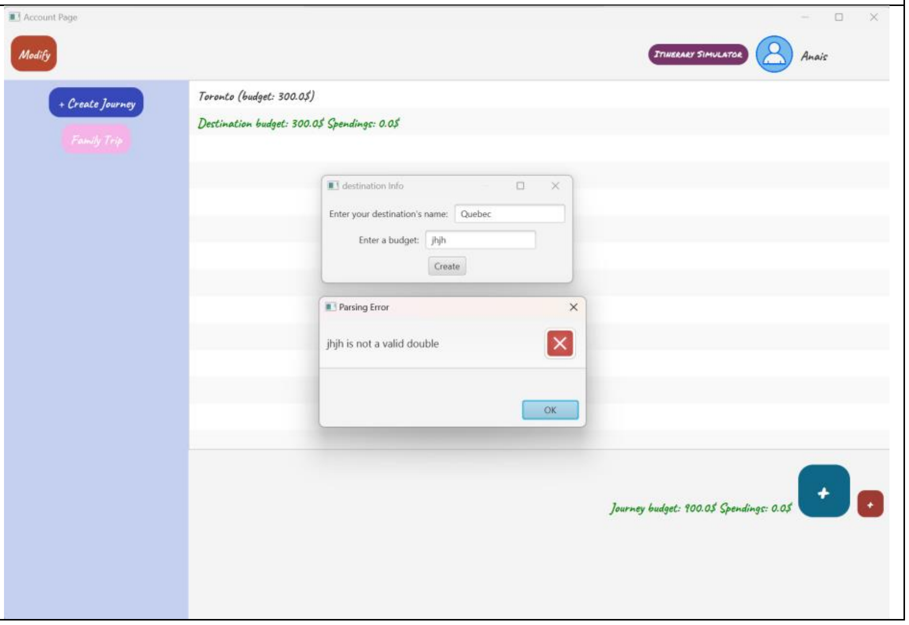
EXCEPTION HANDLING IN THE ACCOUNT PAGE

Exception Handling for adding or modifying a Journey:

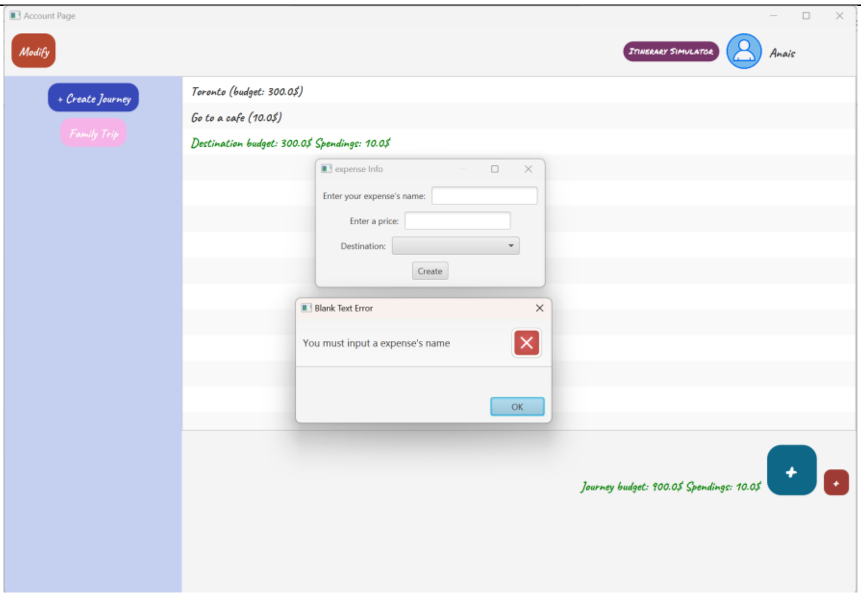
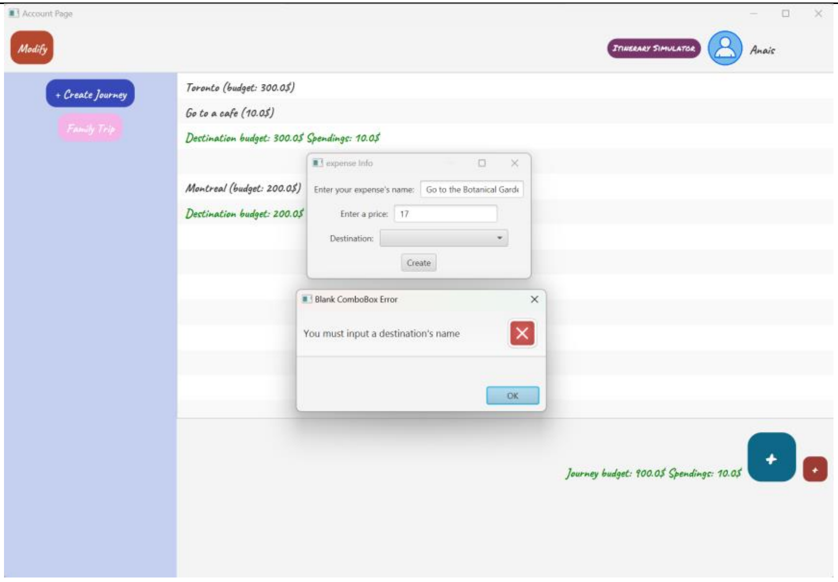
1-If the budget inserted is not a double, an Alert of type "ERROR" appears.	 Account Page Modify Itinerary Simulator Anais + Create Journey Journey Info Enter your journey's name: Family Trip Enter a budget: jdjs Create Delete Parsing Error jdjs is not a valid double OK

