# QA Golf Club Tee Booking

This README.md file contains all the relevant information regarding my project within my first week at the QA Consulting Academy.

### Links:

**Kanban Board:**

https://jackhealy.atlassian.net/secure/RapidBoard.jspa?rapidView=3&projectKey=TEEB&modal=detail&selectedIssue=TEEB-16&atlOrigin=eyJpIjoiYzg1ZjA1YjFiMjE2NDk3OWEyNTQ5OThjMDYxMzY3MWUiLCJwIjoiaiJ9

**Website:**

<http://34.105.137.234:5000/>

## Contents

* Requirements
* Architecture
* Project Tracking
* Testing
* Risk Assessment
* Front-End Design
* Known Issues
* Future Improvements

## Requirements

The purpose of this assignment was to create a CRUD application with utilisation of supporting tools, methodologies and technologies that encapsulate all core modules covered during training. The following must be used/included within the application:

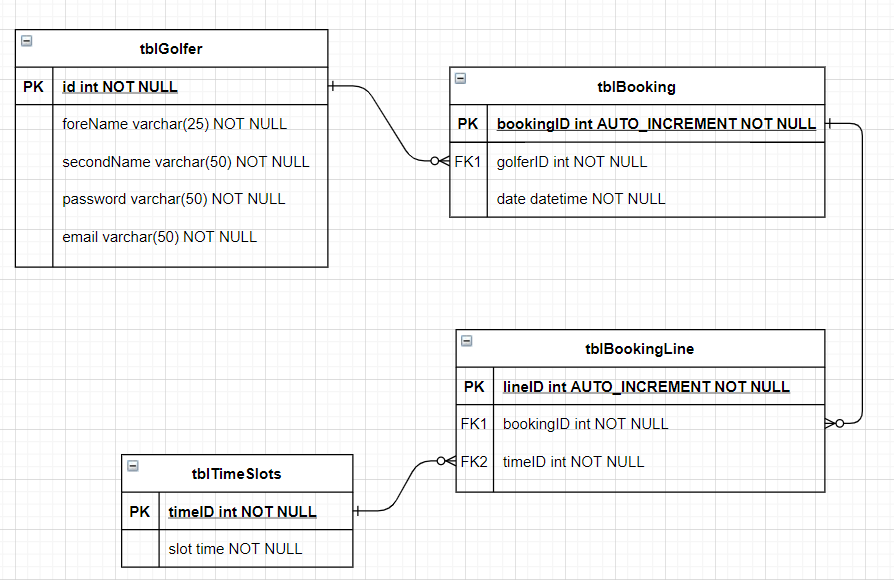
* A Kanban board (using Jira)
* A relational database, consisting of at least two tables that model a relationship
* Clear documentation of the design phase, app architecture and risk assessment
* A python-based functional application that follows best practices and design principles
* Test suites for the application, which will include automated tests for validation of the application
* A front-end website, created using Flask
* Code integrated into a Version Control System (GitHub) which will be built through a CI server (Jenkins) and deployed to a cloud-based virtual machine (GCP)

To fulfil these requirements, I decided to develop an application for a user to create, read, update, and delete tee bookings for a golf course. Users can:

* Create tee bookings by choosing available times via a python select field.
* View their bookings on their accounts web form once bookings have been made.
* Delete their bookings via a python select field if they decide not to play golf anymore.
* Update their bookings via a python select field if they decide to change their tee time.

