1.1 **FLIGHT BOOKING**

**Queens College Airline Booking System (QCABS)**

**prepared by: Queens College Software Engineers (QCSE)**

**prepared for** Queens College Air Control Inc. (QCACI)

**1.2 CSCI 370 SECTION 32, GROUP 3**

**Group 1**

Sagar Patel

Hang Lin

Araceli Castelan

Jaehoon Cho

Thayany Jeyakumaran

Carlo Ace S Sagad

1.3 **DESCRIPTION OF THE PROJECT**

This is a Requirements Specification document for a booking flight ticket for Queens College Air Control Inc. (QCACI). QCACI is preparing to construct an airline with one flight that can go between QC, New York, and UCLA, CA. QCABS will allow users to book flight tickets and make payments.

1.4 **Booking** **software for the people (ease of use.)**

this software will allow the general public to book and cancel flight bookings, as well as ad additional amenities to their bookings. Each customer will have a wide variety of bookings to choose from. They can choose whether to have a windowed first-class seat or choose an affordable economy seat. the software is a means to help the customer select their booking without any complications.

**QCABS will provide an online capability for users/students to book tickets and make payments.**

1.5 **Incentive of this project**

Queens College Air Control Inc. wants a very simplistic design in their software as this would let the users book and manage their booking with ease, This would mean that the customer would have full control over his/ her booking. The aspect of simplicity is predicted to increase the user base of the software by 80%, leading to more bookings. another positive aspect is that customers will run into fewer problems which means we need fewer people for customer service, cutting costs there.

1.6 **SCOPE OF THIS PROJECT:**

This project will allow the customer to book flights and manage bookings more easily.

**QCABS** does not include database services, **QCACI** will have to find its own customers.

1.7 **Acronyms**

1. **FQC is one of the well-known flights for its hospitality and security.**
2. **QCABS A software to sell flight tickets.**

**2.1 USE CASES:**

**Actor – The customer who will use the app to book tickets.**

**Preconditions – the customer has to select all the required fields before booking their tickets.**

**Triggers – Travel, tourism, business…**

**Main success scenarios [Basic Flow] – Customer uses the app, finds the flight time and preferred seat and for his/ her convenience**

**Alternative paths [Alternative Flow] – the selection of other amenities which will affect the cost.**

**Postcondition needs sufficient funds.**

2.2 <<Acceptance Criteria – One Page>>

*Will be used to determine when the project is done.*

*Include a bunch of values which are either pass or fail. At least one page.*

<https://www.productplan.com/glossary/acceptance-criteria/>

<https://agilepainrelief.com/blog/definition-of-done-user-stories-acceptance-criteria.html>

See chart example below:

i.e. Drivers can make an appointment from phone. Try to use metrics. i.e. system can handle 6 reservations at a time. i.e. Game will load in 15 seconds.

Acceptance Criteria example



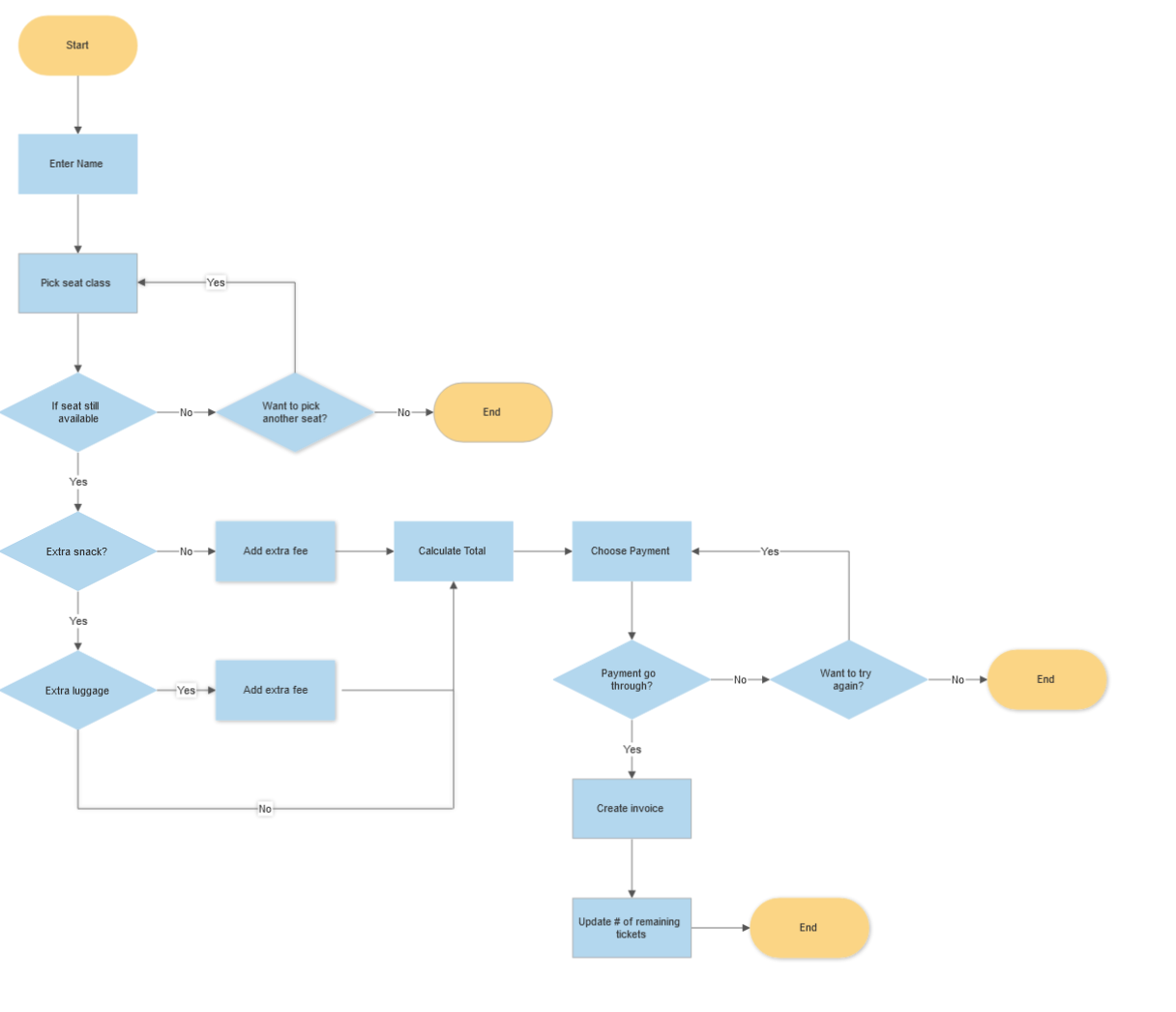
2.3 <<Assumptions and Constraints (one paragraph) >> i.e. Someone from Accounting will be available to test our releases and review progress. We have three months to finish the project. There are only five members of the team who know Java.

