Progress report marking sheet

Student's name: **Christopher Alastair Irvine**	Student's Reg: **100036248**	
Supervisor: **Dr Katharina Huber**	Date:	Signed:
2nd marker:	Date:	Signed:
Title: **CODEX: A Website and Progressive Web App in Node.js & React for table-top role-playing games**		

	Agreed Marks
Description of project: aims, motivation	/10
Description and understanding of issues and problems addressed in the project	/20
Design and planning	/50
Evaluation of progress	/10
Appropriate use of LaTeX	/10
Overall mark	/100

Comments

Report and completed mark sheet should be returned to the LSO.

CODEX: A Website and Progressive Web App in Node.js & React for table-top role-playing games

Christopher Alastair Irvine

registration: 100036248

Figure 1: (Left to Right) A: In this image we can see a typical D&D table set up. The items on the table from left to right is the *Monster Manual* (Perkins, 2014), scratch pad of paper, pens, series of die, DM's Screen, a battle mat, set of tokens, a character sheet, more die and the *Player's Handbook* (Mearls and Crawford, 2014b). A group playing D&D does not need all of the listed equipment each. The minimum equipment needed in a group, are the three core books (Player's Handbook, Monster Manual and *Dungeon Master's Guide* (Mearls and Crawford, 2014a)), one set of die and some paper. Everything else is optional.
B: In this image we see the die necessary to play D&D. They are (from left to right); 4, 6, 8, 10, 12, 20 and 100-sided. The die are denoted by the letter d followed by the number of sides (for example the 8-sided die is a d8). The d100 only has 10 sides but the sides increase in multiples of 10 and when rolled together with the d10 is known as the *percentile die*.

1. Introduction

Dungeons and Dragons (D&D) is a popular table-top role-playing game which (for the purposes of CODEX) is based on the following principles. D&D is random and the game cannot survive without a *Dungeon Master* (DM). For a glossary of D&D terms and concepts, please see Appendix A. Terms and Concepts introduced in Appendix A will be referred to throughout this document. A typical D&D set up can been seen in Figure 1

1.1. Shortcomings of D&D

CODEX is a system that will address some of the shortcomings within the 5th Edition of D&D. Whilst they are labelled as shortcomings they are integral to the game and there are not better alternatives. So rather than replacing these shortcomings, CODEX will attempt to fix them.

1.1.1. The Dungeon Master

As explained in Appendix A, the DM creates the world in front of the party and controls the narrative of the Campaign. Without a DM there can be no D&D game. Therefore, the game of D&D cannot survive without a healthy population of DMs.

However, there is a great deal of pressure on the DM to create fun and engaging content for every single session of D&D. The task of preparing content is time-consuming and often the work done by the DM can go unused. This commitment to a create content is off-putting to a lot of players who want to try their hand at running a game. The pressure put on a DM is no greater than during combat, as they have to track the individual statistics for each character in the battle in addition to maintaining the narrative.

1.1.2. D&D is random

In Appendix A we saw the importance of die rolls in D&D and the role of a DM. It is the die rolls that allows every game of D&D to be unique and exciting. However, for the DM it the die rolls can cause of all their preparation work to be frequently wasted.

1.1.3. D&D content is overlooked

One of the most common questions a Player might ask a DM during a game is *"Who was that guy back in that town from a couple of sessions ago?"*. Player's who do not take sufficient notes during a Campaign can often find themselves lacking in information when solving issues. Either the DM caves to the Player's request and digs out the information themselves, or the Party are stuck in the Campaign.

1.2. Fixing the Shortcomings with CODEX

The features of CODEX are designed to solve the three shortcomings mentioned in Section 1.1, the details of which will be discussed in Section 2. CODEX will support the DM role by providing a platform for easy content creation. When a new *Character* or *Setting* is created within a Campaign, a "Wikipedia" style page will be generated. That both the Party and DMs can access. CODEX has a *Combat Tracker* that will track combat statistics allowing the DM to focus on maintaining the narrative.

1.2.1. Achieving CODEX

These features will be achieved through the use of the Agile Solo methodology (Nyström, 2011). Agile Solo is based to the Scrum Software Engineering Methodology (Schwaber, 1997), which provides a system to track the development of a project and ensure that the deliverables are met. Scrum is developed for a team of people whereas Agile Solo is developed for single developer projects.

