Software Requirements Specification

For the Technical Talent Finding System

Version 0.3
Prepared by Jack Harris
Team 10 of the 2024 Oxford CompSci Second Year Group Design Practical
created 2024/02/27

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Jack	2024/03/12	Adding prototypes and decluttering document	0.3
Jack	2024/03/12	Added “Security” non-functional requirement.	0.4

1. Introduction

This document will act as the specification from which Team 10 shall produce a talent finding system for Microsoft. This system will enable Microsoft to find talented individuals who would be suited to working in a software development role, and it will do so in ways specified in this document.

1.1 Document Purpose

The purpose of this document is twofold: to provide a specification from which the developers can produce the intended software, and to provide a specification of the system on which the developers and the client agree.

1.2 Product Scope

The purpose of the product is to enable Microsoft to find talented potential hires for graduate software development roles. It will do so by providing a series of puzzles which test algorithmic thinking and advanced coding, similar to those found on websites like LeetCode, HackerRank and AlgoExpert. It will also provide ranking metrics for the performance of the users of the system, alongside an overall ranking of the users. There will also be gamified elements which provide extrinsic rewards for the users.

1.3 Definitions, Acronyms and Abbreviations

“The developers”, “Team 10”, “we”: The development team for the project, consisting of Jack Harris, Rory Kemp, Raul Sheth, Luke Tan, Nathan Hardcastle and Georgi Petkov.

“The client”, “Microsoft”: The client for this project, consisting primarily of Lilia Georgieva, but including Microsoft as a whole.

“The users”, “the applicants”, “the candidates”: The people who will be solving the puzzles in this system and participating in the rankings. In other words, the people the client will want to consider as potential applicants for graduate software roles.

2. Product Overview

2.1 Product Perspective

This product is being produced as part of the 2024 Group Design Practicals done by students in their second year of a Computer Science course at the University of Oxford. It emulates the interview process of most software companies, wherein candidates are asked a number of technical questions and puzzles in order for the interviewer to assess the ability of the candidate. The product shall be completely standalone, and not be part of any wider project.

2.2 Product Functions

The product shall:

- Provide a series of puzzles to the user for them to complete
- Provide a place for users to upload their code for their answers
- Provide a series of metrics from which the users' performance can be assessed
- Provide an overall ranking of the users
- Provide a login/accounts system for the users, so their progress can be kept between sessions
- Provide gamified elements within the puzzle to provide extra rewards to users and motivate them to keep trying the puzzles.

2.3 Product Constraints

The product will have the following constraints:

- The product will have to run in the web browser.
- The product will have to have its backend running on a lightweight server.
- The product will have to be developed within 2 months.
- The product will have to be GDPR-compliant.
- The product will be developed by six second-year undergraduates.

2.4 User Characteristics

Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.

User Class 1: Candidates These users will primarily be computer science graduates. They will be proficient in algorithmic thinking, and will have enough capability in a programming language to be able to produce computer programs that allow them to answer the puzzles.

User Class 2: Employers These will primarily be recruiters at Microsoft looking for hidden talent. They will be very proficient in algorithmic thinking and will have some degree of experience within the software industry. They will be able to assess the quality of the code of applicants.

2.5 Assumptions and Dependencies

The following assumptions are present:

