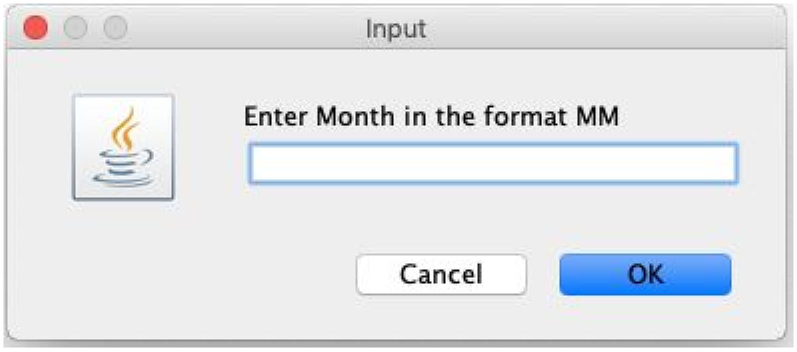
Project 2: Online Airline Reservation

Group Members:

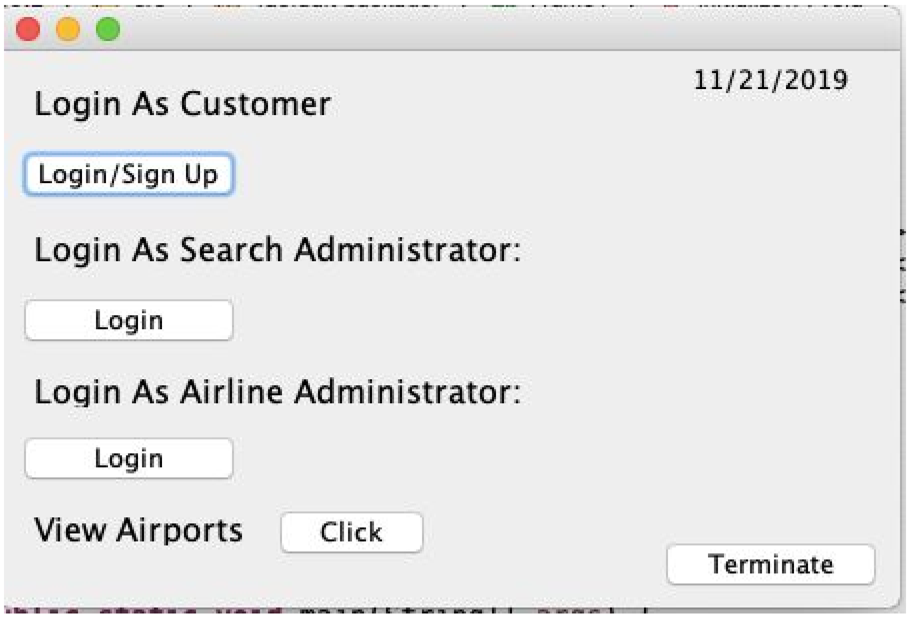
Abubaker Noorzi, Resfred Arthur, Chenhao Li, Rishi Shandal

GUI Screenshot

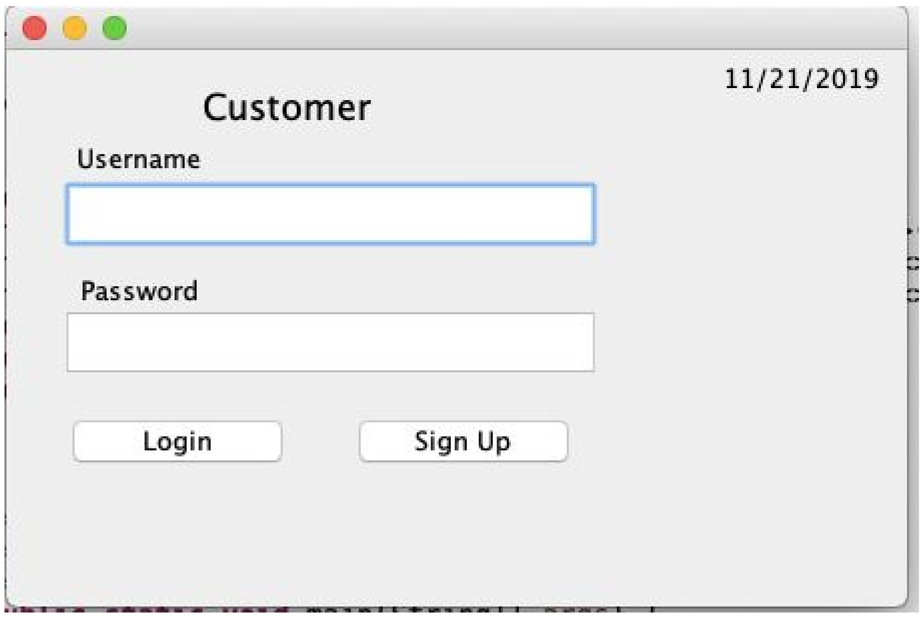
Starting GUI to enter month, day, and year



