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

Christopher Alastair Irvine

registration: 100036248

1. Introduction

Over the years, *Dungeons and Dragons* (D&D) have entertained millions of players across the world (Gygax and Arneson (1974)). Throughout the various editions of this popular table-top role-playing game, the *Dungeon Master* (DM) has formed the core of every group. The role of a DM is to provide content for the players (party) to explore in addition to acting as a; referee, ally, enemy, narrator, historian, peace-keeper and above all friend (Ewalt (2014), Holmes (1980)).

The average D&D campaign occurs once a week, for four hours, until the campaign is finished. Typically, a DM will spend between 5 and 15 hours preparing content per week, but the work of a DM does not end with their preparation. During the session a DM needs to be alert and attentive to all their players, reacting accordingly to player input. The vast range of skills required for the role and the amount of time needed to carry out the duties of a DM create a barrier; preventing many capable players from trying their hand at being a DM, contributing to the shortage of DMs within the community.

CODEX aims to make Dungeon Mastering in D&D more accessible and simpler; by reducing the amount of time needed to prepare content and the number of statistics that the DM needs to track and make the general organisation of the group simpler. CODEX aims to achieve using a website that aims to have Progressive Web App (PWA) integration. D&D has been a global community that and a service such as CODEX will be of great benefit to them.

1.1. A note on Design Documents

The development of CODEX is a Software Engineering project which will include a Design Document[1](McElrath (2007)), allowing the solutions to be planned and tested before development begins.

1.2. Challenges and Strategies

We anticipate that CODEX will face a several challenges over the course of its development, strategies have been outlined below to deal with each of the foreseeable challenges.

1.2.1. Choosing a Software Engineering Model (SEM)

The most extensive problem that influences CODEX is the application of SEM to the development of CODEX, which will affect the structure of the Software. As the majority

[1]A Design Document is a technical guide for Developers to use before, during and after Developing a piece of Software.

of deliverables are non-programming, the chosen SEM will need to provide advantages that can be applied to all tasks. Nystöm details the development and testing of a SEM, called *Agile Solo* (Nyström (2011)). *Agile Solo* provides many time keeping and review benefits that can be applied to all tasks. If *Agile Solo* is unsuitable for CODEX, a substitute SEM can be used called *ExtremeProgramming* (XP). XP can be adapted for a Single Developer Project (ExtremeProgrammingChallenge (2006)).

1.2.2. GUI Design, Development and Testing

As the focus of CODEX is to reduce the burden of being a DM, it is vital that the GUI is well designed and easy to use. Galitz outlines the principle of good GUI design in this book (Galitz (2007)), which will serve as a guide for developing and testing the GUI for CODEX.

1.2.3. Client- and Server-Side Technologies

Maintaining a good balance of Server-side and Client-side technologies ensures the fastest load time possible. Server-side functions will complete on the server prior to the page loading, whereas Client-side functions will complete on the end user's machine after the page is loaded.

1.2.4. Offline Capabilities and Caching

Client-side technologies have the unique advantage of being able to run Offline, given the necessary data. Once the original website based application has been developed, CODEX will be converted to a PWA by JavaScript libraries - such as Node.js. The PWA version of CODEX will come with Offline capabilities that are defined within the Design Document.

1.2.5. Database Design, Development and Management

The database is hosted on a server and will be built in MySQL. Database communication is handled through a PHP or JavaScript back-end. The design of the database will be mapped out in the Design Document, so that a test database can be built to ensure integrity.

1.2.6. Algorithm Design

CODEX contains a multi-dimensional algorithm that will provide a DM with an accurate difficulty score for a fight. The original *Challenge Rating* system, detailed in the Monster Manual (Perkins, 2014) and Volo's Guide to Monsters (Mearls, 2016), will be used so that earlier features can function until the difficulty algorithm can be developed.

1.2.7. Gathering Data from the D&D Player Base

DMs are scattered across the globe; but via circulating a survey using Social Media, CODEX could ascertain the habits of the DM populace for the first time. Improving the features of CODEX during the design phase of the project.

