ENGR302 Project Management
Project Proposal and Requirements Document

Group S4

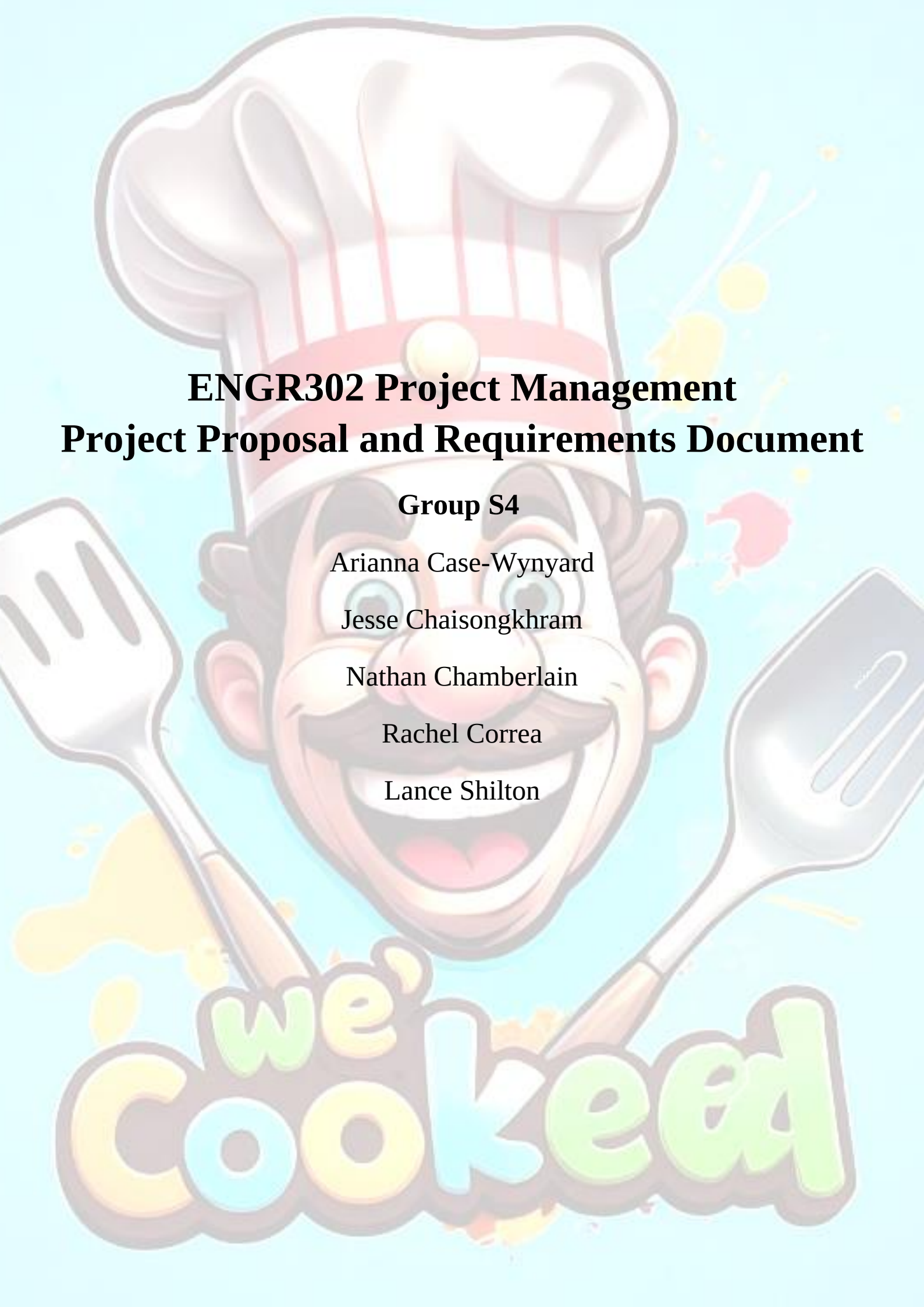
Arianna Case-Wynyard

Jesse Chaisongkhram

Nathan Chamberlain

Rachel Correa

Lance Shilton



Introduction:

This document details the project requirements for our game concept, *we'reCooked*, a cooking game designed to teach the agile software engineering method, Scrum. All requirements may be adjusted based on feedback or recommendations from Chris, Karsten, or the tutor, with the agreement of the group.

1. Plan

Top down 2D team game where players must collaborate to serve cooking orders to customers. Phases of Scrum (an agile methodology) are implemented into the cooking game for the cooking orders e.g. requirements phase = getting the order, development phase = making the order etc. Ingredients can be over/undercooked incorporating a quality aspect into the game. As additional, bugs are also invading the kitchen and the chefs have to smack the bugs to get rid of them otherwise the bugs will start eating the food, decreasing the quality.

2. Game

This plan showcases how Scrum methodology will be incorporated into each aspect of the game, and how players will face opportunities and challenges that incentivise them to learn Scrum and learn how to use it in an effective manner.

2.1.1 Sprint planning

- In this event, the team estimates the work to be completed in the next Sprint. Members define Sprint Goals that are specific, measurable, and attainable. At the end of the planning meeting, every Scrum member knows how each Increment can be delivered in the Sprint.

2.1.2 Cooked Sprint Planning

- In this event, at the beginning of each sprint (day), the team estimates how much work they can complete by the end of the sprint (day). Each day, the restaurant will have a minimum number of orders that the team must sell to break even and continue to the next round, but the team can set higher sprint goals so that they can make profit which can be used to purchase faster ovens/cooking utensils. The team must decide on a specific amount of profit that they believe they can attain by the end of the day, measured by the number of orders that they complete.

