Web Application for a

Boosting Community in

World of Warcraft

Joseph John Farrelly

D1812254



Computing & IT

The Open University

14th September 2020

Table of Contents

**Introduction**

Description3

Proposed Solution4

Literature6

**Project Work**

Lifecycle Model9

Schedule9

Requirements Gathering10

Functional Requirements13

Use Cases14

Non-Functional Requirements16

LSEPI17

Non-Functional Requirements Amendments19

**Increment 1**

Database Design20

Wireframes21

Coding22

Testing26

Increment Evaluation27

**Review**

Project Management28

Resources, Skills & Risks31

Personal Development35

**References & Appendices**

References37

Appendices40

**Introduction**

**My project**

The project I decided to undertake was a website based application that would be used to manage ‘boosting’ services that I myself take part in on the game World of Warcraft (WoW). Some background on the game itself, it is a massively multiplayer online role-playing game (MMORPG) that was released in 2004 by Blizzard Entertainment, with new and exciting content being added regularly that build upon the base game. Roughly every 2 years there is a new expansion released which is essentially a brand new game except it is embedded inside the existing world. The level cap that players can reach is increased along with new continents for players to explore that contain most of the new content, which is only available to players once they reach the new level cap.

**What is there to do inside of WoW?**

The whole premise of the game revolves around your own character which you create when you first log in. You have to choose the race (which determines what your character looks like) and the class (which determines what your character can do) along with a unique name, then you are ready to enter into the world (Blizzard Entertainment, 2020). Your character has a levelling progression system that starts at level 1 and goes all the way up to the current max level of 120. You level up by completing quests for various non-player characters all over the world and fighting through dungeons and rare bosses with friends. However it’s at the maximum level where the game really opens up and you gain access to everything that the current expansion has to offer. There are so many things available to do at level cap so I will only list a few of those that are related to my project.

Each character has a set of armour made up of 14 different pieces, as well a weapon, or weapons. Each of the items in these slots has an item level which is representative of how good that piece of armour is, e.g., the higher the number the better the piece of gear. Once a player has reached the maximum level and so finished the levelling progression, they then have to try and get the best gear they can in order to make their character as strong as possible. There are a variety of sources for gear but two of the most prominent ones are from dungeons and raids. Dungeons and raids are instances that contain 5-30 unique players and require the participants to defeat a number of bosses and their minions in order to complete them. Once completed players are rewarded with armour pieces with an item level based upon the difficulty at which they completed the instance, so the more difficult the raid or dungeon is, the higher the item level of the gear is.

**Boosting in WoW**

One final thing to mention is gold, gold is the backbone of WoW. Everything revolves around gold, you use it to repair your armour and you can loot it from the corpses of bosses and mobs you defeat. It is used to purchase everything and the game has its own auction house which allows players to buy and sell pretty much anything they wish to. The topic of gold is where boosting comes in. The idea of boosting is that someone uses the gold they have accumulated on one of their characters to pay other players to ‘boost’ them through some type of content in order to receive the higher item level gear. The reasons people pay for boosts is usually because either they are not good enough at the game to complete the content themselves or they simply do not have the time to do it, and the reason other players are willing to boost is because they want to have more gold.

**My solution**

That’s a little background on what World of Warcraft is and hopefully provides a little more insight into why boosting is a thing. Now looking at my project and what it entailed. I am currently a member of a few boosting communities and the way they manage all of their bookings and accounting is through the use of a few separate google sheets, along with a discord server that has custom bots to help with bookings. Whilst I would agree that the current system did work correctly most of the time, it was not without its problems. I noticed that the primary cause of most of the problems was down to 1 simple factor, the system itself as a whole was spread out across 2 different applications that had little to no integration with each other whatsoever. This meant that data could not be automatically synced between each application and instead required manual human input in order to move data between each one. Due to the size of the boosting communities there is a large volume of data that needed to be transferred between applications every week, and this translated to several hours of human data entry that was required in order to keep the system operational. As well as the time that needed to be spent on data entry there was also a high risk of inaccurate data being entered due to human error. Whilst the frequency of errors did not appear to be very high, I believe this is because there are likely many errors that have been made that have gone completely unnoticed. The reason they have gone unnoticed again comes down to the system comprising of 2 separate applications, there was no way to cross reference data that has been moved between applications. The problems I have highlighted above I believed could be solved by moving the system to a single web based application that has the same functionality of the 2 existing applications. The end goal of my project was to have created a single web based application that would allow; boosters to create an account, boosts to be booked for buyers, boosters to sign up to take part in boosts, boosters to see an overview of their account including current balance and upcoming boosts, admin to see an overview of all user’s accounts and all boosts.

**Web based vs Mobile application**

The reason I decided to go for a web based application to be used on a desktop instead of a mobile app is because of a few reasons. Firstly, as WoW is a desktop based game and cannot be played on a mobile device, it would be more convenient for players to have access to my application on the same medium as the game. A much larger reason for staying away from a mobile app was because of the smaller screen sizes and varying resolutions that are present across different devices. As my application was going to contain large volumes of data that will need to be displayed in tables with multiple columns, being restricted to a smaller screen would have made it difficult to read and understand for the user.

**Impact**

The potential impact my application could have if it is adopted is it could make the process of booking a boost much more efficient and simplistic when compared with the old system. For example, the old system would require the user that is booking a boost to enter some of the details of the boost into the discord server and then all of the boost details into the google sheet. My application only requires the user to enter the boost details once using a simple form with verification and validation on each of the fields. This greatly reduces the risk of incorrect or inaccurate data being entered due to human error as it is caught before being saved to the system. My application also removes the workload for the admin of boosting communities as they no longer have to update a user’s permissions on multiple applications when a user is promoted/demoted. A negative impact this could have is communities may decide that they no longer need as many members in the admin team and so some users may be disappointed or angry at the adoption of my application if it will result in their demotion.

**Related Literature**

The following paragraphs will be discussing a selection of the literature resources that I have discovered during my project and found useful in helping me develop my application. As my project has progressed any additional resources I have used have been appended to this section.

After creating a set of declarative functional requirements I found that the best way to expand upon each requirement and have a clear understanding of what specifically the system will need to do, is to model the requirements as use cases. Using the book Use Case Modeling (Spence & Bittner, 2002), which covered the fundamentals of use case modelling and went into detail on how best to structure use cases and the things you should and shouldn’t include in them. I found that there was real value in taking declarative requirements and modelling them as use cases. It allowed me to describe a step by step of exactly what the system would need to do to and the events that would occur when carrying out each requirement. The book also briefly touched on modelling non-functional requirements however I didn’t feel there was sufficient information there for me to understand how best to deal with my non-functional requirements. This led me to look for additional sources that focus more on handling non-functional requirements, and from this I found the book listed below.

The book Managing Software Requirements: A Unified Approach (Widrig & Leffingwell, 1999) again covered some of the same topics as the Use Case Modeling book regarding use cases, but it also covered requirements gathering methods and had much more detail on non-functional requirements and the ways in which to model them. When it came to creating use cases for my system I found that making use of both this book and the Use Case Modeling book covered all of the aspects I needed information on in a sufficient level of detail that allowed me to display my functional and non-functional requirements in a clear and descriptive manner.

The online MySQL Reference Manual (MySQL, 2020) is a resource that I heavily referred to during the entire development of my system because it was the database I chose to use for my application. I found it to be incredibly useful when designing the database. During the database design I needed to specify the type of each attribute in all of the entities and in order to do this correctly I needed to know what data types MySQL could support. Without this resource I could have included data types in the database design that were unsupported by MySQL which would have caused me problems later down the line when it came to the implementation of my system. Also when it came to designing the structure of the database, I needed to look at the steps and models that were necessary to go from an initial design, to one that can be implemented into an application. This is where I then came across the website described below.

Visual Paradigm is a ‘leading and globally recognized provider for Business and IT Transformation software solutions’ (Visual Paradigm, 2020). Whilst it wasn’t specifically tailored towards my project I did find one of their support guides on Entity Relationship Diagrams (ERD) to be very beneficial (Visual Paradigm, 2020). Specifically it detailed what needs to be included in an ERD and also how each of the 3 different types of ERD’s differ. I found that the conceptual model needs to just be a basic diagram with the names of entities and their connections to one another, whereas the logical model includes more detail like the attributes of each entity and the multiplicities of the connections between each one. The physical model very closely resembles a blueprint for a database by specifying the data type of each attribute for all entities as well as things like primary and foreign keys. This site is what helped me get started on producing the complete design for my database.

As I decided to use the Django framework for my application the official Django documentation was a resource that just like the MySQL reference manual I needed to refer to frequently throughout development (Django, 2020). I did not have any prior experience with Django so I first needed to learn the basics of the framework before attempting to use it in my application. As well as the documentation there were also tutorials that covered all of the tasks that were needed to set up an application with Django, as well as some of the more advanced features and capabilities of the framework.

After carrying out an LSEPI analysis during development of my application, it was clear that because it would involve the processing and storage of sensitive user data within the EU I needed to make sure that it would be GDPR compliant. When approaching the topic of GDPR and a user’s personal data, there are five key questions that need to be answered. These are; why data is processed, when data is processed, the impact of processing, consent to process, and how data is processed (Netop, 2019).

The why and when data is processed are fairly self-explanatory. There must be documentation that is transparent in outlining the reasons why personal data is being processed, as well as providing proof of any policies and procedures that are being followed. Thought needs to be given to when data is being processed because an application that is constantly processing massive amounts of data will be more at risk when compared to one that processes personal data infrequently.

The impact of processing concerns the risks associated with processing personal data. There needs to be an assessment of the measures that are in place in order to reduce the risk to subjects of the personal data, to make sure they are adequate.

Consent to process data means that user’s must explicitly give consent for their personal data to be processed. When requesting consent from a user the request must be presented clearly and concisely with no ambiguity of the meaning, and it must be documented that the user has given consent for their personal data to be processed.

How data is processed covers a range of topics including data minimization, data security, data breach notification and the right to access/be forgotten. Of these topics data security is an area that I needed to research further into as the other topics simply involve notifying users in the event of a breach and responding to users’ requests to access or delete their personal data. Researching data security led me to the OWASP website below.

The OWASP Top Ten website (OWASP, 2020) provides a list of the top 10 web application security vulnerabilities. Alongside each one is a brief paragraph describing what the vulnerability is and the consequences that can arise if it is not addressed. Each vulnerability has a link to a page that goes more in depth on the issue describing exactly where it comes from, the impacts it can have on a web application, ways to address the vulnerability, and also includes some example scenarios of what can happen if they are exploited. Whilst my project did not need to address every vulnerability listed, there were a few vulnerabilities that were directly applicable to my project and this resource was invaluable in helping me address them.

The vulnerabilities listed on OWASP that applied directly to my project were injection attacks, broken authentication, sensitive data exposure and broken access control. My application is regularly communicating with a database and sending and receiving data from said database using commands and queries. An injection attack involves sending untrusted data to an interpreter as part of a command or query, which can then trick the interpreter into executing unintended commands on the database. Django uses Object Relational Mapping (ORM), which means that I myself didn’t need to write any direct SQL queries. Instead the Django QuerySets API is used which converts python to SQL queries. Django’s QuerySets are protected from SQL injection attacks because their queries are constructed using query parameterization (Django, 2020).

A user will need to create an account within my application which requires them to enter a password. Broken authentication can occur if the application allows a user to create an account using weak or well-known password such as ‘Password1’, something which can then be easily guessed by a hacker, giving them access to someone else’s account. My application uses the Django authentication system, which has built in features to protect against broken authentication (Django, 2020). It includes a built in password strength checker which; doesn’t allow generic passwords, requires at least 8 characters and cannot be entirely numeric, which helps to reduce the risk of accounts being compromised by a hacker.

As my application needs to store each user’s password, this is regarded as sensitive data which means that it falls under the privacy law, GDPR, which circles back to the resource mentioned previously. Sensitive data exposure can occur when data is sent across the internet in plain text, or when a user’s data is not encrypted and hashed before it is saved to a database. As mentioned above I decided to use the Django authentication system because it had built in features that helped to address the security risks highlighted by OWASP. In Django, passwords are not stored in plain text, they are stored using the PBKDF2 algorithm with a SHA256 hash, which is sufficient for most users (Django, 2020). It did have the option to customise the algorithm and hash used however I didn’t believe this was necessary for my application given the strength of the defaults Django provides.

My application has multiple types of users including admin roles who all have varying permissions. Broken access control can occur when pages do not correctly check if a user is authenticated or has the authorization to view said page before loading the content. This can lead to unauthenticated users being able to access the application and its features that are not public, and also unauthorized users being able to access features that should only be available to admins of the application. In my application, on the home page I added checks for 2 different things; is the user authenticated (are they signed in), and what authorization do they have (what role do they have). The results of those checks dictates the buttons a user is able to see in the menu bar. I also needed to duplicate this code on to each page as well because pages could be accessed by typing the path to the page in the address bar rather than clicking a button.

**Project Work**

**Lifecycle Model**

The model I chose to adopt for my project was an incremental one because, based on my research in Appendix 1, I believed it best suited the needs of my project. I first considered a classic waterfall model which would have been a good fit for my project because it works well on smaller projects, however it would have meant that I would’ve needed to develop the application with minimum intervention and communication from the client. A waterfall model would have also required well defined requirements that aren’t likely to be altered during development. I expected to be communicating regularly with clients that will be using my application and as a result I expected the requirements to be updated and changed frequently, thus the lack of flexibility from a waterfall model made it a bad fit for my project.

The second model I considered was a spiral model which initially looked to be much better suited to my needs. It allows for changes or added functionality to be implemented later on in the development cycle as well as allowing plenty of room for client feedback. Whilst these are both strengths that applied directly to my project, the big weakness of the spiral model is that there is a high risk of not meeting the planned schedule. This had the potential to be a major problem for myself because my module had a strict deadline for the EMA to be submitted by and the EMA requires my project to be completed.

The third model I looked at was a rapid application development (RAD) model. I didn’t believe this model would be suitable for my project as it requires highly skilled designers and developers. It also does not require any documentation to be produced to demonstrate what has been done on a project, which I felt would hinder my learning on how to properly document the development of an application.

After evaluating a few different models I decided on an incremental lifecycle model. At a basic level it is simply the classic waterfall model except it’s just repeated multiple times for each increment. The advantage of an incremental model is that it is very flexible in terms of changing the requirements and scope, as well as allowing the client to input their own feedback after each new build. The biggest drawback the incremental model has is that it requires good planning design, however I didn’t view this as a problem as it would give me a chance to showcase my skills in my final module.

**Schedule**

I created an initial Gantt graph for the schedule of my project which included all of the tasks and subtasks required for the development of my application, as well as those that were required for the module like completing TMA’s (Appendix 2). Throughout the development I revisited this schedule and made adjustments when necessary, I will cover these changes later on in the report when it comes to reviewing my project management.

**Requirements Gathering**

For the requirements gathering for my system I used a few different techniques which included; observing the old system to see how users interact and utilise it, interviewing a few people that had different roles within the communities and so all had varying experiences with the system, and then once I identified the high-level functional requirements for my application I created use cases for each one. I didn’t feel that any of the other requirements gathering techniques like workshops and prototyping were best suited for my system which is why I chose to use the alternative options instead (Inflectra, 2020).

**Observation**

I decided to examine the old system that was being used by one of the boosting communities in order to identify the functions and features my application would need to have. From the resources I found the attitude towards observation was that it can be a very useful method of requirements gathering if it is for an updated version of an existing application, rather than a completely new one (Widrig & Leffingwell, 1999). As the new application I was developing aimed to merge the features of a few different tools that were previously being used into 1 complete package, observing how the old systems operated gave me a solid foundation for the requirements of my new application.

Upon carrying out an LSPEI analysis on my project in TMA02 I had to change my approach to observing the old system as I needed to consider any tasks that would no longer be necessary if the switch was made to my application. I now needed to think about how many people within the community would potentially lose their job role as a result of my application.

The old system was a combination of a discord server with a custom bot along with some google sheets. Members who joined the discord server were given a role depending on what they are able to boost, this was assigned by an admin. Once they were given a role they were granted access to parts of the server that were restricted based on a user’s assigned roles. So for example, when you first joined the server you were only able to see a welcome message along with a price list, however when you were given a booster or a different role by an admin you could then see information only available to the role you were assigned. The same was also true with access to the google sheets. The google sheets were used to store the details of every boost, specifically the names of the buyers for each boost. They also handled all of the accounting side of things like how much each buyer was paying and what the cut was for each booster. There was no reason for buyers to have access to the google sheets, and as a result access to the sheets was only available to be requested once you had been given the booster or advertiser role in the discord server. An admin would need to manually approve the roles of boosters and advertisers however once a user had those roles they could then interact with the custom bot in order to be assigned additional roles like the type of armour they could boost etc. The same was true with access to the google sheet. Once a user had been given the role of booster or advertiser they then had to request access to the google sheets which again would require an admin to approve. Due to the sheer volume of existing boosters and advertisers and the number of applications communities received it was a relatively large part of an admins job to manually juggle roles between discord and google sheets.

To begin with I looked at a step by step process of what happened once an advertiser found a buyer for a dungeon boost. Once an advertiser had found a buyer they would create a dungeon boost in the discord server using the custom bot, which contained brief, important details about the boost like the name, level and total pot. Boosters were then able to see this new boost and sign up to it. A dungeon boost would consist of 1 tank, 1 healer, and 2 damage dealers along with the buyer. Once all 4 of those spots have been filled by boosters the bot then notified the advertiser and the boosters that the boost is full and ready to go. The advertiser is the one who sorted out inviting all of the boosters and the buyer in-game, and collected the gold from the buyer before they begin. The advertiser would then leave the group and once the boost is finished the advertiser would use the discord bot to mark the boost as completed. Alongside the discord server which was used for signing up to boosts, the google sheet also needed to be filled in by the advertiser. Here they had to input all of the details pertaining to the boost, more than on the discord server, as the google sheet is where all of the accounting work is done. The details to be included were; name of the boost, level, advertiser name, all of the boosters names, total pot, advertiser cut, booster cut. That completed the process of an advertiser providing a buyer with a boost.  
  
For a raid boost the process was slightly different. First of all, before advertisers could look for buyers, a group leader must have created a raid boost using the discord bot. Due to raids requiring a larger number of people and more coordination they were only done at set times, rather than as and when they were needed like a dungeon boost. A leader would create the boost in discord, providing important details like the name of the boost, the level, and the date and start time. Again just like a dungeon boost, boosters were then able to sign up to the raid boost using the discord bot. A raid boost required 2 tanks, 2 healers, and 6 damage dealers along with up to 5 buyers. Once all of the booster spots had been filled the boost would now be classed as full and wouldn’t allow anymore boosters to sign up. When a leader created a boost in discord they also added the boost to a google sheet which advertisers and boosters could access. Here there was a table with five empty rows ready to be filled with buyers. The name of the boost, level, and start time were also included here. When an advertiser or booster found a buyer for the boost they would add the buyer’s information to the table along with their own name, and the amount the buyer had paid. When it came to the time of the raid boost the group leader would be responsible for inviting all of the boosters who were signed up to the boost, and then invite the buyers whose information was on the google sheet. Once all buyers from the sheet were accounted for and had all paid, the boost is ready to start. Afterwards it was the group leader’s job to then enter all of the details for the boost into the google sheet in charge of all of the accounting. The details that were included here were; name of the boost, level, all of the advertiser’s names, all of the booster’s names, total pot, each advertisers cut, leader cut, booster cut. This then completed the process of a group leader creating and completing a raid boost.

From the observation I carried out I established that there would need to be 2 sets of requirements, one covering dungeon boosts and the other covering raid boosts. Both sets would have a lot of similarities however because there were slightly different needs for each one the requirements would also have to reflect that. It was also clear that there were going to be several requirements needed for tasks carried out by users with the admin role.

**Interview**

I conducted an interview with a member of the management team for one of the boosting communities (Appendix 3). The interview was done via messaging through Discord because this is what they were most comfortable using and it was the most convenient way for them to communicate with me. I had a set of questions prepared so that I had a rough guideline for the interview and to make sure that I would be able to obtain the information I needed. I went with more open ended questions to try and encourage more detailed responses in order to get as much information as possible out of the interview. I began with clarifying who it was I was interviewing and the role they had within one of the boosting communities. I also made sure to probe the interviewee for what they believed the problems and limitations of the old system were, and encouraged them to express any ideas they had to rectify those problems. Finally to end the interview I checked if they were okay with being contacted in the future so that if I needed more input later on in development I could approach them (Eriksson, 2012).

**Functional Requirements**

Using the information I gathered I then produced a list of all of the functional requirements my system would need to have. The idea behind this was to first create a list of high-level functional requirements before modelling each requirement as a use case.

Create account (all)

Log-in to an account (all)

View account (all)  
 Edit account (admin)  
 Delete account (admin)

Edit roles (admin)

View boosts services provided (all)

Book dungeon boost (advertiser/booster)

Cancel dungeon boost (advertiser/booster)

Book raid boost (advertiser/booster)

Cancel raid boost (advertiser/booster)

Book buyer on raid boost (advertiser/booster)

Remove buyer from raid boost (advertiser/booster)

View dungeon boost (advertiser/booster)

View raid boost (advertiser/booster)

Sign up to boost (booster)

Cancel sign up to boost (booster)

Amend balance (admin)

View all boosts (admin)

View all accounts (admin)

Add boost option (admin)

Remove boost option (admin)

**Use Cases**

When it came to writing the use cases for the requirements, I found that there was a great deal of flexibility with the structure and content of each one, however the most important factor is consistency within a project. The fields I chose to include in my use cases were (Spence & Bittner, 2002);

**ID** – A unique ID for each use case so that they can be referred back to and referenced easily in future.

**Name** – A few words that very briefly say what requirement the use case is covering.

**Description** – A detailed description of the requirement which explains specifically what the system will be required to do to satisfy the requirement in said use case.

**Actor(s)** – A list of the actor(s) that are directly involved in the use case.

**Trigger** – The event or action that initiates the use case.

**Preconditions** – Describes conditions that should be true when the use case is triggered.

**Postconditions** – Describes conditions that will be true if the operation in the use case is correct and the preconditions were met.

**Main Success Scenario** – A step by step walkthrough of the use case when it is successful. The first step is the trigger, and each following step describes an action that the user or the system takes in response to the previous step, in order to complete the use case.

**Extensions** – Describes any exceptions or alternative routes that can happen when going through the main success scenario. This will happen when either something goes wrong, or if there is an alternative success scenario.

**Priority** – States the priority of each use case in the development of the system. Will be low, medium or high.

Below are just a few examples of the use cases I created to model my requirements, with the full set of use cases located in the appendices section (Appendix 4).

|  |  |
| --- | --- |
| **ID** | UC01 |
| **Name** | Create account |
| **Description** | A new user accessing the system for the first time can create their own account by providing their email and a password. |
| **Actor(s)** | User |
| **Trigger** | User accesses the web application using a browser and clicks Create Account button |
| **Preconditions** | User does not already have an account on the system with the email address provided. |
| **Postconditions** | An account is created for the user and they are notified of their account being created successfully or the details entered by the user were invalid and they are notified that their account creation was unsuccessful |
| **Main Success Scenario** | 1. User accesses website  2. User clicks Create Account button  3. User inputs a valid email address that is not already registered to an account  4. User inputs a valid password  5. User clicks Complete Sign Up button  6. New User account is created  7. User is notified that their account was created successfully |
| **Extensions** | 3a. User inputs an invalid email address  3b. User inputs an email address that is already registered to an account  4c. User inputs an invalid password 6abc. User account is not created  7abc. User is notified that there is an error with the provided details and asked to enter them again |
| **Priority** | High – First Increment |

|  |  |
| --- | --- |
| **ID** | UC02 |
| **Name** | Log in to existing account |
| **Description** | A returning user is able to log in to their account |
| **Actor(s)** | User |
| **Trigger** | User accesses the web application using a browser and clicks the Log In button |
| **Preconditions** | The user has previously created an account successfully. |
| **Postconditions** | The user is logged into their account and able to access the system or the user entered incorrect login credentials and is not granted access to their account |
| **Main Success Scenario** | 1. User accesses website  2. User inputs their email address  3. User inputs their password  4. User clicks Sign In button  5. User is logged into their account |
| **Extensions** | 2a. User inputs an incorrect email address  3b. User inputs an incorrect password  5ab. User is notified that there is an error with the log in details and is not logged into their account |
| **Priority** | High – First Increment |

**Non-Functional Requirements**

For the non-functional requirements, I used the book Managing Software Requirements: A Unified Approach (Widrig & Leffingwell, 1999), which went into a lot more detail on requirements as a whole as well as how best to model any non-functional requirements. Using this I decided not to model them as use cases and instead just describe them as declarative requirements. Some of those declarative requirements I was then able to link to 1 or more use cases, but others were not applicable to any, which is why I chose to present them in this way. The non-functional requirements I declared all fell into 1 of 4 categories; usability, reliability, performance and supportability. The ID row is the unique ID for each non-functional requirement so I could quickly and clearly reference each one later on in the project. The requirement row is a brief summary of what the non-functional requirement for the system is. The description row is a more descriptive and detailed version of the requirement along with any additional information if necessary. The use case link row lists the ID of any use cases that are directly linked to the non-functional requirement identified.