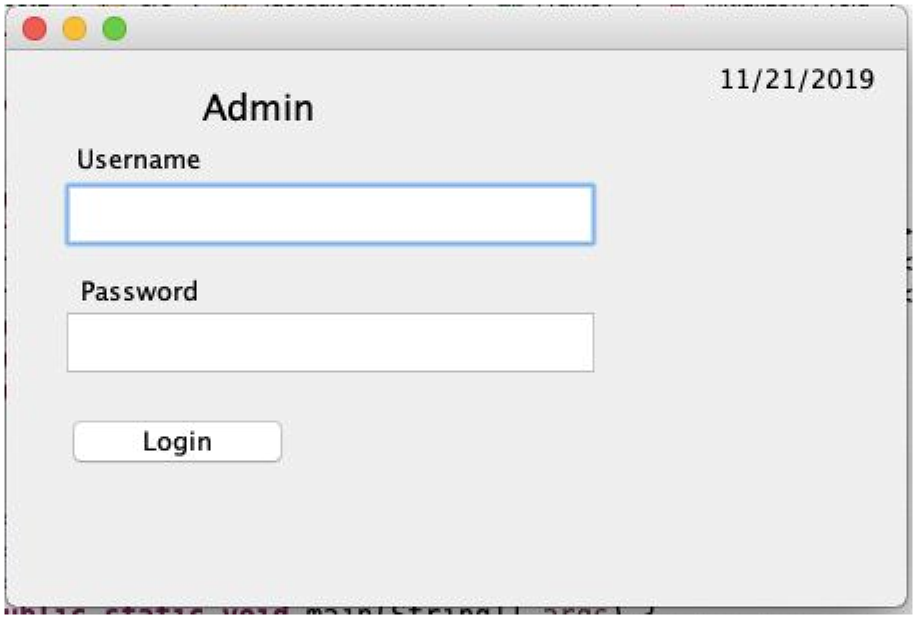
Homepage



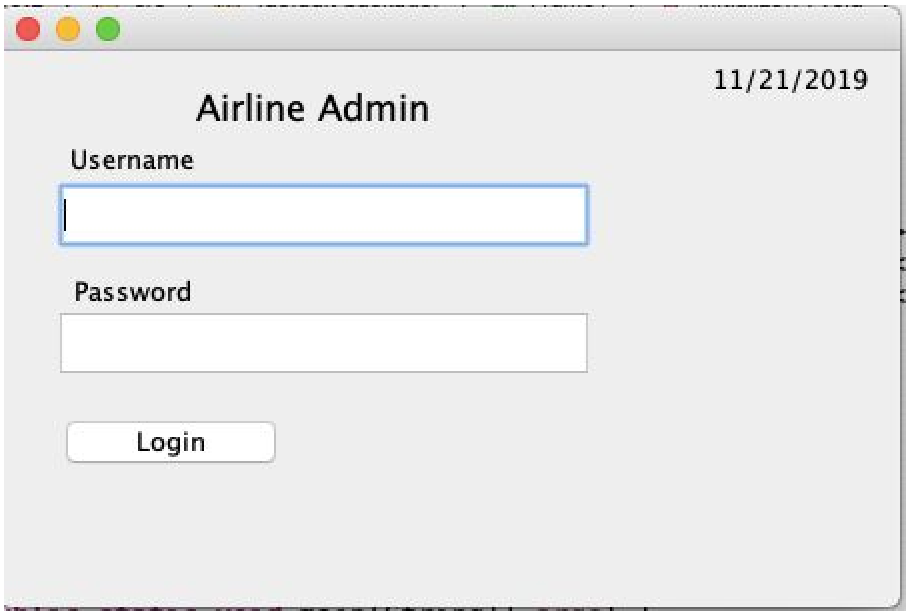
Customer login/signup screen



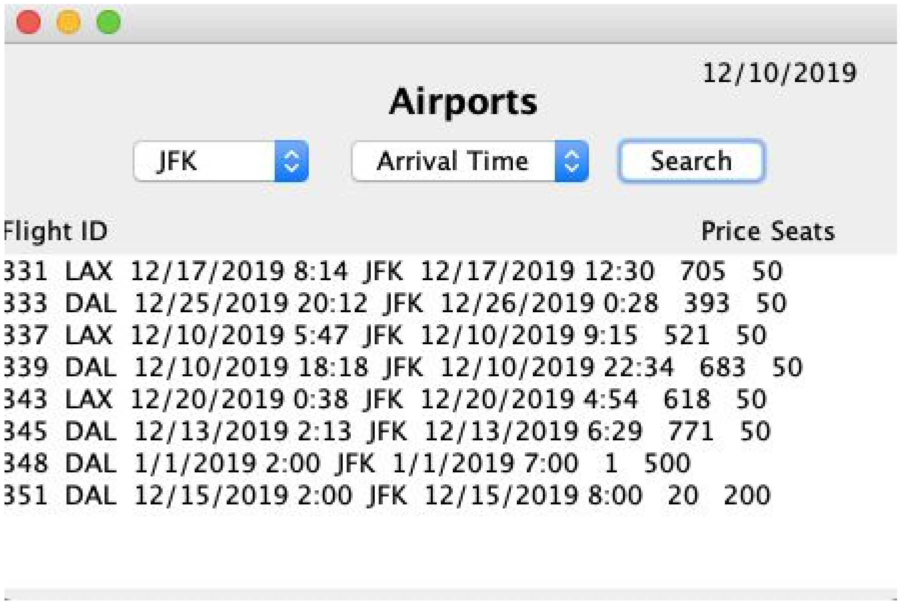
Admin login page



Airline Admin login page

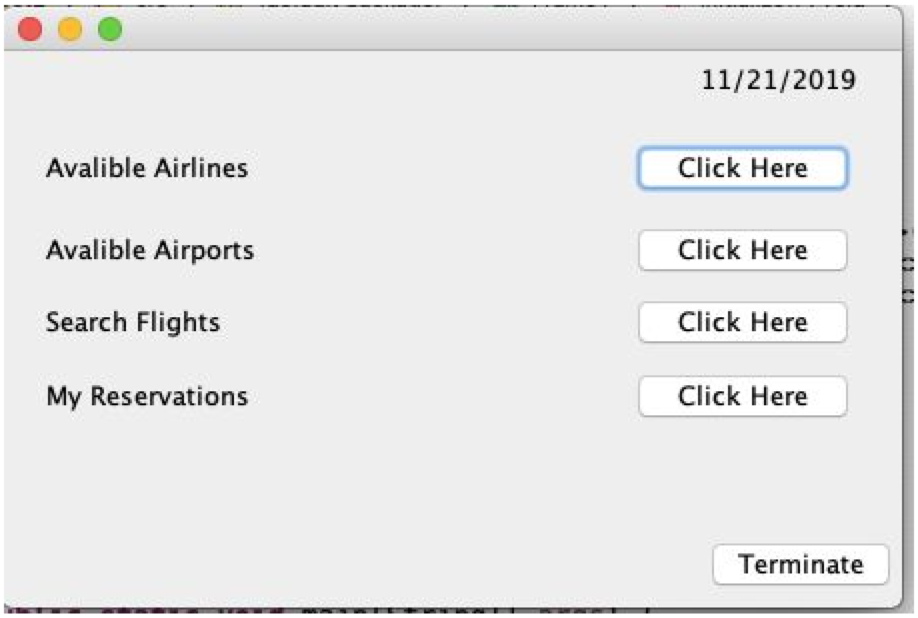


Airports arrival time

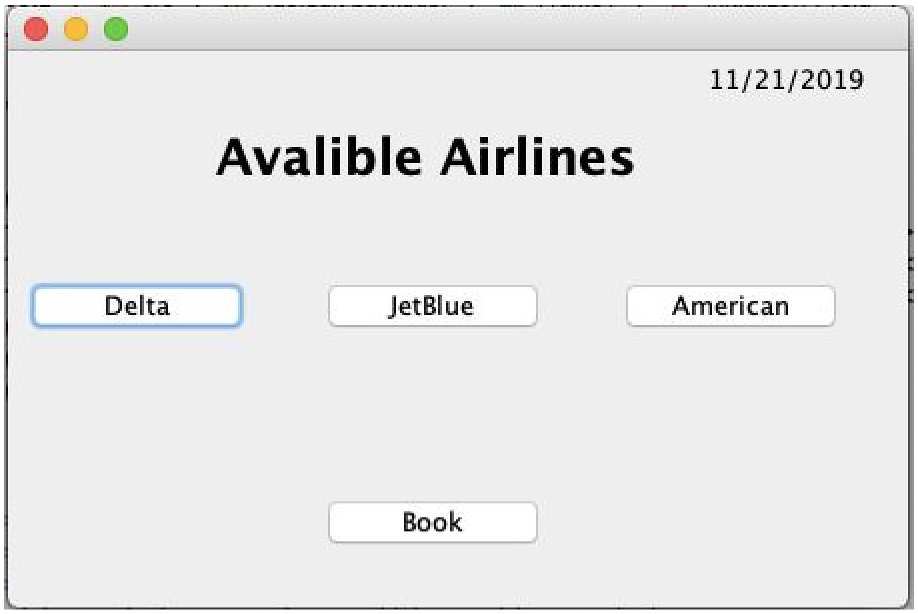


```
Flight ID                                              Price Seats
331  LAX  12/17/2019 8:14  JFK  12/17/2019 12:30   705   50
333  DAL  12/25/2019 20:12  JFK  12/26/2019 0:28   393   50
337  LAX  12/10/2019 5:47  JFK  12/10/2019 9:15   521   50
339  DAL  12/10/2019 18:18  JFK  12/10/2019 22:34   683   50
343  LAX  12/20/2019 0:38  JFK  12/20/2019 4:54   618   50
345  DAL  12/13/2019 2:13  JFK  12/13/2019 6:29   771   50
348  DAL  1/1/2019 2:00  JFK  1/1/2019 7:00   1   500
351  DAL  12/15/2019 2:00  JFK  12/15/2019 8:00   20   200
```