2.2.1 Sprint

- A Sprint is the actual time period when the Scrum Team works together to finish an Increment. Two weeks is the typical length for a Sprint but can vary depending on the needs of the project and the team. The more complex the work and the more unknowns, the shorter the Sprint should be.

2.2.2 Cooked Sprint

- After the sprint goals are set, the sprint begins as a 5-minute day where the shop opens and the Scrum Team work together to complete their orders. Unknowns are implemented using variance in the length and complexity of the orders, so that the team has to adapt and work around this.

2.3.1 Daily Scrum or stand-up

- A Daily Scrum is a short meeting in which team members check in and plan for the day. They report on work completed and voice any challenges in meeting Sprint Goals. It is called a stand-up because it aims to keep the meeting as short as practical—like when everybody is standing.

2.3.2 Cooked Daily Scrum

- A time-limited, 30 second meetings will be held at the start of each day during sprint planning that the team can utilise to type in the chat to plan which cooking stations need more players, which are doing well and don't need as much help, etc

2.4.1 Sprint Review

- At the end of the Sprint, the team gets together for an informal session to review the work completed and showcase it to stakeholders. The Product Owner might also rework the Product Backlog based on the current Sprint.

2.4.2 Cooked Sprint Review

- At the end of each sprint (day), the team are shown stats from their day such as how many orders they completed, how long it took them to finish the orders, etc, to review. The restaurant owner also reworks the product backlog (number of orders needed to sell to break even).

2.5.1 Sprint Retrospective

- The team comes together to document and discuss what worked and what didn't work during the Sprint. Ideas generated are used to improve future Sprints.

2.5.2 Cooked Sprint Retrospective

- At the end of the sprint (day), the players are given a chat lobby that they can utilise to discuss what worked and didn't work during the sprint, as well as discuss how they might improve their restaurant the next day.

3. Requirements

Stakeholders

- **Product Owner/Tutor:** Ensures alignment with educational goals.
- **Development Team:** Implements the game and its features.
- **Players (Students):** Primary users, providing feedback and testing.

Specific - Measurable - Achievable - Relevant - Time Bound

3.1 Functional

A. Game must be web-based, accessible via browser.

- Unity, a cross-platform game engine, will be used to develop this functionality within the game. This will be measured by whether the game is accessible to players who don't have the code files downloaded via their browser. Unity is designed to make creating web-based applications, specifically games, straightforward and easy, which makes this achievable to learn and set up by Week 6.
-

B. Game will be multiplayer with a minimum of 2, and maximum of 6 players.

- Unity Multiplayer connectivity must be researched and implemented by Week 7 of the project.
- Need to ensure ECS Computers/network will allow interconnectivity to demonstrate the game to tutors.
- Collaboration between players is essential for the team to win.

C. **Game will teach an Agile methodology, specifically agile methodologies targeted at software development such as Scrum, by incorporating it into the gameplay.**
    - The gameplay will incorporate Agile/Scrum concepts such as sprints. For example, a batch of customers enter the store, all of their orders are added to the sprint, and all players collaborate to cook all of the orders by the end of the sprint.
    - The game must teach this methodology to first year SWEN students by incorporating the aspects of this software agile methodology within the game, without requiring any sort of complex software knowledge, so that even first-trimester students who have no prior coding knowledge will be able to play and learn.

3.2 Non-functional requirements

A. **The game will be able to be demonstrated within 10-15 minutes.**
    - The tutor/marker must get a clear idea of the game within 10-15 minutes. The game will therefore have 5–10-minute rounds (‘days’), which clearly showcase all aspects of the gameplay. Each round will have the same principles, they just become progressively more difficult.

B. **Our game needs to be safe for under-18-year-olds.**
    - Since our game is being used to teach first year SWEN/CYBR students Agile/Scrum, the game cannot have any NSFW content unsuitable for under-18-year-olds.

C. **The game will run smoothly on standard web browsers without requiring high-end hardware.**
    - Optimize the game to run efficiently on standard web browsers (Chrome) on mid-range computers.
    - The game will be tested on various mid-range hardware (e.g. laptops and desktops) to ensure it is accessibility for all the students.
    - Utilizing Unity's built-in optimization tools and to follow to best practices for web-based game development.
    - Optimization and testing for standard web browsers will be completed by Week 9 – Week10.

4. Git Strategy and Methodology

4.1 Git Strategy

- Share documentation in wikis
- Create branches off issues for traceability
- Create feature branches off main rather than committing directly to main.
- All merge requests require approval from one other member

4.2 Methodology

- Agile Scrum
- Weekly sprints

5. Roles

Role	Member
GUI	Rachel + Nathan
Domain	Nathan
Multiplayer	Lance
Documentation/All	Arianna
Testing/Primary Reviewer	Jesse

6. Timeline

6.1 Gathering Requirements

Week 1 - Week 3

Document project requirements from the product owner (Karsten) from the assignment brief and during lectures

Research Agile and Scrum Methodology

Formulate a game that meets all the requirements

6.2 Planning & Research

Week 4

Break down requirements into manageable tasks, and add them to the gitlab repository as issues

Familiarise the team with Unity, a cross-platform game engine that we will be using to produce our game

6.3 Design & Develop

Week 5 - Week 10

Each week/fortnight, set up the goals that we will aim to achieve this sprint, and allocate our resources appropriately

Produce the game using agile methodology

Start creating basic test cases

6.4 Testing

Week 10 - Week 11

6.5 Review & Report

Week 11 - week 12.

Discuss strengths and weaknesses within the project, areas for improvement, and make recommendations for the next project

Document these findings in a report