Below are a few of the non-functional requirements for my system that fall under the usability category. A full list of all the non-functional requirements is located in the appendices section (Appendix 5).

**Usability**

|  |  |
| --- | --- |
| **ID** | NFR01 |
| **Requirement** | Users of the existing system should require no additional training to utilise the new system effectively |
| **Description** | The new system will have many of the same features as the current system as well as some new ones. However the new system and features should feel familiar to the user and navigation through the system should be self-explanatory |
| **Use case(s) link** | N/A |

|  |  |
| --- | --- |
| **ID** | NFR02 |
| **Requirement** | An advertiser should be able to complete a booking and receive confirmation the booking was successful in less than 60 seconds |
| **Description** | From the time when an advertiser logs in to add a booking to the new system, within 60 seconds they should be able to complete the booking for the buyer and provide them confirmation that they’re booking was successful. Although the speed of the database may be a factor in this requirement, it will have negligible impact on the speed at which the advertiser can complete the task when compared against the impact that the layout of the booking page and the navigation through the system will have. |
| **Use case(s) link** | UC05, UC06 |

**Legal, Social, Ethical and Professional Issues (LSEPI)**

During my project, as part of the TMA02 assignment I did an LSEPI analysis on my project, the result of which you will find below.

**Legal**

As my project involves the storing and processing of sensitive data, and it is being developed for use within the EU, it must comply with the GDPR set up by the European Parliament. As a result of this I will need to create additional non-functional requirements regarding the security of my system and privacy of users’ data. The impact of not making my system GDPR compliant would result in a large fine of between 10 and 20 million euros. To address this issue I will be using the UK government website which contains advice for organisations on being complaint with the GDPR, as well as using tools provided by Django to ensure that user’s data is kept private and secure. Django has its own authentication system used to manage accounts, which has a password strength checker and uses the PBKDF2 algorithm with a SHA256 hash to store passwords. Django also uses an ORM that interacts with their QuerySet API that is used to handle all SQL queries, helping to prevent SQL injection attacks.

There are also legal issues surrounding accessibility when it comes to web applications. The Web Content Accessibility Guidelines (WCAG 2.1) are an internationally recognised set of recommendations for improving web accessibility (gov.uk, 2018). Websites representing bodies in the public sector must meet the requirements set out in the WCAG, however my project does not fall under the public sector category. This doesn’t mean that I should ignore the accessibility of my website as there are likely to be people who will use my application who do have impairments.

Another legal issue pertaining to my project is that Blizzard Entertainment, who own World of Warcraft, do not allow any real-world monetary transactions as stated in their Termination of Service Agreement (Blizzard Entertainment, 2016). This means that anyone found to be selling boosts for real money is liable to have their account terminated. This isn’t an issue that I will need to address with my system as it’s something I do not have any control over, however it is still worth mentioning.

**Social**

The success of my system may have an impact on the society surrounding World of Warcraft. For example, if the system is widely regarded as a huge improvement on the previous system then it may cause other boosting communities to request the use of a system like mine. Outside of World of Warcraft there are many other games that utilise the same boosting system and so they may also become interested in my system should it prove to be better than the ones they are currently using. This would not necessarily be a bad thing as people will just like to use a system that is best suited for them, however I would need to re-consult with any new clients regarding the system as it has some features that will only be applicable to a World of Warcraft boosting system.

**Ethical**

The ethical issues once again overlap with the legal issues in that I must ensure that I am only storing a user’s details if they are required by the system in order for the user to utilise its full functionality. Additionally because my new system is set to replace an existing system in which the biggest problem is the need for manual transfer of data between two different applications, there may be people whose job within the community is solely to carry out this data transfer. My system intends to make this process autonomous so I need to consider the effect it will have on users’ roles within the system to make sure I am not harming or replacing anyone.

Users of the new system will also need to be informed at the sign up screen that they consent to their email and data relating to their in-game character to be stored and processed within the system. To do this I will need to add a check box to the sign up box that allows the user to read information about the rights of their data and how it will be stored and processed. Once they click the tick box and sign up they will have given their consent to the terms that were outlined.

**Professional**

I will need to make sure that I use appropriate techniques for gathering requirements, and not use methods only because they are the most convenient to do so. For example, I could put out a survey on the World of Warcraft subreddit regarding what they would like to see in a boosting system, this would give me a large amount of data taken from multiple peoples viewpoints, however only a small fraction of the users of the subreddit will actually be a part of the boosting communities and actually use my system. Instead I will be better off interviewing members of the boosting communities who perform a variety of different roles. This way I will get a detailed understanding of the ways in which different people interact with the existing system, and better meet their needs in my new system.

As my application will be essential to the day-to-day running of boosting communities it will need to be robust. Any issues with the application will have the potential to cause the communities to lose out on customers due to not being able to utilise the system properly. I will need to consider what will happen if the application does go offline and be able to show the client the measures I have put in place to negate the impact it would have on their community. These measures could be things like; making regular backups with the ability to redeploy a backup hosted at an alternative location should the original location become unavailable.

**Non-functional requirement changes**

As a result of my LSEPI analysis I discovered that there were additional non-functional requirements that I hadn’t considered for my project, which fell under the categories of security and accessibility. After identifying these new non-functional requirements I decided it was best to model them the same way as before which is what I have included below.

**Security**

|  |  |
| --- | --- |
| **ID** | NFR12 |
| **Requirement** | Users private data will remain confidential and secure |
| **Description** | The users email address will be stored on the systems database and will only be used within the bounds of the system. It will be stored securely and accessed securely with the necessary protocols being followed in order to protect the data against unauthorised access. It will not be sold or sent to any third parties or used for any tasks outside of those listed in the users’ agreement when they sign up. The user has control over their data and has the right to request removal of their data from the system at any time. |
| **Use case(s) link** | N/A |

**Accessibility**

|  |  |
| --- | --- |
| **ID** | NFR13 |
| **Requirement** | Options for varying text sizes |
| **Description** | There will be a box on every screen that allows the user to select the size of the text. The options will be small, medium, and large. This means that for any users that have visual impairments and so require larger text in order to read properly will still be able to utilise the application without any problems. |
| **Use case(s) link** | N/A |

|  |  |
| --- | --- |
| **ID** | NFR14 |
| **Requirement** | Colour scheme aids visual impairments |
| **Description** | The colour scheme for the application should make use of colours that contrast each other and so make things like text stand out rather than blend in with other aspects of the app like the background colour. |
| **Use case(s) link** | N/A |

**Increment 1**

**Database Design**

When designing a database, there are two approaches that can be taken, either a top-down or bottom-up approach. A top-down approach begins with a general idea of what is needed for the system and works towards the specifics of what data is to be stored in the database. Bottom-up instead begins with the specific details of what is going to be stored in a database and then works backwards from that to arrive at the general overview of a databases design (Burleson, n.d.). A top-down approach better suits my project because it will allow me to elicit the requirements for my application first, without worrying about the implementation of those requirements in the database. Instead I can go from the elicited requirements to the blueprint for my database using entity relationship diagrams (ERD).

The first step was to produce a conceptual entity relationship diagram (Appendix 6) after completing the requirements gathering process. A conceptual ERD is an overview of the structure of a system that only includes the entities that will exist in the system along with the relationship between each one. This gave me a general idea of what the final database would look like in my system without needing to think about the specifics.

From this conceptual model, together with all of the information I had gathered, I extended it in order to produce a logical model (Appendix 7). This is a much more detailed version of the conceptual model which includes the attributes for each entity, as well as adding in new entities where necessary. It also includes the multiplicities of the relationships between entities as well as identifying any entities that need to be abstract (Visual Paradigm, 2020).

As there was no requirement for which database management system (DBMS) to use I was free to pick whichever one I felt was best. I chose to use MySQL because I already had some experience using it during my level 3 modules, and it is also one of the most popular DBMS used today. After I had chosen which DBMS to use for my system, when creating the physical entity relationship diagram I needed to make sure that I only included data types that are supported in MySQL. To help with that I will referred to the official documentation on the MySQL website which details the support data types in their latest release (MySQL, 2020).

In order to convert the logical model I created previously into a blueprint for the database I needed to specify the type of each attribute, and identify the primary and foreign keys for all entities.

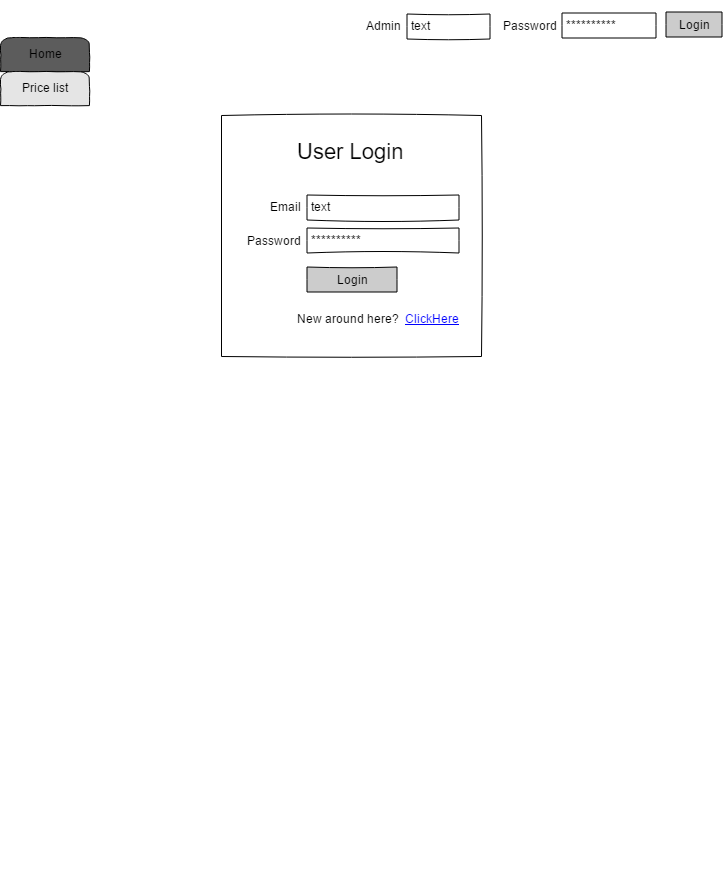
In order to create and display my database design I used SqlDBM which is a database diagram design tool that supports MySQL (Appendix 8). It is used and trusted by companies all around the word like Ford and Lenovo to name just a few (SqlDBM, 2020). It is a paid service but it did have a free option that has limitations on some features but none of those features were required for my system. For previous entity relationship diagrams I used UMLet which I already had installed on my computer from a previous module, and whilst this was suitable for simpler diagrams it did not have the features required to map out a comprehensive database design like SqlDBM.

**Wireframes**

Before jumping straight into the coding of my application I decided to first create wireframes for each screen. Wireframes allowed me to easily see where static information like navigation bars and headings would be located across all pages (usabilty.gov, 2020). It also gave me an idea of how each page would link to the rest of the application and allowed me to role play as a new user navigating through the application. This gave me an understanding of how a new user would experience the application and highlighted some obvious layout design choices that would have caused the user to have a somewhat negative experience.

To create the wireframes I used a piece of software called Pencil. Pencil is an open source GUI prototyping tool that can be used to create screen mock-ups of a web application (Pencil, 2019). I had previously used this software in TM352 and TM356 and the skills that I used in those modules were directly transferable to my work on this project.

Below is an example of the wireframe I created for the login screen. The rest of the wireframes are located in the appendices section (Appendix 9).



**Coding**

**Programming Language**

I chose to write my application in Python because it was a language that I had been briefly exposed to in previous modules which meant I already knew the basics and was able to focus on the more advanced capabilities of Python instead. A Python developer is also the type of job role I would like to pursue after completing my studies so creating my project using Python made the most logical sense.

**Framework**

When looking for a Python web framework to use I discovered that there are 3 different types of frameworks to choose from. They are full-stack frameworks, microframeworks, and asynchronous frameworks (Sanders, 2020).

|  |  |
| --- | --- |
| **Full-Stack Frameworks** | - Generally focused on building larger, full-featured applications.  - Offers a lot of common functionality out of the box.  - Provides defaults for common aspects of larger applications like communicating with databases and templating views. |
| **Microframeworks** | - Good choice for smaller, narrower focused applications.  - Provides a small core of functionality  - Allows for more control over application design by inviting the developer to make their own choices regarding the libraries to use for added functionalities. |
| **Asynchronous Frameworks** | - Focused on high levels of performance by allowing a large number of concurrent connections.  - Require a more rigorous programming style with a limited set of plugins. |

For my project I decided to go with a full-stack framework. This was because my project had strict time constraints to adhere to, specifically that it needed to be completed before the end of the module. Whilst going with a microframework instead would’ve given me more freedom over the application design, in the interest of time I believed it made more sense to allow a full-stack framework to make that choice for me so that I could focus on the features of the application instead.

Once I settled on the type of framework I would be using, along with the language it would be written in, there seemed to be one framework that was recommended over any of its counterparts. That framework was Django. Django is a ‘high-level Python Web framework that encourages rapid development and clean, pragmatic design’ (Django, 2020). Of the features boasted by Django the ones that appealed to me the most were all those related to application security. Specifically, it has its own user authentication system for managing user accounts and passwords as well as tools like their ORM that help to prevent things like SQL injection attacks.

**Integrated Development Environment (IDE) & Database Hosting**

To begin with I first created a new Django application inside the Visual Studio IDE and followed along with a tutorial taken from the Django official documentation page (Django, 2020). This involved creating a new Django project and connecting it to my own MySQL database that I was hosting using the Open Source tool XAMPP (XAMPP, 2020). Once I had the initial file structure sorted for my project I then created a skeleton HTML page for each screen using the wireframes I had made. Using Django I created a base template HTML file called ‘layout’ which was used as a foundation for all of the other pages in the application (Appendix 10). Each subsequent HTML file used for each individual screen simply extended the ‘layout’ file. This removed the need for me to use a lot of duplicated code for things that were consistent across all pages like the navigation bar and title. After creating the HTML pages, I then needed to link them to each other which is done through the use of two files in Django. The file ‘views.py’ is used to define functions that take Web requests and return Web responses, and the file ‘urls.py’ is used to map url paths to each of the functions defined in ‘views.py’, and give them a name that can be referenced by the href attribute of an HTML ‘<a>’ tag. The functions in ‘views.py’ then return each of the pages in the application along with any additional data that is needed by each page, for example; forms for the pages that create new records in the database, and data objects for the pages that retrieve records from the database (Appendix 11).

**Styling**

For the styling of my application I used Bootstrap4 (Bootstrap, 2020) for the general design and layout of each page along with Django-crispy-forms for the styling of each form (Araujo & Greenfeld, 2020). I also used JavaScript and JQuery in the frontend of my application.

Once I had correctly linked the pages together and added some styling to them I then looked at the use cases I had assigned to the first increment one at a time and implemented each of them. I will include screenshots of the code for some of the use cases however in order not clutter the report with code I will leave most of the code in the appendices section.

**UC01 – Create account (Appendix 12)**

In order for a user to create an account they need to enter their details and submit them through the application so that they can be saved into the database. As their username will be the name and realm of their in game character, the username field must be unique. I used the Django User model rather than creating a custom one, this was partly because the existing User Model contained all of the fields I needed but the biggest reason for using the Django User Model, was because it provided an authentication system that handles both authentication and authorization (Django, 2020). This meant that I did not need to implement those features from scratch and could instead build on top of the existing system that already handles password hashing and user permissions by default.

**UC02 – Log In (Appendix 13)**

For the Log In, a user needs to enter their username along with their password and if they match an existing account then the user is granted access to that account. As mentioned above I used the Django User Model which uses its own authentication system that makes the task of a user logging in trivial. Upon submitting the login request to the database the entered username and password is checked against stored usernames and hashed passwords and if there’s a match they will be logged in successfully.

Security was the biggest concern when implementing use cases UC01 and UC02. Drawing on some of the risks identified earlier on in my report from the OWASP website, broken authentication and data security both needed to be addressed when implementing user log in. The Django User Authentication system addressed both of these risks with little to no intervention needed from myself. It prevented the User from creating an account using an insecure password by providing form validation checks on the create account page when an attempt to submit the form is made. If the user’s password was less than 8 characters long, entirely numeric, or was a generic password then the form validation would fail and the account would not be created.

For the storage of passwords, Django does not store raw passwords. Instead it uses the PBKDF2 algorithm with a SHA256 hash on a user’s password before saving it to the database. This meant that on the admin side of my application it would not be possible to see a user’s password as all the admin would be able to see is the hashed value.

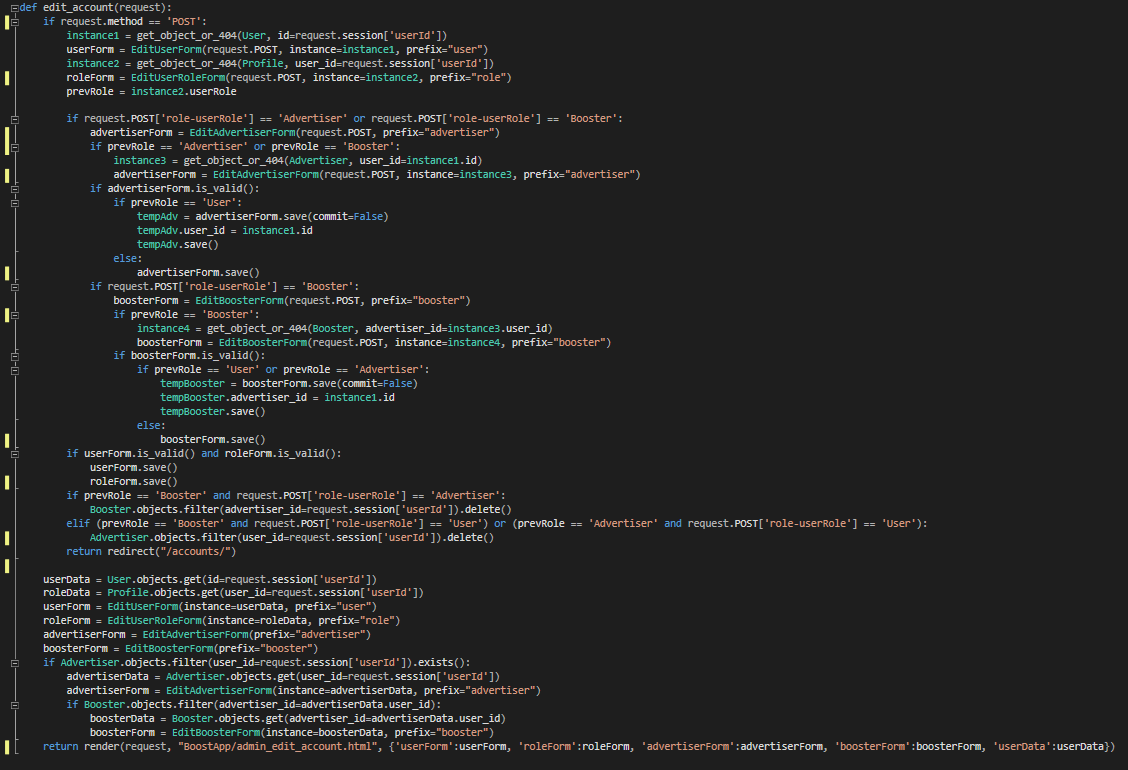
**UC04 – View boost services provided (Appendix 14)**

This page is simply there to display details pertaining to all of the boosts provided. A request is made to the database to retrieve all of the information stored in the Boosts table and the result from the database query is sent with the HTML file when the user requests it.

**UC15 – Edit account (Appendix 15)**

Editing a user’s account can only be carried out by an admin, so the page needs to check that the user is both authenticated and authorized to be viewing it. The admin can select any user’s account and is shown input fields contained inside a form that are already filled in with the user’s existing information. Once any changes have been made the admin can submit the form and the records that match the selected user will be updated where necessary to reflect the changes the admin has made.

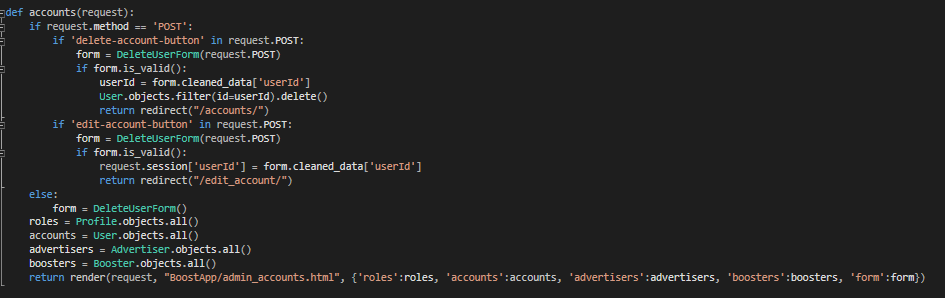
Below is the code for the function ‘edit\_account’ inside the file ‘views.py’ which is responsible for all of the functionality of the edit account page.



**UC16 – Delete account (Appendix 16)**

Deleting an account can also only be done by an admin, so again the page needs to check that user is both authenticated and authorized to be viewing it. The admin can select any user’s account from a table containing all accounts. When they click the ‘Delete’ button a request is sent to the database telling it to delete the selected user account from the User table.

Below is the code for the function ‘accounts’ inside the file ‘views.py’ which is responsible for deleting an account and passing the details of an account to the edit account page.



**UC21 – Add boost option (Appendix 17)**

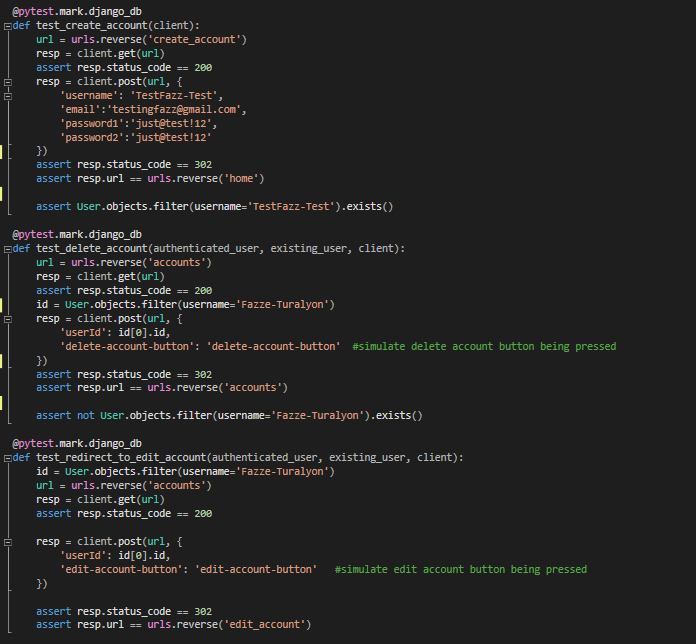
Adding a boost option can only be done by an admin, so the page needs to check that the user is both authenticated and authorized to be viewing it. This page requires the admin to input details of the new boost option and then save the new option to the database. For this I created a new Django Model called Boost, and then created a ModelForm that uses the Boost Model in order to create a form to be displayed on the page. This form is then submitted to the database and the data contained within the form is used to create a new record inside the Boost table.

