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

1. Introduction

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 worked example of D&D please see Appendix A. In the example we will see how a group of players (known as the *Party*) explore the world through a series of challenges (such as hitting an enemy or picking a lock, that is collectively known as the *Campaign*) which the DM has created. Concepts introduced in Appendix A will be referred to throughout this document. A typical D&D set up can been seen in Figure 1

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*.

1.1. 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.

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.

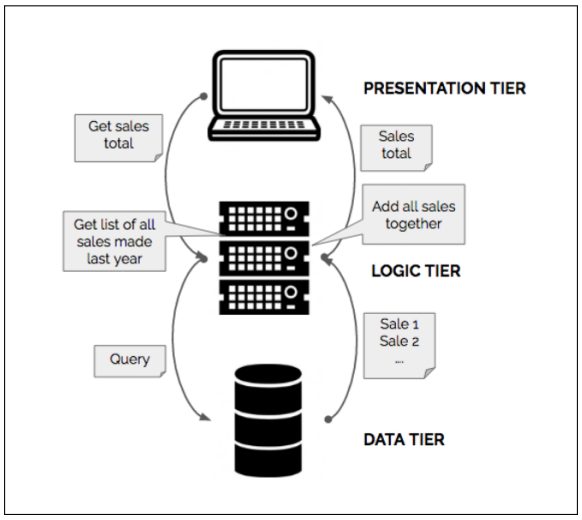

Figure 2: Overview of a three-tier web application and its dependencies (De Groef, 2016). User Input is passed the Web-App through the Graphical User Interface (GUI) in the *Presentation Tier* to the *Logic Tier*. The *Logic Tier* performs calculation on the data before passing it into the *Data- Tier*.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

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.

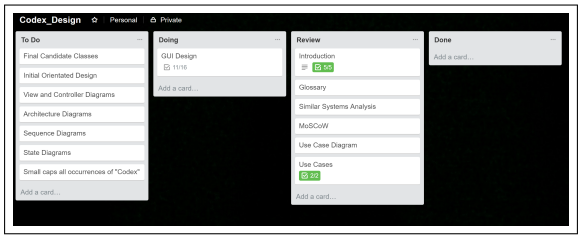

Figure 3: The Trello Board used to track the progress of the CODEX Design Iterations. Each item on Trello acts like a virtual post-it note. Each note containing a task that needs to be addressed.

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 through two phases of design. The first iteration focusing on low-fi prototyping (pen and paper) of the system, see Section . 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 (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.

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.

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 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

Discuss:

- What is Low-Fi
- Purpose of Low-Fi
- Think-Aloud Evaluations & Results
- Changes to GUI made because of Think-Aloud

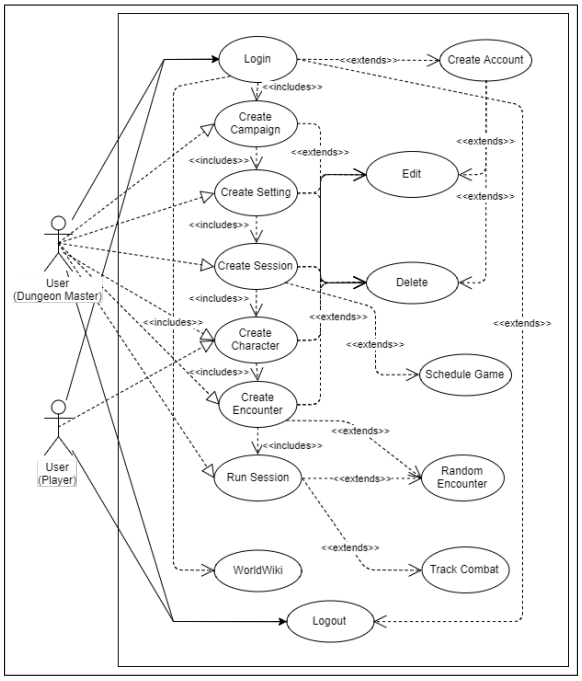

Figure 4: CODEX Use Case Diagram showing the accessibility to functions and the flow of information between them. Each node within the diagram is a Use Case and together explain how the various components of CODEX interact.

2.6. Med-Fi Prototyping

Show the current med-fi prototypes built using MarvelApp.

2.7. Object Orientated Design

Discuss:

- What is OOD?
- What is the purpose of an OOD Diagram?
- Show the OOD Diagram for CODEX

3. CODEX Implementation Plan

Discuss:

- Recap the Design Section that is directly related to Implementation (i.e. Choice of Languages/Architecture/OOD)

- Link to new Gantt Chart (see Appendix 7)
- Show command line modelling for combat? Screenshot of outputs?
- Reiterate Agile Solo

4. Outstanding Issues

Outstanding Issues:

- Coursework taking up more time than expected, a couple of lost development days
- Haven't addressed the questionnaire mentioned in the proposal, pushed back to Design 3 and Development 2

4.1. Solutions

- already simplified the MVP to help with the coursework issue
- need to develop the questionnaire and run it through ethics approval, use Christmas break to start this process

5. Conclusion

Discuss:

- overall happy with the current development of CODEX
- iterate the importance of holiday development time
- summarise the issues and solutions

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.