## Architecture

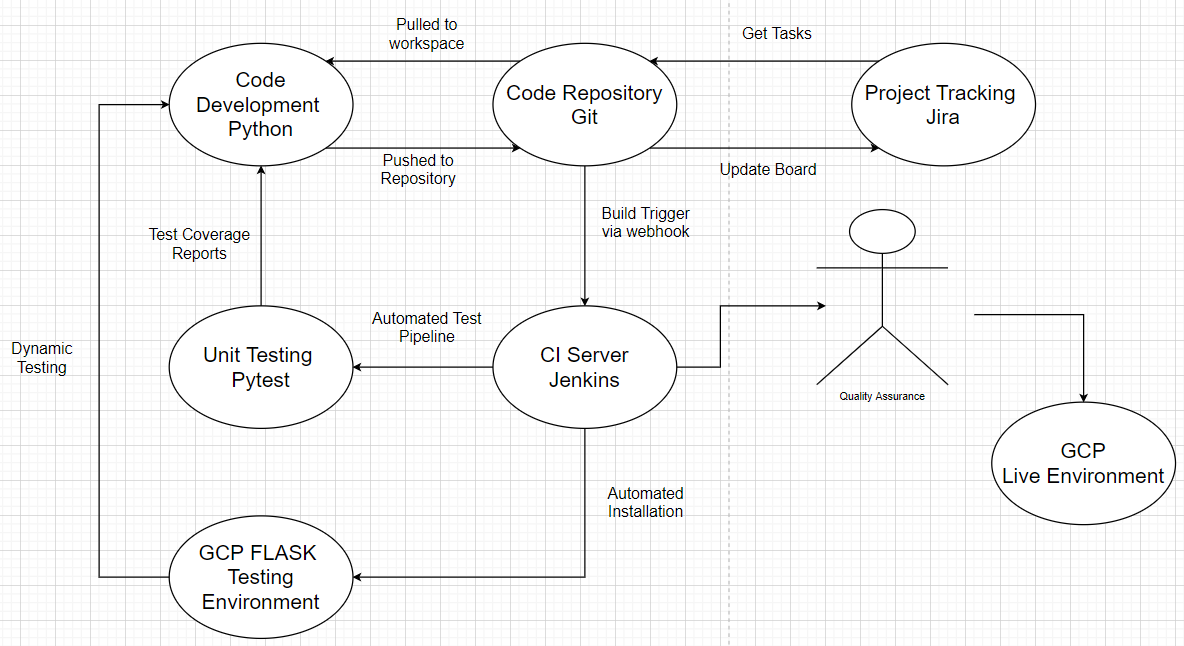
### Database Structure

The screenshot below shows the entity relationship diagram showing the structure of the database. The database was implemented fully as the diagram suggested without any changes.

The app models a many-to-many relationship between tblBooking and tblTimeSlots entities using an association table. Therefore, tblBookingLine was implemented between both entities to enable the user to create one booking containing many time slots.