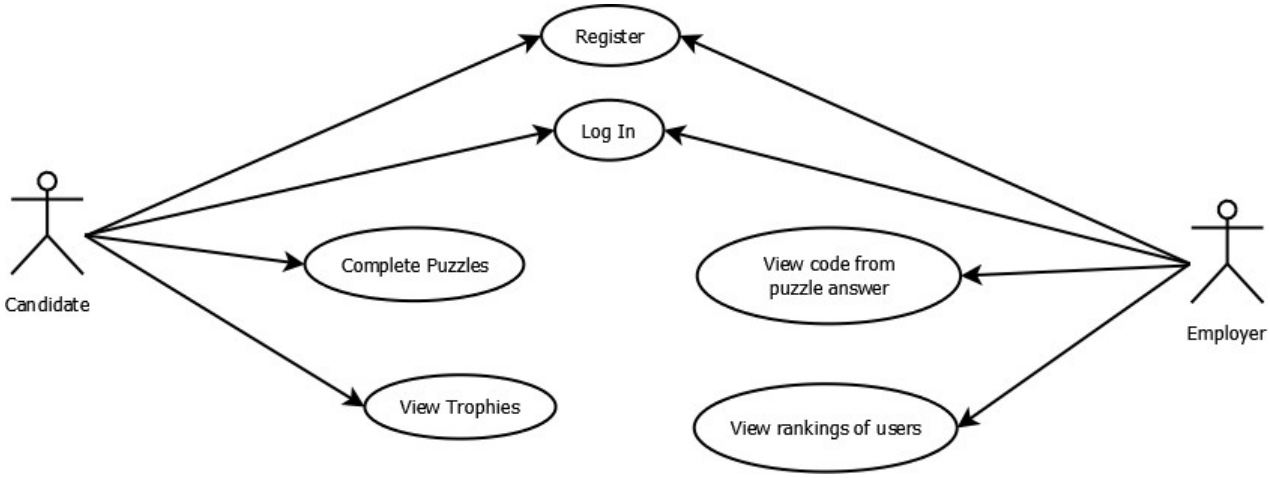
- Candidates will have access to the internet and a web browser
- The metrics specified in this document will be useful in assessing the ability of users.

Since we will be making most of this from scratch, there are very few external assumptions about the operation of other software components.

3. Requirements

3.0 Use Cases

Use Case Diagram



Use Case Diagram

Use Case Descriptions

ID	**UC1**
Name	Completing Puzzles
Description	Candidates should be able to complete puzzles on the website, and must be able to upload their code that generated the solution
Actor	Candidate
Pre-conditions	The candidate is logged in to the website and the puzzle is unlocked and viewable
Trigger	User enters an answer on the page, either in text form or uploading a file.
Basic Flow	1) User navigates to relevant puzzle page. 2) User downloads any required material to solve the puzzle. 3) User solves the puzzle. 4) User submits their answer and code for review 5) If correct, they are awarded points, otherwise they could be given hints to progress
Post-conditions	The user has the relevant points and/or achievements/awards for solving the puzzle
Alternative Flow	User solves an introductory puzzle while not being

ID	**UC1**
	logged in, and upon solving is prompted to create an account to save their progress
Prototype	
ID	**UC2**
Name	Register New Candidate
Description	A new candidate should be able to create a new account to track their progress.
Actor	Candidate
Pre-conditions	User has navigated to the website's registration screen
Trigger	User has clicked on the "register user" button.
Basic Flow	1) User navigates to registration screen. 2) User is prompted for their new login details, including username, password and email. 3) User fills in their details. 4) User details are transmitted to accounts system 5) A new account entry is created for this user with the appropriate details. 6) The user is sent to the login screen to enter their details.
Post-conditions	The user now has an account with login details they know, and they can log into this account to track their progress
Alternative Flow	1) User navigates to the registration screen. 2) User is prompted for their new login details, including username, password and email. 3) User fills in their details, but the username or email have already been used for a previous account. 4) User is informed of the problem, and if their email is the one that has already been used, they are prompted to reset their username/password
Prototype	

ID	**UC3**
Name	Register New Employer
Description	An employer should be able to create a new account to track the progress of candidates.
Actor	Employer
Pre-conditions	User has navigated to the website's registration screen.
Trigger	User has clicked "I'm an employer" button.

