**Product Implementation (Sprint 3)**

*Sudoku Web Application*

|  |  |
| --- | --- |
| **Team** | Lamb Chops |

As part of your project your task for Spring 3 is to update the current version of your project with **additional** or **extended features**. Besides updating **Requirement Template** and **Design Template** you are **required** to answer the following questions regarding the new FR/NFR or existing FR/NFRs that you added or updated in your project.

* **NOTE: Each of the student is required to update the following FR table given below with 10 questions regarding the <new functional feature(s)> that you will be adding to your existing application.**
* **NOTE: Please work individually and not share your answers with your team members unless they are group-based.**

**Definition**: **Functional Requirements (FRs)** – Functional requirements define the basic system behaviour. Essentially, they are what the system does or must not do, and can be thought of in terms of how the system responds to inputs. Functional requirements usually define if/then behaviors and include calculations, data input, and business processes.

# Student: Anna Fortenberry

|  |  |  |
| --- | --- | --- |
|  | **Functional Requirements (FR) Questions** | **Functional Requirements (FR) Answers** |
| **Individual** | **Q1:** To add the **BeginnerBoardGenerator()** how many files you needed to update? | **A1:**  We originally had two files separating logic ang utility functions. For this project, we are learning the JavaScript language while creating our application, so we ended up combining the two files to bypass the import/export syntax. Before we made this update, we could not get them to work correctly. |
| **Individual** | **Q2:** If several files were changed did the **maintainability** aspect of the code **increased** or **decreased**? | **A2:**  Although our new combined file is longer, it is easier to maintain (considering our current knowledge in JavaScript). We do not have to worry about functions or parameters being out of scope. |
| **Group** | **Q3:** What were the **keywords** that you [**used/thought of/searched for/discussed/]** with your team members when you decided to add the new feature(s). **Please list them all.** | **A3:**  **CheckValue():** Row, Game Logic  **DisplayBoard():** Display, Initial Board  **CheckAnswer():** Final Game Board,Removed Values, User Input |
| **Individual** | **Q4:** What were the **keywords** that you [**used/thought of/searched for]** when you worked on a feature on your own. **Please list them all.** | **A4:**  **EmptySpotCoords():** traverse, efficiency, track  **CheckRow():** iterate, compare, Boolean  **CheckValue():** Boolean, nested, validify |
| **Individual** | **Q5:** Please rank the **keywords** from **Q4** by providing a ranking score between [1-5] 5 being the **most informative** **keyword**, and **1** being the **least informative** **keywords** | **A5:**  **EmptySpotCoords():** traverse **(5)**, efficiency **(3)**, track **(3) CheckRow():** iterate **(5)**, compare **(4)**, Boolean **(4)**  **CheckValue():** Boolean **(4)**, nested **(3)**, validify **(5)** |
| **Individual** | **Q6:** Explain the reason why certain **keywords** were more **useful** to help yourself better implement the new **<functional feature>** that you added in the code vs. the one that was not useful or that you needed to **change** or **discard**. | **A6:**  For EmptySpotCoords(), traversing the entire board and tracking all the locations of empty cell spots **corrected previous logic**. Although the original method seemed more efficient, it caused a seg fault. The new logic, despite taking more run time, is a better choice.  For CheckRow(), iterating and comparing the cells in the row **validates one rule of Sudoku** for a play. Thinking of the function as a Boolean **helps with the transition of coding in C++ to JavaScript**.  For CheckValue(), the CheckRow(), CheckColumn(), and CheckSquare() we **nested, to allow all three to be called at once** (necessary for every play). It is also a “Boolean,” used to validify a number filled on the board. |
| **Group** | **Q7**: What were the **phrases** that you **[used/thought of/searched for/discussed]** with your team members when you decided to add the new feature(s). **Please list them all.** | **A7:**  **UniqueBoardGenerator():** “How will it be unique?” “How to make each board different?”  **MoreThanOneSolution():** “What is there is more than one solution?” “Sudoku only has one solution.”  **BeginnerBoardGenerator():** “How many initial tiles for a board to be considered Easy?” “Range of numbers” |
| **Individual** | **Q8:** What were the **phrases** that you [**used/thought of/searched for]** when you worked on a feature on your own. **Please list them all.** | **A8:**  **EmptySpotCoords():** “How can we accurately locate and store every instance of an empty cell on our board?” “How do we ensure this task is completed for a wide range of empty cells?”  **CheckRow():** “How do we check whether there are any conflicts in a number’s coordinating row?” “What is the most efficient method to traverse the array?”  **CheckValue():** “What do we need in a function to validate a play in Sudoku?” “How do we make this function versatile?” |
| **Individual** | **Q9:** Please rank the **phrases** from **Q8** by providing a ranking score between [**1-5]** **5** being the **most informative** **phrases**, and **1** being the **least informative** **phrases.** | **A9:**  **EmptySpotCoords():** “How can we accurately locate and store every instance of an empty cell on our board?” **(5)** “How do we ensure this task is completed for a wide range of empty cells?” **(5)**  **CheckRow():** “How do we check whether there are any conflicts in a number’s coordinating row?” **(5)** “What is the most efficient method to traverse the array?” **(4)**  **CheckValue():** “What do we need in a function to validate a play in Sudoku?” **(5)** “How do we make this function versatile?” **(3)** |
| **Individual** | **Q10:** Explain the reason why certain **phrases** were more **useful** to help yourself better implement the new **<functional feature>** that you added in the code vs. the one that was not useful or that you needed to **change** or **discard**. | **A10:**  Certain phrases such as “accurately locate and store” or “most efficient method” **helped rethink logic and prioritize the most important aspects** of our code. |