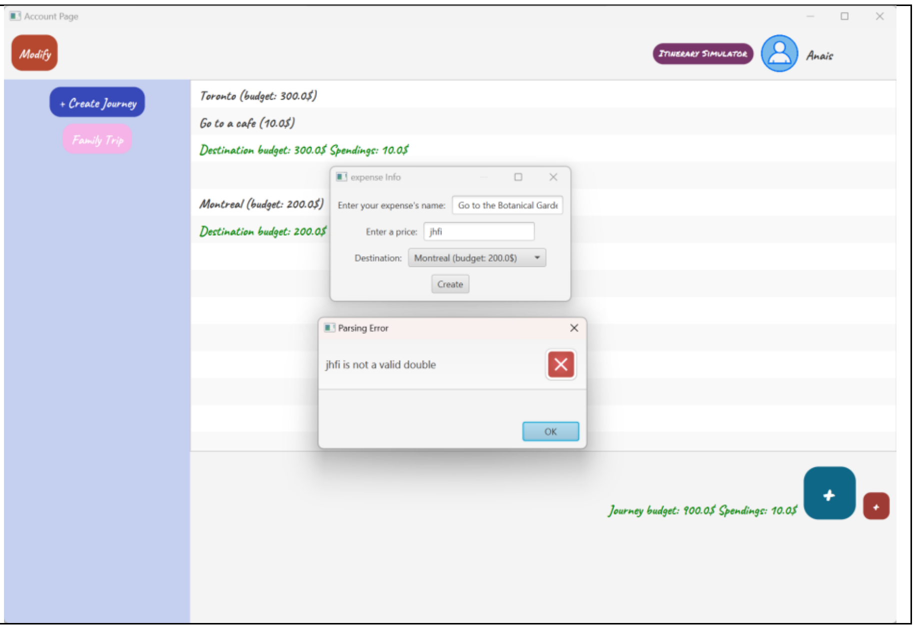
2- If the user does not input a journey's name, an Alert of type "ERROR" appears.	 Account Page Modify Itinerary Simulator Anais + Create Journey Journey Info Enter your journey's name: Enter a budget: Create Delete Blank Text Error You must input a journey's name OK
3- If the user inputs a journey name that already exists, an Alert of type "ERROR" appears.	 Account Page Modify Itinerary Simulator user123 + Create Journey Family Trip Journey Info Enter your journey's name: Family Trip Enter a budget: Create Delete Existing Journey Error This journey name already exists OK

Exception Handling for adding or modifying a destination:

1- If the user inputs a destination name that Already exists, an Alert of type “ERROR” occurs.	 Account Page Modify Itinerary Simulator Anais + Create Journey Family Trip destination Info Create Existant destination Error OK Toronto (budget: 300.0$) Destination budget: 300.0$ Spendings: 0.0$ Enter your destination's name: Toronto Enter a budget: The destination already exists Journey budget: 900.0$ Spendings: 0.0$
2- If the user inputs a budget that is not of type double, an Alert of type “ERROR” appears	 Account Page Modify Itinerary Simulator Anais + Create Journey Family Trip destination Info Create Parsing Error OK Toronto (budget: 300.0$) Destination budget: 300.0$ Spendings: 0.0$ Enter your destination's name: Quebec Enter a budget: jhjh jhjh is not a valid double Journey budget: 900.0$ Spendings: 0.0$

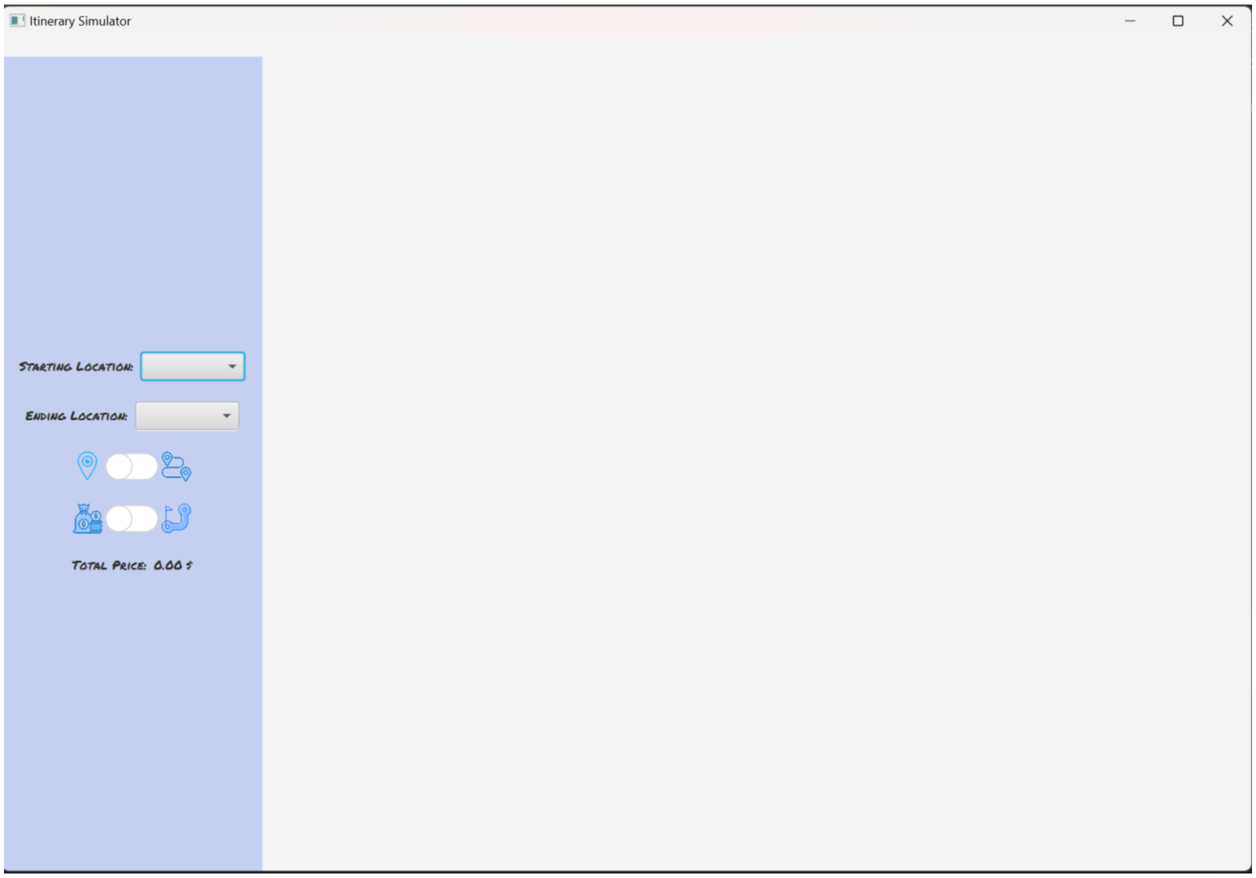
Exception Handling for adding or modifying an expanse:

1-If the user does not input an expanse name, an Alert of type “ERROR” appears.	 Account Page Modify Itinerary Simulator Anais + Create Journey Family Trip expense Info Create OK Toronto (budget: 300.0$) Go to a cafe (10.0$) Destination budget: 300.0$ Spendings: 10.0$ Enter your expense's name: Enter a price: Destination: Blank Text Error You must input a expense's name Journey budget: 900.0$ Spendings: 10.0$
2-If the user does not choose a destination in which to add the expanse to, an Alert of type “ERROR” appears.	 Account Page Modify Itinerary Simulator Anais + Create Journey Family Trip expense Info Create OK Toronto (budget: 300.0$) Go to a cafe (10.0$) Destination budget: 300.0$ Spendings: 10.0$ Montreal (budget: 200.0$) Destination budget: 200.0$ Enter your expense's name: Go to the Botanical Gard Enter a price: 17 Destination: Blank ComboBox Error You must input a destination's name Journey budget: 900.0$ Spendings: 10.0$

3-If the user inputs a price that is not an instance of Double, an Alert of type “ERROR” occurs.	 Account Page Modify Itinerary Simulator Anais + Create Journey Family Trip Toronto (budget: 300.0$) Go to a cafe (10.0$) Destination budget: 300.0$ Spendings: 10.0$ Montreal (budget: 200.0$) Destination budget: 200.0$ expense Info Enter your expense's name: Go to the Botanical Garde Enter a price: jhfi Destination: Montreal (budget: 200.0$) Create Parsing Error jhfi is not a valid double OK Journey budget: 900.0$ Spendings: 10.0$

Itinerary Simulator

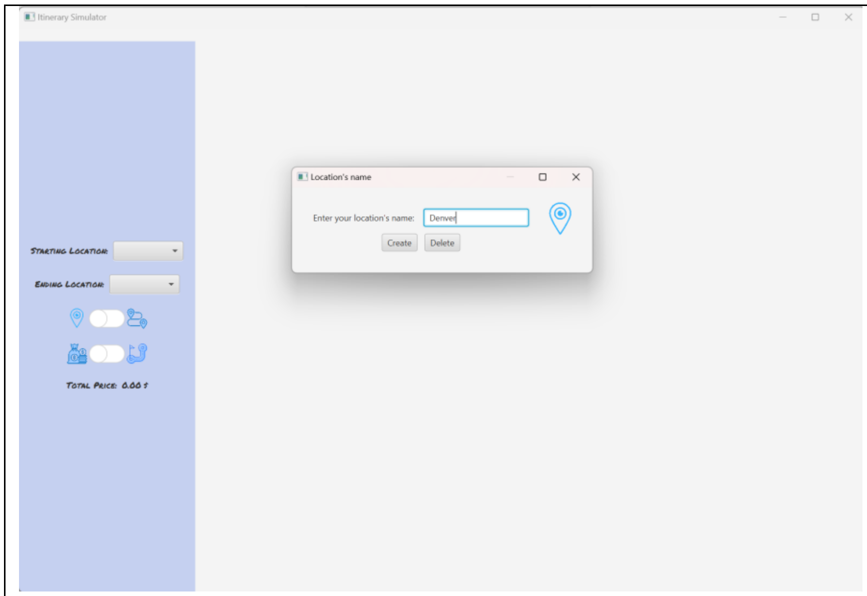
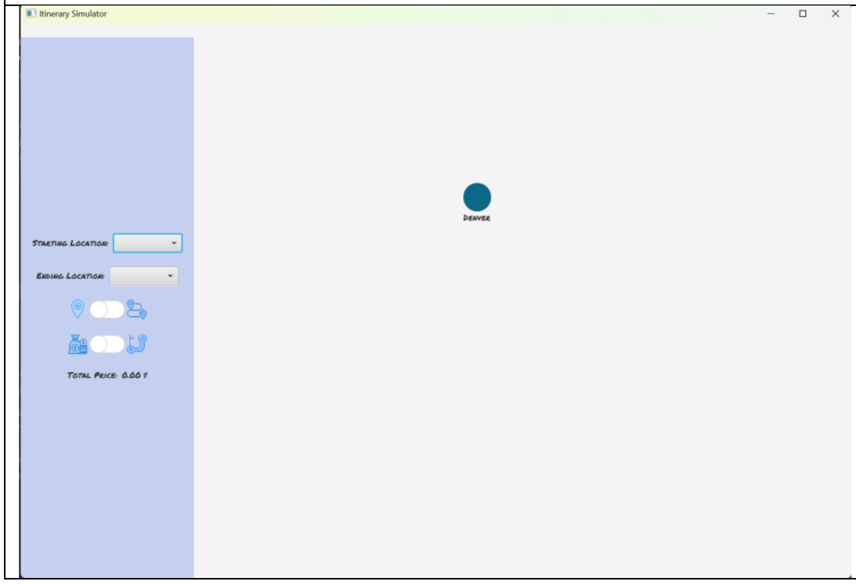
The Itinerary Simulator is the most significant part of this application. It uses Dijkstra's algorithm to find the shortest path between two vertices. It starts off with a simple border pane. It can be accessed by pressing the "Itinerary Simulator" button on the right side of the top pane in the account page or by directly running the "Itinerary Simulator" file.