1.3. Resources

We anticipate that two resources are critical to the success of CODEX. Firstly, the attainment of suitable server space, possibly purchased through companies such as DigitalOcean. Additionally, the Ethics Approval, provided by the Ethics Board, for the Survey before it can be distributed.

2. Literature Review

In this section, we will review academic papers and books that are relevant to CODEX. In particular this includes a review of Software Engineering Models (see Section 2.2), Web-App Architecture (see Section 2.3), Database Design (see Section 2.4) and Graphical User Interface (GUI) Design and Testing (see Section 2.5).

We start with a brief description of D&D.

2.1. CODEX and D&D

In this subsection we will gain an understanding of D&D so that we can understand the features of CODEX better. We examine the equipment needed to play the game (see Section 2.1.1), the core principles of D&D (see Section 2.1.2) and go through a worked example of D&D(see Section **??**).

2.1.1. D&D Equipment

D&D is a game that does not require a great amount of equipment. Every member of a D&D *group* needs the following:

- *Set of Die* (at least one 4-sided, 6-sided, 8-sided, 10-sided, 12-sided, 20-sided and 100-sided)
- *Character Sheet*
- *Pad of Paper & Pencil*

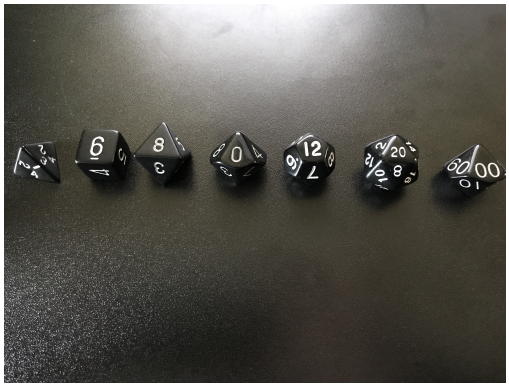
Additionally, in the group there should be a copy of each of the *D&D Core Books*. Any other equipment such as *Battle Mats and Character Tokens* are not necessary.

2.1.2. D&D Principles

Each group of people who play D&D is divided in two, the DM and the *Party*. The DM, as previously stated in the Introduction (Section 1), runs the game and creates the content for the Party to play through. Each member in the Party controls a *Player Character* (PC) for the duration of the game. The DM controls every other character, known as *Non-Player Character* (NPC). The PCs and NPCs interact through *encounters*, which can be friendly/hostile (see Section 2.1.4) and may escalate to *Combat* (for more details see Section 2.1.5). In Combat, characters can affect each other with *Attacks* and *Spells* to either inflict *Damage*, *Healing* or *Conditions*.

As discussed in the Introduction (Section 1), a DM will spend a portion of their time *preparing content* for their game.

(a) A typical D&Dset up



(b) The Die necessary to play D&D

Figure 1: These two figures show the typical equipment for a D&D game

CODEX is a tool for the DM of a group to use in order that they can enjoy running games of D&D to a higher degree. To do this, a limit has been set on CODEX scope. CODEX will not replace any dice rolling or the simple arithmetic that fuels the game. Instead CODEX will be an interface to track game statistics and quickly review rules and ability effects, allowing the DM to instil more narrative and creative content into the game.

2.1.3. Understanding Dungeons and Dragons

D&D is a game that is random by design, but there are two distinct areas to D&D *Exploration* and *Combat*. CODEX will largely deal with the Combat rules, however there is scope to support the DMs in the Exploration area as well. There is one overarching concept about D&D that needs to be understood when dealing with the rules of game. The rules are nothing more than a suggestion to the DM about how to run his or her game. Every group will have their own way of dealing with different situations, which could be an entirely new rule or an adjustment to one of the suggested rules. D&D is governed by the roll of dice. The most common die that is used is the 20-sided die, which dictates if an attack hits, or if you were able to persuade a character in game. It is used in both Combat and Exploration. The other dice (100, 12, 10, 8, 6 and 4 sided) are largely used for damage or the result of a spell. These dice are denoted by a lower-case 'd' and the number of side that dice possess, for example the 20-sided dice are denoted as **d20**.