**UC22 – Remove boost option (Appendix 18)**

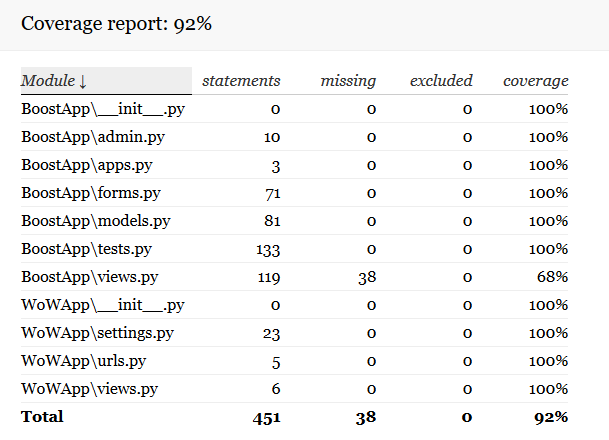
Removing a boost option can also only be done by an admin, so the page needs to check that the user is both authenticated and authorized to be viewing it. This page only needs the admin to select the boost option they wish to delete from a table containing all of the boost options that exist in the Boost table. After they have selected a boost they can click the ‘Delete’ button and a request is sent to the database telling it to delete the selected boost from the Boost table.

**Testing**

For testing my application, I could either use the built in testing suite provided by Django on its own or install some third-party packages that integrate with the Django testing suite to enhance its features. I decided to install the plugin pytest-django which brings the tools offered by pytest to Django. The reason for choosing this plugin for testing rather than using the Django built-in suite is because there is less boilerplate. By that I mean that with pyest-django I was able to simply write the tests as regular functions in the test.py file, instead of needing to first import unittest, and then create a subclass with methods for each test (pytest-django, 2020). An example of some of those tests is below and I have added the rest of the tests to the appendices section (Appendix 19).



In order to know what to test I used a tool called Coverage, which is a tool used for measuring code coverage of Python programs (Coverage.py, 2020). I was able to run tests using pytest-django through coverage in order to see which parts of my code are being exercised by tests and those that are not. Being able to see exactly what parts of my code were not being tested made it easy to know what tests I needed to add next in the test.py file. The results of running coverage on my application at the end of the first increment are shown below. The missing 8% not currently covered is the code for the edit account use case, and the reason for not writing those tests yet I will cover in the review of my project work.



**Increment Evaluation**

The first increment was a success in terms of the features implemented into the application however it was a fail when compared to the schedule. The increment took longer than planned and the reasons for this I will cover in my review, however all of the functional requirements and associated use cases that I assigned to be implemented in the first increment have all been done. The application is in a working state and the client would be able to test out the application to get a feel for how the final product would turn out. The navigation side of the app is completely fleshed out and so the client can see how a user would navigate across all pages in the system, some pages however are just placeholders as they are features that are to be implemented in later increments.

**Review**

**Current Stage of Project Work**

I feel that the quality of my project work has remained at a high standard throughout the entire module, however my ability to keep up to date with the schedule I had created was very poor. I believe that my biggest downfall of this entire process was consistently underestimating the time that tasks would take to be completed. When first creating the project schedule I assigned a similar amount of time to each task in each increment when in reality the first increment has taken almost double the allocated time for each task. Upon reflection it makes sense to me that the first increment would need more time because many of the tasks required new skills to be learned in order to complete them, for example; I had no previous experience using the Django framework, or using the testing suites. This is something that if I were to do the project over again, I would assess in depth how long it would take for me to learn to use any new tools that I am not familiar with, and include that in the schedule.

One thing I did do well was adapting to the impact that Covid-19 had on my project. Due to the pandemic putting the country on lockdown, my family members were working from home for the majority of my project. This meant that when I would usually do most of my studying during the day with no interruptions because no one else is home, I instead had multiple people home all throughout the day. The change I made is that I allocated a few hours very late in the evenings to do my project work rather than during the day. At this time everyone else had gone to bed and I was free from interruptions. I noticed almost immediately that the quality and quantity of the work I was producing in the evening compared to during the day was very significant.

**Project Management**

I do believe that choosing an incremental life cycle model did benefit me, in that it required me to break down the project into smaller subtasks which made it easy for me to see what tasks must be completed before new ones could start. Being made to think in detail about the different increments and the work that will be assigned to each one gave me a clear view on the bigger picture right from the beginning of the project. It allowed me to create a detailed schedule that I have been able to check in with to compare my progress against.

**Scheduling**

My schedule had 4 iterations. The first of which I referred to early on in the report. This was my initial schedule that had more general tasks as I hadn’t yet elicited the requirements for the application and so wasn’t able to break tasks down into smaller more specific tasks yet (Appendix 2).

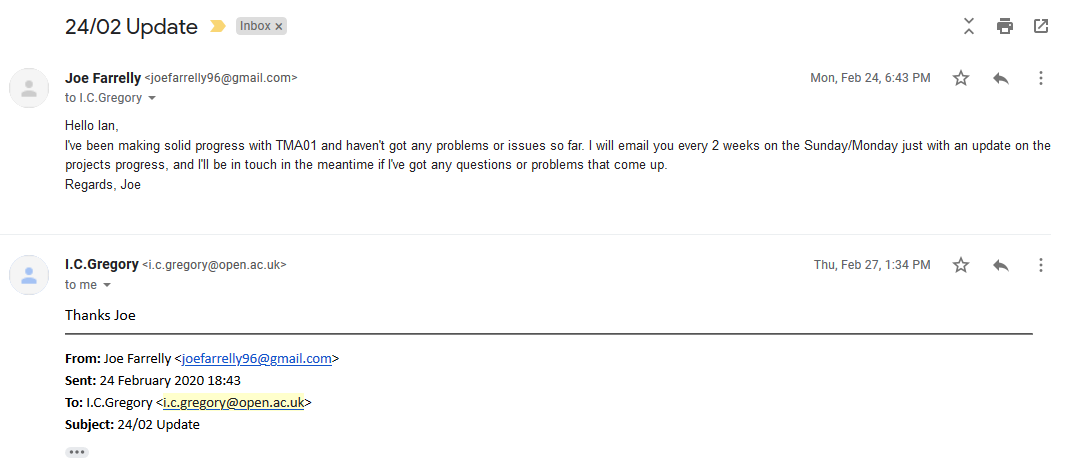
The second iteration of my schedule included smaller sub tasks which I could only include in the schedule after I had completed the requirements for my application. This gave me a more detailed view on the work I would be doing over the next few months and the deadlines I had set myself for each of the smaller tasks (Appendix 20).

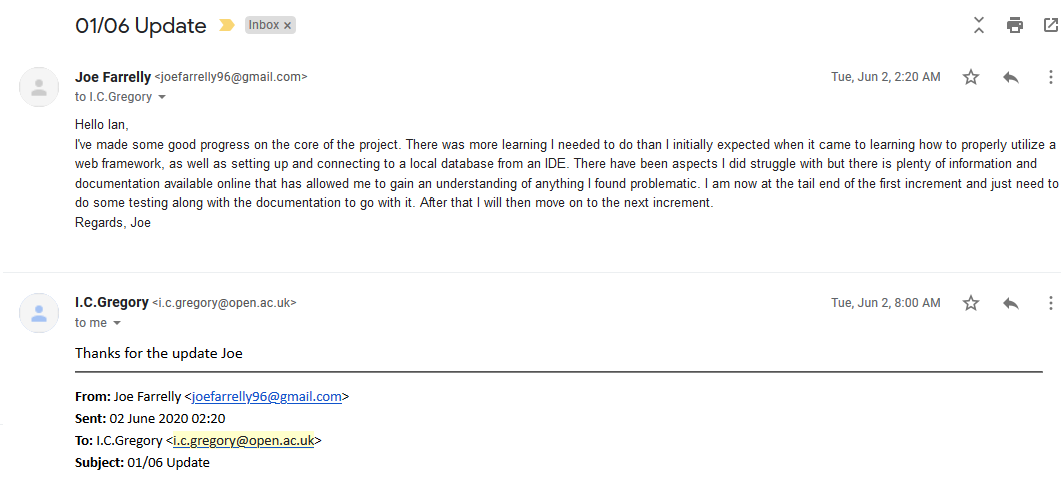
The third iteration of my schedule came after I had noticed that the tasks in increment 1 were all taking longer than I had planned, and in some cases they were taking double the allocated time. An example being the skeleton for each screen. I adjusted the schedule to reflect the increased time taken on tasks in increment 1 however I didn’t adjust the allocated time for increment 2 or 3. The reason for this is that I put the delay on increment 1 tasks mostly down to the fact that I was using new technologies and software and so previously had not taken that into account when drawing up the schedule. I made the assumption that whilst I was incorrect in my estimation for increment 1 tasks, my estimates for increments 2 and 3 were accurate as I would not need to learn any new skills to complete those tasks (Appendix 21).

The final iteration of my schedule came after TMA03 when I realised that I was too far behind my schedule that it was not feasible for me to complete all of the tasks. As a result of this I made the decision to remove the features of the application that were to be implemented in increment 3 and so remove increment 3 in its entirety. This left me with a much more attainable goal of completing increment 2 before the end of the module (Appendix 22).

**Tutor Contact**

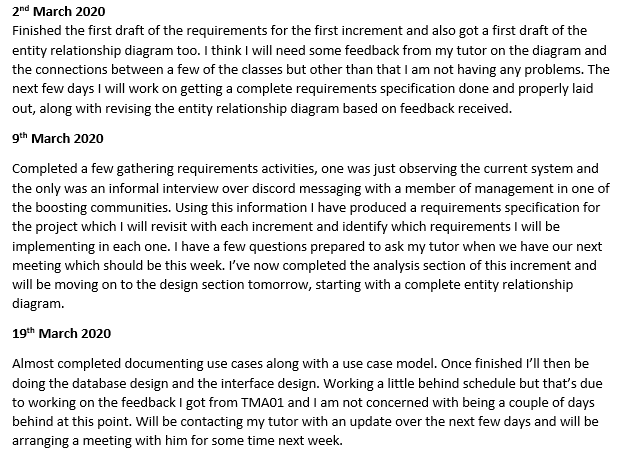
I kept in regular contact with my tutor throughout my project, updating him via email with my progress every 2 weeks. This meant that I was able to keep him informed of any issues I was having and also of how I was progressing against my schedule. An example of a few of those emails are below, with an overview of my correspondence with my tutor located in the appendices section (Appendix 23).



****

**Project Journal**

Throughout my project I maintained a journal located in a separate word document that allowed me to keep track of my thoughts as I progressed through the project. I will only include a screenshot of a few entries here however the full contents of my project journal will be located in the appendices section (Appendix 24).

****

**Resources**

There were many resources I needed throughout my project, some of which weren’t known right at the beginning of my project and only became relevant once development had begun. The table below contains a list of all the resources I identified at the beginning of my project along with additional resources that were used later on.

|  |  |
| --- | --- |
| **Resource** | **Notes** |
| Users | I had access to a large number of potential clients of my application. I am a member of 2 separate boosting communities with a combined 30,000+ members with whom I was able to gather information from directly. I am also part of a World of Warcraft subreddit forum where I would have been able to post questions to the community which could have provided me with some larger more generalised feedback, however I didn’t feel it was necessary for my application. As well as being a part of the communities above I also play the game daily with friends and so was able to have a much more casual chat with them about the things they’d like to see in an application like the one I have been developing. |
| Project Planner | I needed to have a detailed plan for my project containing all of the tasks that needed to be completed, both for the application itself and for the coursework pertaining to this module. I decided to use a Gantt graph to display my schedule, but to make one from scratch would’ve been very time consuming, so I instead looked online for a tool that would make the process of creating a Gantt graph much simpler. I found a web app called TeamGantt which is a subscription based app that is specifically used for creating Gantt graphs that can be shared and accessed between teams and team members (TeamGantt, 2020). However they did offer a free subscription which is limited to only 1 Gantt graph and only 3 people can access it, but those limitations did not affect my project as I was the only one working on it. |
| Integrated Development Environment (IDE) | For coding the system I chose to use an IDE instead of a simple text editor. After looking at numerous different sources online I found that one of the most popular IDE’s available today was Microsoft Visual Studio. It has all of the basic features you would expect from an IDE as well as allowing you to install specific component bundles to suit your needs. In the case of my system, because I coded it in Python, Visual Studio offers a Python component bundle which can be easily installed alongside the IDE from the visual studio website (Microsoft, 2020). |
| Interface of existing system | As I am a member of existing boosting communities I have access to the old system they used and so was able to use that when designing the interface for my application. From the research I did with gathering requirements I found that users liked the interface of the existing system and the way that information was displayed to them so this is something I wanted to keep as close to the old system as possible so that my application would seem familiar to them. |
| Database design software | The software I used for my project diagrams was UMLet which is a basic tool for creating UML diagrams that I came across on a previous module. However when it came to mapping out my database this software was not adequate for my needs. For the final blueprint of my database I used a tool called SqlDBM, which is a database diagram tool that supports MySQL. It was a paid service however they did have a free option as a trial run for their service and this was sufficient enough for my project. |

**Skills**

|  |  |
| --- | --- |
| **Skill** | **Notes** |
| Coding Language | I decided to write my project in Python, a language with which I had some brief experience with in a previous module, so although I was familiar with the basics there was still a lot of room to expand upon my current skill level. It is also the job role I would like to pursue after completing my studies so I felt it was appropriate to for me to create something that will help with showing off my skills to a future employer. |
| Database | On another module I had some experience using MySQL databases which is the database I chose to use for my application. I did some research into the different databases available and I found that in terms of open source software, MySQL is by far regarded as the best. I did not want to be spending any money on any pieces of software for my project, so that ruled out a few of the most popular databases like Oracle, as they require payment. Out of the available open source databases MySQL was frequently ranked as the best or one of the best (Guru99, 2020). Couple the reputation of MySQL with the fact that I already had some experience using it on previous module meant it made the most sense to go with MySQL for my project. |
| Requirements Gathering | There were a number of different methods of requirements gathering that I used in order to elicit the requirements for my application. Specifically I observed the existing system and also conducted a 1-to-1 interview with a member of the admin team in one of the boosting communities. From this I was able to get a good idea of the requirements and features that my new application would need to have. |
| Task scheduling | As well as needing to have the software to clearly display the schedule for my project, I also needed to use my own skills in creating the deadlines for each task. In previous modules I have been given schedules on what to read and the work to do week-to-week, whereas in this module the only schedule provided was the deadline for each assignment. This meant that I needed to think ahead several months and allocate my time appropriately to the various different tasks in my project. |

**Risks**

Below are the risks I identified that could hinder the progress of my project. There is a description of each one along with the plans I made to mitigate each one in order to deliver a solution within the time frame of this module. There is also a column describing the effectiveness my mitigation tactics had on each risk. I created this table at the beginning of my project and added any new risks that occurred during my project; one example being the coronavirus pandemic.

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Description** | **Mitigation** | **Effectiveness** |
| Illness | If I become ill during the development of my system and am unable to continue working on it then this will delay my schedule and the completion of each task. | When scheduling the tasks for my project I left a 2-3 week buffer between the scheduled completion of my project and the date I planned to begin writing up my EMA. This allowed for any delays in my project work due to illness. | I spent most of the project working behind schedule however I was always still on track to complete the project within the time frame of this module due to the 2-3 week buffer I left at the end of the target completion date. |
| Scope of the project | The project scope must be refined so that it is feasible for me to complete a working solution of the system in the time frame allocated for this module. | I liaised with my tutor to make sure that the system I plan to implement for my project is one that meets the requirements for the module and can also be completed within the time that I had. | Through guidance from my tutor and feedback on TMA’s I refined the scope of the project so that it fulfilled the module criteria and was feasible to be completed on time. |
| Hardware failure | Computers are not immortal and there is always the possibility for components to break. Due to the nature of this module and the way it is delivered being completely online, I am reliant on having computer and internet access for the duration of my module. | The only way in which I was able to minimise this risk is by regularly making backups of all the work I did and store it somewhere outside of my own computer. In my household I am fortunate that there is another desktop computer available for me to use should my own one fail so this is where I regularly made backups of all my project work to. | This is one risk that I seemingly ignored for the first few months of my project as I did not make any backups of my work. However once I began to code my application I started to manually make a backup of my work weekly on the second computer we have in my home. |
| Coronavirus pandemic | The recent outbreak of coronavirus has led to the UK being put in lockdown for the foreseeable future. This can impact my project work in many ways, for example, I will not be able to meet with any clients face to face when conducting interviews. Also it has resulted in my immediate family all working from home which can make focusing on my project more difficult due to the household being busy and full of distractions. | The method I used for conducting the interview was done entirely online through a messaging app that was already used by the interviewee. With regards to my family, I informed them that I will be doing work on my project during typical normal working hours so could they try not to disturb me. Additionally I planned to do more work on my project late in the evenings when everyone else has gone to bed as there would be no disturbances. | The coronavirus pandemic is one that is still going on today, I was able to conduct interviews with the client with no problems however working during typical working hours very quickly became a problem. The change I made was switching to do most of my project work late at night where there were no interruptions, and this is a change that had a very positive impact on the progress being made on my project. |
| Availability of software | There are a few pieces of software I will need to use that I do not currently have access too. These include an IDE, a database, and a database design tool. Many of these require payment in order to use them and as I am a full time student this may not be something I will be able to afford. | There are many free alternatives to the paid versions of the software I required with some of the best ones surprisingly being the free ones. I did have to do a bit of research into which of the free options were best suited to the needs of my project, but it was no more work than I would have needed to do if I was looking into the paid options instead. | I had to spend a little more time looking for alternatives to the paid pieces of software, however I was able to find free or open source software for all of the needs my project had. |

Of the risks in the table above, some remained active risks for most of my project. Illness was an active risk for the entire duration as there was a chance that I would become ill at any point during the project. There was no change needed for the mitigation for illness because my schedule always had a buffer of a few weeks between the targeted completion date and the end of the module. Hardware failure was also always an active risk because any part of my computer could have failed at any point right up to the final EMA due date. The mitigation for this stayed the same in that I made regular backups of all my work so that should anything have happened the loss would have been kept to a minimum. Finally the coronavirus pandemic was unfortunately an active risk for most of the project with some parts of the UK still being in lockdown as my module is almost finished. The mitigation for this stayed the same, however I put more focus on doing my project work late in the evenings as there were lots of interruptions during the day due to my family members all working from home.

**Personal Development**

Through working on my project I have learnt that when creating a schedule for the development of an application, there needs to be an emphasis on properly researching the amount of time each task is likely to take to complete. In my project I think this is one of the areas I handled the worst, as my estimates were consistently poor because I didn’t take into account any new skills I would need to learn to complete some of the tasks. From my own experience I don’t believe I put enough time and thought into all of the tasks on my schedule, for example when looking back at the estimated time for the coding in the first increment, I didn’t consider that I would likely have to do a significant amount of research and learning first in order to be able to utilise new tools correctly.

When it came to actually learning the new skills or expanding upon existing ones I did find this to be an area I was quite good at. The official documentation for each of the tools I used like Django and MySQL were by far the most valuable resources that I had identified when going into the development of my application. They were both resources that I used when initially starting my application and I still regularly refer to them when working on the code for any of the application features. I feel as though I did sufficient research at the beginning of my project into the resources and literature that I believed would benefit me later on in the development process. The proof for this is shown by the fact that I did not really need to look for many extra resources or help throughout the entirety of the first increment, most of the answers to any questions or problems I had were available in the resources I had already bookmarked.

In terms of extending my current knowledge and skills, the testing for my first increment and creating and running tests for a Django project was something I had never done before. This required me to read the documentation for the Django testing suite along with documentation for the third party tools I used like pytest-django and Coverage. Once I had learned how to write a few different tests it became easier to create tests for the rest of my project. I believe that if I had moved on to work on increment 2 and 3 I would have found sticking to the schedule I had set much easier than with the first increment. This is because I would have already learned all of the new skills during the first increment, and so the latter increments would have been done using tools that I was now familiar with. The reasons for only completing the first increment in my schedule I will cover at the end of this section.

Throughout my project I have tried a few different working environments, including; working early morning/late at night, listening/not listening to music with words, listening/not listening to only instrumental music, working for long/short periods. I have discovered that I find it much easier to concentrate on my work when I am listening to music as it helps to eliminate any distractions like background noise from other people in the house. I also find it difficult to think clearly if the music has words or speaking in it which is why I ended up mostly listening to instrumental music when I was working. I also found that taking frequent breaks whilst working wasn’t something that worked well for me. Once I take a break I find it very difficult to get back into the working mind-set so I would much rather work for a solid few hours in that working mind-set without interruptions. One other thing I noticed is that I also seem to work much better late at night compared to in the mornings or during the day. Whilst that may be due to the rest of my household being asleep at night and so there are no interruptions, I believe that I find myself to be much more relaxed at night and so find it easier to focus on work. The reason I find myself more relaxed is because everything is much more peaceful at night due to there being no cars on the roads and it being dark in my room. This has a very large impact on my productivity which is something I had never realised prior to working on this project.

The biggest lesson I have learned through completing this project is the importance of one’s mental health and the impact having unrealistic deadlines can have. At the beginning of my project I felt optimistic about the schedule I had created and excited to monitor my progress against it. However after falling behind on the schedule because I had made poor estimations for the time to complete tasks in the beginning of the first increment, I noticed a shift in my mental well-being. Each revision of my schedule was done with my mental-health in mind as well as making sure that my project would still be completed. The final revision where I made the decision to refine the scope of the project to not include the requirements of the third increment, was done because it was unfeasible to complete all 3 increments. Instead I shifted my focus onto the second increment and getting that completed and figured that it would help to fix problems I was having with my mental health. This turned out to not be enough because at the end of July I had a rapid decline in my mental health which resulted in an admittance to hospital for a few days for treatment. Looking through my project journal and also the emails with my tutor you will be able to see that July is the last month that I had any interaction with either, which is because of the reason mentioned above.

My closing thoughts on the way in which I tackled this project are that I have learned a lot about the ways in which a project like this should be approached. I wouldn’t have put that much value on the preparation side of a project prior to this module, however looking back at it I think if I had put a much larger focus on the organisation and planning for my project then I would’ve been able to deliver a more polished application that satisfied all of the requirements.

# References

Araujo, M. & Greenfeld, D., 2020. *django-crispy-forms.* [Online]   
Available at: https://django-crispy-forms.readthedocs.io/en/latest/  
[Accessed 3 July 2020].

Bagui, S. & Earp, R., 2003. *Database design using entity-relationship diagrams.* 1st ed. Boca Raton: Auerbach.

Blizzard Entertainment, 2016. *Blizzard Termination of Service Agreement.* [Online]   
Available at: https://www.blizzard.com/en-gb/legal/b8a1574a-8137-44ad-acf0-11c92e90b26f/blizzard-termination-of-service-agreement  
[Accessed 14 April 2020].

Blizzard Entertainment, 2020. *WoW New Player's Guide.* [Online]   
Available at: https://worldofwarcraft.com/en-us/game/new-players-guide  
[Accessed 20 February 2020].

Bootstrap, 2020. *Bootstrap: Getting Started.* [Online]   
Available at: https://getbootstrap.com/docs/4.5/getting-started/introduction/  
[Accessed 4 July 2020].

Burleson, D., n.d. *Top-down vs. Bottom-Up Object Database Design.* [Online]   
Available at: http://www.dba-oracle.com/t\_object\_top\_down\_bottom\_up.htm  
[Accessed 9 June 2020].

Coverage.py, 2020. *Coverage.py.* [Online]   
Available at: https://coverage.readthedocs.io/en/coverage-5.2.1/  
[Accessed 4 July 2020].

Django, 2020. *Django.* [Online]   
Available at: https://www.djangoproject.com/  
[Accessed 12 July 2020].

Django, 2020. *Django Documentation.* [Online]   
Available at: https://docs.djangoproject.com/en/3.0/  
[Accessed 14 June 2020].

Django, 2020. *Password management in Django.* [Online]   
Available at: https://docs.djangoproject.com/en/3.1/topics/auth/passwords/  
[Accessed 24 June 2020].

Django, 2020. *Security in Django.* [Online]   
Available at: https://docs.djangoproject.com/en/3.1/topics/security/#sql-injection-protection  
[Accessed 16 June 2020].

Django, 2020. *User Authentication in Django.* [Online]   
Available at: https://docs.djangoproject.com/en/3.0/topics/auth/  
[Accessed 1 July 2020].

Django, 2020. *Using the Django authentication system.* [Online]   
Available at: https://docs.djangoproject.com/en/3.1/topics/auth/default/  
[Accessed 15 June 2020].

Eriksson, U., 2012. *How to Use Interviews to Gather Requirements.* [Online]   
Available at: https://reqtest.com/requirements-blog/how-to-use-interviews-to-gather-requirements/  
[Accessed 23 June 2020].

gov.uk, 2018. *Understanding WCAG 2.1.* [Online]   
Available at: https://www.gov.uk/service-manual/helping-people-to-use-your-service/understanding-wcag  
[Accessed 12 Jun 2020].

