SE 4G06: Software Requirements Specification
Team #12 - CodeChamp

Kanugalawattage, Anton Subedi, Dipendra
Rizkalla, Youssef Leung, Tamas Zhao, Zhiming

October 4, 2022

Table 1: **Revision History**

Date	**Version**	**Notes**
Oct. 4, 2022	0.2	Prioritization, Traceability and Phase In Plan for Requirements; Appendix Reflections; Minor changes to requirements; State Machine Formal definition
Sept. 27, 2022	0.1	Initial Version

Contents

1 Project Drivers **1**
- 1.1 The Purpose of the Project . . . 1
- 1.2 The Stakeholders . . . 1
  - 1.2.1 The Client . . . 1
  - 1.2.2 The Customers . . . 1
  - 1.2.3 Other Stakeholders . . . 1
- 1.3 Mandated Constraints . . . 1
  - 1.3.1 Solution Constraints . . . 1
  - 1.3.2 Budget Constraints . . . 2
- 1.4 Relevant Facts and Assumptions . . . 2

2 Functional Requirements **2**
- 2.1 The Scope of the Work and the Product . . . 2
  - 2.1.1 The Context of the Work . . . 2
  - 2.1.2 Work Partitioning . . . 2
  - 2.1.3 Individual Product Use Cases . . . 2
- 2.2 Functional Requirements . . . 3
  - 2.2.1 Priority 0 . . . 3
  - 2.2.2 Priority 1 . . . 4
  - 2.2.3 Priority 2 . . . 5
- 2.3 User State Finite State Machine . . . 6
  - 2.3.1 State Machine Definition . . . 6
- 2.4 Phase In Plan . . . 7

3 Non-functional Requirements **7**
- 3.1 Look and Feel Requirements . . . 7
- 3.2 Usability and Humanity Requirements . . . 7
- 3.3 Performance Requirements . . . 8
- 3.4 Operational and Environmental Requirements . . . 8
- 3.5 Maintainability and Support Requirements . . . 8
- 3.6 Security Requirements . . . 8
- 3.7 Cultural Requirements . . . 8
- 3.8 Legal Requirements . . . 8
- 3.9 Health and Safety Requirements . . . 9

4 Traceability Table **9**

5 Project Issues **9**
5.1 Open Issues . . . 9
5.2 Off-the-Shelf Solutions . . . 9
5.3 New Problems . . . 9
5.4 Tasks . . . 10
5.5 Migration to the New Product . . . 10
5.6 Risks . . . 10
5.7 Costs . . . 10
5.8 User Documentation and Training . . . 10
5.9 Waiting Room . . . 11
5.10 Ideas for Solutions . . . 11

6 Appendix **12**
6.1 Identifying Required Knowledge . . . 12
6.2 Acquiring Required Knowledge . . . 13
6.3 Symbolic Parameters . . . 14

List of Tables

1 **Revision History** . . . i
2 Phase In Plan for Functional Requirements . . . 7
3 Traceability Table for Functional and Non-Functional Requirements . . . 9

List of Figures

1 Diagram for User State Finite State Machine . . . 6

Changes from Volere Simplified Template

- Added Phase in Plan
- Added Requirement Priorities
- Added Traceability Table

Naming Conventions and Terminology

- Leetcode: An online platform to learn data structures and algorithms. It provides the user with the ability to solve questions from the database by giving them the ability to write code and compile within the platform. It will also provide feedbacks by giving number of test cases passed.
- DMOJ: Another online platform to learn data structures and algorithms
- SRS: Software Requirement Specification
- UI: User interface

1 Project Drivers

1.1 The Purpose of the Project

Practicing coding the traditional way can be daunting. The current most popular method to learn is to use a problem database site like LeetCode. This method of learning can often feel tiring and endless. There are over 2000 problems on the website which can be intimidating to many new coders. CodeChamp will introduce a collaborative and fun way to interact with your friends while improving your algorithmic skills.