A. Dungeons and Dragons Example

A.1. Mathematics of D&D

As previously stated, almost everything that happens during a Campaign is decided by die rolls. However the Party (and the *characters* they encounter) do have a small degree of control over their actions. Each character in D&D has a set of *ability scores* that make them unique, each score having a value of 0 to 20 (unless an item changes that) (Mearls and Crawford, 2014b). These ability scores are translated into *ability modifiers*, which, for better or worse, affect the outcome of the die roll. If a character is particularly skilled in a task they are said to be *Proficient* in it, allowing the player to add their *proficiency* to the die roll. With the exception of damage rolls, the d20 die is used to make decisions in D&D. Albeit with some minor variations, die rolls in D&D 5th Edition follow the following formula:

$$d20 + ability\ modifier + proficiency = result$$
$$For\ example:$$
$$15 + 4 + 2 = 21$$

However, as far as CODEX is concerned, the Party and DM will perform these calculations for themselves as its part of the fun of the game. Attack rolls, damage rolls and ability checks will be held out width CODEX. The mathematics that CODEXwill handle is the calculation of Ability Modifiers:

$$\frac{(ability\ score - 10)}{2}\ (rounded\ down)$$

In addition to the tracking of each character's *hit points* during combat. Which is a simple addition or subtraction depending on whether a character took damage or gained healing.

B. Gantt Chart - Original

C. Gantt Chart - New

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.