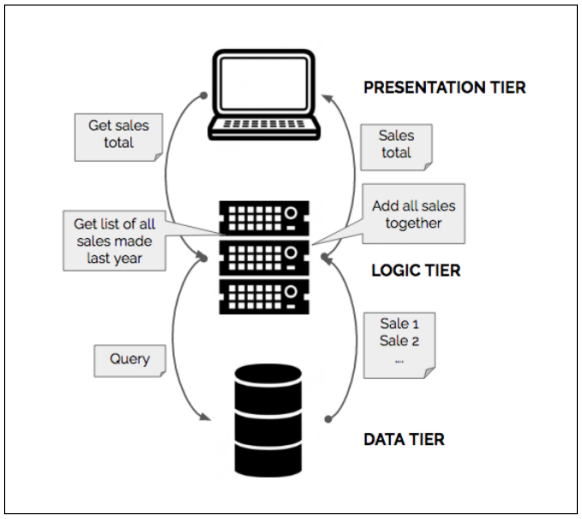

Figure 2: Overview of a three-tier web application and its dependencies (De Groef, 2016). User Input is passed the Web-App through the Graphical User Interface (GUI) in the *Presentation Tier* to the *Logic Tier*. The *Logic Tier* performs calculation on the data before passing it into the *Data- Tier*. The *Logic Tier* can also request data from the *Data Tier*, to perform calculations, before passing the data back into the GUI to be displayed to the User.

The development of Codex will use the ReactJS and Node.JS JavaScript libraries. ReactJS can rapidly built and prototype User Interfaces (UI) across multiple platforms (Inc., 2017). Whilst, Node.JS will handle the passing of data between CODEX and the corresponding database (Node.js, 2017). Both of these technologies follow the basic Web-App Architecture laid out in Figure 2.

2. CODEX Design

In this section, we will review the progress of the Design elements of CODEX. As we saw in Section 1.2.1, CODEX following the Agile Solo methodology, which assists with maintaining project progress. Trello is used in conjunction with Agile Solo, see Figure 3, and requires a weekly review of the status of the project. Bi-weekly meetings with an external supervisor is also advised by Agile Solo.

2.1. Iterative Design

Agile Solo, see Sections 1.2.1 and 2, supports an iterative form of development. The ReactJS JavaScript library complements iterative design due to the ability to rapidly prototype interfaces. However, before any development is performed, CODEX has passed

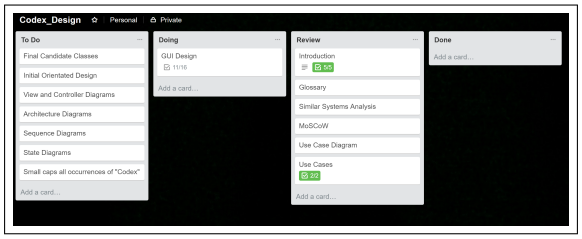

Figure 3: The Trello Board used to track the progress of the CODEX Design Iterations. Each item on Trello acts like a virtual post-it note. Each note containing a task that needs to be addressed.

through two phases of design. The first iteration focusing on low-fi prototyping (pen and paper) of the system, see Section 2.5 . Think-Aloud Evaluations were then performed using the low-fi prototypes, with the feedback received incorporated into the second Development iteration, which focused on mid-fi prototyping (see Section 2.6) (using services such as the Marvel App (App, 2013)).

2.2. The Use of ReactJS and Node.JS

As discussed in section 1.2.1 and, CODEX aims to follow the Web-App Architecture displayed in Figure 2. The *presentation* and *logic* tiers will be built using ReactJS with the communication to and from the *data* tier will be written in Node.JS (with the database itself written in SQL). In addition to providing a clear separation of languages and functionality the three-tier Web-App Architecture also follows the Model-View-Controller (MVC) architectural pattern (Kruchten, 1995). However, the MVC architecture is complicated by the use of ReactJS, which blurs the line between the *Model* (logic) and *View* (presentation) tiers. This is a problem that will be addressed in Section 4.

2.3. CODEX System Requirements

The requirements for CODEX were generated by identifying the features that would solve the problems with D&D that were discussed in Section 1. Once the features were generated, a *MoSCoW Analysis* was performed on the list, see Table 1 for the CODEX MoSCoW analysis.

