Software Requirement Specification Document
TEDx Manert El Farouk Web Application

Mohamed Tarek Ali, Ahmed Mohamed Soufy,Foad Osama Abdellatef,Moataz Samir Elsayed

March 11, 2019

1 Introduction

1.1 Purpose of this document

Its specify in Reserving tickets , managing events and manage the organization.

1.2 Scope of this document

Make a web application that is user friendly and educational where this software will help the organization to organize and manage the event from reserving tickets and help plan the budgets and the admin can hand tasks for each different department and users

1.3 Overview

Provides a brief overview of the product defined as a result of the requirements elicitation process.

1.4 Business Context

It helps the User to book tickets easily and mange the budget of organization and help the stakeholder to convert from the old fashion paperwork to technology.

2 General Description

2.1 Product Functions

Describes the general functionality of the product, which will be discussed in more detail below.

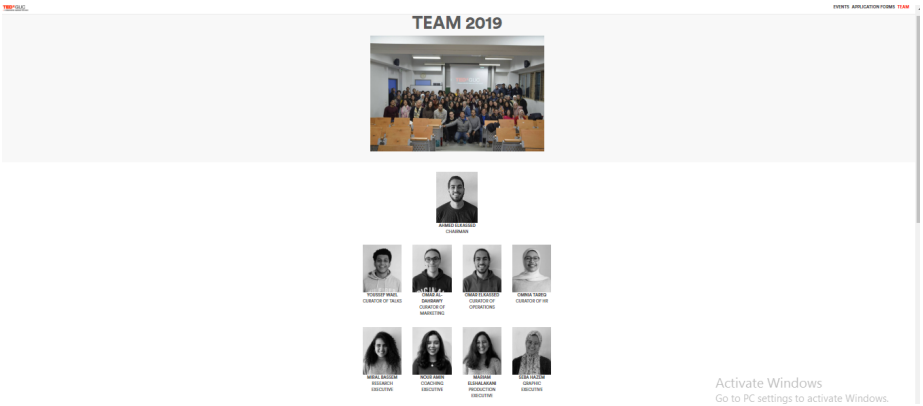

Figure 1: Members of TedxGUC.

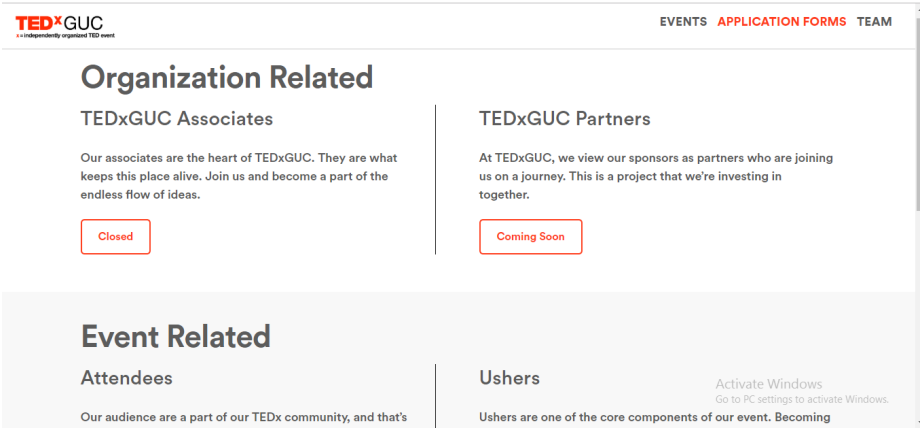

Figure 2: TedxGUC Eve.tsn

Figure 3: TedxGUC Hom.Pagee

Figure 4: TedxAmsterdam HomePage.

2.2 Similar System Information

we have the same reservation for events and showing the event team who work on it

2.3 User Characteristics

Using paper works and google forms

2.4 User Problem Statement

No forms reliability or flexibility tickets Reservation user information privacy of data

2.5 User Objectives

ticket information about us permission collecting information about products and user budget processing

2.6 General Constraint

3 Functional Requirements

4 Interface Requirements

This section describes how the software interfaces with other software products or users for input or output. Examples of such interfaces include library routines, token streams, shared memory, data streams, and so forth.

Name	getData
Triggers	it get triggered when we fetch data from DB
Input	tablename,columnName
Pre-condition	database isn't empty
Post-condition	fetch data
Output	Data from database
Description	getting general data from database
Basic flow	no
Exceptions	no

Name	getUserData
Triggers	it get triggered when we fetch user data from DB
Input	id
Pre-condition	database has user data
Post-condition	fetch data
Output	User from database
Description	getting User data from Data Base

Name	insert
Triggers	it get triggered when we put data from DB
Input	id
Pre-condition	database has user data
Post-condition	fetch data
Output	User from database
Description	getting User data from Data Base

4.1 User Interfaces

Use some software for primitive plan of your project. Describes how this product interfaces with the user.

