Software Development Planning and Risks

B098688 - s1671778

February 10, 2017

Contents

1 Introduction

This report will provide a series of guidelines on how to proceed with the creation of a web application. The web application's code has been initiated by an unknown party, that is written in `html`, `JavaScript`, `python` and uses the web framework `flask`. Said code must be enhanced such that it fulfills the base requirements for the project. The skeleton of the web application has a few elements already built in. It is, however, very rough and needs much work to become a feasible end product.

The first part of the report will briefly outline what is the intended project, and the background needed to understand what it is attempting to achieve. While this is being discussed, some of the important factors to keep in mind are highlighted. After this is done, several comments regarding issues in the provided code are made. Afterwards, solutions for most of these are proposed, and eventually presented in a *time/effort* estimation plan. Finally, once the project has been understood, and its purpose is clear, risk analysis and management strategy are posed. For the latter, a Gantt chart is put together to act as a guide for the next steps: *UI and Evaluation Plan* and *Refactoring and Reflection*.

The management strategy in this small project will help make a successful end product. There are many factors to be analysed in order to properly develop the software development planning. Considering personal inexperience in web security, `flask`, `html`, and `JavaScript`, it is safe to assume that many issues with the program will be overlooked at the beginning. This means that the problems mentioned in this document will be a fraction of those that will actually be fixed in the later stages of the project. This will require allocating time to research on the aforementioned topics as well as choosing carefully the best development model. After careful consideration and taking into account that this project will need to cycle from the current code to an improved version a few times, the *Waterfall with Subprojects* development model is going to be used.

Having mentioned how the lack of experience will impact the development model required, it is important to analyze how it will modify other elements of the project's progress. The best way to do this is most likely through the Gantt chart. In it, tasks that have to be completed and are related to areas of inexperience will need some time allocation for 'research' or 'practice', but this will decrease as more tasks of the same area are performed, i.e. account for knowledge growth.

2 Project Specifications

The goal is to further develop a web based application that allows the user to generate a squad (Warband) for a tabletop game - with certain constraints. The application must allow users to create and edit squads with an unchangeable name, no more than 10 members, and no fewer than 1. The user begins with a Captain, that can become better with time, and has a limited amount of money to hire other regular team members, that cannot improve with time, and a special member, the Ensign, who, like the Captain, can evolve in time.

The idea of the web application is to check that all constraints are fulfilled for a user's Warband, and that if one or more are violated, a warning message is displayed such that the user is able to modify their decision. The base stats for each team member are shown in the Annex.

2.1 New Warband

A new Warband begins with **one** Captain and **500** credits. For a new Warband to be created, the following conditions must be met:

1. Warband Name.
2. Player Name.
3. Positive number of credits.
4. Captain:
    (a) **ONE** per Warband.
    (b) Starts with **one** EXPERIENCE.
    (c) Must be assigned **one** SPECIALISM (Engineering, Psychology, Marksman, Tactics, Melee, Defence).
    (d) **One** Associated Skill must be assigned to the specified SPECIALISM:
        i. Engineering: [Repair, Sabotage, Augment].
        ii. Psychology: [Bolster, Terror, Counter].
        iii. Marksman: [Aim, Pierce, Reload].
        iv. Tactics: [Squad, Ambush, Surround].
        v. Melee: [Block, Riposte, Dual].
        vi. Defence: [Shield, Sacrifice, Resolute].
    (e) Must be given **one** WEAPONS/EQUIPMENT:
        i. Blaster: **5** credits.
        ii. Needle Gun: **12** credits.
        iii. Blade: **3** credits.
        iv. Cannon: **15** credits.
        v. Whip: **5** credits.

5. Ensign:
    (a) Price of hiring: **250** credits.
    (b) Maximum **ONE** (Warbands can be created without an Ensign).
    (c) Must be assigned **one** SPECIALISM.
    (d) Must be assigned **one** Associated Skill.
    (e) Must be given **one** WEAPONS/EQUIPMENT.
6. The total number of members is **10** (including the Captain and Ensign).
7. Prices of hiring are:
    (a) Augment Gorilla: **20** credits.
    (b) Lackey: **20** credits.
    (c) Security: **80** credits.
    (d) Engineer: **60** credits.
    (e) Medic: **50** credits.
    (f) Commando: **100** credits.
    (g) Combat Droid: **150** credits.

