s

**Names:** Sian Lennon 16343896 Shannon Mulgrew 16304263

**Project:** StudySmart

**Type:** Functional Spec

**Supervisor:** Andy Way

**Date:** 19/11/19

# Table of contents

# 1. Introduction

## 1.1 Overview

---

Our project “StudySmart” will be a web-app to assist students in the School of Computing when studying for exams. StudySmart will accept either study material (eg. text files) or a web url. The system will then process the material given and extract questions based on the text using NLP.

The idea of this web-app is inspired by the fact students study differently, meaning that the traditional methods such as note-taking etc do not work for everyone. This web-app is supposed to help students revise by quizzing them on material they have provided.

StudySmart will have many functions. The user will have the option to create an account and keep track of past quizzes and answers including a ‘score’ for how well they performed. Alternatively, they can just go forward without being logged in and not save any progress. All users can input text directly, upload a file (text,pdf,etc) or link a web page with the information they want to be quizzed on.

## 1.2 Business Context

---

Our product would be used mainly in an education environment. It could be deployed within the School of Computing. We are beginning our development based on modules within CASE and we may broaden this to adapt to all courses within the School of Computing. This could assist students within the school of computing in preparation for exams.

## 1.3 Glossary

---

- ***NLP*** *- Natural Language Processing*
- ***NLTK*** *- Natural Language Toolkit for Python*
- ***AWS*** *- Amazon Web Service used to host the Web-App*
- ***Docker*** *- software used to containerised code for good practice and architecture.*
- ***ML*** *- Machine learning*

# 2. General Description

## 2.1 Product / System Functions

Our product is StudySmart. It will be a dynamic web-application made to assist students within the school of computing in preparation for exams. Our web-app will take data (eg. Study Material) from the user in order to generate questions and quiz the user. The material can be submitted in three different forms:

**1)** Upload text directly
**2)** Upload a text file
**3)** Link a web url

Each of these forms will require NLP to process. Each form of information extraction will involve reading through the text, separating it into tokens and rearranging the tokens into a question. Once the questions have been generated and answered, the user will be presented with a score of how well they did in the quiz. If a user has created an account they will be able to save quizzes to their profile in order to view or redo them later.

Our web-app will offer an option to login/sign up to create an account or continue as a guest. If a user has an account they will be able to save quizzes and view/complete past quizzes also.

StudySmart will make use of Docker and AWS. Docker will be used to create containers and keep all our code/dependencies/libraries together to be shipped out together. AWS will be used to host our web app and also make use of the security/cloud computing features it offers.

## 2.2 User Characteristics and Objectives

This product will be designed to be as easy to use as possible. Basic knowledge of internet browsing is expected as they will have to navigate to/ around the web-app. The user will have to be able to upload a file/copy and paste text/link html page. This product will mainly be used by students who are familiar with technology.

### 2.3 Operational Scenarios

---

| **Use Case 1:** | User objective: Create account |
|---|---|
| **Goal in context:** | Create an account for a user |
| **Scope & Level:** | User Interface |
| **Success:** | Creates profile, move to login page. |
| **Failure:** | Profile not created. |
| **Actor(s):** | UI, User. |
| **Trigger:** | User chooses “Create new Profile” |
| | |
| **Step** | **Action** |
| 1. | User selects -> sign-up |
| 2. | User inputs username and password for new profile. |
| 3. | User clicks -> Create profile. |
| 4. | Profile is created with user specifications. |
| 5. | User is directed to sign-in page. |

| **Use Case 2:** | User objective: Upload file. |
|---|---|
| **Goal in context:** | Upload file to get quizzed on.. |
| **Scope & Level:** | User Interface. |
| **Success:** | File is uploaded. |
| **Failure:** | File does not upload. |
| **Actor(s):** | UI, User. |
| **Trigger:** | User chooses -> “Upload file for testing”. |

| | |
|---|---|
| **Step** | **Action** |
| 1. | User selects -> “Upload file for testing”. |
| 2. | Interface shows file uploader screen. |
| 3. | User selects file for upload. |
| 4. | “File uploaded” is shown on screen. |
| 5. | ‘Start Quiz’ Button appears on interface. |

| | |
|---|---|
| **Use Case 3:** | User objective: Give application a URL |
| **Goal in context:** | Give URL to get quizzed on |
| **Scope & Level:** | User Interface |
| **Success:** | URL is linked. |
| **Failure:** | URL is not linked. |
| **Actor(s):** | UI, User. |
| **Trigger:** | User inputs url to field box, clicks ‘Upload URL’ |
| | |
| **Step** | **Action** |
| 1. | User selects url and inputs into text box. |
| 2. | User selects -> “Upload URL’ |
| 3. | UI displays heading of website. |
| 4. | UI displays ‘Quiz me!’ button. |

