CS335 AIRLINE SYSTEM

Mark McCormack & Rufus Tenali

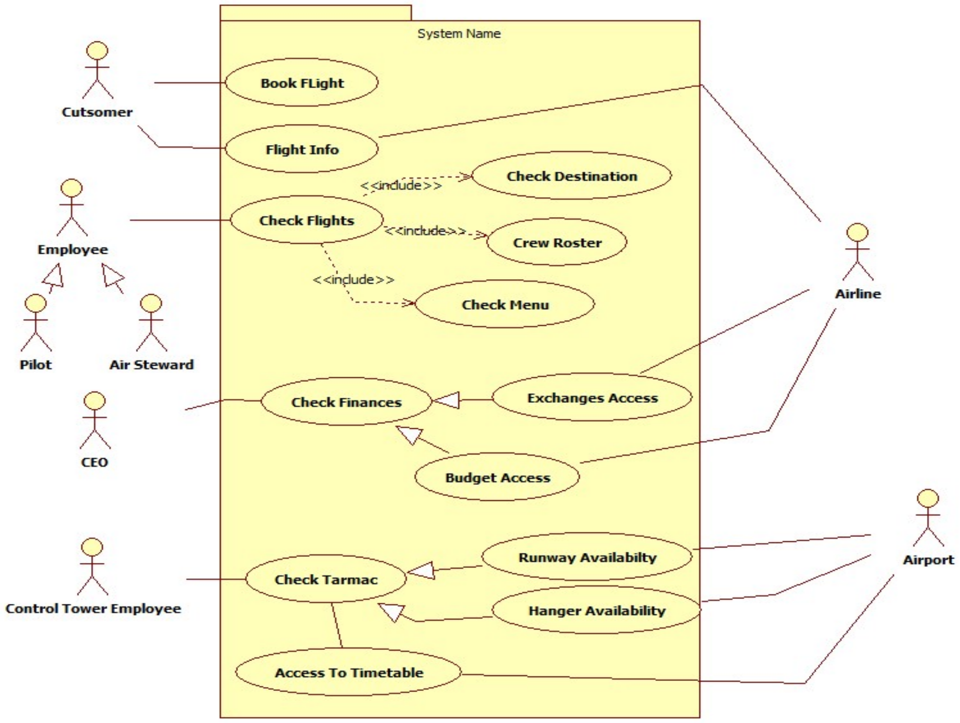
18370106 & 18378581

Abstract

In this project, we were tasked to design a system based on vague specifications from an end-user, turn them into UML diagrams and design a mock-up for the systems interface. We chose to design a piece of software for the “Transport” sector, basing it around the concept of an airport. Our end-user would want software to manage airport activities and provide services to their customers and employees.

One of our main challenges was to divide the activities between employees and the customers, deciding which parts of the system each could access. We decided in the end to have four types of employees; CEO, Control Tower Worker, Pilot and Steward/Hostess. We made these actors alongside the customer and designed the system based on this. We devised some User Stories as requirements and began work there.

Use-Case Diagram



Description

The purpose of the Use Case Diagrams was to create scenarios for Airlines. To give us an idea of what are the different kind of requirements that users of our app, would desire, to fulfil their needs. We had 5 different users: A Pilot, a CEO, a Control Tower Worker, an Air Steward and finally the Customer. We created 2 scenarios or Use Case statements for each actor, giving us 10 User Stories in total.

User Stories: *The User Stories were as follows:*

-> As a Pilot, I would like to know what my Destination is, so I can go to the right place.

-> As a pilot, I would like to know who my cabin crew and co-pilot are, as to talk beforehand and plan before our journey.

-> As a CEO, I would like to know my budget, so I can invest better in my Company.

-> As a CEO, I would like to have access to the Exchanges and Bookings, to supply refunds to customers.

-> As a Control Tower Worker, I would like to have access to the airline timetable, as to organize the runways and hangars, for inbound and outbound aircrafts.

-> As a Control Tower Worker, I would like to have access Hangars, Runways and other essential Resources to determine whether they are free or not to direct aircrafts around the tarmac.

-> As an Air Steward, I would like to know what plane I am serving, as to know where to go.

-> As an Air Steward, I would like to know the menu and resources for each Class, to know what to serve and to who.

-> As a Customer, I would like to book a flight, to travel somewhere.

-> As a Customer, I would like to know when and where my flight is, so I can prepare and board it.

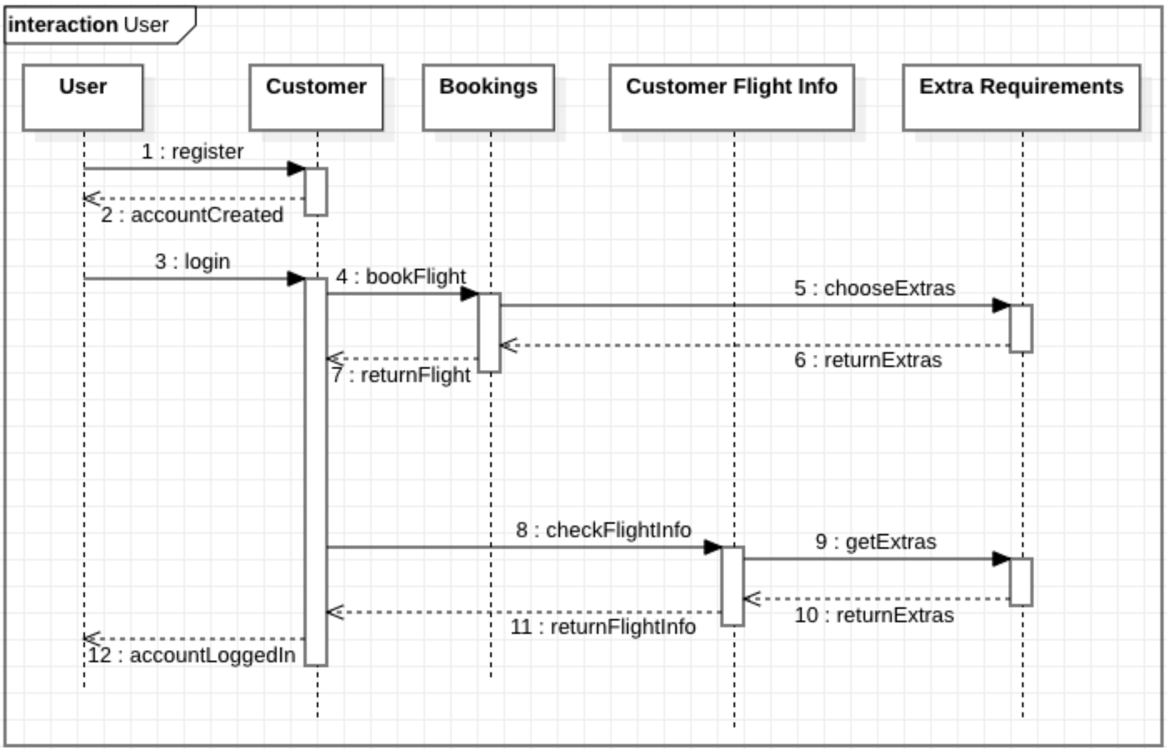
We called and came up with these different, possible User Stories which could be applicable in real life, in each circumstance to each user. With this in mind, we set out to design our Use Case Diagram on Star UML.