2.4. Use Cases

Another critical part of the Design of CODEX is planning how the functionality of the system will flow. *Use Cases* and the *Use Case Diagram* inform a developer how a data is expected to flow between systems during a function. See Figure 4 for the CODEX Use Case Diagram. Each of the Use Cases listed in the Use Case Diagram is a single

Must have	Should have	Could have	Won't have
Track Combat	Multiple Campaigns per DM Account	Game Scheduling	Full descriptions of D&D
Accounts	Settings	Items	Pre-made Settings & Campaigns
Encounter Planning	Characters		Inter-account Direct Messaging
Random Encounters	WorldWiki		
Populated Database	Run, Save & Delete Games		
Session Planning			

Table 1: MoSCoW analysis for CODEX, showing the differing importance of wanted features to the system as a whole. The *Minimum Viable Product*(MVP) for the project is derived from the *Must have* column. The *Should have* column represents what features CODEX should contain for the system to be complete. The *Could have* is what features might give CODEX an edge over similar systems and what would be "nice to have". Finally the *Won't have* column represents what features will never be part of the CODEX system.

function within the CODEX system. The execution of each function is explained the corresponding Use Case, which the developer can refer to during development. Use Cases and Use Case Diagrams prevent errors and saves time during the development of a system.

2.5. Low-Fi Prototyping

Low-Fi Prototyping, as mentioned in Section 2.1 was used to test CODEX during the first iteration of Design. A low-fi prototype is a Pen and Paper sketch of what the GUI will look like. These sketches are then used in Think-Aloud Evaluations (where a subject is asked to vocalise their thoughts as they perform a series of tasks on a system) to ascertain the quality of the proposed GUI (Nielsen et al., 2002). The GUI, initially, had several pop-up elements for the Editing and Deleting functions.

The subjects all reported the constant pop-ups as "annoying" and "distracting". During the second iteration of design the pop-up elements were minimised to system messages only, and were a lot less obtrusive.

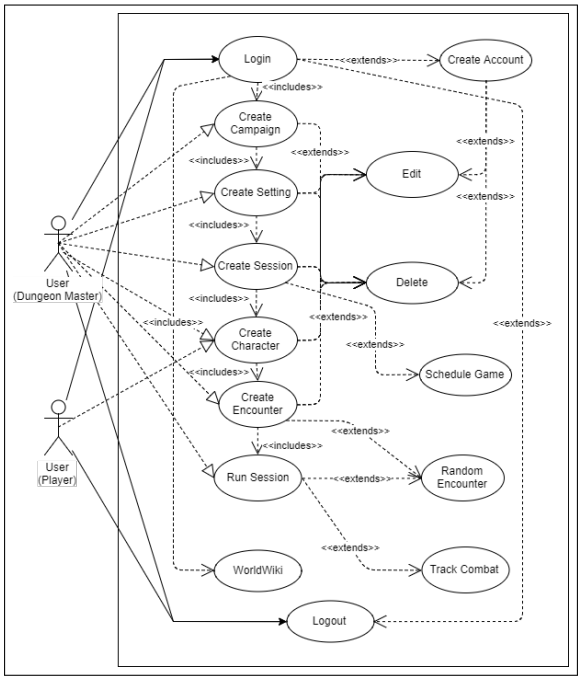

Figure 4: CODEX Use Case Diagram showing the accessibility to functions and the flow of information between them. Each node within the diagram is a Use Case and together explain how the various components of CODEX interact.

2.6. Med-Fi Prototyping

The Med-Fi prototypes were developed using MarvelApp (App, 2013), an online tool for the prototyping of system. Figure 6 shows two example of the many med-fi screens that were developed. These med-fi screens are used during the development process (see Section 3) as a guide for the developer to follow. Not only for how the system will appear to the user, but also how the screens will interact with each other. This process is also known as *Wireframing*.

2.7. Object Orientated Design

An Object Orientated Design Diagram, like the other diagrams mentioned in Section 2, provide guidance to a developer when programming a system. We can see in Figure 7 the flow between the current containers predicted to be CODEX. Each container possess a series of a components. CODEX aims to reuse as many of these components between containers as possible. Additionally the Navigational Elements of the system will be a shared view.