**Please, in a few words describe the strategy you used to log/list your keywords [before/while/after] coding functional/Non-functional requirements (NFRs/FRs).**

My group and I meticulously discussed the functionality of our application while completing the Requirements document. At that point, we started coding. We revised our functionality and discussed it further while completing the Design document. Completing this form, we were able to log our keywords based on those conversations that occurred before, during, and after the completion of some functions.

# Non-Functional Requirements (NFR)

Add or update new features related to non-functional requirements that pertain to:

* Functional Suitability or (Accuracy) **(A)**
* Performance Efficiency **(P)**
* Compatibility **(C)**
* Usability **(U)**
* Reliability **(R)**
* Security **(S)**
* Maintainability **(M)**
* Portability **(Pb)**

***See the characteristics for each NFRs above:*** [***https://iso25000.com/index.php/en/iso-25000-standards/iso-25010***](https://iso25000.com/index.php/en/iso-25000-standards/iso-25010)

* **NOTE: Each of the student of each team is required to update the following NFR table given below with 10 questions regarding the <new non-functional feature(s)> that you will be adding to your existing application.**
* **NOTE: Please work individually and not share your answers with your team members unless they are group-based.**

**Reading Supporting Literature: We recommend reading the following papers to help you with the questions.**

**[1] On Non-Functional Requirements**

<http://www.ptidej.net/courses/log3410/summer11/Lectures/Article_6.pdf> (online google scholar source)

**[2] On Non-Functional Requirements in Software Engineering**

<http://ce.sharif.edu/courses/96-97/1/ce475-1/resources/root/NonFunctionalRequirements.pdf> (online google scholar source)

**[3] Rethinking the Notion of Non-Functional Requirements.**

<http://www.ptidej.net/courses/log3410/fall11/Lectures/Article_5.pdf> (online google scholar source)

**NOTE: <\*NFR Type\*> refers to the NFR type that you implemented. It could be the followings: 1) Accuracy (A), 2) Performance Efficiency (P), 3) Compatibility (C), 4) Usability (U), 5) Reliability (R), 6) Security (S), 7) Maintainability (M), 8) Portability (Pb) or 9) Other NFR (O)**