| | |
|---|---|
| **Use Case 4:** | User objective: Do quiz based on their upload. |
| **Goal in context:** | Get quizzed on their uploaded document/URL |
| **Scope & Level:** | User Interface |
| **Success:** | Get asked answers on text, obtain score. |
| **Failure:** | Not get asked questions, no score given. |
| **Actor(s):** | UI, User. |
| **Trigger:** | User clicks -> “Quiz me!” |
| | |
| **Step** | **Action** |
| 1. | User selects -> “Quiz me!” |
| 2. | UI shows question and multiple choice answers. |
| 3. | User selects and answer. |
| 4. | User clicks -> ‘Go’ |
| 5. | Steps 3 and 4 are repeated until no more questions are available. |
| 6. | UI displays ‘score’ based on number of answers user got correct. |
| 7. | UI displays “Try again” and “home-page” buttons. |

| | |
|---|---|
| **Use Case 5:** | User objective: Save Quiz As Logged In User. |
| **Goal in context:** | Save Quiz to Users Account. |
| **Scope & Level:** | User Interface/ Database. |
| **Success:** | Saves Quiz to ‘saved’. |
| **Failure:** | Quiz does not save. |
| **Actor(s):** | UI, User, Database. |
| **Trigger:** | User chooses ‘Save Quiz”. |
| | |
| **Step** | **Action** |
| 1. | User selects -> ‘Save Quiz’. |
| 2. | Application saves quiz to db. |

| | |
|---|---|
| **Use Case 6:** | User objective: View saved quiz. |
| **Goal in context:** | User views a saved quiz. |
| **Scope & Level:** | User Interface/ Database. |
| **Success:** | User sees saved quiz.. |
| **Failure:** | Quiz does not see saved quiz |
| **Actor(s):** | UI, User, Database |
| **Trigger:** | User chooses ‘See quiz’ |
| | |
| **Step** | **Action** |
| 1. | User selects -> ‘Saved quizzes’ |
| 2. | User clicks a quiz to view. |

### 2.4 Constraints

---

**Time:** Balancing this project and college work will be difficult but trying to stick to a regular routine and small deadlines should improve productivity. We want to use the Gannt chart as a plan to keep and to also communicate regularly with our supervisor in regards to deadlines and any issues we come across.

**Testing/Training**: Providing the system with enough testing material to be accurate. We plan to get this material from our course notes for CASE4 and websites that refer to these topics, from there we may need to outsource for more lecture material.
Unit testing and user testing will be carried out throughout the project but we may have to apply for ethical approval form in regards to getting other students to test our project.

**Communication:** As this year is the first to complete CA400 in pairs, we will have to communicate very well as we are dependent on each other for this project.

**Techniques**: There are many different techniques for NLP. Our initial plan is to implement regular expressions with NLTK and basic machine learning but this may prove to be the incorrect direction.

## 3. Functional Requirements

---

## Requirement 3.1: Retrieving website text.

- **Description -** Retrieving website text from inputted url. This is necessary to allow the application to apply the NLP on the text.
- **Criticality -** Critical (system as a whole does depend on requirement).
- **Technical issues -** API calls and sifting through HTML/XML tags could prove tricky.
- **Dependencies with other requirements -** N/A.

## Requirement 3.2: Retrieving file/input text.

- **Description -** Retrieving file/input text from upload file/text. This is necessary to allow the application to apply the NLP on the text.
- **Criticality -** Critical (system as a whole does depend on requirement).

- **Technical issues -** Different file types/permissions could be difficult to manage.
- **Dependencies with other requirements -** N/A.

## Requirement 3.3: Obtaining questions from text.

- **Description -** Obtaining questions from text is the main feature of our app.
- **Criticality -** Very critical (very important for system).
- **Technical issues -** Ensuring that appropriate questions and questions are generated through NLP and ML.
- **Dependencies with other requirements -** Retrieving website text or Retrieving file/input text is necessary to apply the algorithms the yext.

## Requirement 3.4: MySQL Database.

- **Description -** MySQL storage for users/saved answers.
- **Criticality -** Critical (System will not operate efficiently without proper storage).
- **Technical issues -** Linking database with other requirements. SQL queries.
- **Dependencies with other requirements -** N/A.

## Requirement 3.5: User interface.

- **Description -** User interface.
- **Criticality -** Critical (system as a whole does not depend on requirement however will need for display).
- **Technical issues -** Need to ensure ease of use of application.
- **Dependencies with other requirements -** N/A.

