# Design Document

## System Overview

The purpose of the system is to provide software to give travelers the opportunity to book flights and hotels. This system will work directly as a competitor to popular travel sites such as *Expedia*. The target users are anyone who is interested in traveling that need to book flights or hotels. It will be very user friendly so that all age demographics can easily book their trip. The environment for this system is a web app.

## References

- Requirements Document: https://docs.google.com/spreadsheets/d/1Xk8rFzc-kmWawB75tlEwK1by8CzAQHuVhvXCxGaEMFE/edit?usp=sharing

## Environment Overview

Our application will be a console application running on personal computers. The program will be compiled as a java archive file (JAR) that will run on the user's computer. The program will store and access data from JSON file databases stored locally. These include booking (flight, hotel, etc) and user databases in the form of folders containing JSON files.

## User Interface

The User Interface can be found at

https://docs.google.com/document/d/1Ux6zcl5d-zO8fcipzrjDOWc2352LD5b8lL-4iim_u5Q/edit?usp=sharing

## Data Storage

The data for Flytr will be stored in a collection of JSON files. Each User and Booking (Flights/Hotels/etc.) will be stored in their own individual files following the model given below. An X represents a single numerical digit (0-9).0

Flight Schema:

```
{
    "id": "flight_<UUID>",
    "airport": "Example International Airport",
    "from": "City, State",
    "to": "City, State",
    "departure": "YYYY-MM-DDTHH:MM:SS"
    "arrival": "YYYY-MM-DDTHH:MM:SS"
    "seats": [
        [ true, true, true, true ],
        [ true, true, true, true ],
        [ true, true, true, true ],
        [ true, true, true, true ]
    ],
    "allowsDogs": true
}
```

Hotel Data Schema:

```
{
    "id": "hotel_<UUID>",
    "name": "Motel *",
    "location": "City, State",
}
```

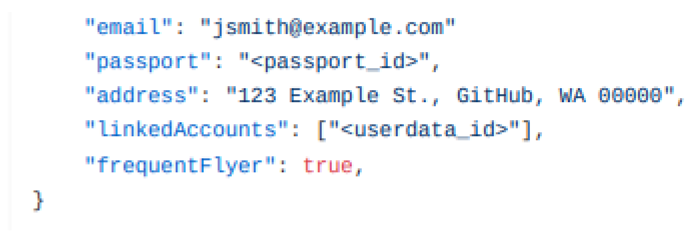
Hotel Room Schema:

```
{
    "number": "001",
    "capacity": 4,
    "takenDates": ["YYYY-MM-DDTHH:MM:SS"]
}
```

User Data Schema:

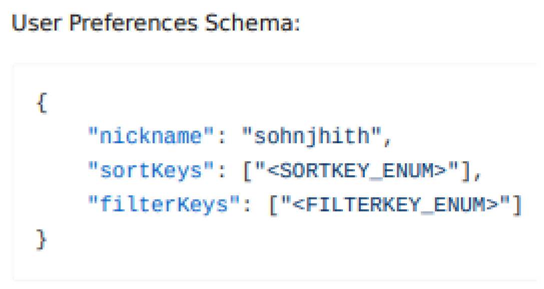
**User Data Schema:**

```
{
    "id": "userdata_<UUID>"
    "username": "exampleusername",
    "firstName": "John",
    "lastName": "Smith",
    "birthDate": "YYYY-MM-DDTHH:MM:SS",
    "phone": "555-555-5555",
```



```
    "email": "jsmith@example.com"
    "passport": "<passport_id>",
    "address": "123 Example St., GitHub, WA 00000",
    "linkedAccounts": ["<userdata_id>"],
    "frequentFlyer": true,
}
```

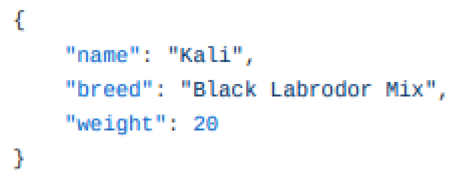
User Preferences Schema:



User Preferences Schema:

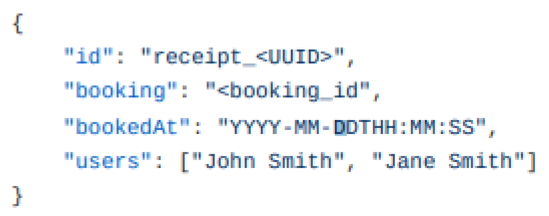
```
{
    "nickname": "sohnjhith",
    "sortKeys": ["<SORTKEY_ENUM>"],
    "filterKeys": ["<FILTERKEY_ENUM>"]
}
```

Dog Schema:



```
{
    "name": "Kali",
    "breed": "Black Labrodor Mix",
    "weight": 20
}
```

