**Singapore Institute of Technology**

**Infocomm Technology Cluster**

**ICT4001 Capstone Project**

Capstone Project Final Report Submission Form (Form E)

# Preamble

This form duly filled and signed must be attached to each Capstone Project Final Report submitted.

**Reporting Period:**

(Reporting Period): 9/1/2017 to 15/12/2017

**Candidate Particulars:**

(Name of Candidate): Chow Jie Jin Edmund

(Organization): Mandai Park Development Pte Ltd

(Project Period): 9/1/2017 to 15/12/2017

**Main Academic Supervisor Details:**

(Name): Dr Tan Chek Tien

(Email Address): ChekTien.Tan@Singaporetech.edu.sg

(Contact Number): 9227 8780

(Designation): Assistant Professor

**Industry Supervisor Details:**

(Name): Danny Tan Wei Yang

(Email Address): danny.tan@mandai.com

(Contact Number): 9369 8836

(Designation): Assistant Vice President

(Department / Division): Digital

# Declaration of Conformity

### Declaration by Industry Supervisor

I hereby acknowledge that I have read and understood the contents of the Capstone Project Final Report, and deem the contents appropriate for release to the Singapore Institute of Technology for use in the assessment of the candidate.

|  |  |
| --- | --- |
| *Signature* |  |
| **Name of Industry Supervisor:** Danny Tan Wei Yang  **Designation:** Assistant Vice President  **Department:** Digital  **Contact No:** 9369 8836  **Email:** danny.tan@mandai.com  **Date:** 23/11/2017 | |

### Declaration by Candidate

I hereby acknowledge that I have engaged and discussed with my Industry Supervisor on the contents of the Capstone Project Final Report, and have sought approval for the release of the report to the Singapore Institute of Technology.

|  |  |
| --- | --- |
| *Signature* |  |
| **Name of Candidate:** Chow Jie Jin Edmund  **Contact No:** 9151 3429  **Email:** 14sic014c@sit.singaporetech.edu.sg  **Date:** 21/11/2017 | |

**END OF FORM E**

|  |
| --- |
| **Information and Communication Technology Cluster** |
| Building the community through digital means  Capstone Project Final Report  (Mandai Park Development Pte Ltd)  For the reporting period from 9/1/2017 to 15/12/2017  Chow Jie Jin Edmund  14SIC014C |
| Name of Industry Supervisor: Danny Tan Wei Yang  Name of Academic Supervisor: Dr Tan Chek Tien |
| Submitted as part of the requirement for ICT4001 Capstone Project |

**Captive WiFi Portal**

# Specialised Terminology

|  |  |  |
| --- | --- | --- |
| Abbreviated | Full Name | Description |
| Captive Portal | Captive WiFi Portal | A web page which is displayed to newly connected users before they are granted broader access to network resources.Commonly presented as a landing page or a log-in page which may require authentication. |
| Web Server | Web Server | Powerful computers called servers that store files and information on the form of website so that it can pass this information to whoever requests to view a webpage on a browser |
| Backend | Backend | Components of a webpage that deals with logic and data storage |
| Frontend | Frontend | Components of a webpage that deals with the layout and template of a webpage |
| Database | Database | A structured set of data held in a computer |
| MailChimp | MailChimp | MailChimp is a marketing automation platform and an email marketing service |
| CRM | Customer Relationship Management | New system as an upgrade to the current CRM 2011 to act as centralized collection source of customer data |
| SSID | Service Set Identifier | Primary names of the network you see that is available to connect when connecting to WiFi |
| Justinmind | Justinmind Prototyping Tool | An authoring tool for web and mobile prototypes and high-fidelity wireframes |

# Introduction

This section provides an overview of what the project, the way it works from a broad picture and how relevant in the industry in today’s world.

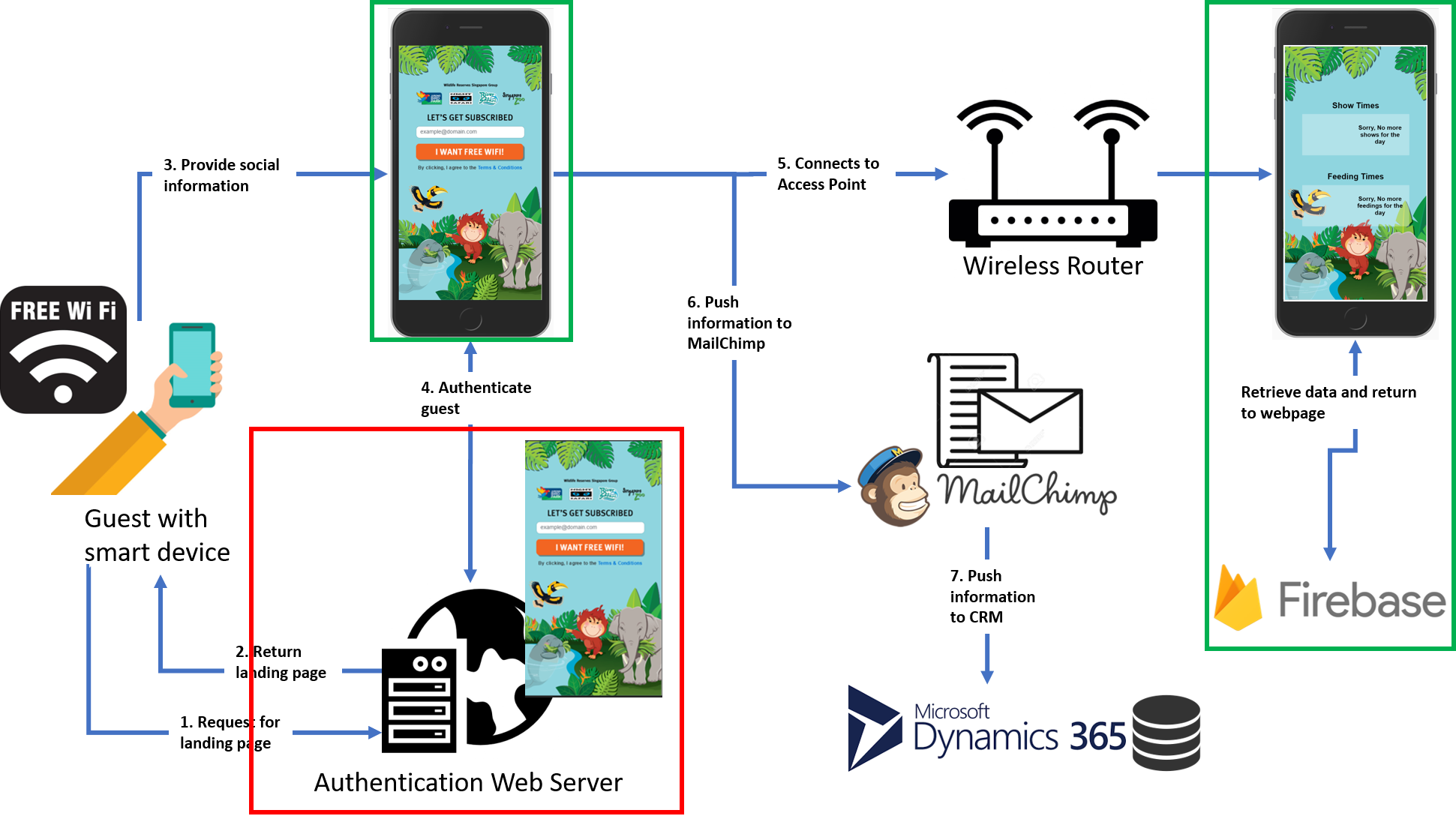
# What is the project about?

The parks in Mandai do provide free wifi to customers to ensure they stay connected and contactable. However, some visitors may not be aware of the existence of free park wifi that they can utilise when visiting the parks in Mandai. Visitors who are aware of the existence of the free wifi in the parks may not be aware of the terms and conditions that comes with it when utilising the free wifi provided by the parks. As the terms and conditions are not clearly stated anywhere to the customers, this can pose a potential problem if a customer were to utilise the free wifi provided by the parks to conduct illegal operations.

To curb this potential problem, the solution would be to implement a Captive WiFi Portal otherwise known as Captive Portal. The solution is fitting for the problem as the wireless network is open to the public and by informing customers the conditions of access, users will have to take responsibility for their actions and avoid any major problems for the organization. Waterfall model is used for this project as it’s software lifecycle model. Waterfall model is chosen because of the simplicity of the project, it being a webpage with some backend logic. The project will be divided into 2 phases, Phase 1 & 2.

# Complexity

To implement a Captive Portal would require a **web server** to store the webpage files, a **database** to store data that needs to be populated on the webpage, **front-end** and **back-end** codes. For this project only the database, front-end and back-end codes would be needed as the web server portion which includes authentication is handled by an external vendor. Additional features will be included into the Captive Portal to enhance the park experience of visitors. The additional features will be mentioned later in this section.



#### Figure 1. Captive Portal Architecture Diagram

To give an overview of how the captive portal works. Whenever a visitor connects to the free wifi provided by the parks using a smart device such as a smartphone, it will automatically launch the device’s default browser to display the landing page as shown in Figure 1 after step 3. Before the browser can display the webpage, the device will first request the webpage from the web server*(Figure 1 step 1)* in which the web server will return the necessary webpage information in the form of a website to the browser *(Figure 1 step 2)*. After displaying the landing page, the user may or may not have to provide user information in exchange for internet access follow by clicking a button on the landing page. Once the button has been clicked, the user will be authenticated against the authentication web server and if authenticated will be granted internet access.

Once the user has been granted access, if there was information provided by the user, the information will be saved into MailChimp and CRM. The user will also be redirected to another webpage where it contains information on the next earliest animal show time and animal feeding time.

As mentioned earlier, the web server which includes authentication boxed in red as shown in [Figure 1](#_Figure_1._Captive) will be handled by an external vendor. My work for this project involves the portion boxed in green, mainly the front-end, back-end and the database areas. The main purpose of the captive portal is to captivate the user, ensures that user agrees to the conditions of the access before releasing the internet usage to the user. Since the main highlights in the parks are the animal shows and animal feedings, I added the functionality of suggesting the next earliest possible time for both the animal show and animal feeding. The suggested time will be based on the next closest time from the current time. In this way, visitors who are connected to the free wifi will know when is the next available shows and feedings without the need to flip through the map brochure to search and plan for it.

# Industry Relevance

In today’s world, the importance of having a captive portal for open public wifi is extremely relevant to the industry other than the fact that many businesses are already using it for their public wifi access. The following paragraphs in this section will explain the relevance of the captive portal in today’s industry, with each paragraph emphasizing on one point.

# You need a page identifying it is your business and your wifi network

Because it is a public wifi, it is open and vulnerable to all kinds of online malicious threats. Not having a captive portal with a landing page can potentially result in a hacker posing as a legitimate wireless network and start stealing data of whoever is connected to the malicious network. Once a user starts associating with the fake network, the hacker can easily grab all the user’s personal information. Hence it is important to have a captive portal to protect the public wifi as the landing page serves as an identification of the business and the wireless network.

# Protect your business from being sued

Whenever there is network access, there is always an IP Address issued. IP addresses works just like physical address to identify the location of where something is happening. Imagine someone pirating music or movies using the free network provided by the park, the companies looking to sue can track the source location of where the event took place, essentially the business location. Captive Portal allows the terms and conditions to be included on the landing and this forces the guest to accept and adhere to the conditions of utilizing the internet access. This will help limit the business’s liability for any bad stuff if it were to happen.

# Using data to improve the user experience

Captive portals not only can provide you and your customers a safe and enjoyable wireless experience, they can also provide you with valuable data that you can then use to improve your services and products to ultimately create a better user experience. One of the easiest ways to do this is by allowing your guests to use their social login credentials on your captive portal to gain access to your network. In exchange for this option your guests will be authorizing you to have access to a variety of social data points.

For example, if someone using your captive portal logs in using their Facebook account, you could have permission to collect information such as, age, date of birth, what their interests are etc. With this information you can create more personalized experiences, based on real information. Next, we move on to talk about the details happening in the different phases of Waterfall model adopted for this project.

# Requirements Phase

This section provides an overview on the requirements phase, which explains in detail how requirements were gathered, how requirements were analysed.

# Requirements Gathering

To kick start this project, I had to gather requirements to formalize the necessary details essential for the subsequent phases. For this project, requirements were gathered from the marketing department and it mainly consists of requirements related to the layout of the landing page such as what font-family to be used, which set of company logos to be used and etc… All requirements gathered were documented into a document.

The requirements gathered from marketing department were as followed:

* Landing page of the captive portal is to be mobile friendly
  + Because park visitors accessing the free network will mostly be on the smart devices
* To use the background created by marketing department to be the background of the landing page
  + Background to be used 🡪 
* To use the 4 park logos in the landing page, logos are side by side each other
  + Logo 🡪 
* To use the submit button created by marketing department and in the button, the text inside the button has to be 🡪 “I WANT FREE WIFI”
  + 
* All fonts used in the landing page must be of sans-serif family
* The company logo must be at the top of the landing page
* 3 ways of providing user information in exchange for internet access
  + **No email or social sign-in in exchange for internet access** - user will have to click a button on the landing page and user will be granted access to the free wifi
  + **Provide email address in exchange for internet access** – user will have to provide his/her email address in the landing page, hit the submit button and will be granted access to the free wifi
  + A choice between providing email address or perform social login in exchange for internet access
    - User will have to provide his/her email address in the landing page, hit the submit button and will be granted access to the free wifi

OR

* + - User logins to his/her social media account, the common ones like Facebook or Google which grants the captive portal access to user’s profile information such as email address and will be granted access to the free wifi

The requirements I added for the additional features for the landing page:

* To display the next earliest possible animal show details when user is redirected to another page
  + Details of the animal show must have
    - Animal Show Image
    - Animal Show Time
    - Animal Show Park Name
    - Animal Show Title
    - Location of animal show in the park
* To display the next earliest possible animal feeding details when user is redirected to the page mentioned above
  + Details of the animal feeding must have
    - Animal Feeding Image
    - Animal Feeding Time
    - Animal Feeding Park Name
    - Animal Feeding Title
    - Location of animal feeding in the park
* Show timings, park name, location and title to be stored in Firebase database
* Images to be stored separately in project directory

Knowledge applied during requirements gathering:

* ICT 2101 – Introduction to Software Engineering
  + The understanding of the commonly used software lifecycle models in today’s world of which Waterfall model is chosen to be used for this project
  + The basic understanding of what user requirements are is required in this section
* ICT 2103 – Information Management
  + The basic understanding of how databases work and how the data should be stored is required in this section

# Requirements Analysis

After gathering the requirements, I had to analyse the requirements. This section will explain in more details about how the requirements were analysed from what was gathered. Analysis such as use cases and user stories will be used to analyse the requirements.