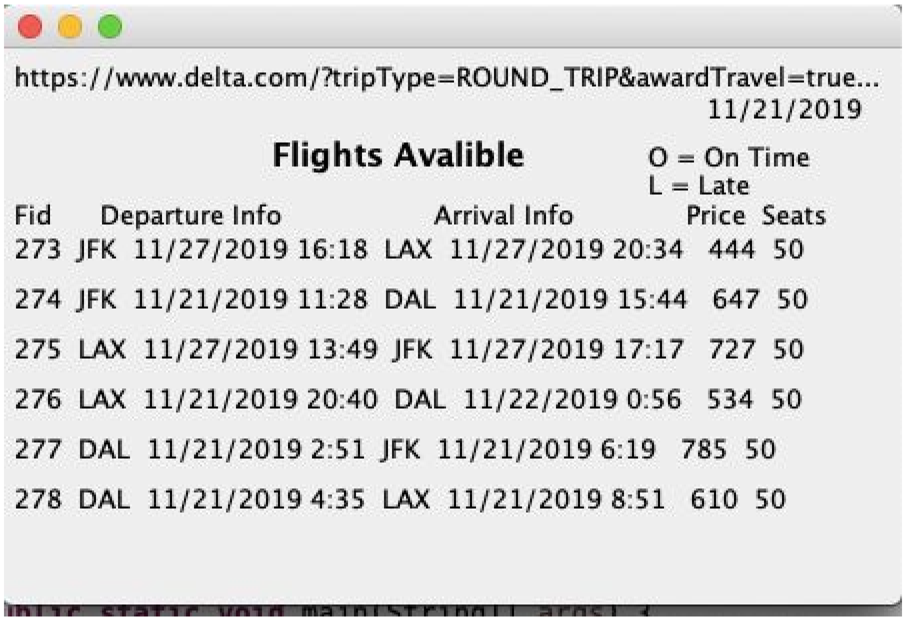
Customer account UI



Available Airlines page

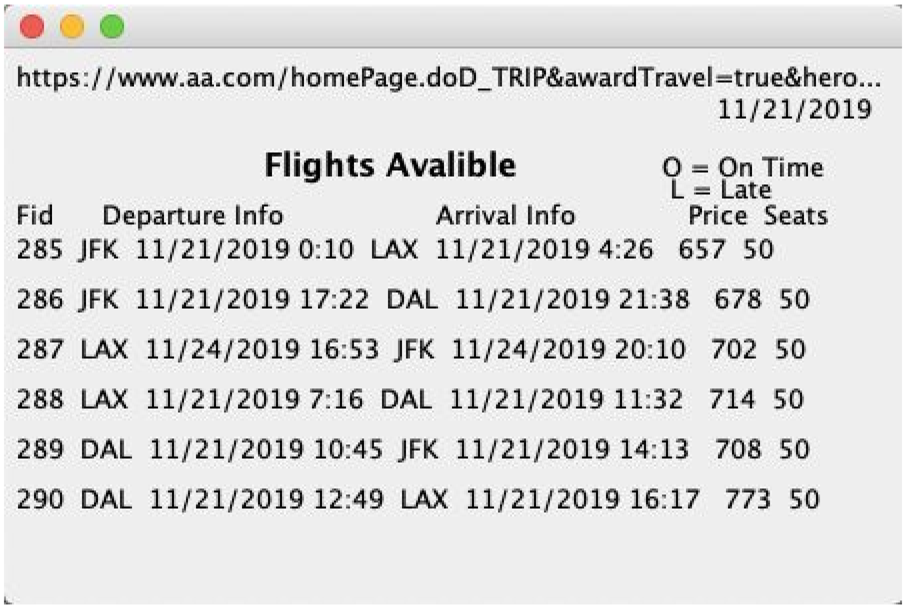


Delta Airlines webpage (displaying available flights)



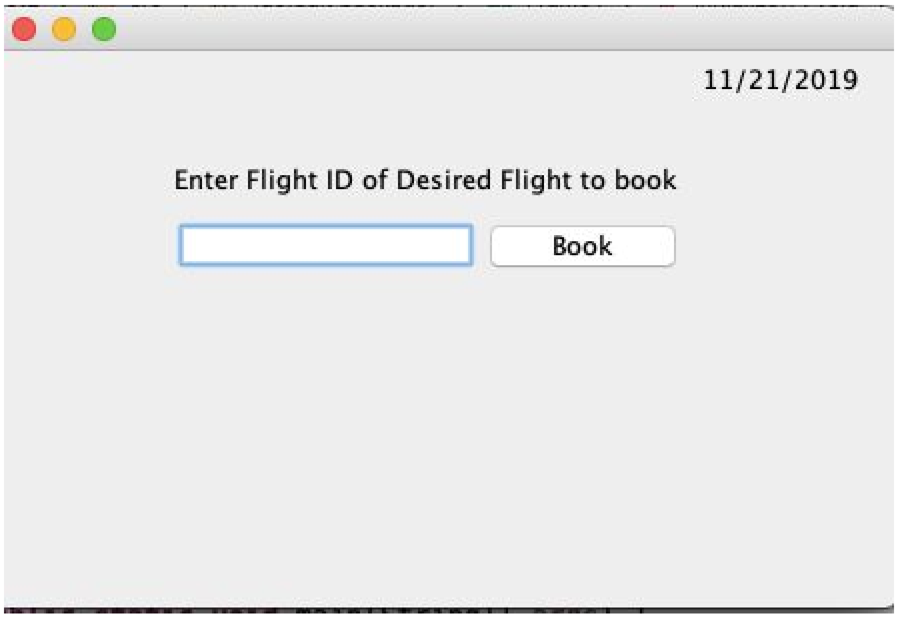
```
https://www.delta.com/?tripType=ROUND_TRIP&awardTravel=true...
                                                          11/21/2019
                    Flights Avalible              O = On Time
                                                  L = Late
Fid    Departure Info              Arrival Info             Price  Seats
273  JFK  11/27/2019 16:18  LAX  11/27/2019 20:34   444   50
274  JFK  11/21/2019 11:28  DAL  11/21/2019 15:44   647   50
275  LAX  11/27/2019 13:49  JFK  11/27/2019 17:17   727   50
276  LAX  11/21/2019 20:40  DAL  11/22/2019 0:56   534   50
277  DAL  11/21/2019 2:51  JFK  11/21/2019 6:19   785   50
278  DAL  11/21/2019 4:35  LAX  11/21/2019 8:51   610   50
```

American Airlines webpage (displaying available flights)