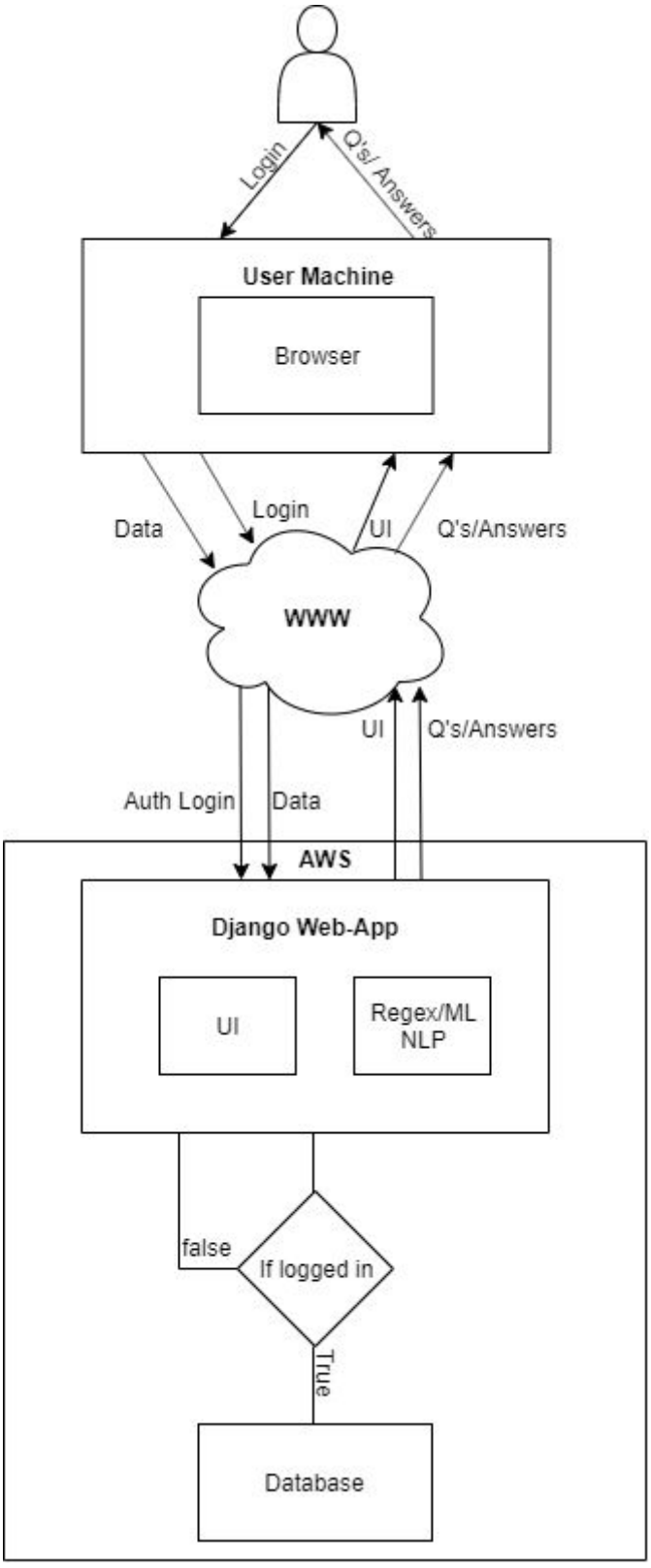
## Requirement 3.6: AWS and Docker.

- **Description -** AWS for hosting & Docker containers for ease of use and good architecture.
- **Criticality -** Critical (system as a whole does not depend on requirement however will need for hosting).
- **Technical issues -** Linking the django and the two together. Containerizing all sections.