# User Stories

This section will list out the user stories to describe the needs of the users using the captive portal. It gives a high-level description of a scenario and enables readers to immediately understand what it means just by reading it. It uses the template of **as a, I want to, so that.**

The user stories are as follows:

* **As a** user, **I want to** be able to view the terms and conditions **so that** I can understand the conditions of access when using surfing the web on the free network
* **As a** user, **I want to** be able to view the animal show details **so that** I know which animal show closest to the current time I can watch
* **As a** user, **I want to** be able to view the animal show details **so that** I know which animal show closest to the current time I can watch
* **As a** user, **I want to** be able to view the animal feeding details **so that** I know which animal feeding closest to the current time I can watch
* **As a** user, **I want to** be able to provide zero personal information in exchange for internet access **so that** my personal information will not be disclosed to external parties and still be able to access and utilise the free network
* **As a** user, **I want to** be able to provide email address information in exchange for internet access **so that** I can receive personalised updates like newsletter or park promotions
* **As a** user, **I want to** be able to log-in to my social account like Facebook or Google in exchange for internet access **so that** I do not have to input any information and still can provide personal information by allowing the captive portal have access to my social account and receive personalised updates like newsletter or park promotions

Knowledge learned in school that was applied in this section:

* ICT 2101 – Introduction to Software Engineering
  + Skills and practice I had gone through in learning how to create a proper user story is being used here.

# Use Cases

This section will display the functional requirements of the project in a visual manner to have a better understanding of what actions can the user perform when using the captive portal. It describes the behaviour that will be built into the software to meet the required needs. It also describes the complete interaction between the software and the users (and possibly other systems). Each use case will be accompanied with a detailed description of what the use case is about.

# Use Case Diagram

# 

#### Figure 2. Captive Portal Use Case Diagram

As shown in Figure 2, it lists out all the functional aspects of the Captive Portal that the user can perform and there are 3 ways for a user to provide personal information in exchange for the free internet. As mentioned in the introduction paragraph, there will be 2 phases to this project. Phase 1 will exclude the functional requirements of user providing email address and user logging in to their social media account in exchange for internet access. Phase 2 will encompass the functional features in Phase 1 and the ones excluded from Phase 1.

The use cases are as follows:

* UC-1

|  |  |
| --- | --- |
| **Name** | **UC-1: View terms and conditions** |
| Summary | Parks visitors can view the terms and conditions on the landing page |
| Rationale | Viewing the terms and conditions allows the user to understand what are the dos and don’ts when using the free wifi provided by the parks |
| Users | All users |
| Preconditions | Smart device connected to the **PARK\_FREE\_WIFI** SSID |
| Basic Course of events | 1. User connects to the PARK\_FREE\_WIFI SSID 2. Web browser automatically launches and renders the landing page 3. User locates the terms and conditions link and click on it 4. User is redirected to the terms and conditions page |

* UC-2

|  |  |
| --- | --- |
| **Name** | **UC-2: Provide zero personal information in exchange for internet access** |
| Summary | Park visitors do not need to provide any personal information in exchange for internet access |
| Rationale | Park visitors may not want to reveal their personal information to others |
| Users | All users |
| Preconditions | Smart device connected to the **PARK\_FREE\_WIFI** SSID |
| Basic Course of events | 1. User connects to the PARK\_FREE\_WIFI SSID 2. Web browser automatically launches and renders the landing page 3. User inputs the email address into the text field 4. User clicks the button 5. User is redirected to a new page displaying the details of animal shows and animal feedings |

* UC-3

|  |  |
| --- | --- |
| **Name** | **UC-3: View animal show details** |
| Summary | Details about the animal show such as the title, location, time etc. is displayed to the user |
| Rationale | This is to enhance the visitor’s experience without the user going through the trouble to flip the map brochure to search for animal shows |
| Users | All users |
| Preconditions | 1. Smart device connected to the **PARK\_FREE\_WIFI** SSID 2. User has agreed to the terms and conditions |
| Basic Course of events | 1. User connects to the PARK\_FREE\_WIFI SSID 2. Web browser automatically launches and renders the landing page 3. User may or may not provide personal information 4. User clicks the button 5. User is redirected to a new page displaying the details of animal shows and animal feedings and can look through the details |

* UC-4

|  |  |
| --- | --- |
| **Name** | **UC-4: View animal feeding details** |
| Summary | Details about the animal feeding such as the title, location, time etc. is displayed to the user |
| Rationale | This is to enhance the visitor’s experience without the user going through the trouble to search for animal feedings on the park website |
| Users | All users |
| Preconditions | 1. Smart device connected to the **PARK\_FREE\_WIFI** SSID 2. User has agreed to the terms and conditions |
| Basic Course of events | 1. User connects to the PARK\_FREE\_WIFI SSID 2. Web browser automatically launches and renders the landing page 3. User may or may not provide personal information 4. User clicks the button 5. User is redirected to a new page displaying the details of animal shows and animal feedings and can look through the details |

* UC-5

|  |  |
| --- | --- |
| **Name** | **UC-5: Provide social media account in exchange for internet access** |
| Summary | Park visitors can login to their social account like Google or Facebook and allows the captive portal to access their personal information |
| Rationale | With the email address collected by the captive portal, it can enhance the customer experience by sending newsletters on promotions and etc… |
| Users | All users |
| Preconditions | Smart device connected to the **PARK\_FREE\_WIFI** SSID |
| Basic Course of events | 1. User connects to the PARK\_FREE\_WIFI SSID 2. Web browser automatically launches and renders the landing page 3. User either clicks Facebook or Google button 4. User is redirected to a new page displaying the details of animal shows and animal feedings and can look through the details |

* UC-6

|  |  |
| --- | --- |
| **Name** | **UC-6: Provide email address in exchange for internet access** |
| Summary | Park visitors provide their email address and click a button on the landing page in exchange for internet access |
| Rationale | With the email address collected by the captive portal, it can enhance the customer experience by sending newsletters on promotions and etc… |
| Users | All users |
| Preconditions | Smart device connected to the **PARK\_FREE\_WIFI** SSID |
| Basic Course of events | 1. User connects to the PARK\_FREE\_WIFI SSID 2. Web browser automatically launches and renders the landing page 3. User inputs the email address into the text field 4. User clicks the button 5. User is redirected to a new page displaying the details of animal shows and animal feedings |

Knowledge learned in school that was applied in this section:

* ICT 2101 – Introduction to Software Engineering
  + Skills and practice I had gone through in learning how to create a proper user cases is being used here.

# Design Phase

Once the requirements phase is completed, it is time to move on to the design phase. In the design phase, I will be creating prototypes with good usability. This section explains in details of what was being done during the design phase, how the requirements were transformed into high-level prototype. It will also describe certain usability principles as to why certain elements of the webpage are of this manner.

# High-Fidelity Prototyping

Normally the prototyping would start from low-fidelity prototyping before proceeding on to high-level prototyping. For this project I skipped the low-fidelity prototyping portion because of the simplicity of the project that requires only 2 webpages which does not require any complex drawings or user actions on the landing page. By jumping straight to high-fidelity prototyping, it also saves me time so that more time can be allocated towards the implementation phase.

For this project, the high-fidelity prototyping will be done using a tool called Justinmind. Justinmind is an authoring tool for web and mobile app prototypes and high-fidelity wireframes. It is a flexible prototyping tool that supports many devices and works well for simple click-through prototypes or more complex interactions. Prototypes can be created from existing mocks or building one from scratch using the given library assets. The best part about this tool is that once you have completed the high-fidelity with the necessary interactions, a web preview of a fully interactive prototype with a device frame can be generated. Since this tool was used in one of the modules learned in year 2, it would be a perfect prototyping tool as there is no need for a learning curve.

I created three sets of prototypes, each catering to a different method of user providing personal information in exchange of internet access.

# No personal information provided in exchange for internet access

# 

#### Figure 3. Landing Page V1 Figure 4. Details Page

In this version of landing page, no personal information is required from the user, user is just required to click on the orange button and will be redirected to the page displaying the animal show and feeding details.

# Provide email address in exchange for internet access

# 

#### Figure 5. Landing Page V2 Figure 6. Details Page

In this version of landing page, the user is required to key in his/her email address follow by clicking on the orange button and will be redirected to the page displaying the animal show and feeding details.

# Log-in to social media account in exchange for internet access

# 

#### Figure 7. Landing Page V1 Figure 8. Details Page

In this version of landing page, the user has a choice of entering an email address follow by clicking on the orange button or to social sign-in to Facebook/Google account and will be redirected to the page displaying the animal show and feeding details.

Knowledge learned in school that was applied in this section:

* ICT 2102 – Human Computer Interaction
  + Skills and practice into learning how to use Justinmind to create wireframes is being applied here.

# Usability Principles

In this section, the rationale of why the design of the captive portal in this manner will be further explained through the usage of usability heuristics.

# Usability Heuristic – Consistency and standards

#### Figure 9. Landing Pages

Consistency means that there is some form of harmonious uniformity or agreement among things or parts. As shown in Figure 9, notice how the body content in each of the pages is contained within the same width, this is to give a uniform look and feel across all the webpages. It would not look nice if the body content is of different sizes, this would make users feel odd and less inclined to visit the website.

# Usability Heuristic – Error Prevention



#### Figure 10. Details Page

Error prevention means that there is some form of design and logic to prevent a problem from occurring in the first place. Since the purpose of this page is to display the details of animal shows and feedings, timings will be involved. This also means that the animal shows or feedings will not happen all day round, therefore there needs to be a message to indicate to users that there are no more shows or feedings for the day as seen in Figure 10.

# Usability Heuristic – Aesthetic and minimalist design

#### Figure 11. Landing Pages

Aesthetic and minimalist design means that the webpage should not be overly cluttered with loads of unnecessary information which makes it hard for the users to read. The webpage should be nice and clean, populated with the necessary information to get the message or intention across to the user. Notice from Figure 11, all the webpages are not populated with unnecessary information, it only contains information that is relevant to each individual webpage.

Knowledge learned in school that was applied in this section:

* ICT 2102 – Human Computer Interaction
  + The design principles and usability principles learned in this module is being applied here.

# Relevant Literature

This section describes the comparison made between the project’s captive portal and other captive portals. The scope of comparison would be to compare captive portals located in Singapore. There are many different type of captive portals some of which include Application based access, Short Duration based access, registration based and no registration based.

# Application Based Access

Application based captive portal is one of the ways to implement a captive portal to captivate user to register an account with the application in exchange for free wifi. It is a great way to collect data on user’s personal information, but it becomes a hassle towards some users. This is because not all users are inclined to download an app just to use the free wifi in an area. However, there are solutions to curb this issue and that is to provide incentives when a user downloads an app and registers an account through the app. This solves the problems but also means that the pool of registered users will be relatively small. This type of captive portal is widely used in hotels as the perks can all be linked through the app.

The application based access method of implementing a captive portal will not bode so well visitors visiting the park. The purpose of visitors coming to the parks is to visit the animal exhibitions and have a memorable fun time, they will not want to download an app just to use the free wifi provided by the parks.

# Short Duration Based Access

Short duration based access is another way to implement a captive portal. It is normally implemented as a web application to captivate users requiring users to exchange their personal information for the usage of free wifi. However, things are a little different here. Compared to the rest of captive portals, if the user is granted access to the wifi, the duration users can use is unlimited but not for the short duration based access. In short duration based access, users who are granted access to use the free wifi will be limited to a finite amount of time before it expires. It also comes with tier based access in which the usage duration of wifi depends on the type of account registered with the business.

This type of captive portal will not really bode well with visitors in the parks. Because of the location of the parks, the surrounding areas are mainly forested areas, and this means network connectivity from the sim card may get weaker as visitors venture deeper into the parks. If this type of captive portal were to be implemented, it will degrade the visitor’s experience which does not fit with the value of enhancing visitor’s experience. This type of captive portal can also be widely seen in cafes.

# Registration Based Access

Registration based access works almost as the short duration based access except that users who are granted access to the wifi network will have unlimited duration of wifi usage. The key difference is that users will have to provide some form of personal information in exchange for usage of free wifi. This type of captive portal is normally implemented on a web application so that the landing page of the captive portal will automatically launch when users connect to the wifi network. Personal information such as email address collected via the captive portal can be used to create new user accounts and e-newsletter can be emailed to them.

This type of captive portal would prove to be useful to visitors in the parks, as they do not have to worry about downloading an app just to use free wifi or worrying about the limited duration to the usage of free wifi. On top of that users can receive e-newsletter through their email and be updated on any promotions or events happening in the parks. This will be implemented in phase 2 of this project. This type of captive portal can be widely seen in shopping malls.

# No Registration Based Access

No registration based access works the same as registration based access except the fact that no personal information is required from the user in exchange of the usage of free wifi.

Users will just have to agree to the terms and conditions, click a button and they will be granted access to use the free wifi. This is the easiest and simplest way of implementing a captive portal. This will appeal to visitors in the park as they do not have to provide any form of personal information and will be implemented in phase 1 of this project.

# Implementation Phase

This section explains in details of how the project is to be implemented, the type of codes that is to be used and the explanation of user features with reference to the codebase (documentation).

# Implementation Details

Once the design phase is over, implementation phase will take over. In the implementation phase, I will be dwelling on the technical aspect of the project, to implement the captive portal by writing codes.

This project will be implemented in 2 phases, Phase 1 and 2. Phase 1 implementation will be focused on the no registration based access type of captive portal whereby no personal information is required from the user in exchange for free wifi. Phase 2 implementation will be focused on the registration based access where by users will have a choice to either provide their email address in exchange for usage of free wifi or social sign-in to their social media account in exchange for usage of free wifi. Phase 1 has been completed and Phase 2 is still work in progress. Next details on the codebase will be mentioned. After every implementation, the captive portal will be checked against a set of test cases to ensure that the requirements are met.

# Codebase

This section will explain in detail the various types of codes used to implement the captive portal which covers the area of front-end, back-end and database areas. It includes justification as to why certain languages are used over others.

For the front-end portion, the following codes will be used:

* HTML5
  + To define the components of the webpage
* CSS3
  + To add styling to the components of the webpage
* Flexbox
  + To add layout to the components of the webpage similarly to bootstrap

The rationale behind choosing the above mentioned front-end codes is because the nature of the captive portal is to be implemented as a web application, the easiest and simplest way to is to use HTML5 and CSS3 as that is the basis of a typical front-end webpage.

For the back-end portion, the following codes will be used:

* JavaScript
  + To apply to the HTML document to provide dynamic content change on websites
* Firebase Realtime Database
  + Act as an online storage to store the shows and feedings details aside from the images

Using Firebase Realtime Database is a good option as it would come in handy in the event where a certain action needs to be performed whenever there is a change of data in the database. Aside from that, it can function as a normal database to store data and in this case details of animal shows and feedings. Another advantage of Firebase Realtime Database is the structure of the stored data. Data in Firebase Realtime Database is stored in a JSON format which makes the data easily readable. Data stored in the JSON structure are also associated with a key, making it a key value pair. (See [Appendix A](#_Appendix_A)) The very first level of data is called the parent node which can potentially have 0 or more children nodes (See [Appendix B](#_Appendix_B)). Whenever data is requested to be retrieved, providing the key will return the value (the actual data). This makes the data stored very easy to be referenced and retrieved.