Creation of Use Case Diagram

The Customers, CEO, Control Tower Worker (CTW), Pilot, and Air Steward, were all Primary Actors for our application, as they interacted directly with the app. While the Airlines and Airport were Secondary Actors as they were more reactionary. They would react or rely information when the Primary Actors would interact directly with the application, seeking to complete a certain task and need information back.

I grouped Pilot and Air Steward together, making them both children classes to Employee, a parent class. Their job and requirements for the application, would be very similar, combatting unnecessary cases in the diagram. Creating this diagram helped us realize what kind of requirements users may need, how to group them and it gave us a lay out of how the application should behave when users interact with it.

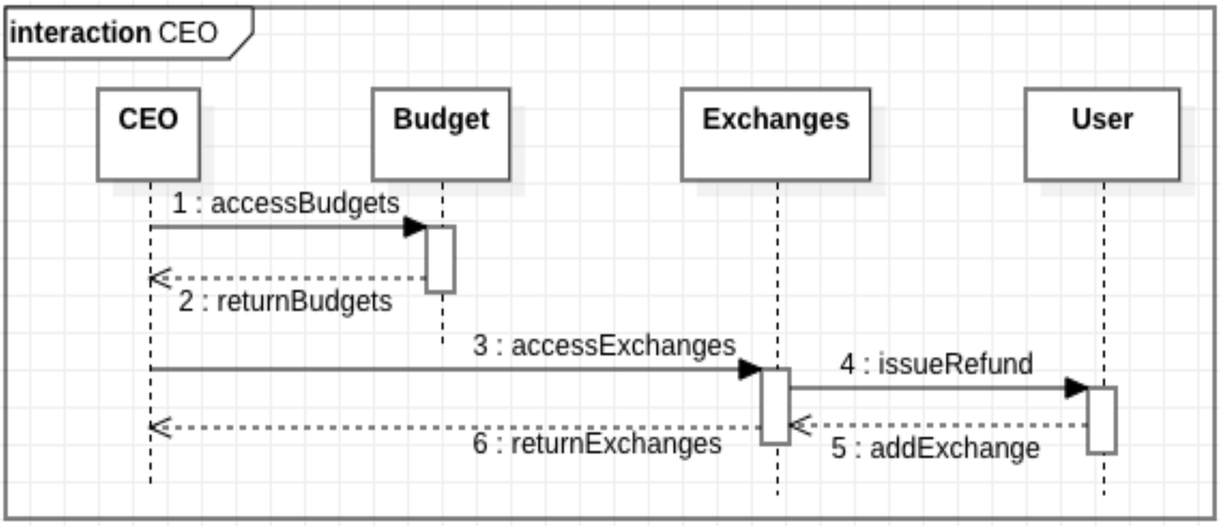
Sequence Diagram



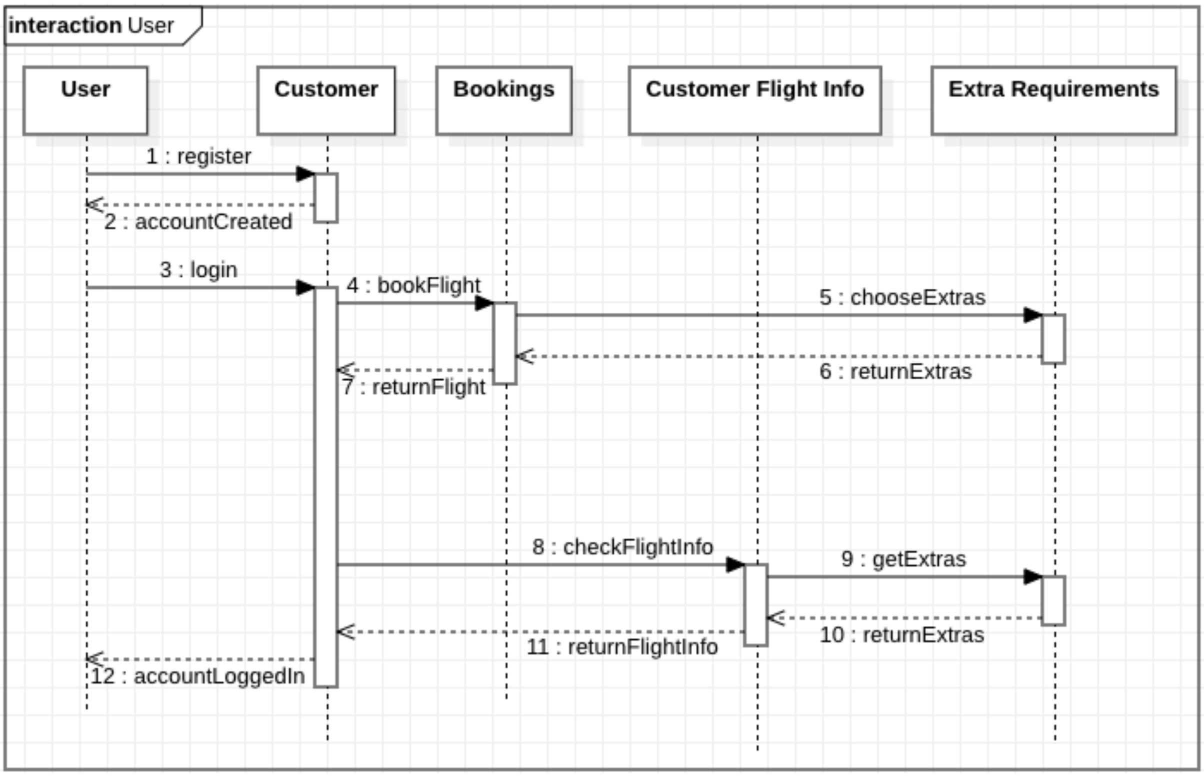
Description

> ***Intro:*** *The purpose of the Sequence Diagrams is describing how objects will interact with each other over time. We designed five Sequence Diagrams, each to fit an employee of the company or the customer object.*

