Team 18

Software Quality Strategy Document

Nexus

Saeed Othman fully contributed to the hack and everything to do with the hack. The team advised him on what to do and how to go about certain aspects, but the functionality, code and design is to his credit.

Adam Mohammed fully contributed to making the website and created most website aspects. Adam also contributed to the Software Quality Strategy document as well as the Software Design Document by answering questions.

Naomi Adesiyan contributed to the website. Naomi overlooked the information side of the website by writing about the game, how it works, how to play etc. Naomi also contributed to the Software Quality Strategy document as well as the Software Design Document by answering questions.

Ayshah Malik fully contributed to making the Software Design Document. Ayshah also contributed to the Software Quality Strategy document by answering questions and creating the final draft and format for the document.

Norman Nguyen contributed to making the Software Design Document. Norman also contributed to the Software Quality Strategy document by answering questions.

Adam Ejaz fully contributed to making the Software Quality Strategy document and answering most the questions.

Jashan Gandham fully contributed to making the Software Quality Strategy document and answering most the questions.

Mohammed Mohammed did not contribute anything for submission 2.

Werner Gutierrez Biela did not contribute anything for submission 2.

Gurnaek Singh did not contribute anything for submission 2.

Scope

This document specifies the software quality strategy for the gameplay of a game with the interim title “Nexus”. It is based on elements discussed in various meetings held since 14th October 2021 and involving Adam Ejaz, Jashan Gandham, Ayshah Malik, Norman Nguyen, Naomi Adesiyan, Adam Mohammed and Saeed Othman.

Nexus aims to create a fact paced, interactive environment where players take turns to place their tokens on the board in hopes of creating the most links (of five). Nexus intends to accommodate for players of various ages and cognitive abilities. By participating, players can compete against opponents in various ways and win using unique strategies. Implementing random cards and dead tokens mean no two games will be the same.

The software quality comes by creating a sustainable, entertaining game which runs efficiently, this will be extended using power cards, doom cards, a shopping system, and an online leader board. Power cards will help players by providing advantages over their opponent, whereas doom cards will hinder a player's chances of winning, for example the player must miss their go. A score-based system allows players to track their highest scores within the game, which is then uploaded onto the leader board. The leader board will fuel rivalry between contestants and create a competitive environment which will attract players and keep them coming back to play.

A high-level goal we have for Nexus, which determines software quality within the game, is having a fully functional, turn-based game where only players who have created an account can play. Another high-level goal we’ve established is successfully integrating the “random cards” and dead tokens to be introduced to the player at the right time to perform the tasks they’re intended to do. Instead of just appearing on the screen and offering nothing at all we want them to create obstacles for the players to overcome.

The remainder of this document is structured in a way to provide an overview into the external quality of the software and then will delve into the internal factors of building the product in a manner which will also verify software quality has been reached. The document will conclude by explaining the security aspects of the game in terms of securing the software and providing a secure system for the users playing the game etc.

External quality: Building the right product

Nexus has different game modes, random cards, and dead tokens to create an atmosphere of excitement. One of the game modes will be timed to add pressure and tension as players will have to beat their opponent in a limited amount of time whilst calculating the effects tile sinking will have on their tokens. Tile sinking is when the tokens on the tiles above are shifted down on the board to replace the tokens from the link made. Other elements included are “random cards” and “dead tokens” which arbitrarily appear in the game to add to the fun and provide a factor of unpredictability. This causes users to change their approach when playing, allowing no two games to be the same.

Prototyping by using sketches and diagrams appears to be the best approach as it is clear and easy to understand. We believe it displays a clear visualisation of not only how the game will be played but its interface as well. The sketches and diagrams will be accompanied by labels to explain certain

parts of the game in further detail. Through this, various changes can easily be made leaving everyone with a clear understanding of the outcome.