ID	**UC3**
Basic Flow	1) User navigates to registration screen. 2) User clicks the “I’m an employer button”. 3) User is prompted for their new login details, including username, password and email, alongside a method to verify they are an employer. 4) User fills in their details. 5) User details are transmitted to accounts system 6) The details are verified to ensure that this account should be an employer account. 7) A new account entry is created for this user with the appropriate details. 8) The user is sent to the login screen to enter their details.
Post-conditions	The user now has an employer account with login details they know, and they can log into this account to track their progress
Alternative Flow	1) User navigates to the registration screen. 2) User is prompted for their new login details, including username, password and email. 3) User fills in their details, but their details fail the employer verification process. 4) User is informed of the problem, and can try to enter the correct details instead.
Prototype	

ID	**UC4**
Name	Log In
Description	Log a user in to their account.
Actor	Candidate, Employer
Pre-conditions	User must have previously registered an account, administrators must not be able to access credentials of any student.
Trigger	User opens application and has not clicked ‘Remember Me’ on prior use.
Basic Flow	1) User enters credentials into relevant fields, as well as a checkbox that allows them to skip log-in process in the future. 2) If credentials are verified by the database, provide access to home page. • For students, home page will display relevant details

ID	**UC4**
	such as their score, position on leaderboard, previously submitted code and future problems. • For administrators, home page will display options to edit and add problems as well as view student solutions. 3) If credentials are incorrect, allow the user to enter incorrect details a total of 5 times before locking account for 5 minutes.
Post-conditions	Students and administrators will have access to their relevant accounts.
Alternative Flows	• Allow the user to reset their password through a button prompt and a relevant email entered. • If the application is opened and user has previously selected ‘Remember Me’, skip log in process and open account straight away.
Prototype	

ID	**UC5**
Name	View Code for Puzzle Answers
Description	Be able to access/download the code provided alongside a puzzle answer.
Actor	Employer, Candidate
Pre-conditions	User must have an account or be an administrator, and the specified puzzle must have been completed by the specified user.
Trigger	User clicks a “view submission” button on the page for a puzzle
Basic Flow	1) User navigates to the page for a puzzle. 2) User clicks the “view submission” button 3) Relevant submissions will be displayed: a) If the user is a Candidate, their previous submitted code will be displayed b) If the user is an Employer, they will be provided with a list of Candidates that have solved the puzzle, which is by default sorted by time. User can then click on a Candidate to see their submitted code
Post-conditions	Relevant code will be displayed by the page or provided for download.
Alternative Flow	1) User navigates to the account page of a Candidate. 2) If the user is either an administrator or logged in to the relevant account, the page will display a list of

ID	**UC5**
	puzzles that the Candidate has solved 3) User can click on a problem to see the Candidate's code submission for the problem

ID	**UC6**
Name	View Rankings of Candidates
Description	View the rankings of all the candidates in the system. Also provides the ability to view just the top 10% of candidates.
Actor	Employer
Pre-conditions	User must have an Employer account.
Trigger	User clicks a "view ranking" button on the home page.
Basic Flow	1) User navigates to the home page. 2) User clicks the "view ranking" button 3) The ranking will be displayed as a list of Candidates, each identified by their name and number of puzzles solved. The list is sorted in decreasing order of the number of puzzles solved. 4) User can click the "top 10%" button to filter the list, only displaying the top 10% Students with most puzzles solved. 5) User can click on a Candidate's name to view their profile
Post-conditions	The ranking will be displayed by the page.
Alternative Flow	1) User navigates to the account of a Candidate 2) User clicks on the "view ranking" button. 3) The ranking will be displayed as a list of Candidates, each identified by their name and number of puzzles solved. The list is sorted in decreasing order of the number of puzzles solved. 4) The entry associated with the account will be highlighted and the page will display the list with the highlighted entry in the center.