Use Case 6 (Edit Campaign)

USE CASE NAME	Edit Campaign	
Goal in Context	DM edits the details of a Campaign they have already created.	
Scope & Level	Dungeon Master System	
Preconditions	DM already has created a Campaign on their account	
Success End Condition	DM edits details about their campaign and the changes are saved to the database	
Failed End Condition	Changes fail to save to the database. DM must attempt their changes again at a later date.	
Primary Actor	Dungeon Master (DM)	
Trigger	DM selects the "Edit Campaign" button	
SUCCESS SCENARIO	**Step**	**Action**
	1	DM is redirected to the Edit Campaign screen
	2	DM changes the data about the campaign that they have previously entered.
	3	DM selects the "Submit Changes" button, the new data is sent off to the Database and changes are saved. Confirmation code is returned to Codex.
	4	DM is redirected to the Homepage, with pop-up message informing them that changes have been saved.
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	1a	DM is not redirected, will have to try again later
	3a	Changes to the Campaign are not sent to the database/fail to save/success code failed to be returned
	3b	Codex presents DM with error pop-up saying that changes failed to save, check connection and try again later. Redirected to homepage
	4a	DM is not redirected and/or success message not presented.
	4b	DM will need to manually return to homepage
	4c	DM will have to manually check Campaign details are correct.
RELATED INFORMATION		
Priority	Low	
Performance Target	0.7 seconds	
Frequency	Low	
Subordinate Use Cases	None	
Channel to Primary Actor	Edit Campaign Screen	
Secondary Actors	Database, Codex	
Channel to Secondary Actors	Internal System Communication	
OPEN ISSUES	None	
SCHEDULE	N/A	
AUTHOR	Christopher Irvine (23/11/17)	

Figure 5: One of the Use Cases for CODEX, this is for Editing an existing Campaign. Use Cases provide details to the developer about how this function is expected to function, preventing error and saving time.

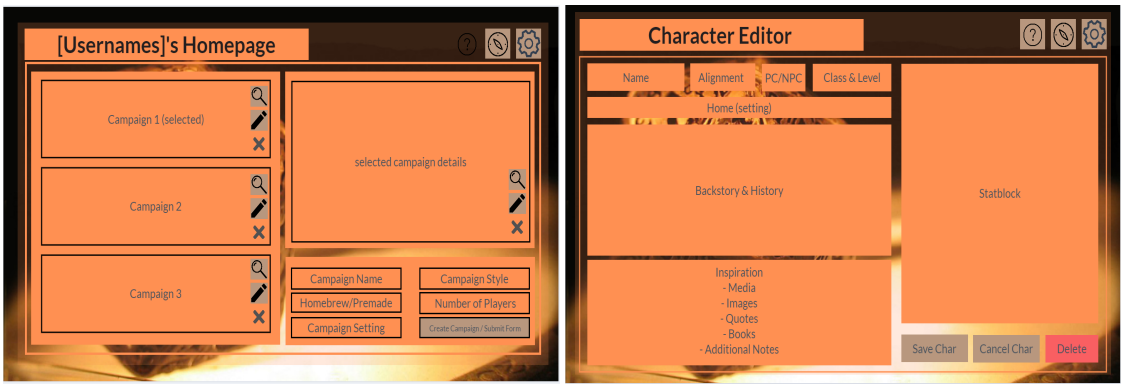

Figure 6: (Left to Right)A: Med-Fi Prototype of the homepage, developed using MarvelApp.
B: Med-Fi Prototype of the Character Editor Screen, used in the Creation, Editing and Deletion of a Character. Developed using MarvelApp.
In both prototypes Navigation Elements can be seen along the top of the screen.