2.1.4. Exploration

We shall begin with a brief look into Exploration, this is where the majority of DM preparation time is spent. DMs need to have landscapes, settlements, characters, dia-

logue, plot points and consequences for Players to interact with whilst they explore the world. The Dungeon Master's Guide (Mearls and Crawford, 2014a) contains a lot of information about the officially published settings, as well as tips and tricks for creating your own adventures. One of the most useful sources in the Dungeon Master's Guide for creating adventures for the Players to explore, is the multitude of random chance tables that allow you to create all the necessary information that forms the core of any adventure you might need for your game. These random chance tables are not limited to adventure story lines, other tables cover the characters the Players may interact with or the treasures players might receive at the end of an adventure. CODEX aims to include a planning feature that will integrate these tables to allow the busy DM to instantly create the framework for adventures and characters.

2.1.5. Combat

There are a lot more combat rules than exploration rules spread throughout the three D&D core books (Mearls and Crawford (2014a), Perkins (2014) and Mearls and Crawford (2014b)) . These rules are also more heavily suggested towards the DM as quite often the lives of the Player's Characters depend on these rules. Some DMs do choose to alter these rules to suit their game (Irvine, 2017), however CODEX will follow these rules as written. The Combat rules are the same for both the Player Characters and the enemies they face, and can be broken down into a set of simple calculations. For example, when seeing if an attack with a hammer will hit; the character's *proficiency* plus *strength modifier* will be added to the result of a *d20 roll*. This can be summarised to be:

The attack roll is 23, this is then compared against the targets *Armour Class (AC)*. Plate Armour has an AC of 18, so the attack roll of 23 is greater than the AC of the target and will hit (Mearls and Crawford, 2014b). This rule will not be included in CODEX's scope as it is taking away enjoyment from the players and is not helping the DM do their job. Instead it is rules like the *poisoned* rule that will be taken into account, where a suitable icon will be placed over the character who is *poisoned*, to remind the DM that character has the condition.

2.2. Software Engineering Models

A Software Engineering Model at its core, is a model for time management in relation to a project, particularly the Agile Software Engineering Models (Fowler and Highsmith, 2001) - such as Scrum (Schwaber, 1997) . The majority of Software Engineering Models are focused on controlling development time and code reviews, however some can extend to managing research and design work as well. For CODEX, we will look at two potential Software Engineering Models to support the research, design and development phases.

2.2.1. Agile for One

Agile for One is an adaptation of the Agile Manifesto to be more suitable for a single developer project, with the outcome of defining a new Software Engineering Model, called Agile Solo (Nyström, 2011). Agile Solo is similar to traditional Scrum in that there are weekly iterations inside a longer over-arching period of time, but rather than a two-week sprint encapsulating a daily scrum meeting, there is a monthly deliverable with a weekly iteration. At the end of each week, the Developer meets with the Customer or a Supervisor to discuss what occurred during the week and what the next focus should be. A longer review takes place at the end of each month. Agile Solo also recommends a daily meeting with a fellow developer to ensure that code quality is maintained.

2.2.2. Extreme Programming for One

Extreme Programming (XP) is a Software Engineering Model which emphasises the importance of constant feedback and maintaining simplicity in a system, allowing a project to thrive in an ever-changing environment. XP For One preserves the traditional stages of XP, but the *Pair Programming* recommendation is impossible to replicate in a solo developer project (ExtremeProgrammingChallenge, 2006). So instead XP For One suggests; that the developer has a friend or colleague constantly check that the developer has run the tests on the current section of work, that a log book is kept detailing the tests and development results and that the developer meet with a supervisor or client regularly to maintain the constant feedback.