1.2 The Stakeholders

1.2.1 The Client

- Dr. Spencer Smith
- TAs of SE 4G06

1.2.2 The Customers

- Learners who wants to practice and improve on their data structures & algorithmic skills.
- Existing users of popular coding practice sites like Leetcode and DMOJ.
- Groups looking to compete and practice their coding skills.

1.2.3 Other Stakeholders

- Developers on the project

1.3 Mandated Constraints

1.3.1 Solution Constraints

- System shall be accessible through the internet to any device with a browser.

1.3.2 Budget Constraints

- For the scope of the project, deployment and hosting shall cost $0.

1.4 Relevant Facts and Assumptions

- Users have and will maintain a steady internet connection throughout the game.
- User is able to navigate a computer or a smart device.
- Input from user will be from web input devices(keyboard and mouse).
- Output from the system will be displayed on user's screen through the web application.

2 Functional Requirements

2.1 The Scope of the Work and the Product

2.1.1 The Context of the Work

- This application will allow 2-20 users in a lobby to compete against each other across multiple browsers.

2.1.2 Work Partitioning

- Server will maintain all match data
- Server will compile user's submissions and evaluate the correctness of the submission.
- User clients will request match data and match state from server
- User clients will communicate with server to update match state

2.1.3 Individual Product Use Cases

- User creates an account
- User logins into application

- User creates a lobby
- User creates invite code to invite friends
- User joins lobby with an invite code
- User finds lobby to join
- User writes code in editor
- User submits code for compilation
- User returns to main menu after winning/losing
- User views previous match history

2.2 Functional Requirements

2.2.1 Priority 0

FR.1 The system shall match-make users into a match.

FR.2 The system shall start a match when 20 players are match-made.

FR.3 The system shall have 3 rounds per match.

FR.4 The system shall display one coding problem per round.

FR.5 The system shall allow users to submit code through the web interface.

FR.6 The system shall compile the user's code.

FR.7 The system shall run the user's code against the test cases for a problem.

FR.8 The system shall display the result of running the user's code against the test cases to the user.

FR.9 The system shall qualify the first 10 users in the first round to solve the coding problem into the second round and shall disqualify the remainder.

FR.10 The system shall qualify the first 5 users in the second round to solve the coding problem into the third round and shall disqualify the remainder.

FR.11 The system shall declare the first user to solve the coding problem in the final round as the winner and shall disqualify the remainder.

FR.12 The system shall allow developers to create, modify and delete coding problems.

FR.13 The system shall allow developers to create, modify and delete test cases for a coding problem.

2.2.2 Priority 1

FR.14 The system shall detect malicious code.

FR.15 The system shall prevent malicious code from compiling.

FR.16 The system shall display the result of the code's compilation and report any errors to the user.

FR.17 The system shall allow users to sign up using a username, email and password.

FR.18 The system shall allow users to log in using a username and password.

FR.19 The system shall display an error if the user fails to log in.

FR.20 The system shall track the code's running time in seconds and its memory usage in megabytes.
Rationale: An important part of an algorithm is its time and memory complexity. For some problems, the developers may want to disallow solutions that are not of the optimal time or memory complexity.

FR.21 The system shall time-out any code that exceeds the time or memory limit for a coding problem.
Rationale: Inputs that consume more resources than needed should be timed out for performance and safety of the system.

FR.22 The system shall allow developers to define a time in seconds and a memory limit in megabytes for each coding problem.

2.2.3 Priority 2

FR.23 The system shall allow users to create a lobby link.
Rationale: Links will be used to allow users to share a lobby to other users

FR.24 The system shall allow users to join a lobby using a link.

FR.25 The system shall allow users to view a history of their previous matches.
Rationale: Allows the user to see their history and progress to track their skill changes over time.

FR.26 The system shall display the user's win percentage.
Rationale: Allows the user to track their history and progress to quantify their skills against the competition.

FR.27 The system shall display the rounds in which the users qualified or disqualified for a match's history.
Rationale: Looking at a previous matches' result is helpful for the user experience, as it can help them recognize what they did wrong and what they could do to improve in the next one.