Guru99, 2020. *13 Best Free Database (2020 List).* [Online]   
Available at: https://www.guru99.com/free-database-software.html  
[Accessed 23 June 2020].

Guru99, 2020. *Incremental Model in SDLC: Advantage & Disadvantage.* [Online]   
Available at: https://www.guru99.com/what-is-incremental-model-in-sdlc-advantages-disadvantages.html  
[Accessed 1 March 2020].

Guru99, 2020. *RAD Model: Advantage & Disadvantage.* [Online]   
Available at: https://www.guru99.com/what-is-rad-rapid-software-development-model-advantages-disadvantages.html  
[Accessed 1 March 2020].

Guru99, 2020. *Spiral Model: Advantage & Disadvantage.* [Online]   
Available at: https://www.guru99.com/what-is-spiral-model-when-to-use-advantages-disadvantages.html  
[Accessed 1 March 2020].

Guru99, 2020. *Waterfall Model SDLC: Advantage & Disadvantage.* [Online]   
Available at: https://www.guru99.com/what-is-sdlc-or-waterfall-model.html  
[Accessed 1 March 2020].

Inflectra, 2020. *Requirements Gathering.* [Online]   
Available at: https://www.inflectra.com/ideas/topic/requirements-gathering.aspx  
[Accessed 7 April 2020].

Microsoft, 2020. *Visual Studio 2019.* [Online]   
Available at: https://visualstudio.microsoft.com/vs/  
[Accessed 13 April 2020].

MySQL, 2020. *Chapter 11 Data Types.* [Online]   
Available at: https://dev.mysql.com/doc/refman/8.0/en/data-types.html  
[Accessed 10 April 2020].

MySQL, 2020. *MySQL 8.0 Reference Manual.* [Online]   
Available at: https://dev.mysql.com/doc/refman/8.0/en/  
[Accessed 14 April 2020].

Netop, 2019. *GDPR: TL;DR?.* [Online]   
Available at: https://blog.netop.com/the-5-keys-of-gdpr-compliance  
[Accessed 26 June 2020].

OWASP, 2020. *OWASP Top Ten.* [Online]   
Available at: https://owasp.org/www-project-top-ten/  
[Accessed 14 April 2020].

Pencil, 2019. *Pencil Project.* [Online]   
Available at: https://pencil.evolus.vn/  
[Accessed 14 April 2020].

pytest-django, 2020. *pytest-django Documentation.* [Online]   
Available at: https://pytest-django.readthedocs.io/en/latest/  
[Accessed 2 July 2020].

Sanders, M., 2020. *The Most Popular Python Web Frameworks.* [Online]   
Available at: https://scoutapm.com/blog/the-most-popular-python-web-frameworks-in-2020  
[Accessed 18 June 2020].

Spence, I. & Bittner, K., 2002. *Use Case Modeling.* 1st ed. s.l.:Addison-Wesley Professional.

SqlDBM, 2020. *Develop Database Online.* [Online]   
Available at: https://sqldbm.com/Home/  
[Accessed 11 April 2020].

TeamGantt, 2020. *TeamGantt.* [Online]   
Available at: https://www.teamgantt.com/  
[Accessed 16 May 2020].

usabilty.gov, 2020. *Wireframing.* [Online]   
Available at: https://www.usability.gov/how-to-and-tools/methods/wireframing.html  
[Accessed 3 June 2020].

Visual Paradigm, 2020. *Visual Paradigm.* [Online]   
Available at: https://www.visual-paradigm.com/  
[Accessed 10 July 2020].

Visual Paradigm, 2020. *What is Entity Relationship Diagram (ERD)?.* [Online]   
Available at: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/  
[Accessed 9 April 2020].

Widrig, D. & Leffingwell, D., 1999. *Managing Software Requirements: A Unified Approach.* 1st ed. s.l.:Addison-Wesley Professional.

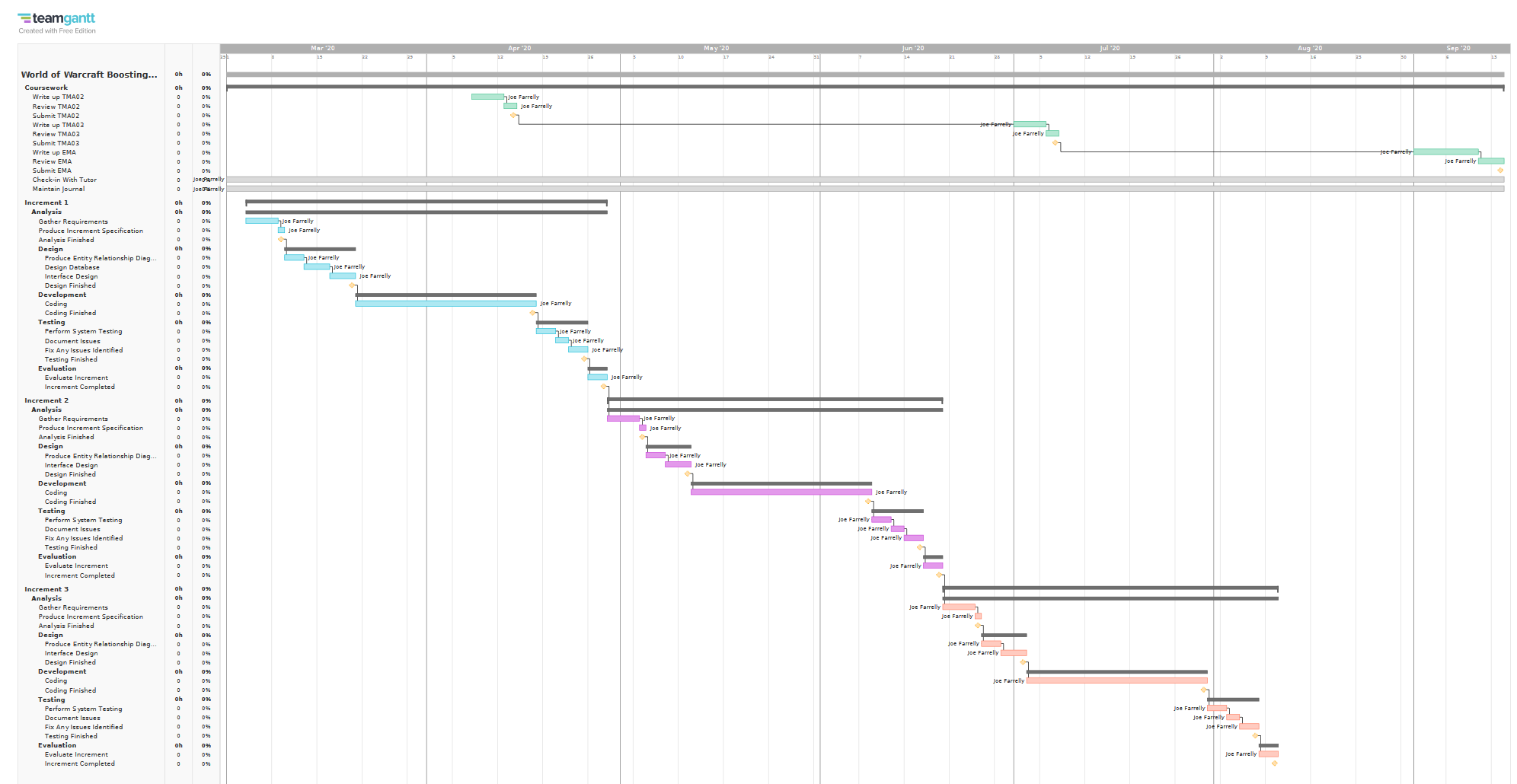
XAMPP, 2020. *XAMPP.* [Online]   
Available at: https://www.apachefriends.org/index.html  
[Accessed 24 June 2020].

**Appendices**

|  |  |  |
| --- | --- | --- |
| **Lifecycle model** | **Strengths** | **Weaknesses** |
| Classic Waterfall  (Guru99, 2020) | - Works well on smaller projects where requirements are well defined  - Quality assurance tests are done before completing each stage  - Elaborate documentation is done for every phase of the software’s development cycle  - Project is completely dependent on the project team with minimum intervention from the client | - Not suited for larger more complex projects with frequent requirement changes  - The testing stage doesn’t occur until very late in the developmental process  - Documentation takes up a lot of developers time  - Clients feedback cannot be included with ongoing development |
| Spiral  (Guru99, 2020) | - Additional functionality or changes can be done at a later stage  - Continuous or repeated development helps in risk management  - Development is fast and features are added in systematically  - There is always space for client feedback | - High risk of not meeting the planned schedule  - For smooth operation the spiral model protocol needs to be followed very strictly  - A lot of documentation because there are intermediate phases  - Isn’t advisable for smaller projects mainly due to cost |
| Incrementation  (Guru99, 2020) | - Software will be generated quickly during the software life cycle  - It is flexible in terms of changing requirements and scope  - The client can provide feedback and input and each new build  - Errors are easily identified | - Requires good planning design  - Each iteration phase is rigid and does not overlap with any others  - Fixing a problem in one unit requires the correction to be done in all of the units, so can be time consuming |
| Rapid Application Development (RAD) (Guru99, 2020) | - Flexible and adaptable to changes  - The usage of code generators and code reuse means there is a reduction is manual coding  - Each phase in a RAD cycle delivers the highest priority functionality to the client | - It cannot be used for smaller projects  - Requires highly skilled designers/developers  - Progress and problems are hard to track as there is no documentation to demonstrate what has been done |

**Appendix 1**

Table exploring the advantages and disadvantages of various lifecycle models.

****

**Appendix 2**

Iteration 1 of project schedule presented as a Gantt graph.

# 

**Appendix 3**  
Screenshot of the interview I conducted with a member of one of the boosting communities over messaging on the application Discord.

|  |  |
| --- | --- |
| **ID** | UC01 |
| **Name** | Create account |
| **Description** | A new user accessing the system for the first time can create their own account by providing their email and a password. |
| **Actor(s)** | User |
| **Trigger** | User accesses the web application using a browser and clicks Create Account button |
| **Preconditions** | User does not already have an account on the system with the email address provided. |
| **Postconditions** | An account is created for the user and they are notified of their account being created successfully or the details entered by the user were invalid and they are notified that their account creation was unsuccessful |
| **Main Success Scenario** | 1. User accesses website  2. User clicks Create Account button  3. User inputs a valid email address that is not already registered to an account  4. User inputs a valid password  5. User clicks Complete Sign Up button  6. New User account is created  7. User is notified that their account was created successfully |
| **Extensions** | 3a. User inputs an invalid email address  3b. User inputs an email address that is already registered to an account  4c. User inputs an invalid password 6abc. User account is not created  7abc. User is notified that there is an error with the provided details and asked to enter them again |
| **Priority** | High – First Increment |

|  |  |
| --- | --- |
| **ID** | UC02 |
| **Name** | Log in to existing account |
| **Description** | A returning user is able to log in to their account |
| **Actor(s)** | User |
| **Trigger** | User accesses the web application using a browser and clicks the Log In button |
| **Preconditions** | The user has previously created an account successfully. |
| **Postconditions** | The user is logged into their account and able to access the system or the user entered incorrect login credentials and is not granted access to their account |
| **Main Success Scenario** | 1. User accesses website  2. User inputs their email address  3. User inputs their password  4. User clicks Sign In button  5. User is logged into their account |
| **Extensions** | 2a. User inputs an incorrect email address  3b. User inputs an incorrect password  5ab. User is notified that there is an error with the log in details and is not logged into their account |
| **Priority** | High – First Increment |

|  |  |
| --- | --- |
| **ID** | UC03 |
| **Name** | View account |
| **Description** | User can view an overview of their account details (name, realm, and other details depending on their role) |
| **Actor(s)** | User |
| **Trigger** | User clicks Account button on their home page |
| **Preconditions** | User has already created an account and the user is logged in to the system |
| **Postconditions** | User is shown an overview of their account details |
| **Main Success Scenario** | 1. User clicks Account button  2. User is shown their account details |
| **Extensions** | N/A |
| **Priority** | Medium – Second Increment |

|  |  |
| --- | --- |
| **ID** | UC04 |
| **Name** | View boost services provided |
| **Description** | Anyone can view a page containing the details of all the boosting services offered |
| **Actor(s)** | Buyer, User, Advertiser, Booster, Admin |
| **Trigger** | They click the Prices button located on the home page of the system |
| **Preconditions** | There is at least one boost that the community provides |
| **Postconditions** | They are shown a page containing all of the boosts that are offered along with the price of each one |
| **Main Success Scenario** | 1. User clicks Prices button  2. User is shown a price list for all boosts provided |
| **Extensions** | N/A |
| **Priority** | High – First Increment |

|  |  |
| --- | --- |
| **ID** | UC05 |
| **Name** | Book dungeon boost |
| **Description** | An advertiser books a buyer onto a dungeon boost |
| **Actor(s)** | Advertiser, Buyer |
| **Trigger** | A buyer reaches out to an advertiser looking to book a dungeon boost |
| **Preconditions** | A dungeon boost matching the requirements of the buyer is one that the community offers. The advertiser is logged in and has the correct role within the system |
| **Postconditions** | A new booking is created for the buyer with the details of the boost now stored in the system |
| **Main Success Scenario** | 1. Buyer contacts advertiser requesting a dungeon boost  2. Advertiser signs in to the system  3. Advertiser goes to book dungeon boost page  4. Advertiser fills in relevant details on the booking page  5. Advertiser clicks Complete Booking button  6. A new booking is created using the details provided by the advertiser  7. Advertiser is notified that the booking was successful  8. Advertiser notifies the buyer that their dungeon boost has been booked successfully and informs them to wait patiently whilst boosters sign up to their dungeon boost |
| **Extensions** | N/A |
| **Priority** | Medium – Second Increment |

|  |  |
| --- | --- |
| **ID** | UC06 |
| **Name** | Book buyer on raid boost |
| **Description** | An advertiser books a buyer onto a raid boost |
| **Actor(s)** | Advertiser, Buyer |
| **Trigger** | A buyer reaches out to an advertiser looking to book a raid boost |
| **Preconditions** | A raid boost matching the requirements of the buyer is one that the community offers and a raid boost booking for the specific boost has already been created. The advertiser is logged in and has the correct role within the system |
| **Postconditions** | The buyer is added onto the existing raid booking along with the details of the transaction between the advertiser and the buyer |
| **Main Success Scenario** | 1. Buyer contacts advertiser requesting a raid boost  2. Advertiser accesses the system  3. Advertiser goes to book raid boost page  4. Advertiser fills in the relevant details on the booking page and selects one of the existing raid bookings to add the buyer to  5. Advertiser clicks Complete Booking button  6. The buyer is added onto the raid boost booking along with the details of the transaction  7. Advertiser is notified the booking was successful  8. Advertiser notifies the buyer that their place on the raid boost has been booked successfully and gives them the time and date it will take place |
| **Extensions** | N/A |
| **Priority** | Medium – Second Increment |

|  |  |
| --- | --- |
| **ID** | UC07 |
| **Name** | Cancel dungeon boost |
| **Description** | An advertiser cancels the dungeon boost they have booked for a buyer |
| **Actor(s)** | Advertiser, Buyer |
| **Trigger** | A buyer who has booked a dungeon boost tells the advertiser they booked with that they no longer want the dungeon boost |
| **Preconditions** | The buyer has an existing dungeon boost booked through said advertiser that has not started yet. The advertiser is logged in and has the correct role within the system |
| **Postconditions** | The booking for the dungeon boost has been deleted from the system and the advertiser is notified that it has been cancelled successfully |
| **Main Success Scenario** | 1. Buyer contacts their advertiser requesting to cancel their dungeon boost  2. Advertiser accesses the system  3. Advertiser goes to Cancel Booking page  4. Advertiser selects the correct booking they wish to cancel  5. Advertiser clicks Cancel button  6. The booking is deleted from the system  7. The advertiser is notified that the cancellation was successful  8 The advertiser notifies the buyer that their booking for a dungeon boost has been cancelled successfully |
| **Extensions** | N/A |
| **Priority** | Medium – Second Increment |

|  |  |
| --- | --- |
| **ID** | UC08 |
| **Name** | Remove buyer from raid boost |
| **Description** | An advertiser removes a buyer from a raid boost that they have been booked for |
| **Actor(s)** | Advertiser, Buyer |
| **Trigger** | A buyer who has booked a raid boost tells the advertiser they booked with that they no longer want the raid boost |
| **Preconditions** | The buyer is already booked on a raid boost that has not begun yet and was booked by said advertiser. The advertiser is logged in and has the correct role within the system |
| **Postconditions** | The buyer has been removed from the raid boost booking and the advertiser is notified that the buyer was removed successfully |
| **Main Success Scenario** | 1. Buyer contacts their advertiser requesting to cancel their raid boost  2. Advertiser accesses the system  3. Advertiser goes to Cancel Booking page  4. Advertiser selects the correct booking they wish to cancel  5. Advertiser clicks Cancel button  6. The booking is deleted from the system  7. The advertiser is notified that the cancellation was successful  8 The advertiser notifies the buyer that their booking for a raid boost has been cancelled successfully |
| **Extensions** | N/A |
| **Priority** | Medium – Second Increment |

|  |  |
| --- | --- |
| **ID** | UC09 |
| **Name** | View dungeon boost |
| **Description** | An advertiser and booster can view all of the details linked to a dungeon boost that they are directly connected to (either they are an advertiser who booked the dungeon boost or a booster who is signed up for the boost) |
| **Actor(s)** | Advertiser, Booster |
| **Trigger** | Advertiser or Booster clicks the View Details button next to a dungeon boost listed in their account summary |
| **Preconditions** | There is a dungeon boost booked that the advertiser booked or the booster has signed up for. The advertiser/booster is already logged in and has the correct role within the system |
| **Postconditions** | The advertiser/booster is shown a page containing all of the details pertaining to the dungeon boost that they selected |
| **Main Success Scenario** | 1. Advertiser/Booster clicks the View Details button next to a dungeon boost listed in their account summary  2. They are shown the details of the dungeon boost they selected |
| **Extensions** | N/A |
| **Priority** | Low – Third Increment |

|  |  |
| --- | --- |
| **ID** | UC10 |
| **Name** | View raid boost |
| **Description** | An advertiser and booster can view all of the details linked to a raid boost that they are directly connected to (they are an advertiser who has booked a buyer onto the boost or they are a booster who is signed up for the boost) |
| **Actor(s)** | Advertiser, Booster |
| **Trigger** | Advertiser or booster clicks the View Details button next to a raid boost listed in their account summary |
| **Preconditions** | There is a raid boost booked that an advertiser has booked a buyer on, or a booster has signed up for. The advertiser/booster is already logged in and has the correct role within the system |
| **Postconditions** | The advertiser/booster is shown a page containing all of the details pertaining to the raid boost that they selected |
| **Main Success Scenario** | 1. Advertiser/Booster clicks the View Details button next to a raid boost listed in their account summary  2. They are shown the details of the raid boost they selected |
| **Extensions** | N/A |
| **Priority** | Low – Third Increment |

|  |  |
| --- | --- |
| **ID** | UC11 |
| **Name** | Sign up to boost |
| **Description** | A booster can sign up to a dungeon or raid boost |
| **Actor(s)** | Booster |
| **Trigger** | Booster navigates to the open boosts page and clicks on a boost they want to sign up to |
| **Preconditions** | There is at least 1 dungeon or raid boost booked that the booster is eligible to sign up for. The booster is already logged in and has the correct role within the system |
| **Postconditions** | Booster is signed up to the boost they selected and is shown confirmation that they have signed up successfully |
| **Main Success Scenario** | 1. Booster selects a boost they wish to sign up to  2. Booster clicks sign up button  3. Booster is shown a pop up detailing the boost they are signing up for along with their role in the boost  4. Booster confirms sign up by clicking confirm on the pop up  5. Booster is signed up to the selected boost and is notified that they have been successfully signed up to the boost |
| **Extensions** | N/A |
| **Priority** | Low – Third Increment |

|  |  |
| --- | --- |
| **ID** | UC12 |
| **Name** | Cancel sign up to boost |
| **Description** | A booster can cancel their sign up to a dungeon or raid boost |
| **Actor(s)** | Booster |
| **Trigger** | Booster navigates to their account summary page and clicks on the boost they would like to cancel their sign up for |
| **Preconditions** | The booster is signed up to a dungeon or raid boost that has not begun yet. The booster is logged in and has the correct role within the system |
| **Postconditions** | Booster is removed from the boost they have selected to cancel and is shown confirmation that they have successfully removed their sign up to the boost |
| **Main Success Scenario** | 1. Booster clicks view details on a boost they want to un-sign for that is listed in their account summary  2. Booster is shown the details of the boost they selected along with a cancel signup button  3. Booster clicks cancel sign-up button  4. Booster is shown a popup asking them to confirm the boost they wish to cancel signup for  5. Booster is removed from the selected boost and is notified that they have successfully cancelled their sign up to said boost |
| **Extensions** | N/A |
| **Priority** | Low – Third Increment |

|  |  |
| --- | --- |
| **ID** | UC13 |
| **Name** | Book raid boost |
| **Description** | An advertiser can book a raid boost |
| **Actor(s)** | Advertiser |
| **Trigger** | Advertiser clicks the Create Raid Boost button on their home page |
| **Preconditions** | The community offers the raid boost that the advertiser wants to book. The advertiser is logged in and has the correct role within the system. |
| **Postconditions** | A new raid booking has been created using the details provided by the advertiser and they are notified that the booking has been created successfully |
| **Main Success Scenario** | 1. Advertiser clicks the Create Raid Boost button on their home page  2. Advertiser fills in the relevant details on the create raid boost page  3. Advertiser clicks Create Boost button  4. The raid boost booking is created  5. Advertiser is shown a page detailing their raid boost with confirmation that it has been created successfully |
| **Extensions** | N/A |
| **Priority** | Medium – Second Increment |

|  |  |
| --- | --- |
| **ID** | UC14 |
| **Name** | Cancel raid boost |
| **Description** | An advertiser can cancel a raid boost |
| **Actor(s)** | Advertiser |
| **Trigger** | Advertiser clicks cancel boost button on the raid boost they would like to cancel |
| **Preconditions** | The advertiser has a raid boost booked that has not begun yet. The advertiser is logged in and has the correct role within the system |
| **Postconditions** | The raid boost has been deleted from the system and the advertiser is notified that their raid boost has been cancelled successfully |
| **Main Success Scenario** | 1. Advertiser clicks cancel boost button on the raid boost they want to cancel  2. Advertiser is shown a pop up with the details of the raid boost asking them to confirm they would like to cancel the boost  3. Advertiser clicks confirm and the boost in cancelled  4. Advertiser is notified that the selected boost has been cancelled successfully |
| **Extensions** | N/A |
| **Priority** | Medium – Second Increment |

|  |  |
| --- | --- |
| **ID** | UC15 |
| **Name** | Edit account |
| **Description** | Admin can edit the details of an account |
| **Actor(s)** | Admin, User, Advertiser, Booster |
| **Trigger** | User, advertiser or booster contacts the admin requesting to change some details on their account |
| **Preconditions** | User, advertiser or booster already has an account registered on the system. The admin is logged in |
| **Postconditions** | The user, advertiser or booster’s account has been updated on the system and the admin is notified that the account has been updated successfully |
| **Main Success Scenario** | 1. User/advertiser/booster contacts the admin requesting details to be changed on their account  2. Admin accesses the system  3. Admin navigates to the manage accounts page  4. Admin selects the account they wish to change the details for  5. Admin clicks edit account button  6. Admin changes the relevant details and clicks Save  7. Admin is notified that the account details have been changed successfully  8. Admin informs the user/advertiser/booster that their account details have been updated successfully |
| **Extensions** | 6a. The new details provided are invalid  7a. Admin is notified that there was an error and the details have not been changed  8a. Admin informs the user/advertiser/booster that their account details could not be updated |
| **Priority** | High – First Increment |

|  |  |
| --- | --- |
| **ID** | UC16 |
| **Name** | Delete account |
| **Description** | Admin can delete an account |
| **Actor(s)** | Admin, User, Advertiser, Booster |
| **Trigger** | User, advertiser or booster contacts the admin requesting for their account to be deleted |
| **Preconditions** | User, advertiser or booster already has an account registered on the system. The admin is logged in |
| **Postconditions** | The user, advertiser or booster’s account has been deleted from the system and the admin is notified that the account has been successfully deleted |
| **Main Success Scenario** | 1. User/advertiser/booster contacts the admin requesting for their account to be deleted  2. Admin accesses the system  3. Admin navigates to the manage accounts page  4. Admin selects the account they wish to delete  5. Admin clicks delete account button  6. Admin is shown popup requesting confirmation they want to delete the selected account  7. Admin is notified that the account has been deleted successfully  8. Admin informs the user/advertiser/booster that their account has been deleted successfully |
| **Extensions** | 5a. Account cannot be deleted because it is signed up to boosts that have not happened yet  7a. Admin is notified that the account cannot be deleted due to being signed up to boosts  8a. Admin informs the user/advertiser/booster that their account cannot be deleted as they are signed up to future boosts |
| **Priority** | High – First Increment |