2.2 Edit Warband

When editing a Warband there is also a set of conditions that have to be met. They are outlined here.

1. Can hire squad members as long as the total number of squad members does not exceed **10**, and as long as the Warband's number of credits does not become negative.
2. Captain and Ensign can gain EXPERIENCE.
3. Acquired EXPERIENCE can be traded for upgrades for Captain and Ensign (**10** EXPERIENCE points can be turned into **1** point for a Stat up to specified maximum values shown in the Appendix section).
4. Can remove squad members, but their hiring price is **not** returned to the number of credits.
5. The Captain can have up to **two** weapons and **four** items.
6. The Ensign can have up to **one** weapon and **3** items.
7. Weapons cost credits. Changing or adding weapons **cannot** result in a negative number of credits.

3 Identified Problems

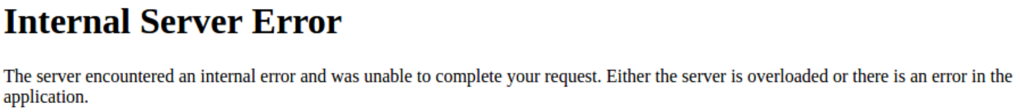
Now that the constraints of the teams have been covered, it is possible to go through the code to check missing or wrong information. But first, going through the current web application to see errors in its functioning is important. The following is a list of problems that are observed by working though the web application alone.

3.1 New Warband

1. The most obvious deficiency is the fact that Warbands are not linked to a specific player. This means that anyone can edit any Warband. The game must support more than one player, where only the team's owner can edit it, but all other players can see it.

2. There is no way to go from the “New Warband” or “Edit Warband” pages to the “Home” page without using browser back button or address bar.

3. There has been no attempt at testing the current product. For example, when attempting to generate an illegal “New Warband” (Captain without weapon / skill / specialism), an error message is given, but no explanation or way to return and fix the mistakes are provided.



Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

Generic error message.

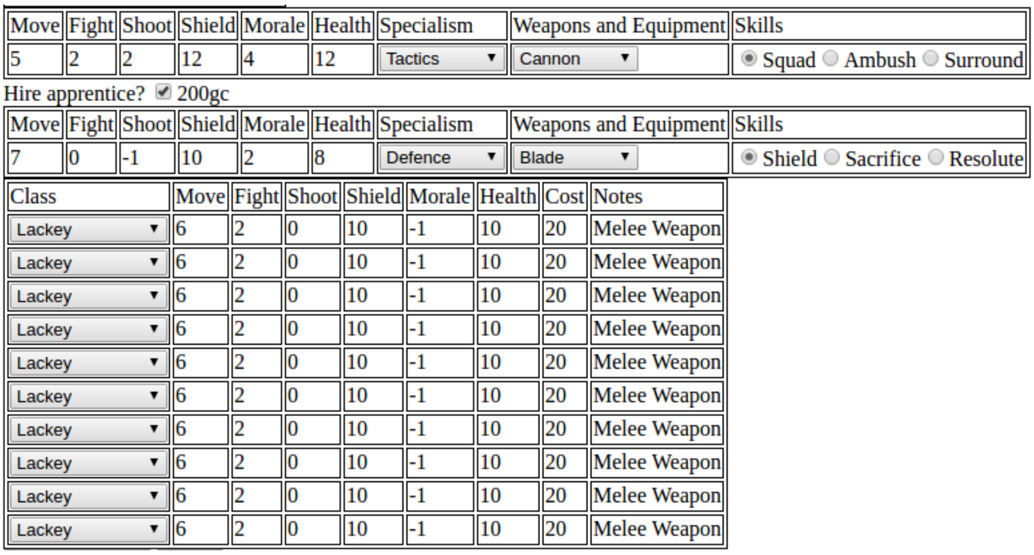
4. The price of weapons is not discounted from the number of credits from the screen when creating a “New Warband”.