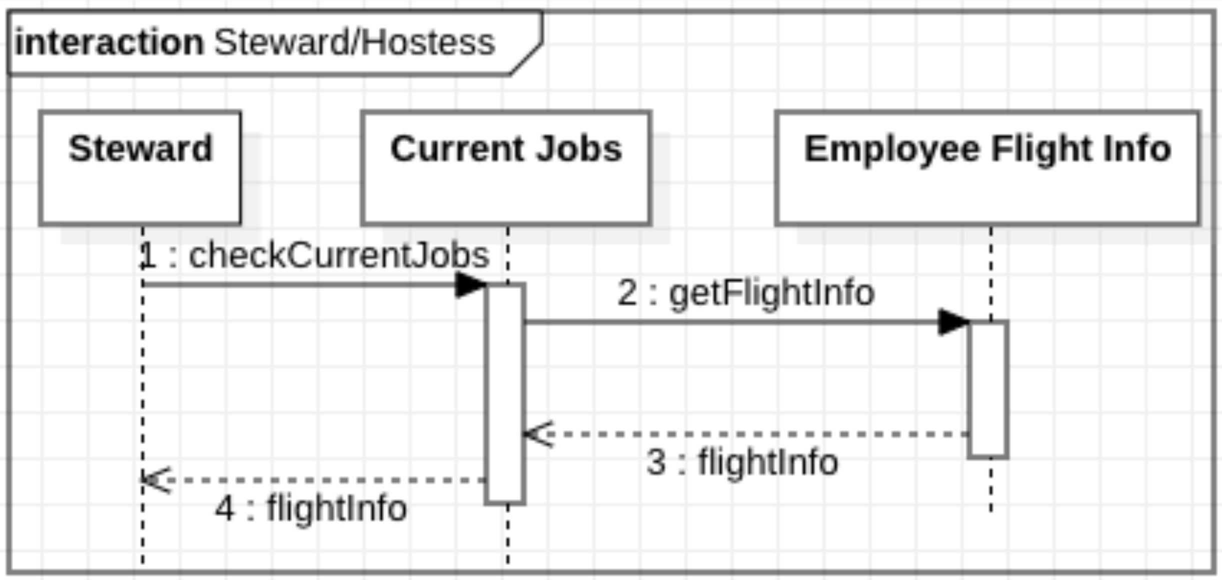
[Chief Executive Officer]: *The CEO may request the budget from the "Budget" object. They may also access the exchanges through the "Exchanges" object, which can issue a refund if requested and log it.*



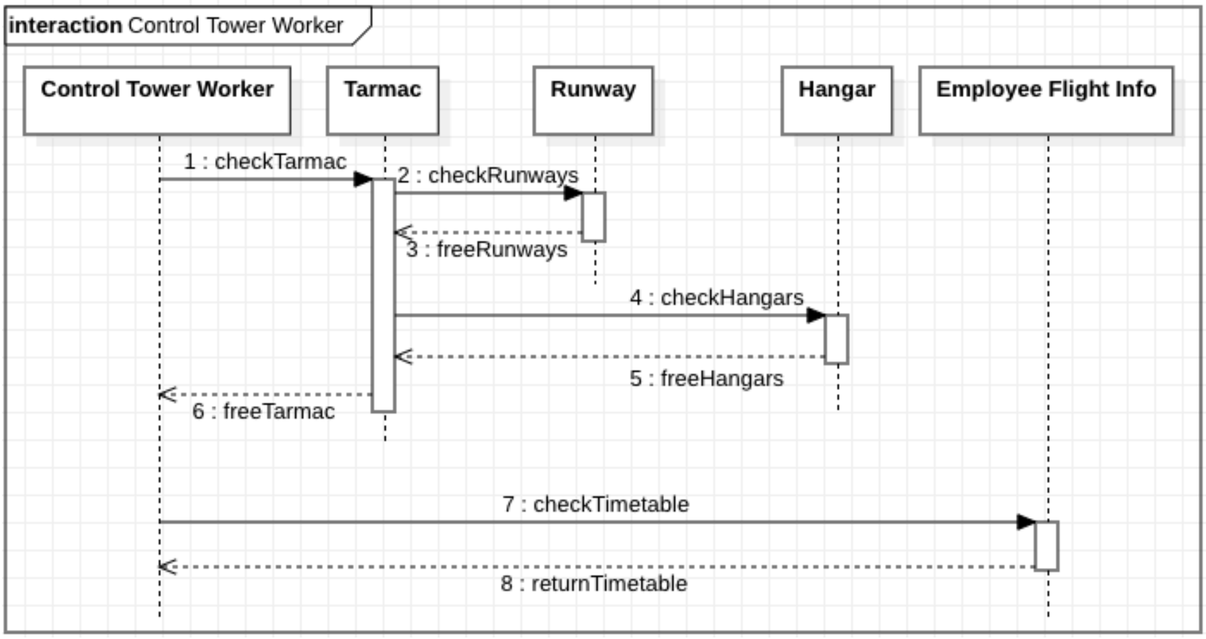
[Customer]: *A "User" may request to register, which will use the "Customer" object to create a new customer. A user may then login as a registered customer. The "Customer" object may then make bookings by requesting it from the "Bookings" object. This will confirm details and send a request to the "Extra Requirements" object, if the User will need any extras. A Customer may also check flight into, where they request data from the "Customer Flight Info" object, which then may request data from the "Extra Requirements" object and return it to the customer.*



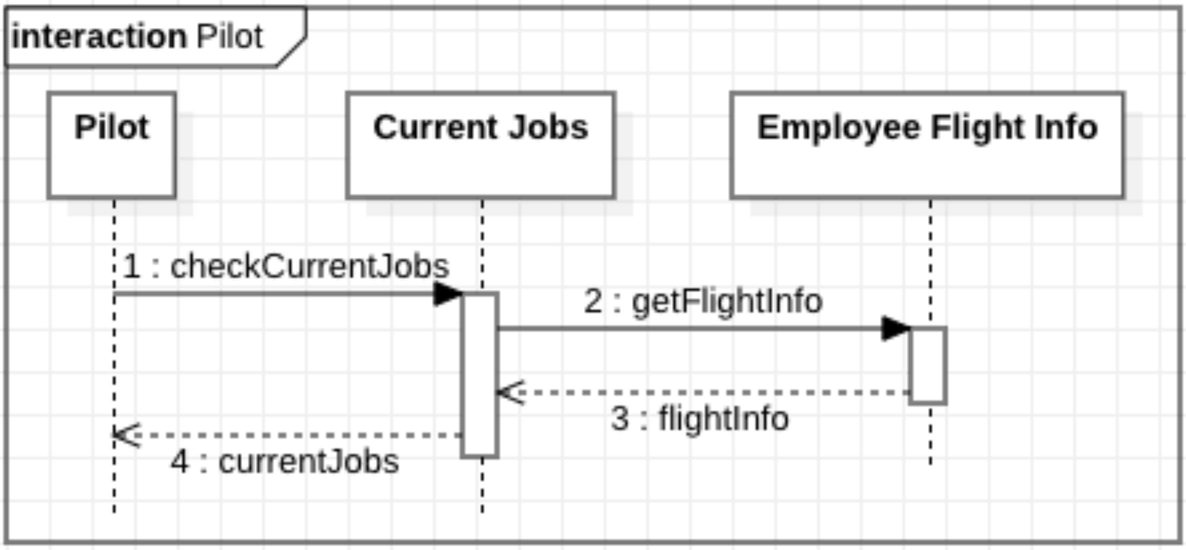
[Steward/Hostess]: *The steward needs a minimum number of functions, they only need to know what their current job is. The "Steward" Object may make a request to the "Current Jobs" object, which will find the job the "Steward" currently has, and request data from the "Employee Flight Info" object and return this to the steward.*