|  |  |
| --- | --- |
| **ID** | UC17 |
| **Name** | Edit roles |
| **Description** | Admin can edit the role of an account |
| **Actor(s)** | Admin |
| **Trigger** | Admin selects the account that they want to change the role of |
| **Preconditions** | An account exists within the system. The admin is logged in |
| **Postconditions** | The account selected has had their role updated and the admin is notified that the account role has been assigned successfully |
| **Main Success Scenario** | 1. Admin selects the account they want to change the role of  2. Admin selects the new role they want to give the selected account  3. Admin enters any new details that are required for the new account role  4. Admin clicks Update button  5. Admin is shown popup requesting confirmation that they want to make this role change on the selected account  6. Admin is notified that the account role has been updated successfully |
| **Extensions** | 3a. Admin doesn’t enter all of the required details  6a. Admin is notified that there is an error and the account role could not be updated |
| **Priority** | Low – Third Increment |

|  |  |
| --- | --- |
| **ID** | UC18 |
| **Name** | Amend balance |
| **Description** | Admin can amend the balance of an account |
| **Actor(s)** | Admin |
| **Trigger** | Admin selects an account they want to amend the balance of |
| **Preconditions** | An account exists within the system. The admin is logged in |
| **Postconditions** | The account the admin has selected has had the amount specified on the amend balance page credited/deducted from their account and the admin is notified that the account balance has been updated successfully |
| **Main Success Scenario** | 1. Admin selects the account they want to amend the balance of from the accounts page  2. Admin selects either credit or deduct and enters an amount  3. Admin clicks Save button  4. Admin is shown popup requesting confirmation that they want credit/deduct the selected account by the specified amount  5. Admin is notified that the balance of the selected account has been updated successfully |
| **Extensions** | 2a. Admin selects deduct and enters an amount  5a. Admin is notified that the balance of the selected account could not be updated because the balance on the account would become negative |
| **Priority** | Low – Third Increment |

|  |  |
| --- | --- |
| **ID** | UC19 |
| **Name** | View all boosts |
| **Description** | Admin can view a list of all boosts on the system |
| **Actor(s)** | Admin |
| **Trigger** | Admin navigates to the Boosts page |
| **Preconditions** | At least one boost booking exists on the system. The admin is logged in |
| **Postconditions** | Admin is shown a table containing all of the boost bookings that are stored on the system |
| **Main Success Scenario** | 1. Admin navigates to the Boosts page  2. Admin is shown a table containing every boost booking saved on the system |
| **Extensions** | N/A |
| **Priority** | Low – Third Increment |

|  |  |
| --- | --- |
| **ID** | UC20 |
| **Name** | View all accounts |
| **Description** | Admin can view a list of all account on the system |
| **Actor(s)** | Admin |
| **Trigger** | Admin navigates to the Accounts page |
| **Preconditions** | At least one account exists on the system. The admin is logged in |
| **Postconditions** | Admin is shown a table containing all of the accounts that are currently stored on the system |
| **Main Success Scenario** | 1. Admin navigates to the Account page  2. Admin is shown a table containing every account saved on the system |
| **Extensions** | N/A |
| **Priority** | Low – Third Increment |

|  |  |
| --- | --- |
| **ID** | UC21 |
| **Name** | Add boost option |
| **Description** | Admin can add a new boost option that will be available to be booked |
| **Actor(s)** | Admin |
| **Trigger** | Admin clicks Add Boost button on the admin home page |
| **Preconditions** | There is not a boost option already in the system with the same details as the one being created. The admin is logged in |
| **Postconditions** | The new boost option is added to the system and is now available to be booked. The admin is logged in |
| **Main Success Scenario** | 1. Admin clicks Add Boost option on the admin home page  2. Admin selects the type of boost to be added, dungeon/raid  3. Admin fills in the details required for the boost  4. Admin clicks Save button  5. A new boost option is saved to the system  6. Admin is notified that the new boost option was crated successfully |
| **Extensions** | 3a. The boost ID that has been entered is invalid 6a. Admin is notified that there is a problem with the ID they have entered |
| **Priority** | High – First Increment |

|  |  |
| --- | --- |
| **ID** | UC22 |
| **Name** | Remove boost option |
| **Description** | Admin can remove an existing boost option |
| **Actor(s)** | Admin |
| **Trigger** | Admin selects the boost option they want to remove from a list of all available boosts on the home page |
| **Preconditions** | A boost option exists. The admin is logged in |
| **Postconditions** | The boost option selected has been removed from the system and is no longer available to be booked. The admin is notified that the option was removed successfully |
| **Main Success Scenario** | 1. Admin selects the boost option they want to remove  2. Admin is shown a popup containing the details of the selected boost option  3. Admin clicks Delete boost button at the bottom of the popup  4. Admin is asked for confirmation that they want to delete the selected Boost option  5. The selected boost option is removed from the system and no longer available to be booked  6. Admin is notified that the option was deleted successfully |
| **Extensions** | 3a. There are boost bookings that have not happened yet for the selected boost option  6. Admin is notified that the selected option cannot be deleted due to there being future boosts already booked for the selected option |
| **Priority** | High – First Increment |

**Appendix 4**  
Use cases for all of the functional requirements.

**Usability**

|  |  |
| --- | --- |
| **ID** | NFR01 |
| **Requirement** | Users of the existing system should require no additional training to utilise the new system effectively |
| **Description** | The new system will have many of the same features as the current system as well as some new ones. However the new system and features should feel familiar to the user and navigation through the system should be self-explanatory |
| **Use case(s) link** | N/A |

|  |  |
| --- | --- |
| **ID** | NFR02 |
| **Requirement** | An advertiser should be able to complete a booking and receive confirmation the booking was successful in less than 60 seconds |
| **Description** | From the time when an advertiser logs in to add a booking to the new system, within 60 seconds they should be able to complete the booking for the buyer and provide them confirmation that they’re booking was successful. Although the speed of the database may be a factor in this requirement, it will have negligible impact on the speed at which the advertiser can complete the task when compared against the impact that the layout of the booking page and the navigation through the system will have. |
| **Use case(s) link** | UC05, UC06 |

**Reliability**

|  |  |
| --- | --- |
| **ID** | NFR03 |
| **Requirement** | The system must be operational for all users 99.7% of the time |
| **Description** | The users of the system must be able to utilise all features of the system 99% of the time. Any loss of access for any user due to a system failure fits the criteria for the system being unavailable. |
| **Use case(s) link** | All use cases |

|  |  |
| --- | --- |
| **ID** | NFR04 |
| **Requirement** | The system must be available for use by the admin 99.7% of the time |
| **Description** | The features that are only available to the admin should be available at all times even if the rest of the system is inaccessible to all other users. |
| **Use case(s) link** | UC15, UC16, UC17, UC18, UC19, UC20, UC21, UC22 |

|  |  |
| --- | --- |
| **ID** | NFR05 |
| **Requirement** | The system should have an average failure rate no higher than 1 per month |
| **Description** | A failure is any loss of access that prevents a user from accessing the system, or prevents the system from communicating with the database. |
| **Use case(s) link** | N/A |

|  |  |
| --- | --- |
| **ID** | NFR06 |
| **Requirement** | All failures should be fixed with the system restored back to its original functionality within 24 hours |
| **Description** | A failure is only declared fixed once all users are able to utilise the full functionality of the system. A 24 hour window allows for failures that occur during the night to be handled the following day whilst still giving adequate time to fix the failure. |
| **Use case(s) link** | N/A |

|  |  |
| --- | --- |
| **ID** | NFR07 |
| **Requirement** | Numeric values and calculations must be accurate to the nearest whole number |
| **Description** | Due to the system primarily handling numerical values and making calculations on those values, all results must be accurate and then truncated to a whole number. Because the values represent an in game currency they cannot be rounded up as that would gradually result in the system showing that there is more of the currency available than there actually is. |
| **Use case(s) link** | N/A |

**Performance**

|  |  |
| --- | --- |
| **ID** | NFR08 |
| **Requirement** | The system should allow up to 20,000 users to sign up |
| **Description** | The database must be able to store the details of up to 20,00 users |
| **Use case(s) link** | UC01 |

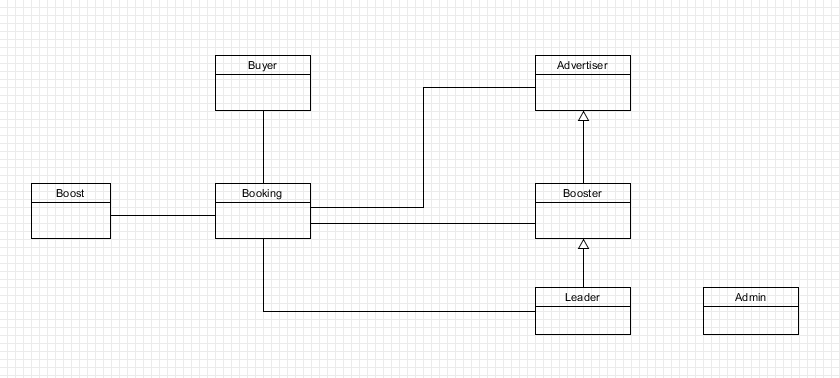
|  |  |
| --- | --- |
| **ID** | NFR09 |
| **Requirement** | The response time when querying the database should be less than 500ms |
| **Description** | Every query call made to the database should have a response time no higher than 500ms |
| **Use case(s) link** | N/A |

|  |  |
| --- | --- |
| **ID** | NFR10 |
| **Requirement** | The system should be able to handle up to 50 transactions per second |
| **Description** | Due to the high volume of users that will be accessing the system and utilising the booking features, the system must be able to handle every transaction comfortably and cope with peak times. 50 transactions per second is a reasonable maximum based on the number of users and the frequency of the transactions each user will make. |
| **Use case(s) link** | N/A |

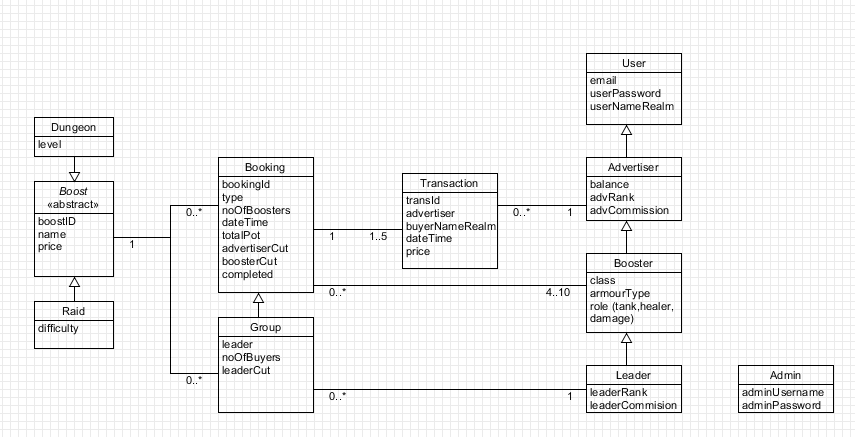
**Supportability**

|  |  |
| --- | --- |
| **ID** | NFR11 |
| **Requirement** | The capacity for the number of users should be increased by 10,000 within 2 days of the decision to increase the capacity being made |
| **Description** | The system is one that deals with an expanding user base so the beginning capacity of 20,000 users will need to be increased in the future. The system should be capable of having its capacity increased by 10,000 users within 2 days of beginning the task to increase the capacity. |
| **Use case(s) link** | UC01 |

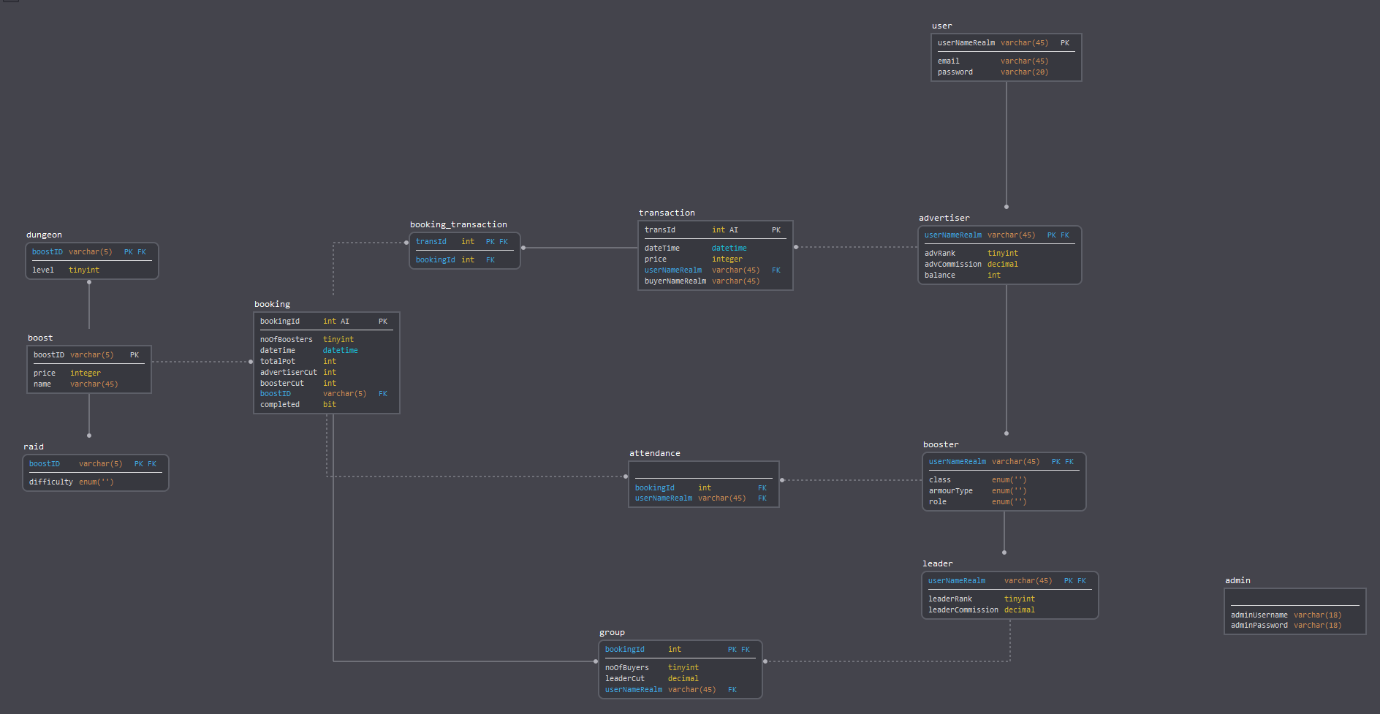
**Appendix 5**  
All of the non-functional requirements modelled as declarative requirements.