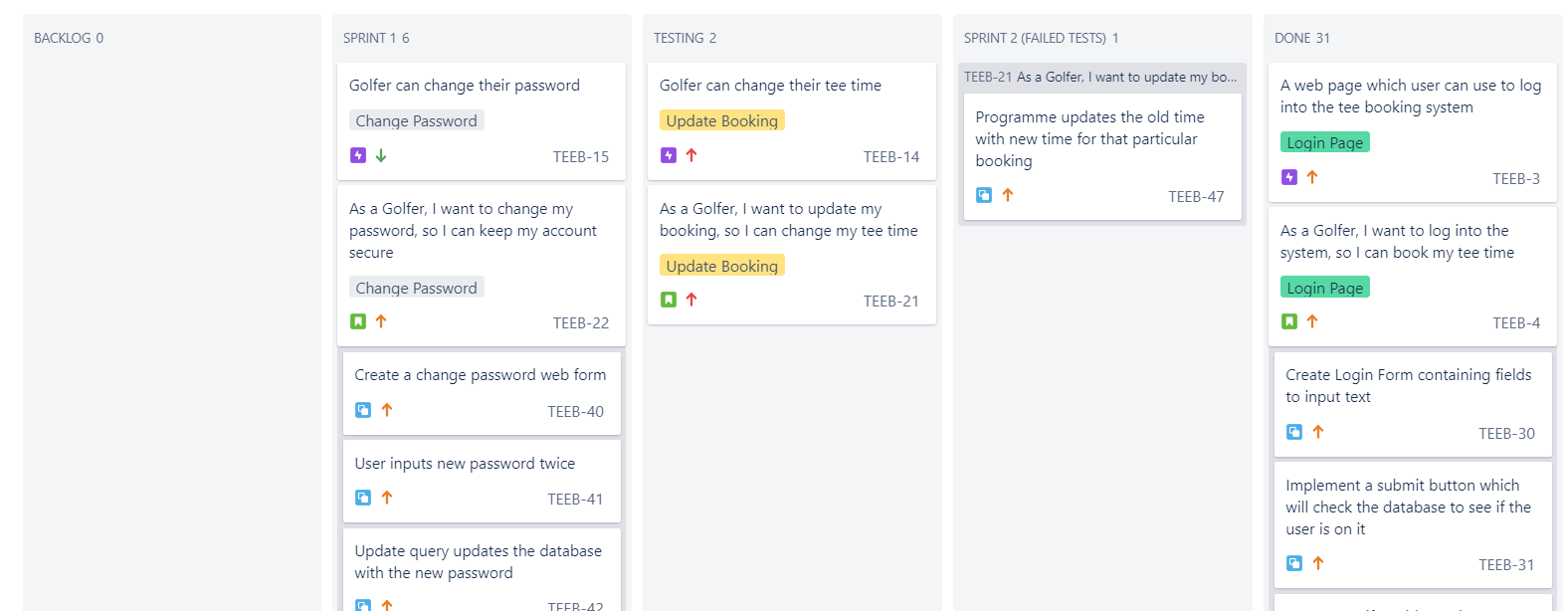
### CI Pipeline

Pictured below is the continuous integration pipeline with the associated frameworks and services related to them. This pipeline allows for rapid and simple development-to-deployment by automating the integration process, i.e. programmes cam produce code on their local machines and push it to GitHub, which will automatically push the new code to Jenkins via a webhook to be automatically installed on the cloud VM. From there, tests are automatically run, and reports are produced.

