H&W Airlines

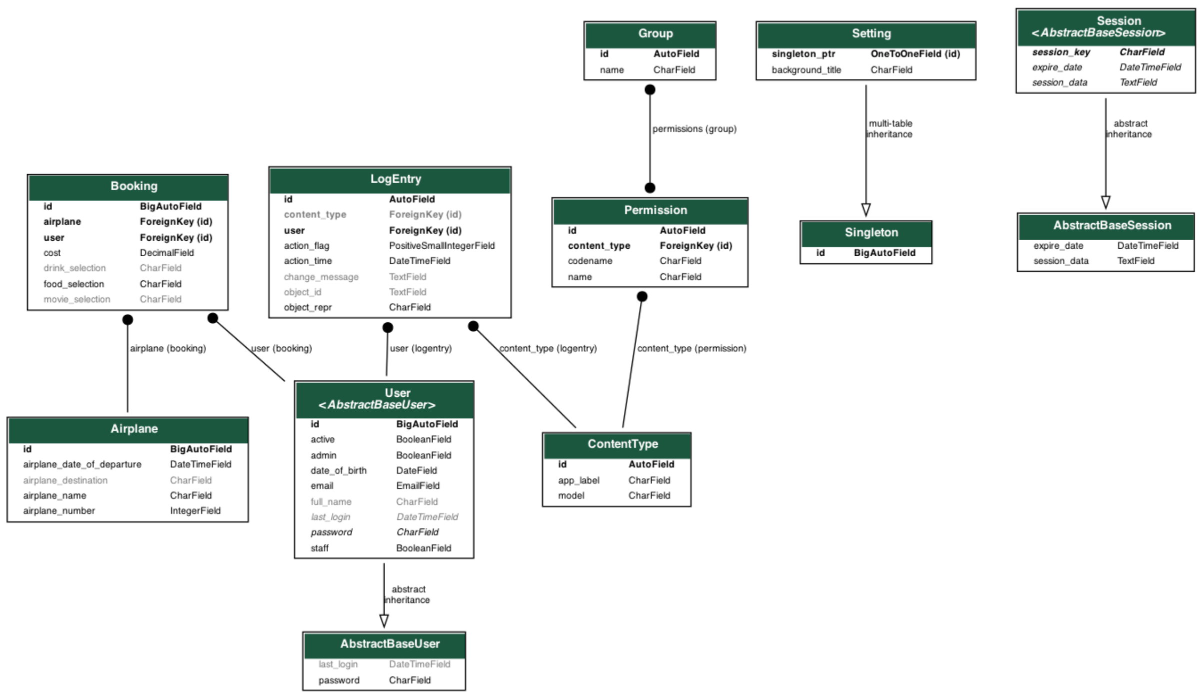
Henock Zemenfes, Evan Swett, and AJ Abam

https://github.com/evsw5643/Airline-Project.git

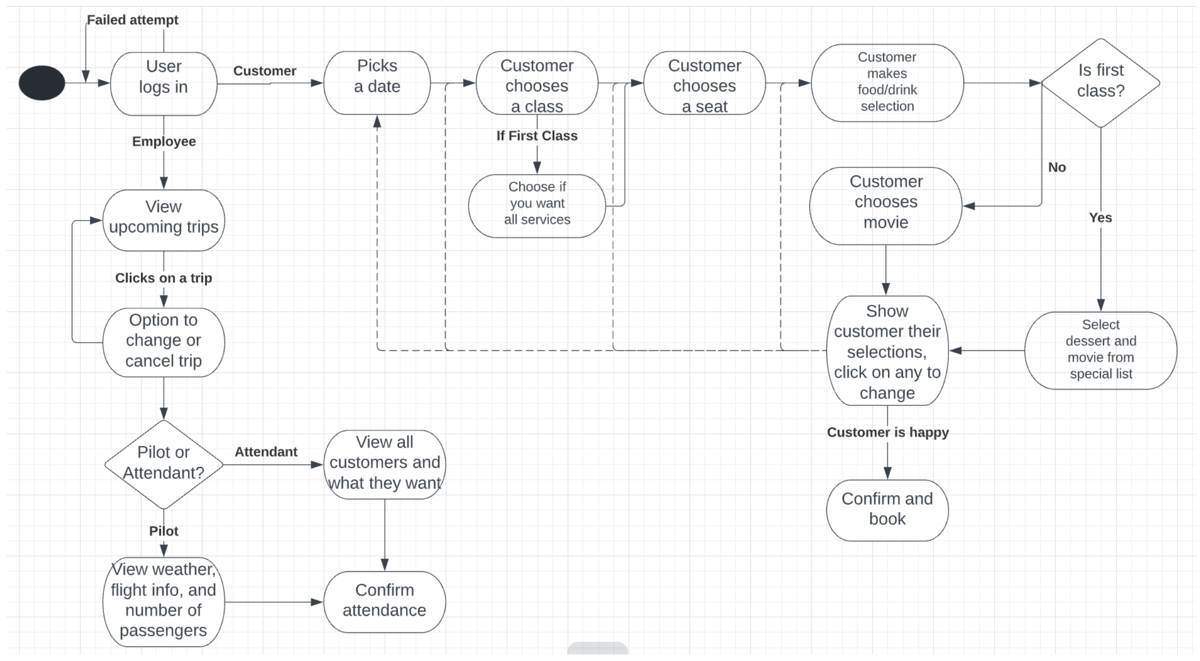
- Final State of System Statement
    - Building out this project has been a really great experience, as we were able to challenge ourselves to learn a framework that we were not initially familiar with. A lot has changed since we began initially discussing the structure of our project. Originally we thought we would implement the project with Kivy. After discussion and a bit more research, we decided that Django offered a lot more interesting and efficient functionality that would enable us to build out the project the way we wanted to. One major change that we made was that instead of incorporating three different levels of access, those being pilot, employee, and customer (first class and coach specifically), we instead focus on creating a process that enables users to register, and returns their booking to them. Along with that, we were not able to implement the libraries that we wanted to, as we ran out of time. The current system enables users to sign up, book a flight, and view the confirmation of the flight.