On top of that with the help of JavaScript, it can be applied to any HTML document to provide dynamic content change on the HTML components, it is able to dynamically populate data retrieved from Firebase realtime database onto the HTML webpage.

Knowledge learned in school that was applied in this section:

* ICT 1004 – Web Systems and Technologies
  + Languages learned in this module to implement a full web application is applied here
* ICT 2103 – Information Management
  + The understanding of how data is being stored and structured in a database is used here
  + The understanding of how to create a database and populating it with data is also applied here

# Documentation

Codes used in this project comes with comments and documentation. Comments will reside on the front-end portion while on the back-end portion, it includes both comments and documentation. The comments give a brief description of what this line of code is doing while documentation gives a top-down detailed description of what this chunk of code is supposed to do. In this section, the functions that is responsible for retrieving data of shows, feedings from Firebase and identifying the current time will be explained here. The functions all resides in a single JavaScript file.

Before jumping into the explanation of the functions responsible for retrieving data from Firebase, Firebase will first have to be initialized within the HTML codes. The purpose of the initialization codes is to configure the Firebase JavaScript SDK to use the features of Firebase. This can be seen from lines 43 to 51 where all the parameters are being specified and passed to a function Firebase.InitializeApp(parameters) (See [Appendix C](#_Appendix_C)).

Each line of the firebase configuration codes will be explained more below. (See [Appendix C](#_Appendix_C))

Line 43 🡪 var config {}

* Create a variable named config that is of tuple type which will contain key value pairs, found in lines 44 to 49

Line 44 🡪 apikey: “some value in string”

* It is some form of secret key used to identify the origin of the request and is required the web application is trying to request data from the database

Line 45 🡪 authDomain: “Domain name”

* To let the web application know which domain to route to when accessing Firebase

Line 46 🡪 databaseURL: “Actual URL of the database”

* The actual URL of the database for the web application to visit

Line 47 🡪 projectID: “ID of the project”

* Unique identification key of the project

Line 48 🡪 storageBucket: “Some form of URL”

* URL to the online storage of Firebase to store files, images and etc…

Line 49 🡪 messagingSenderId: “some form of key”

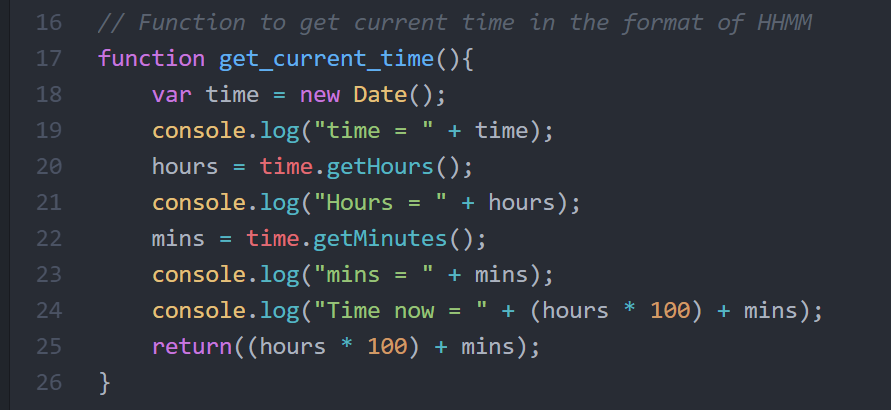
* Identification key for the messaging feature in Firebase

Line 51 🡪 firebase.initializeApp(config)

* Call the function to initialize the web application with Firebase based on the key value parameters located on config tuple variable

Once this has been initialized, the features and services of Firebase can be used. For this project, the only feature that will be focused on is the Firebase Realtime Database responsible for storing the information regarding the animal shows and animal feedings.

The first function that will be explained here is the function responsible for identifying the current time.

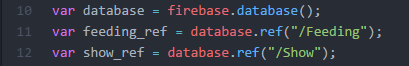


#### Figure 12. get\_current\_time function

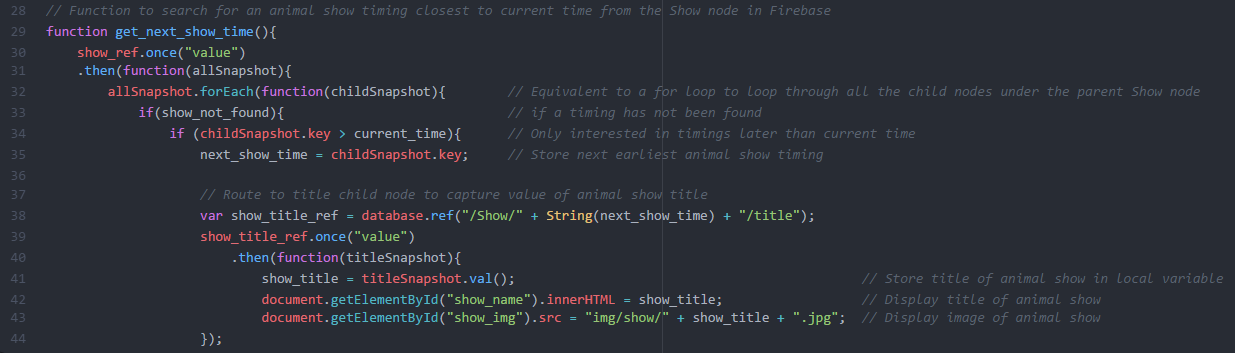
This function takes in no parameter but returns the time in 24hr format. First, it calls the Date() function and saves it onto a variable called time. Secondly, the hours and minutes are retrieved by calling the respective functions .getHours() and .getMinutes(). Lastly, the hours and minutes collected go through some calculation to convert the 2 values into a 24hr format time and return the value. Console.log() is mainly used to print out the values into the console for checking purposes to see whether if I am getting what I want.

The second function that will be explained here is the function responsible for retrieving data from the Show node. The same explanation will apply for the function responsible for retrieving data from the Feeding node, logic is the same except that it is referencing from a different node.

As mentioned earlier, to retrieve data from Firebase, a node must be referenced first. To maintain code reusability, I created 3 global variables, one for the database itself, one for parent node show and one for parent node feeding. This can be seen from Figure 12 as shown below in lines 10 to 12.



#### Figure 13. Database reference global variables

The first half of codes in the function looks like this:

#### Figure 14. get\_next\_show\_time function

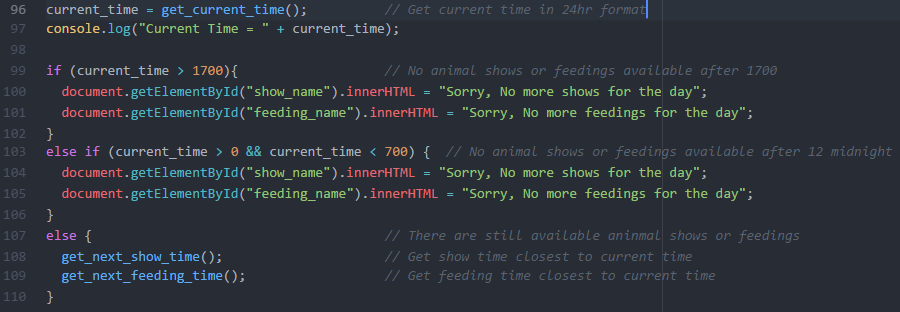
As shown in Figure 14, the first section of codes (lines 30 to 34) creates a for loop to loop through all the child nodes under the Show node. All child nodes are structured in a way there it is also a parent node for the data nested within it. At the parent node level, it contains the time of the animal show and the child nodes contains the details of the animal show, title and location. At every child node, it checks for 2 things, has the show been found yet and is the value of the show timing greater than the current time. Only when both checks results in a true, the code nested in it will be executed.

The second half of codes in the function looks like this:

#### Figure 15. get\_next\_show\_time function

Next step would be to retrieve the details of the animal show from the child nodes nested within the animal show time. Line 38 and 47 shows the code to reference to the exact location of title and exhibit. The reason I could not globally declare these 2 variables is because the show time, a child a under the feeding parent node is required in order to reference the data nested within it, hence it has to locally declared within the function get\_next\_show\_time(). Lines 39 to 41 and lines 48 to 50 shows the actual code to retrieve both the title and exhibit data. Once the data has been retrieved, the data can be displayed and rendered to the HTML page via JavaScript which can be seen in lines 42-43 and lines 51-52. Lastly since there is a for loop and needs to stop after data has been retrieved, the variable show\_not\_found is updated to the value False. This will stop the for-loop iteration and exit the function.

The codes mentioned above are the functions itself, in this section, the flow of code will be discussed like where would the functions declared be called.



#### Figure 15. Actual Code Flow

As seen in Figure 15, the first line of code (line 96) is to find out what is the current time in 24hr format and save it into variable. Next, the current time is used to check against the times 1700 and 0, 1700 representing 5pm and 0 representing 12am. The animal shows and feedings only happens between the timings of 9am to 5pm, so if the current time is out of that range, there will be a text message displayed on the webpage stating that “Sorry, No more shows/feedings for the day”. Lastly, if the current time is within the range of 9am to 5pm, there will be shows and feedings available and the functions to retrieve the details of animal shows and feedings needs to be called.

Knowledge learned in school that was applied in this section:

* ICT 1004 – Web Systems and Technologies
  + The knowledge of knowing how to use JavaScript to display content over to the HTML page
* ICT 2111 – Integrative Team Project
  + The practice I had in utilizing Firebase API to implement the web application as part of the project.

# Verification Phase

Once the implementation phase has come to an end, verification would need to be conducted to ensure that the requirements have been met. In this section I will be explaining the process of how I conducted the verification for this project.

# Test Approach

Functional testing is as an approach to test the captive portal. Functional testing is a software testing process used in which a software is testing to ensure that it conforms with all the specified requirements. It also ensures it has all the required functionality it is supposed to. In functional testing, there are 2 kinds of testing and black-box testing would be used to test that the software work as expected. It ensures that by checking that whatever is inputted should have an expected output and this will be checked against the actual output. To carry out black box testing, test cases are needed to ensure that there is a guide to follow, to know what needs to be tested based on the functional requirements gathered from the requirements phase.

# Test Cases

The test cases are linked back to the use cases identified in the requirements analysis section. This ensures traceability of test cases back to requirements and no requirements are left out from the test cases.

Test cases have a template to follow and will have the following fields:

* Pre-conditions
  + Conditions that need to happen for this test case can be carried out
* Dependencies
  + Preceding tasks or events that required to be completed before the starting the test case
* Use Case ID
  + To reference from the list of use cases to ensure that this test case is testing for the relevant requirement
* Test Case ID
  + To make sure that every test case can be identifiable
* Test title
  + To give an overview of what this test case is about
* Test Description
  + To provide a more detailed version of the test title to allow readers to understand what this test case is about
* Steps
  + To give a step by step guideline to follow to ensure the test results are not rigged
* Expected Result
  + What would the result look like in theory
* Actual Result
  + The actual result of the testing itself
* Status (Pass/Fail)
  + To indicate whether did this test case pass or fail.

An example of how a test case template look like would be this:

|  |  |
| --- | --- |
| Test Case ID |  |
| Use Case ID |  |
| Test Title |  |
| Test Description |  |
| Pre-conditions |  |
| Dependencies |  |
|  | |
| Test Step |  |
| Expected Result |  |
| Actual Result |  |
| Status |  |

#### Table 1. Black-box test case template

In this section, the actual test cases are displayed here to show the results of the test cases.

* TC-1 – Users wishes to view the terms and conditions on the landing page

|  |  |
| --- | --- |
| Test Case ID | TC-1 |
| Use Case ID | UC-1 |
| Test Title | View terms and conditions |
| Test Description | Users wants to view the terms and conditions when at the landing page |
| Pre-conditions | 1. Needs a smart device like a mobile phone   Needs to connect to the PARK\_FREE\_WIFI SSID |
| Dependencies | Specified requirements |
|  | |
| Test Step | 1. User click on the terms and conditions link on the landing page 2. User read through the terms and conditions |
| Expected Result | Upon clicking the link, user will be redirected to the page where it lists out the conditions of using the wifi |
| Actual Result | The link was clicked, and was redirected to another page where the terms and conditions were stated |
| Status | **Pass** |

* TC-2 – Users wishes to view the terms and conditions on the landing page

|  |  |
| --- | --- |
| Test Case ID | TC-2 |
| Use Case ID | UC-3 |
| Test Title | View animal show details |
| Test Description | Users able to view the animal show details displayed on the webpage |
| Pre-conditions | 1. Needs a smart device like a mobile phone   Needs to connect to the PARK\_FREE\_WIFI SSID |
| Dependencies | Specified requirements |
|  | |
| Test Step | 1. User click on button “I WANT FREE WIFI” 2. View the animal show details displayed on the webpage |
| Expected Result | Upon clicking the button, user will be redirected to a new page where the animal show details are displayed |
| Actual Result | The button was clicked, it redirects to a new page and displays the animal show details |
| Status | **Pass** |

* TC-3 – Users wishes to view the terms and conditions on the landing page

|  |  |
| --- | --- |
| Test Case ID | TC-3 |
| Use Case ID | UC-4 |
| Test Title | View animal feeding details |
| Test Description | Users able to view the animal feeding details displayed on the webpage |
| Pre-conditions | 1. Needs a smart device like a mobile phone   Needs to connect to the PARK\_FREE\_WIFI SSID |
| Dependencies | Specified requirements |
|  | |
| Test Step | 1. User click on button “I WANT FREE WIFI” 2. View the animal feeding details displayed on the webpage |
| Expected Result | Upon clicking the button, user will be redirected to a new page where the animal feeding details are displayed |
| Actual Result | The button was clicked, it redirects to a new page and displays the animal feeding details |
| Status | **Pass** |

Knowledge learned in school applied in this section:

* ICT 2101 – Introduction to Software Engineering
  + The knowledge of how to create a proper black-box test cases is required here in this section

# Technical Challenges

This section describes any notable events where I face technical challenges and what steps I took to overcome the challenges.

# Technical Challenge 1 – Unable to retrieve the correct data from Firebase

# Description of challenge

During the implementation phase, when I was coding the logic to retrieve the data from Firebase to display on the HTML through JavaScript, I realised that I was getting empty values displayed on the HTML page. Immediately, I thought that there might 2 causes to this problem. This could be due to the wrong usage of API, meaning I may not be using the right set of functions to retrieve the data. This could also be due to the error in my logic in retrieving the data.

# How I solved the issue

Based on those 2 causes mentioned in the previous paragraph, I started reviewing the portions of code that could have been affected. The first round of check was to relook through the Firebase API documentation to ensure that I was using the right set of functions to retrieve the data. To ensure that I was right, I included this line of code “console.log()” to print out the variables to make sure that I was getting the correct data. It proved me right that I was using the right set of functions to retrieve data from Firebase.