- **Dependencies with other requirements -** N/A

## 4. System Architecture

---



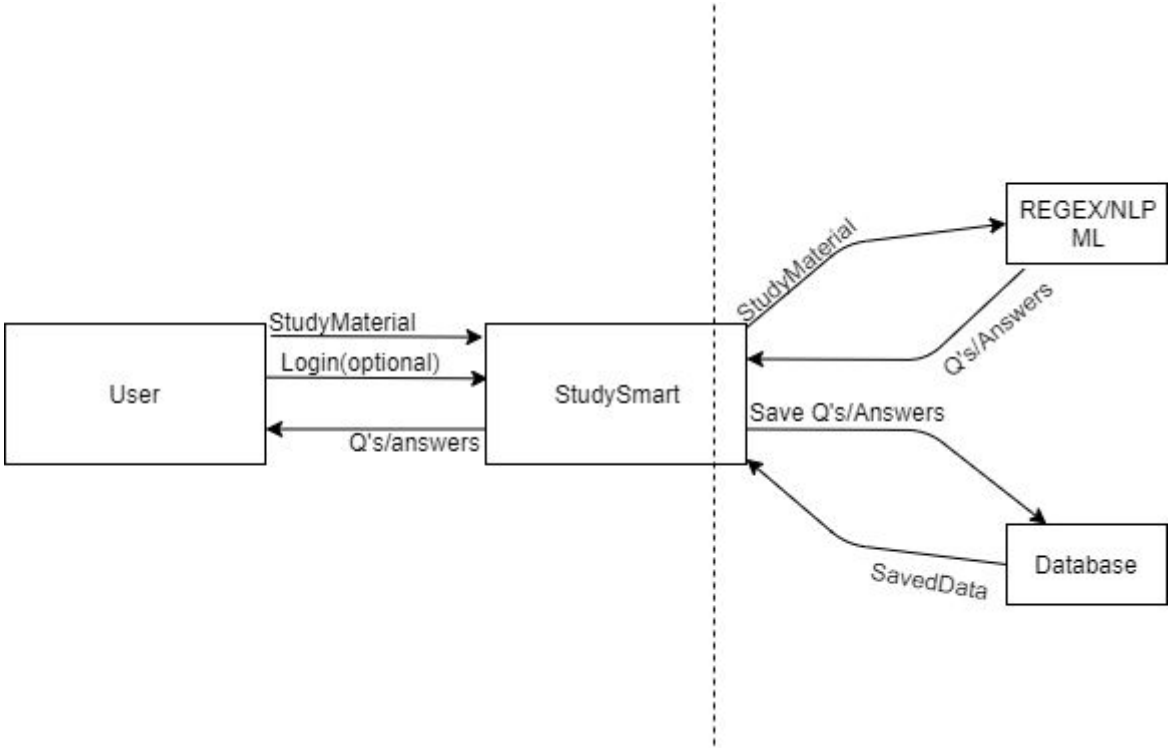

The system architecture diagram above shows the interaction between the user and our application. The user gets access to the internet via their machine and interacts with our web-app which is hosted on the AWS. The web-app provides a frontend UI and the backend takes care

of the NLP. If a user is logged in the web-app will then interact with the database.