Booking Receipt Schema:



```
{
    "id": "receipt_<UUID>",
    "booking": "<booking_id",
    "bookedAt": "YYYY-MM-DDTHH:MM:SS",
    "users": ["John Smith", "Jane Smith"]
}
```

With each Json file located in its corresponding directory:

```
database/
├──userdata/
|  ├──username1/
|  |  ├──data.json
|  |  ├──preferences.json
|  |  ├──saved_people.json
|  |  ├──dogs.json
|  |  └──history/
|  |     ├──receipt_<booking_UUID>.json
|  |     └──...
|  └──...
└──bookings/
   ├──flights/
   |  ├──<flight_id>.json
   |  └──...
   ├──hotels/
```

```
|  ├──<hotel_id>/
|  |  ├──data.json
|  |  └──rooms/
|  |     ├──001.json
|  |     └──...
|  └──...
└──...
```

General Information:

We have chosen to separate each booking item and user into their own file because of the overhead and complexity involved in updating objects if they were all contained in a single file. This way, instead of reading a possibly large JSON file into memory, changing some number of lines, and writing it all back to the file, only a small object file will need to be overwritten. Properties in these schemas such as departure, arrival, birthDate, and other with a set string of "YYYYMM-DDTHH:MM:SS" represent Java LocalDateTimes. This format is simple to both parse into memory and write to a file. As Flytr evolves and later versions are released, these JSON schemas may have additional properties added on or some properties changed.

User Schemas:

Users will have their own separate directory named with their username and password which will contain files for their data, preferences, saved people, and dogs. Saved people refers to people that a user may commonly book with but either do not have their own account or are not linked to the user's account. Users will also a subdirectory for their history of bookings.
Booking Schemas:

The Flight schema contains a seats property which is a 2D-array of booleans representing seats on a flight. True represents an available seat and false is a taken seat. Hotels will be separated into separate directories labled with their id and will contain a data JSON file as well as a subdirectory of JSON files representing the rooms of the hotel.

## Class Diagram – Structural Design

The UML class diagram can be found:

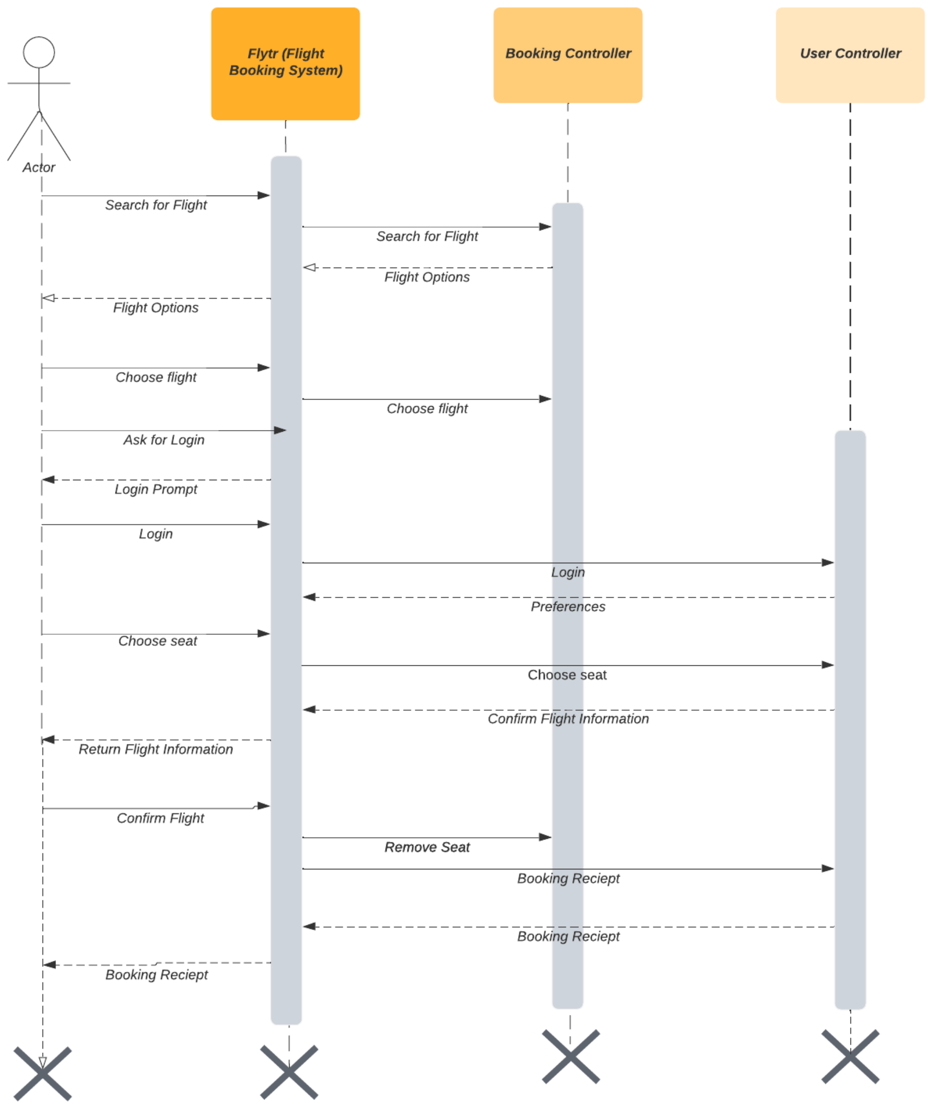
https://lucid.app/lucidchart/6f6d22d9-5665-4b96-a90f-2933b7245743/edit?invitationId=inv_49fdbe83-e100-4a05-954c-b6affaf20c4b