I then move on to the second issue that could be causing this issue. I rerun through the flow of my code to check my logic to see if it makes sense. During the check, I happened to chance upon my mistake. The reason I was getting empty values displayed on the HTML page was because I was using a locally declared variable outside of its scope to display the data on the HTML page. I corrected my error and recheck to make sure that everything was working fine.

# Technical Challenge 2 – Unable to centre centre items using Flexbox

# Description of challenge

During the implementation phase, when I was writing CSS codes to style and align HTML components, I had a problem of not being able to make certain components to be horizontally centre and vertically centre. I immediately knew that my error lies in the CSS code I wrote.

# How I solved the issue

During my first check, I could not find out what was wrong the CSS code, why it would not work and display the effect it was supposed to have when applied. So, I went online to search for tutorials to deepen my understanding on how to use Flexbox to align HTML components. To my surprise, I found the answer I was looking for. The reason why I could not get the HTML component to be in the centre centre was because I did not declare in my CSS code for that component to have flex properties. I was missing this line of code 🡪 display: flex; I edited the code and now the HTML component is in the centre centre.

# Conclusion

This project has taught me a lot of things. One of the valuable lessons worth mentioning was the development of self-troubleshooting skills, to try to resolve the issue as much as I can on my own before asking others for help. There certainly was a lot of time to practice and polish that skill during this IWSP period.

**Contract Management System**

# Specialised Terminology

|  |  |  |
| --- | --- | --- |
| Abbreviated | Full Name | Description |
| Django | Django | Free and open source web framework, written in Python which follows the model-view-template architectural pattern |
| SQL | Structured Query Language | A standard language for relational database management systems such as MySQL |
| MVC | Model View Controller | A form of design pattern used to create web applications |
| MTV | Model Template View | A form of design pattern Django used, similar to MVC to create web applications |
| FDD | Feature Driven Development | A lightweight iterative and incremental software development process. |
| Anaconda | Anaconda | An open source package management system and environment management system |
| CMS | Contract Management System | The implemented system |

# Introduction

This section provides an overview of what the project, the way it works from a broad picture and how relevant in the industry in today’s world.

# What is the project about?

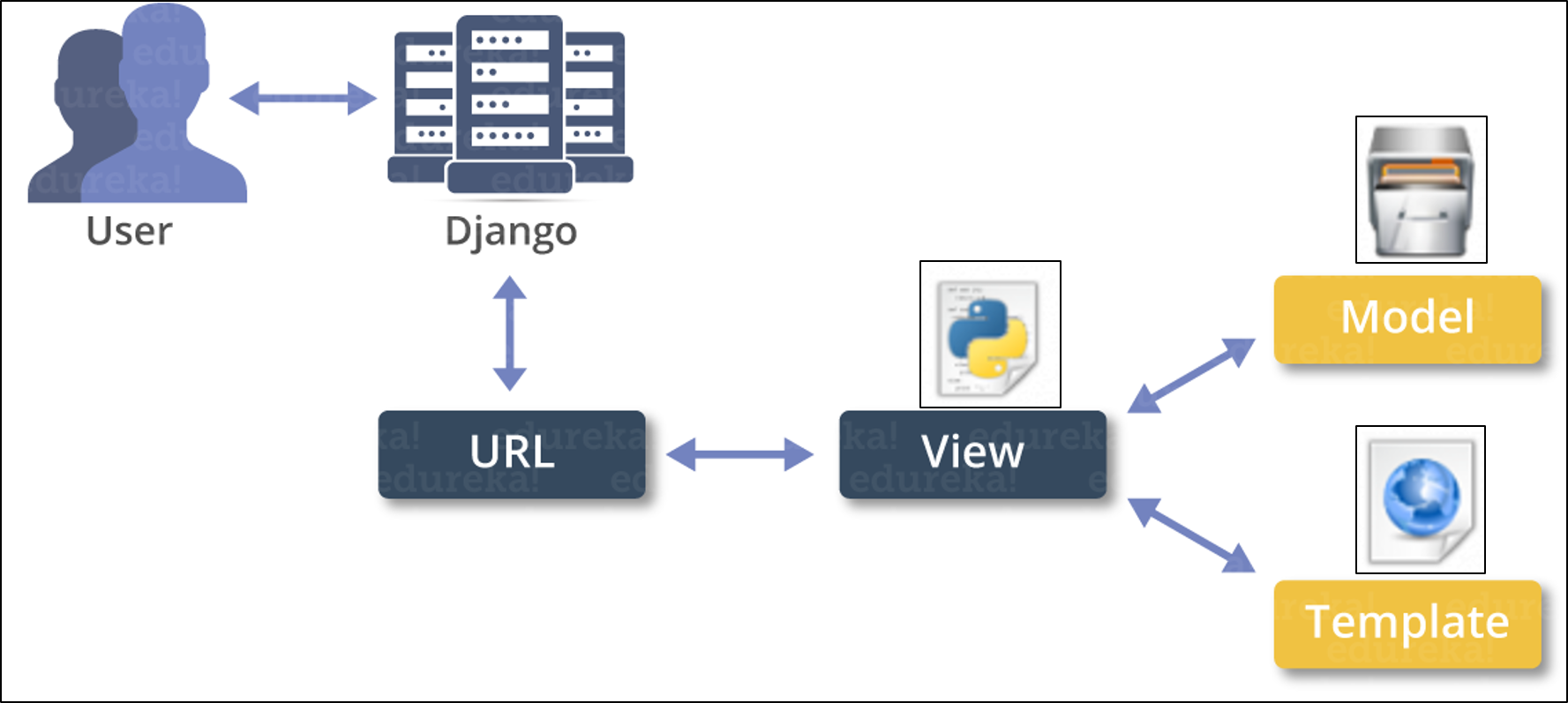
Now, there isn’t a management system to manage and track the contracts. A lot of manual tasks are involved, and it is difficult to keep track of the contract status and this results in lower productivity. However, such concerns are hardly ever being raised to the higher management until recently. There was a new initiative during this year called IT Masterplan and the objective of this initiative was to identify the pain points and wish lists. Pain points refer to the difficulties that staff faced when doing their work daily. Wish lists refers to what would be the ideal solution to resolve their pain points encountered during their daily job basis.

I also happened to be part of this project and I happened to chance upon on the wish list from one of the staff from the legal department. The wish list was to have a contract management system that can track the statuses of contracts. I decided to implement a contract management system as my 2nd capstone project. Basically, the contract management system contains a dashboard where it can track the statuses of all the contracts. The project will also undergo the software lifecycle model of waterfall.

# Complexity

Since it’s a contract management system and it involves some form of admin management, this means that there needs to be a admin portal to handle the data which is connected to a database. It would require a great deal of time if I were to build those from the scratch. Fortunately, there are wonderful web frameworks to help with that. Django is one such great example.

Django is a free and open source web framework that is written in python language and it follows the model-view-template architectural pattern. It is suitable for building dynamic websites fast. It has a built-in admin portal and database. The admin portal acts as a platform for administrators to mange the data in it. The built-in database that comes with Django is of a SQL database known as dbsqllite3, a minimised version of the normal SQL database.

An django architecture diagram will give an overview of how it works and how data flows.

#### Figure 16. Django Architecture Diagram

Django follows the MVC architectural pattern and MVC pattern is widely used in many of the web applications and is also considered one of the best design patterns when it comes to creating web applications. However, django used a slightly different design pattern from MVC, it uses MTV, model template view as seen in Figure 16. MTV is very similar to MVC and provides the same functions.

As shown in Figure 16, whenever a user requests for a webpage, django will look for the requested URL from the urls.py file which contains the function to be called. Django will then look this function in views.py which contains the logic to access the necessary data from the data via the model and also which template to use and return back to the user. The next few sections will explain in detail the difference between MVC and MTV.

In normal web applications using the MVC design pattern:

* The model(M) is a model or representation of the data. It is not the actual data, but it serves as an interface to the data. The model allows data to be pulled from the database without knowing the underlying structure of the database. The model also provides an abstraction layer with the database, so this means that the model can be related with multiple databases.
* The view(V) is basically what a user can see. It serves as the presentation layer of the model. Whatever you see on a webpage on a browser is the view.
* The controller(C) is where the logic resides in. it controls the flow of information between the model and the view. The logic will determine what kind of data to pull from the database via the model and what information should be passed to the view to be displayed. It can also work the other way around. The logic in the controller can also collect information from the user via the view and perform some form of modification to the data via the model.

In django web applications using the MTV design pattern:

* The model(M) behaves the same as the model in MVC pattern. It contains everything about the data: how to access it, how to validate it, and the relationships with the data.
* The template(T) also behaves the same way as the view in MVC. It also servers as the presentation layer, basically depicting how should the layout of the web application should look like.
* The view(V) in MTV works differently from the view in MVC. Over in MTV, the view acts as the logic layer, similarly to how the controller works in MVC. It contains the logic that bridge the gap between the models and templates, telling the web application how to access data via the model and display it onto the template.

# Industry Relevance

Having a Contract Management System bring about a lot of benefits to the business.

The benefits are listed in bullet points and are as follows:

* Increase Contract Visibility

As files are stored in one centralized document repository, it allows for optimum compliance. It can also make sure that the staff is working on the latest version of the document

* Improve Document Management

Times of cumbersome manual filing of paperwork are over. Users using the contract management system can now upload the documents into a centralized location or retrieved documents from it.

* Decrease Approval Times

With the dashboard showing how many documents there are of each status, users can have a better understanding of the current situation at hand and quickly notify the responsible parties to act on it. This will improve the overall efficiency.

* Accessibility

Users of the contract management system can have an easier time when they need to search for contracts from the huge list of contracts. It simplifies the search process and saves lots of time. This also will in turn improve the productivity of the business.

# Requirements Phase

This section provides an overview on the requirements phase, which explains in detail how requirements were gathered, how requirements were analysed.

# Requirements Gathering

To kick start this project, I had to gather requirements to formalize the necessary details essential for the subsequent phases. For this project, requirements were gathered from the legal department. This time round, the requirements were more focused on the functional side and less on the design. The design however should not be a clean and nice design, not over cluttered with information.

The requirements gathered from legal department were as followed:

* There needs to be a dashboard to give user an overview of all the contract status
  + This is to allow the user to have a grasp of the how many contracts needs urgent action just at a glance
  + The status displayed on the dashboard should be separated from one another
  + The number of contracts under each status must be displayed
  + The dashboard should not be overly cluttered with information, it should be nice and clean.
* There needs to be a page where I can upload the contract document with other relevant information and they include:
  + Title
  + Description
  + Status
* There needs to be a admin portal where an administrator can log-in to manage the contracts
  + Administrator must be able to create a new contract record
  + Administrator must be able to view a list of contracts and their respective status
  + Administrator must be able to search for individual contracts and view it
  + Administrator must be able to edit the contract information and save the changes
  + Administrator must be able to delete contracts from the admin portal
  + Administrator must be able to sort the list of contracts according to the fields available
* Every contract can have 5 different type of status:
  + **Draft Status** – In the midst of making amendments, document not uploaded yet
  + **Pending Approval** – Amendments made, document has been uploaded, however changes made to the document are pending for approval
  + **Pending Acceptance** – Pending signature from the relevant party to formally accept the contract document
  + **Active** – Contract is active between the seller and buyer
  + **Closed** – Contract has expired or has passed the valid date
* There needs to be a navigation bar to navigate around the features of the system
  + Navigation bar needs to be at the top of the webpage
  + To navigate around the various pages
    - Dashboard page
    - Upload page
    - Administrator site
* Actions performed by the user cannot be performed by the administrator
* Actions performed by the administrator cannot be performed by the user

Knowledge applied during requirements gathering:

* ICT 2101 – Introduction to Software Engineering
  + The basic understanding of what user requirements are is required in this section
* ICT 2108 – Software Modelling Analysis
  + The knowledge of asking stakeholder questions to gather the relevant requirements is required in this section

# Requirements Analysis

After gathering the requirements, I had to analyse the requirements. This section will explain in detail how the requirements were analysed from what has been gathered. Analysis such as creating user cases and user stories will be used to analyse the requirements.

# User Stories

This section will list out the user stories to describe the needs of the users using the contract management system. It also gives a high-level description of a scenario and enables readers to immediately understand what it means just by reading it. It uses the template of **as a, I want to, so that.** There will be 2 kinds of users in this system, user and administrator.

The user stories are as follows:

* **As a user**, **I want to** be able to view the number of documents in each status from the dashboard **so that** I know the number of contracts that require urgent attention and give a quick update to the higher management
* **As a user**, **I want to** be able to create a new contract record and upload the document **so that** I can track its status
* **As a user**, **I want to** be able to create a new contract record without uploading the document **so that** I can track its status
* **As a user, I want to** be able to navigate through the different pages of the webpage, **so that** I can perform certain actions
* **As an administrator, I want to** be able to view the full list of contracts available in the admin portal, **so that** I can manually search for a contract record
* **As an administrator, I want to** be able to search for an individual contract record on a search bar, **so that** it shortens my time to locate a specific contract
* **As an administrator, I want to** be able to perform create operations **so that** I can create a new contract record in the admin portal
* **As an administrator, I want to** be able to perform update operations **so that** I can make changes to the contract record in the admin portal
* **As an administrator, I want to** be able to perform delete operations **so that** I can delete contract records which are of no longer of use
* **As an administrator, I want to** be able to sort the list of contracts using the fields available **so that** I can narrow down the search when looking for a contract document

# Use Cases

This section will display the functional requirements of the project in a visual manner to have a better understanding of what actions can the user perform when using the contract management system. It describes the behaviour that will be built into the software to meet the required needs. It also describes the complete interaction between the software and the users (and possibly other systems). Each use case will be accompanied with a detailed description of what the use case is about.

# Use Case Diagram

# 

#### Figure 17 Contract Management System Use Case Diagram

As shown in Figure 17, it lists out all the functional aspects of the Contract Management System that the users and administrators can perform. The functions between the 2 roles drew a clear line. Functions that the administrator can perform cannot be performed by the user themselves. Functions that the user can perform cannot be performed by the administrator. Although it makes sense that the administrator should have access to all functions, but in the requirements, it is a very clear that the administrator can only perform actions within the job scope.