FR.28 The system shall display the coding problem for each round in a match's history.
Rationale: Looking at previous problems can help users identify their weaknesses and give them a second chance to solve a problem which they could not previously solve in time.

FR.29 The system shall track and display the top 100 users with the largest number of wins.
Rationale: Recognizing the top users can motivate others to practice and give an incentive for improvement, which is important for our user experience.

2.3 User State Finite State Machine

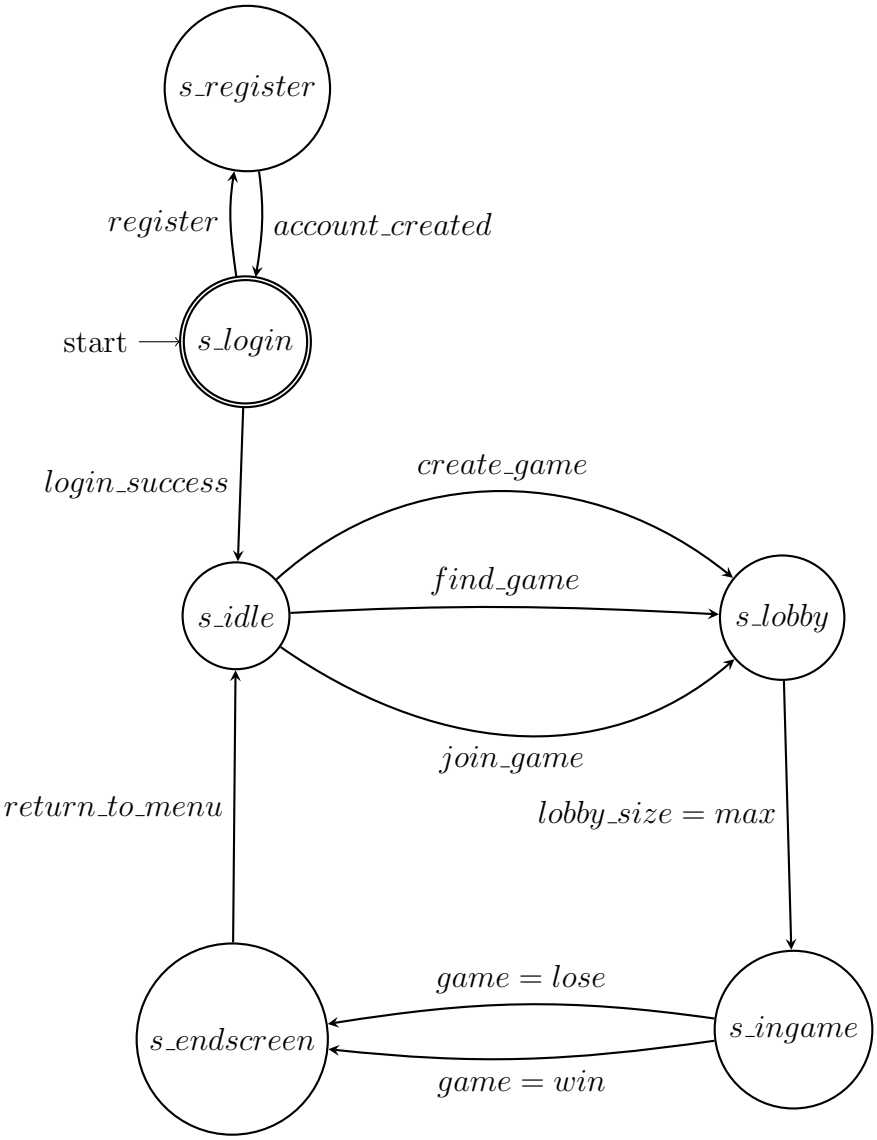

Figure 1: Diagram for User State Finite State Machine

2.3.1 State Machine Definition

Notation is based on standard academic conventions (1).