There will be certain design principles in place to include numerous types of depictions, examples of these are displayed with the game board being drafted in an arrangement of a 10 by 10 grid system. Players are required to create an account so their points can be saved and seen when can login to the game. When creating their accounts, they will be given the opportunity to select and username and avatar for their character. Also, once selecting a game mode they'll need to select a colour that will represent their token to begin the game. The game is designed to ensure that there is an overall winner; this is determined by the player who can create the most links. Once the game has concluded, the scores from all users will be taken and uploaded onto a leader board where players with the top ten highest scores will be ranked. Each game mode will have its own leader board.

Nexus aims to be inclusive and accessible to as many people as possible. A way in which this will be established is through the avoidance of flashing lights for those will epilepsy. Furthermore, nexus will cater to the colour blind through the colour system in place, as each player can select their own individual colour. So, if a player is having difficulty when viewing certain colours, they will be able to access a range of other colours.

The game will be tested for not only a generic game test but with a complex aspect as well. In terms of the regular testing this will be achieved through checking that the overall game concepts are working as it should. For example, the tokens that players use will need to be able to successfully link with one another in any horizontal direction they choose. For the complex aspect of testing the game we will be ensuring that all game modes such as the deathmatch, are performing correctly and successfully by ending the game within the allotted time slot, regardless of whichever opponent is leading in relation to the score.

The game will be designed in a way where assets such as the graphics will offer a dark nautical theme. In accordance with this, the colour scheme will be in an achromatic format. Other assets like sound include background audio being deployed within the game, to extend the dark nautical theme: oceanic audio with sailor and pirate sound affects to make the game more enjoyable for players. This can be accessed within the main menu to be turned off. The website design has been carefully crafted to persuade users to want to play. With a logo to create a Nexus brand which players can associate with, a short script of what the game entails, and above all, the graphics which will entice a range of audiences.

One of the restrictions placed within the game is that the chatbot feature can only be used once the game has started and won’t allow the player who is currently taking their turn to chat until they’ve made their move. Another restriction is that once users select the colour of their token to play, they won't be able to change it mid game, they will have to exit to the main menu. Furthermore, players will have a 60 second timer to make their move else nexus will skip to the next player, this will add pressure whilst monitoring which users are still actively playing the game.

When considering the target audience of the game, the ESRB rating would be E 8+ (everyone 8 and over) which would allow almost all users to be able to play Nexus. This is because, although this game requires a certain level of thought and strategy to win against the opposition, it doesn’t offer gruesome violence or contain any offensive language within its gameplay. So, with it being best suited to support a rating of E 8+, it allows for majority of users to play. In having this as a rating for the game, it ensures that although not all users would be able to play a vast amount of people will be able to. The reason as to why the game contains a specific rating where players under 8 can’t compete, is simply down to the fact that younger users wouldn’t be able to create accounts that

require private information to play. This restriction has been designed to protect users under those ages as they'll probably have a lack of knowledge of the information they are uploading.
We will be able to keep the project scope under control and deliver the game by the end of the year by setting ourselves multiple mini deadlines to ensure that numerous sections of the game are completed to a standard which verifies software quality being met. For instance, there are key parts of the game that need to be reached a lot sooner than others, which include the setting up of user accounts and storing login information. This is one of the key areas within the game as this needs to operate as it's intended to, so that only users who have successfully created an account are able to log onto the game.

The game has been designed in a way where if we believe changes need to be made then they can be implemented to further improve the overall game prior to the final deadline. One way we've accomplished this is by creating classes within the source code which can be easily accessed in case they need editing. For example, if we feel the graphics of the power cards need tweaking, we will enforce those changes easily to provide the greatest experience for users when playing nexus without having to change a lot of code.

There are multiple meetings that take place every week, which is what allows us to be on top of the amount of work we all need to put in with regards to the game. These frequent meetings are crucial as it allows us to see the direction in which the game is heading and analyse if its best for the game to continue in this direction. It allows everyone in the game to contribute and communicate their ideas. These sessions are where we can conceptualise our ideas and discuss ways on implementing them correctly.

Through having multiple meetings, we can place a certain type of structure in terms of workload for the game, where we assign specific members to cover a particular area. By structuring the meetings subgroups can host their own meetings and produce a part of the game which will eventually come together to create Nexus. This enables us all to better manage our time and understand who we can ask for help as we know who is doing what.