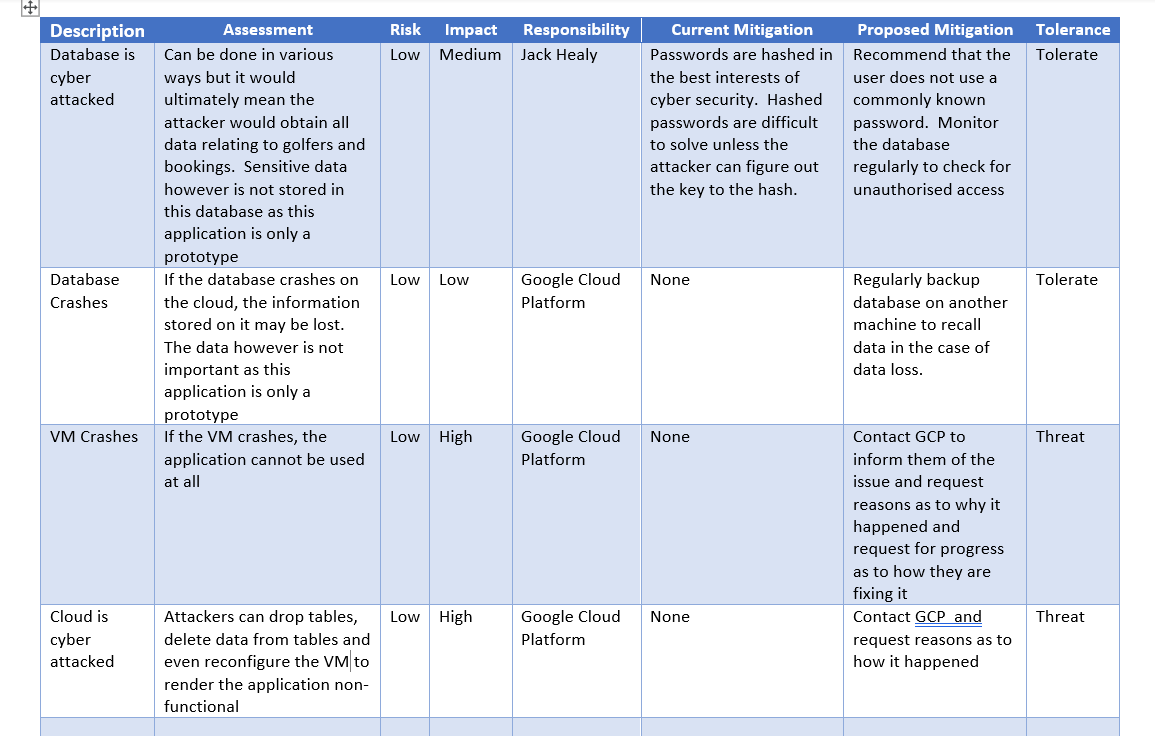


Project Tracking

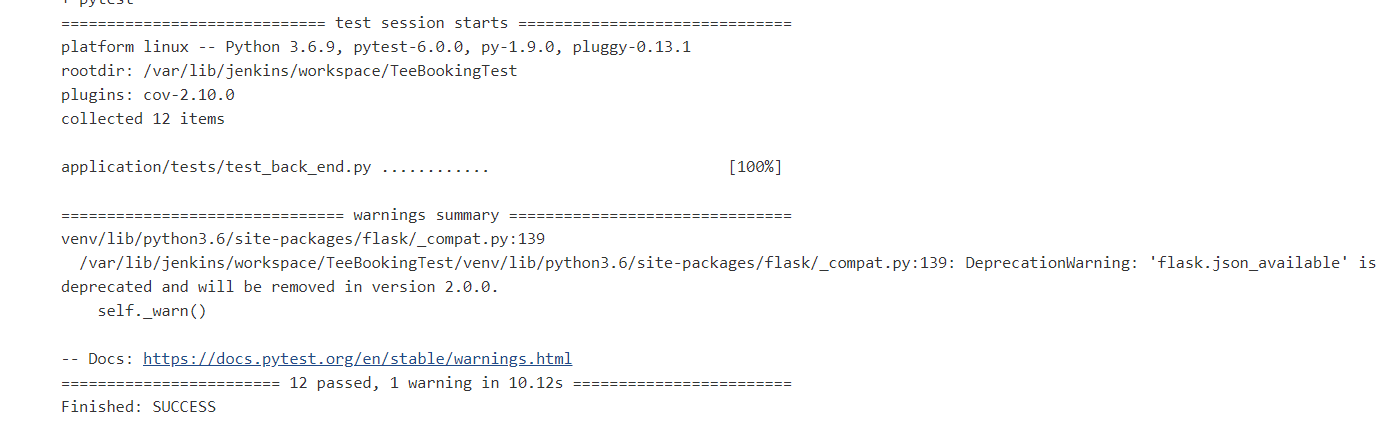
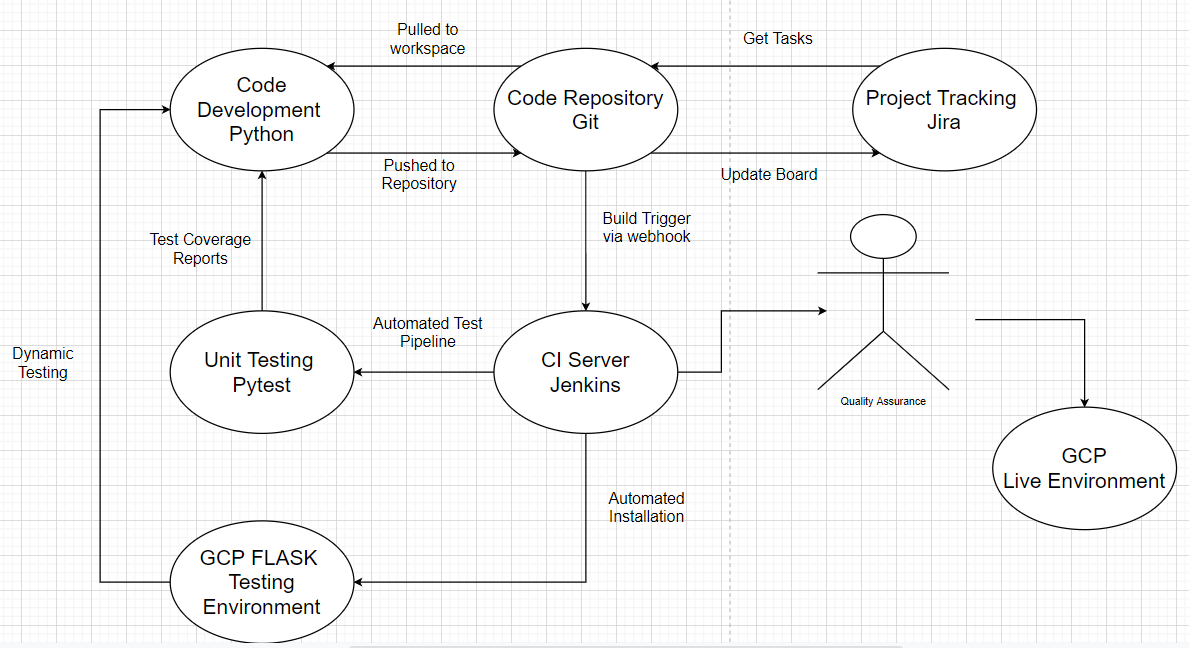
Hyperlink showing the full Kanban board is at the top of this file, but a screenshot is provided below. Most of the tasks have been completed except the change password which is still in progress and the updating tee time failed the test and so therefore needs fixed