2.2.3. Development of CODEX

The development of CODEX, which includes the non-programming activities (such as reports and presentations) will be achieved by utilising Agile for One Software Engineering Model for all aspects of the report. Trello, a website dedicated to project management, will aid in the management of tasks (Atlassian, 2017). Should Agile for One be unsuitable for the development of CODEX then Extreme Programming for One will be used.

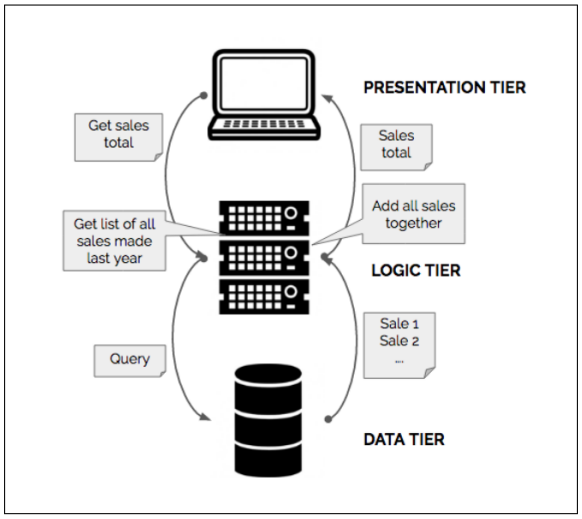


Figure 2: Overview of a three-tier application and its dependencies (De Groef, 2016)

2.3. Web-App Architecture

Web-App Architecture can be simplified down to three layers. As we can see from Figure 2 the first layer (Presentation Tier) is held locally on the Client's machine. This is what is known as 'Client-Side'. Here HTML, CSS and JavaScript is used to translate data into a format the Client will understand. The second layer (Logic Tier), here languages such as PHP or Node.js perform functions on raw data to generate web content. Traditionally the Logic Tier is hosted on a Server that Clients connect to, this is the 'Server-Side' The third layer (Data Tier) communicates with the Logic Tier through SQL Queries (De Groef, 2016).

2.3.1. Server- vs Client-Side Technologies

From Figure 2, we can create a rule for the division of CODEX' functions between Server- and Client-Side technologies. All data and logical functions should be hosted on the Server-Side and the display functions should be hosted on Client-Side. However, this does create a problem. This rule means that CODEX will have no Offline Functionality, which CODEX aims to have.

2.3.2. Test Driven Development (TDD) and Web-App Architecture

TDD is a common Software Engineering practice which dictates that when developing a function, the test for that function must be developed first so that the function can be developed to pass the test. The advantages of TDD are numerous; code is tidier, documentation is more thorough and the lower number of bugs are the main advantages

towards CODEX. TDD is typically used for functional programming only therefore, it is not suited towards Web-Based languages such as HTML and CSS (Astels, 2003). But we can use TDD with PHP and JavaScript. To simplify the development of CODEX we will not use PHP as JavaScript can run on both the Server- and Client-Side of the app.

2.4. Database Design

Traditional Database Design is performed using *Entity Relationship Diagrams* which tables of objects and how they are related to one-another. From there a SQL database can be built and populated on a server, ready for access (Teorey et al., 2008).

2.4.1. Databases for Web-Based Internet Applications

The problem with this way of designing and building a database is speed. When you have a multi-tiered system, such as a Web-App (see Figure 2), the access times to the database is too slow for modern users (Chen et al., 2012). The solution to this issue is to have a multi-module database with the minimal amount of overhead as possible.

2.5. Graphical User Interfaces

The GUI is, arguably, one of the most important party of any system where a user interacts directly with any part of the system. A good GUI is easy to understand and simple in its design. Ideally, every GUI should follow Norman's 8 Design Principles (Norman, 1983), but in the most systems compromises must be made.