4.1.1 GUI

Describes the graphical user interface if present. This section should include a set of screen dumps or mockups to illustrate user interface features. If the system is menu-driven, a description of all menus and their components should be provided.

4.1.2 CLI

No CLI is present.

4.1.3 API

TED API: is one of the API that are Ready made that we will use in the project to fetch data like the Talks from different Talks from TED

4.1.4 Diagnostics or ROM

Describes how to obtain debugging information or other diagnostic data.

4.2 Hardware Interfaces

the hardware needed are a computer and a router/server

4.3 Communications Interfaces

Router/servers

4.4 Software Interfaces

Windows

5 Performance Requirements

the web application must efficient and fast

6 Design Constraints

Specifies any constraints for the design team using this document.

6.1 Standards Compliance

6.2 Hardware Limitations

6.3 others as appropriate

7 Other non-functional attributes

Specifies any other particular non-functional attributes required by the system. Examples are provided below.

7.1 Security:

Protecting data regarding the personal information of any user when collecting data to use in analysis

Binary Compatibility:

7.2 Reliability:

Must be Reliable

7.3 Maintainability:

it's maintainable

7.4 Portability:

It can fit on every screens

7.5 Extensibility:

7.6 Re-usability:

7.7 Application Affinity/Compatibility:

7.8 Resource Utilization: Using data from users and analyze it to use it to improve the sorting of data in the systems.

7.9 Serviceability:

7.10 others as appropriate

8 Preliminary Object-Oriented Domain Analysis

This section presents a list of the fundamental objects that must be modeled within the system to satisfy its requirements. The purpose is to provide an

alternative, "structural" view on the requirements stated above and how they might be satisfied in the system. A primitive class diagram to be delivered.

8.1 Inheritance Relationships

This section should contain a set of graphs that illustrate the primary inheritance hierarchy (is-kind-of) for the system. For example:

8.2 Class descriptions

This section presents a more detailed description of each class identified during the OO Domain Analysis. For more details on the process giving rise to these descriptions, see Lecture 5.3: OO Domain Analysis and/or texts on object-oriented software development. Each class description should conform to the following structure:

8.2.1 Class name

Abstract or Concrete: Indicates whether this class is abstract or concrete.

8.2.2 List of Superclasses:

Names all immediate superclasses.

8.2.3 List of Subclasses:

Names all immediate subclasses.

8.2.4 Purpose:

States the basic purpose of the class.

8.2.5 Collaborations:

Names each class with which this class must interact in order to accomplish its purpose, and how.

8.2.6 Attributes:

Lists each attribute (state variable) associated with each instance of this class, and indicates examples of possible values (or a range).

8.2.7 Operations

: Lists each operation that can be invoked upon instances of this class. For each operation, the arguments (and their type), the return value (and its type), and any side effects of the operation should be specified.

8.2.8 Constraints:

Lists any restrictions upon the general state or behavior of instances of this class.

9 Operational Scenarios

This section should describe a set of scenarios that illustrate, from the user's perspective, what will be experienced when utilizing the system under various situations. In the article Inquiry-Based Requirements Analysis (IEEE Software, March 1994), scenarios are defined as follows: In the broad sense, a scenario is simply a proposed specific use of the system. More specifically, a scenario is a description of one or more end-to-end transactions involving the required system and its environment. Scenarios can be documented in different ways, depending up on the level of detail needed. The simplest form is a use case, which consists merely of a short description with a number attached. More detailed forms are called scripts. These are usually represented as tables or diagrams and involved identifying an action and the agent (doer) of the action. FOr this reason, a script can also be called an action table. Although scenarios are useful in acquiring and validating requirements, they are not themselves requirements, because the describe the system's behavior only in specific situations; a specification, on the other hand, describes what the system should do in general.

10 Preliminary Schedule Adjusted

This section provides an initial version of the project plan, including the major tasks to be accomplished, their interdependencies, and their tentative start/stop dates. The plan also includes information on hardware, software, and resource requirements. The project plan should be accompanied by one or more PERT or GANTT charts.

11 Preliminary Budget Adjusted

This section provides an initial budget for the project, itemized by cost factor.

12 Appendices

Specifies other useful information for understanding the requirements. All SRS documents should include at least the following two appendices:

12.1 Definitions, Acronyms, Abbreviations

Provides definitions of unfamiliar definitions, terms, and acronyms.

12.2 Collected material

13 References