Risk Assessment

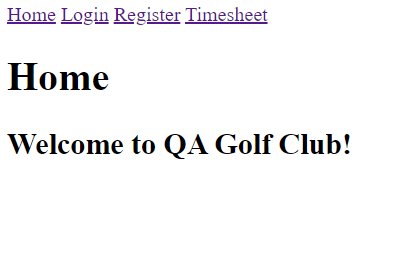
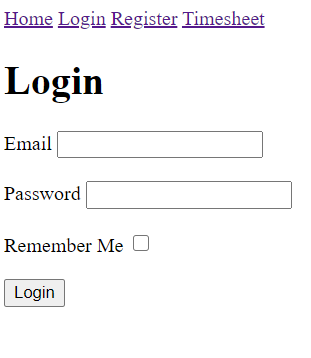


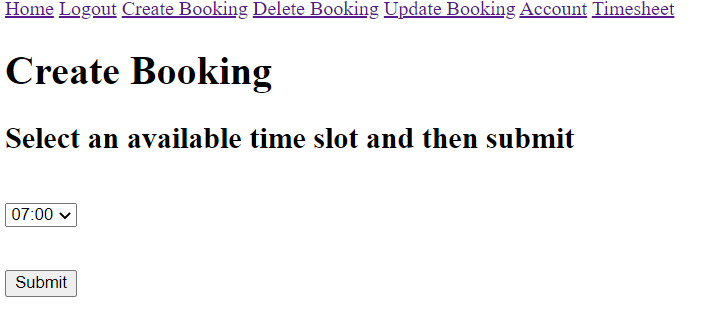
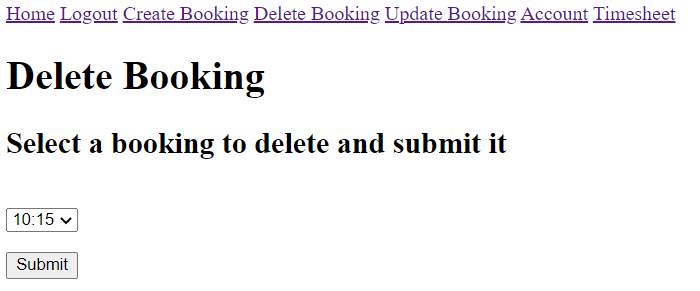
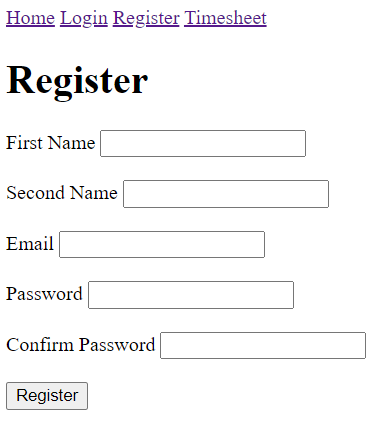
Testing