[Control Tower Worker]: *The Control Tower Worker is key to the airport system, and next to the customer was the user we examined the most. The "Control Tower Worker" may check the "Tarmac" object, which will check to see if any runways or hangers are free by requesting data from the "Runway" and "Hanger" objects. They may also check the timetable of flights, done by requesting the timetable from the "Employee Flight Info" so they can predict where planes will land and at what time so they can designate runways and hangers to them.*

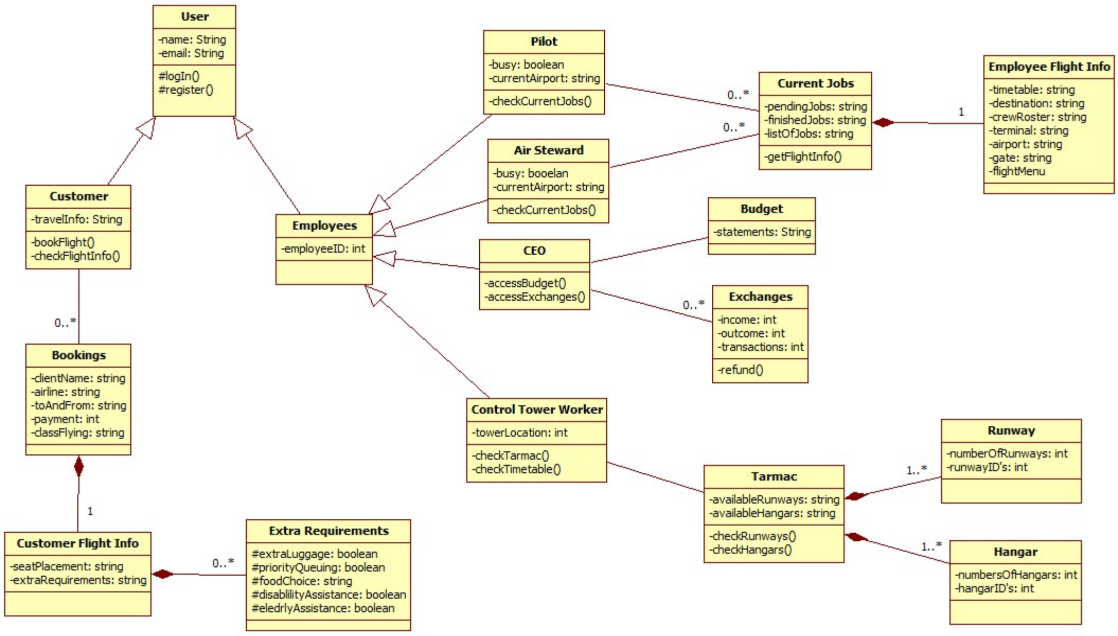


[Pilot]: *Similar to the steward, the Pilot will need minimal interaction with the system. They may make a request to the "Current Jobs" objects which will request info on their current job and return it to the Pilot. This will include the terminal and gate so they will know where to go for their flight.*



[Sequence Diagram Conclusion]: *We designed the Class Diagram before we looked at the Sequence Diagram, and this was beneficial as It let us see how we could optimise our system. Using the sequence diagrams, we could see objects that could be combined to make a simpler system.*

Class Diagram



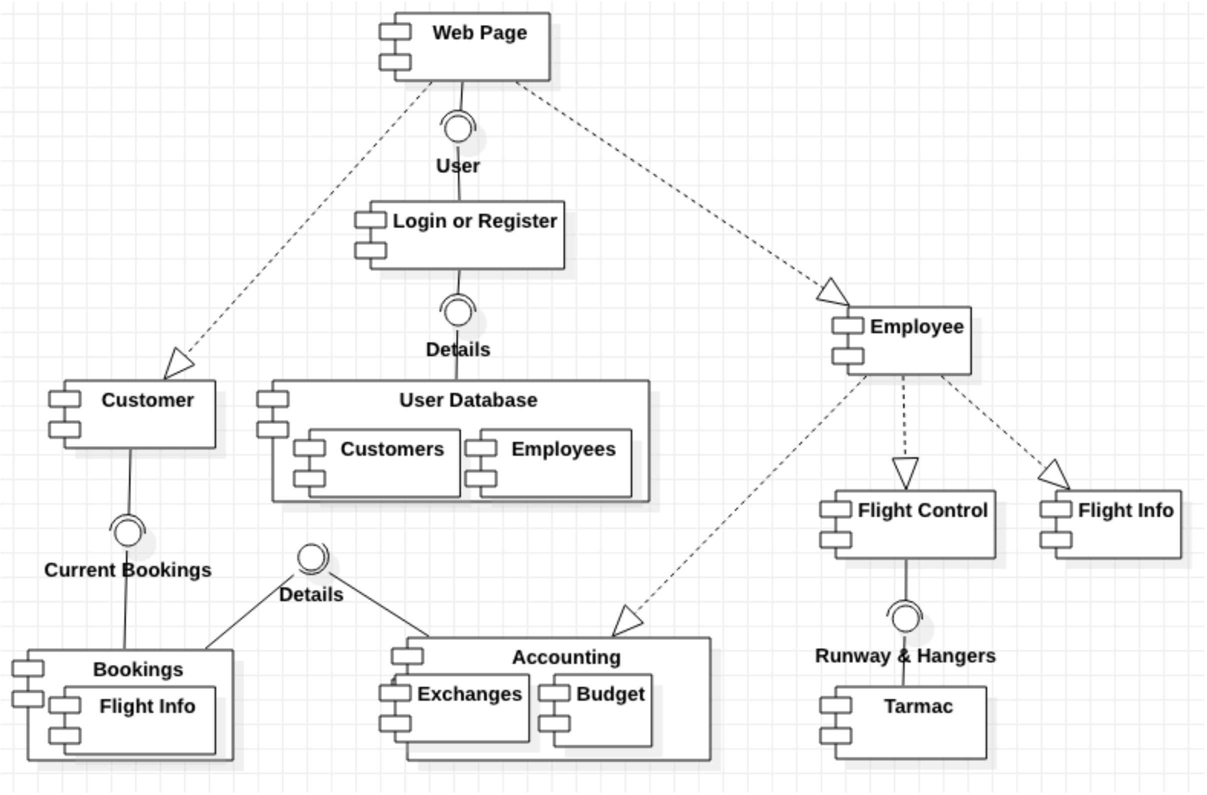
Description

> ***Intro:*** *With the Class Diagram, we were tasked with creating classes for all the different objects that would exist in our applications and the users would encounter it. We created each class and their own unique attributes, that would do several things, such as Identify them, pass on attributes to children classes, inherit attributes from parent classes and even have own unique methods to run different tasks.*

Classes: *Each class has different attributes which can have different visibility, most are either private or protected. Classes have these attributes which hold information specific for them or for their class type. Some classes have these methods inside of them to run certain tasks. These methods can be accessed inside these classes or inheriting classes only. For example, the Pilot can use the method checkCurrentJobs(), this allows them to run and call back and receive the information that they are seeking. Unlike the CEO, he can't use checkCurrentJobs(), as that is a method that is in the Pilot class, but the CEO can accessBudgets(). The relationship between the various classes are very vital as well. They determine how classes will interact with each other. Some can have a simple association relationship. Some are parent or child classes, meaning the child depends on the parent class to pass on attributes to it or methods. Some classes could not even exist without the upper class*