2.5.1. GUI Design

From a Software Engineering Perspective, GUI Design should come before anything else is designed. Once the system requirements have been set, *low-fi* prototyping should take place for the GUI (this is usually sketched on using pen and paper). Once the *low-fi* has been tested and approved, the prototypes of the GUIs become increasingly *hi-fi* until you have reached an entirely digital prototype. GUI Design follows this methodology because *low-fi* prototypes are cheaper and quicker to produce than *hi-fi* prototypes. Once the GUI has been designed, functions and classes are created to fit the GUI (Galitz, 2007).

2.5.2. GUI Testing

At each stage of GUI prototyping, usability testing needs to occur. This can either be a Heuristic Evaluation, with "expert users" (such as Developers, Trained Heuristic Evaluators or similar system users) or a Usability Study. In the early *low-fi* stages,

usually only one of these tests are applied, but as the prototype gets *hi-fi* both tests are employed to provide the greatest level of feedback possible (Galitz, 2007).

2.5.3. Mobile GUI Design and Testing

Mobile GUIs have been around for a while, ever since the rise of the smart phone in late 00's. At first Mobile Apps and Websites had to be separate, dedicated entities (Nielsen and Budiu, 2013). There have been advances in Web-Technologies since then allowing for the same service to scale down to a smaller screen side.