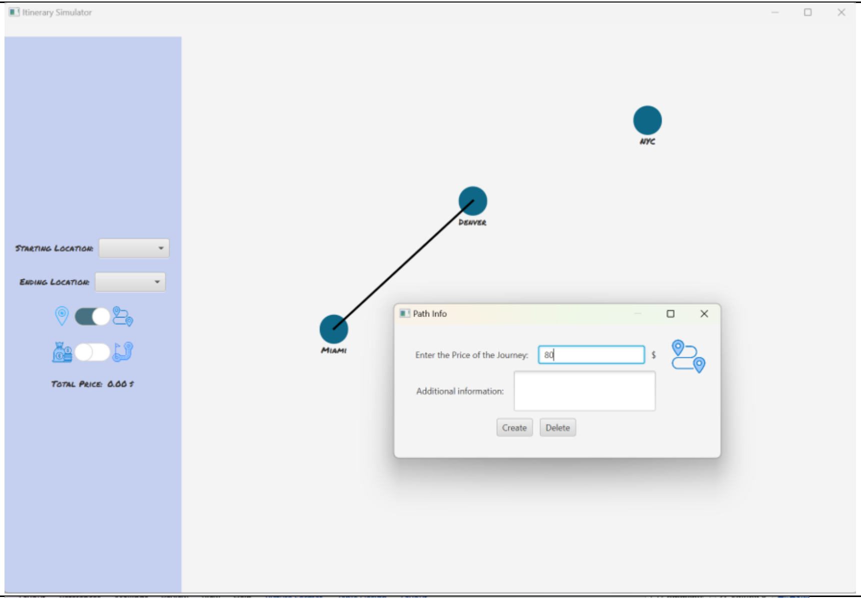
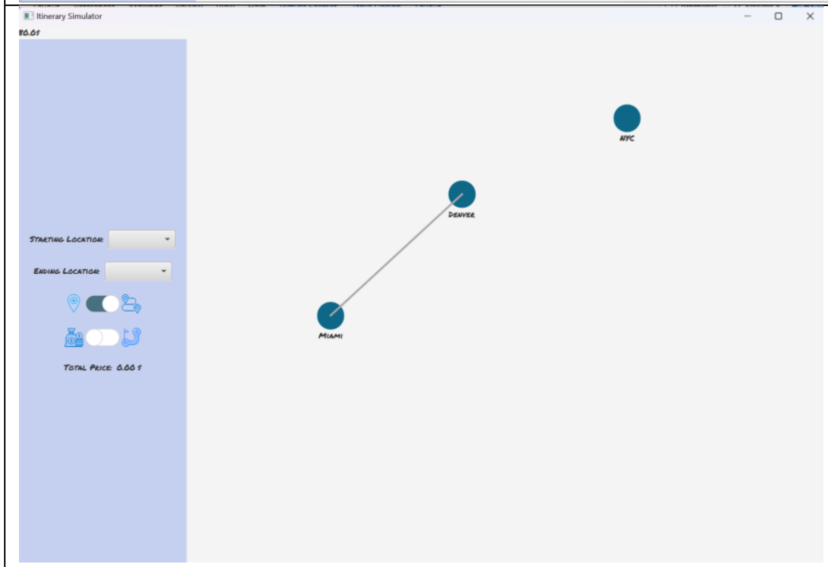