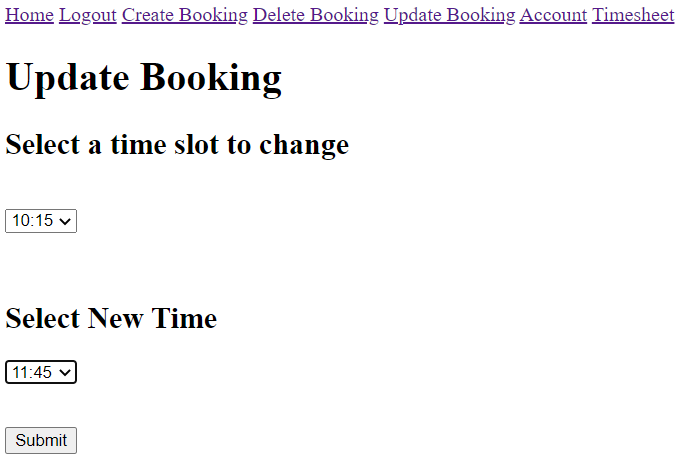
Pytest is used to run unit tests on the app. These are designed to assert that if a certain function is run, the output should be a known value. Jenkins produces console outputs (pictured below) that will inform the developer how many tests the code passed, and which tests they failed. In this instance, they all passed but the unit tests only test if the web forms run, there are no tests written to test the functionality of the CRUD requirements.

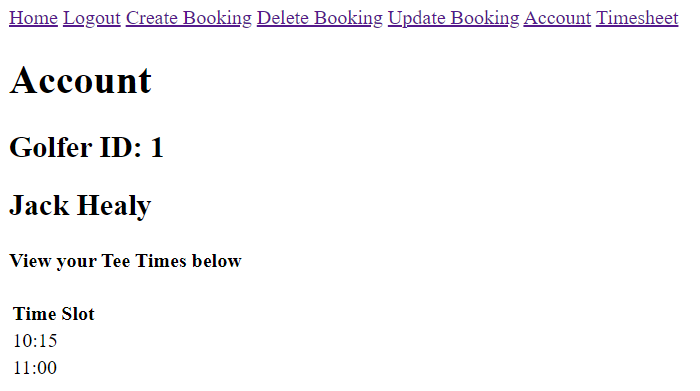
## Front-End Design

The front-end of the app is rudimentary at this stage, as the front-end is built purely with quite simple HTML and contains no CSS. It is functional and stable. The screenshots below show the various web forms:









## Known Issues

The app contains the following bugs:

* While creating a booking, you can only select one slot per booking rather than several slots. The database however is still implemented to allow the app to be implemented in a way to allow the user to book multiple slots per booking.
* Once bookings are made, the slot is not made unavailable which enables different users to book the same slot.
* Bookings cannot be updated onto the database. The user can however still delete slots and add new ones.
* The user can book the same slot twice.

Future Improvements

Improvements which I would like to make include

* The timesheet page to show the user who booked certain slots
* To add some CSS to improve the usability
* The application to allow the user to pick different dates along with their tee time rather than only have slots based on 1 day.

## Authors

Jack Healy