ID	**UC7**
Name	View Trophies
Description	View all the trophies a user has gained over the course of their time using the system.
Actor	Candidate, Employer

ID	**UC7**
Pre-conditions	User must be logged in.
Trigger	User clicks on the “Trophies” tab of the puzzle screen/main screen
Basic Flow	1) User clicks on the Trophies tab. 2) User can see a graphical depiction of the trophies they have, alongside a name for each trophy. 3) User can hover over trophies to learn what that trophy was earned for. 4) User can view the percentage of trophies they have gained, and the requirements for each one. 5) User can close the Trophies tab.
Post-conditions	User knows what trophies they have unlocked and feels motivated to unlock more.
Alternative Flow	1) User is an Employer. 2) User has navigated to a Candidate’s page. 3) User can view all the trophies a User has earned.
Prototype	

3.1 External Interfaces

3.1.1 User interfaces

Requirement Name: Login Screen

Requirement Number: 1

Requirement Type: User Interface - Functional

Use Cases: Used when candidate wants to log in and start/continue solving the puzzles.

Description: A login screen, where a user can enter their details and “log in”, allowing them to continue the puzzle they are currently working on with their progress saved.

Rationale: If we are implementing an accounts system then there should be a way to log in to your account.

Fit Criterion: A test user can enter their details and log in, and then be taken to the main part of the system.

Priority: Should have

Conflicts: None

Dependencies: Accounts System

Requirement Name: Puzzles Screen

Requirement Number: 2

Requirement Type: User Interface - Functional

Use Cases: User is solving puzzles/ submitting solutions

Description: There will be a portion of the screen displaying the current puzzle, alongside another part of the screen where the answer to the puzzle should be entered. There will also be an area where the user can submit their code for that puzzle.

Rationale: The user needs tio be able to view the puzzle, and be able to submit their answer and their code

Fit Criterion: A test user can view a puzzle and submit a solution alongside their code.

Priority: Must have

Conflicts: None

Dependencies: Partial dependency on account system

Requirement Name: Progress Bar

Requirement Number: 3

Requirement Type: User Interface - Functional - Gamified Element

Use Cases: The user wants to see how far they are through the puzzles

Description: A progress bar that shows the user how many puzzles they have solved and how many more they have to solve

Rationale: A progress bar would fall under one of our gamified elements - seeing it will motivate our users and make them feel like they're making progress. Seeing the bar go up will be a kind of reward.

Fit Criterion: There is a progress bar that increases when a test user goes through the puzzles

Priority: Could have

Conflicts: None

Dependencies: Account System, Puzzle System, Puzzles Screen

Requirement Name: Trophy Area

Requirement Number: 4

Requirement Type: User Interface - Functional

Use Cases: A user wants to see which trophies they have earned.

Description: A sidebar that the user can enter to see which trophies they have earned.

Rationale: When awarding the user trophies and other similar gamified elements, a huge part of why that is engaging is the user's ability to view their trophies. If this is not implemented, the users will not have a way of reflecting on their achievements.

Fit Criterion: A test user can click on the sidebar and view their trophies.

Priority: Could have

Conflicts: None

Dependencies: Trophy System, Puzzle System, Puzzles Screen

Requirement Name: Metrics Interface

Requirement Number: 5

Requirement Type: User Interface - Functional

Use Cases: The employer wants to view a user's metrics, or the overall rankings of users

Description: An area solely for the employer where they can view user metrics and the overall rankings of viewers. It should be easy to view the top 10% of users in particular. These metrics will include the time taken for each question and the ability to view a user's code for each puzzle they have solved.

Rationale: We want the client to be able to view the data on the users, otherwise this whole system does not fulfill its purpose of enabling the client to find technical talent.

Fit Criterion: A test user can view the metrics of other candidates. This data should be verified to be correct by another test user logged into the candidate accounts.

Priority: Must have

Conflicts:None

Dependencies: Accounts system, Puzzles system

Requirement Name: Intuitive UI

Requirement Number: 6

Requirement Type: Non-functional - User Interface

Use Cases: Whenever either type of user uses the software

Description: The user interface should be easy to use and easy to understand.

Rationale: We want anyone with talent to be able to complete the puzzles, and any employee to be able to review users.

Fit Criterion: The UI should not be cluttered, and should have its look verified by the client as acceptable.