Understanding: *While doing this Class diagram it granted me a better understanding of how the classes and objects inside the application should behave and interact with each other. It really made me think, what certain classes should be able to do, how relationships affect the outcome and how to lay out and create easier layouts for the classes inside. Showing similarities in certain things and differences in others, really shaping how I saw these diagrams and, later on, would truly help with the User Interface.*

Architecture Diagram



Description

[Intro]: *The purpose of the Architecture Diagram is describing how we can package classes together for reuse in the future, as well as a high level abstraction of the system which can be presented to the end-user.*

User Interface Design

Description

Intro: *Going into this User Interface was quite the challenge. Having to create an interface which was user friendly and simple for people to use yet look pleasing. I looked and studied the lecture slides on bad interface designs understanding the different concepts and various boxes that must be ticked to determine if it was user friendly.*

Development: *I decided to use MockFlow. I was really enticed by their own user interface, accessibility and their large arsenal of different elements I could use to implement in my own app design. I enjoyed creating* *user interface and designing the application pages. However, sometimes I felt a need to step back and look at the pages differently. I asked myself questions for self-evaluation. If I were a user, would this entice me? Is it simple to use and understand what different parts do? Can I complete the task without having to ask for additional help? Can I navigate easy to where I need to go?*

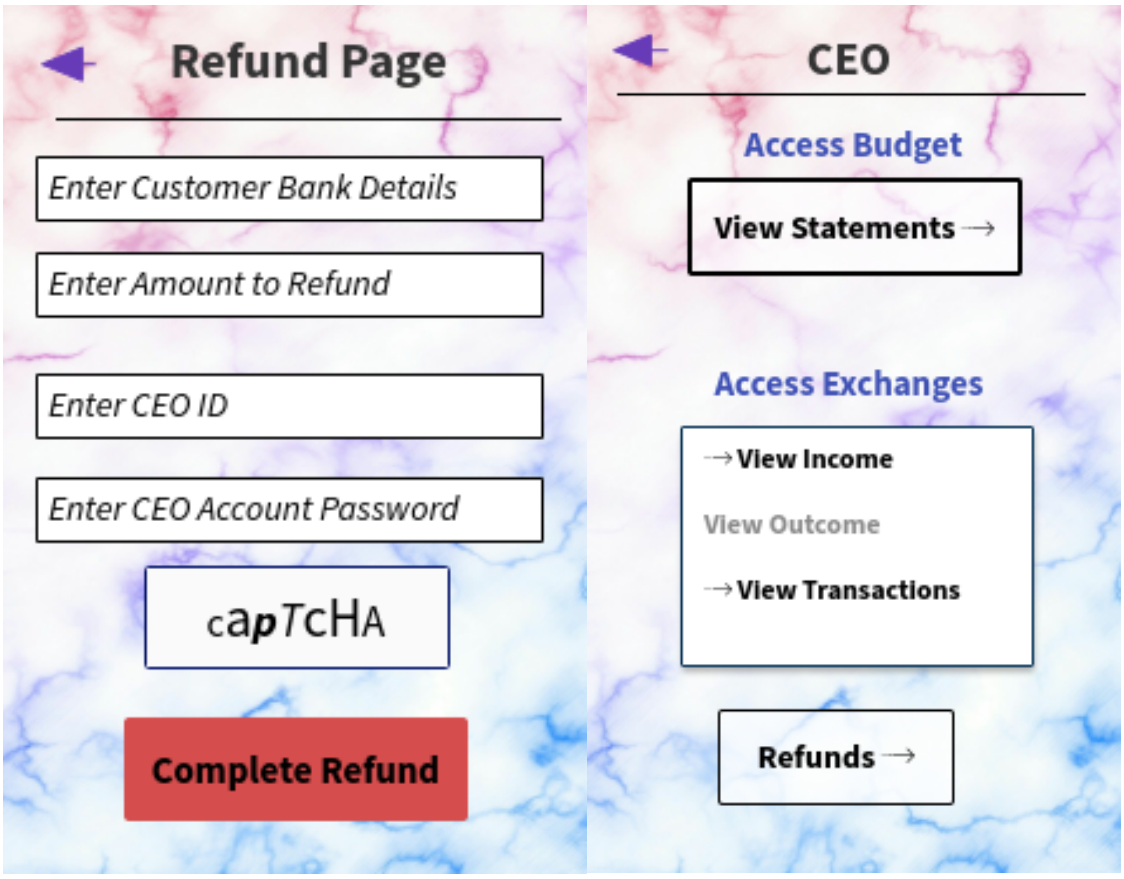
Testing: *I decided to do a little testing of my own. I asked family members and my friends online if they would look at my designs and give feedback and see if they would understand it. This gave me valuable information to change and tweak my interface. Introducing testing gave me understanding from a user's point of view, which can be easily lost while designing the process through your own eyes. Knowing what users desire and their preferred requirements, I think, boost proficiency and the quality of work. Showing the shining importance of testing and feedback to me.*

System Testing

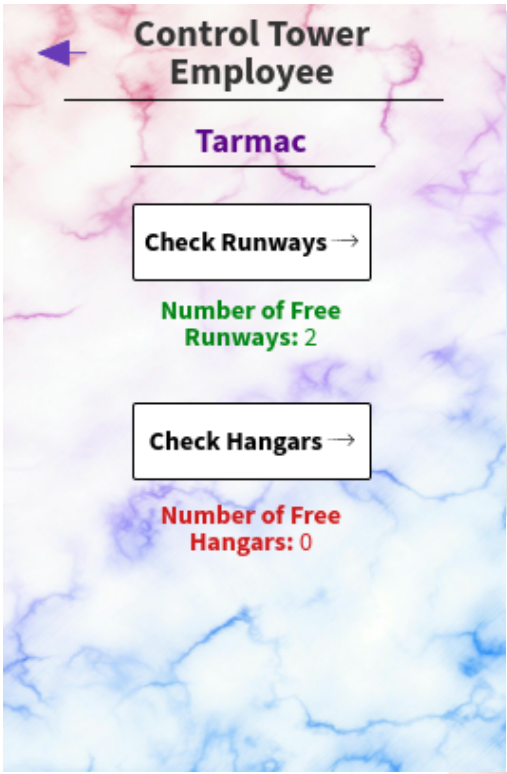
System Testing: *System testing is an important part of design, as it allows us to see what mistakes we may have made and account for them before deploying our software. We chose to use the "User Stories" we gathered earlier as test cases and designed our system tests around this. We choose what results would be expected given some test data as provided in the diagram below.*

