Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description** |
| 1.0 | 2019.07.17 | Tamas Attila Horvath | Modified the order of system features, and aligned them with the diagrams |
|  |  |  |  |

# 1. Intorduction

## 1.1 Overview

This document defines the requirement for the <app>. The purpose of this document is to represent the system requirements in a readable way so that clients and stakeholders can understand them and verify them for correctness but with enough detail that developers can design and implement a software system from them.

This document doesn’t address *project* issues such as schedule, cost, development methods, development phases, deliverables and testing procedures. Those are addressed in a separate project document and quality assurance test plan.

The <app> is a web and Bluetooth based iOS application for travelers who doesn’t travel alone, to save money while travelling with low budget airlines. People don’t have to buy their seat to be able to sit next to each other, they can negotiate from the moment they received their free random seat, even during the flight.

## 1.2 Goals and objectives

1. Provide interface for users to select their flight.
2. The application should notify the user to check in the flight, when it is available depending on the company.
3. Provide interface for users to check in the seat they got.
4. Users should also see other users’ seats, these are open for negotiation.
5. Users should be able to contact other users.
6. The application should send notifications if there is a possibility to change seats.

## 1.3 Scope

The <app> will use an API to get flight data to be able to notify the user when they can check in. It will store it’s own data in a cloud-based database to be able to connect people.

## 1.4 Definitions

**Use case** – describes a goal-oriented interaction between the system and an actor. A use case may define several variants called scenarios that result in different paths through the use case and usually different outcomes.

**Scenario** – one path through a use case

**Actor** – user or other software system that receives value from a use case.

**Role** – category of users that share similar characteristics.

**Product** – what is being described here; the software system specified in this document.

**Project** – activities that will lead to the production of the product described here. Project issues are described in a separate project plan.

**Shall** – adverb used to indicate importance; indicates the requirement is mandatory. “Must” and “will” are synonyms for “shall”.

**Should** – adverb used to indicate importance; indicates the requirement is desired but not mandatory.

**May** – adverb used to indicate an option. For example, “The system may be taken offline for up to one hour every evening for maintenance.” Not used to express a requirement, but rather to specifically allow an option.

**Controls** – the individual elements of a user interface such as buttons and check boxes.

**Check in** – By this term, the airline check-in procedure is meant, which is independent from this application

**Update seat information** – After checked in, the user gets a random seat (in most cases). This term means to upload the given random seat to the application.

## 1.5 Document Conventions

Portions of this document that are incomplete will be marked with TBD. Each TBD item will have an owner and estimated date for resolving the issue. There may be any kind of change in the document itself, even structural changes or addition of new points.

# 2. General Design Constraints

## 2.1 Product Environment

The application will run under iOS, the lowest version it has to support is TBD. It will use TBD database to store it’s own data, like user authentication data, statistics and TBD. The application will be in relation with it’s own cloud based database and the flight database provided accessed by the TBD API which will be used. The application also has to use APN service, for sending remote push notifications to a user based on another user’s actions.

## 2.2 User characteristics

There won’t be any distinguished user, every user who wants to use the app will be equal (for now, in the future there might be premium users). Every registered user should have full access to the application.

## 2.3 Mandated Constraints

The application must run on iOS and must be be written is Swift 4 language.

# 3. Nonfunctional Requirements

## 3.1 Operational Requirements

The application has to allow different users to view and modify its data concurrently, while preserving the accuracy of the data. After the seats are exchanged, the system should indicate it immediately to other users. The applications shall have two modes, non-flight mode and flight mode. If there is no internet connection on startup or meanwhile the application is running, or airplane mode is enabled on the phone, flight mode should be enabled. This means that the features which requires external databases or internet connection has to be disabled.

## 3.2 Security Requirements

User authentication data should be stored securely, using hash values. Google?, Facebook, and other common authentication methods should be accessible. Users shall also be able to create account manually.

## 3.3 User interface

The user interface shall consist of native controls, animations shall help the users to figure out what happened on the screen (changing tabs, screens, etc.). The application shall be intuitive, logical so most of the users who used iOS applications before shall be able to use the application without any documentation or tutorial.

## 3.4 Software interface

The application will use have two main interfaces to the outside, it will have it’s own cloud based database, in which the users’ information will be stored, and it will also use an API to get the flight information of the selected flights by the user.

# 4. System features

System features in this paragraph will be listed as a mixture of narratives and use cases from perspective of the users. These are not in priority order, but in the order that users will use the application.

