Software Requirements Specification

for

Vrei sa fii miliardar in Lei

Version 1.0 approved

Vlad Corduneanu, Alexandru H. Baciu

and Marian Haralamb

1306A

Facultatea de automatica si calculatoare

Gh. Asachi Iasi

18 May 2020

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Vlad Corduneanu	20.05.20	First Writting	1.0

1. Introduction

1.1 Purpose

Vrei sa fii miliardar in Lei is a general knowledge game with the same rules as the Vrei sa fii miliardar Romanian TV show. The game is a single player that wants to offer the possibility to see how it would've been to participate in this contest while sitting comfortably in a couch or at a desk. The in-game images and design depict the show's atmosphere while also including dialogues with its famous presenter Virgil Ianțu. The game play as well as the details and specifications of the functionality of this game are defined in sections 3 and 4. An overview of the game is given in section 2, and a list of requirements is given in section 5. This documentation will be for the final version of the game.

1.2 Document Conventions

This document follows the IEEE standard formatting for software development. The standard defines a regular formatting which this document follows including writing to be done in third-person, passive voice as well as readable and grammatically correct text.

1.3 Intended Audience and Reading Suggestions

This document is not intended for the end user because it provides a detailed specification of how the software is to be implemented. Since a user needs information on how to play the game, instead of how to make the game, this document is more geared towards testers and mostly the developers of the game. The document starts off with an overview of the functions and specifications for this game in section 2, then moves on to describe the requirements for interfacing with external hardware and software in section 3. Section 4 describes the game functions in great detail and section 5 lists various requirements the game must adhere to after completion. It is suggested that all audiences of this document start with section 2 first to get a general idea of the software requirements. Testers should next read sections 5.1 through 5.4 (performance, safety, security requirements and software quality attributes). This is to get an idea of how the game will affect them and the system they are running it on, as well as the aspirations for quality. Next a tester should read section 3.1 (user interfaces) followed by all of section 4 (system features). Reading the document in this order will give the tester an idea of what to expect in the interface at first glance, and then they may test all the individual functions to make sure they adhere to the specifications. After reading section 2, game developers should read the remaining sections in order because this document was designed specifically for the purpose of developing the game. The developer needs to get an overall idea in section 2 of the game. Then, how it needs to interface with everything else in section 3 (so they have an idea of what tools to use and possibly how they should use them). Section 4 is the most important to a developer because it describes all the functions of the game in great detail and it will help with making decisions in writing actual code for the game. Section 5 is considered least important but the developer should still read it to make sure their game has adhered to the given ideals.

1.4 Product Scope

Vrei sa fii miliardar in Lei it's a general knowledge game developed on DOT NET platform. It only supports single player mode and is designed to test your knowledge on different fields like math, literature e.g. . The player can always check his evolution because of the score section and compare his score with the other users on that computer.
The game is easy to play and in case the user needs additional information, he can always access the help section in the menu.

1.5 References

N/A

2. Overall Description

2.1 Product Perspective

Vrei sa fii miliardar in lei is a new general knowledge game developed in C#. It wants to offer the experience of participating in the show with the same name that was popular on TV a few years ago recreating the atmosphere of the TV contest with suggestive images and the game rules. In this game the player will be able to answer to questions with helping options in a given time with the purpose of winning the grand prize, 1 000 000 000 Lei. For the helping options the player will be able to use the 50/50 option which will cut 2 wrong answers, the call a friend option which will highlight the answer a friend would give and the ask the public option which will show in percent how the public thinks the question should be answered. The last 2 options are not always correct, which imitates the real feeling of the show.

The application will be developed with Windows Forms framework, the user interface and the interactions with it being the most important part of the application.

2.2 Product Functions

Major functions that Vrei sa fii miliardar in Lei must perform for its end user is as follows:

- *Offer the experience of participating in the real contest - The game must resemble the original shows atmosphere and give the player the sensation of actually participating in the tv show contest.*
- *Keep the progress of every player – The user must observe his progress in the game and also see what other players managed to achieve in order for this game to be competitive*
- *Be easy to use – The game must be easy to play in order to be an enjoyable and relaxing experience for everyone*

2.3 User Classes and Characteristics

This game should be designed for ease of use, thus any user class that accepts gaming as a past time should be able to sit down and play. Interfaces and options should be simple, descriptive and easy to navigate. The simplicity of this game merged with the TV shows theme should make the user feel like in those relaxing evenings when the show would be aired, with the exception that now he is the one in the center of attention.