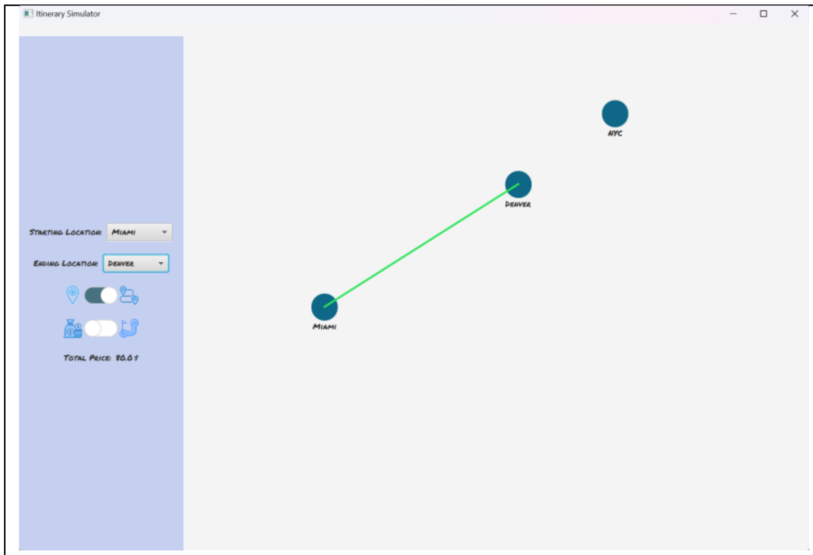
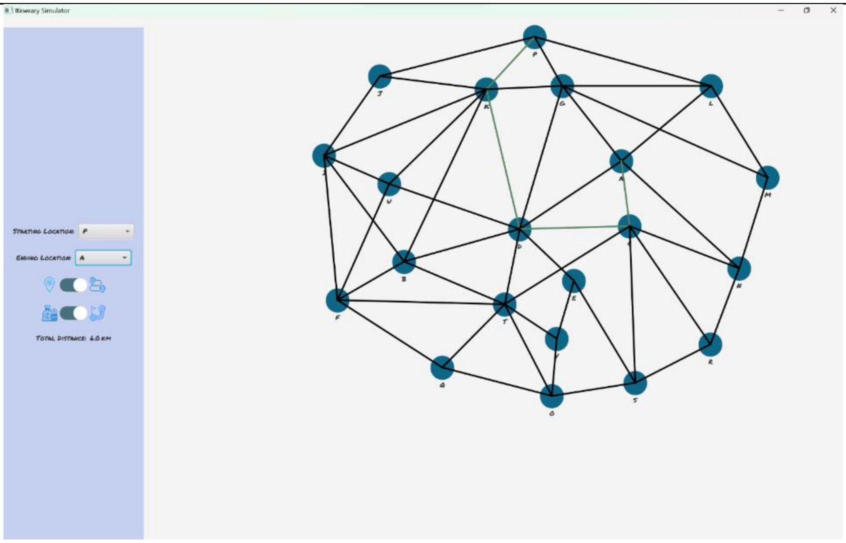
1_ The **LEFT** pane of the BorderPane contains the combo box meant to choose the starting location, the combo box meant to choose the ending location, a ToggleSwitch in order to choose to either add a location or to add a path, , a ToggleSwitch in order to choose to either calculate the lowest amount of money from one location to another or the shortest distance between one location to another, and a label that either displays the lowest amount of money or the shortest distance.

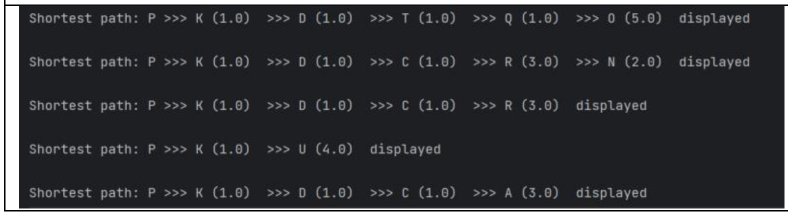
2_The **TOP** of the BorderPane displays the weight of a path if the user's mouse enters a line.

4_The **CENTER** of the BorderPane is where locations and paths are added.