**Appendix 6**  
Conceptual entity relationship diagram for my system drawn using UMLet.

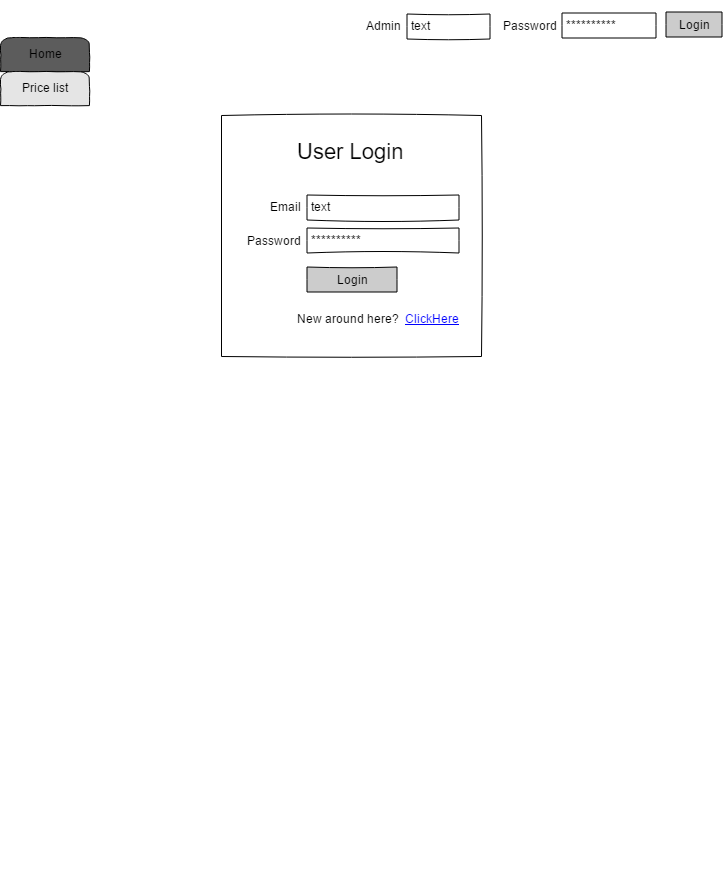


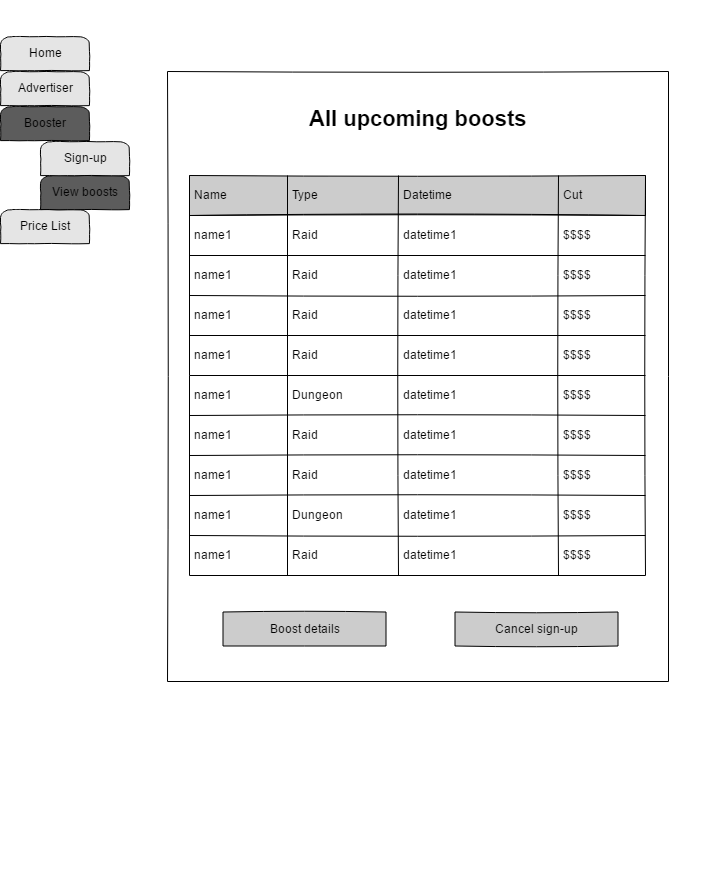
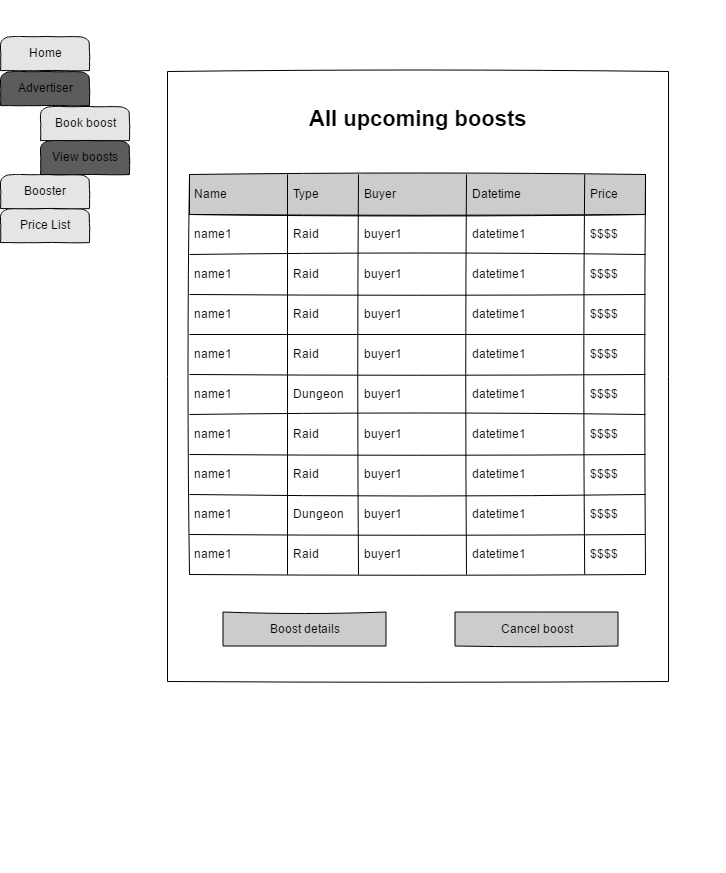
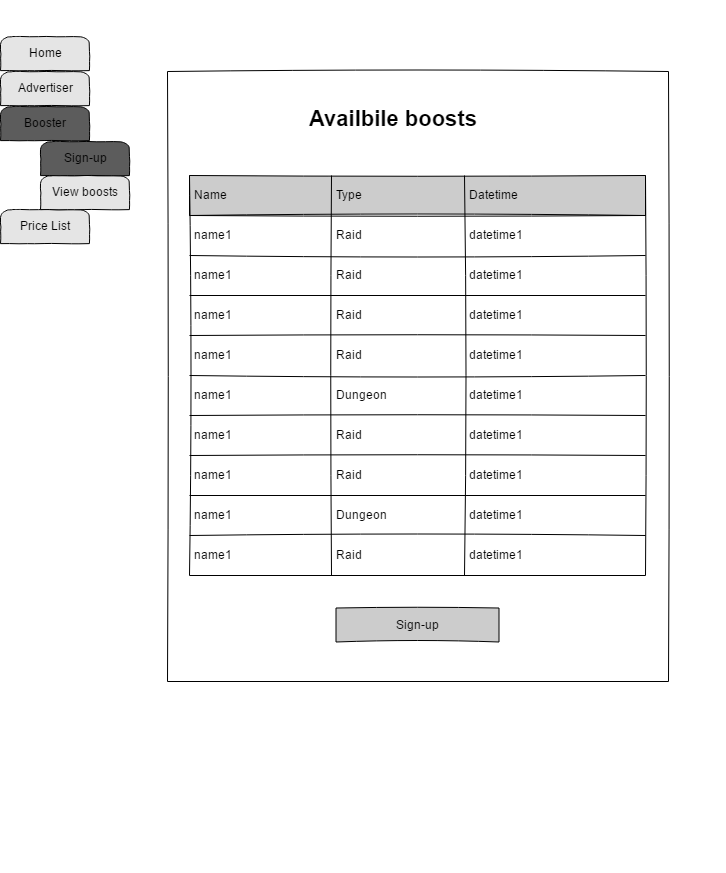
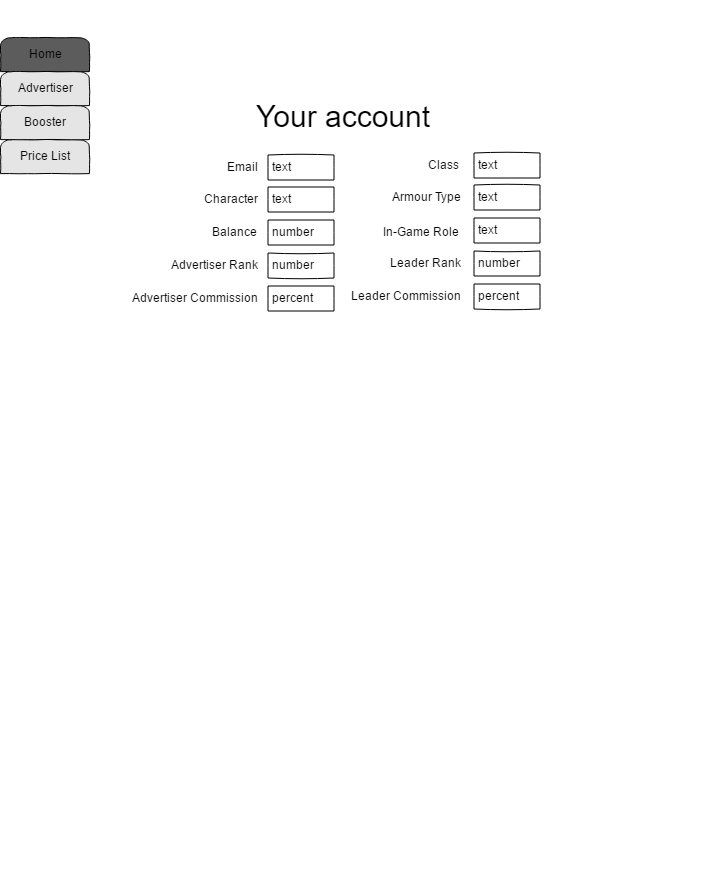
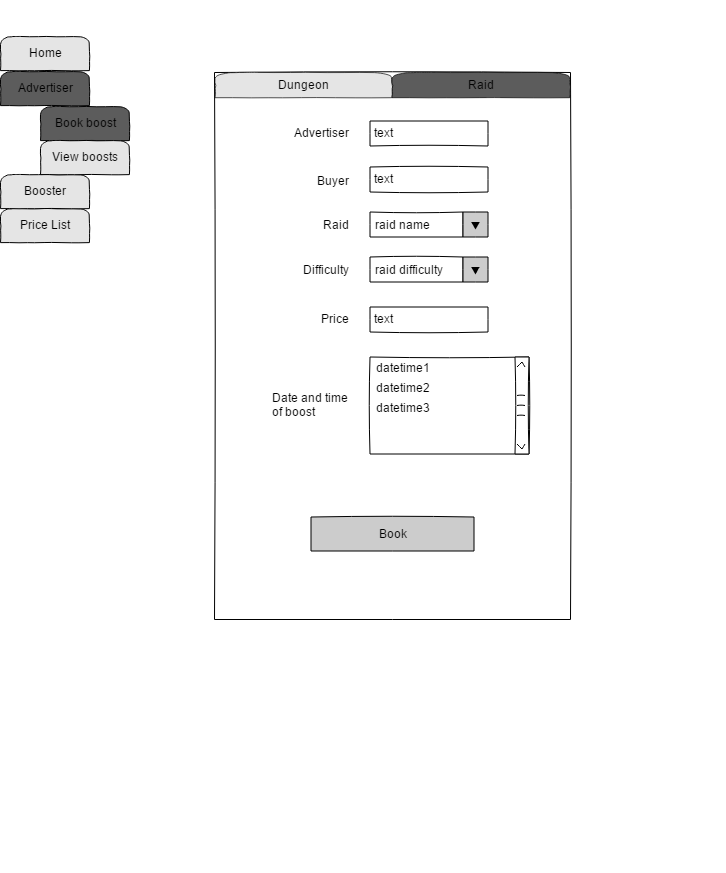
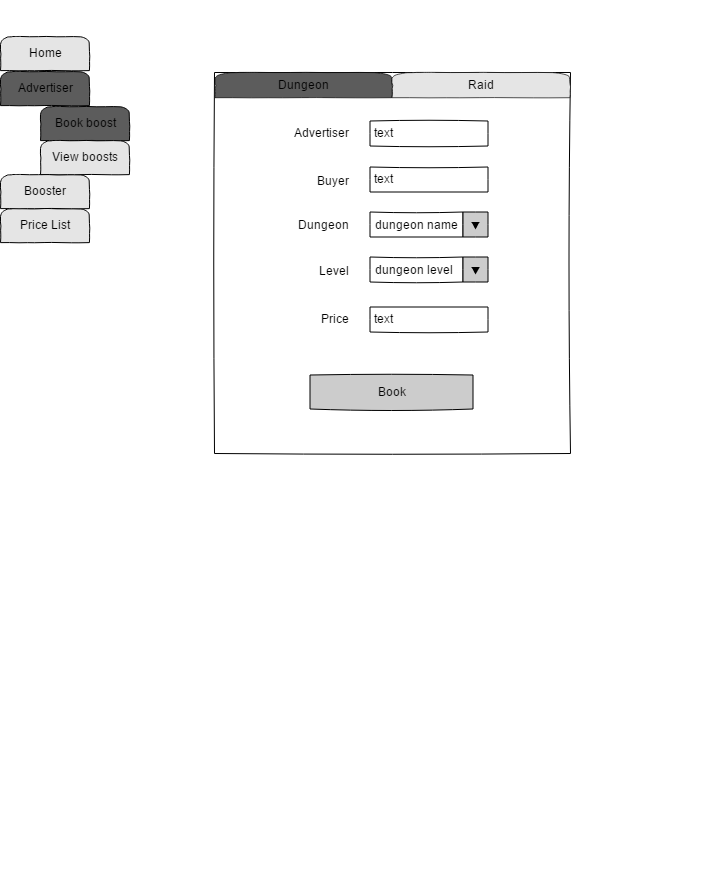
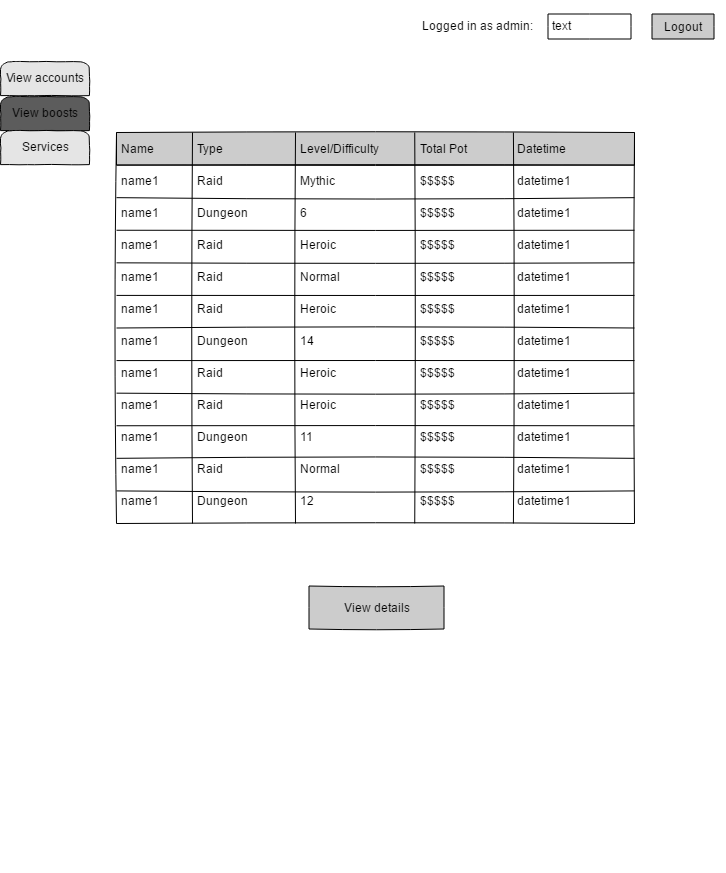
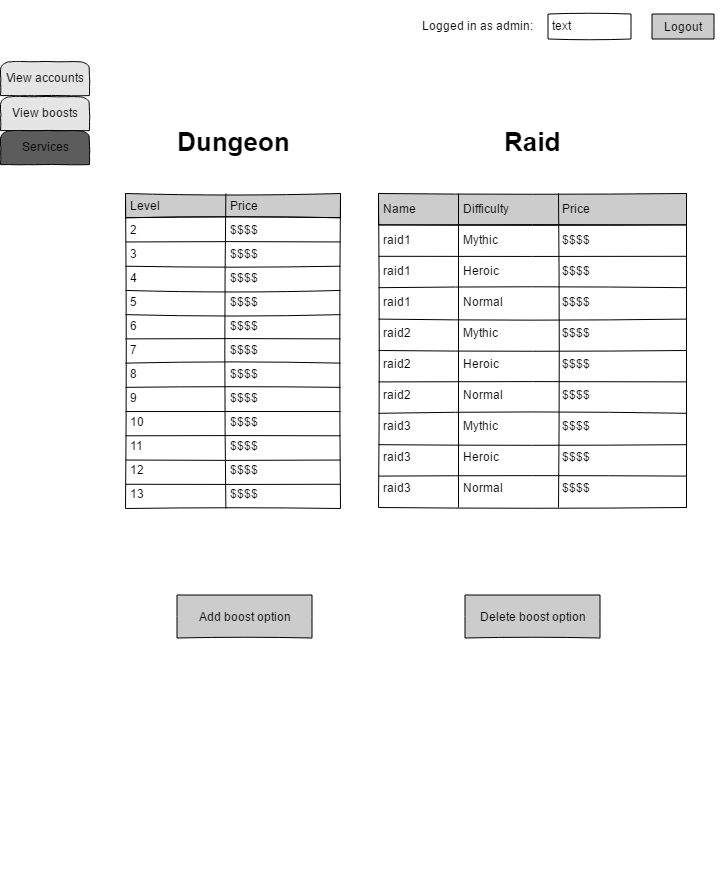
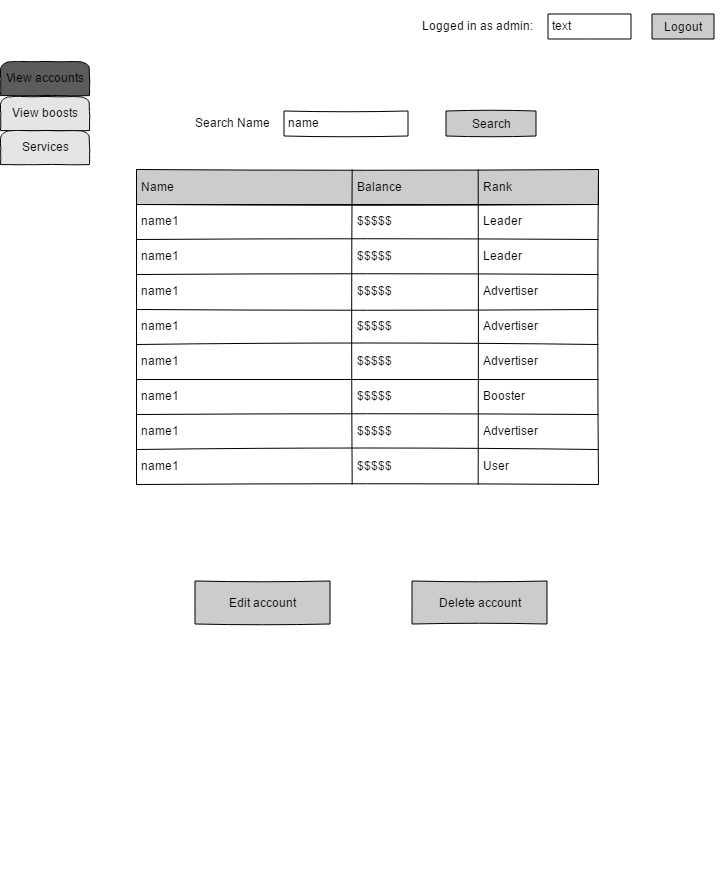
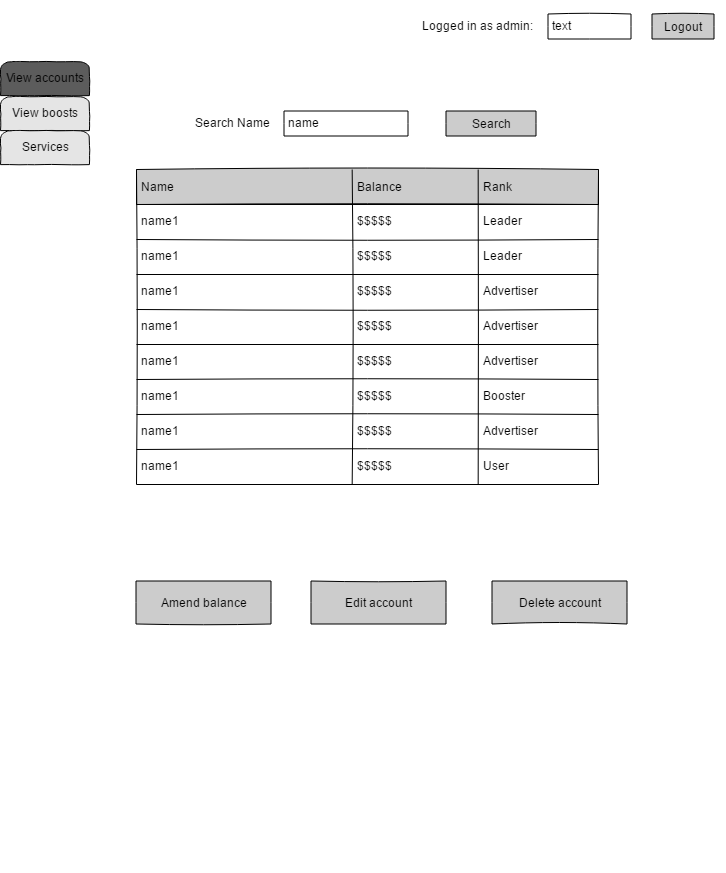
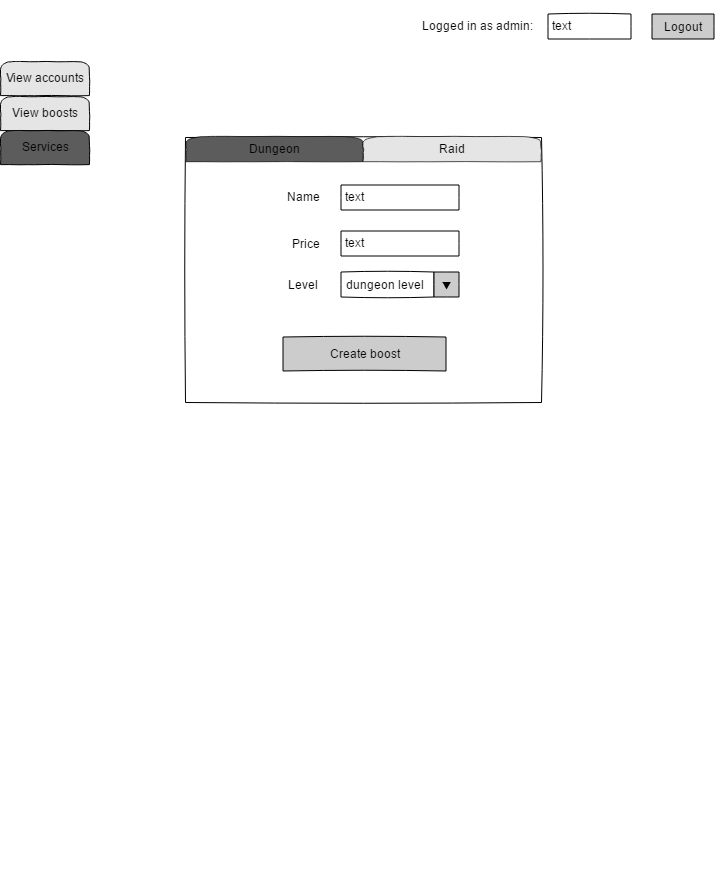
**Appendix 7**Logical entity relationship diagram for my system drawn using UMLet.



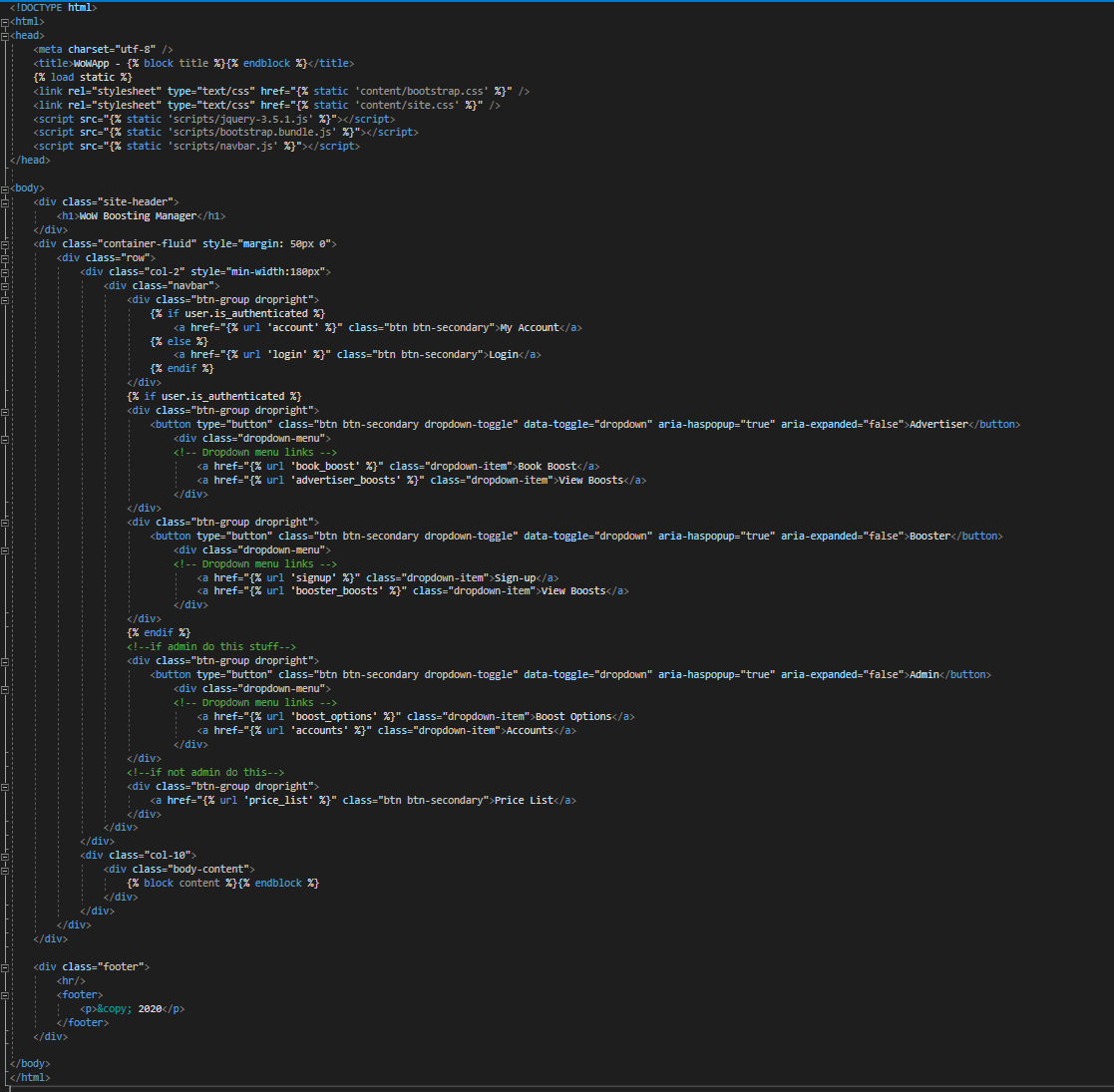
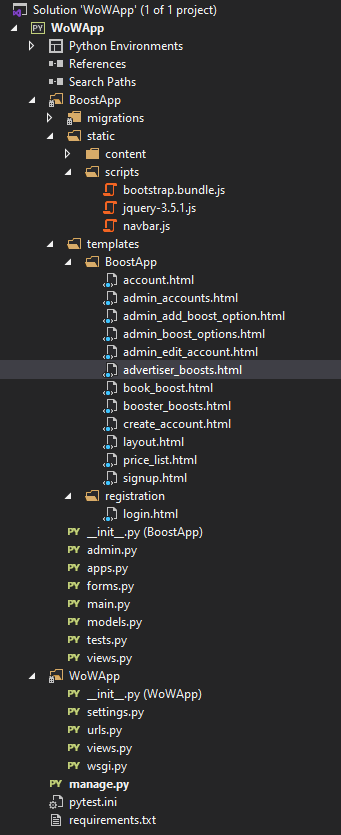
**Appendix 8**  
Database design for my system drawn using the online tool SqlDBM (SqlDBM, 2020).



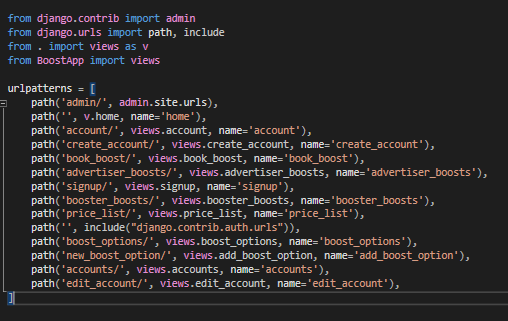




**Appendix 9**  
Wireframes for all of the screens that make up my web application.

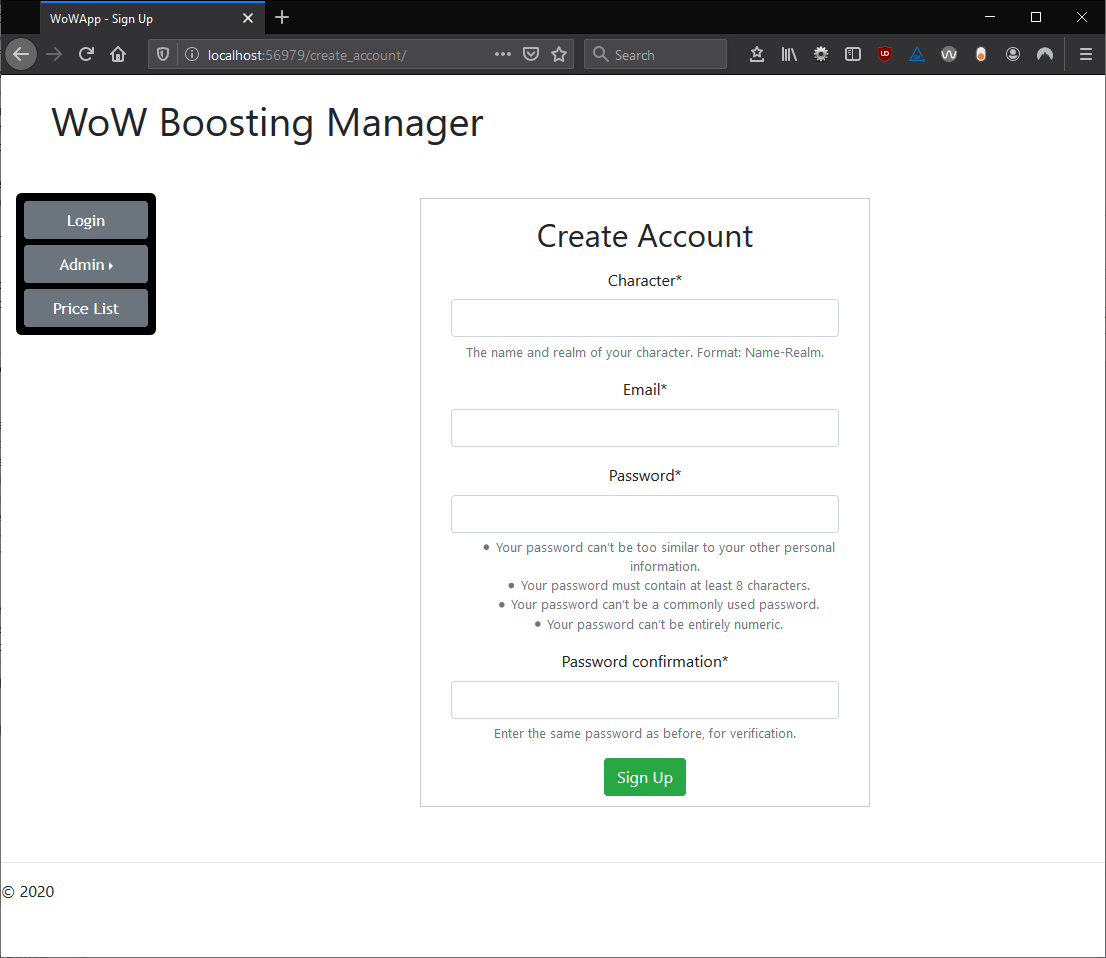


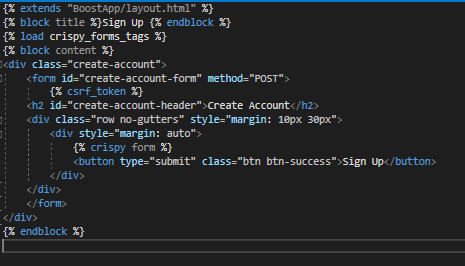
**Appendix 10**  
File structure of my application (left) and the contents of the ‘layout.html’ file (right) that I used as a base for all of the screens.