The use cases for **users** are as follows:

* UC-1

|  |  |
| --- | --- |
| **Name** | **UC-1: View number of documents in each status in dashboard** |
| Summary | Users can view how many documents are there in each status |
| Rationale | To have a quick understanding on the overall status and give an update to the higher management |
| Users | Users |
| Preconditions | -NIL- |
| Basic Course of events | 1. User launch web browser 2. User goes to Contract Management System dashboard 3. User views the number of documents in each status |

* UC-2

|  |  |
| --- | --- |
| **Name** | **UC-2: Create new contract record and upload document** |
| Summary | Users can upload the document and create a new contract record |
| Rationale | To track the status of the new contract record |
| Users | Users |
| Preconditions | Contract document ready to be uploaded |
| Basic Course of events | 1. User launch web browser 2. User goes to Contract Management System dashboard 3. User goes to upload page 4. User input the necessary information, uploads the contract document and click upload |

* UC-3

|  |  |
| --- | --- |
| **Name** | **UC-3: Create new contract record without uploading any document** |
| Summary | Users can create a new contract record without uploading the document as document is still in draft status |
| Rationale | To track the status of the new contract record |
| Users | Users |
| Preconditions | Contract document ready to be uploaded |
| Basic Course of events | 1. User launch web browser 2. User goes to Contract Management System dashboard 3. User goes to upload page 4. User input the necessary information, leaves the document upload field blank and click upload |

* UC-4

|  |  |
| --- | --- |
| **Name** | **UC-4: Navigate through the different pages of the webpage** |
| Summary | Users can go to the different pages of the webpage via the navigation panel |
| Rationale | So that users can easily navigate around the contract management system |
| Users | Users |
| Preconditions | -NIL- |
| Basic Course of events | 1. User launch web browser 2. User goes to Contract Management System dashboard 3. User click on the links to navigate to the different pages of the system |

The use cases for **administrators** are as follows:

* UC-5

|  |  |
| --- | --- |
| **Name** | **UC-5: View full list of contract records** |
| Summary | Administrator can view the full list of contracts when they have gained access to the system |
| Rationale | So that administrator can browse through the full list of contract records |
| Users | Administrators |
| Preconditions | Administrator has an account in the admin portal  Administrator logged into the admin portal |
| Basic Course of events | 1. Administrator launch web browser 2. Administrator goes to Contract Management System dashboard 3. Administrator clicks admin 4. Administrator is redirected to admin portal login page 5. Administrator logins 6. Administrator views the full list of contract records |

* UC-6

|  |  |
| --- | --- |
| **Name** | **UC-6: Search for individual contract record** |
| Summary | Administrator can search for an individual record in the admin portal |
| Rationale | To view the details of that individual contract record |
| Users | Administrators |
| Preconditions | Administrator has an account in the admin portal  Administrator logged into the admin portal |
| Basic Course of events | 1. Administrator launch web browser 2. Administrator goes to Contract Management System dashboard 3. Administrator clicks admin 4. Administrator is redirected to admin portal login page 5. Administrator logins 6. Administrator search for the individual record and view the details |

* UC-7

|  |  |
| --- | --- |
| **Name** | **UC-7: Perform create contract record operation** |
| Summary | Administrator can create a new contract record in the admin portal |
| Rationale | To track the status of the newly created contract record |
| Users | Administrators |
| Preconditions | Administrator has an account in the admin portal  Administrator logged into the admin portal |
| Basic Course of events | 1. Administrator launch web browser 2. Administrator goes to Contract Management System dashboard 3. Administrator clicks admin 4. Administrator is redirected to admin portal login page 5. Administrator logins 6. Administrator creates a new contract record |

* UC-8

|  |  |
| --- | --- |
| **Name** | **UC-8: Perform update contract record operation** |
| Summary | Administrator can update an existing contract record in the admin portal |
| Rationale | To allow changes to be made to the contract record in the event there is wrong information |
| Users | Administrators |
| Preconditions | Administrator has an account in the admin portal  Administrator logged into the admin portal |
| Basic Course of events | 1. Administrator launch web browser 2. Administrator goes to Contract Management System dashboard 3. Administrator clicks admin 4. Administrator is redirected to admin portal login page 5. Administrator logins 6. Administrator search for that particular contract record 7. Administrator updates an existing contract record |

* UC-9

|  |  |
| --- | --- |
| **Name** | **UC-9: Perform delete contract record operation** |
| Summary | Administrator can delete an existing contract record in the admin portal |
| Rationale | To allow contract records to be deleted if there is a need to delete |
| Users | Administrators |
| Preconditions | Administrator has an account in the admin portal  Administrator logged into the admin portal |
| Basic Course of events | 1. Administrator launch web browser 2. Administrator goes to Contract Management System dashboard 3. Administrator clicks admin 4. Administrator is redirected to admin portal login page 5. Administrator logins 6. Administrator search for that particular contract record 7. Administrator deletes an existing contract record |

* UC-10

|  |  |
| --- | --- |
| **Name** | **UC-9: Sort full list of contract records** |
| Summary | Administrator can sort the fill list of contract records in the admin portal |
| Rationale | To organize the list of contract records to have a better view |
| Users | Administrators |
| Preconditions | Administrator has an account in the admin portal  Administrator logged into the admin portal |
| Basic Course of events | 1. Administrator launch web browser 2. Administrator goes to Contract Management System dashboard 3. Administrator clicks admin 4. Administrator is redirected to admin portal login page 5. Administrator logins 6. Administrator clicks on one of the field in the list to sort |

Knowledge applied during requirements analysis:

* ICT 2101 – Introduction to Software Engineering
  + Skills and practice I had gone through in learning on how to create a proper use case is applied here

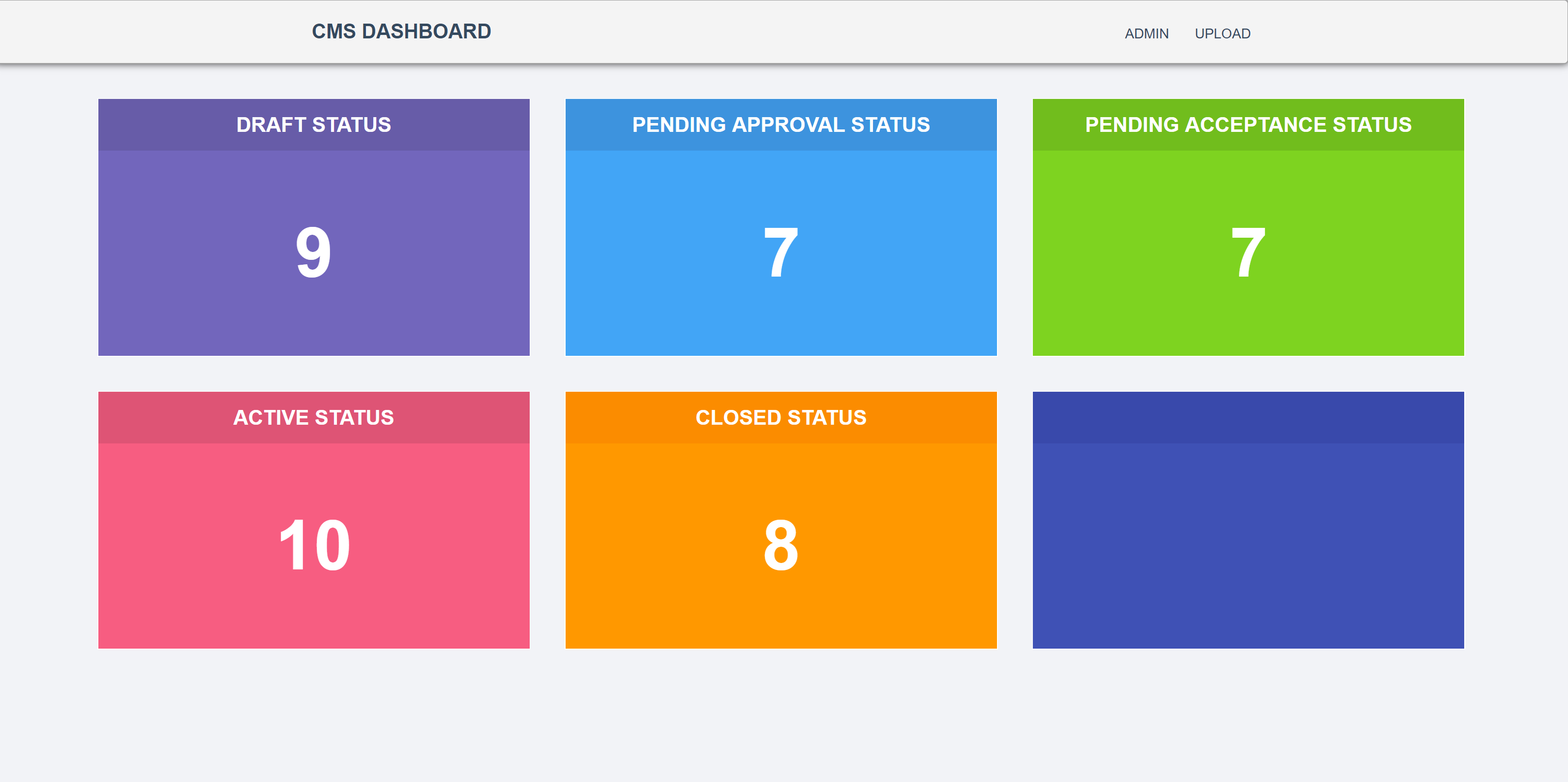
# Design Phase

Once the requirements phase is completed, it is time to move on to the design phase. In this phase, I will be explaining in detail about the various usability heuristics I applied in designing the contract management system.

# Usability Principles

In this section, the rationale as to why the design of the contract management system is being designed in such a manner will be further explained through the usage of usability heuristics.

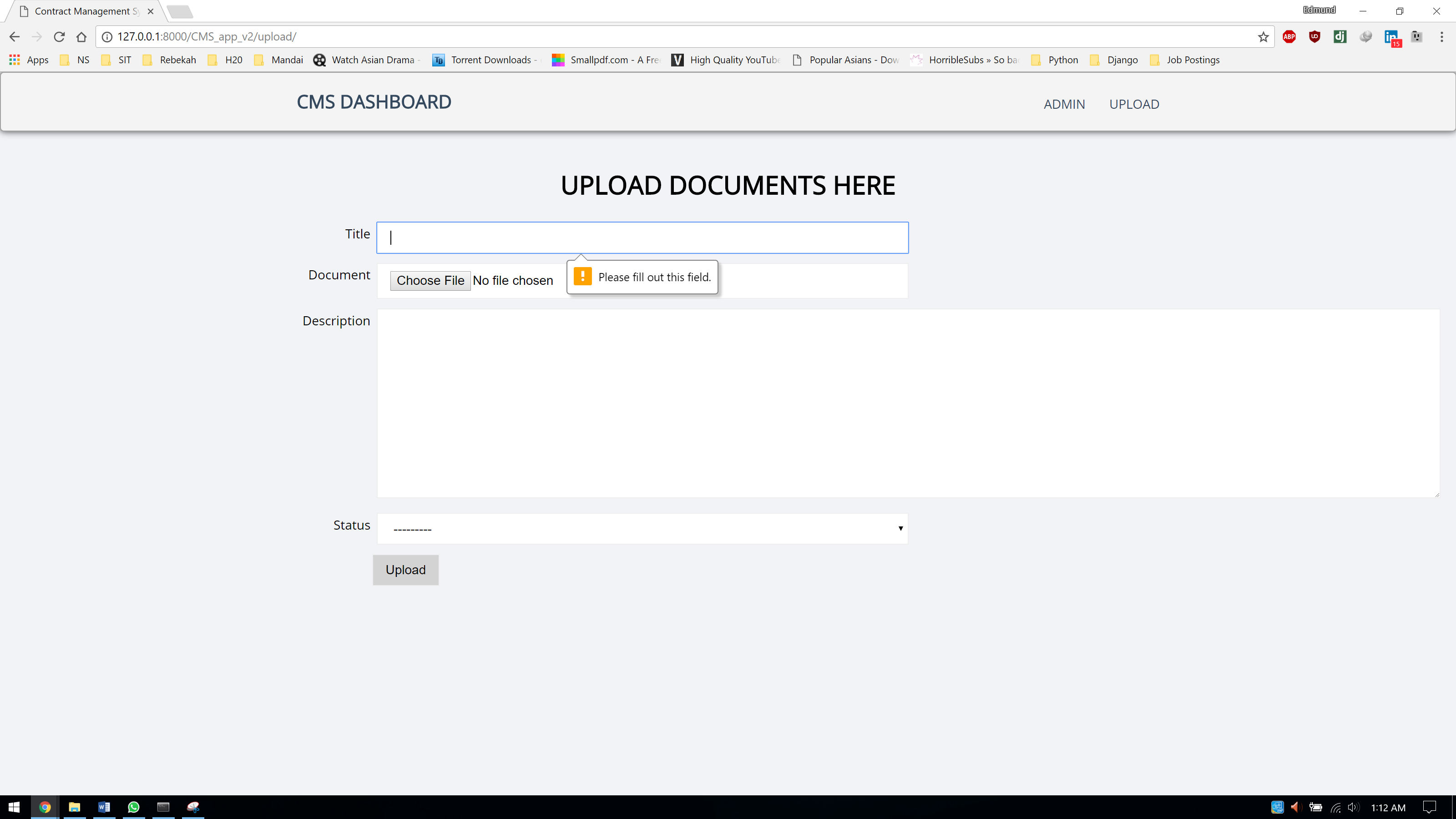
# Usability Heuristic – Consistency and Standard



#### Figure 18. Dashboard Page

Consistency means that there is some form of harmonious uniformity or agreement among things or parts. As shown in Figure 18, the navigation bar at the top of the page will stay the same throughout all the pages, giving it a uniform look. The content in the body however will vary depending which page it is at.

# Usability Heuristic – Error Prevention



#### Figure 19. Upload Page

Error prevention means that there is some form of design and logic to prevent a problem from occurring in the first place. Since the purpose of this page contains a form and allows users to create a new contract record with the relevant details and document, there will be certain fields in the form that is mandatory. Hence, we need some form of message to indicate to the user when he/she forgets to fill up a field in the form.

# Usability Heuristic – Aesthetic and minimalist design

Aesthetic and minimalist design means that the webpage should not be overly cluttered with loads of unnecessary information which makes it hard for the users to read. The webpage should be nice and clean, populated with the necessary information to get the message or intention across to the user. In the contract management system, all the webpages are not populated with unnecessary information, it only contains information that is relevant to each individual webpage. The dashboard page should only contain relevant information to give users an overview of the status of all contracts and the upload page should only contain a form for users to create a new contract record (See [Appendix D](#_Appendix_D))

Knowledge applied during the design phase:

* ICT 2102 – Human Computer interaction
  + The design principles and usability principles learned in this module are applied during the design phase

# Implementation Phase

This section explains in detail how the project was implemented, the type of codes that were used and the documentation where the codes will be further explained to provide a detailed description.

# Implementation details

As Django is new to me, I had to take time to go through the API documentation and also watch online tutorial videos to have a better understanding before I start the implementation. During the course of learning how to use Django web framework, I would try out mini projects to get myself familiarized and comfortable with the language before I start the actual work.

The way the contract management system is through the process of FDD. FDD is called Feature Driven Development and as the name suggests, the development is driven by features. I used this process because it is lightweight and easy to use. Since Django is python driven, it relies heavily on the version of python and django used. The correct version of python has to be installed with the correct version of Django otherwise it will not work. Because of this, having a virtual environment to implement the contract management system will be very useful. The virtual environment tool that is used for this project is Anaconda.