 Itinerary Simulator Location's name Enter your location's name: Denver Create Delete Starting Location: Ending Location: Total Price: 0.00 $	If the first ToggleSwitch is in the location option, then whenever the user clicks somewhere in the center pane, there will be a new stage displayed asking the user to input the location's name.
 Itinerary Simulator Denver Starting Location: Ending Location: Total Price: 0.00 $	The button will be created where the user clicked on the pane (the button is styled like a circle).

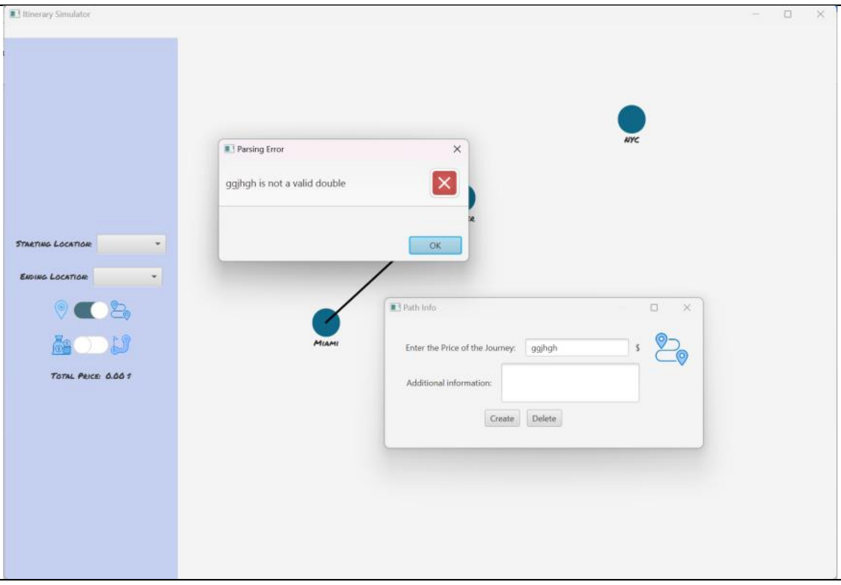
 Itinerary Simulator Starting Location: Ending Location: Total Price: 0.00 $ NYC Denver Miami Path Info Enter the Price of the Journey: 80 $ Additional information: 	If the first ToggleSwitch is in the path option, then whenever the user clicks on two different locations, there will be a line starting from the first pressed button and ending on the second pressed button.
 Itinerary Simulator 80.0$ Starting Location: Ending Location: Total Price: 0.00 $ NYC Denver Miami	If the mouse enters the line, its weight will be displayed in the top left of the BorderPane.

 Starting Location: Miami Ending Location: Denver Total Price: 80.0 $	When the user chooses a starting and ending location, the shortest path is highlighted in green. In addition, the "Total" label in the left pane is set to the shortest path's weight. The shortest path is found based on Dijkstra's algorithm.
 Starting Location: P Ending Location: A Total Distance: 6.0 km	The user can add as many paths or locations as needed (if the space allows it). Here, the shortest path is from P to A, and it goes in the order: P, K, D, C, A. The shortest path's distance is 6 km, as displayed. Note: the path is highlighted in dark green because the image comes from an older version of the project. In the earliest version, the path is highlighted in light green to be more visible.


`Shortest path: P >>> K (1.0)  >>> D (1.0)  >>> T (1.0)  >>> Q (1.0)  >>> O (5.0)  displayed`
`Shortest path: P >>> K (1.0)  >>> D (1.0)  >>> C (1.0)  >>> R (3.0)  >>> N (2.0)  displayed`
`Shortest path: P >>> K (1.0)  >>> D (1.0)  >>> C (1.0)  >>> R (3.0)  displayed`
`Shortest path: P >>> K (1.0)  >>> U (4.0)  displayed`
`Shortest path: P >>> K (1.0)  >>> D (1.0)  >>> C (1.0)  >>> A (3.0)  displayed`
	Here are different combinations of paths chosen. The first shortest path that I

	wanted displayed was from P to O, the second one was from P to N, the third one was from P to R, the fourth one was from P to U, and the fifth one was from P to A. The last one was the one displayed above.