5. Neither Captain nor Ensign have ”Notes“ section, and merged ”Weapons and Equipment“, whereas the two should be different.

6. The Ensign is called “Apprentice” for some reason - this is confusing.

7. The cost of the “Apprentice” is 200 credits instead of 250 as it should. This and the previous item are actually related, and comprise a larger problem - the information from each team member is actually stored in the `python` code and separately in the `html/JavaScript` code. Due to this, the price on screen is 200 credits, but it actually subtracts 250 credits (as it should).

8. A 10 row table allows the player to fill out the form with up to 12 members (returns the generic error message when done so).



Move	Fight	Shoot	Shield	Morale	Health	Specialism	Weapons and Equipment	Skills
5	2	2	12	4	12	Tactics ▾	Cannon ▾	● Squad ○ Ambush ○ Surround

Hire apprentice? ☑ 200gc

Move	Fight	Shoot	Shield	Morale	Health	Specialism	Weapons and Equipment	Skills
7	0	-1	10	2	8	Defence ▾	Blade ▾	● Shield ○ Sacrifice ○ Resolute

Class	Move	Fight	Shoot	Shield	Morale	Health	Cost	Notes
Lackey ▾	6	2	0	10	-1	10	20	Melee Weapon
Lackey ▾	6	2	0	10	-1	10	20	Melee Weapon
Lackey ▾	6	2	0	10	-1	10	20	Melee Weapon
Lackey ▾	6	2	0	10	-1	10	20	Melee Weapon
Lackey ▾	6	2	0	10	-1	10	20	Melee Weapon
Lackey ▾	6	2	0	10	-1	10	20	Melee Weapon
Lackey ▾	6	2	0	10	-1	10	20	Melee Weapon
Lackey ▾	6	2	0	10	-1	10	20	Melee Weapon
Lackey ▾	6	2	0	10	-1	10	20	Melee Weapon
Lackey ▾	6	2	0	10	-1	10	20	Melee Weapon

Chance to include 12 members.

9. There is no method for handling attempts at creating a new Warband with a name that already exists. This has to be carefully thought out when including multi-player - no two teams can have the same name, even if they are built by different players - involving the client is important for this decision.

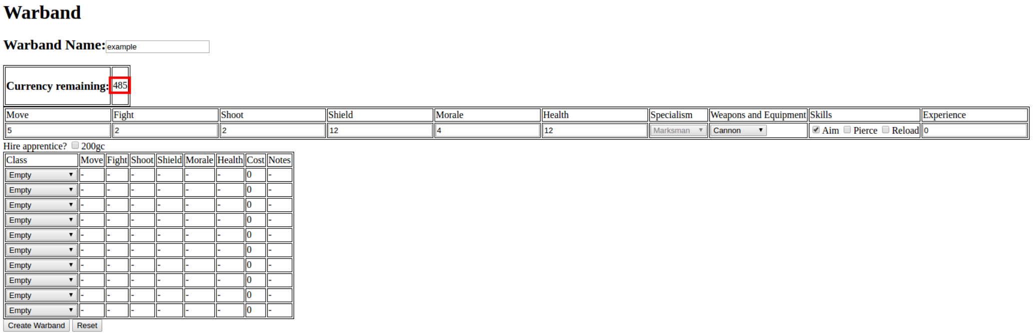
10. Consecutively creating new Warbands will accumulate the price of weapons and squad members (big problem).

11. Trying to create two new Warbands without leaving the “New Warband” menu does not allow to select weapon for Captain.

12. After creating a Warband with the Captain’s weapon being a cannon, the next new Warband’s Captain will also have a cannon, regardless of the choice of weapon.