## Sequence Diagrams - Dynamic Model

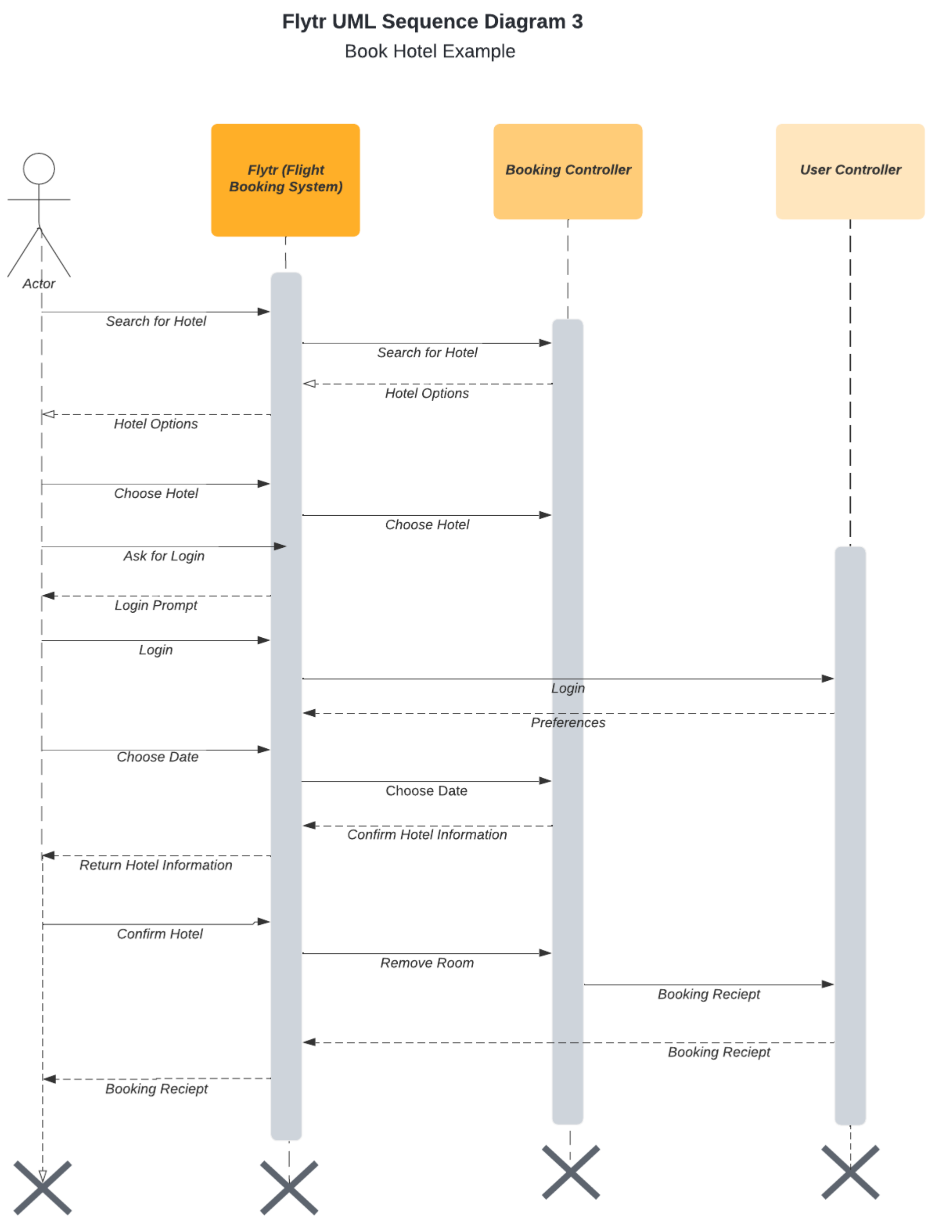
The first scenario is for booking a flight. The user will be going through the process of searching for and booking a flight.