Priority: Could have

Conflicts: Deadline

Dependencies: All UI requirements

3.1.2 Hardware interfaces

Requirement Name: Website Implementation

Requirement Number: 7

Requirement Type: Hardware Interface - Functional

Use Cases: The user wants to access our website

Description: The system should be implemented so a web browser can interact with it

Rationale: We want this system to be as easy to access as possible, and so a website is the natural choice.

Fit Criterion: A test user can view the website using their web browser.

Priority: Must have

Conflicts: None

Dependencies: None

3.1.3 Software interfaces

Requirement Name: Database Interface

Requirement Number: 8

Requirement Type: Software interface - Functional

Use Cases: Within the account system

Description: The system must interface with a database to store user data

Rationale: In order to have an ordered way of storing this information, it would be best to have a database with all this information on it. In order to do this, we must have some way of interfacing with the database management system.

Fit Criterion: Database features work

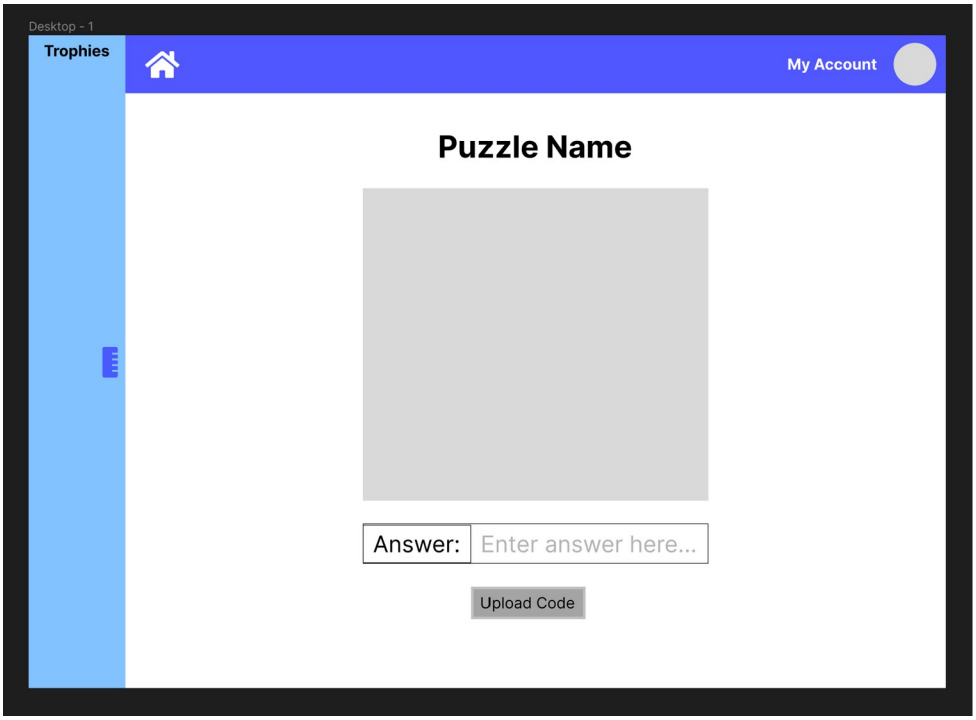
Priority: Should have

Conflicts: Cost

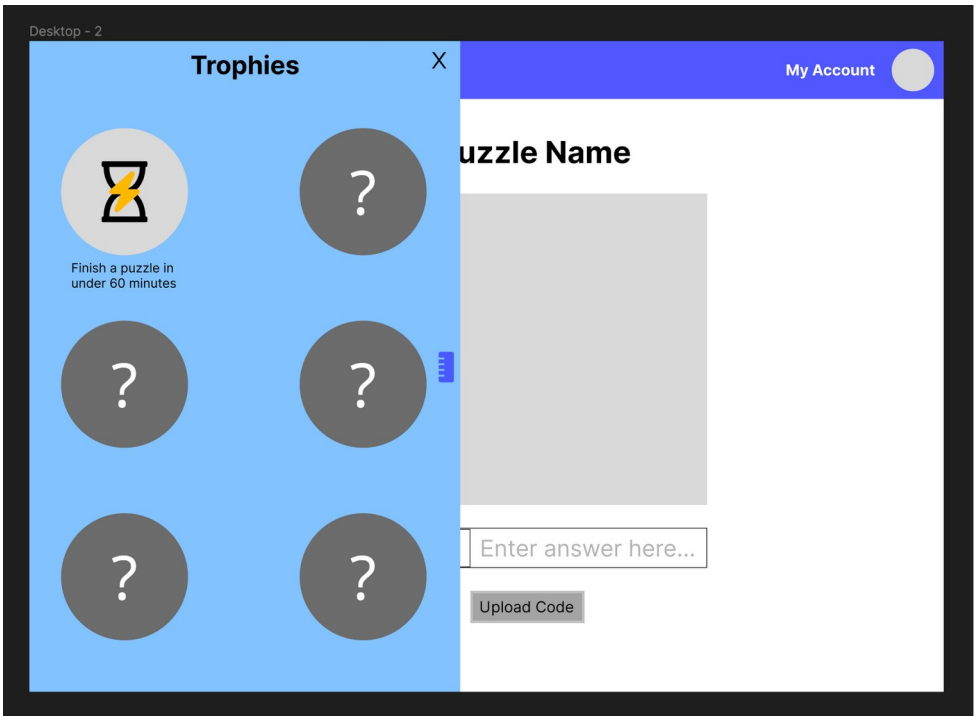
Dependencies: None

3.1.4 Prototypes

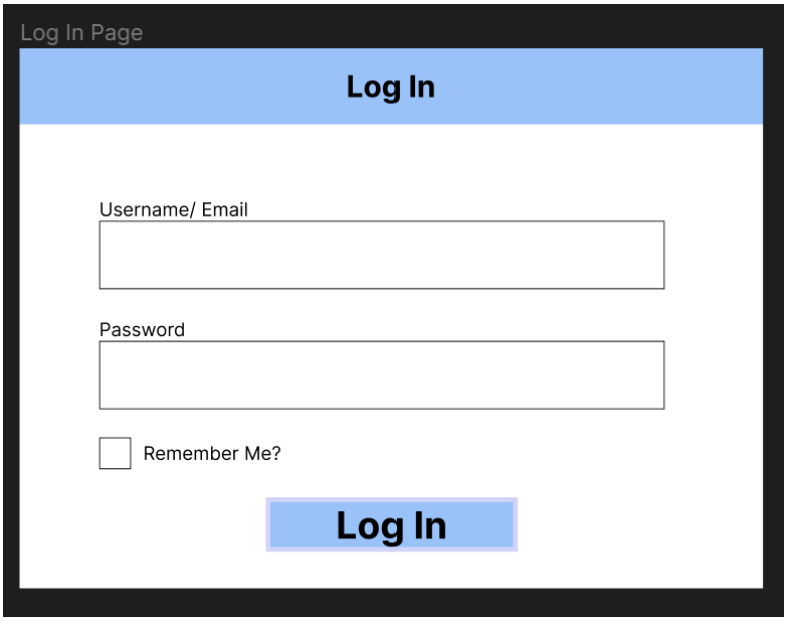
Main Puzzle Screen:



Trophies Side-screen:

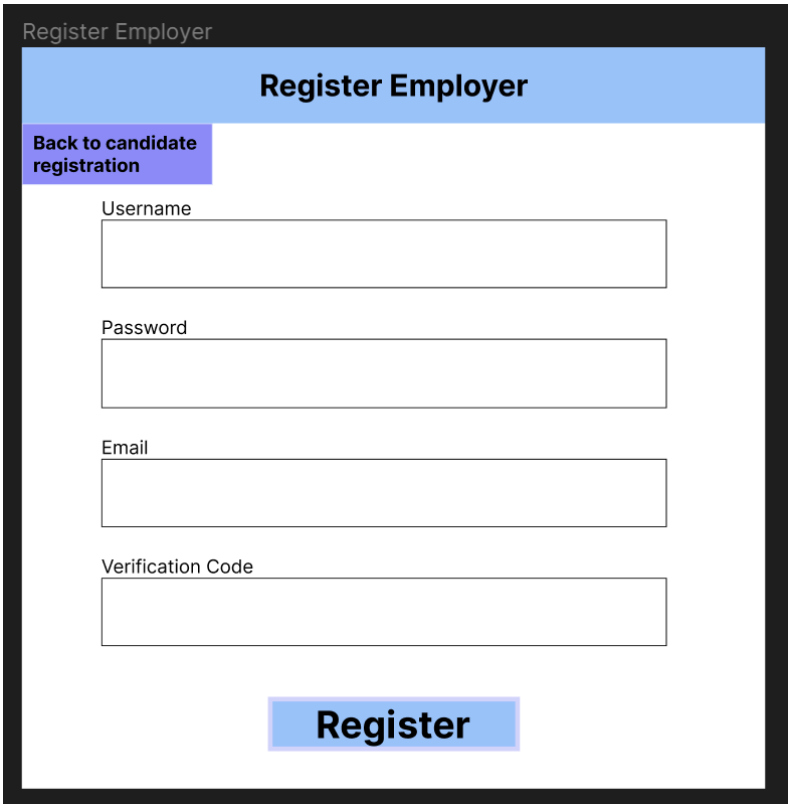


Log In:



Register Candidate

Register Employer



3.2 Functional

Requirement Name: Puzzle System

Requirement Number: 9

Requirement Type: Functional

Use Cases: Answering or viewing puzzles

Description: A system that allows users to view puzzles and submit answers for them. If the answer to the puzzle is wrong, the system should not allow the user to proceed. If the answer is correct, the user should be provided the next puzzle.

Rationale: We want the users to be able to view and answer the puzzles, so we need to build that functionality.

Fit Criterion: A test user can view and answer the puzzles.

Priority: Must have

Conflicts: None

Dependencies: Website implementation. Partial dependency on Accounts System.

Requirement Name: Accounts System

Requirement Number: 10

Requirement Type: Functional

Use Cases: The user wants to track their progress between visits to the site. The employer wants to view a user’s data.

Description: A system that interacts with a database to store user data. This should include username, password and email, alongside progress through the puzzles and any unlocked trophies.

Rationale: We want to keep track of the user data to allow the client to assess their ability. This necessitates a system that stores and manages all this data.

Fit Criterion: Test users can retain their progress between sessions if they have completed any puzzles, and other test users can view this progress.

Priority: Should have

Conflicts: Cost

Dependencies: Database interface, Puzzle system

Requirement Name: Trophy System

Requirement Number: 11

Requirement Type: Functional - Gamified Element

Use Cases: A user achieves something noteworthy and should be rewarded

Description: A system that awards “trophies” to users to reward them for doing something noteworthy. It should keep track of which trophies each user has, and detect when it should award them new trophies.

Rationale: This is the main gamified element of the overall system, and we want these gamified elements to motivate players to keep trying at the puzzles. Giving them bonus rewards further incentivises them to keep trying their hardest, and also provides more metrics for the employer to differentiate candidates by.

Fit Criterion: A test user should unlock trophies for completing certain tasks.

Priority: Could have

Conflicts: Deadline

Dependencies: Accounts System

Requirement Name: Number of puzzles

Requirement Number: 12

Requirement Type: Functional

Use Cases: When a user proceeds through the puzzle or an employer views the puzzle answers.

Description: There should be 10-15 puzzles

Rationale: This is a large enough number of puzzles to be sufficient to measure the capability of candidates, but not so large as to make development infeasible.

Fit Criterion: There are 10-15 puzzles.

Priority: Must have

Conflicts: Deadline

Dependencies: Puzzle System, Puzzles Screen

3.3 Non-Functional

Requirement Name: Security

Requirement Number: 13

Requirement Type: Non-functional

Use Cases: Registering, logging in, getting rankings

Description: System should be resistant to unauthorised access.