3.2 Edit Warband

1. The most serious problem in the “Edit Warband” menu is the handling of credits. When editing the Warband the number of credits is modified erroneously:
   A new Warband with only a Captain with a cannon (15 credits) is generated. Then, in the “Edit Warband” page, clicking “Create Warband” decreases the number of credits by 15 (shown in the figures below). Like this, there are other errors with the credit handling in the “Edit Warband” page.



Warband

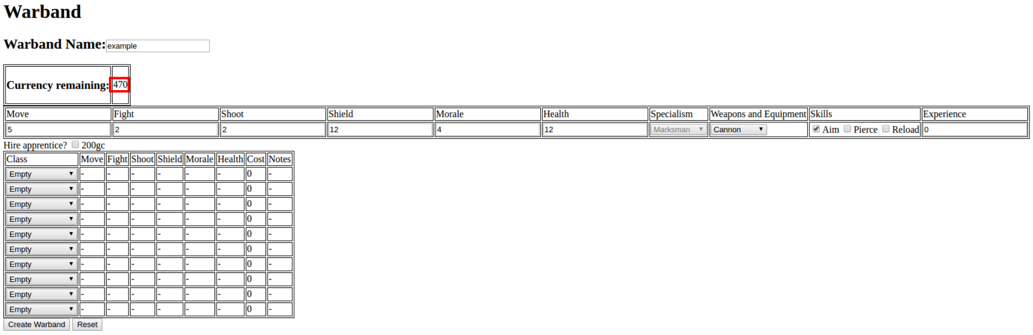
Warband Name: example

Currency remaining:	485

Move	Fight	Shoot	Shield	Morale	Health	Specialism	Weapons and Equipment	Skills	Experience
5	2	2	12	4	12	Marksman ▾	Cannon ▾	☑ Aim ☐ Pierce ☐ Reload	0

Hire apprentice? ☐ 200gc

Class	Move	Fight	Shoot	Shield	Morale	Health	Cost	Notes
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-



Warband

Warband Name: example

Currency remaining:	470

Move	Fight	Shoot	Shield	Morale	Health	Specialism	Weapons and Equipment	Skills	Experience
5	2	2	12	4	12	Marksman ▾	Cannon ▾	☑ Aim ☐ Pierce ☐ Reload	0

Hire apprentice? ☐ 200gc

Class	Move	Fight	Shoot	Shield	Morale	Health	Cost	Notes
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-
Empty ▾	-	-	-	-	-	-	0	-

2. It is not possible to hire an Ensign after creating the Warband.

3. Squad members are treated like slaves, i.e. it is possible to sell them for the original amount of credits that they were “hired” with. This needs to change according to the game rules.

4. The rules for increasing the Captain’s and Ensign’s skills is completely off - it is unlimited and completely uncorrelated to the experience points.

5. There is also no limit regarding the amount of experience points that can be added with a single edit.

6. The Captain and Ensign maximum values of skills (Annex) are not taken into consideration.

7. Captain is not able to have **two** weapons.

8. Neither Captain nor Ensign can obtain extra equipment.

9. When attempting to make an illegal Warband nothing happens - not even the generic error message returned previously.

10. Impossible to add number of credits.

The past two lists were obtained exclusively from using the current web application, and valuable information was extracted from this method. The focus of the report will return to this in section 4 for proposed solutions for the mentioned problems. These are, obviously, intimately linked to the problems found in the current source code. Thus it is also important to review the actual files that have generated the web application. An overview of the `python` files that are used for going back and forth between `html` files, as well as the actual `html` files is done.

3.3 `.py` and `.html` Files

The following lists some problems found with the given `python` script that is used to navigate inside the web application dealing with checking (or attempting to check) the validity of new or edited Warbands as well as the `html` files.

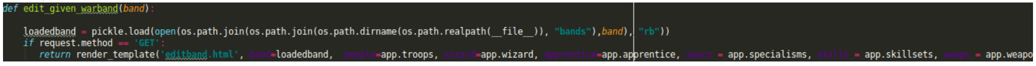
1. An important thing that must be considered is that this application does not install (or even check) for the dependencies it has (`Flask` is not among `python`'s built-in modules).