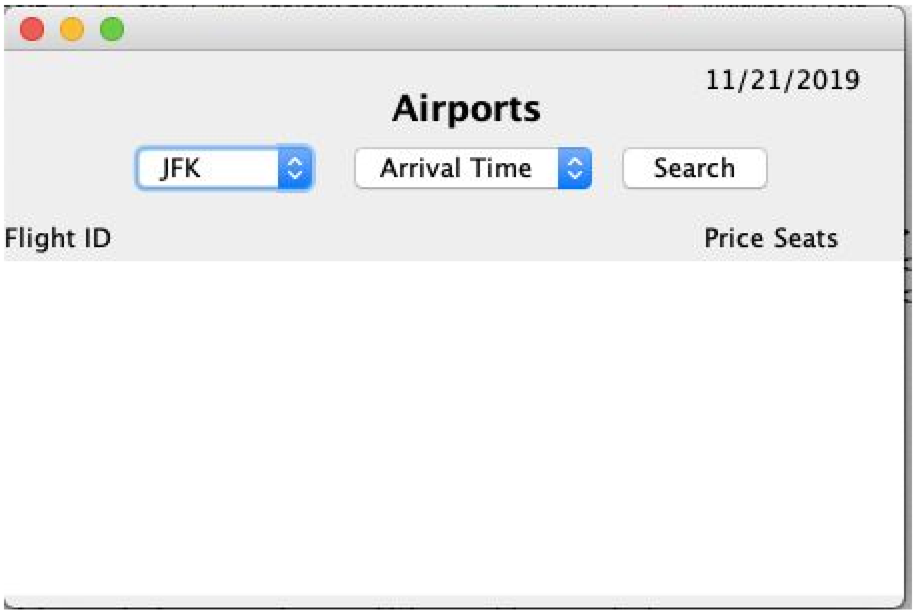


```
https://www.aa.com/homePage.doD_TRIP&awardTravel=true&hero...
                                                          11/21/2019
                    Flights Avalible              O = On Time
                                                  L = Late
Fid    Departure Info              Arrival Info             Price  Seats
285  JFK  11/21/2019 0:10  LAX  11/21/2019 4:26   657   50
286  JFK  11/21/2019 17:22  DAL  11/21/2019 21:38   678   50
287  LAX  11/24/2019 16:53  JFK  11/24/2019 20:10   702   50
288  LAX  11/21/2019 7:16  DAL  11/21/2019 11:32   714   50
289  DAL  11/21/2019 10:45  JFK  11/21/2019 14:13   708   50
290  DAL  11/21/2019 12:49  LAX  11/21/2019 16:17   773   50
```

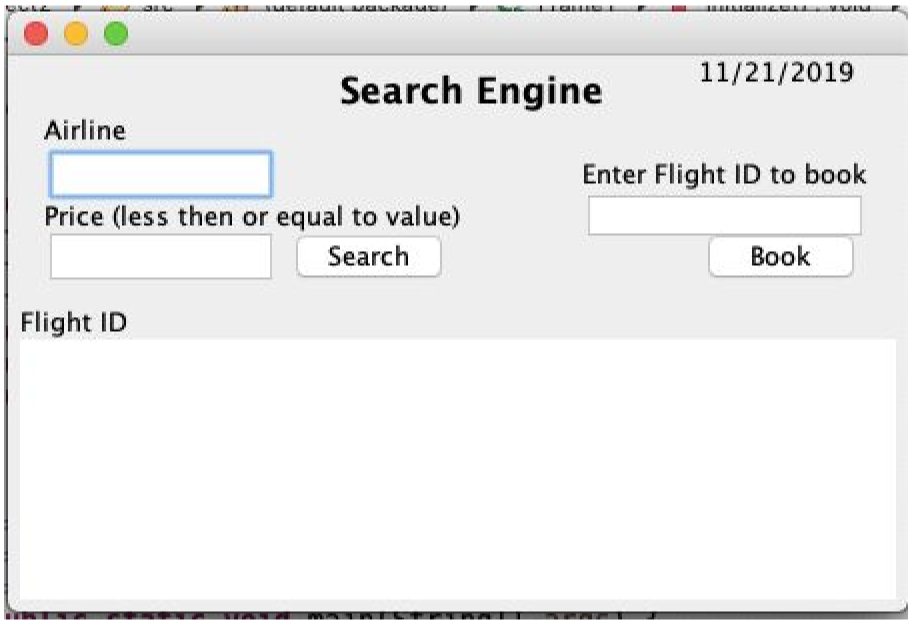
UI to book a flight



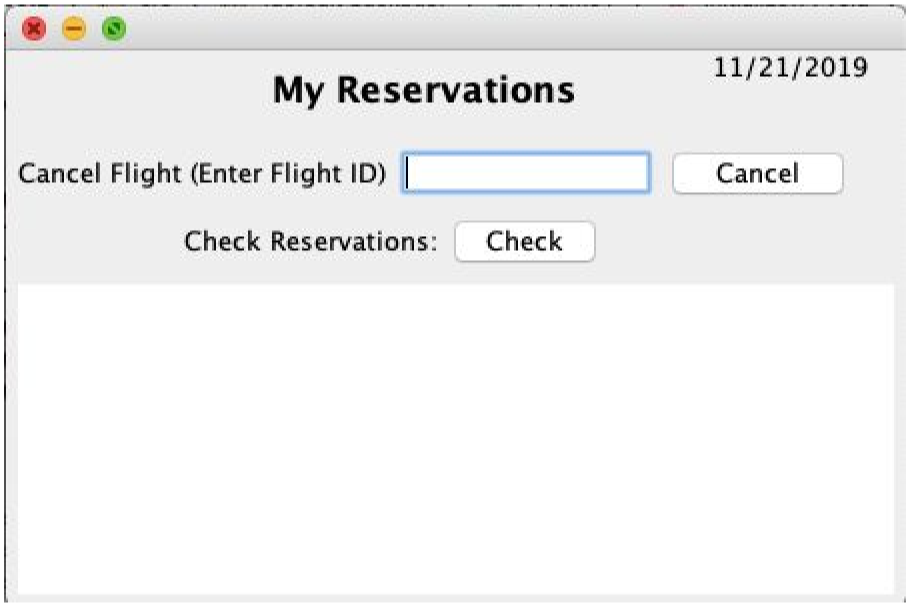
Public airport GUI to view arrival (before search is clicked)



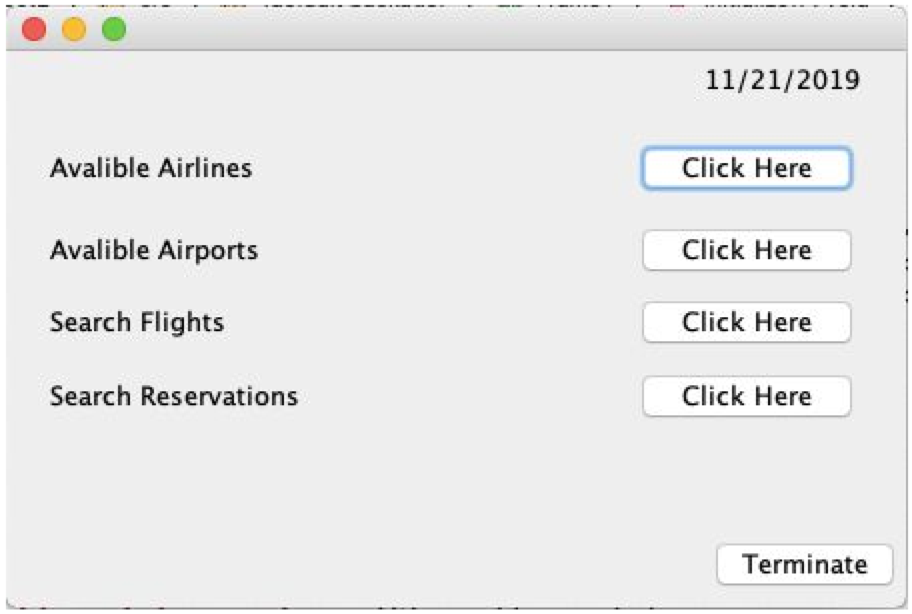
Search Engine to sort for desired flight