$Let\ M = (Q, \Sigma, \delta, s, F)$
$Q = \{s\_login, s\_register, s\_idle, s\_lobby, s\_ingame, s\_endscreen\}$
$\sigma = \{account\_created, register, login\_success, return\_to\_menu, create\_game,$
$find\_game, join\_game, lobby\_size = max, game = lose, game = win\}$
$\delta =$ Shown in Figure 1
$s = s\_login$
$F = \{s\_login\}$

2.4 Phase In Plan

Priority Level	Date
0	Nov 14th 2022
1	Jan 7th 2022
2	March 10th 2022

Table 2: Phase In Plan for Functional Requirements

3 Non-functional Requirements

3.1 Look and Feel Requirements

NFR.1 The interface of the system shall be easy to read and the elements are not clumped together.

NFR.2 The elements on the interface shall be organized and placed in a logical way.

3.2 Usability and Humanity Requirements

NFR.3 People with basic computer understanding shall be able to navigate the system.

NFR.4 The system shall be simple to understand and user should spend less than 5 minutes to understand what each part of the system does.

NFR.5 The system should be easily accessible so that the user can navigate around the system without the use of mouse.

3.3 Performance Requirements

NFR.6 The system shall respond to a user's action within 2 seconds.

NFR.7 The system shall process and give results base on user's inputs within 10 seconds.

NFR.8 The system shall be operating 24 hours a day, 7 days a week.

3.4 Operational and Environmental Requirements

NFR.9 The system should run on major browsers and devices that support these browsers.

3.5 Maintainability and Support Requirements

NFR.10 The system shall provide the ability for developers to add questions and test cases when needed.

3.6 Security Requirements

NFR.11 The system shall require users to register their own accounts in order to use the system.

NFR.12 The system shall deny the user access if the user fails to login.

3.7 Cultural Requirements

NFR.13 The system shall not have anything that is or suggests cultural inappropriate content to society.

3.8 Legal Requirements

NFR.14 The system shall be protected by GNU General Public License (GPL).

3.9 Health and Safety Requirements

NFR.15 The system shall not have any flashing lights that can put the user under the risk of epilepsy or seizure.

4 Traceability Table

Non-Functional Requirements	Functional Requirements	Rationale
NFR.1 NFR.2 NFR.3 NFR.4 NFR.5 NFR.15	FR.4 FR.8 FR.16 FR.19 FR.26 FR.27 FR.28 FR.29	Requirements that are focused on interface design requirements.
NFR.6 NFR.7	FR.6 FR.7 FR.14 FR.15 FR.20 FR.21 FR.22	Requirements that are focused on performance of user experience and code compilation.
NFR.11 NFR.12	FR.17 FR.18 FR.19	Requirements that are focused on account security and management

Table 3: Traceability Table for Functional and Non-Functional Requirements

5 Project Issues

5.1 Open Issues

- Judging and evaluation of a user submission
- Safety, managing malicious code submissions

5.2 Off-the-Shelf Solutions

N/A

5.3 New Problems

N/A

5.4 Tasks

- Back-end architecture design
- Front-end UI design
- Communication schema design between front-end and back-end
- Database schema design
- Unit tests for front-end and back-end
- End to End integration tests

5.5 Migration to the New Product

N/A

5.6 Risks

- Multiple concurrent communications between front-end and back-end
- Back-end server may not be able to handle multiple concurrent of compilation
- Distributed Denial-of-Service attacks
- Malicious code injections

5.7 Costs

- Back-end hosting server costs
- Front-end hosting server costs
- Database hosting costs

5.8 User Documentation and Training

N/A

5.9 Waiting Room

N/A

5.10 Ideas for Solutions

For concurrent communications and concurrent code compilations, vertical scaling for hardware is an easy fast method to remove the limitations of concurrency. However this leads to future higher costs. Load balancing can be used to allow for horizontal scaling of system.

For costs, many options exists for free/hobby tier project hosting, allowing for cost savings.

6 Appendix

6.1 Identifying Required Knowledge

What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain-specific knowledge from the domain of your application, software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, writing, presentation, team management, etc. You should look to identify at least one item for each team member.