2. The application assumes that `python` is installed, while this might not be the case.

3. The use of `flask` framework requires an installation in many machines.

4. `framework.py` uses cryptic names like `wizard` to refer to the Captain, or `apprentice` to refer to the Ensign.

5. The `framework.py` file is not written for humans: Horrible style and formatting and almost no comments are used in the code.



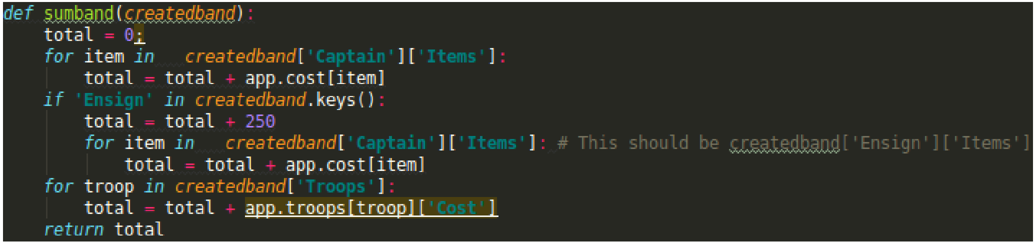
```
def edit_given_warband(band):

    loadedband = pickle.load(open(os.path.join(os.path.join(os.path.dirname(os.path.realpath(__file__)), "bands"),band), "rb"))
    if request.method == 'GET':
        return render_template('editband.html', band=loadedband,  people=app.troops, wizard=app.wizard, apprentice=app.apprentice, specs = app.specialisms, skills = app.skillsets, weaps = app.weapo
```

Careless formatting.

6. There is no `try/except` methods for actions that could incur in errors. For example, opening files in `framework.py`.

7. The validate functions (check that there's a positive number of credits, number of squad members, etc.) need to be written.

8. `framework.py`'s `sumband` function works incorrectly when an Ensign is included - it adds the Captain's weapons twice instead of summing the Captain's and the Ensign's weapons. (Related to problem 3.1.10).



```
def sumband(createdband):
    total = 0;
    for item in   createdband['Captain']['Items']:
        total = total + app.cost[item]
    if 'Ensign' in createdband.keys():
        total = total + 250
        for item in   createdband['Captain']['Items']: # This should be createdband['Ensign']['Items']
            total = total + app.cost[item]
    for troop in createdband['Troops']:
        total = total + app.troops[troop]['Cost']
    return total
```

`sumband` error.

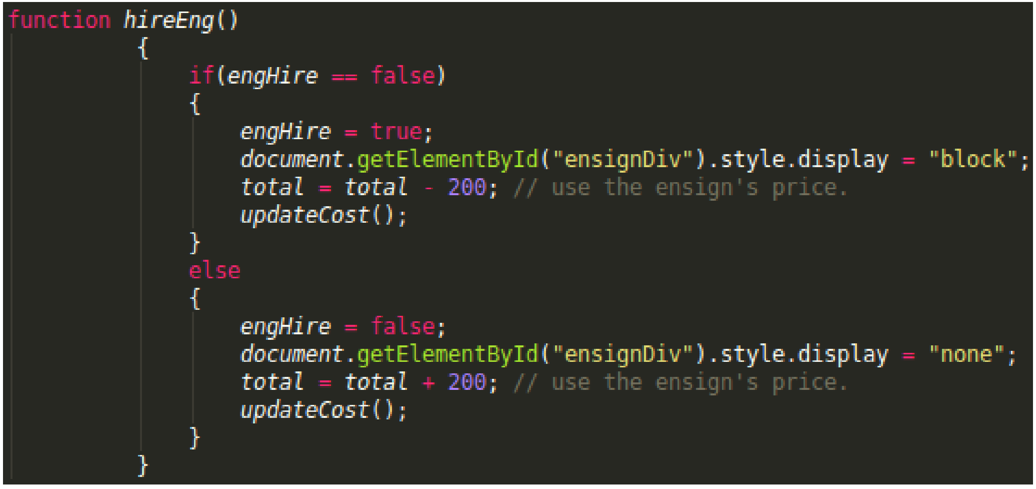
9. Use augmented assignment in `sumband` function.