No.	Test Cases	Test Data	Expected Results	Actual Result
1	Check if as a pilot I can access my current jobs.	Employee ID	Returns Current Jobs	PASS
2	Check if as a CEO, can I access the budgets.	Employee ID	Return Budget	PASS
3	Check if as Control Tower Worker, I can access timetables.	Employee ID	Returns Timetable	PASS
4	Check if as a Hostess, I can access my current jobs.	Employee ID	Returns Current Jobs	PASS
5	Check if as a Customer, I can book a flight.	Customer Email & Password	Returns Booking & Payment Interface	PASS
6	Check if as a pilot, I can access who my crew is.	Employee ID	Returns Flight Info	PASS
7	Check if as a CEO, I can access exchanges.	Employee ID	Returns Exchanges	PASS
8	Check if as Control Tower Worker, I can access resources.	Employee ID	Return Tarmac(Runways, Hangers)	PASS
9	Check if as a Hostess, I can check the flight menu.	Employee ID	Return Flight Info	PASS
10	Check if as a Customer, I can access my booked flights.	Customer Email & Password	Return Bookings	PASS

User Interface Design (Diagrams)



User Interface Design (Diagrams)



Control Tower Employee

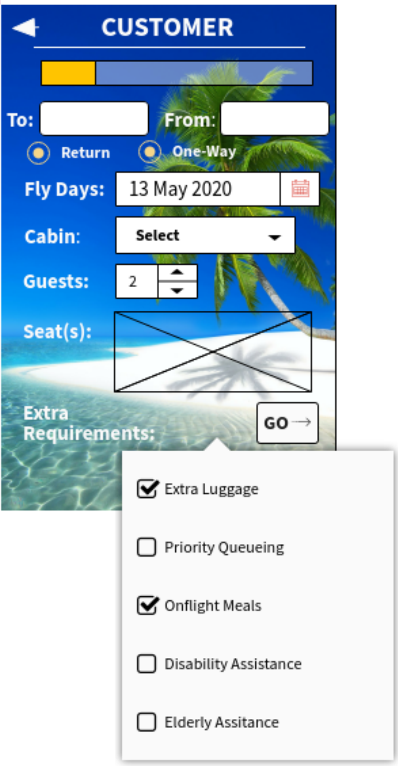
Tarmac

Check Runways →

Number of Free Runways: 2

Check Hangars →

Number of Free Hangars: 0



CUSTOMER

To: From:

◉ Return ◉ One-Way

Fly Days: 13 May 2020

Cabin: Select ▾

Guests: 2

Seat(s):

Extra Requirements: GO →

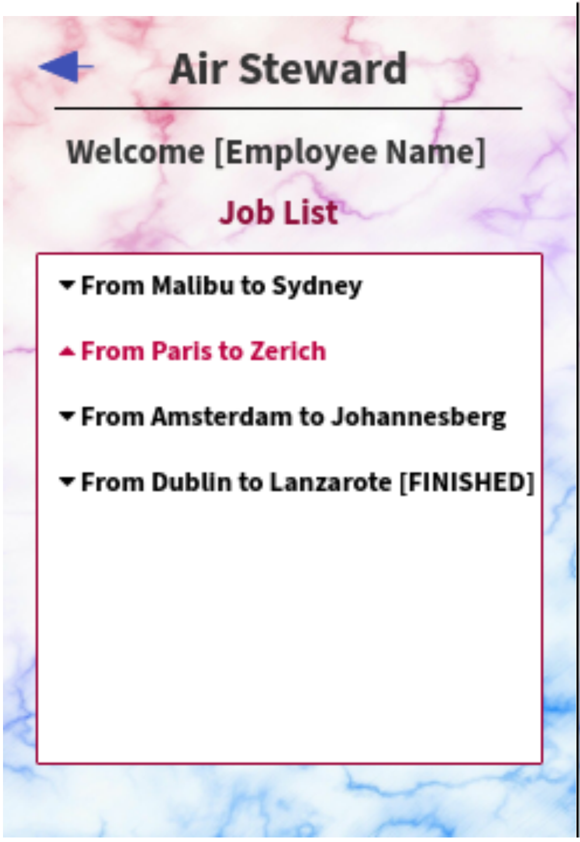
☑ Extra Luggage

☐ Priority Queueing

☑ Onflight Meals

☐ Disability Assistance

☐ Elderly Assitance

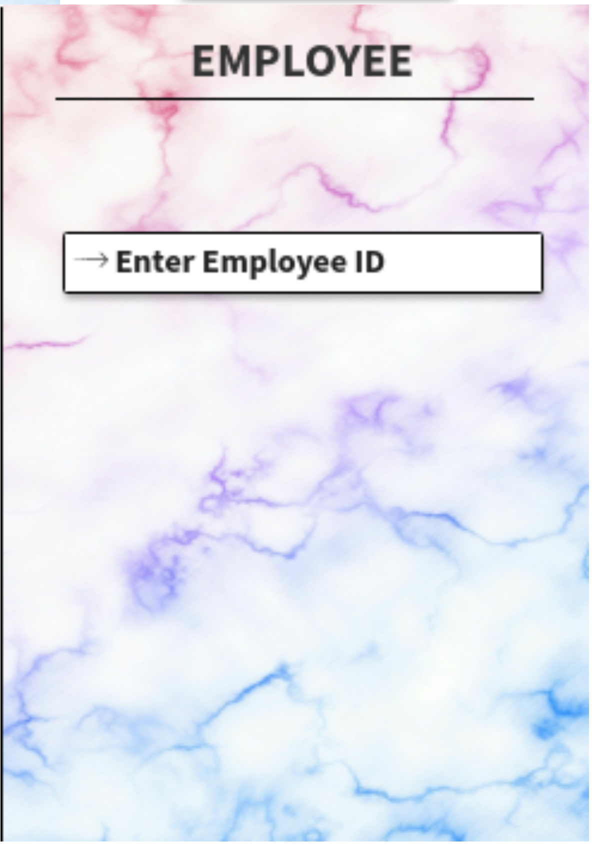


Air Steward

Welcome [Employee Name]

Job List

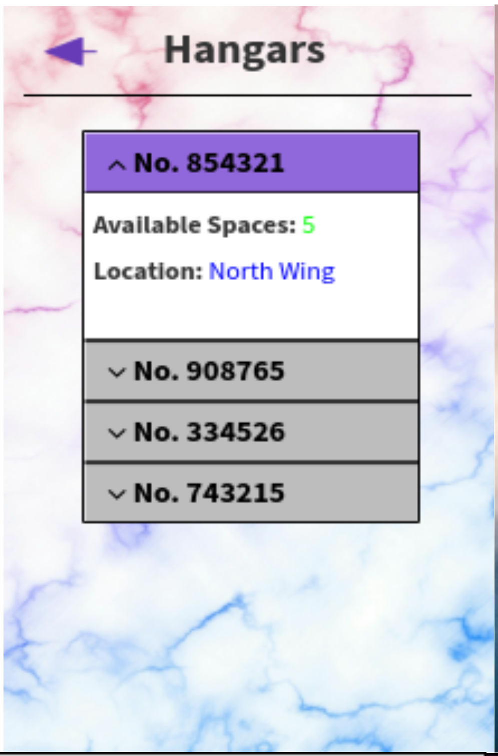
- ▾ From Malibu to Sydney
- ▴ From Paris to Zerich
- ▾ From Amsterdam to Johannesberg
- ▾ From Dublin to Lanzarote [FINISHED]