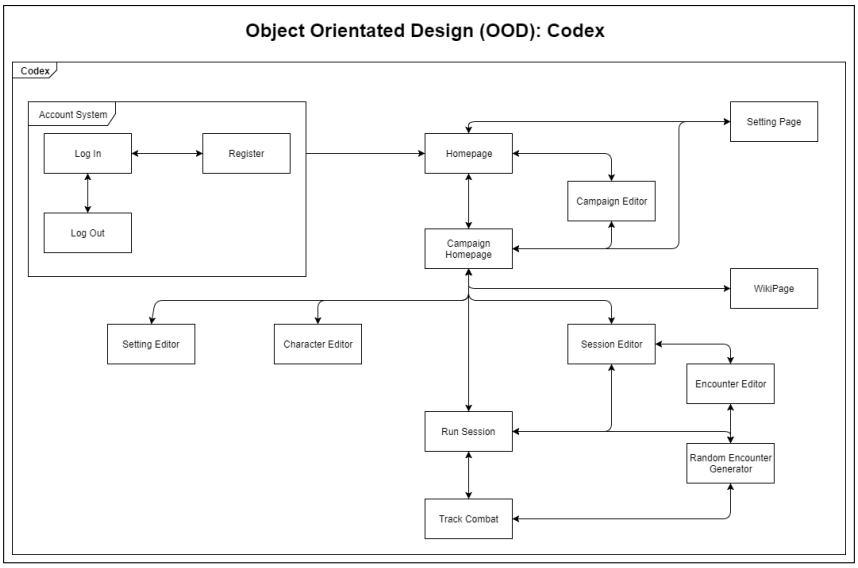

Figure 7: Object Orientated Diagram for CODEX displaying the Container Components. Each container has a 1:1 relationship with each other. There is a subsystem present, called Account, that can be accessed through the Homepage regardless of the logged in state of the user.

3. CODEX Implementation Plan

In Section 2 we saw that several design elements for CODEX exist with the common goal of providing guidance for the development of CODEX. Figure 7 and 4 provide a "road map" for the development whilst Figures 5 and 6 provide details of how the data flows between screens.

The development of CODEX should now be able to progress at a faster pace because of the planning and considerations taken throughout the first two design iterations. It is expected that the design will further alter as the development progresses, when functions that were expected to occur in one way can only work in another.

As discussed in Sections 1.2.1 and 2.1, CODEX will be developed using the Agile Solo methodology. This will help the project stay on track through constant reviewing of already completed work and allowing the development to focus on one task at a time.

3.1. CODEX Gantt Chart

The initial Gantt Chart (see Appendix B) generated for CODEX has changed slightly. The second design iteration of CODEX was brought forward in order to accommodate the Think-Aloud Evaluation results for the low-fi prototyping (see Section 2.5) before the development commenced. To compensate a Third Design Phase was added and the Development Iteration 1 was incorporated into Development Iteration 2. We should expect that the MVP for CODEX be completed by the end of Development Iteration 1 (see Appendix C).

3.2. Language and Architecture

As we saw in Sections 2.2 and 1.2.1 CODEX aims to use ReactJS in conjunction with Node.js in order to replicate a traditional Web-App Architecture (see Figure 2). The use of ReactJS, whilst providing a platform for rapid prototyping, testing and deployment, will also allow CODEX to be easily adapted for cross-platform use (iOS and Android).

4. Outstanding Issues

So far the progress of CODEX is satisfactory. There has not been a major delay or issue that CODEX has had to overcome. Minor issues do exist however. These are that CODEX has suffered from the needs of other university projects. Requiring time initially set aside for the development of CODEX be spent on these other projects. Additionally the Questionnaire mentioned in the Project Proposal has yet to be written or approved. Outstanding Issues:

4.1. Solutions

In order to compensate for the lost development time the MVP for CODEX has been simplified to allow for faster development iterations. This also relieves the pressure from the developer so that the quality of the development remains high.

The Questionnaire will be completed as part of Design Iteration 3 within the Gantt chart, however it will be written and approved as part of Development Iteration 1. By providing an extended period of time for the development of the Questionnaire and Ethics Approval, it increases the chance of the Questionnaire passing the Ethics Approval, as well as permitting time for adjustments to be made should it fail the Ethics Approval.

5. Conclusion

In this document; we have discussed the concepts of D&D that is critical to CODEX, overlooked content, the DM and the random nature of D&D (see Section 1.1). Next we examined the current Design of CODEX, specifically examining the iterations CODEX has already gone through, the requirements CODEX must meet, use cases written to detail how the requirements will be achieved and the choice of languages (see Section 2). Finally we reviewed the new Development Plan for CODEX and created Solutions for to handle the minor outstanding issues with the project (see Sections 3 and 4).

References

App, M. (2013). Marvel app.

De Groef, W. (2016). *Client- and Server-Side Security Technologies for JavaScript Web Applications*. Phd thesis, Faculty of Engineering Science.

Inc., F. (2017). Reactjs.

Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *IEEE software*, 12(6):42–50.

Mearls, M. and Crawford, J. (2014a). *Dungeon Master's Guide*. Wizards of the Coast.