10. `bandlist.html`, `blankband.html`, and `editband.html` show no route to go to `index.html` - this should be supported.

11. `blankband.html` and `editband.html` have strange formatting decisions.

12. `blankband.html` and `editband.html` mistakes the name of the Ensign with "Apprentice". This is related to problem 3.1.6.

13. blankband.html's hireEng() function uses a hardcoded (mistaken) value for the Ensign's hiring price. This is related to problem 3.1.7.



```
function hireEng()
        {
            if(engHire == false)
            {
                engHire = true;
                document.getElementById("ensignDiv").style.display = "block";
                total = total - 200; // use the ensign's price.
                updateCost();
            }
            else
            {
                engHire = false;
                document.getElementById("ensignDiv").style.display = "none";
                total = total + 200; // use the ensign's price.
                updateCost();
            }
        }
```

Hardcoded values in hireEng() function.

14. editband.html and blankband have several formatting errors - again, not designed for humans.

These are the problems found in the web application source files. Nevertheless, they are probably just scratching the surface, and when the process of fixing them starts, considerably more problems will be discovered. As it was mentioned in the introduction section, it is important to take into account the level of experience at hand. Having no knowledge of Flask, JavaScript or html a learning curve behaviour will need to be included in time and effort estimations.

This means that two factors will increase the uncertainty of the project planning. Both are directly related to a lack of experience in the programming languages that are used in the current code. The first factor is that the errors identified in the current report constitute only a fraction of the entirety of errors that actually exist in it. This will need to be included in the project planning (Gantt chart) in the form of an increase in the time needed to fix all problems in the code. This will be done through a combination of estimation and accountability of research/study time: tutorials of Flask and html (the main source of uncertainty at the moment) will make up around 8 to 10 work hours in the Gantt chart, and by considering that the number of problems found will be roughly double the number of problems that this report has found (only considering section 3.3).

4 Solution Proposals

The problems identified in the previous section are, in general, not extensively complicated. Some of them require previous knowledge, and thus will require more time than others. A subset of these problems – those related to the multiplayer implementation – requires the most amount of effort. Similarly, becoming acquainted with html will require some effort at the beginning. Throughout the progress of the project, time is allocated for error handling and testing.

The problems are classified as follows:

1. Formatting problems. For close to 1200 lines of code (generally not the best metric for estimations, but this is solely for formatting), around 2 to 3 hours will be required with the help of an adequate IDE. This also includes fixing cryptic names, style, formatting and comments. This is imperative before starting any coding, and also important before delivering the end product (it has to be properly commented).

2. Dependencies problems - implement methods that check the presence of `Flask` and other dependencies. In the end it should install the required modules such that the application can run smoothly. Around 1 to 2 hours to complete. This can be done among the final tasks - when all the necessary modules will be referenced to.

3. Validate functions. This will be among the first few methods to go deep into the code, so a thorough understanding of it is necessary. Some effort has to be directed at familiarising with the `flask` framework. Around 3 to 5 hours.

4. Update functions. This will deal with experience points conversion to skills in a legal manner, Captain and Ensign's extra weapons or equipment, addition of credits, removing squad members, and many more (including any problems that are found in the future). Around 5 to 6 hours.

5. `html` tutorial. Going through a basic `html` tutorial and writing methods to go from "Edit" or "New" Warband menus to "Home". Around 2 to 4 hours.

6. `html` functions. Go through problems 3.3.11-13, and very likely find several more errors. Around 5 hours.

7. Internal error message handling. This relates implementing `try/except` and `assert` methods to handle errors and help in unit testing. This is something that needs to be done throughout the coding process - around 2 to 4 hours.

8. External error message handling. This relates to problem 3.1.3, and means identifying illegal squads and writing `html` error messages (with ways to return to "Back" or "Home". Without considering the time for writing the validate functions, around 1 to 2 hours to complete.