    - Strategy:
        - Our User model implementation uses a strategy pattern because at runtime it either creates a regular user object (for a passenger), or a superuser if the client is an administrator.
    - Singleton:
        - We used a singleton pattern to give the administrator access to a 'change background' setting. Because it's a singleton the admin can only make this change one time per session.
    - MVT:
        - Object Relational Mapping: We used Django's object-relational mapper to write python code to interact with our database. In our models.py file, you can see the definitions of all of the tables that we have in our table. By running 'python3 manage.py makemigrations' you essentially stage the changes that are to be made to the DB. Running 'python3 manage.py migrate' pushes those tables to the actual Django project. Now we have access to the models via '<ModelName>.objects.all()'

- Final Class Diagram and Comparison Statement



- ○
  - ■ Final UML Diagram


Failed attempt
User logs in
Customer
Picks a date
Customer chooses a class
Customer chooses a seat
Customer makes food/drink selection
Is first class?
Employee
If First Class
Choose if you want all services
No
Customer chooses movie
Yes
View upcoming trips
Clicks on a trip
Option to change or cancel trip
Show customer their selections, click on any to change
Select dessert and movie from special list
Customer is happy
Pilot or Attendant?
Attendant
View all customers and what they want
Confirm and book
Pilot
View weather, flight info, and number of passengers
Confirm attendance

- ○
  - ■ UML from Project 5
- ○ Our UML diagrams did have many changes from our original implementations in Projects 5 and 6. We had to focus on specific tasks as the further we coded the project the more we realized the many elements needed. We decided to focus on main components such as the user and less on the actual employees of the airlines. By doing this we were able to resolve errors pertaining to users instead of trying to balance too much at once. We started big and were able to narrow it down to

focus on a more concise project. The UML diagrams showcase this as instead of having unneeded factors like checking the weather, we could focus on airplane numbers, confirmation, and important factors in our project.

- Third-Party code vs. Original code Statement
    - We used various frameworks and tools to help with this project. The main framework we used was Django. Django is an advanced Python framework that allows us to create very manageable web applications. Combining this with the usage of HTML gave us a very streamlined framework that benefited our project. We researched Django thoroughly through Youtube tutorials and things of that nature because we never had any experience prior to this project. On top of Django, we utilized Bootstrap 5 to format our many HTML files.
        - https://docs.djangoproject.com/en/4.0/
        - https://getbootstrap.com/docs/5.1/getting-started/introduction/
            - https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css
- Statement on the OOAD process for your overall Semester Project
    - List three key design process elements or issues (positive or negative) that your team experienced in your analysis and design of the OO semester project
    - Although it was both beneficial and difficult at times, the use of Django helped us tremendously in our project. It was a difficult learning curve but it opened the doors for us to better understand and create web applications. In terms of using both design patterns and analyzing how to solve certain problems, there seemed to be a feature in Django that we were able to utilize and make use of in our semester project.
    - One difficult experience was in overriding Django's default User object model. We wanted to include attributes other than the ones that were offered to save selections for drinks/movies/food. So finding a way to include a custom User object in the correct rank of our system's hierarchy definitely gave me some trouble.
    - Another issue we ran into was the use of extending in Django. Extending in Django allows for users to utilize inheritance in templates. It was useful because we didn't need to reuse code over and over again. However a couple of times we ran into an issue where a file getting extended (generic.html) would continually overwrite our other files. We tried using {% block content %} but at times we would run into the same issue. We decided to copy the code and put it directly into a file and that seemed to resolve most of the issues but it took us a while to figure it out.