Rationale: We don’t want the wrong users to get access to other users’ information

Fit Criterion: There should be measures in place to stop ordinary users from getting access to the employer accounts or a superuser account.

Priority: Should have

Conflicts: Deadline, Cost

Dependencies: Account System

Requirement Name: Password Encryption

Requirement Number: 14

Requirement Type: Non-functional

Use Cases: Whenever a user has a password

Description: A system that ensures passwords are not stored in plaintext and are transmitted thoroughly

Rationale: We want our users’ passwords to be safe and secure.

Fit Criterion: Passwords should not be stored or transmitted in plaintext

Priority: Should have

Conflicts: Deadline

Dependencies: Account System

Requirement Name: GDPR Compliance

Requirement Number: 15

Requirement Type: Non-Functional - Legal

Use Cases: Whenever user data is stored

Description: Our accounts and database systems should be GDPR-compliant.

Rationale: This is a legal requirement.

Fit Criterion: All requirements given by GDPR are met.

Priority: Must have

Conflicts: Deadline

Dependencies: Accounts System, Password Encryption

Requirement Name: Increasing Difficulty

Requirement Number: 16

Requirement Type: Non-functional

Use Cases: When a candidate proceeds through multiple puzzles

Description: The puzzles should increase in difficulty as the user progresses through them. Furthermore, there should be 3 “tiers” of difficulty: the first 5 puzzles should be completable by most computer science students, the next 5 should be graduate-level difficulty, and the final 5 should be challenging interview-level questions.

Rationale: We want to properly assess the capability of candidates while not scaring away people who may not initially be able to solve the more difficult puzzles.

Fit Criterion: We categorise the puzzles according to creativity needed, subject matter and complexity of problem, and then group our puzzles into the 3 tiers using these measures.

Priority: Should have

Conflicts: None

Dependencies: Puzzle System

Requirement Name: Cross-browser compatibility

Requirement Number: 17

Requirement Type: Non-functional - Installation - Portability

Use Cases: Accessing the system from any major web browser

Description: Users should be able to access the system from any major web browser. This means the system should only use features compatible with the HTML5 standard (i.e not any features

that are only supported by Chromium-based browsers).

Rationale: We want all possible users to be able to use the system, but also want full functionality for the system. This means while we can't support some of the more fringe browsers like GNU IceCat, we would still like to support Firefox, Safari, and all of the Chromium-based browsers. This means we should stick to the standards common to all 3, i.e. the HTML5 standard

Fit Criterion: We test the system on Firefox, Safari and Chrome.

Priority: Could do

Conflicts:

Dependencies: None

Requirement Name: Good Documentation

Requirement Number: 18

Requirement Type: Non-functional - Maintainability

Use Cases: Other developers want to understand the code

Description: We should document our code well so that any other developer has a good idea of what this code code. This should mean that any other team member could understand the code, and any future developers would be able to as well.

Rationale: It makes the software more maintainable for future developers, and ensures developers can work on each other's code within the team.

Fit Criterion: The code has thorough documentation and is comprehensible to other developers on the team.

Priority: Could do

Conflicts: Deadline

Dependencies: None

Requirement Name: Low Cost

Requirement Number: 19

Requirement Type: Non-functional

Use Cases: None

Description: The system should not cost a lot of money to run.

Rationale: As a team we do not have any money to put forward for development.

Fit Criterion: The team does not have to pay money towards the project

Priority: Must have

Conflicts: Database Interface, Account System

Dependencies: None

Requirement Name: Deadline

Requirement Number: 20

Requirement Type: Non-functional

Use Cases: None

Description: The project must be completed by the week commencing 29th April

Rationale: This is the deadline set out in the group project briefing

Fit Criterion: Our final meeting and product delivery happens that week

Priority: Must have

Conflicts: Number of features

Dependencies: None _____

4. Verification

We will verify the requirements by verifying their fit criteria are met.