With Anaconda, it can easily create, saves, loads and switches between environments on the local computer. With different environments to switch around, each project on one environment can have a specialised python version and this will not affect the other projects in the other environments. The best part of all is that python can still be installed in the local computer directory.

# Codebase

This section will explain in detail the various types of code being used to implement the contract management system and it covers the area of front-end, backend-end and database. The framework that envelops everything is Django, which is written in Python language. It also includes the justification as to why these languages were chosen.

For the front-end portion, the following codes are being used:

* HTML5
  + To define the components of the webpage
* CSS3
  + To add styling to the components of the webpage
* Flexbox
  + To add layout to the components of the webpage similarly to bootstrap
  + Useful for creating dashboards where there is a lot of containers and grids involved to give the box look

The rationale behind choosing the above mentioned front-end codes is because the nature of the contract management system is to be implemented as a web application, the easiest and simplest way to is to use HTML5 and CSS3 as that is the basis of a typical front-end webpage.

For the back-end portion, the following codes are being used:

* Python
  + The underlying code that Django uses

Using Django saves implementation work lots of time as most of the codes and components of the webpage has been automatically created upon project creation. All I have to deal with is the code portion residing in the model, template and view files. Code reusability is widely practiced when building web applications with Django, this make the code very readable and maintainable.

Knowledge learned in school that was applied in this section:

* ICT 1002 – Programming Fundamentals
  + Skills and practice accumulated with coding in python has helped me a lot in this section
* ICT 1004 – Web Systems and Technologies
  + The skills learned in this module in building web application is required in this section
* ICT 2103 – Information Management
  + The basic understanding of how data is being stored and structured in a database is required here
* ICT 2106 – Software Design
  + The understanding of the MVC design pattern is required in this section to understand the MTV design patter Django is using
  + The fundamentals of good coding practices such as code reusability, coupling, cohesion is required during the implementation portion.

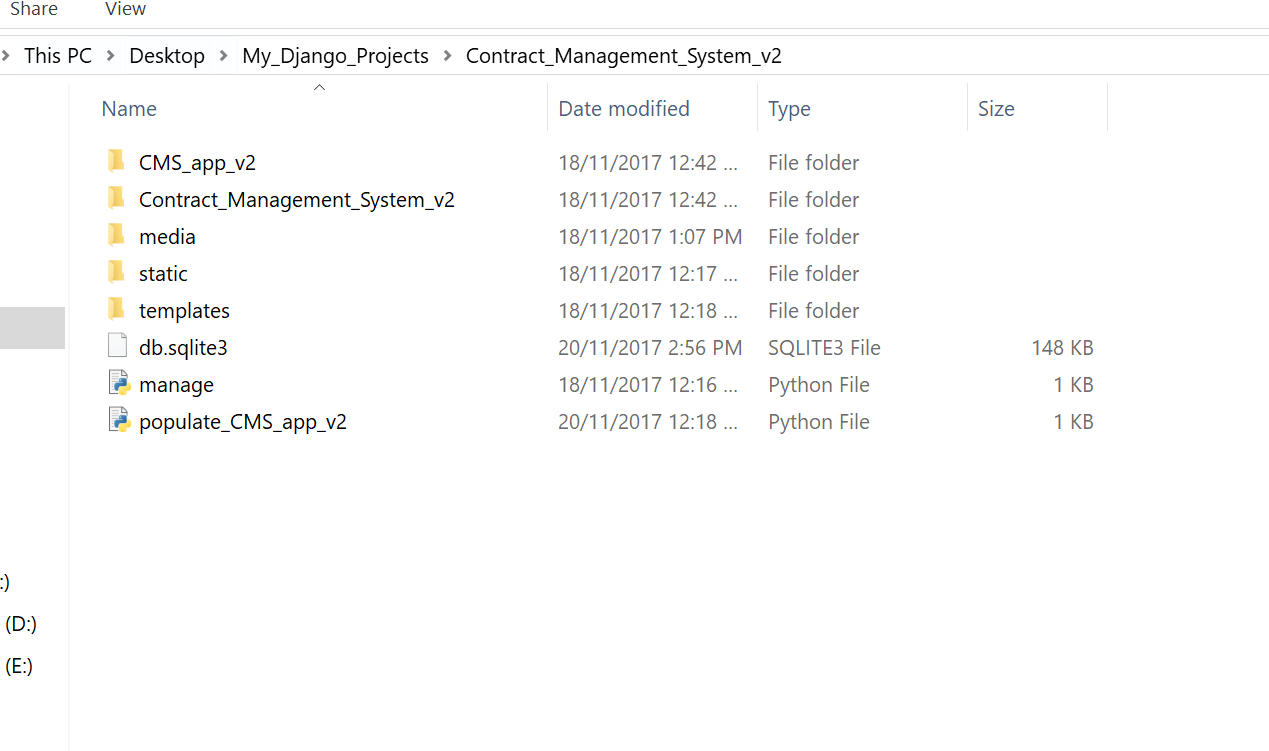
# Documentation

Codes used in this project comes with comments and documentation. Comments will reside on the front-end portion while on the back-end portion, it includes both comments and documentation. The comments give a brief description of what this line of code is doing while documentation gives a top-down detailed description of what this chunk of code is supposed to do.

Before jumping right into the details of the codes, we will need to set up a virtual environment in anaconda follow by calling some commands to create the project files. Follow the steps shown below:

* To create a new virtual environment, use “**conda create”.** For example, the command “**conda create –name snowflakes biopython**” creates a new environment named “snowflakes” with the program “biopython”.
* The environment has to be activated once it has been created. Use “**activate snowflakes”** to activate the newly created environment.
* Alternatively, an environment can be created with the specified python version. An example would look like this “**conda create –name bunnies python=3.5 babel**”. This means that an environment named bunnies is created with python 3 and babel program installed.
* Once the program has been created, the project and app can be created, and this contains the actual project files. Before that, make sure that the file directory is the correct one. Simply call these commands:
  + Django-admin startproject <project name>
  + Django-admin startapp <app name>

Once everything has been setup, the actual editing of the project files can commence. Before we can start editing the project files, there is one final step of setup that needs to be done. In order to do that, let us understand the project directory structure. A typical project directory would look like this:



#### Figure 20. Django Project Directory

As shown in Figure 20, the first file refers to the app directory file, and this is created when the command Django-admin startapp <app name> is called. The second file refers to directory of the project, it contains most of the files including the logic. Media folder is an optional file to be created but its main purposes is to store images or documents, basically content that can be dynamically change. Static folder on the other hand is also an optional file where is store all the static files such as CSS and JavaScript files, files that will remain constant once created. Lastly, the templates folder is the folder that contains the html files.

For this project, we will only be focusing on editing the following project files:

* Under the app folder
  + Admin.py file
  + Forms.py file
  + Models.py file
  + Urls.py file
  + Views.py file
* Under the project folder
  + Settings.py file
  + Urls.py file
* Under the static folder
  + Mainly CSS files or JavaScript files
* Under the templates folder
  + HTML files

# Admin.py under app folder

This is where the models created in models.py file are registered. (See [Appendix E](#_Appendix_E))

# Models.py under app folder

This is where the models are created. Models created are actually classes itself and each class have attributes associated with the class. Django models will be imported into this file (See [Appendix F](#_Appendix_F))

# Forms.py under app folder

This is similar to models.py except that modelforms is imported. As the name implies, this file is meant to auto populate the HTML form automatically based on the attributes defined in the class. One good thing about Django is that it allows class inheritance so repeated attributes would not have be declared and defined again. (See [Appendix G](#_Appendix_G))

# Urls.py under app folder

This is where the logic of URL routing is defined. Basically, whenever a URL is requested, it will tell the web application to look for codes in this file, locate the requested URL and look for the function in the views.py file (See [Appendix H](#_Appendix_H))

# Views.py under app folder

This is where the real logic happens. Views.py is meant to bridge the gap between templates and models. It tells the web application which set of data is to be retrieved from the database via the model and which set of retrieved data is to be displayed to the HTML side for rendering. (See [Appendix I](#_Appendix_I))

# Settings.py under project folder

This is where the initialization codes reside in. Any new app created will have to be registered here. New folders created also have to defined here. (See [Appendix J](#_Appendix_J))

# Urls.py under project folder

This is very similar to the urls.py file under the app folder, except that the URL routing codes under the app folder is meant to serve URLs with the app name extension while the URL routing codes under the project folder is meant to route the generic URLs such as the index page, admin and etc… (See [Appendix K](#_Appendix_K))

# Static Folder

This is where all the static files like CSS or JavaScript files are stored. (See [Appendix L](#_Appendix_L))

# Templates Folder

This is where the HTML files are stored. HTML files can be extended, meaning that a base html is created which contains the uniform looks of the webpage can be extended from another webpage with the help of template tagging {% %}. (See [Appendix M](#_Appendix_M))

Once all the files have been edited, it is time to run the server and display the webpage. The best way is to implement a feature, run the server to test it out, make sure everything is in order before moving on to the next feature. In order to get project files running, these commands have to be called:

The first 2 commands are responsible to migrate the changes to the database and to create the new table in the database based the model’s attributes. Remember to always to run these 2 commands whenever changes are being made to existing models.

* Python manage.py migrate
* Python manage.py makemigrations

This command is to run the server on the computer’s local host so that the webpage can be displayed.

* Python manage.py runserver

# Verification Phase

Once the implementation phase has come to an end, verification would need to be conducted to ensure that the requirements have been met. In this section I will be explaining the process of how I conducted the verification for this project.

# Test Approach

Functional testing is as an approach to test the contract management system. Functional testing is a software testing process used in which a software is tested to ensure that it conforms with all the specified requirements. It also ensures it has all the required functionality it is supposed to. In functional testing, there are 2 kinds of testing and black-box testing would be used to test that the software work as expected. It ensures that by checking that whatever is inputted should have an expected output and this will be checked against the actual output. In order to carry out black box testing, test cases are needed to ensure that there is a guide to follow, to know what needs to be tested based on the functional requirements gathered from the requirements phase.

# Test Cases

The test cases are linked back to the use cases identified in the requirements analysis section. This ensures traceability of test cases back to requirements and no requirements are left out from the test cases.

Test cases have a template to follow and will have the following fields:

* Pre-conditions
  + Conditions that need to happen for this test case can be carried out
* Dependencies
  + Preceding tasks or events that required to be completed before the starting the test case
* Use Case ID
  + To reference from the list of use cases to ensure that this test case is testing for the relevant requirement
* Test Case ID
  + To make sure that every test case can be identifiable
* Test title
  + To give an overview of what this test case is about
* Test Description
  + To provide a more detailed version of the test title to allow readers to understand what this test case is about
* Steps
  + To give a step by step guideline to follow to ensure the test results are not rigged
* Expected Result
  + What would the result look like in theory
* Actual Result
  + The actual result of the testing itself
* Status (Pass/Fail)
  + To indicate whether did this test case pass or fail.

An example of how a test case template look like would be this:

|  |  |
| --- | --- |
| Test Case ID |  |
| Use Case ID |  |
| Test Title |  |
| Test Description |  |
| Pre-conditions |  |
| Dependencies |  |
|  | |
| Test Step |  |
| Expected Result |  |
| Actual Result |  |
| Status |  |

#### Table 2. Black-box test case template

In this section, the actual test cases are displayed here to show the results of the test cases.

* TC-1

|  |  |
| --- | --- |
| Test Case ID | TC-1 |
| Use Case ID | UC-1 |
| Test Title | View number of documents of each status in dashboard |
| Test Description | Users wants to know how many documents are in each status so that he can have a good grasp of the situation |
| Pre-conditions | -NIL- |
| Dependencies | Specified requirements |
|  | |
| Test Step | 1. User launch contract management system dashboard 2. User views the number in each status box |
| Expected Result | Upon launching the dashboard, user should be able to the dashboard populated with some data |
| Actual Result | Dashboard was populated with some data upon launching the dashboard |
| Status | **Pass** |

* TC-2

|  |  |
| --- | --- |
| Test Case ID | TC-2 |
| Use Case ID | UC-2 |
| Test Title | Create new contract record and upload document |
| Test Description | User has a new contract at hand and wants to create a new record with the document |
| Pre-conditions | -NIL- |
| Dependencies | Specified requirements |
|  | |
| Test Step | 1. User launch contract management system dashboard 2. User navigates to the upload page 3. User fills up the form and uploads a document and hit upload 4. User is redirected back to the dashboard 5. User should see the value of one of the status increase |
| Expected Result | Upon submitting the form, user is redirected back to dashboard displaying the updated numbers |
| Actual Result | Upon submitting the form, user is redirected back to dashboard displaying the updated numbers |
| Status | **Pass** |

* TC-3

|  |  |
| --- | --- |
| Test Case ID | TC-3 |
| Use Case ID | UC-3 |
| Test Title | Create new contract record without uploading document |
| Test Description | User has a new contract at hand and wants to create a new record without the document |
| Pre-conditions | -NIL- |
| Dependencies | Specified requirements |
|  | |
| Test Step | 1. User launch contract management system dashboard 2. User navigates to the upload page 3. User fills up the form, leave the document upload field black and hit upload 4. User is redirected back to the dashboard 5. User should see the value of one of the status increase |
| Expected Result | Upon submitting the form, user is redirected back to dashboard displaying the updated numbers |
| Actual Result | Upon submitting the form, user is redirected back to dashboard displaying the updated numbers |
| Status | **Pass** |

* TC-4

|  |  |
| --- | --- |
| Test Case ID | TC-4 |
| Use Case ID | UC-4 |
| Test Title | Navigate through the different links in the navigation bar |
| Test Description | User wants to navigate through the webpage via the navigation panel at the top of the page |
| Pre-conditions | -NIL- |
| Dependencies | Specified requirements |
|  | |
| Test Step | 1. User launch contract management system dashboard 2. User clicks all the links in the navigation bar one by one |
| Expected Result | User should be redirected to the respective pages with the correct layout rendered |
| Actual Result | User is redirected to the respective pages with the correct layout rendered |
| Status | **Pass** |

* TC-5

|  |  |
| --- | --- |
| Test Case ID | TC-5 |
| Use Case ID | UC-5 |
| Test Title | View full list of contracts in the admin portal |
| Test Description | Administrator wants to view the full list of contracts after logging in to the admin portal |
| Pre-conditions | -NIL- |
| Dependencies | Specified requirements |
|  | |
| Test Step | 1. Administrator launch CMS dashboard 2. Administrator clicks Admin link in navigation panel 3. Administrator logs in to the admin portal 4. Administrator clicks documents 5. Administrator views the full list of contracts |
| Expected Result | Administrator should be able to see the full list of contracts |
| Actual Result | The full list of contracts is populated and displayed to the Administrator |
| Status | **Pass** |