Mearls, M. and Crawford, J. (2014b). *Player's Handbook*. Wizards of the Coast.

Nielsen, J., Clemmensen, T., and Yssing, C. (2002). Getting access to what goes on in people's heads?: reflections on the think-aloud technique. In *Proceedings of the second Nordic conference on Human-computer interaction*, pages 101–110. ACM.

Node.js, F. (2017). Node.js.

Nyström, A. (2011). Agile solo-defining and evaluating an agile software development process for a single software developer. Masters thesis, Department of Computer Science and Engineering.

Perkins, C. (2014). *Monster Manual*. Wizards of the Coast.

Schwaber, K. (1997). *SCRUM Development Process*, pages 117–134. Springer London, London.

A. Dungeons and Dragons Glossary

Player: An individual who participates in a **Campaign** that is ran by a **Dungeon Master**. The player can either be a guest for a **session** or two, or a regular in the Campaign. Players control a **Player Character**.

Party: A group of **Player Characters** that are allied with each other to achieve a common goal, usually the objective of the **Campaign**. Player Characters can join and leave the party at will, but typically **Players** will stay together for the duration of the Campaign that are playing through.

Dungeon Master (DM): The individual who runs the game of Dungeons and Dragons for the **Players**. The role of a DM includes preparing content for each **session** of D&D as well as narrate and control the sessions themselves.

Campaign: A series of **challenges** that together form a story, known as the Campaign. The Campaign is created, prepared and ran by the **Dungeon Master**.

Setting: Campaigns occur within a Setting. A Setting can be anything from a small fishing hamlet on the side of a lake to an entire planet or beyond. There can be multiple Settings within a Campaign. Usually, a DM will create their general Setting and populate it with more specific, smaller settings.

Sessions: A Session is an episode within the Campaign that is delivered by the Dungeon Master to the Players, who react and interact with the session, which will impact the Campaign and in turn the Setting. Sessions need not be planned in detail, but do need guidelines, so that there is enough content for the Players to experience over the course of the session.

Character: A fictional character within a **Setting** of Dungeons and Dragons. All characters have the same abilities, with different values. Characters are defined as **Player** or Non-Player.

Challenges: A challenge is an obstacle that prevents the **Party** from progressing on with the **Campaign**. These challenges can be anything from persuading the Captain of a ship to give you passage, finding the entrance to a long lost ruined city or defeating an enemy in **Combat**. Challenges are resolved through **Die Rolls**.

Combat: Combat is a physical altercation between the **Party** and a group of **Enemies**. Combat is over once one side has either fled or defeated the other. This is achieved by **Characters** attacking each other with **Attacks**, **Spells** and **Abilities**.

Spells/Abilities: Spells/Abilities are powerful actions that a character can perform during the game. The number of times these can be performed is limited. Spells/Abilities can inflict **Damage** or **Healing** onto a target.

Attacks: All Characters can make a number of Attacks with their action each turn. When an attack hits a target, All Characters can receive Healing, which restores lost HP. Unless stated otherwise, HP cannot be restored above the Maximum HP or applies to Temporary HP for the character. is inflicted.

Damage: All Characters can receive Damage, when this happens that **Character Hit Points** drop an equal amount to damage received.

Healing: All Characters can receive Healing, which restores lost **Hit Points**.

Hit Points (HP): This represents how much damage a character can withstand before collapsing unconscious at 0 Hit Points. Characters have a Maximum limit to their Hit Points which increases when they grow stronger over time. Hit Points can be lost through **Damage** and restored **Healing**.

Proficiency: When a **Character** is particularly skilled at a task, it is said they are *Proficient* at it. When a character is Proficient they can apply a bonus *Proficiency Modifier* to all rolls for that task. The proficiency modifier increases as characters get stronger.

Ability Statistics: Every **Character** is defined by a set of six Ability Statistics *(Strength, Dexterity, Constitution, Intelligence, Wisdom and Charisma)*. These can range from 0 to a potentially infinite amount, but usually the maximum is around 30. It is down to the preference of the **Dungeon Master**.

Ability Modifier: The **Ability Statistics** translate into modifiers that can be applied to the relevant **Die Rolls**. This is done by taking the Ability Statistic, subtracting 10 and diving by 2 (then rounding up). This modifier can be positive or negative to reflect the strengths and weaknesses of each **character**.