## 4.1 User login

At first start, user shall create an account using authentication types listed above in point 3.2, or create it’s own account manually. After that (after the first successful login), the application should remain logged in until the user chooses to log out. If the user exits the application, login data shall be stored, and the application shall log in automatically once the user opens the application again.

## 4.2 Main screen

The start screen shall consist of a list, which contains the flights the user registered on. On the beginning, this list shall be empty. There shall be a button (preferably a + icon) to register on a flight (add a flight), and another (preferably a -) to unregister (delete a flight). One list item shall contain the following information: flight number, departure time. A ‘>’ icon shall be present on the right side of the list item, to indicate that it has more details. If the user clicks on it, the application loads the details of the flight. If it is available to check in on any of the flights, a notification shall pop up, indicating that based on the airline’s check-in policy, the it is possible to check in with random seats.

## 4.3 Register on/unregister from a flight

If the user clicks on the add flight button, an input field should appear, so that users can enter the choice of their flight. The application shall check the input flight number in the external database accessed by the API mentioned in point 2.1. If the flight exists, the application should accept the user’s choice, and add it to the list with the information detailed in point 4.2.

## 4.4 Flight detail screen

This screen contains the flight number and the logo of the airline, if the users already updated their seat information, the seat number(s) also should be visible. If they did not update it, and check-in is not available yet (depending on the airline, how much time before do they allow you to check in with random seats), there shall be a timer which indicates when they can start the check in. If it is available to check in, an alert shall pop up, indicating that based on the airline’s check-in policy, the it is possible to check in with random seats. In the alert window a button shall be present with a link of the airlines application (check feasibility and legal info).

## 4.5 Update seat information

The user can update seat information using the notification sent by the application or on the detail screen of a flight. If check-in with random seat is possible, the users shall be able to pick their and seat(s), and if they travel with a group, the seat(s) of their group members. This is important, because that means that you want to sit together. The application doesn’t have to check if the seat is already occupied or not, as it is possible that users from a group join later, after someone from that group already indicated that their seats pertain to their group. In that case, a notification shall be sent to the user creating that group first.

## 4.6 Check seats

Users can proceed to the check seats screen, to check the seats occupied by the other users of this application, only if their seat info is already updated. The new screen shall contain all the seats of that specific airplane, symbolizing the seats with rectangular shapes with colors. The colors shall indicate the state of that specific seat. Used colors:

* grey – unused, not occupied by any of the users of this application
* green – updated by the user
* blue – updated by other users
* red – users already exchanged seats, may not want to change again

## 4.7 Chat

The users shall be able to chat with each other, in order to negotiate the seat changes. When internet connection is available, this conversation should go on internet, when it is not available(when the mobile phone is on airplane mode), the application should try to use Bluetooth instead flight mode.

## 4.8 Seat exchange agreement

If the users agreed on some exchanging the seats, a form shall appear, where the users have to enter their own seats, and the seats they want to change for. For the other user this form will be sent as a notification, and has to agree on the change. After that, the flight screen should indicate that there was a change already for those seats.

//TODO: Specify online and offline(bluetooth) modes, what services shall be available and how. Flowcharts shall be done as well.

Contents

[1. Intorduction 2](#_Toc14281599)

[1.1 Overview 2](#_Toc14281600)

[1.2 Goals and objectives 2](#_Toc14281601)

[1.3 Scope 2](#_Toc14281602)

[1.4 Definitions 2](#_Toc14281603)

[1.5 Document Conventions 3](#_Toc14281604)

[2. General Design Constraints 3](#_Toc14281605)

[2.1 Product Environment 3](#_Toc14281606)

[2.2 User characteristics 3](#_Toc14281607)

[2.3 Mandated Constraints 3](#_Toc14281608)

[3. Nonfunctional Requirements 4](#_Toc14281609)

[3.1 Operational Requirements 4](#_Toc14281610)

[3.2 Security Requirements 4](#_Toc14281611)

[3.3 User interface 4](#_Toc14281612)

[3.4 Software interface 4](#_Toc14281613)

[4. System features 4](#_Toc14281614)

[4.1 User login 4](#_Toc14281615)

[4.2 Main screen 5](#_Toc14281616)

[4.3 Register on/unregister from a flight 5](#_Toc14281617)

[4.4 Flight detail screen 5](#_Toc14281618)

[4.5 Update seat information 5](#_Toc14281619)

[4.6 Check seats 6](#_Toc14281620)

[4.7 Chat 6](#_Toc14281621)

[4.8 Seat exchange agreement 6](#_Toc14281622)