|  |  |  |
| --- | --- | --- |
|  | **Non-Functional Requirements (NFRs) Questions** | **Non-Functional Requirements (NFRs) Answers** |
| **Individual** | **Q1:** To add the **compatibilty** how many files you needed to update? | **A1:**  We combined our files with Sudoku logic and utility functions together to allow the functions to cooperate. Using the import/export statements in JavaScript was problematic. |
| **Individual** | **Q2:** If several files were changed did the **maintainability** aspect of the code **increased** or **decreased**? | **A2:**  The maintainability increased because all features are now accessible in the same scope. |
| **Group** | **Q3:** What were the **keywords** that you [**used/thought of/searched for/discussed/]** with your team members when you decided to add the new feature(s). **Please list them all.** | **A3:**  **SelectTile(), Usability:** clarity, visibility, user-friendly  **AutoDetectMistakes(), Customizability:** accommodate, audience, settings  **ValueRemover(), Reliability:** consistency, fail-safe, error-proof |
| **Individual** | **Q4:** What were the **keywords** that you [**used/thought of/searched for]** when you worked on a feature on your own. **Please list them all.** | **A4:**  **EmptySpotCoords(), Accuracy:** locate, compare, store  **CheckColumn(), Reliability:** compare, validate, fail-safe  **CheckValue(), Maintainability:** clear, efficient, versatile |
| **Individual** | **Q5:** Please rank the **keywords** from **Q4** by providing a ranking score between **[1-5]** **5** being the **most informative** **keyword**, and **1** being the **least informative** **keywords** | **A5:**  **EmptySpotCoords(), Accuracy:** locate **(3)**, compare **(3)**, store **(5)**  **CheckColumn(), Reliability:** compare **(4)**, validate **(5)**, fail-safe **(5)**  **CheckValue(), Maintainability:** clear **(5)**, efficient **(4)**, versatile **(4)** |
| **Individual** | **Q6:** Explain the reason why certain **keywords** were more **useful** to help yourself better implement the new **<non-functional feature>** that you added in the code vs. the one that was not useful or that you needed to **change** or **discard**. | **A6:**  Keywords such as “validate” and “clear” **effectively communicate the objective of the function**. This helps us **carefully create and reconsider our logic**, ensuring our **function produces the intended results**. |
| **Group** | **Q7**: What were the **phrases** that you **[used/thought of/searched for/discussed]** with your team members when you decided to add the new feature(s). **Please list them all.** | **A7:**  **FillBoard(), Performance Efficiency:** “What will ensure the fastest run-time?” “What will use the least memory?” “How can we optimize our logic?”  **RandomNumberGenerator(), Functional Suitability:** “How can we accurately generate unique boards?” “What is necessary to produce several quality game boards?”  **BoardReducer(), Maintainability:** “How can we maintain states between React components?” |
| **Individual** | **Q8:** What were the **phrases** that you [**used/thought of/searched for]** when you worked on a feature on your own. **Please list them all.** | **A8:**  **EmptySpotCoords(), Accuracy:** “find every single valid coordinate,” “check every cell,” “correctly store valid coordinates”  **CheckColumn(), Reliability:** “ensure fundamental function works 100% of the time,” “check every cell in coordinating column”  **CheckValue(), Maintainability:** “clear and concise logic,” “essential to multiple other functions,” “easy to utilize” |
| **Individual** | **Q9:** Please rank the **phrases** from **Q8** by providing a ranking score between **[1-5]** **5** being the **most informative** **phrases**, and **1** being the **least informative** **phrases** | **A9:**  **EmptySpotCoords(), Accuracy:** “find every single valid coordinate” **(5),** “check every cell” **(5)**, “correctly store valid coordinates” **(5)**  **CheckColumn(), Reliability:** “ensure fundamental function works 100% of the time” **(5)**, “check every cell in coordinating column” **(5)**  **CheckValue(), Maintainability:** “clear and concise logic” **(4)**, “essential to multiple other functions” **(5)**, “easy to utilize” **(3)** |
| **Individual** | **Q10:** Explain the reason why certain **phrases** were more **useful** to help yourself better implement the new **<non-functional feature>** that you added in the code vs. the one that was not useful or that you needed to **change** or **discard**. | **A10:**  Each phrase helped **identify how to best achieve/ correct our logic to achieve the coordinating nonfunctional requirement**. For example, with accuracy, we need to check “every single cell” for a potential blank cell, rather than stopping the search after we think we have found them all. This ensure our stored coordinates are **entirely accurate**. Considering maintainability, it is essential we write the CheckValue function with “clear and concise logic”because it will be **fundamental to other functional requirements we implement later**. |

**General Question:** Please, in a few words describe the **strategy** you used to log/list your keywords/phrases **[before/while/after]** coding functional requirements (FR) and nonfunctional requirements (NFRs).

For the keywords, it is the answer I mentioned before - My group and I meticulously discussed the functionality of our application while completing the Requirements document. At that point, we started coding. We revised our functionality and discussed it further while completing the Design document. Completing this form, we were able to log our keywords based on those conversations that occurred before, during, and after the completion of some functions.

For phrases, we used the same process. Sometimes the phrases more accurately clarified our goals and objectives regarding both FRs and NFRs. For example, we discussed the BoardReducer() function for meeting the non-functional requirement of maintainability. It “maintains states between React components.” If we tried to track this with keywords, we would have listed words such as “states” or “harmonize,” which don’t relay what the function maintains.