2.4 Operating Environment

Since our application was developed using Windows Forms framework, it can be runed on any computer that runs on windows. Users on a PC with a Windows operating system shouldn't have a problem running the software. It is being developed in C#, and thus the user is required to have a compatible version of the 3.5 .NET Framework installed on their system, which can be found free, at Microsoft website.

2.5 Design and Implementation Constraints

This application is developed as a single semester project for the university and because of this, time is possible to be a limited resource for the entire team. Because we are students, we also might have problems caused by lack of experience working in organized projects. This could lead to longer times to develop certain features and also to less efficient solutions for some problems.

2.6 User Documentation

The application is documented in this file and also in the associated help that can be accessed when running the application in the help section. The document accessed in the help section is oriented for the end-user while this piece is for the people interested in the development and functionality of this app.

2.7 Assumptions and Dependencies

We will be depending upon some previous code written to encrypt data. We also depend on the Windows Forms framework to have the possibilities of developing the interface for our needs that are further detailed in this document in the next section. We also depend on how database connections are done in C# and some research will be done in this direction.

3. External Interface Requirements

3.1 User interface

3.1.1Description and Priority

The user interface is really important in this project because almost all the functionality of the app is based on how the user interacts with the menu and with the game interface. The game interface is represented by the answer buttons, the score board and the 3 helping options.

3.1.2Stimulus/Response Sequences

From the menu the user should be able to access the options section, where he has the settings for the user account, the score section, the help section and the start game option. In the game the user should be able to answer the questions in a given time and see his progress after every question. He should be able to choose helping options during the allowed time of answering and he should also have information about his answer after every question. The game should end when he loses (the time runs out or he chooses a wrong answear) or when he wins (answers all the questions correctly). A game end page should appear with the option to go back to the menu

3.1.3Functional Requirements

- *Have a menu with Exit, Score, Help, Options and Start Game buttons*
- *Have a game interface with a shown question, 4 answer buttons, 3 helping option buttons and timer shown*
- *Have an end game with a back to menu button*
- *Have an Options view where the user information can be altered*

3.2 Hardware Interfaces

The necessary hardware to run this app is the usual for any PC (mouse, keyboard and monitor).

3.3 Software Interfaces

This application is not required to communicate directly with other software, the communication it needs should be covered by C# libraries available in the framework we are using.

3.4 Communications Interfaces

This application should be able to communicate with a SQL database in order to store and access data.

4. System Features

Our system features are also covered in the class diagram.

4.1 User and questions database

4.1.1 Description and Priority

The database should contain the user's information like username, password and the progress in the game. This is one of the features that should be implemented as fast as possible because the implementation and testing of other functionalities depend on this

4.1.2 Stimulus/Response Sequences

It should have a simple interface via which the developers can interact with the database, like get, modify and delete data in it. No complicated sequences are necessary, because the data is not complex.

4.1.3 Functional Requirements

- *Have Get functions for the users and the questions*
- *Have Delete functions for the user*
- *Have Modify functions for the user*

4.2 Login

4.1.1Description and Priority

The Login function is important because we want to store data about every user's progress in order to show the evolution of each player and the top of the best scores. The priority for this is medium because the development of the other parts of the application need data from this, but only for development purpose it can be hardcoded till this part is done. The password needs to be encrypted and because of this an encryption module is necessary.

4.1.2Stimulus/Response Sequences

The first thing that should appear when the app is opened is the login menu, with the possibility of creating a new user for the application. The credentials will be the username and the password. After the login is done, the user should be taken to the principal menu page from where he can choose what he wants to do next.

4.1.3Functional Requirements

- *Login should be done with username and password*
- *Possibility of creating a new user*
- *The password should be encrypted in the database*

5. Other Nonfunctional Requirements

5.1 Performance Requirements

This app should run on a windows platform with the usual resources.

5.2 Safety Requirements

No safety requirements are necessary for this app.

5.3 Security Requirements

The password in the database is crypted. No private information other than the progress, username and password is stored.

5.4 Software Quality Attributes

This software must be robust and as bug-free as possible to ensure the players have a positive experience. The game should be easy for a beginner to pick up and get started, with a minimal learning curve. The game should be flexible enough to allow the easy creation of additional content, while preserving the ease of use to the consumer.

5.5 Business Rules

It is the policy of the development team to follow all codes of conduct established by the University and also the rules established by the team at the beginning of the developing process.

6. Other Requirements

Appendix A: Code Annexes

Viewer
Presenter
ModelController

Appendix B: Analysis Models

A class diagram is available offline.
A sequence diagram is available offline.
A cases diagram is available offline.
An activities diagram is available offline.