## r5t. High Level Design

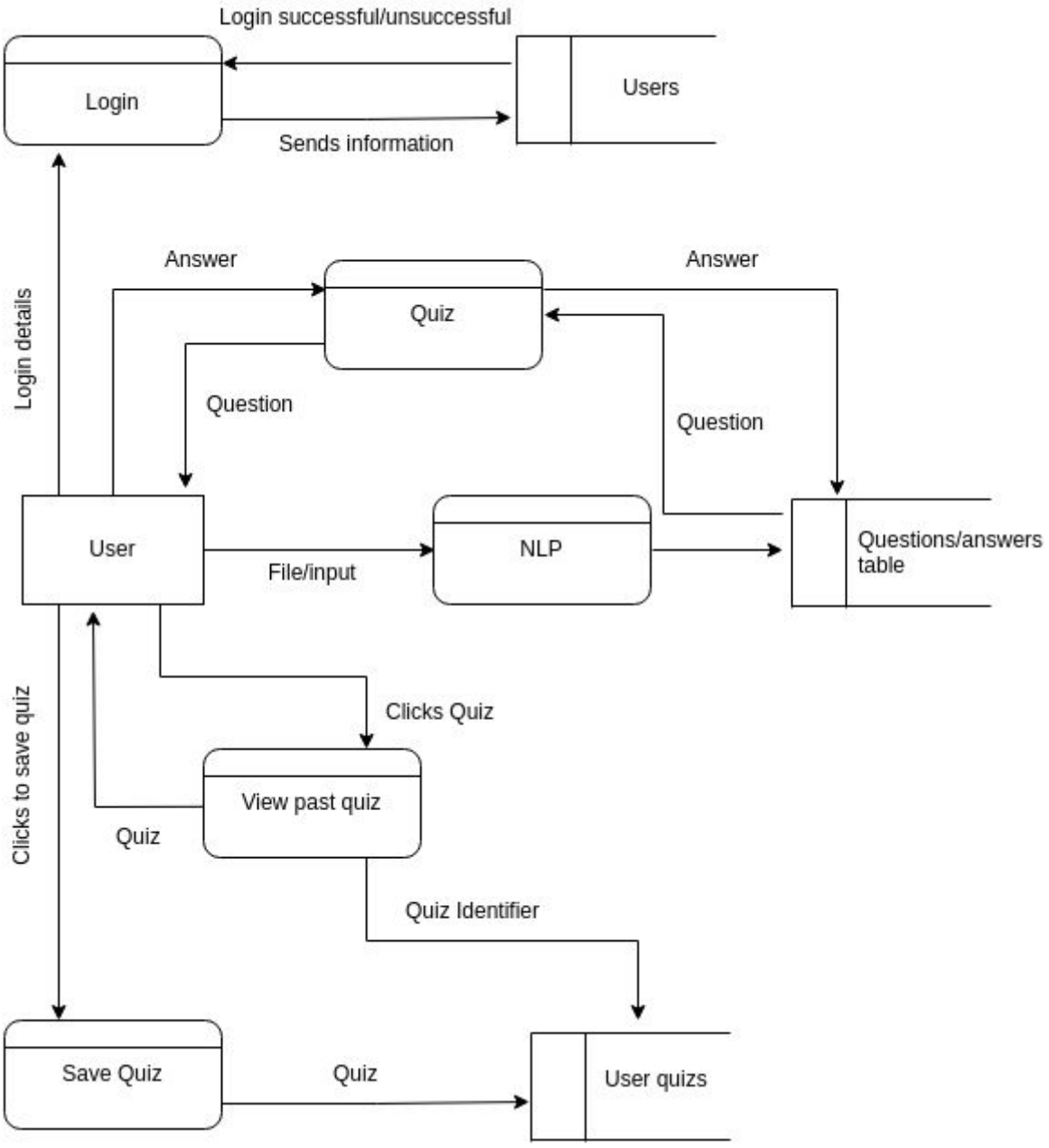
---

**CONTEXT DIAGRAM**




The context displays the boundaries within and interacting with our system.
As seen above the user will interact with the Web-app and its UI. The system will display the front end and interact with a database and a NLP backend

**DFD**




This DataFlow Diagram describes the flow of data through the application.

The user here is the external entity interacting with multiple processes of the application. From there, each process will interact directly with the datastores to store modify or extract data for the user.
E.g User sends a file or input into the NLP process. The Process works NLP on the text and stores the questions and answers in the Questions/answers data store. The questions will then be sent to the Quiz process for quizzing the user.

## 6. Preliminary Schedule

The GANTT chart below shows the Preliminary schedule we have drawn up for the development of the CA400 project.

We broke the project down into the main tasks we must undertake separated by person assigned to that task.The first semester we plan to undertake the bulk research and initial setup of the system and NLP. The second semester will focus on the training, testing and linking of the application. Finally the documents and preparation for the final presentation will be done.

| | Task | Assigned To | Start | End | Dur | % | 2019 | | 2020 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Q3 | Q4 | Q1 | Q2 | Q3 |
| | CA400 Project ⊖ | | 9/23/19 | 5/29/20 | 172 | | | | | | |
| 1 | Project idea & Proposal | Sian & Shannon | 9/23/19 | 10/31/19 | 28 | | | | | | |
| 2 | Functional Specification | Sian & Shannon | 10/31/19 | 11/22/19 | 16 | | | | | | |
| 3 | NLP | Sian | 11/22/19 | 1/22/20 | 40 | | | | | | |
| 4 | Docker & AWS research | Shannon | 11/22/19 | 12/6/19 | 10 | | | | | | |
| 5 | Django setup | Shannon | 12/6/19 | 1/22/20 | 31 | | | | | | |
| 6 | NLP Training | Sian | 1/23/20 | 2/23/20 | 21 | | | | | | |
| 7 | User Interface | Shannon | 1/23/20 | 2/23/20 | 21 | | | | | | |
| 8 | Linking systems | Sian & Shannon | 2/24/20 | 4/3/20 | 30 | | | | | | |
| 9 | Testing | Sian & Shannon | 4/3/20 | 4/24/20 | 16 | | | | | | |
| 10 | Docs | Sian & Shannon | 4/25/20 | 5/10/20 | 10 | | | | | | |
| 11 | Submission | Sian & Shannon | 5/11/20 | 5/11/20 | 1 | | | | | | |
| 12 | Demonstration | Sian & Shannon | 5/19/20 | 5/29/20 | 8 | | | | | | |
| 13 | Semester 1 exams | Sian & Shannon | 1/6/20 | 1/22/20 | 12 | | | | | | |
| 14 | Semester 2 exams | Sian & Shannon | 5/5/20 | 5/18/20 | 10 | | | | | | |

## 7. Appendices

Django Framework - https://www.djangoproject.com/
AWS - https://aws.amazon.com/
NLP - https://en.wikipedia.org/wiki/Natural_language_processing
NLTK - https://www.nltk.org/