* TC-6

|  |  |
| --- | --- |
| Test Case ID | TC-6 |
| Use Case ID | UC-6 |
| Test Title | Search for individual contract record |
| Test Description | Administrator wants to search for individual contract record after logging in to the admin portal |
| Pre-conditions | -NIL- |
| Dependencies | Specified requirements |
|  | |
| Test Step | 1. Administrator launch CMS dashboard 2. Administrator clicks Admin link in navigation panel 3. Administrator logs in to the admin portal 4. Administrator clicks documents 5. Administrator search for the individual record and views it |
| Expected Result | Administrator should be able to search for the record and view its details |
| Actual Result | Administrator was able to view the searched contract record |
| Status | **Pass** |

* TC-7

|  |  |
| --- | --- |
| Test Case ID | TC-7 |
| Use Case ID | UC-7 |
| Test Title | Perform create contract record operations |
| Test Description | Administrator wants to be able to create a new record in the admin portal |
| Pre-conditions | -NIL- |
| Dependencies | Specified requirements |
|  | |
| Test Step | 1. Administrator launch CMS dashboard 2. Administrator clicks Admin link in navigation panel 3. Administrator logs in to the admin portal 4. Administrator clicks documents 5. Administrator clicks add document 6. Administrator fills up the form and hit save |
| Expected Result | The new record should appear in the list of records |
| Actual Result | The new record appeared in the list of records after adding |
| Status | **Pass** |

* TC-8

|  |  |
| --- | --- |
| Test Case ID | TC-8 |
| Use Case ID | UC-8 |
| Test Title | Perform update contract record operations |
| Test Description | Administrator wants to be able to update an existing record in the admin portal |
| Pre-conditions | -NIL- |
| Dependencies | Specified requirements |
|  | |
| Test Step | 1. Administrator launch CMS dashboard 2. Administrator clicks Admin link in navigation panel 3. Administrator logs in to the admin portal 4. Administrator clicks documents 5. Administrator search for record to update 6. Administrator edits the fields in the form and hit save |
| Expected Result | Existing record should be populated with the new detail |
| Actual Result | Existing record was populated with the new detail |
| Status | **Pass** |

* TC-9

|  |  |
| --- | --- |
| Test Case ID | TC-9 |
| Use Case ID | UC-9 |
| Test Title | Perform delete contract record operations |
| Test Description | Administrator wants to be able to delete an existing record in the admin portal |
| Pre-conditions | -NIL- |
| Dependencies | Specified requirements |
|  | |
| Test Step | 1. Administrator launch CMS dashboard 2. Administrator clicks Admin link in navigation panel 3. Administrator logs in to the admin portal 4. Administrator clicks documents 5. Administrator search for record to delete |
| Expected Result | Deleted record should no longer exists in the list of records |
| Actual Result | Deleted record no longer exists in the list of records |
| Status | **Pass** |

* TC-10

|  |  |
| --- | --- |
| Test Case ID | TC-10 |
| Use Case ID | UC-10 |
| Test Title | Sort list of contracts |
| Test Description | Administrator wants to be able to sort the list of contracts according to the fields in the admin portal |
| Pre-conditions | -NIL- |
| Dependencies | Specified requirements |
|  | |
| Test Step | 1. Administrator launch CMS dashboard 2. Administrator clicks Admin link in navigation panel 3. Administrator logs in to the admin portal 4. Administrator clicks documents 5. Administrator clicks on a field in the list |
| Expected Result | The list of records should be sorted based on the field he clicked |
| Actual Result | The list of records was sorted based on the field he clicked |
| Status | **Pass** |

Knowledge learned in school applied in this section:

* ICT 2101 – Introduction to Software Engineering
  + The knowledge of how to create a proper black-box test cases is required here in this section

# Technical Challenges

This section describes any notable events where I face technical challenges and what steps I took to overcome the challenges.

# Technical Challenge 1 – HTML components not adopting the flex properties specified in the CSS file

# Description of challenge

During the implementation phase, when I was writing CSS codes to style and align HTML components, the components were not adopting the flex properties that was mentioned in the CSS codes. I knew that I must have missed out something in my CSS codes.

# How I solved the issue

I went back to the CSS codes to look through again. The first check was to ensure that I have declared flex properties with this line of code 🡪 display: flex; and it passed. The second check was to drill down to the CSS code of the component I was trying to give flex properties to. When using the web browser to view the webpage, I inspected that element to view what CSS it was having. That was where I found my error. I realised that the flex properties were missing so I went back to my code editor to find out what’s wrong. To my surprise, I had placed the flex properties code in another HTML component. I edited the code, recheck again and now the HTML component is adopting the flex properties I gave it.

# Conclusion

This project has taught me a lot of things. One of the valuable lessons worth mentioning was the development of self-troubleshooting skills, to try to resolve the issue as much as I can on my own before asking others for help. There certainly was a lot of time to practice and polish that skill during this IWSP period. Through this period, I found myself picking up more skills and languages to further improve my knowledge. With the new skills and languages acquired, it also helped me a lot in improving the quality of my code.