We are following a software development process known as DSDM (Dynamic Systems Development Method), and through using this type of process we can plan future steps to meet our mini deadlines collectively and efficiently. Given that DSDM is a full project lifecycle we have catered it to our own specifications by using it to illustrate a diagram of a smaller section with comments below it. For instance, by using a small diagram labelled game board, the comments that would follow include many ideas in terms of requirements as to what would be best suited for the game board, such as the colour of the board and its dimensions etc. This process displays a unique view for us as we can see what area of the game needs to be discussed next, and what we all feel could be used or added within that section.

Internal quality: Building the product right

We are going to be using GitHub which will allow us to manage our depository. GitHub will allow multiple team members to code at once. This way we can also backup different documents and make different versions of the game code should we require it to revert to another version. Notes can be made so that each team member understands what has been done in their absence from other team members. This is great as it allows people to continue coding areas that they didn't begin with, allowing contribution from all members.

The code will be written to a high standard, so that it is intelligible and simple enough for people to interpret. The code has been created in a sophisticated way so that if any changes were to be introduced there would be limited side effects. We will have a structure when producing code which will be in the following order: Constants, read-only fields, Private fields, Properties, Constructors, Public methods, Internal/package-private methods, Protected methods, and Private methods. The code will be grouped into folders which will contain subfolders for all related entities. Specific titles will be given to classes in proper camel case form, making it more organised and straightforward to understand.

The file structure of the game has been organised to make it as clean as possible. For example, prefabs, scripts, and sprites are in their individual folder. When possible different classes were created to spread the code into different files instead of having one big file for it. This allows code to be readable instead of being cluttered and for debugging to be easier.

A form of documentation that we are maintaining are the comments, by explaining the code vividly in the comments, it will inform the developer of how a certain task is accomplished. Another form of documentation to make a developer's job simpler would be through using UML diagrams, this in turn will be making things less stressful to grasp. The Software Design document outlines the specifications in the game, the developer will be able to understand what is to be implemented in their own category. Each content provides what is expected within the game but is not limited.

The way forward with this was for the developers of the hack to work together weekly via discord as this allows screen sharing. Furthermore, to update each other by creating sub-categories like game design on discord which is updated with what is done so each developer knows what's going on and are on the same level of understanding.

When two people are working on the same part of software, we use the technique called pair programming. This is when two people are working together at one computer, and one person is writing the code whilst the other person is analysing each line written. Although this may seem counter intuitive as you'd save time having two people work on different things it means that only quality lines of code are written as they are being scrutinised by another individual. When we have two people working on different parts of the software we will then, and on GitHub when you fork the repository so they have their own copies meaning the main repository will be unaffected by changes they make. When they are finished with the code, they will make a pull request to the main repository and their code will be analysed to see if it conflicts with any other parts of the program, if their code is given the green light, it will be merged with the main repository.

GitHub is the method we have chosen to keep track of the different Nexus versions which we will be producing. This allows us to safely backup our code because if anything goes wrong which cannot be fixed in a new version, we can revert to an older version. GitHub allows for great documentation of projects, making it is easy to understand for new users too. GitHub is a large coding community; it is a good platform for presenting work to the public. This allows Nexus to get the exposure it needs to draw in its players and entice people to play. Lastly, GitHub allows you to track changes made to code when building new versions of a game so that broken versions can be fixed quickly and efficiently.

The branching strategy we have opted to use is GitFlow. This strategy consists of 5 branches. main/master, hotfixes, release branches, develop, feature branches. The benefits of using GitFlow is that it makes parallel development very simple by isolating new development from finished work. Features and non-emergency bug fixes are done in feature branches, and then are merged back into the main repository when it is agreed that the code is good for release. There are also hotfix branches in GitFlow which are used to make emergency changes.

Using Trello, we will have different boards for different parts of the projects. Different people will be assigned to these boards, so they know exactly what needs to be done without any questions. If they are developing software, once they have made a pull request, they can attach it to the Trello board for it to be checked by someone else. If the pull request is successful, they will be assigned more work if there is more to do.