Die Rolls: All **challenges** in D&D are resolved through Die Rolls. These are normally settled by rolling a 20-sided die (d20) and then applying the **Ability Modifier** and **Proficiency** to that die roll. The higher the result the more positive the outcome.

For example when Persuading a Guard for to give up information about the corrupt Baron who runs the town, would roll a Persuasion Roll. The character rolls a 11, a low result, but once he adds his Charisma (+4) and his Proficiency (for being Proficient in persuasion, +3) the roll is now 18. Which is likely more than enough to get some information out the guard.

B. Gantt Chart - Original

C. Gantt Chart - New

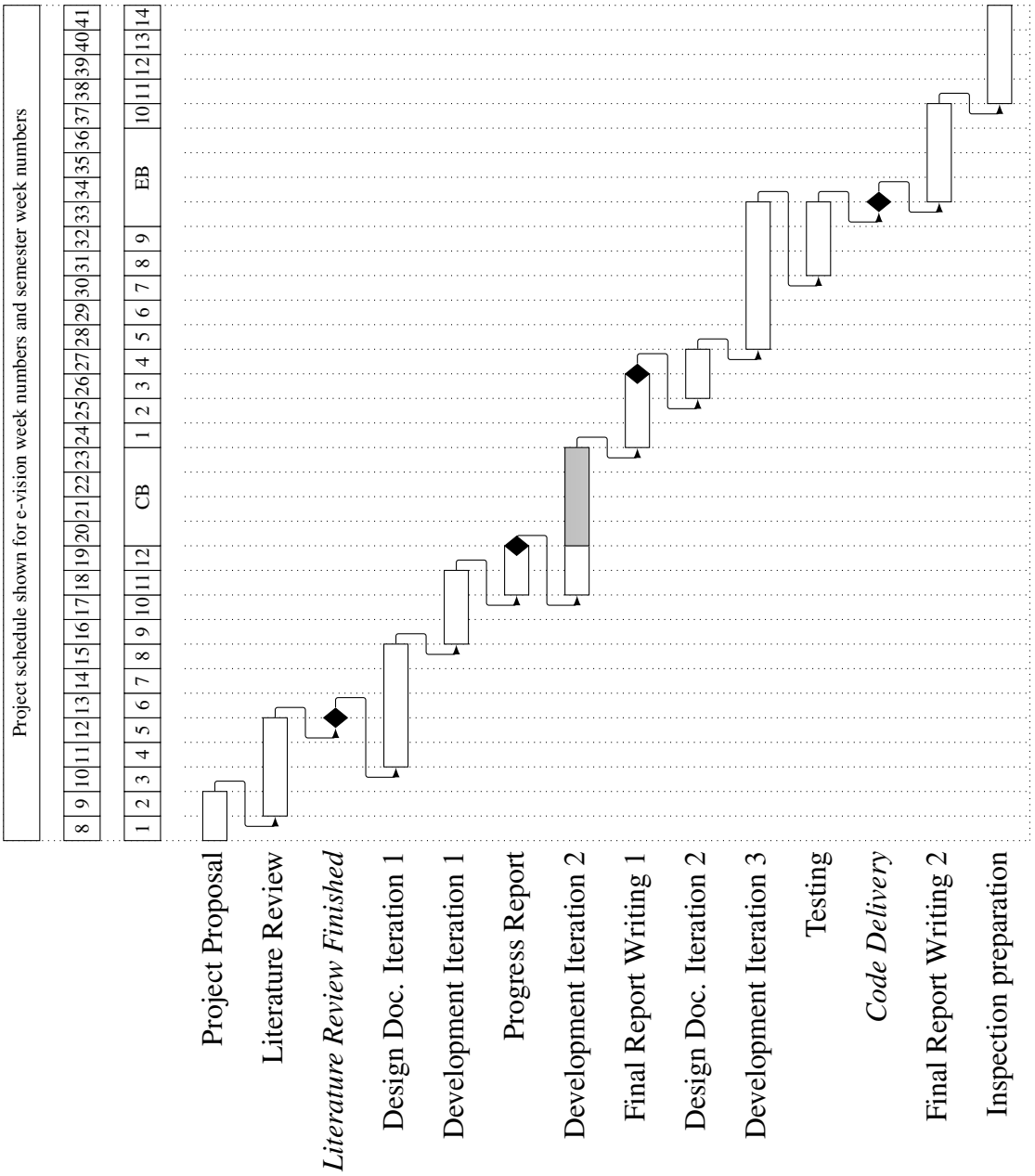

Figure 8: CODEX Gantt Chart, outlining the major tasks and deliverables

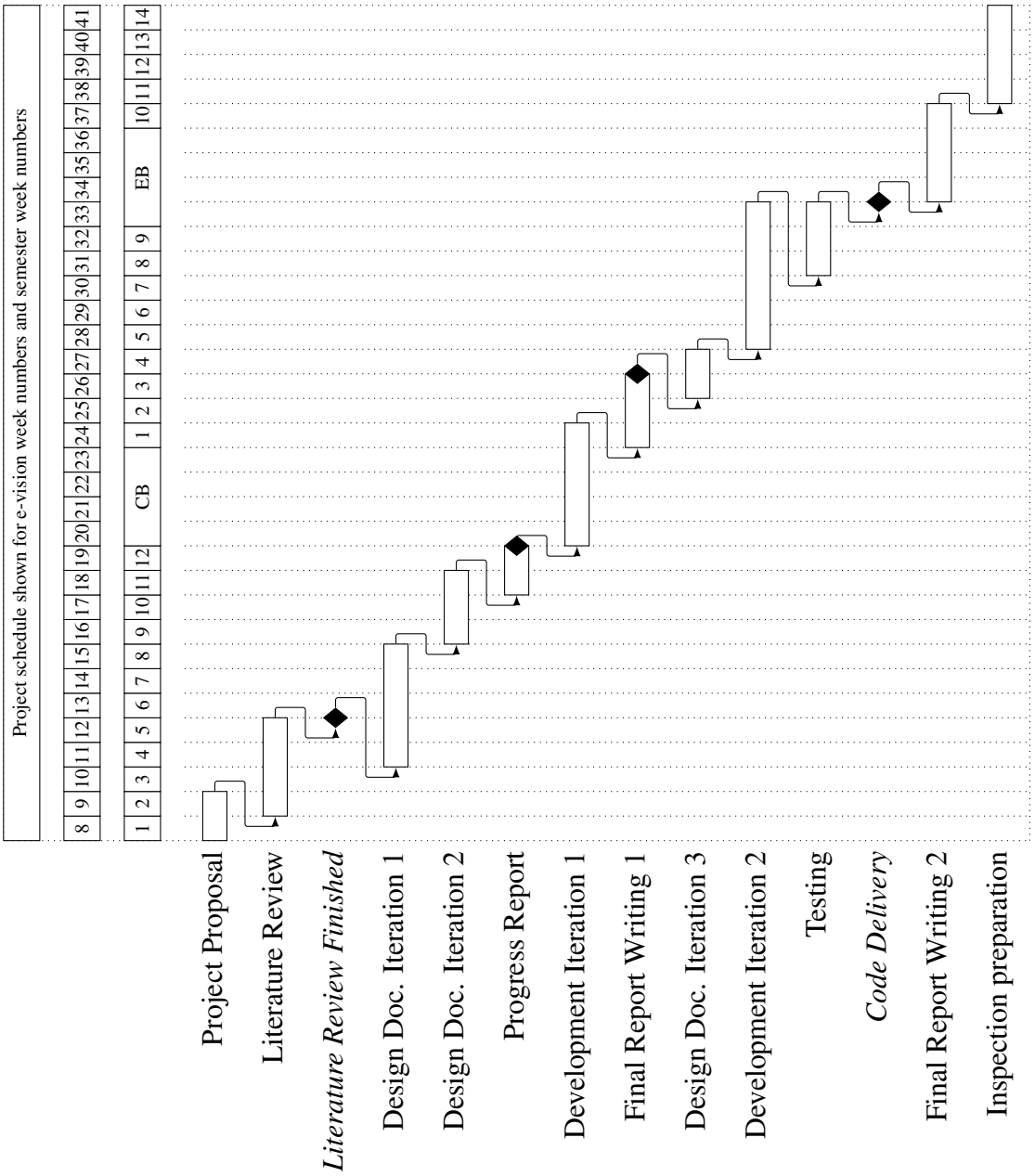

Figure 9: Revised CODEX Gantt Chart, outlining the major tasks and deliverables