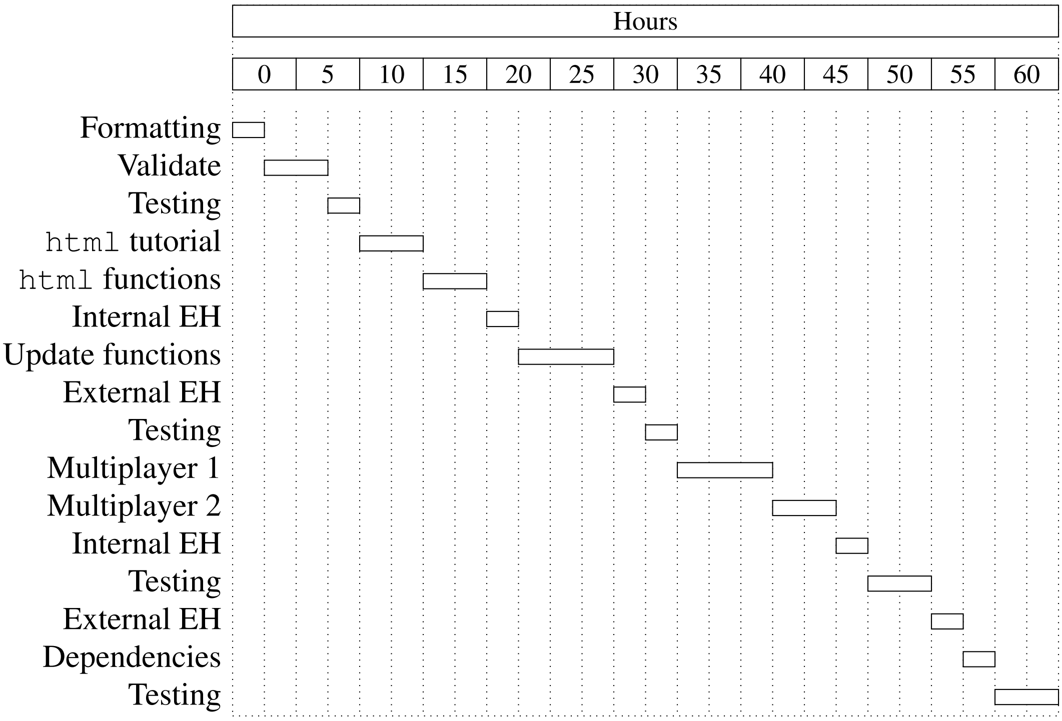
9. Multiplayer 1. Web application security tutorial and implementation of login with player name and password (talk with client first). Only the Warband's owner can edit it, but everyone else can see it. Around 4 to 7 hours.

10. Multiplayer 2. Implement the web application for multiplayer, adding player's name to the Warband's information. Around 4 to 7 hours hours.

11. Testing. Repeat previous tasks to keep consistency with previous advancements. Around 1 to 2 hours.

5 Time/Effort Estimation Plan

This section features a Gantt chart that uses the problems found on the code as well as the proposed solutions as a guideline. Said chart summarises the proposed solutions with the estimation of the effort and time that each of them will require. Several instances of error handling and testing are allocated in the Gantt chart. This is very important to produce a well rounded end product and intimately related to the fact that the chosen development model is the waterfall with subprojects (this will be further discused in the following section).



6 Risk Analysis and Management Strategy

As it was mentioned before, the strategy that will be used for the development of this project will follow the waterfall with subprojects model. For this model, the software concept and the requirements analysis have been already covered and explained in section 2.

The architectural design of the web application has already been built and exists in the provided code, but has to be extended to multiplayer support. Before doing so, one full “loop” from architectural design to testing and back is performed. Afterwards, the multiplayer support is created and a full loop is done once more. A few instances of error handling are also used in order to catch errors through the code.

