Airline Crew Scheduler

Software Design Document

Team 8 - Skywalkers
Rahul Prajapati
Dipal Bhandari
Aniruddh Saxena
Shivani Tamkiya
Alexis Saltzman

Introduction

Purpose

The purpose of this system is to create scheduling software for Cornhusker Airlines, which will enable the airline crew administrator to schedule flights, assign airplanes, pilots, and other crew. The system will also allow crew members to search for their schedules, and the public to search for flight takeoff and landing times.

Design Goals

After designing and deploying the ACS, the airline will have improved reliability in data retrieval. To test reliability, the accuracy, precision, and recall will be measured and compared to values the client deems acceptable. Because of the repercussions for incorrect or incomplete results, these values need to be very close to 100%. This system will also have the ability to backup and restore all scheduling data, which reduces the risk of downtime. For the crew administrator, after login the system will provide the ability to create new flights and allocate resources for each flight, track employees hours, track flight takeoff and landing times, edit flight information and cancel flights. Additionally, crew will also be able to login to search for their assigned flights and track their working hours. As an extension, customers and guests will be able to search for takeoff and landing times of flights.

Design Trade-Offs

The accuracy of results and useability of the product for employees who are not technically proficient is of utmost importance. Security, to prevent intruders from changing, adding, and deleting flight information at great cost to the airline is also vital. Cost to build, due to increased quality control and system testing and revising, may be affected to ensure that these important specifications are fulfilled. The size allowance for entries in the database, and the size of the database as a whole, will all for more detailed recording that users will be able to refer back to, but longer entries and a larger database mean access time could increase beyond what is acceptable for users.

Interface documentation guidelines

- Make status of system visible to user with mouse symbol such as spinning wheel when system is working. on a process, blinking cursor when system is ready for input.
- Classes are named with singular nouns, i.e. “Person” instead of “Persons” or “People”.
- Classes are capitalized.
- Methods are lowerCamelCase and are named with verb phrases.
- Fields are lowerCamelCase and are named with noun phrases.
- Error status is returned as exception. Exceptions created for all possible errors, user should not see an internal error message but a user-friendly message to tell them what to do next instead. Anticipate that if the user can input anything, they will eventually try to input anything.
- All input will be constrained, sanitized, and validated for type, length, format, and range with regular expressions before being submitted to the database.
- If an input field has a limited or knowable number of acceptable inputs, utilize a drop down box or radio button to allow data to be normalized.
- A verification window must show and require “OK” from user before input is submitted to prevent mistakes.
- User guide documentation will be easily accessible to the user from the main menu.

Definitions, Acronyms, and Abbreviations

CHA - Cornhusker Airlines

ACS - Airline Crew Scheduler

Captain - Qualified pilot for a particular aircraft

First Officer - Qualified pilot or co-pilot for a particular aircraft

Flight Attendant - Crew member responsible for the safety of the passengers in the main cabin for the duration of a flight.

GBR-10 - Type of aircraft, capacity 45 passengers

NU-150 - Type of aircraft, capacity 75 passengers

References

Group 8 Requirements Analysis Document

Object Oriented Software Engineering, 3rd edition. Bruegge, Dutoit

Firebase Authentication (https://firebase.google.com/docs/auth/)

Firebase Realtime Database documentation (https://firebase.google.com/docs/database/)

Joda Time API (https://www.joda.org/joda-time/)

Overview

The scheduling system should be able to keep track of the CHA crew members. The system should be able to assign correct position and number of crew members required for each type of airplane. Furthermore, the system should be able to keep track of both the estimated and actual time of takeoff/touchdown of the airplanes. The system should also be able to log in all the updates made to the schedules and these updates should be accessible for searching based on the flight number. The system will be comprised of a website interface as well as a smartphone application with a database backend.

Current Software Architecture

This section will remain blank because there is not information available on current system.