# Appendix A

# 

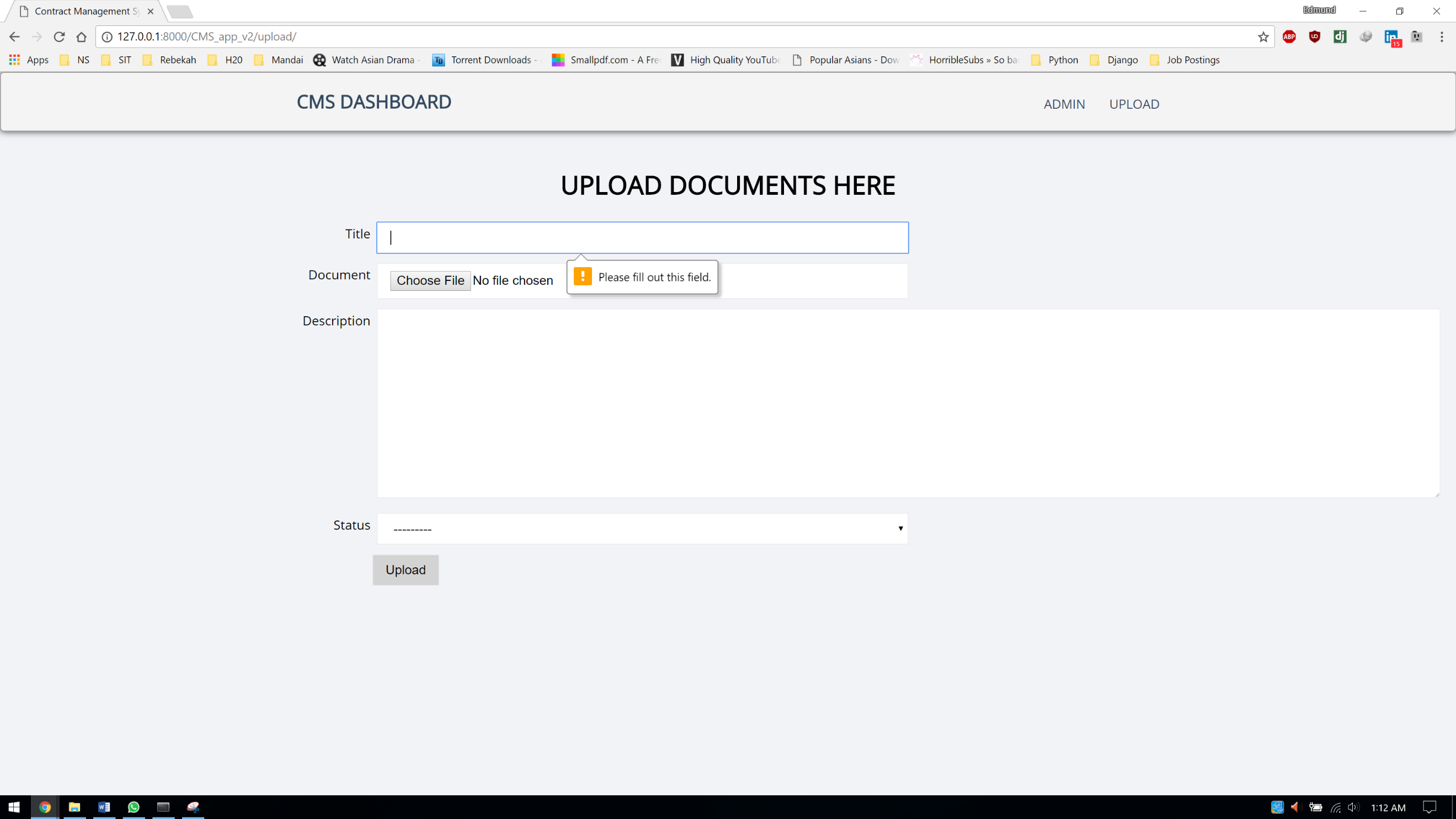
# 

# Appendix B

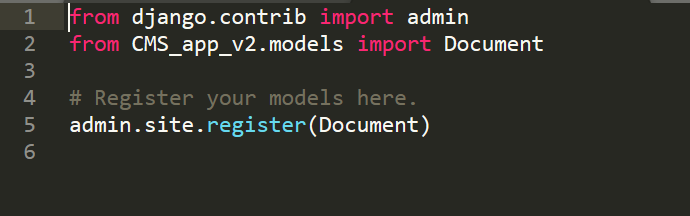
# 

# Appendix C

# Appendix D



# Appendix E

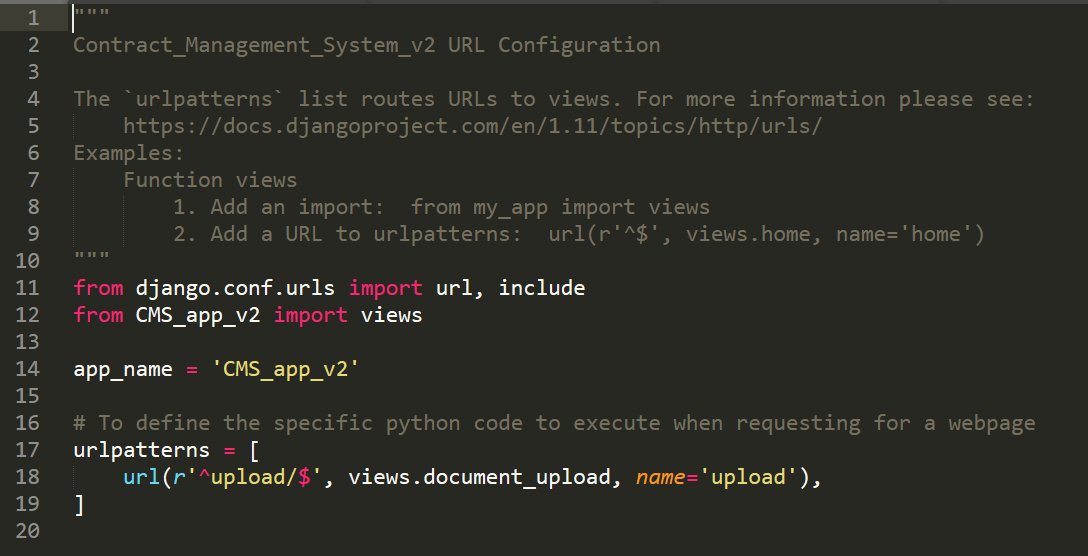


# Appendix F

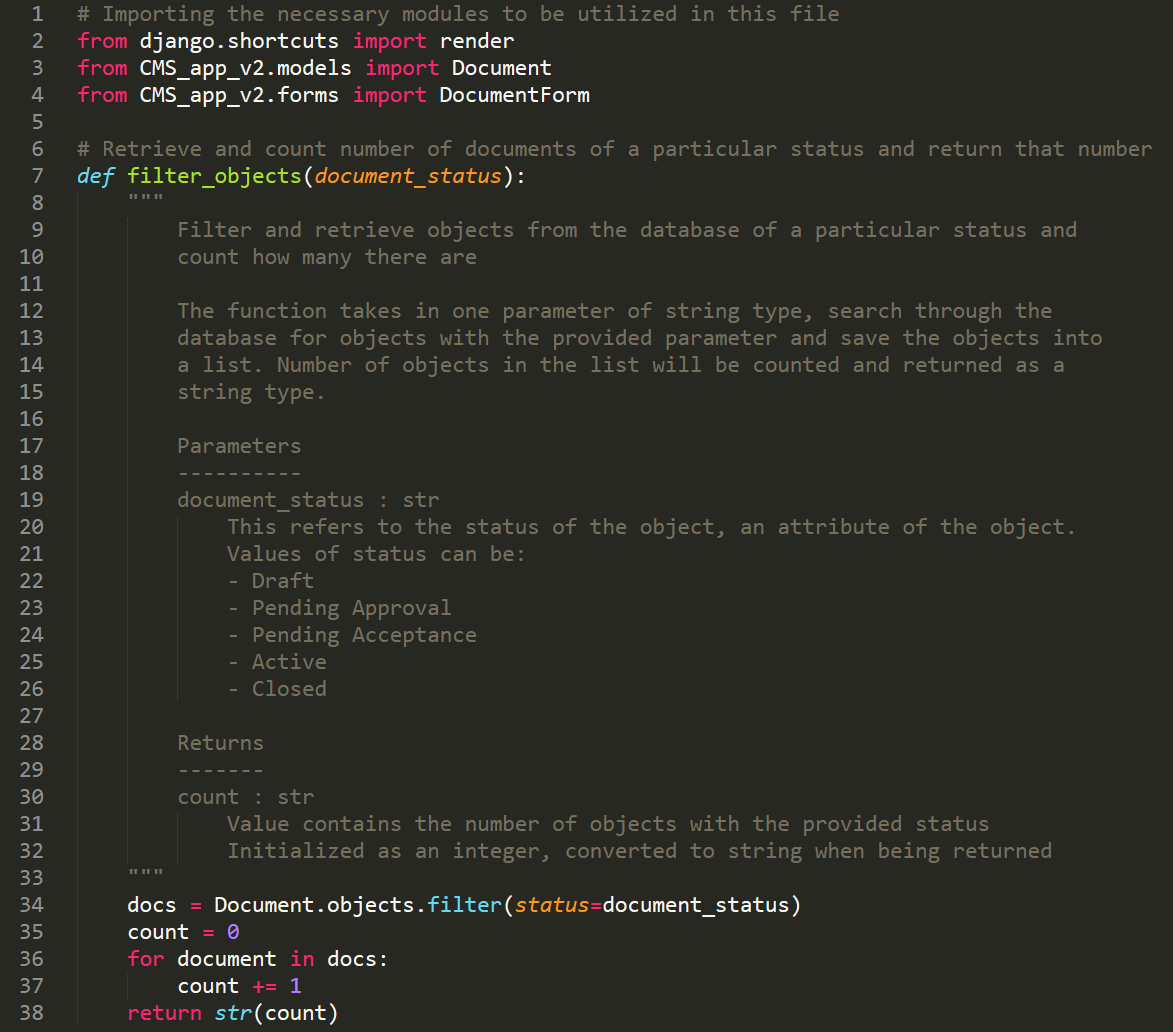
# Appendix G

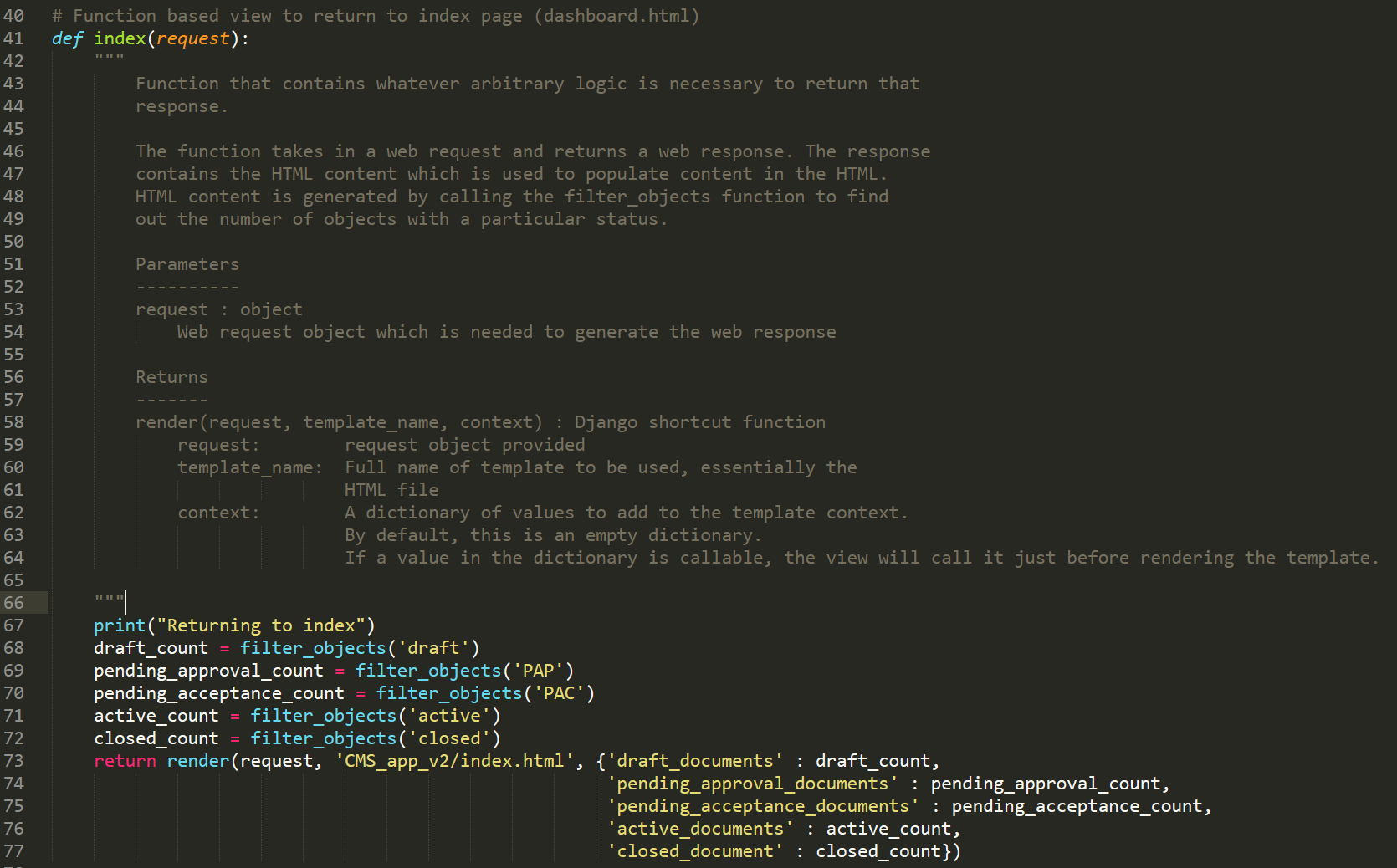


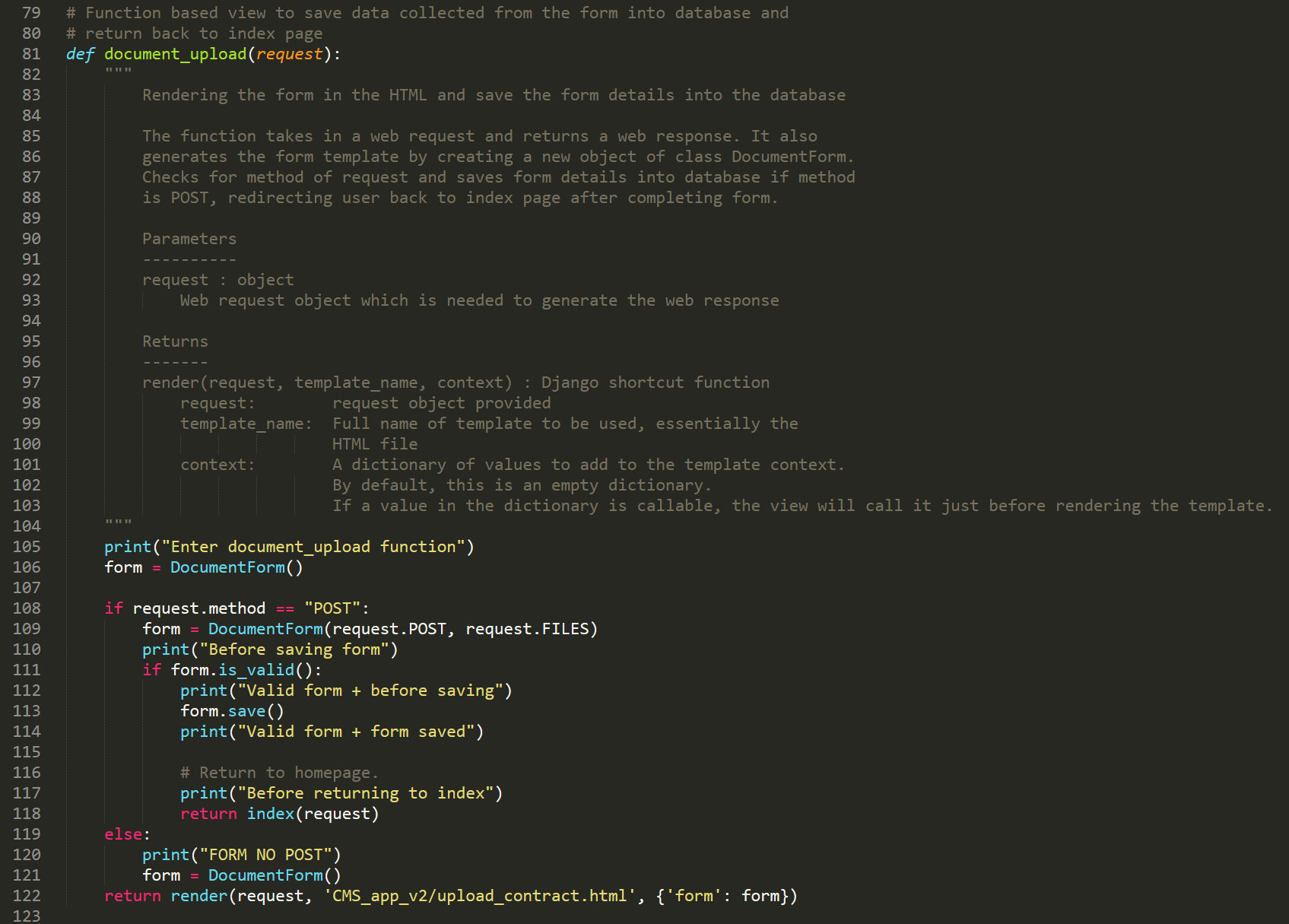
# Appendix H



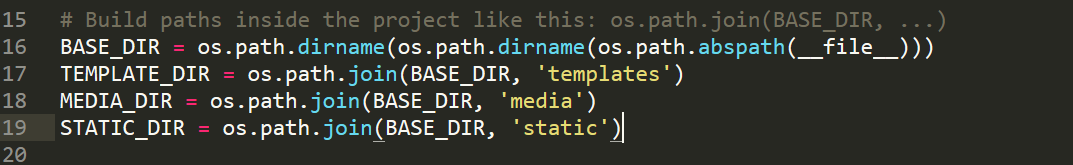
# Appendix I

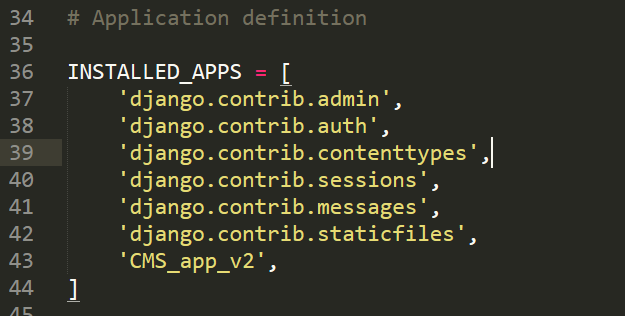


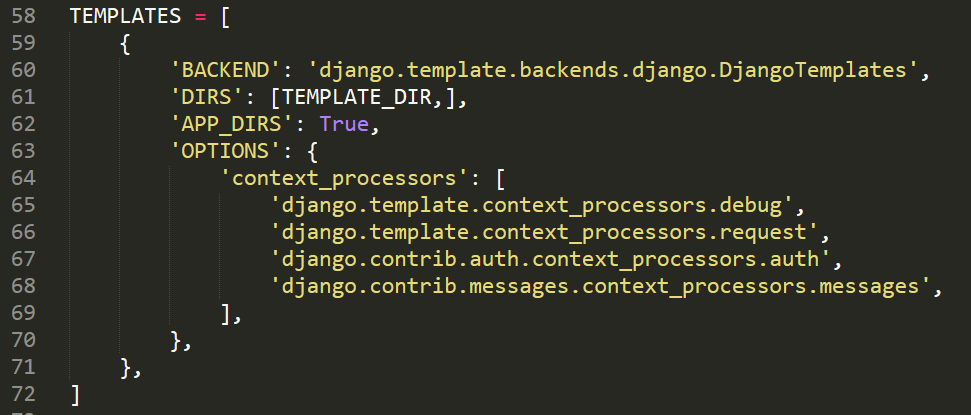




# Appendix J

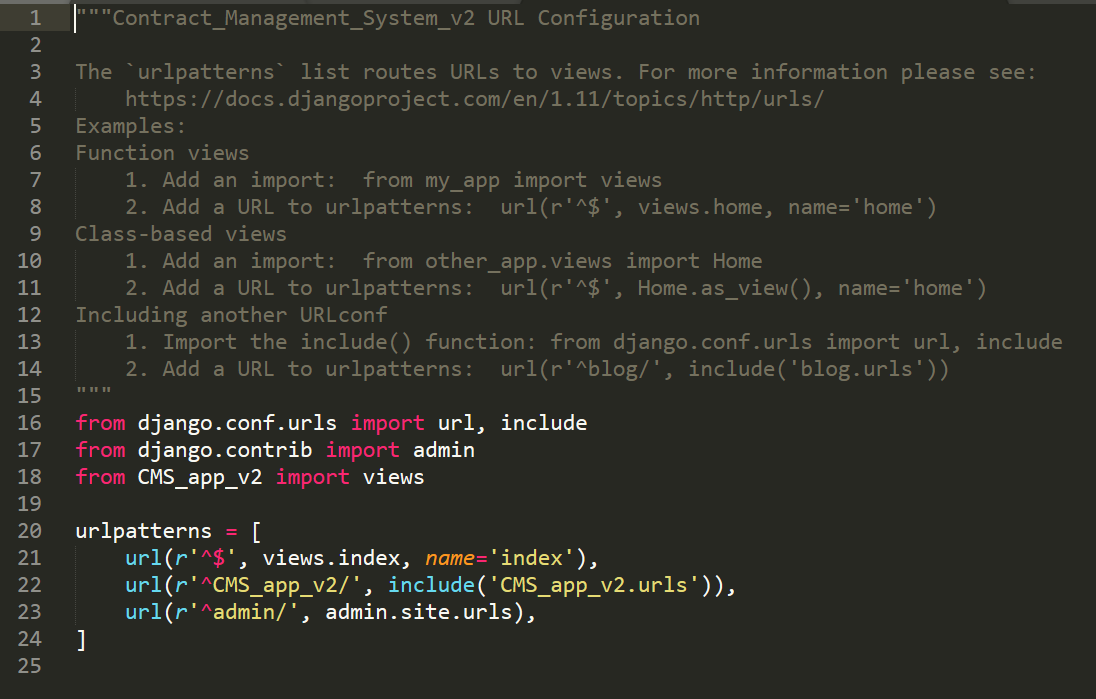




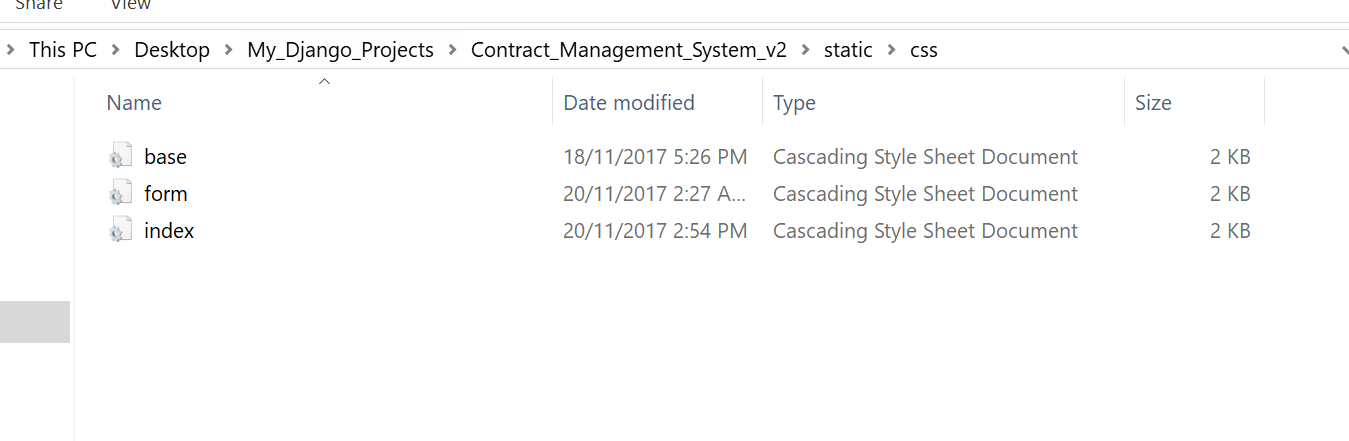




# Appendix K



# Appendix L



# Appendix M

# 