Tuckwood, Dan

Software Engineering & Agile Assignment

**ASSIGNMENT COVER SHEET**

|  |  |  |
| --- | --- | --- |
| **Student’s name** | Daniel | Tuckwood |
| **Module name** | Software Engineering & Agile | |
| **Title of assignment** | Web Development Assignment | |
| **Complete Word Count in my assignment** |  | |
| **Date submitted** |  | |

All work must be submitted by the due date. If an extension of time to submit work is required, a [Mitigating Circumstances Extension Form](https://canvas.qa.com/courses/1041) must be submitted.

<https://canvas.qa.com/courses/1041>

Has an extension been approved? Yes No If yes, please give the new submission date ….…/..…./…….

|  |
| --- |
| IMPORTANT: THIS STATEMENT MUST BE READ & SIGNED  **Academic Integrity Statement**  Academic integrity and honesty are fundamental to the academic work you produce at the University of Roehampton. You are expected to complete coursework that is your own and which is referenced appropriately. The university has in place measures to detect academic dishonesty in all its forms. If you are found to be cheating or attempting to gain an unfair advantage over other students in any way, this is considered academic misconduct and you will be penalised accordingly.   ​  **I declare that the work I am submitting is my work, is properly referenced and has not been submitted elsewhere.** |
| **Student Signature (Full Name): Daniel Tuckwood**  **Date:** |

Contents

[Figures 2](#_Toc113545507)

[Task 1 – Web Application Development 3](#_Toc113545508)

[Task 2 – Report on Web Application 3](#_Toc113545509)

[Artefact Structure 3](#_Toc113545510)

[File Breakdown 4](#_Toc113545511)

[Python Files 4](#_Toc113545512)

[HTML Files 4](#_Toc113545513)

[Other 4](#_Toc113545514)

[Entity Relationship Diagram 4](#_Toc113545515)

[Planning & Hosting 5](#_Toc113545516)

[Task 3 – Report on Agile Overview 6](#_Toc113545517)

[Agile Development Analysis 6](#_Toc113545518)

[Agile Development Improvement 6](#_Toc113545519)

[References 8](#_Toc113545520)

# Figures

|  |  |
| --- | --- |
| **Name** | **Page Number** |
| [Figure 2.1 – Artefact Structure](#_Artefact_Structure) | 3 |
| [Figure 2.2 – Entity Relationship Diagram](#_Entity_Relationship_Diagram) | 4 |

# Task 1 – Web Application Development

Code submitted via zip folder – Accessible via this URL: **https://webappdt.herokuapp.com/**

# Task 2 – Report on Web Application

The web application serves as a HUB for a support team. It allows them to create, edit and delete personal notes as well as information associated with ongoing incidents within the supported application. The primary use of the application will be to discuss ongoing work during stand-up meetings between the support team.

It is written in Python and HTML and uses Flask as a framework to facilitate the requirements set in the brief, such as CRUD, as a basic Model View Controller web app.

## Artefact Structure

The project consists of a **main.py** file as well as a **/website** directory, that contains more files and directories.

Within the **/website** folder is the application database, **QAappDB**, four python files: **\_\_init\_\_.py, auth.py, models.py** and **views.py**. In addition to this, there are **\_\_pycache\_\_**, **static** and **templates** directories. The **static** directory contains all the images used within the application, and the **templates** folder contains all of the front-end html pages: **admin.html**, **base.html**,**home.html**, **login.html**, **note.html**, **sign\_up.html**, **update.html,** **userGuide.html, updateCase.html and updateNote.html**. (See figure below).

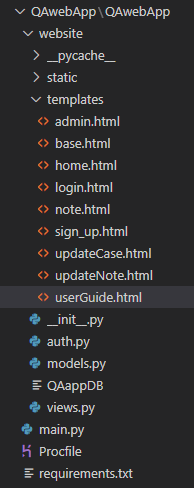


Figure 2.1 – Artefact Structure

## File Breakdown

### Python Files

**main.py**: Contains the location of where the application is created and run from.

**\_\_init\_\_.py**: Contains the creation and instantiation of the application db, as well as the creation of *login\_manager* features.

**auth.py**: Contains all the functions around the authentication of users within the application, such as creating an account, and login validation. It features functionality such as getting the data from a user form, performing validation on the data, and storing that within the database.

**models.py**: Contains the database model, consisting of three tables, **Case, Note & User**, as well as multiple data types, keys, and relationships.

**views.py**: Contains all the functions around the different views within the application, such as creating, updating, and deleting user notes. It features basic CRUD functionality.

### HTML Files

**admin.html**: Contains the front-end webpage that can only be accessed by system administrators.

**base.html**: Contains the base front end template, that will feature on all webpages within the application. It features html functionality such as a navbar and dropdowns.

**home.html**: Contains the front-end webpage for the ‘Cases’ screen. It contains a user form that stores data within the application database, as well as a list group that displays all cases stored within the database. Each case includes functional buttons to edit or delete the case.

**login.html**: Contains the front-end webpage for the Login page. It contains a user form that inputs the data entered into a function to check the login credentials.

**note.html**: Contains the front-end webpage for the ‘Notes screen. Similarly, to **home.html**, it contains a user form that stores data within the application database, as well as a list group that displays all notes stored by the logged in user within the database. Again, each note includes functional buttons to edit or delete the note.