We will ensure the software is reliable by doing thorough checks to see that the messages are end to end, to prevent unauthorised users that are outside of the game from viewing messages that players send between each other. We will also make sure that players aren't allowed to make any illegal moves, otherwise an error message will occur if the game picks up an attempted illegal move. Therefore, the player will need to proceed with the game in the way that the game is designed to, so that they can make their move once again. The integrity of the software will be kept by coding as best as possible to reduce bugs. Unit testing will take place meaning parts of the software will be tested as they are produced. This reduces chances for errors whilst also ensuring that Nexus runs smoothly as production comes to an end. This is a better approach in comparison to testing once everything is finished as it is harder to discover where each error is. Errors could also be linked making things extremely difficult to fix, hence the reason as to why unit testing is required for Nexus.

Unit testing is one strategy which we intend to use, this is because when problems occur, it's easier to resolve them as there are no previous errors which can be linked to new errors. This strategy involves testing smaller components of a program to eliminate bigger errors in the future. For example, it could involve testing classes individually or testing smaller parts of the game to begin with such as animations or the user interface operation etc. Tests can be written before a specific part of the game is produced once it has been produced it can then be tested.

Performance testing can also be carried out, this can be done on Nexus by testing if the max number of players can be handled adequately without any network dropouts. This is an example of load testing. Load testing is designed to see if a software or application can carry out what it was designed to without any problems. Stress testing will also take place to identify potential flaws in the game, the goal of this is to break the game in any way possible. This could involve exploring ways to cheat the game and trick the system also known as glitches. Compatibility testing will also take place to ensure that no matter what device the end user is using, they will still be able to play the game with players using different devices to them. This will be tested by trying to play the game amongst each other whilst all using different devices.

The main provision when there is a network dropout, will consist of pausing the game for every player meaning no illegal moves can be made that eliminates the ability for players to cheat. The game will stay paused until all players have a stable connection and then it will allow each player to resume and continue playing the game. Should a player's connection not recover in a specified amount of waiting time then the game will end for them and allow other players to continue. If the game does contain a network issue that isn't recoverable, the game will simply declare no winner to that match and instead of counting that as a completed game, it won't even register as a match ever taking place. This is to ensure that users can keep the record intact without having to worry over an unforeseeable loss on their account.

Although it is not set in stone, we are considering using GitHub Actions as our repository is on GitHub. With GitHub Actions whenever somebody attempts to push to main, we can have an automated linter run to check the quality of the code, if the code doesn't meet the correct standards, it will be rejected and not pushed. This is beneficial as it saves us time from having to manually check every single file made to see if the code is messy, or if the developer doesn't conform to the programming styles for that language making it hard to read for others.

A way to secure good performances within the software would be to assess over the code, and decide which section is taking up the most usage whilst the game is being played. This can be a situation which can cause a large issue as it will prevent players from being able to play the game uninterrupted and will even have them losing focus or even potentially selecting the wrong move in a match. To determine if the software is sluggish and not running to its maximum capacity, we will be using a profiler known as JProfiler. In using this we will be immediately notified if a disruption were to take place, where most of the usage is coming from etc. With this information being granted to us we will be able to personally improve the response times of gaming interactions, movements which will in turn provide the player with a high-performance game that they can enjoy.

One of the benefits of using the software Unity, is the fact that it contains a characteristic which will automatically prepare nexus to be release-ready for its due date. This software offers this unique feature by scanning the entire code whilst also checking to see if there are any outstanding problems, loss of data etc. Once those checks have been completed Unity is then able to export the game into a usable file format that can later be externally imported allowing users to be able to take part.

Nexus will be presented through the game-based website, where it will contain the improved and final version of the product. Once the website is loaded up it will show information not only how to play the game but how to play it correctly, and so from there users will be able to compete against one another in matches. The use of an installer isn't necessary for Nexus thanks to the development software Unity, which is what allows the game to be created in a way that it only requires a download, rendering any types of installations useless and obsolete.