A. CODEX Gantt Chart

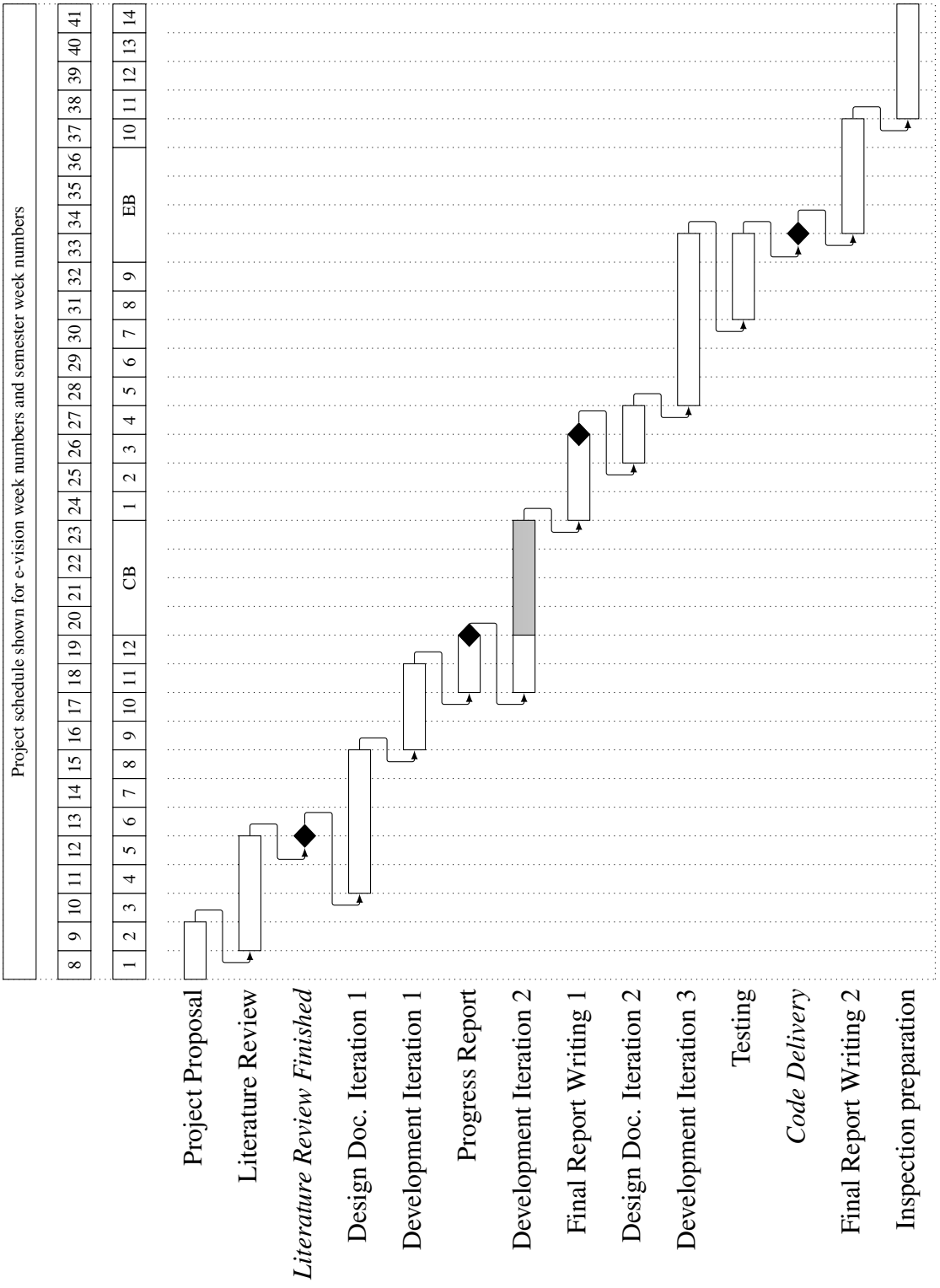


Figure 3: CODEX Gantt Chart, outlining the major tasks and deliverables

References

Astels, D. (2003). *Test Driven Development: A Practical Guide*. Prentice Hall Professional Technical Reference.

Atlassian (2017). Trello.

Chen, Y.-C., Jeng, J.-Y., Huang, T.-S., and Lin, P. (2012). Design and implementation of database schema evolution for service continuity of web-based internet applications. In *Network Operations and Management Symposium (APNOMS), 2012 14th Asia-Pacific*, pages 1–4. IEEE.

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

Ewalt, D. M. (2014). *Of dice and men: The story of Dungeons & Dragons and the people who play it*. Simon and Schuster.

ExtremeProgrammingChallenge (2006). Extreme programming for one.

Fowler, M. and Highsmith, J. (2001). The agile manifesto. *Software Development*, 9(8):28–35.

Galitz, W. O. (2007). *The essential guide to user interface design: an introduction to GUI design principles and techniques*. John Wiley & Sons.

Gygax, G. and Arneson, D. (1974). *Dungeons and dragons*, volume 19. Tactical Studies Rules Lake Geneva, WI.

Holmes, J. E. (1980). Confessions of a dungeon master. *Psychology Today*, 14(6):84–94.

Irvine, C. (2017).

McElrath, R. (2007). Xml legal document utility software design document.

Mearls, M. (2016). *Volo's Guide to Monsters*. Wizards of the Coast.

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. and Budiu, R. (2013). *Mobile usability*. MITP-Verlags GmbH & Co. KG.

Norman, D. A. (1983). Design rules based on analyses of human error. *Communications of the ACM*, 26(4):254–258.

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.

Teorey, T. J., Buxton, S., Fryman, L., Güting, R. H., Halpin, T., Harrington, J. L., Inmon, W. H., Lightstone, S. S., Melton, J., Morgan, T., et al. (2008). *Database design: know it all*. Morgan Kaufmann.

Literature review

Introduction: brief description of project, areas of knowledge required, roadmap	First	2.1	2.2	3	Fail
Discovery of suitable quantity and quality of material	First	2.1	2.2	3	Fail
Description of key issues and themes relevant to the project	First	2.1	2.2	3	Fail
Evaluation, analysis and critical review	First	2.1	2.2	3	Fail

Quality of writing

Clarity, structure and correctness of writing	First	2.1	2.2	3	Fail
Presentation conforms to style (criteria similar to conference paper reviews)	First	2.1	2.2	3	Fail
References correctly presented, complete adequate (but no excessive) citations	First	2.1	2.2	3	Fail

Revised Workplan (if applicable)

Measurable objectives : appropriate, realistic, timely	First	2.1	2.2	3	Fail

Comments

Supervisor: Dr Katharina Huber

Markers should circle the appropriate level of performance in each section. Report and evaluation sheet should be collected by the student from the supervisor.