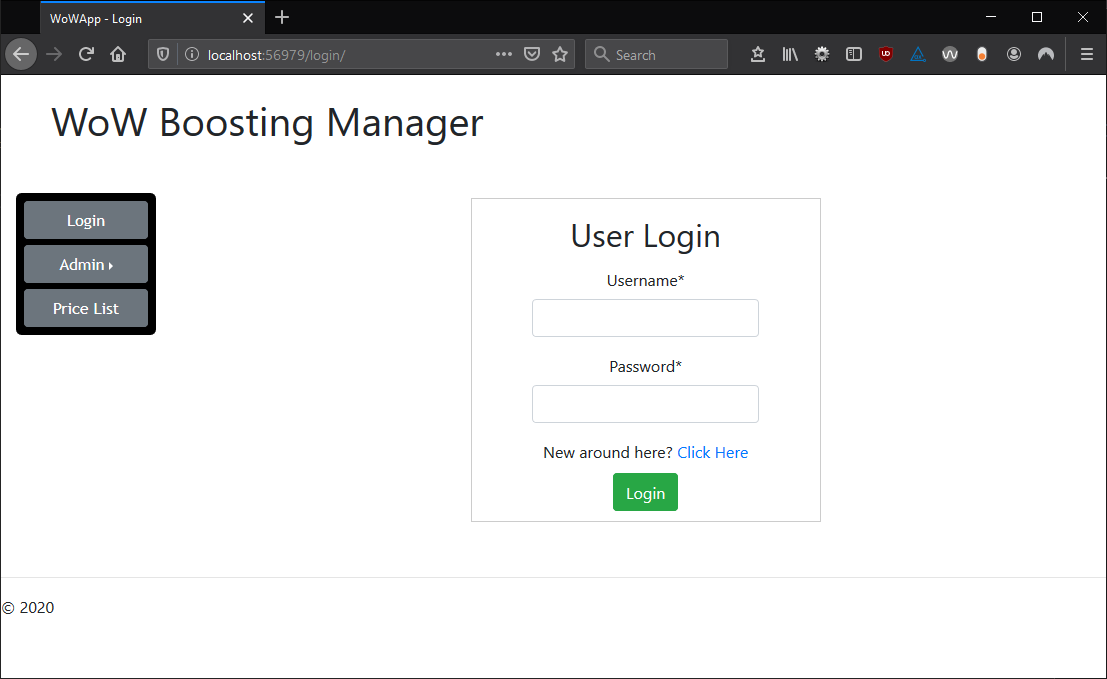


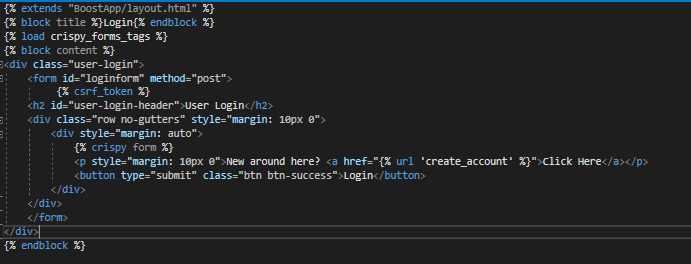
**Appendix 11**  
Contents of the ‘urls.py’ file (previous page) and the ‘views.py’ file (above).

****

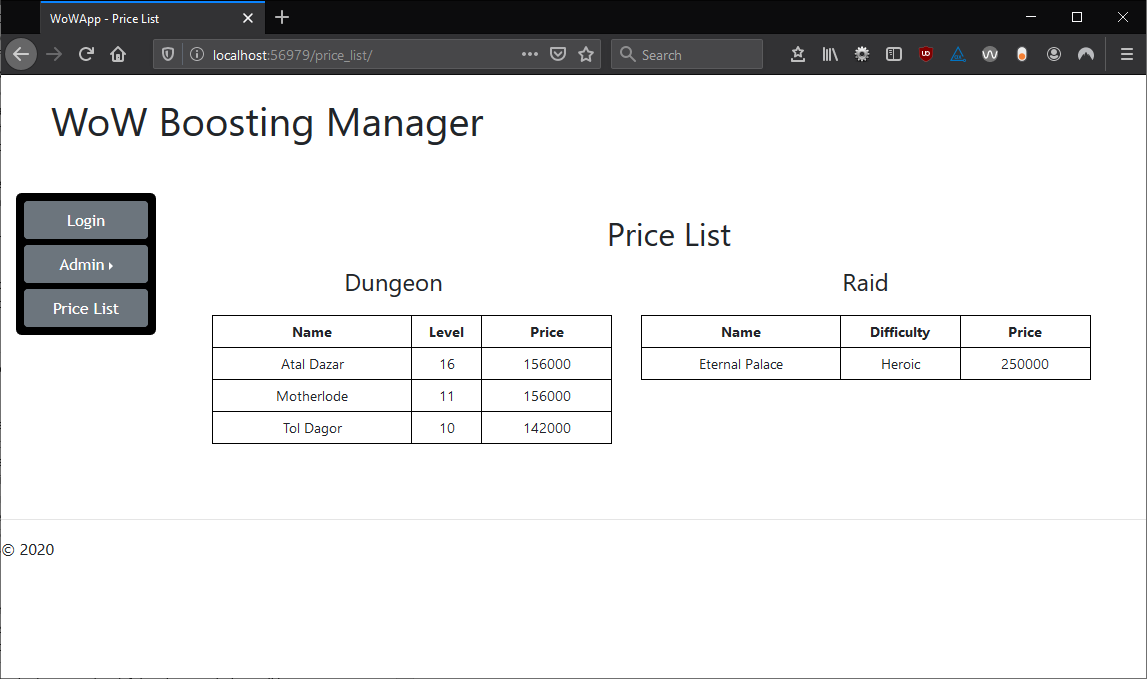
****

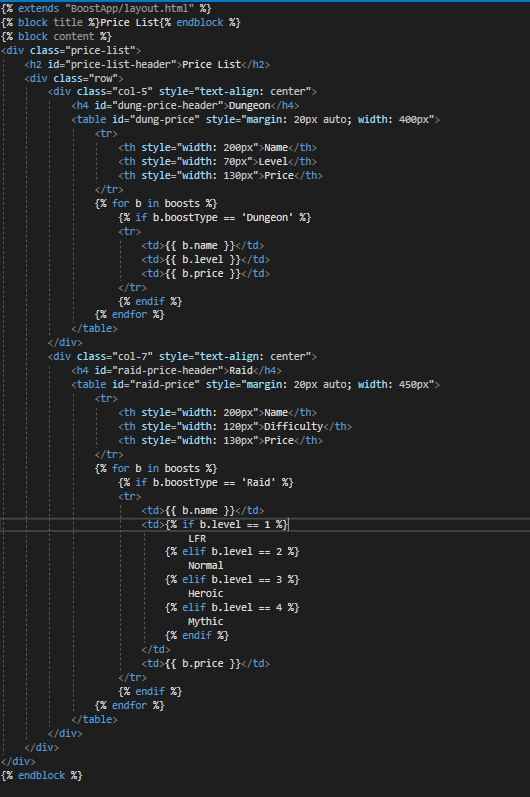
**Appendix 12**  
Create account page (top) and the contents of the ‘create\_account.html’ file (bottom).



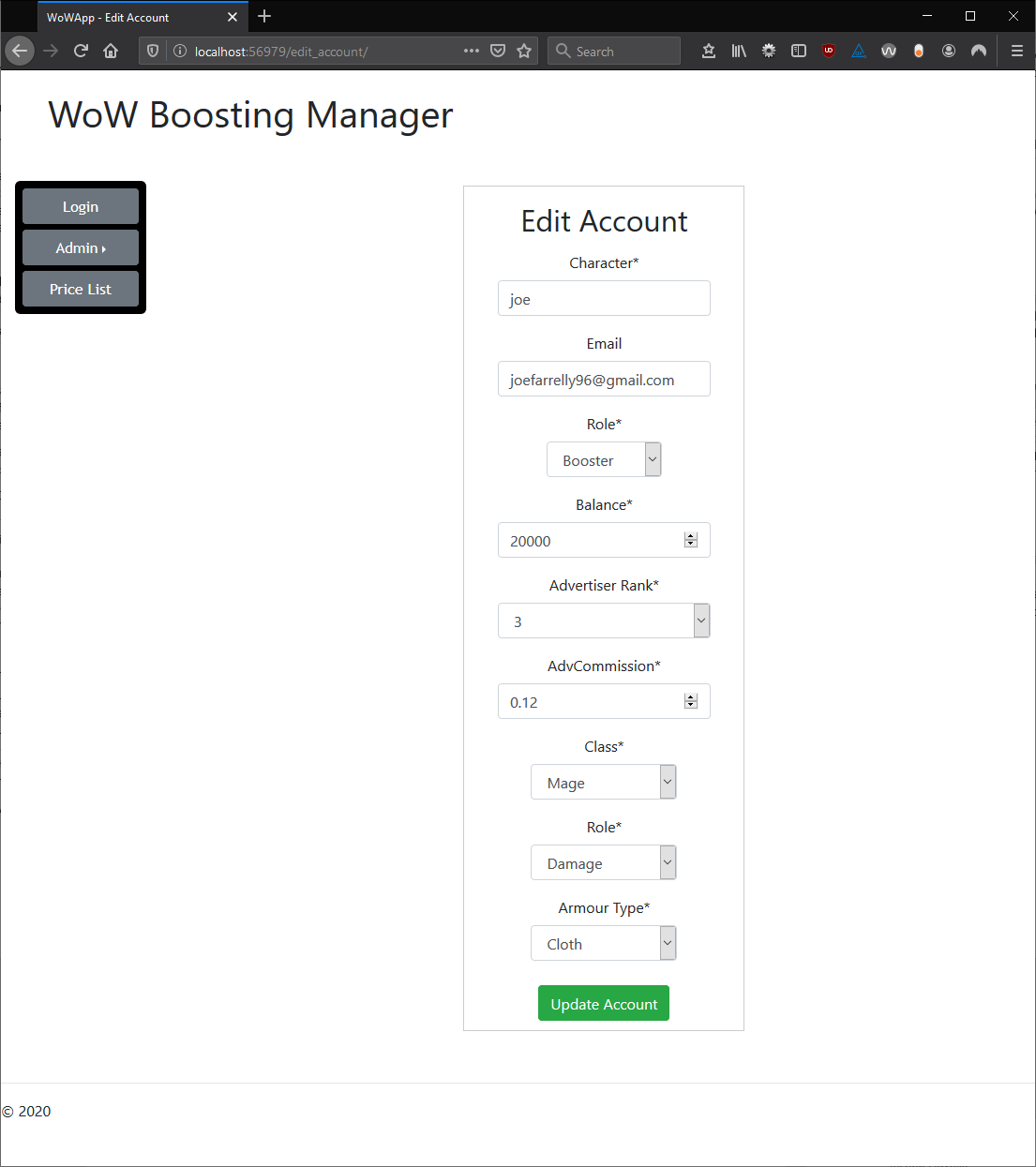


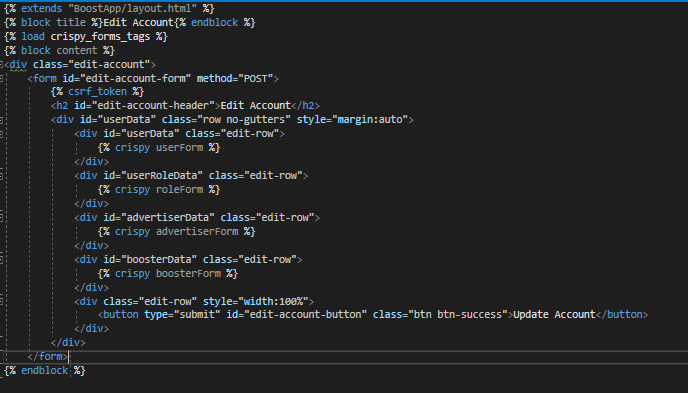
**Appendix 13**  
Login page (top) and the contents of the ‘login.html’ file (bottom).

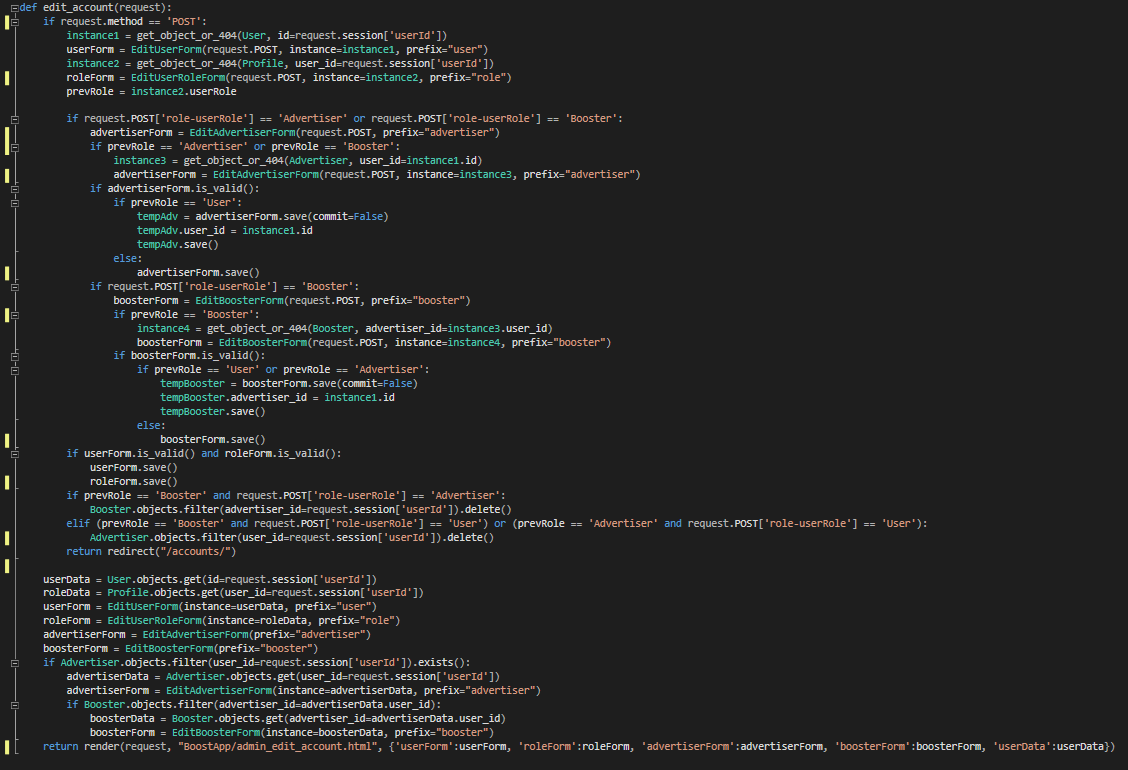




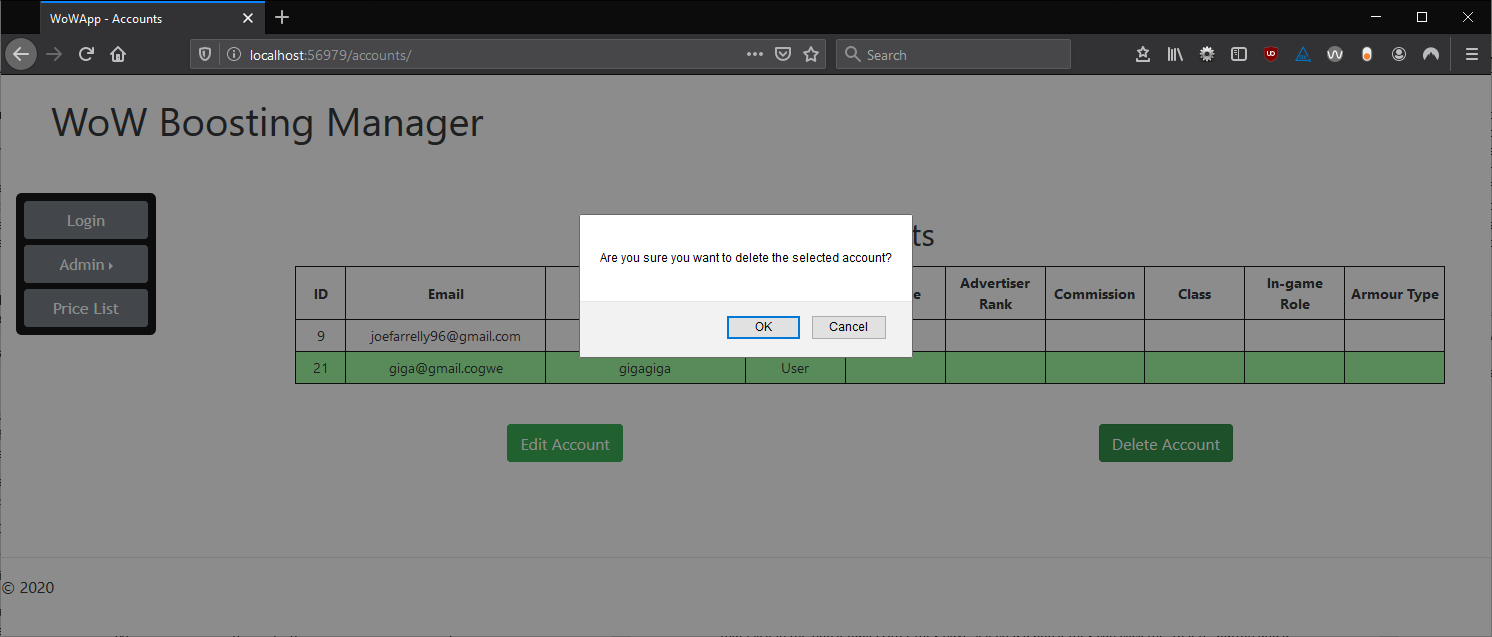
**Appendix 14**  
Price list page (previous page) and the contents of the ‘price\_list.html’ file (above).

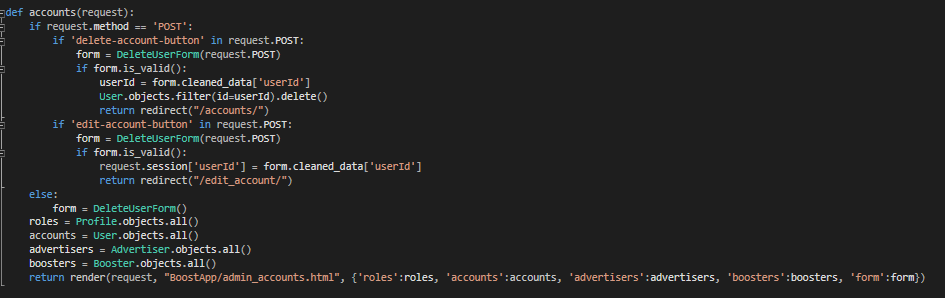




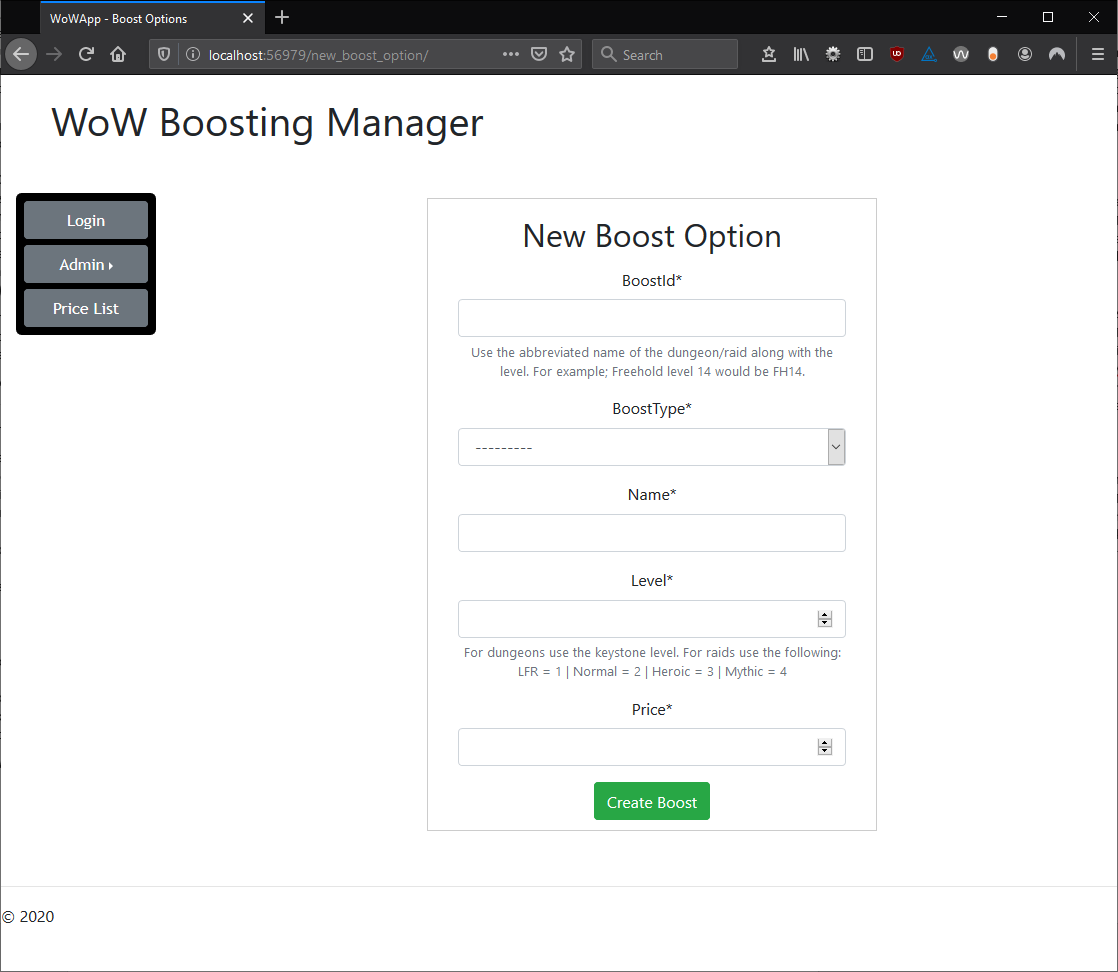
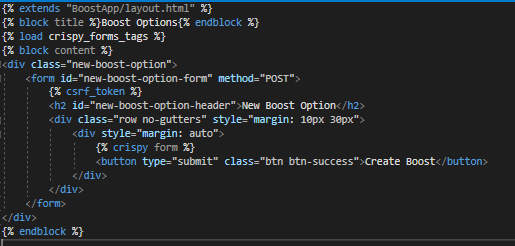


**Appendix 15**  
Edit account page (previous page), contents of the ‘admin\_edit\_account.html’ page (top) and the code inside the file ‘views.py’ responsible for the functionality of editing an account (bottom).

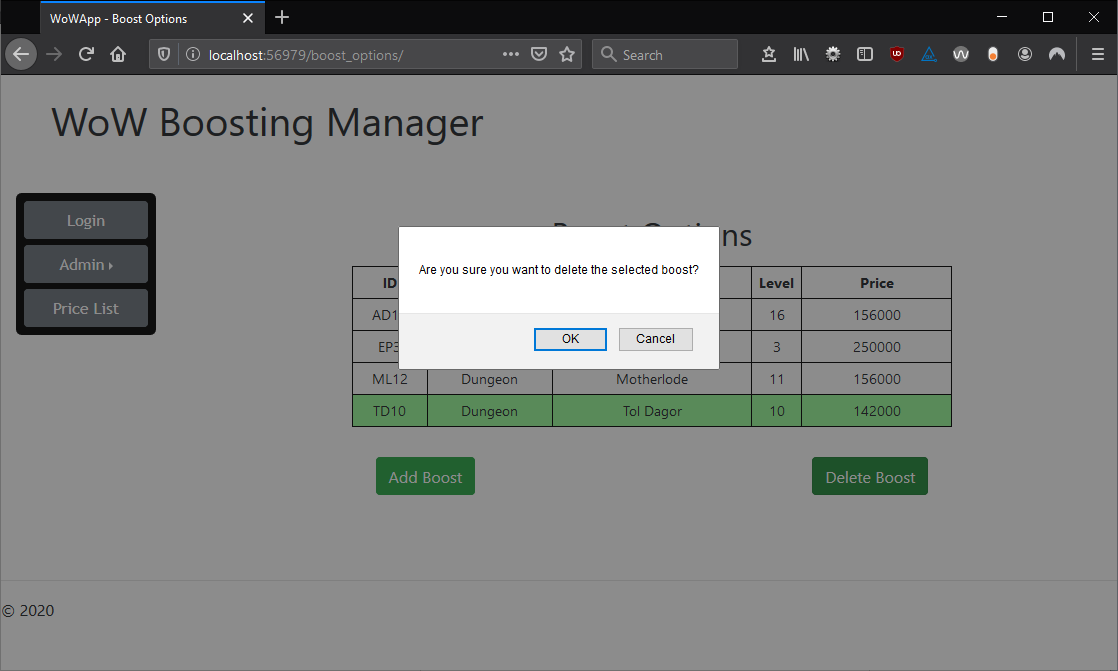


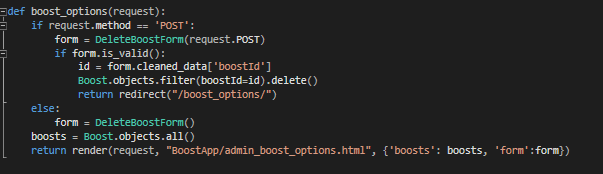


**Appendix 16**  
All accounts page (top) and the code inside the file ‘views.py’ that is responsible for deleting an account (bottom).