Proposed Software Architecture

Overview

Our primary architecture is a client server architecture. But instead of hosting it to a local server we're using firebase tools which changes our whole architecture to a serverless architecture. A serverless architecture eliminates the need of server and hardware management by the developer to host the application, in our case the database and authentication. Instead it hosts it on a third party service like the firebase tools. We're using the real time database and authentication functionality of firebase.

The system is dividedinto the following subsystem

1) LoginSubsystem: is assigned to provide authentication by using firebase authentication tool for all the employees and passengers of CHA.
2) DatabaseSubsystem: is assigned to provide a real-time database through firebase.
3) RoutingSubsystem: is assigned to calculate the flight timing for the airports.
4) SchedulingSubsystem: is assigned to schedule the flights and the employees.

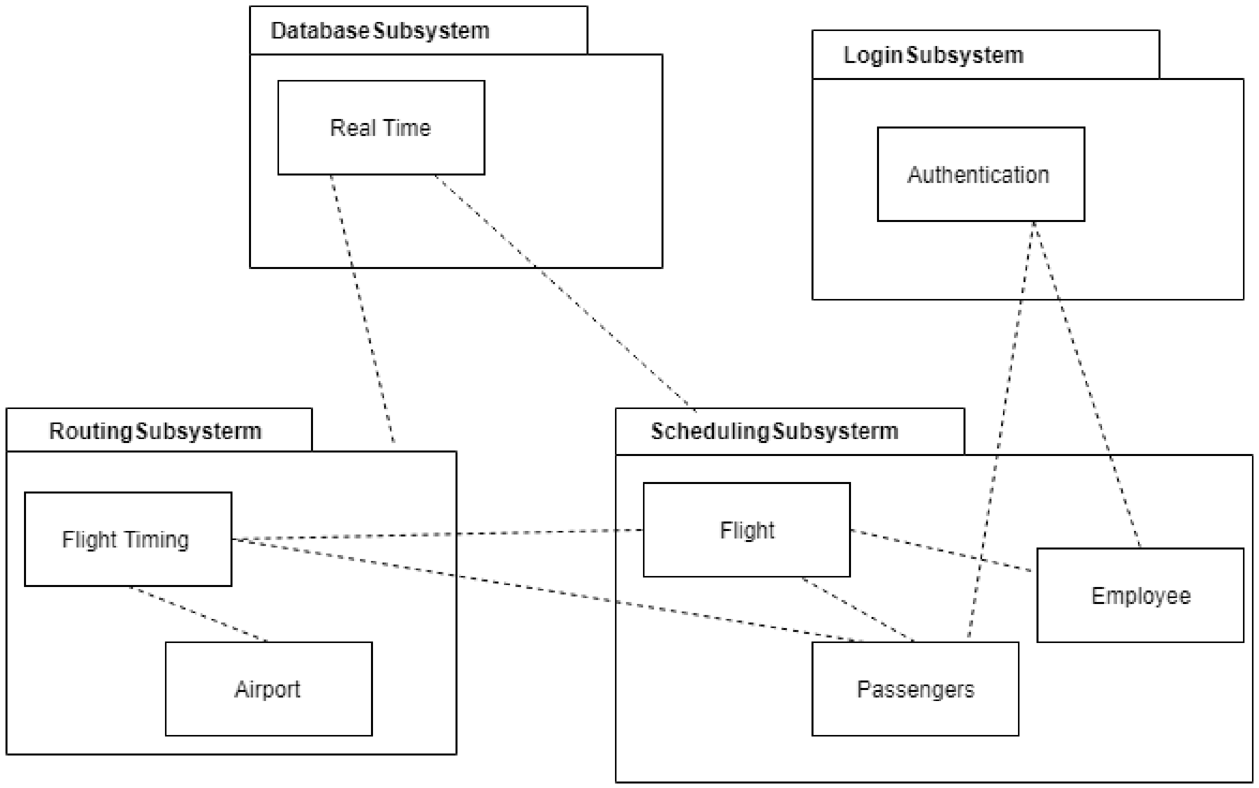
Subsystem decomposition

The System is divided into the following subsystems-

1) Login Subsystem: is responsible for providing login and authentication functionality through firebase tools. Which allows the employees as well as the passengers to login.
2) Database Subsystem: is responsible for storing and providing the data of CHA in realtime through the firebase tools.

