Scope of Work

CITS3200 Team 15 – UniPark Booking System

Preface:

This document outlines the requirements and scope of work for the UniPark booking system.

Target Audience:

Client and Developers

Team 15 Members:

- Angus Shaw
- Chauntelle Bonser
- Haoyuan Li
- Nur ‘Iffah
- Qiulan Huang
- Thomas Cleary

Project Description:

UWA’s UniPark office would like a web application to allow approved users from various UWA departments to place bookings on ‘30 mins unless reserved’ car bays for staff members in that department.

The main idea is that each ‘department’ has an elected staff member contact the UniPark office via email to request an account be made for them to book bays.

UniPark through the application can then create an account for the staff member to log in to the booking site. This staff member is then responsible for placing the bookings of staff members in that department. UniPark should be able to revoke access of a user if they are found to not be following booking guidelines.

These bookings do not require approval from UniPark and can be edited / deleted by the user who placed them (or the UniPark admin account).

The generated PDF reservation sign should be emailed to the respective user when a booking is placed, not the UniPark admin. The department / user is responsible for displaying these signs on the respective booking day.

This design removes the administrative burden of approving / denying booking request on the UniPark office whilst also relieving UniPark officers from the job of placing out the reservation signs.

Project Requirements:

- UniPark staff can create user accounts for various UWA staff members
    - Client value = $25

- The application should only allow UWA staff who are given accounts by UniPark to access the application. The application should list a list of these users (contact information), sorted by department so that a UWA staff member can contact them to make a booking on their behalf.
    - Client value = $20

- The application should send an A3 pdf reservation sign to the user's email address once they have placed a booking.
    - Client value = $17

- Users can view available bays, organised by car park, and the times / dates that they are available and make a reservation on behalf of a UWA staff member from their department.
    - Client value = $15

- Each bay viewed in the application should have a link to the Google Maps location of that bay and an image of a map displaying the location of the car park it is in.
    - Client value = $13

- Users can edit / delete bookings they have made. UniPark can also manage these bookings.
    - Client value = $10

Project Extensions:

- In app google maps intergration of some kind.

- Converting the application to a single page web application.

- Some form of API for digital signs to access and display reservation information.

- Intergrating with UWA's payment system to charge for bay bookings.

1.0 General Goals:

The overall goal of building the system is to relive the UniPark office of the work associated with making a car bay reservation for a UWA staff member.

The system will aim to allow Users to make bookings without UniPark having to approve or deny such requests. It will move the responsibility of placing reservation signs from UniPark staff to UWA staff making these bookings.

2.0 Current System:

The current system used by UniPark to manage bay reservations is a simple Excel spreadsheet. At present, for a UWA staff member to book a bay for a guest the process is as follows:

- The UWA staff member must send an email to the UniPark office requesting a reservation for a certain date and time.
- A UniPark staff member attempts to validity of the request (not always possible) and replies via email with availabilities for that date and time.
- The UWA staff member replies via email with their preference.
- A UniPark staff member approves the reservation, replies with relevant information to the UWA staff member and updates the spreadsheet with the new booking.
- On the day of the booking, in the morning UniPark officers display A3 reservation signs at the relevant bay. As staff are limited in number, these signs remain there all day and they do not contain pertinent information regarding the time of the booking or the individual whom the booking relates to.

3.0 Proposed System:

3.1 Overview

The proposed solution to this problem is to develop a web application that only authorised Users can access. These authorised users will be able to book bays on behalf of staff in their relevant department at UWA. These bookings will not require UniPark authorisation as to relieve the burden of administration work from the small team. The application will also send generated pdf reservation signs to the authorised user who will be responsible for it being displayed on the relevant date.

3.2 Functional Requirements

3.2.1 Account creation and management to facilitate the access authorisation of the application

3.2.2 A view of when each car bay is available for a specified date

3.2.3 The ability to book a car bay for a specified date and time

3.2.4 The ability to edit / delete a booking

3.2.5 The ability to generate an A3 reservation pdf from booking information and to be sent to the relevant users email address

3.3 Non-functional Requirements

3.3.1 A simple, easy to use user interface

3.3.2 The design of the application should conform to similar UWA website styles

3.4 Documentation

3.4.1 The application code must be documented well to ensure future maintenance can be performed (possibly by new developers). This documentation should be a technical overview of how the system is designed and works. Explanations of code structure / decisions should also be included.

3.4.2 A manual to describe how to install the system should be provided. This would mainly be aimed at more technical users that are familiar with deploying web applications. A simpler document on how to run the application on a local machine could also be included for the less tech savvy.

3.4.3 A manual describing how to use the system as a user should be created so that those wanting to use the system can better understand how to use / access it. This should be provided to UniPark, for their benefit and so that they can distribute it to the departments at UWA that will use the application.

3.5 Hardware Considerations

The application will likely be hosted on a cloud server, so no physical hardware other than developer machines and client / user machines will be needed to use / access the system.

3.6 Quality Issues

3.6.1 Possible Quality Enhancements

- Converting the application to a single page web application to allow for a smoother user experience (no page reloading).
- Using a separate frontend framework (instead of Flask templating) to build a nicer UI
  - Smoother animations
  - An overall more attractive UI
- Ensuring the web application has a mobile friendly UI
- Hosting the application on a reliable server so that there is no 'spin up time' when the site has not been used in a while

3.6.2 Possible Quality Compromises

- Forgoing a mobile friendly UI and focusing on desktop browsers
- Hosting the site on a platform that puts the application to 'sleep' if not accessed for a certain amount of time (longer initial app loading time).
- Forgoing a quality UI to be able to get the base functionality implemented. This would be the biggest concern when faced with time constraints.

3.7 System Modifications

3.7.1 Integrating UWA's payment platform

3.7.2 Integrating with digital signs to replace the A3 reservation signs (this may require an API to be built on top of the existing application).

3.7.3 Enhancing the UI by decoupling the frontend from the backend. For example, using a framework like React to build the user interface whilst using the existing Flask backend to service the data needed to construct the user interface.

3.8 Security Issues

3.8.1 Access to the application must be controlled by the UniPark office. Only user accounts created by UniPark should have access to the application.

3.8.2 User passwords should not be stored in plaintext in the applications database but as a salted hash. This will ensure if the database is accessed with authorisation, then user credentials could not be ascertained.

3.8.3 The server should be run over HTTPS to ensure secure connections between users and the server the application is hosted on.

3.8.4 Any input given by the user should be thoroughly cleaned / checked to prevent against attacks such as SQL injections.

3.8.5 All forms in the application should include a ‘secret token’ to prevent against CSRF attacks.

3.8.6 The database and environment variable files should not be hosted on the applications GitHub repository to ensure that no one unauthorised can gain access to this information. (Database contains user information and .env files will contain admin passwords and information).

4.0 Constraints:

4.1.1 Programming Languages

Backend

- Majority of the development team is familiar with Python. This will be used to build most of the application backend.

Frontend

- HTML, CSS, JS need to be used to create the frontend as this is what all browsers use.

4.1.2 Libraries

Backend

- Of the team members who have taken CITS3043 Agile Web Development, all three have only used the Flask framework. This will be the framework used to build the backend of the application.

Frontend

- Bootstrap and Flask templating will be used initially to build the frontend of the application as that is what most of the group is familiar with. It is also relatively simple to learn, allowing members of the team who are unfamiliar with frontend development to get up to speed quickly.