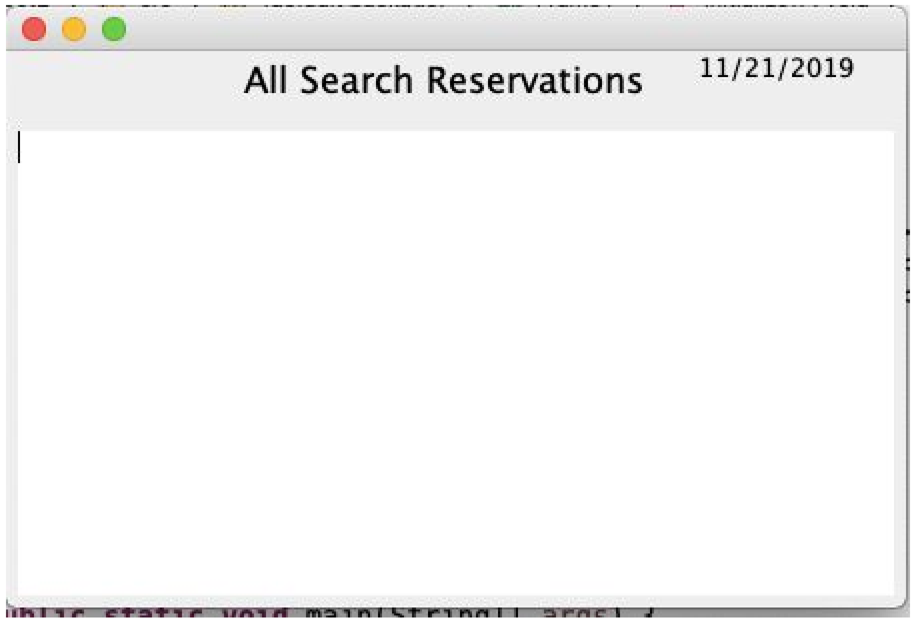
Customer GUI to view and cancel reservations



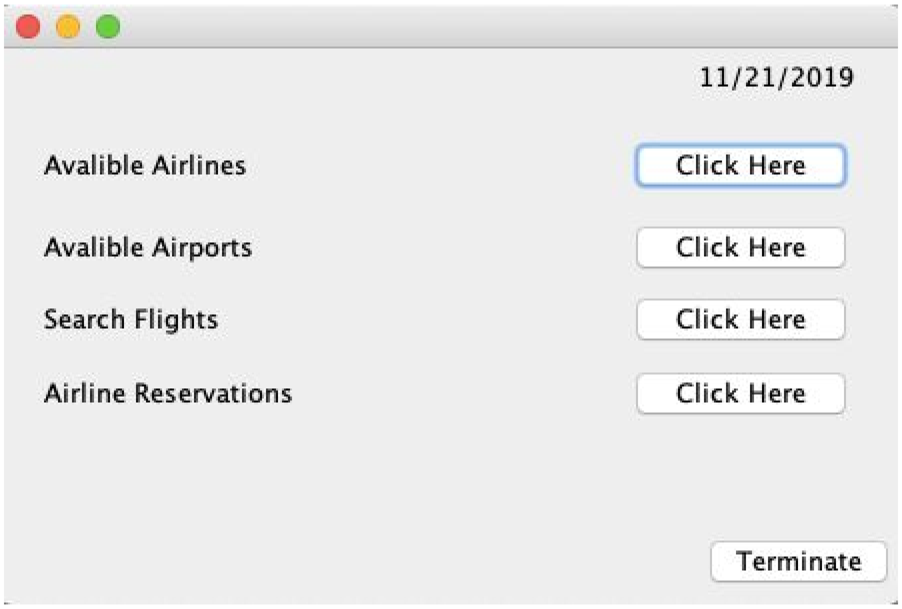
Search Engine Admin UI GUI



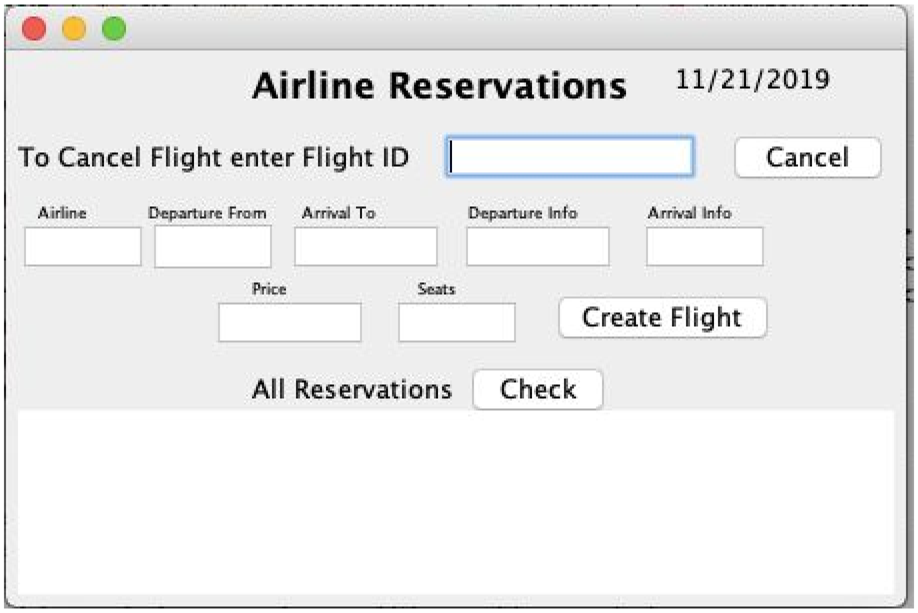
All Reservations made using the search engine(Before reservations)