**Flytr UML Sequence Diagram 1**

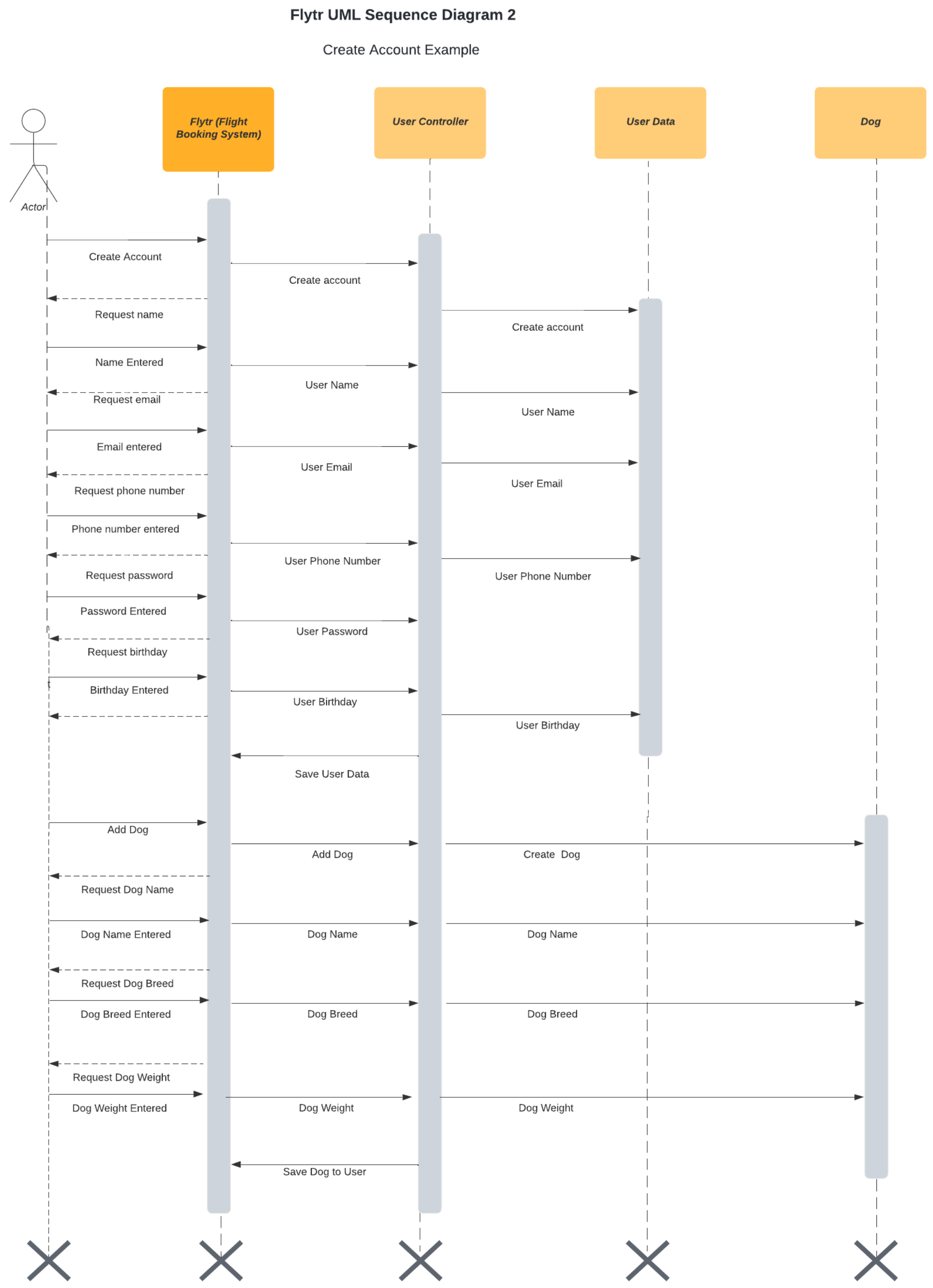
Book Flight Example



The second scenario is booking a hotel. The user will be going through the process for booking a hotel.



The third scenario is for creating an account.


Flytr UML Sequence Diagram 2
Create Account Example
Actor
Flytr (Flight Booking System)
User Controller
User Data
Dog
Create Account
Create account
Request name
Create account
Name Entered
User Name
Request email
User Name
Email entered
User Email
Request phone number
User Email
Phone number entered
User Phone Number
Request password
User Phone Number
Password Entered
User Password
Request birthday
Birthday Entered
User Birthday
User Birthday
Save User Data
Add Dog
Add Dog
Create Dog
Request Dog Name
Dog Name Entered
Dog Name
Dog Name
Request Dog Breed
Dog Breed Entered
Dog Breed
Dog Breed
Request Dog Weight
Dog Weight Entered
Dog Weight
Dog Weight
Save Dog to User