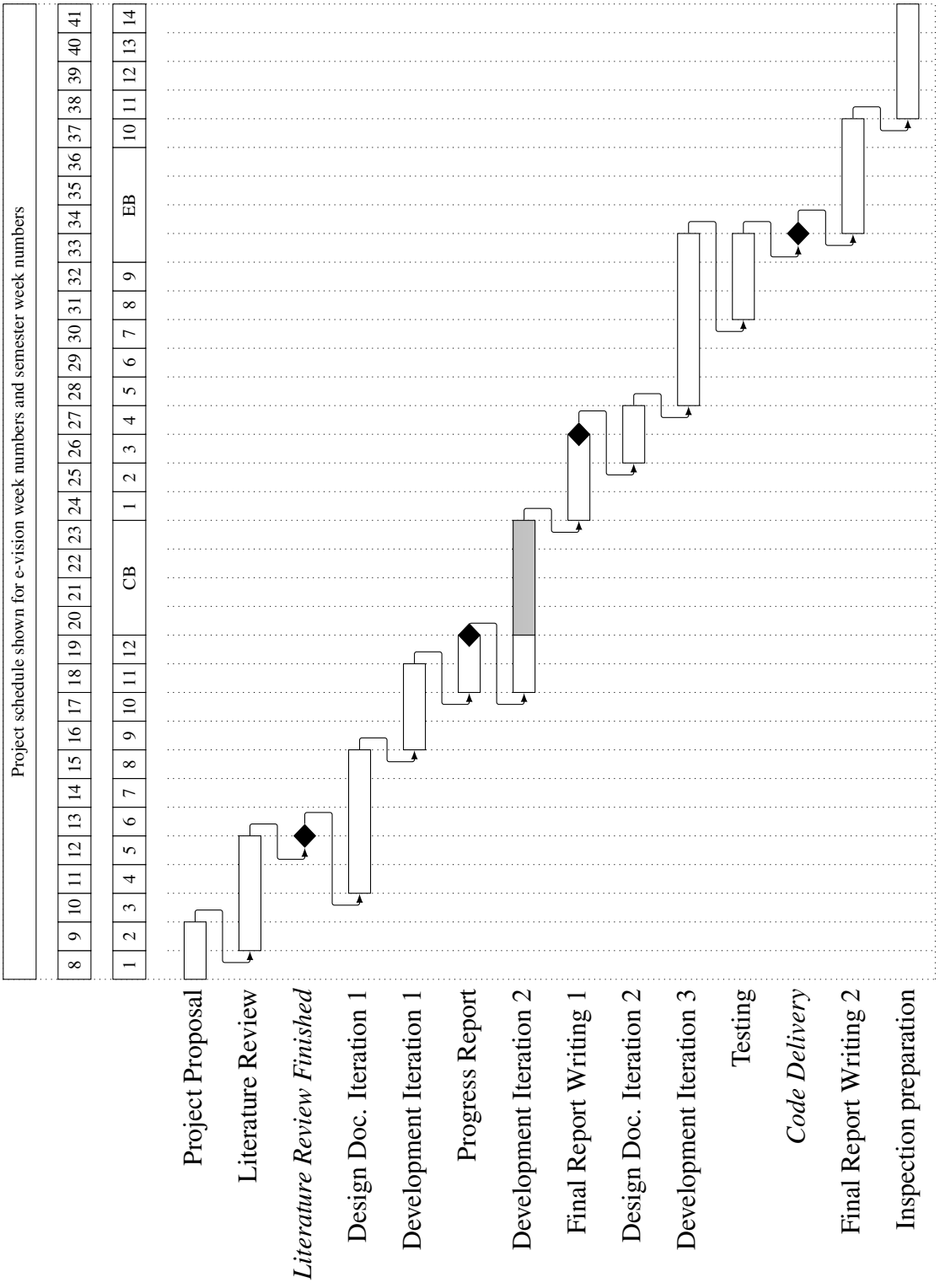

Figure 6: CODEX Gantt Chart, outlining the major tasks and deliverables

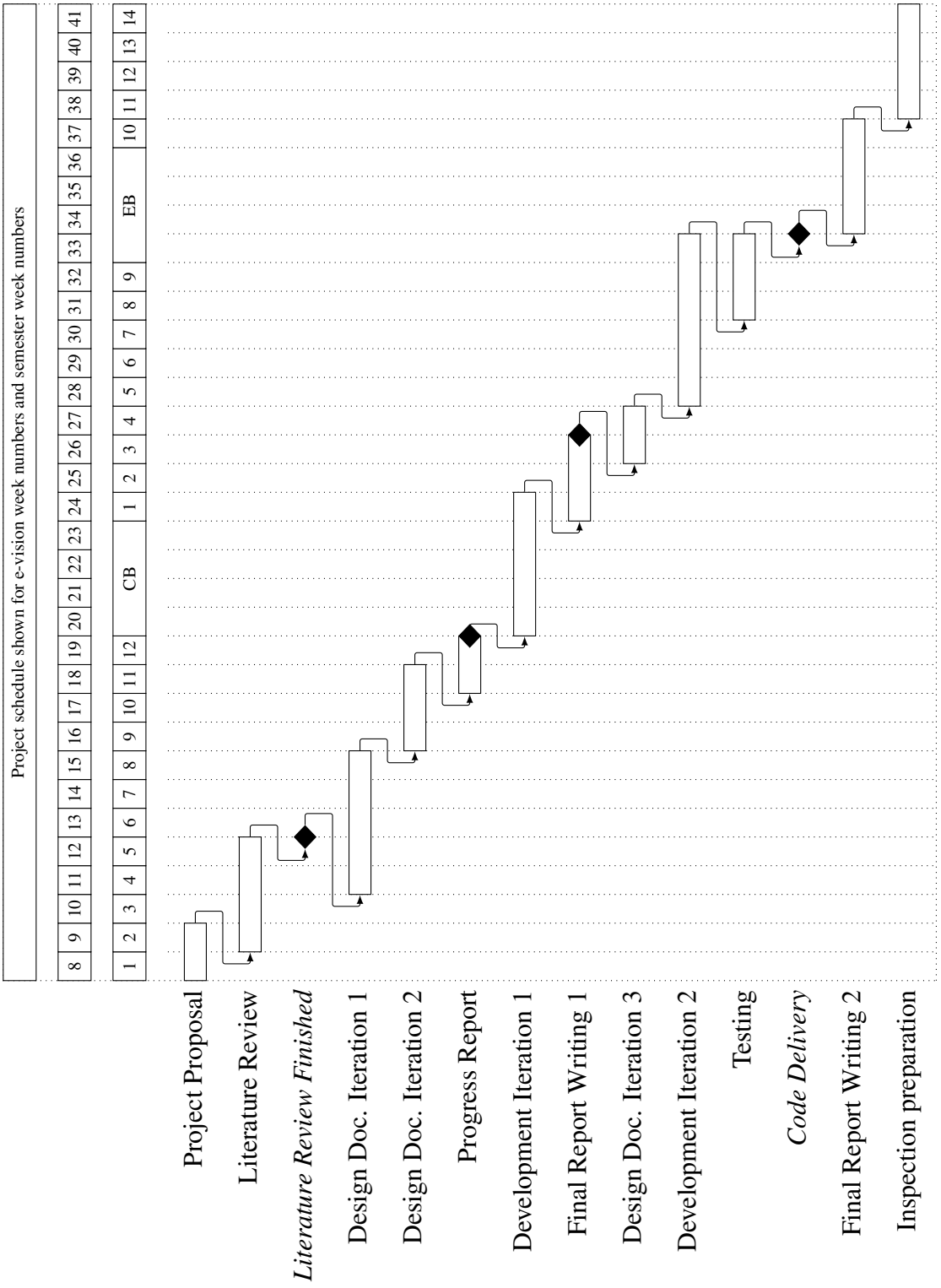

Figure 7: CODEX Gantt Chart, outlining the major tasks and deliverables

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.

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.

Software, F. C. (2011). Trello.