**1. Introduction**

**1.A.** ***Purpose***

This SRS describes the functional and nonfunctional requirements for the Mane Game. The Mane Game will be used by Tampa Mane Fix, LLC to gamify their website ([www.tampamanefix.com](http://www.tampamanefix.com)) and attract potential clients. This document is intended to be used by the developer to implement and verify proper functionality. All requirements specified here are committed for the Mane Game, unless otherwise noted.

**1.B. *Coding Conventions***

* Code will be written in Java.
* All classes and methods must be documented with JavaDoc comments
* JavaDoc comments must sum-up the purpose of class of method in the first sentence.
* All JavaDoc comments must list necessary parameters, return values, and any data relevant to the function(s) carried out by the code it documents.
* JUnit tests will reside in the same package.
* Every method that manipulates a data field must have tests to ensure proper functionality.
* Code will be done in a sequential approach.
* Tests may be written in conjunction with code, however it will be favorable to write tests prior to coding where foreseeable bugs.
* Changes in requirements must follow the Change Control Procedure outlined in section X.X.
* Any changes to requirements during development requires all previously written code be tested to the new requirement.

**1.C. *Scope***

The Mane Game uses directional arrows to allow players to navigate the board. The game will follow a similar design to that of the original Pac-Man, with the exception that the player object will be scissors that cut through emojis of hair. Players progress in levels and boards change. Upon reaching level three, players must sign-up for the Tampa Mane Fix newsletters to continue playing the game. A secure web form via GoDaddy allows clients to register for the newsletter.

**2. Overall description**

**2.A. *Product Vision “Gamifying Tampa Mane Fix Website to Attract Clients”***

The Mane Game is intended to add gamification to the Tampa Mane Fix website. It will be embedded on [www.tampamanefix.com](http://www.tampamanefix.com) and advertised through Facebook and Instagram to draw attention to Tampa Mane Fix, LLC. Context diagrams {[//attached//]} illustrate external entities and system interfaces for initial release. The system is expected to evolve over several releases and ultimately contain over 100 levels, allowing users to share their progress with friends and family. The Mane Game will require players to register their email address with [www.tampamanefix.com](http://www.tampamanefix.com). The game will follow a Pac-Man like structure with the player’s icon (a pair of scissors) constantly moving until a directional arrow is pressed. Points are earned by cutting through hairspray, foils, and blow dryers. Players must avoid tax collectors, landlords, and health inspectors, who each deduct points and slow progress (slow down the player). Player progress will be saved via Google Cloud using minimal storage. Once players login to [www.tampamanefix.com](http://www.tampamanefix.com), they must be able to resume the game at the level they left at. Initial prototype will use a local database to save player information. A form will collect and store the data for this prototype.

**2.B. *User Classes and Characteristics***

|  |  |  |
| --- | --- | --- |
| **Class Name** | **Number** | **Description** |
| Players | Approximately 25 in the first month, 100 the next month, and up to 1,000 in the first year. | Potential clients that may visit the Tampa Mane Fix store front at 1249 Bruce B Downs Blvd., Wesley Chapel, FL 33643 or who will help endorse its services. |
| Windows Form App | Same as number of players | All players will sign-up through the established with the Windows Form already and data will be written to the database. |
| SQLite Database | 20 Gigabytes | Store players progress and score using a unique identifier that does not consist of any Personally Identifiable Information. |
| Administrator  (other) | One | The person responsible for the maintenance and added features of the Mane Game. |

**2.C. *Operating Environment***

**OE-1:** The Mane Game prototype must operate in a Windows 10 environment.

The Mane Game shall operate on the following web browsers: Windows Internet Explorer

versions 7 and higher; Firefox versions 12 and higher; Google Chrome (all Versions); Apple Safari

versions 4.0 and higher. The Mane Game will eventually operate on iOS and Android devices.

**2.C.1. Design and Implementation Constraints**

**DIC-1:** Level objects must be stored in a List.

**DIC-2**: Database should update when the player closes the program and when a new level is

reached.

**DIC-3:** A message box must appear if the application fails to connect with the database and notify

the player that if they continue to exit the program progress will be lost.

**DIC-4:** Player icon must be in perpetual motion and react to user input.

**2.C.2. Assumptions and Dependencies**

**A&D-1:** Users of the prototype will have access to a Windows 10 environment.

**A&D-2:** Prototype users will be able to access the SQLite Database storing player information.

**3. Architecture Specification**

**3.A. *Level Class***

The Level class defines the Graphical User Interface (GUI) that the user interacts with the Mane

Game. All other classes (objects) will interface with each Level as defined by the programmer.

Level(s) will be stored in a List that allows additional for Levels in the future. Players will iterate

through the List of Level(s) in sequential order.

**3.A.1.** Each Level will increase the speed of objects the player is intended to avoid.

**3.A.2.** Each Level will add obstacles for the player to navigate.

**3.B. *Player Class***

The Player class will construct a new Player object whose data will stored in the database for

future access. At a minimum, the Player class will have fields for:

|  |  |
| --- | --- |
| **playerID** | Unique identifier automatically assigned to the player. |
| **playerName** | Must be normalized and sterilized prior to writing to the database. |
| **playerScore** | Visible to the player and can only be modified by other methods. |
| **playerLevel** | Visible to the player and only modified by other methods |

**3.C. *BenefitObject Class***

The BenefitObject class defines the basic constructs of a beneficial object to the player. The class

will be open for extension and closed for modification. All extension BenefitObject(s) will have the

base characteristics. Standard BenefitObject(s) may be stored as an ENUM, as the programmer

determines what is best suited for the case (switch-case).

|  |  |
| --- | --- |
| **scoreIncrease()** | Method that increases the player’s score. |

**3.C.1.** HairCut object increases score by 15.

**3.C.2.** BlowDryer object increases score by 5.

**3.C.2.** Foil object increases score by 25.

**3.D. *DeductionObject Class***

The DeductionObject class defines the basic constructs of objects that deduct points from the player.

These objects will decrease the players score and temporarily slow the player’s movements on the

board.

|  |  |
| --- | --- |
| **scoreDeduction()** | Method the decreases the player’s score. |
| **slowDown()** | Method that slows the player’s movement across the board. |

**3.D.1.** HealthInspector object deducts 10 points and slows the player for 5 seconds.

**3.D.2.** LandLord object deducts 25 points and slows the player for 1 second.

**3.D.3.** TaxCollector object deducts 50 points and slows the player for 3 seconds.

**3.E. *ManeDB Class***

The ManeDB class defines methods to connect, read, and write data to the database.

|  |  |
| --- | --- |
| **dbConnect()** | Method that connects the application to the database. |
|  | Must be normalized and sterilized prior to writing to the database. |
| **playerScore** | Visible to the player and can only be modified by other methods. |

**4. Quality Attributes**

Which 3 quality attributes listed in Table 14-1 on page 289 do you feel will be the most important ensuing user acceptance and usage of this portion of the system? Write an acceptance test for each. Acceptance test example on page 293.

***4.A. Security***

**4.A.1.** (attribute description)

**4.A.2.** (acceptance test)

***4.B. Portability***

**4.B.1.** (attribute description)

**4.B.2.** (acceptance test)

***4.C. Usability***

**4.C.1.** (attribute description)

**4.C.2.** (acceptance test)