EMPLOYEE

→ Enter Employee ID

User Interface Design (Diagrams)



Hangars

No. 854321

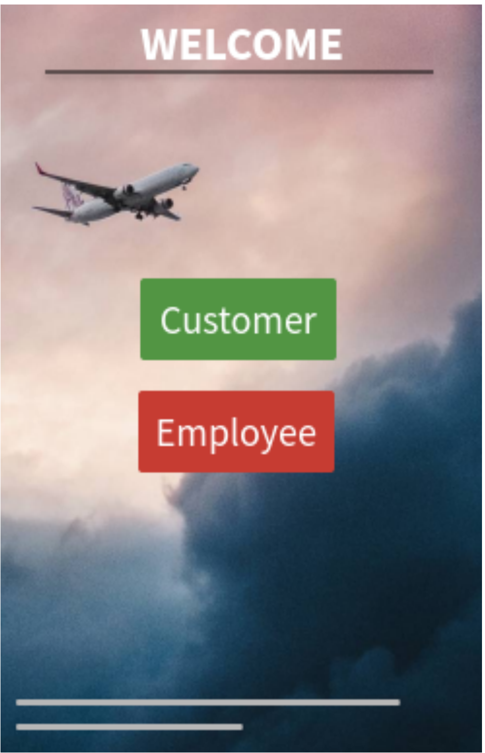
Available Spaces: 5

Location: North Wing

No. 908765

No. 334526

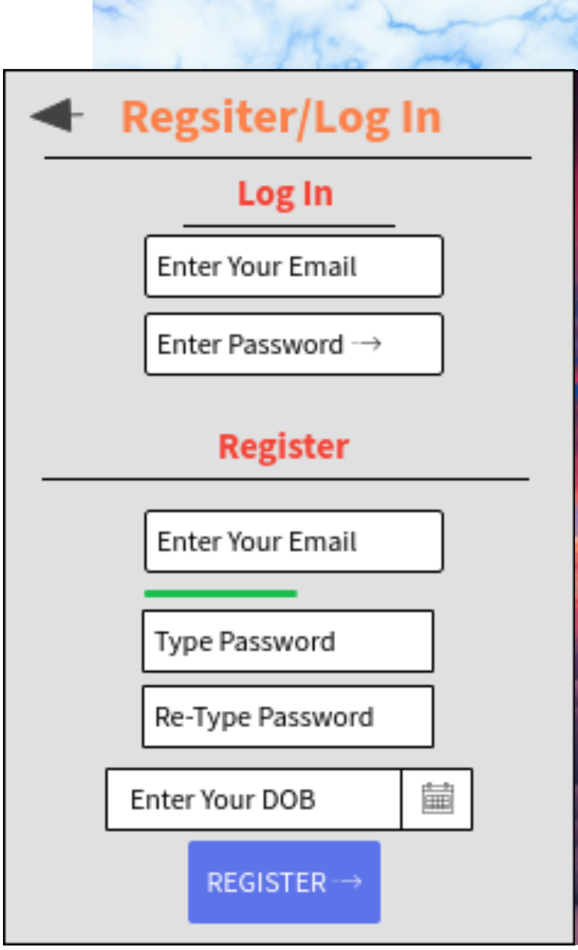
No. 743215



WELCOME

Customer

Employee



Regsiter/Log In

Log In

Enter Your Email

Enter Password →

Register

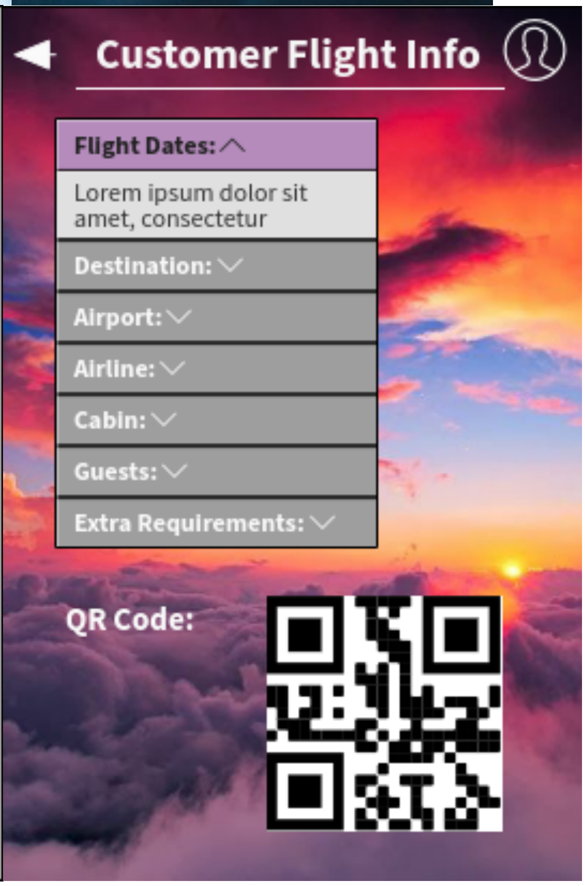
Enter Your Email

Type Password

Re-Type Password

Enter Your DOB

REGISTER →



Customer Flight Info

Flight Dates:

Lorem ipsum dolor sit amet, consectetur

Destination:

Airport:

Airline:

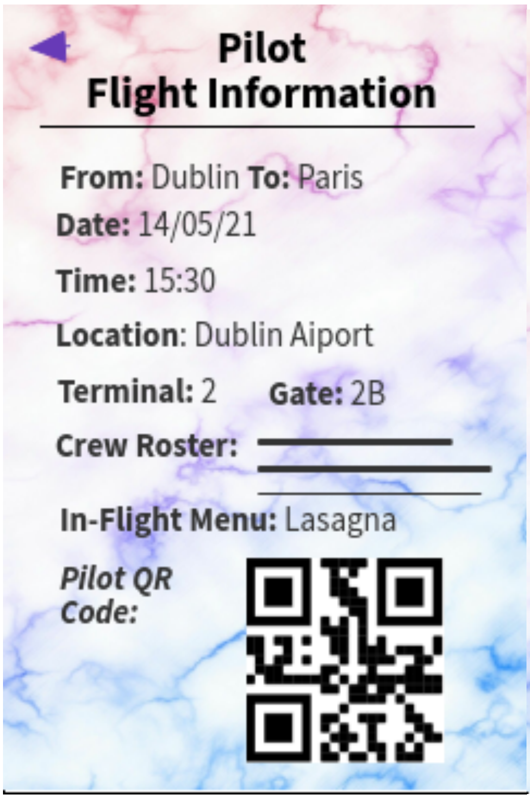
Cabin:

Guests:

Extra Requirements:

QR Code:

User Interface Design (Diagrams)