**sign\_up.html**: Contains the front-end webpage for the Sign-Up page. It contains a user form that stores the data entered into the database and registers it as a new system user.

**userGuide.html**: Contains the front-end webpage for the application’s user guide.

**updateCase.html**: Contains the front-end webpage for updating a case.

**updateNote.html**: Contains the front-end webpage for updating a note.

### Other

**Procfile:** Enables compatibility with Heruko.

**Requirements.txt:** List of application stack requirements, enables compatibility with Heruko.

## Entity Relationship Diagram

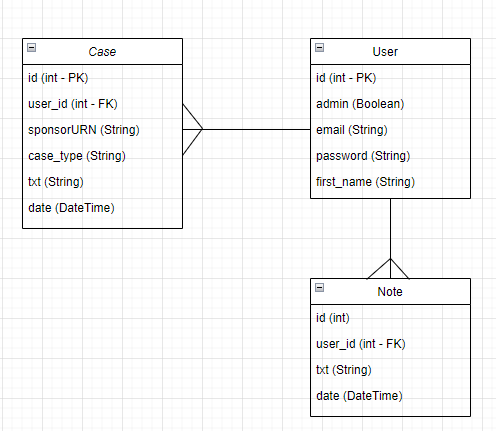


Figure 2.2 – Entity Relationship Diagram

The application database contains three tables, **Case, Note** & **User** which are linked together using **user\_id** as a foreign key.

## Planning & Hosting

The application be developed using Agile, specifically using the Kanban methodology, and will be hosted on Heruko as a cloud app and deployed in a DevOps manner.

# Task 3 – Report on Agile Overview

## Agile Development Analysis

The aspect of Agile of software development approach that is of interest to me is: The approach to running IT/Software development teams.

Firstly, it is important to assess what Agile actually is, and how it can be used within IT teams to increase productivity within the workplace: “The Agile approach to software development is defined by its commitment to creating software incrementally -- or in regular increases and stages. The approach offers users new versions, or releases, of software following brief periods of work. Those brief periods of work are often called [sprints](https://www.techtarget.com/searchsoftwarequality/definition/Scrum-sprint).” **(TechTarget, 2022)**

Within my workplace there is a cross-functional team that consists of primarily developers and testers, who constantly liaise with each other to ensure the constant incremental developments of a piece of software. This teamwork under the management of a project manager who closely liaises with stakeholders and product owners of the application in question. The methodologies used within the workplace are: Sprints, Kanban, and Stand-Up Meetings, which are undertaken by a variety of different roles within the team.

As well as continuous development and maintenance, part of the role of working in the agile team within my workplace is also to provide 3rd line user support for the application once it has gone live, which is where the use for the web application created is introduced.

Every fortnight, a sprint plan meeting takes place, which hosts discussions about goals and targets to be achieved within the next sprint period. Each task identified is given an estimated duration of hours that it will take to complete, also known as story points. This is done to manage expectations of stakeholders as well as aligning a clear path of progression. Any other items that will not fit into the timebox of the next sprint are then assigned to the sprint backlog, this is influenced by the prioritising of the stakeholders/ product owners. Once the sprint has begun, daily stand-up meetings take place in order to identify the following from each team member:

* What did you achieve yesterday?
* What is your goal for today?
* Do you have any blockers to prevent your progress?

A Kanban board is used to track the progress of each item within the sprint that features swim lanes:

* To-Do
* In-Progress
* Blocked
* Resolved

A Kanban board is there to provide a visualisation to the team and stakeholders on progress made during the sprint, it also adds the ability for team members to update tickets from one central location that all the team can access, providing clarity. At the end of each sprint, a sprint review meeting is held, which servers the purpose of showing the progress made by the team in the last two weeks, as well as hosting an open discussion between the Stakeholders and the project team about how things are going during work.

## Agile Development Improvement

A suggestion to improve the agile process within my organisation is to improve the product backlog, this includes the creation and the updating of the tickets themselves. The more detail that features in each story points means the time developers take to gain an understanding of the task at hand will decrease. This can also be accompanied by more regularly reordering and reprioritising the backlog in order to ensure higher impact tickets are completed sooner. This could be due to the product owner not spending enough time performing this task.

Another improvement suggestion could be to implement the use of a burndown chart into the daily stand-up, meetings. This acts as another visual representation of progress towards the sprint goals within the team and can act as praise or motivation to encourage the team to keep going.

Another suggestion could be to introduce the use of management software such as JIRA, as well as continuous integration tools such as Jenkins. The use of software to aid with tasks such as management and deployment will allow for the use of DevOps methodologies to be implemented within the team. This would drastically improve efficiency and productivity within the team, particularly when it comes to deploying new releases.

The idea of improving the agile process within the organisation has many benefits such as, increased learning for staff, resource saving and improving reputation. But ultimately the primary purpose is to increase satisfaction with your customers, as this may lead to improved contracts and future business.

To implement the suggested improvements, additional training sessions could be provided to staff to further improve their knowledge around agile. In order to monitor progress improvements, a designated person could be brought in to oversee this process, essentially another project manager. This person would be responsible for developing a roadmap of progress and facilitating individual needs.

# References

TechTarget. 2022. *Agile Manifesto.*[Online] Available at: <https://www.techtarget.com/searchcio/definition/Agile-Manifesto> [Accessed 18 August 2022]