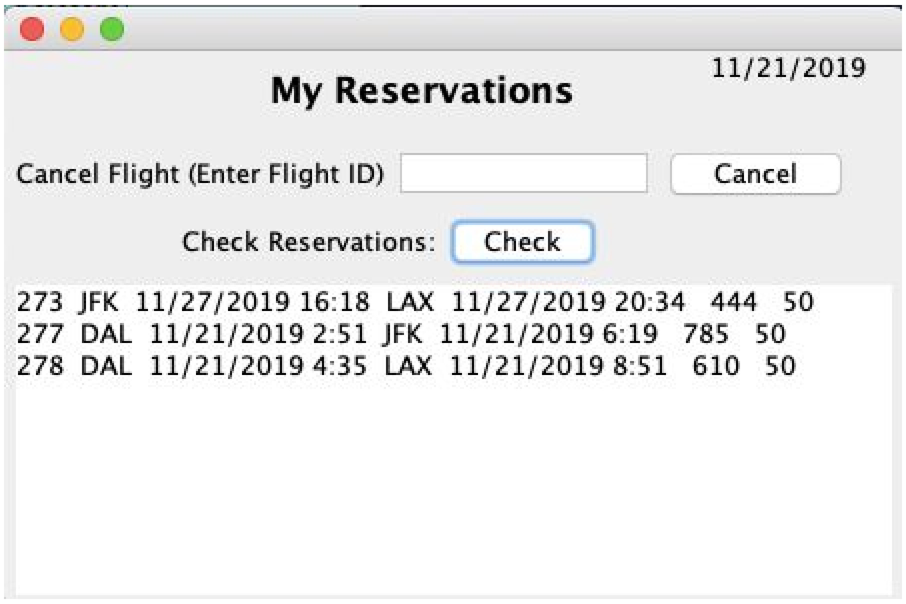
Airline Admin UI GUI



Airline Admin page to create, check and cancel flights

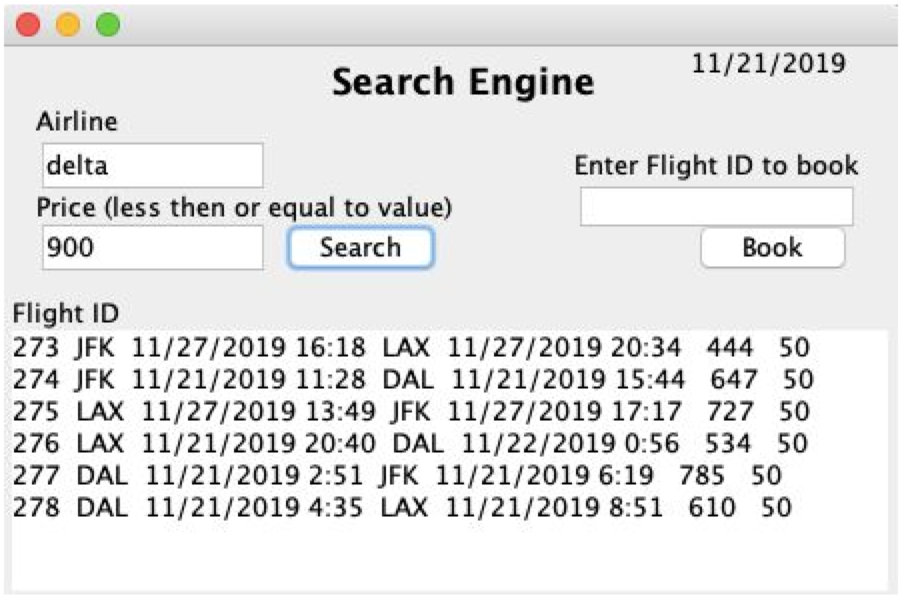


Customer UI checking his/her reservations and UI and cancel flight function



```
273  JFK  11/27/2019 16:18  LAX  11/27/2019 20:34   444   50
277  DAL  11/21/2019 2:51  JFK  11/21/2019 6:19   785   50
278  DAL  11/21/2019 4:35  LAX  11/21/2019 8:51   610   50
```

Search Engine w/ information

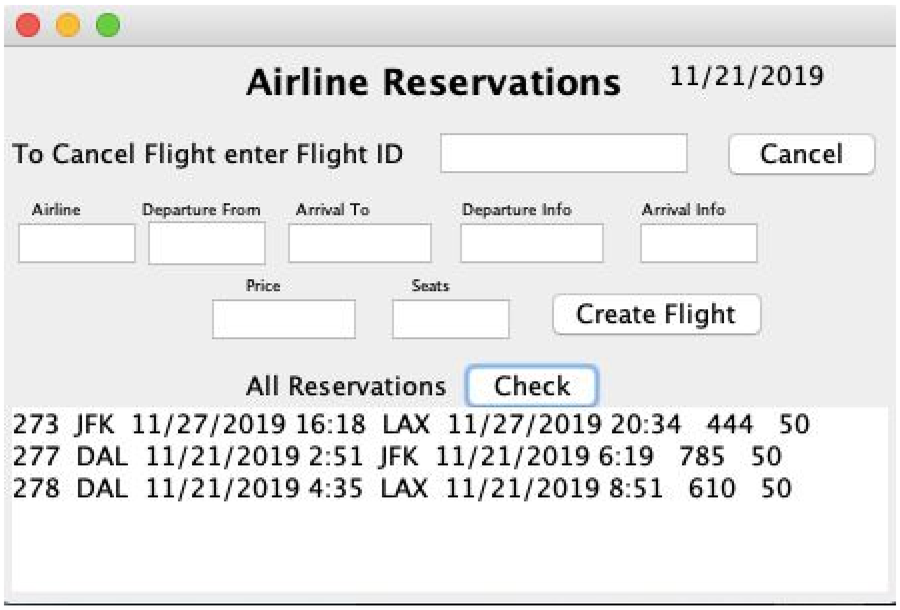


Airline: delta

Price (less then or equal to value): 900

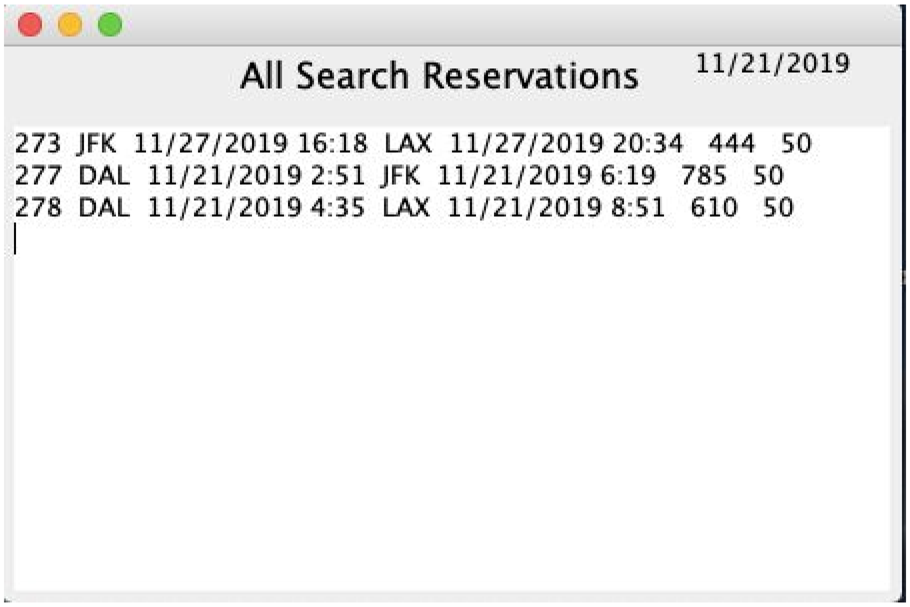
```
273  JFK  11/27/2019 16:18  LAX  11/27/2019 20:34   444   50
274  JFK  11/21/2019 11:28  DAL  11/21/2019 15:44   647   50
275  LAX  11/27/2019 13:49  JFK  11/27/2019 17:17   727   50
276  LAX  11/21/2019 20:40  DAL  11/22/2019 0:56   534   50
277  DAL  11/21/2019 2:51  JFK  11/21/2019 6:19   785   50
278  DAL  11/21/2019 4:35  LAX  11/21/2019 8:51   610   50
```

Airline Reservation Check from by Airline Admin



```
273  JFK  11/27/2019 16:18  LAX  11/27/2019 20:34   444   50
277  DAL  11/21/2019 2:51  JFK  11/21/2019 6:19   785   50
278  DAL  11/21/2019 4:35  LAX  11/21/2019 8:51   610   50
```

All Reservations made through search Engine. Viewed by Search Engine Admin



```
273  JFK  11/27/2019 16:18  LAX  11/27/2019 20:34   444   50
277  DAL  11/21/2019 2:51  JFK  11/21/2019 6:19   785   50
278  DAL  11/21/2019 4:35  LAX  11/21/2019 8:51   610   50
```

Scrum meetings:

11/23/2019

Team met up to share what we have been working on and discuss design specifications for the program. We all had started with implementations of the GUI to request current date and had created varying implementations on how it affects airlines, airports, and their respective flights in pseudocode. Abubaker had at that point created a database to store customer, flights, customerFlights, and a checker to check whether the program was manually exited to restore database information. We utilized his database schema, we decided to progress with his implementation. We decided that a SQL database would be efficient for us since we would be retrieving a large amount of records from the database and since all of us are familiar with its usage. He shared his current code, then we discussed the functional and non-functional requirements of each entity as a team to note down everything that has to be implemented. We all decided worked independently to develop as far as we could until our next meet up.