1. **Presentation**: Learning how to speak confidently in front of an audience allows for the best demonstration of our project. As we present our capstone, presentation skills will allow the team to reduce miscommunication and fully get our message across. Not only will this benefit the capstone project, it is a transferable skill to the real world.

2. **Project Management**: Project management for a team of 5 with a large project isn't something the team is familiar with and it is key for completing our project successfully. This includes splitting off features/work among us, reviewing each other's pull requests and planning meetings or sprints.

3. **Two way web communication technology**: Learning this technology will be necessary for the success of our project as it is required to make our project collaborative. This technology will allow the project to communication between server and client and will be a key part of our project.

4. **Continuous Integration and Continuous Delivery (CI/CD) - Zhiming Zhao**: CI/CD connects development and operation activities together, it will enforce automation in developing, building, testing and code deployment. CI/CD compiles the incremental code changes made by developers, then package them as software deliverable. Automated tests are running constantly to verify the software functionality, and automated deployment services deliver them to end users.

5. **System Design - Youssef Rizkalla**: Architecting a good design for the back-end and front-end of the system is essentially to creating a

friction-less environment that allows us to fulfill and test the requirements according to their deadlines. Learning design patterns and applying them when needed will be necessary, as well as learning how to enforce coding standards when reviewing and writing code to maintain codebase health.

6.2 Acquiring Required Knowledge

For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? From the identified approaches, which will each team member pursue, and why did they make this choice?

1. **Presentation - Dipendra Subedi** An approach to develop presentation skills is to practice speaking in front of audiences. In front of an audience, we can increase the confidence for public speaking, practice maintaining eye-contact, and speaking coherently. Another approach is to participate as a member of the audience in a presentation with great speakers. Being able to mimic exemplary presentation skills can be a guide when we are presenting ourselves. Dipendra will work on this skill as he is the design lead, so it is crucial to be able to effectively communicate ideas. When presenting the capstone project, this will also help since he will take more of a speaking role to describe the system.

2. **Project Management - Tamas Leung**: In order to learn project management is through researching and testing out different project management strategies. This involves learning tools such as GitHub project board/kanban boards. This can be practiced by assigning priorities and weights to tasks and ensuring other teammates have a good weight of tasks per week. Tamas will work on this skill as he is the scrum lead. He will ensure that every task is on a timeline and are on track to be finished as stated on the development plan.

3. **Two way web communication technology - Anton Kanugalawattage**: This skill could be acquired by reading and following examples of existing implementation using this technology. Another approach to acquire this skill is to read the documentation or blogs of how this technology is used in systems (networking classes, technology company's engineering

blogs). Anton will pursue this skill as he has worked on a project which has utilized a similar technology before. Also, as he is the front-end lead this technology will be a crucial part of the front-end and back-end integration.

4. **CI/CD - Zhiming Zhao**: There are a lot of benefits to adapt CI/CD. First of all, there will be reduced risk on delivery. The automated tests will test the code changes before it's deployed, this will result in a higher product quality and low rates of bugs in production. The software will ship quickly and more efficiently since CI/CD pipelines move applications from coding to deployment at scale. The team will also have improved productivity since everything is moving at a faster pace.

5. **System Design - Youssef Rizkalla**: This can be developed by observing and critiquing open-source projects, as well as projects previously worked on. Youssef will work on this as he is the back-end lead and main code-reviewer. By doing this, he can identify issues that have occurred in the past and develop a design to get around them. Additionally, design patterns can be learned from official documentation or trustworthy blog posts that are published by professional Software Engineers. When reviewing code, Youssef can ensure that others are following the correct standards and design patterns. Additionally, reiterations of the design can be done as needed if engineers feel that features are taking longer to develop and/or test than they should be.

6.3 Symbolic Parameters

N/A

References

[1] N.R. Satish, University of California, Berkeley. (n.d.). Finite State Machine. Our Pattern Language. Retrieved October 3, 2022, from https://patterns.eecs.berkeley.edu/?page_id=470