<https://wikispaces.psu.edu/display/SASPMO/How+to+Define+Project+Assumptions%2C+Constraints%2C+Dependencies+and+Critical+Success+Factors>

2.4**<<Platform Requirements Specification hardware/memory/Operating System requirements (***one paragraph max)>>*

*i.e. Need 1 GB of memory certain speed CPU/ use IPhone/Chrome Browser*

2.5 <<Context Diagram and the Process Flow Diagrams One each per Project>>



[*https://www.edrawmax.com/context-diagram/*](https://www.edrawmax.com/context-diagram/)

Example of a process flow diagram

<https://www.conceptdraw.com/How-To-Guide/picture/Business-processes-Cross-Functional-Flowcharts-Trading-process-Sample.png>

3.1 **Optional <<Architecture and Design Philosophy - no more than one page**

**Note: If you are not sure if you have any design pretend that you are and explain its benefits for your program See Chapter 6>>**

<https://www.simform.com/blog/software-architecture-patterns/>

1. This section discusses the key goals for your architecture and design that you followed while formulating your architecture and designing the major process components.
2. You should mention other architectural or design alternatives that were considered but which were eventually discarded.
3. Explain in what ways the chosen system architecture may be extended in the future to handle additional user capacity or system load. i.e., Is your project scalable?

**4.1 <<Design Patterns – Repeated once for each member of group>>**

**see this doc** [**What should be submitted**](https://docs.google.com/document/d/14PhnNRDL_C8QFJofS8Kusho2ZToyjFvb/edit?usp=sharing&ouid=114737963514957364787&rtpof=true&sd=true)

Make sure that your code follows the Java coding convention. That is that Types (or Classes) are capitalized while everything else is lower case and then camel cased afterwards.

**For group member, please include the following**:

4.2 << Student Name>>

4.3 << Design Pattern Name>>

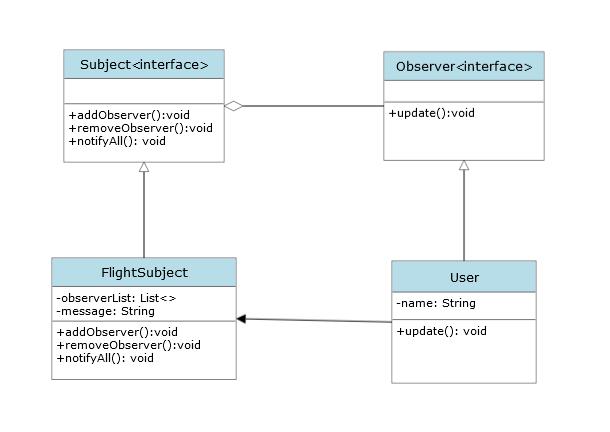
4.4 << UML diagram for design pattern using **your** class names>>

4.5 << Java Classes that implement your design pattern (should be only a few)>>

4.6 << Code for two unit tests and the paragraph for 1 component test >>

4.2 Student Name: Hang Lin

4.3 Design Pattern: Observer Pattern

4.4 UML diagram for design pattern using **your** class names:

4.5 Java Classes that implement your design pattern: FlightSubject.java, Subject.java, Observer.java, and User.java.

4.6 Code for two unit tests and the paragraph for 1 component test :

Sample of two unit tests.

void testPositive() {  
 Math m = new Math();  
 *assertTrue*(m.add(1,2)==3);  
}@org.junit.jupiter.api.Test  
void testNagative() {  
 Math m = new Math();  
 *assertTrue*(m.add(-1,2)==1);  
}

Example of component test –

User Starts program. Garage has 100 parking spots available. 5 spots taken. There should be 95 spots available.

5.0 Zip File containing working Java code for group project.