11/27/2019

Team discussed the implementation of our first prototype since we had created a program that satisfied most basic project requirements. After that, we revisited the design specification and functional requirements to see what we have to work on. We had now to place constraints on Customer. Lastly, we look forward to the next stage of our project work. We decided to move on with GUI implementations of airlines as well as the functionalities that come with it. We also decided to work virtually and send each other updates and share files over the cloud so we're all working on the updated code.

12/01/2019

Abubakar was compiling all our codes together so on this day we met to run tests as well as revisit our specifications and requirements to see what we had to do moving forward. During testing, Abubakar's database unexpectedly crashed so that delayed us a bit. Abubaker redesigned the class and rebuilt the database's structure to support the Implementation. Also, we found a better algorithm to make the data transaction safely through compiler and database

12/05/2019

> We implemented the search engine administrator. With the search engine administrator we allowed the admin to create or cancel flights at their discretion. At the end we verified that the database was indeed accurately representing the data we modified. With shared code we decided to individually debug and unit tested bits of the software.

12/09/2019

> We met together to run through the entire program one final time to test and see that all the functions work as intended. Going through the project checklist we considered attempting the bonus credit which allowed the creation of a new airline or airport, but quickly realized that with our current code it would be difficult to accomplish and would involve most of the code to be modified. We simulated a presentation by asking each other to perform specific tasks based on the project requirements. This helped to both bug test the program and ensure that all of the requirements were met. We updated our report to include the final scrum meeting and made adjustments to the timeline chart to accurately to reflect this.

Functional and NonFunctional Requirements

Customer

Functional Requirements:

- Customer should be able to login/signup for an account.
- Customer should be able to go to search engine to look up availability of flights and make reservations.
- Customer can search web page of airline for available flights and make reservations.
- Customer should be able to cancel booked flights.
- Customer can view his/her reservations made.

NonFunctional Requirements:

- Login/signup information will be saved in the table “customer” in our database.
- Available flights will only be from the current date.
- Customer can only view reservations made by him/her.
- Customer reservations will be updated on the database in real time, but would have to be refreshed to view updated information.
- Database is updated in real time and a checker is implemented to check for abnormal termination of the system. If the implemented “terminate” button is clicked all

information from the database is deleted. If the program is exited any other way, information is saved and can be retrieved upon starting up the program again.
- Information will be consistent across all entities as foreign key restraints are properly placed.
- Each customer will have a unique ID to keep track of customer activity and to provide consistency throughout the database.
- There is no constraint on password requirement.

Airline

Functional Requirements:

- Airline will have webpage displaying all its flights and fares.
- Airline GUI will also show departure and arrival time from and to airports, as well as seat availability.
- Airline webpage will allow customers to book flights
- Airline will have an administrator that can insert/cancel flights
- Cancelled flights will update customer reservations
- Airline Admin can view all reservations made my customer

NonFunctional Requirements:

- All flights displayed will be from current date.
- Webpages will be timed to show how long since the page was loaded to ensure chronology of information.
- Each airline will be its distinct entity (class), and will have information stored in SQL database.
- Airline GUI information will be updated on the database in real time, but would have to be refreshed to view updated information.
- Every flight will have a unique flightID.
- Customer ID and flightID will the candidate key for flight bookings for consistency.

Airport

Functional Requirements:

- There will be a public GUI that displays the arrival information of flights coming to the selected airport.
- There will be a public GUI that displays the departure information of flights going out from the selected airport.

- If an airline cancels a flight, a cancel status must be displayed on the arrival and departure GUI, else display on time.

NonFunctional Requirements:

- There will be a timer since the arrival and departure screen was refreshed.
- Each airport will be its own entity with information saved on the SQL database.
- All GUI for the airport is public.
- Arrival and departure GUI will only show flights from current date.
- Airport GUI information will be updated on the database in real time, but would have to be refreshed to view updated information.
- Each airport will be unique.

Search Engine

Functional Requirements:

- User should be able to look up information for available flights as well as fares
- The user should be able to sort display by fares and airline.
- If flight is full, the display must alert the user.
- Customer should be able to make reservations using search engine.
- Customer should be able to cancel reservations via the search engine.
- There should be a Search Engine Admin GUI where only the Search Engine admin can view reservations made using the search engine.

NonFunctional Requirements:

- There will be a search engine component of the SQL database that links to each airline and flights that keeps track of reservations made by customers.
- Search Engine will only be accessible by customers.
- The sorting of flights by fare will be a “less than” sort where it will display flights that cost less than the specified amount.

Group Member Assigned Task

Abubaker Noorzi:

- Lead Developer, Database Manager, Operational Manager, Debugger
- Wrote code major classes Customer, Airlines, Airport, Search Engine etc
- Combined code with his and Resfred's.
- Wrote GUI to accept currentDate, Customer, and Search Engine.
- Connected code to database.
- Debugged major and minor errors in the program.
- Communicated constantly with Resfred to ensure program met requirements and design specifications.
- Provided code to Chenhao and Rishi for testing.

Resfred Arthur:

- Project Functional Manager, Developer, Scrum Leader, Business Requirements Analyst,
- Assigned roles to members of the team.
- Defined code outline (skeleton) for the project as well as necessary classes.
- Wrote implementations of the software to be submitted to Abubaker for integration.
- Wrote code for Airline GUI, and implemented smaller functional changes to the program such as adding CurrentDate displayed on relevant frames,
- Documented specifications and requirements to guide developers.
- Ensured program followed design specifications and functional, as well as nonfunctional requirements by communicating with Lead developer Abubaker.
- Debugged portions of the program and submitted to Abubaker for validation and integration.
- Worked extensively on the project report.
- Organized and led scrum meetings to ensure tasks were completed in a timely manner.

Chenhao Li:

- Test Analyst, Software Architect, Developer, Business Analyst, Project Documenter
- Installed program on his computer and Rishi's computer.
- Wrote implementations of the software to be submitted to Abubaker for integration.
- Assisted Abubaker in connecting code to database and sorting through with relevant queries

- Received code from Abubaker after Resfred had checked for functional and nonfunctional requirements. Tested for any potential bugs.
- Debugged parts of the code and submitted it to Abubaker for validation and integration.
- Worked with Rishi to document progress throughout software development.
- Worked with Resfred on the project report.

Rishi Shandal:

- Test Analyst, Project Documenter.
- Worked with Chenhao to test and debug code received from Abubaker.
- Wrote implementations of the software to be submitted to Abubaker for integration.
- Documented errors and presented it to be fixed.
- Consulted Resfred to discuss potential bug fixes which later was presented to Abubaker for fixing, or if already fixed, integration.
- Fixed a couple of bugs.
- Worked with Chenhao to document progress throughout software