EXCEPTION HANDLING IN ITINERARY SIMULATOR

	When creating a path, if the weight is not an instance of double, then an alert of type "ERROR" appears.

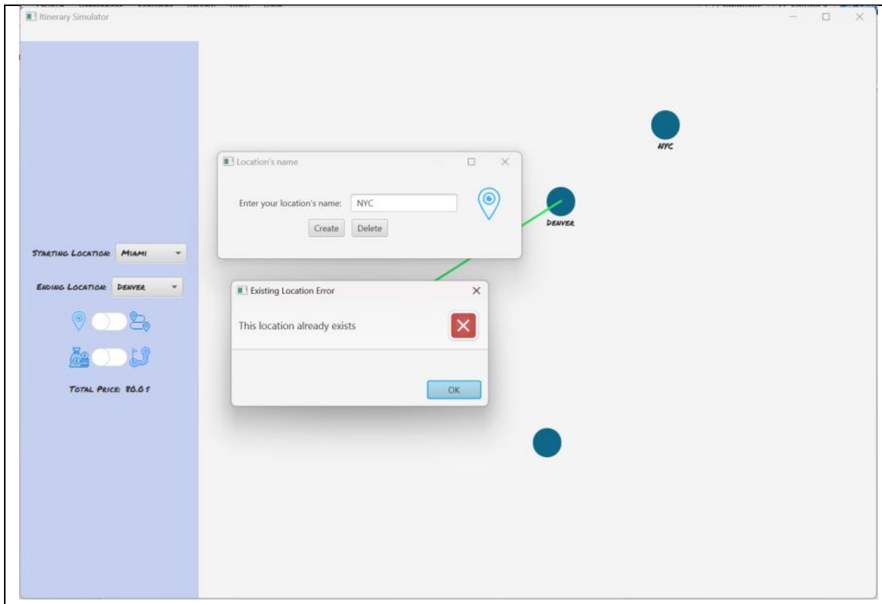
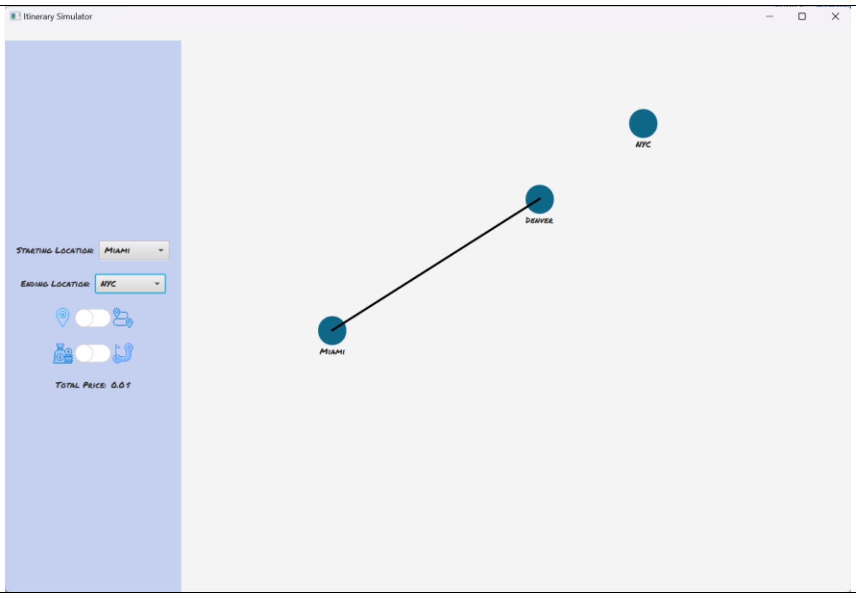
Parsing Error

ggjhgh is not a valid double

Path Info

Enter the Price of the Journey: ggjhgh $

Additional information:

 Itinerary Simulator Location's name Enter your location's name: NYC Create Delete Existing Location Error This location already exists OK Starting Location: Miami Ending Location: Denver Total Price: 80.0 $ 	If the user tries to create a location that already exists, an Alert of type "ERROR" appears.
 Itinerary Simulator Starting Location: Miami Ending Location: NYC Total Price: 0.0 $ 	If the user tries to find the shortest path between two locations that are not connected, then the total price (or the total distance) is simply set to 0, without any action being performed.