The processes that have been mentioned help with risk management. Throughout the

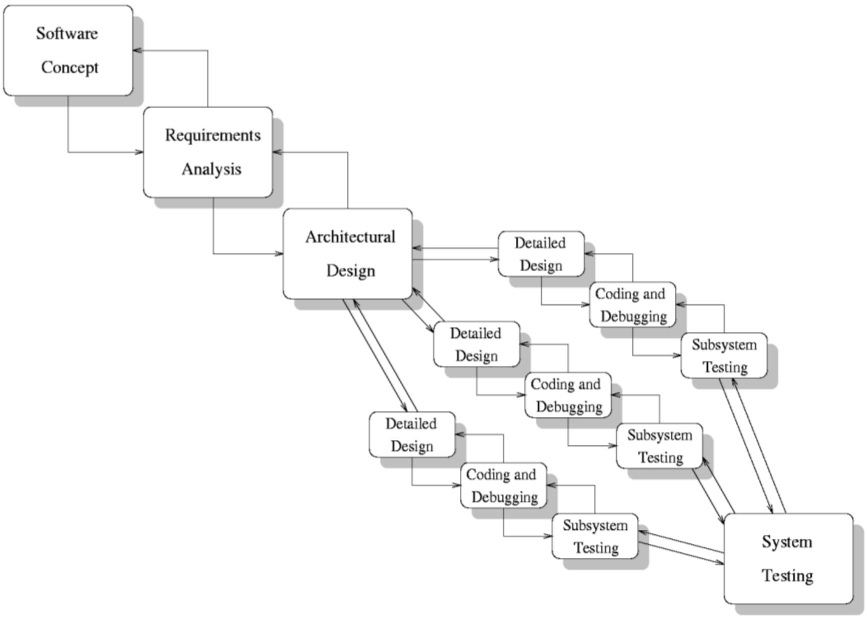


Figure 1: Waterfall with subprojects

improvement of the code it is necessary to check if the Warbands that are being generated are legal before and after dealing with the multiplayer platform. The Gantt chart attempts to account for this through systematically testing as well as Internal and external error handling. Another possible risk that is mitigated by the project planning is the possible dependencies that the application may need.

The largest risk is, as has already been mentioned, the lack of experience in many aspects of the project's current build. This has been referenced several times and it is something that is being taken into consideration through the development of the project planning. This has been accounted into the necessary time for "preparation" for some of the tasks.

7 Annex

7.1 Base stats

Besides the stats shown in table 1, both Captain and Ensign start with empty Skillset (that has to be filled when creating a squad), as well as an Associated Specialism, and no Items (one Weapon has to be added when creating a squad). All other stats are summarized in the following table.

Squad Member	**Move**	**Fight**	**Shoot**	**Shield**	**Morale**	**Health**	**Cost**	**Experience**
Captain	5	2	2	12	4	12	0	0
Ensign	7	0	-1	10	2	8	250	0
								Notes
Augment Gorilla	8	3	0	10	2	8	20	Animal, cannot carry treasure or items
Lackey	6	2	0	10	-1	10	20	Melee Weapon
Security	6	2	1	12	2	12	80	Blaster, Blade
Engineer	4	0	3	12	2	10	60	Blaster, Repair Kit
Medic	5	0	0	12	3	10	50	Blade, Medkit
Commando	8	4	0	10	4	12	100	Stealth Suit, Blade, Needle Gun
Combat Droid	3	2	4	14	0	14	150	Mechanoid, Dual Blaster, Claws

Table 1: Base stats for each squad member.

7.2 Maximum Stats

As the Captain and Ensign are able to increase their stats, it's important to state their maximum stats. In order to increase any of these stats by **1** point, the corresponding squad member (Captain or Ensign) will loose **10** EXPERIENCE points. In one edit no stat can be increased by **more than 1** point.

Squad Member	**Move**	**Fight**	**Shoot**	**Shield**	**Morale**	**Health**
Captain	5 (+0)	2 (+8)	2 (+8)	12 (+0)	4 (+8)	12 (+8)
Ensign	7 (+0)	0 (+6)	-1 (+6)	10 (+0)	2 (+4)	8 (+8)

Table 2: Maximum stats for Captain and Ensign.