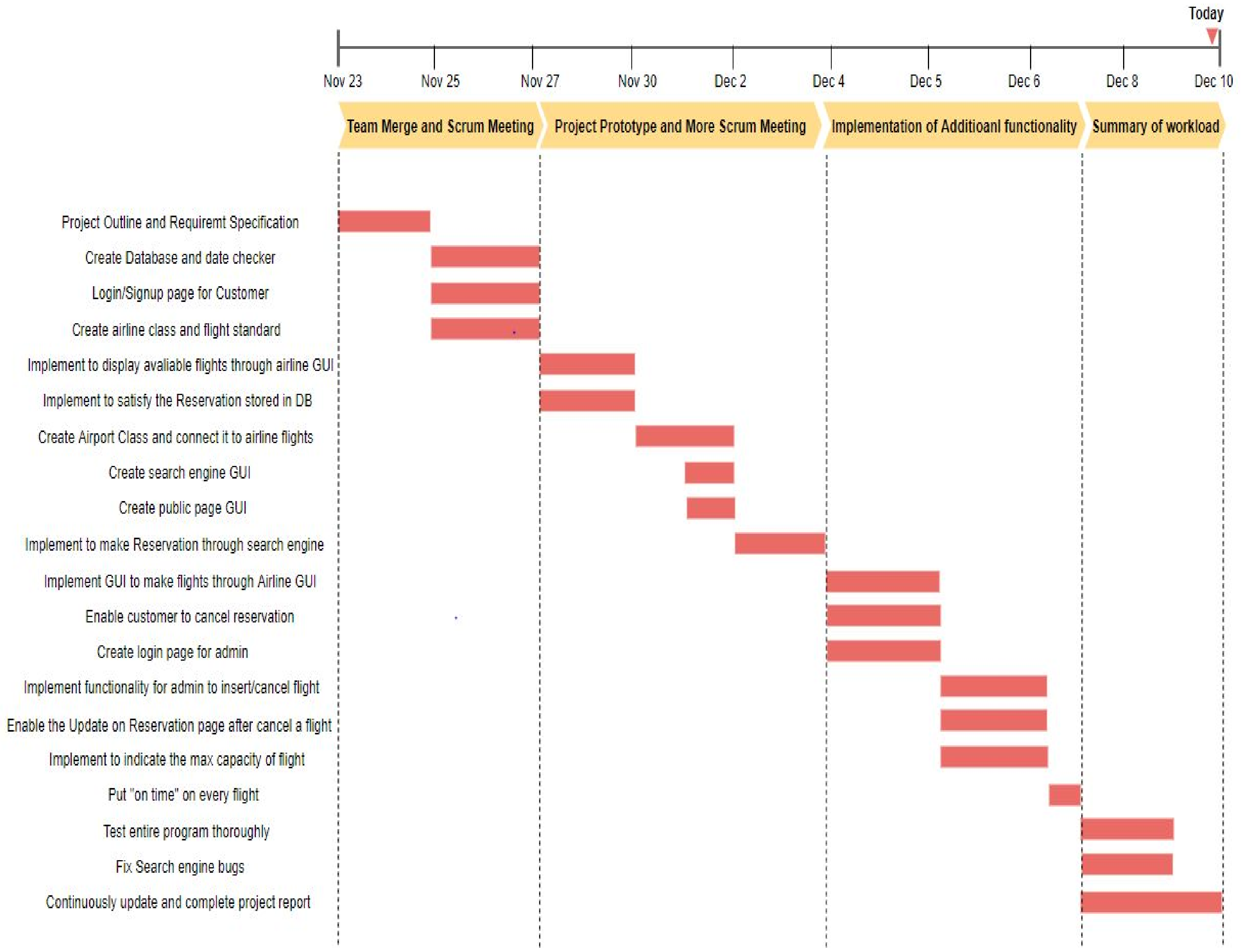
Project Goal/Milestone

[*goals were continuously updated based on where we were*]

- ➢ Outline Project Requirement Specification **[COMPLETED]**
- ➢ Outline Project Design Specification **[COMPLETED]**
- ➢ Create GUI to accept user date input **[COMPLETED]**
    - ○ Month, day, year
- ➢ Create Database and relevant tables following design specifications. **[COMPLETED]**
    - ○ Customer, flights, customerFlights
- ➢ Create checker to check how program was terminated and to save data after exiting. **[COMPLETED]**
- ➢ Create login and signup page GUI for customer.. **[COMPLETED]**
- ➢ Connect customer information to database. **[COMPLETED]**
    - ○ Check to make sure we can read and write into database
- ➢ Create flights standard to current time. **[COMPLETED]**
    - ○ Hardcode airlines and airports for tests
    - ○ Time is randomized
- ➢ Create Airline Classes. **[COMPLETED]**
    - ○ Delta, American, JetBlue
- ➢ Implement GUI to display each airline's available flights. **[COMPLETED]**
- ➢ Create sort functionality for available flights. **[COMPLETED]**
- ➢ Implement GUI to make reservations through on airline's GUI. **[COMPLETED]**
    - ○ Airports will be hardcoded for test reasons
- ➢ Connect reservations to database. **[COMPLETED]**
- ➢ Test reservations in database for correct foreign keys. Ensure consistency. **[COMPLETED]**
- ➢ Create airport classes and connect it to airline flights and database. **[COMPLETED]**
- ➢ Test database for airport consistency. **[COMPLETED]**
- ➢ Create search engine GUI and connect it to the available flights from all airlines. **[COMPLETED]**
- ➢ Create public page GUI to display airports' arrival and departure screen. **[COMPLETED]**
- ➢ Implement functionality for customer to be able to make reservations via the search engine. **[COMPLETED]**
- ➢ Create login page GUI for search engine admin. **[COMPLETED]**
- ➢ Enable customer to cancel reservation which updates database and relevant frames. **[COMPLETED]**
- ➢ Create login page GUI for airline admin. **[COMPLETED]**

- Implement functionality for airline admin to insert and cancel flights and update relevant database which should update relevant frames. **[COMPLETED]**
- Create GUI and implement functionality for airline admin to view reservations made by customers. **[COMPLETED]**
- Customer reservations should be updated if a flight is cancelled. **[COMPLETED]**
- Create date checker to ensure dates not in the calendar are not accepted. **[COMPLETED]**
- Implement timer on relevant frames to show how long since it was opened.
- Implement functionality to keep track of max capacity of flights and prevent full flights from enabling further booking.
- Database crashed. Rebuild Database. **[COMPLETED]**
- Test entire program thoroughly. **[COMPLETED]**
- Fix Search engine bugs. **[COMPLETED]**
- Continuously update and complete project report. **[COMPLETED]**

Timeline Chart



Unit Testing

Tests	Date passed
Create GUI to accept user date input	11/14/2019
Create Database and relevant tables following design specification	11/23/2019
Create checker to check how program was terminated and to save data after exiting	11/27/2019
Create login and signup page GUI for customer	11/27/2019
Connect customer information to database	11/27/2019
Time is Randomized	11/27/2019
Create Airline Classes	11/30/2019
Implement GUI to display each airline’s available flights	11/30/2019
Implement GUI to make reservations through airline’s GUI.	11/30/2019
Connect reservations to database	11/30/2019
Create airport classes and connect it to airline flights and database	11/30/2019
Create search engine GUI and connect it to the available flights from all airlines	12/01/2019
Create public page GUI to display airports’ arrival and departure screen	12/01/2019

Implement functionality for customer to be able to make reservations via the search engine	12/01/2019
Enable customer to cancel reservation which updates database and relevant frames.	12/01/2019
Create login page GUI for search engine admin	12/05/2019
Create login page GUI for airline admin	12/05/2019
Implement functionality for airline admin to insert and cancel flights and update relevant database which should update relevant frames	12/05/2019
Create date checker to ensure dates not in the calendar are not accepted.	12/06/2019
Customer reservations should be updated if a flight is cancelled	12/06/2019
Implement timer on relevant frames to show how long since it was opened	12/06/2019
Implement functionality to keep track of max capacity of flights and prevent full flights from enabling further booking	---
Put “on time” on every flight that isn’t cancelled	12/06/2019
Test entire program thoroughly	12/09/2019
Fix Search engine bugs	12/09/2019
Continuously update and complete project report	12/10/2019