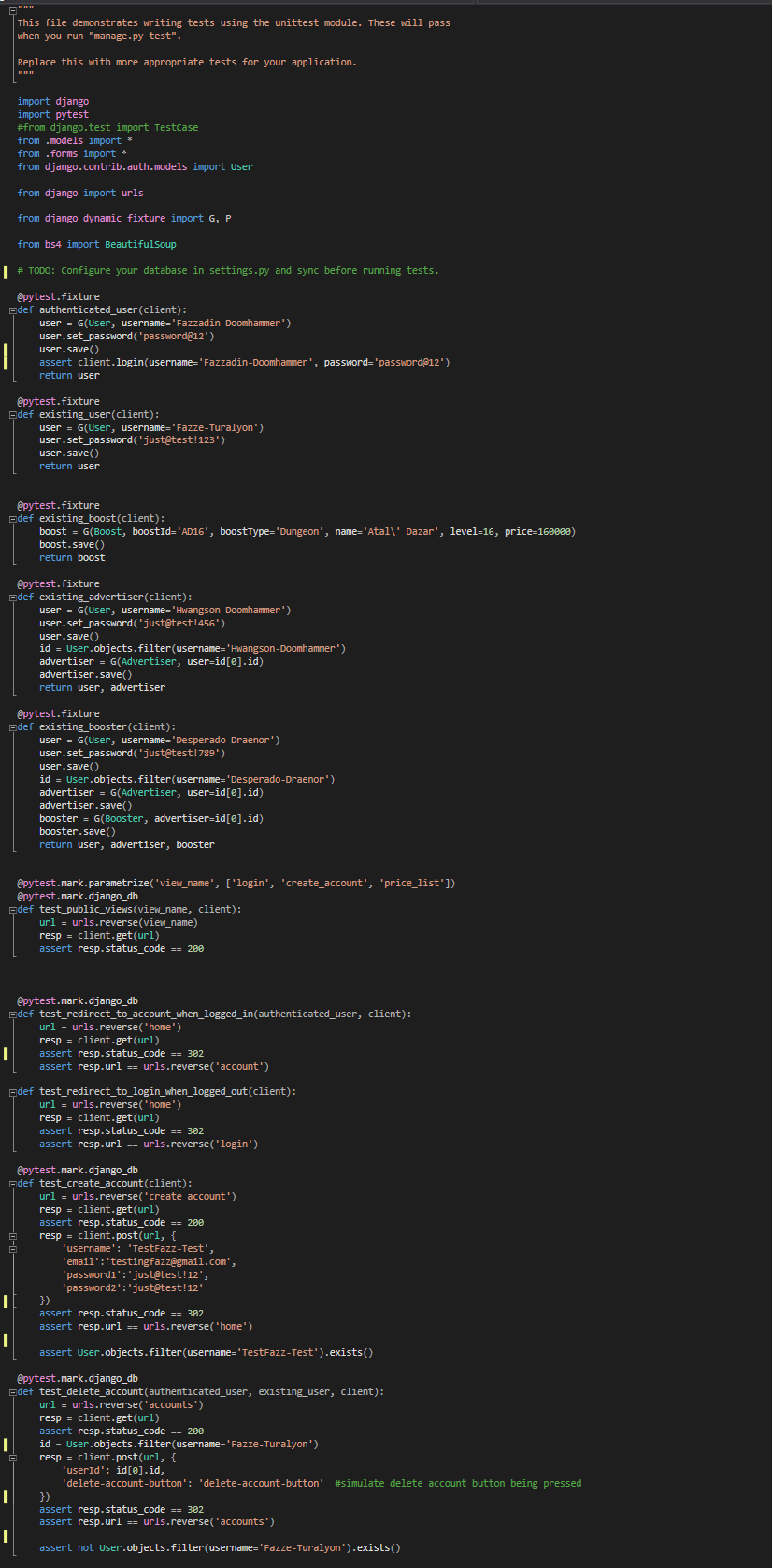


**Appendix 17**  
Create new boost option page (top) and the contents of the ‘admin\_add\_boost\_option.html’ file (bottom).



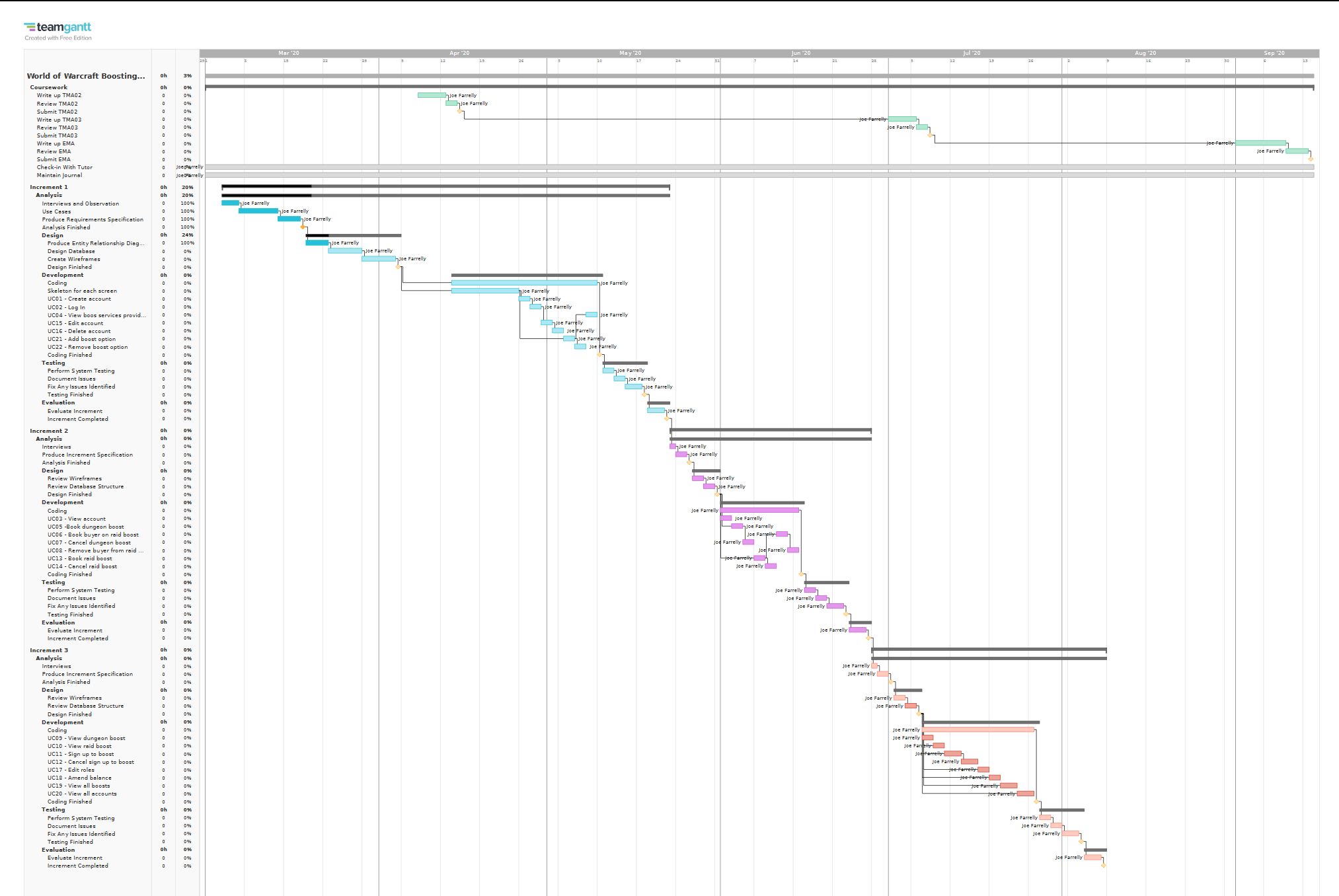


**Appendix 18**  
All boosts options page (top) and the code inside the file ‘views.py’ that is responsible for deleting a boost option (bottom).

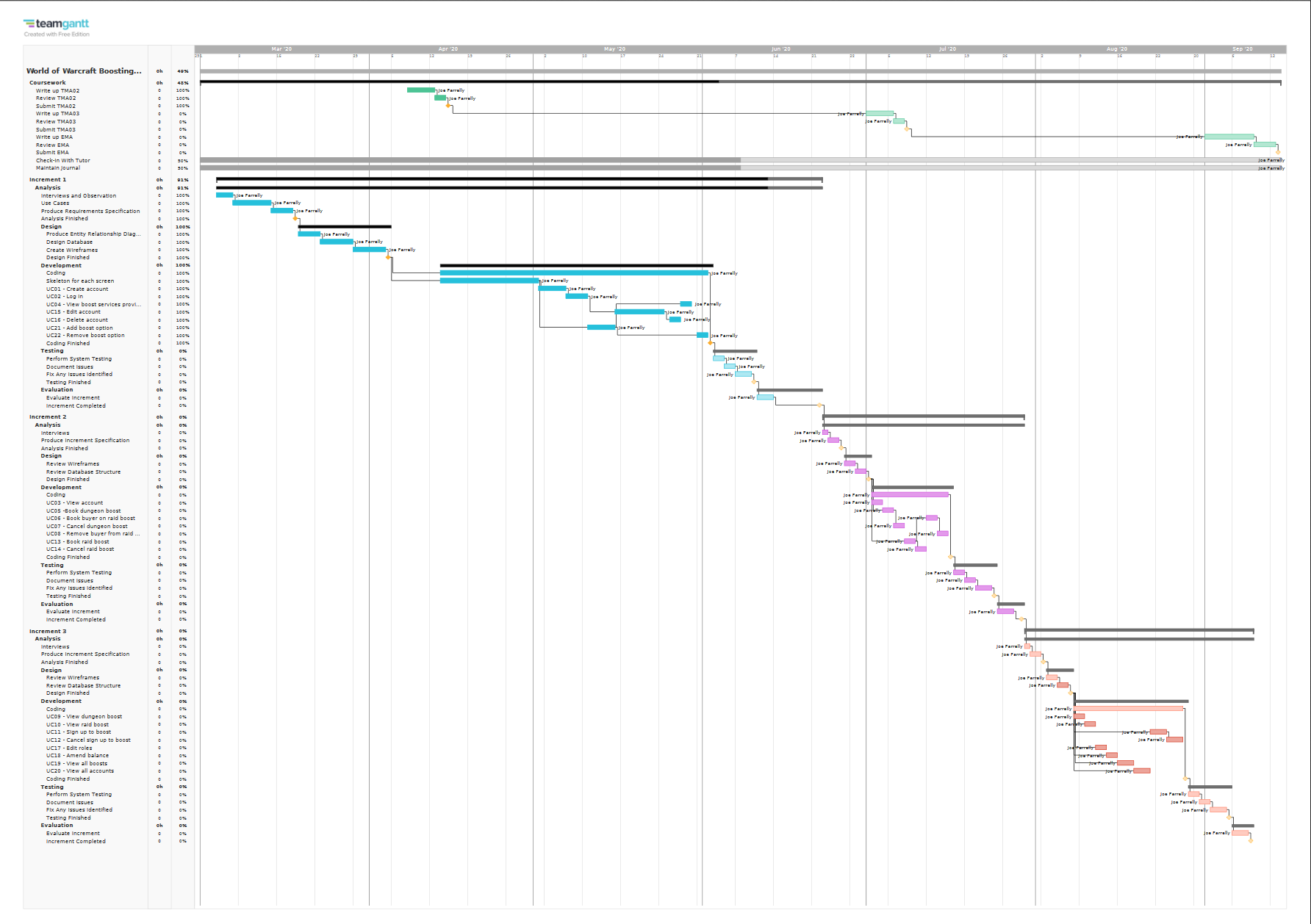




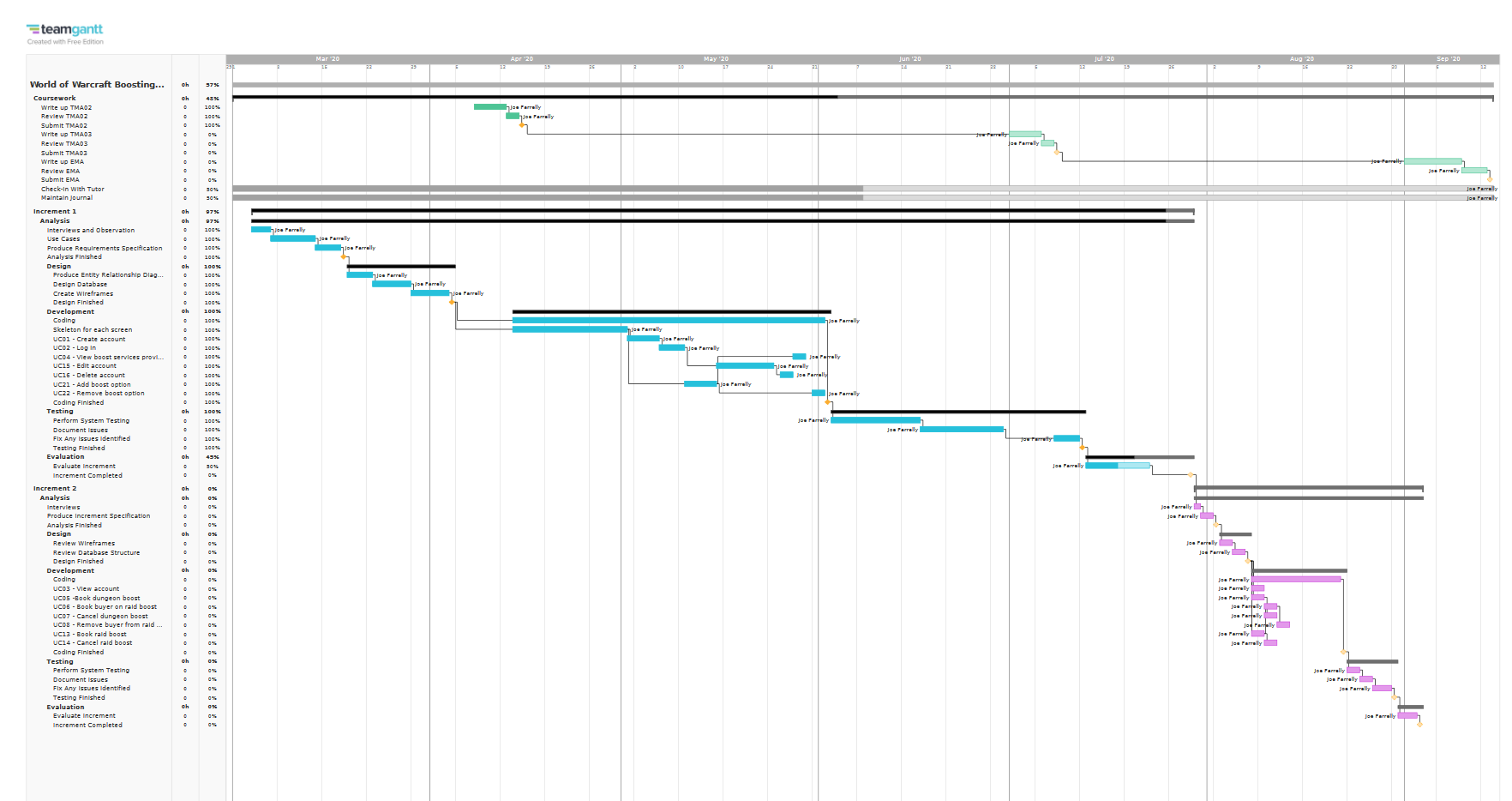
**Appendix 19**  
Contents of the file ‘tests.py’ which is responsible for all of the testing for my application.



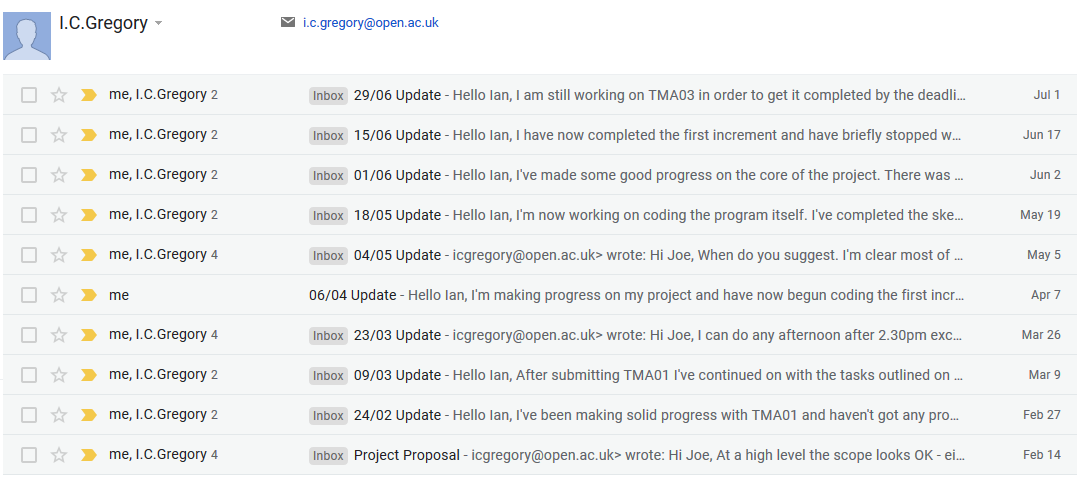
**Appendix 20**  
Iteration 2 of my Gantt graph.

****

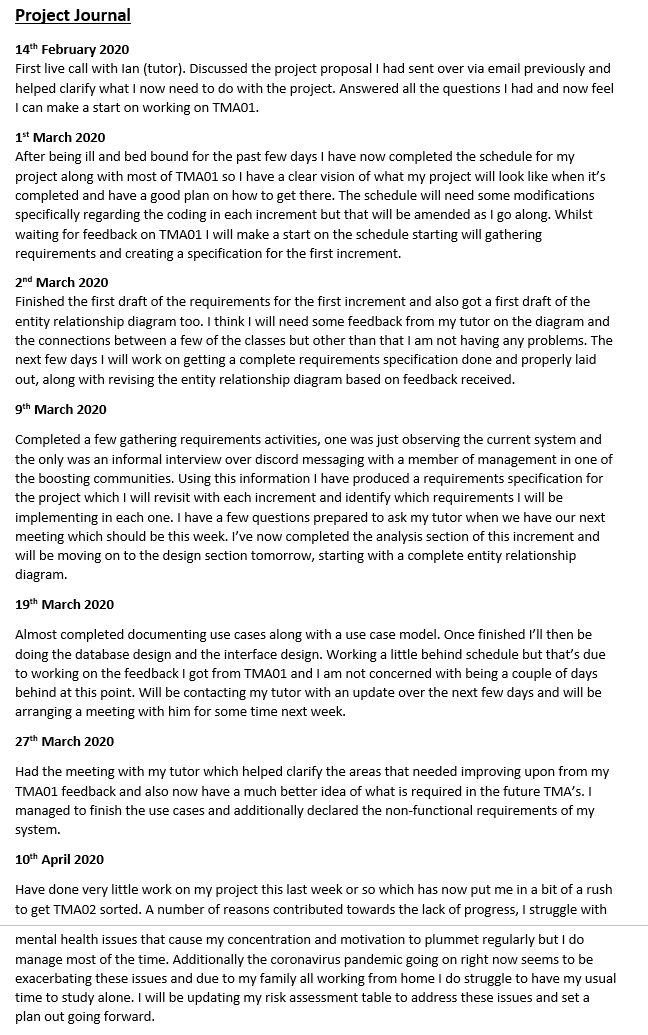
**Appendix 21**  
Iteration 3 of my Gantt graph.

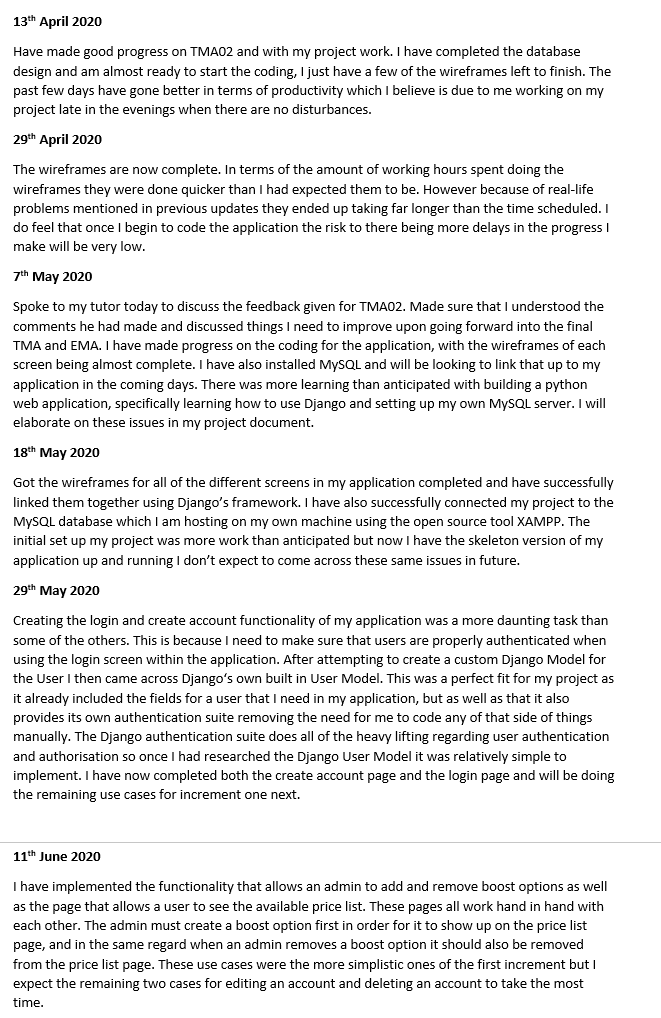
****

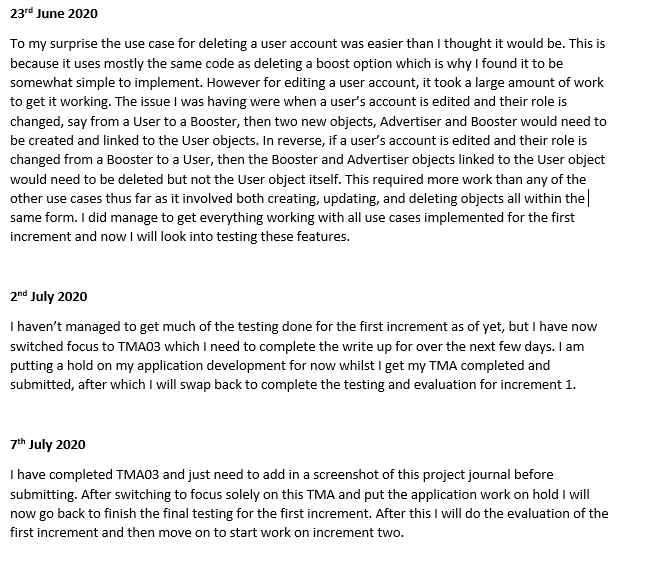
**Appendix 22**  
Iteration 4 of my Gantt graph.

****

**Appendix 23**  
Overview of the email correspondence between myself and my tutor.







**Appendix 24**  
Contents of my project journal document.