Pilot
Flight Information

From: Dublin **To:** Paris

Date: 14/05/21

Time: 15:30

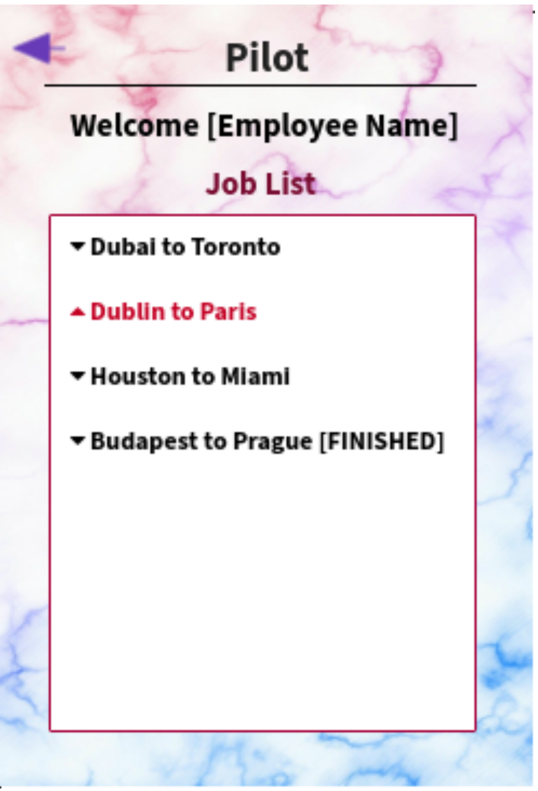
Location: Dublin Aiport

Terminal: 2 **Gate:** 2B

Crew Roster:

In-Flight Menu: Lasagna

Pilot QR Code:

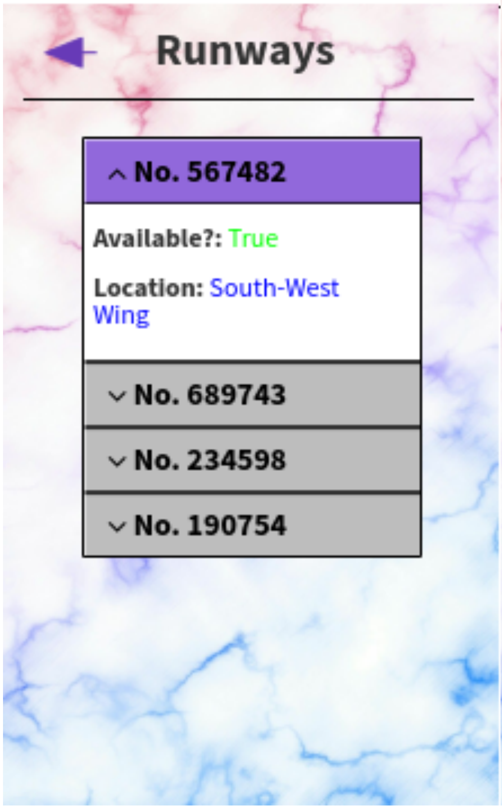


Pilot

Welcome [Employee Name]

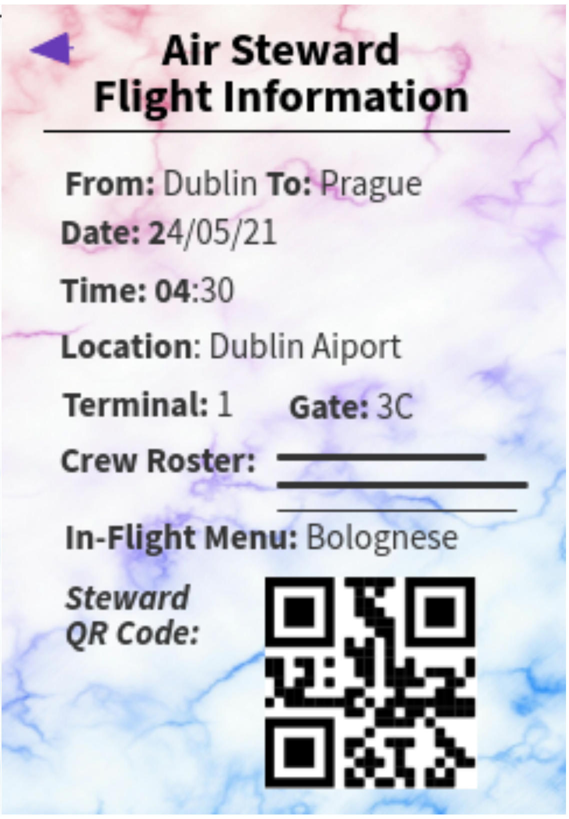
Job List

- ▼ Dubai to Toronto
- ▲ Dublin to Paris
- ▼ Houston to Miami
- ▼ Budapest to Prague [FINISHED]



Runways

- ^ No. 567482
  - **Available?:** True
  - **Location:** South-West Wing
- v No. 689743
- v No. 234598
- v No. 190754



Air Steward
Flight Information

From: Dublin **To:** Prague

Date: 24/05/21

Time: 04:30

Location: Dublin Aiport

Terminal: 1 **Gate:** 3C

Crew Roster:

In-Flight Menu: Bolognese

Steward QR Code:

Experiences

Mark McCormack

> ***Experiences:*** *This project was beneficial in many ways. First and foremost, I got hands on experience in the design of software for a pseudo-company, which will be beneficial to similarly natured projects in the future. Secondly, I found working with a colleague on this project to be productive, as it allowed both of us to delegate activities, increasing the quality of our software and the speed of our production. To conclude, the project was also rather enjoyable as it allowed us to be creative and system of our design. (Worked on Abstract, Sequence Diagram, Architecture Diagram & Test Cases)*

Understanding: *Upon finishing this project, I have become far more comfortable with UML diagrams, system architecture and UI design as a whole. Going into my next year of studies, I feel this will stand to me for future career opportunities.*

Also, becoming more familiar with software such as "StarUML" and the skills it has taught me have made me consider designing software as a part time activity, or perhaps a career.

Rufus Tenali

> ***Experiences:*** *Truly doing this project, creating the diagrams hands on, and designing the application granted me a higher perspective. Understanding the essential nature of these diagrams and their vital importance to software designing aspects.*
>
> *Learning of these different and various methods to plan out a project or an application idea in my head has spurred me to pursue Application designing further. Having learnt these basics, giving me a solid foundation, I desire to pursue this further in the future. I never thought I could receive this much enjoyment while doing a Graded project as such as this. (Worked on Use-Case Diagram, Class Diagram, UI Diagram & Test Cases)*

Understanding: *After working eagerly on this project, I've grasped a better understanding of how desirable these traits of planning out different diagrams and the processes truly are in the computer industry. Having a well-planned and thought out documentation and diagrams create an easier understanding of employer and customer needs and smoother and easier workflow if the steps and requirements are clearly pointed out.*

Not having anything left up in the air for indifferent interpretation. This pushes me to pursue software designing more and to put in action, the basics and foundation I have gained here. This was a great experience for us to gain perspective, understanding and a massive growth by putting our skills to action in a real-world application.

References

Google Images:

Main Page: https://ru.phoneky.com/wallpapers/?id=w44w2058131

Customer Page: https://cutewallpaper.org/21/ipad-beach-wallpaper/view-page-21.html

Customer Flight Info: https://m.duitang.com/blog/?id=537572531&belong_album=77303778

Employee Pages: https://wallpaperaccess.com/marble-pink