Given that we have not only created most of the assets from scratch, and for the remainder of the others having been used by the development software Unity, we don't need to be worried about any problems regarding the licensing of our assets. This is down to us legally having the rights to our own created assets, and with unity containing the licensing to their transferable assets as well. In terms of us meeting multiple regulations concerning data privacy we will be following the guidelines of GDPR (General Data Protection Regulation), to prevent user data from being accessed by third parties. The GDPR guidelines in this scenario consist of only allowing the user to be able to have the rights to share their personal data, unless they provide us with their authorisation allowing us to upload their information online. Furthermore, to provide reassurance for the players that may be reluctant at first glance to not only create an account but enter in their private information as well. To protect users, we have specific measures in place, to ensure that other gamers will never be able to gain access to another player's name, address etc. Instead, the opponent will only be presented with a username that the user has chosen to call themselves.

Security

A way in which the software will be secure would be through the players creating accounts that can be used to verify the user's information in terms of name, ranking etc. Due to the use of player accounts, players outside of the game won't be able to take part or change any settings, therefore they won't be able to provide any input regarding the game. To provide security within the software we have made sure that the users personal information like name, age address is encrypted. This is something that's very crucial when creating a game, so that players never need to feel that their private data could either be hacked or leaked, but in fact protected in a way where only they can view that information so that it can always remain classified.

To begin with we will identify the vulnerable data assets which are exchanged within the game, for example we have identified the following threats: intentional threats and accidental threats. This means that we need to ensure the game is not susceptible to threats such as ransomware, malware and especially Denial of Service which is the most common in online multiplayer games. Accidental threats are difficult to protect against but there should be a protocol in place should an accident take place making the game a harmful environment. Accidental threats can include not correctly following protocols, breaking code during development or updates, computer malfunction etc. The next step in how to carry out this risk assessment and threat analysis is to determine the likelihood of an incident taking place. This can be done by using categories such as low, medium, medium, and high. We should then assess the impact that a threat could have and the actions that must be taken place to minimise the amount of harm it has on users or the game altogether. Lastly before documenting the results of all these points, we must determine a resolution to potential threats which have a high likelihood of taking place. This allows us to focus on the most critical threats first and secure the least dangerous threats as we go along.

An example of where secure-by-design principles are applied is within the software by privatising the variables. In doing this it restricts access to everyone except for the developer, so that no one outside of the developer can add their own changes to the game. In an instance such as, a user can't suddenly decide to change the game board, or the colour of the token that their opponents have selected just because they don't share the same type of appreciation for them.

Within nexus there are multiple security guidelines which include the validation of data input, this is where certain sections in the game contain validity checks which is when only specific responses will be allowed to be entered in. An example of this is the chat box, which is like a type of forum where players can communicate with one another when either playing in a game or once the game has ended. So, within the chat box the security guidelines entail, that users will not be allowed to type in explicit text or even deliver abuse or else they will be immediately kicked off the game. If a specific user carries on trying their luck by spouting malicious comments within the chat, their username will be flagged and permanently suspended leaving them unable to log onto the game.

The user's game name can be protected by encrypting their name and displaying something different to other players in the game. They will also have the choice to display their name if they wish to. The game assets will be protected by having various private classes and integrated blocks of hidden code so that no user is able to edit or manufacture their own changes. This way the game cannot be broken or manipulated by anybody; the game will also require a user to be the main host which will be through a secure server. This is so that no entity outside of the game can tamper with the data being sent across the devices of the players or cause a disruption of any online matches that are currently set in motion.

To test the effectiveness of the security measures in place, various checks will be considered for instance the user data. We will play the game and test if we are able to see other players' names and game data without their permission. We will essentially try to do everything we are not supposed to do to see if we can break the rules and see things we aren't supposed to see. We will then put malicious comments into the chat box and see if they get flagged up effectively. The game environment is going to be safe for children to play so we need to ensure that appropriate testing strategies are carried out. When a user uses harmful messages in the chat box they will be warned before being removed from the game. If all these measures are carried out effectively then we can ensure that the game does not have any security flaws deeming it safe to play.