3) Scheduling Subsystem: is responsible for scheduling the flights and the employees through the realtime database. This subsystem would be able to schedule the aircrafts and the crew required for the aircraft to operate.

4) Routing Subsystem: is responsible for flight timing providing both the actual and the estimated time for take off/landing time of the flight for all the airports that CHA operates on.



Hardware/software mapping

Our model does not require hardware/software mapping.

Persistent data management

"The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data." (Firebase Realtime Database documentation, https://firebase.google.com/docs/database. Accessed Oct 17, 2018.)

Access control and security

“Most apps need to know the identity of a user. Knowing a user's identity allows an app to securely save user data in the cloud and provide the same personalized experience across all of the user's devices.
Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more.” (Firebase Authentication, https://firebase.google.com/docs/auth. Accessed Oct 17, 2018.)

Global software control

As there is potential for two clients to attempt retrieving information simultaneously, each server is made event-driven in order to avoid the race condition between them. Control flow is distributed within the Airline Schedule System and each services behaves on its own control flow.

Boundary conditions

Boundary conditions give the overview of how the system starts and what services need to be initialized first. The Firebase server handles all the data retrieval, so this server has to be started at the beginning as the client provides services to this server and retrieves the information from it. In our design, the Manager has control over the server . He/She is the one who manages all the databases.Once the server starts , the manager has to detect whether there is an error during the start of the server. With the help of Firebase analytics manager, we can detect the error and report the problem to the maintainer.

Subsystem Services

LoginSubsystem

Provides authentication for CHA Employees. Upon method calling login() the authentication process requires the user to login, then the credentials used are verified from the database record.

DatabaseSubsystem

Provides real time database through firebase tool. Some methods that are called upon the DatabaseSubsystem are as follows
Flight in SchedulingSubsystem calls getflight() , setflight(), createFlight(), cancelFlight() on Database Subsystem to fetch/change the desired data.
Employee in SchedulingSubsystem calls getcrewmember(), setCrewmember() etc to assign crew members to the flights which is via the database.
Flight Timing in RoutingSubsystem calls setActualtakeofftime(), getEstimatetakeofftime(), setActualtouchdowntime() on DatabaseSubsystems to update the actual/estimated takeoff/landing time.

SchedulingSubsystem

This subsystems handles the non functional requirement of calculating timings for flights and crew member. By using real time database it will always update the time schedule as soon as managers enter the data related to flight. It has abilities to setEmployeetype() of a flight, getWorkinghrs() of employee from the realtime database, getWorkingschedule() of the employee.

RoutingSubsystem

It provides the functionality of calculating the correct timing for flights on the basis of distance from one place to another. It will update the scheduling subsystem and adds the traveling time to maintain other non functional requirements. Provides information about the flight through databasesubsystem to the SchedulingSubsystem, crew assignment from the SchedulingSubsystem. RoutingSubsystem uses google-map API for finding the distance and duration of flight between the two airports.

"Distance Matrix API is a service that provides travel distance and time for a matrix of origins and destinations, based on the recommended route between start and end points."(Google maps API, https://developers.google.com/maps/documentation/distance-matrix/start, Accessed 17, Oct 2018)

Packages

Yada

Class Interfaces

Yada

Detailed Design

Yada

Glossary

Actual Takeoff/Landing – the precise time at which the aircraft takeoff/landed.
Administrator – Someone who has all the access over the system.
Aircrafts – vehicles operated by the Airways.
Captain – a senior pilot who commands the crew of an airplane.
Estimated Takeoff/landing – predicted time at which the aircraft might takeoff/land.
Functional Requirements – describes the functionality of the system.
Grounded – An aircraft not being used for a while.
Non-Functional Requirements – User level requirements including usability, reliability and implementation.
Pilot – a person who flies or is qualified to fly an aircraft.