Software Requirements Specification

For

<BRACU Entrance System>

Prepared by

Sadman Sakib Mridul (17101157)

S.M. Ali Ijtihad Nasif (16301054)

Contents

1. Introduction

Students, faculties and staffs are core elements around which a university is made. So, it is vital for a university to have a good entrance system so that the university can maintain its integrity and keep it a safe place for its students.

The process of upgrading the entrance system can be started by looking at the pros and cons of the current system so that it can be used as a base when creating a new and better version of an entrance system.

The next step is to add features that will help make the system more efficient and authentic. The system will adapt new technologies to cover up for the cons the old system had.

1.1. Purpose

The University Entrance system is a system that will make the entrance system faster and efficient at the same time will provide better authentication. We would like to emphasize on the system's capability to better handle the large amount of student body a university has with proper authentication system.

1.2. Documents Convention

This document uses the following conventions.

DB	Database
DDB	Distributed Database
ER	Entity Relationship

1.3. Intended Audience and Reading Suggestions

This project is a prototype for the entrance system and it is restricted within the university premises. This has been implemented under the guidance of university professor and will be evaluated by faculties, staffs and students.

1.4. Product Scope

The projects aim is to automate the system and make the security system better and efficient. The project will check the students, faculties and staffs from previous date inputs stored in its database. The data used by the system is stored in a database that will be the center of all information held about every one of the universities and the base for authentication of IDs and personnel.

1.5. Reference

- IEEE 830-1998 standard for writing SRS document
- Software engineering a practitioner’s approach by roger s.pressman_7th.pdf

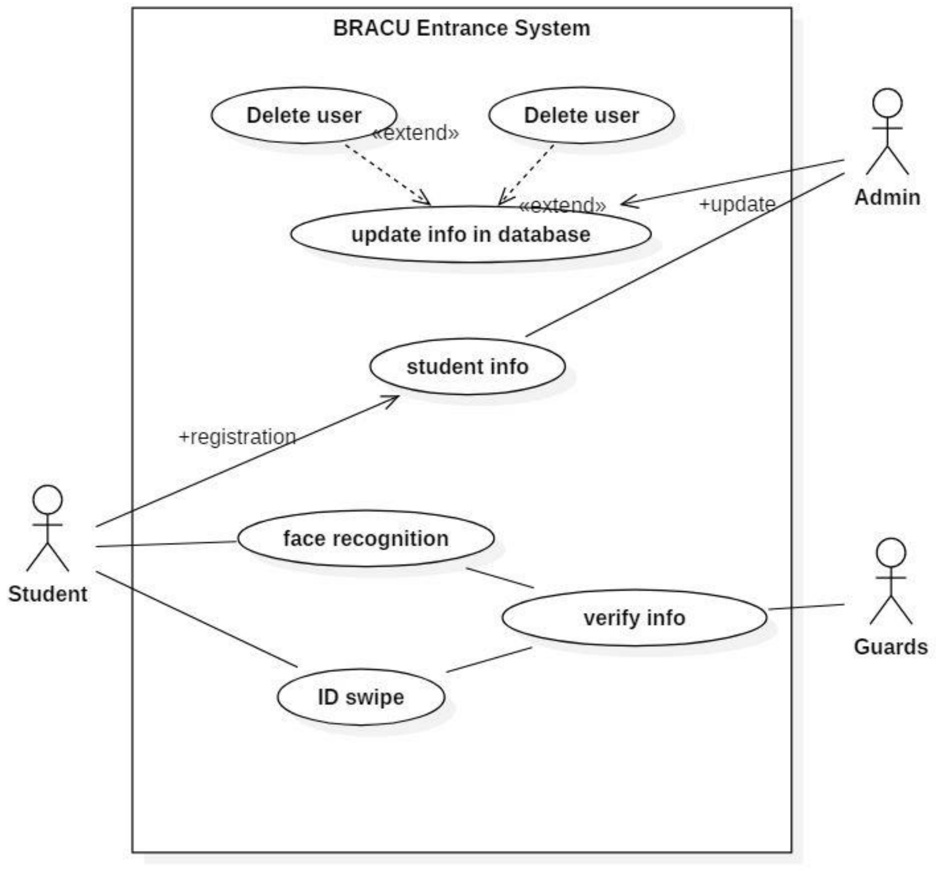
2. Overall description

2.1. Product Perspective

The aim of the software is to make the university more secure and make the entry system more efficient with the collected information about the students, faculties and staffs. The system will allow the security system to keep track of the entry and exit timing of all the body of the university. Our goal is to extend this system to other universities and work places as this will help make the entire system more secure and efficient.
The software should be user-friendly, “quick to learn”, and reliable software for the above purpose. This software is intended to be a stand-alone product and should not depend on the availability of other software. It should run on both UNIX and windows-based software.

2.2. Product Function

Use case diagram



2.3. User classes and characteristics

All

All stakeholders share the following key needs:

1. Security against abuses by other site visitors

Administrators

Make updates and add or remove contents. They can also set up configurations

Executives

They are not directly involved or puts any impacts.

Students

They are the key users in this system as this system is made for the student to verify their identity.

2.4. Operating Environment

Operating environment for the entrance system is as listed below.

- Distributed database
- Cross-check and authentication from database
- Operating system: Windows, Linux
- Database: SQL+, database
- Platform: C+/Java/Python

2.5. Design and Implementation Constraints

1. The global schema, fragmentation schema, and allocation schema.
2. SQL commands to recheck information from database for authentication
3. How the response to different scenarios will be generated. Explain how various fragments will be combined to do so.
4. Implement the database at least using a centralized database management system.
5. Proper security checks.

2.6. Assumptions and Dependencies

It is used in the following application:

Automatically searches student information, compare for authentication, gives a green signal for successful check and gives a red signal for invalid ID.

Assuming both the transactions are single transactions, we have designed a distributed database that can keep the record of students, faculties and staff's information, their facial structure record and card swiping information.

3. Interface Requirement

3.1. User Interfaces

- Front-end software: java version,
- Back-end software: SQL+

3.2. Hardware Interfaces

- Windows
- A browser which supports CGI, HTML & JavaScript

3.3. Software interfaces

Software used	**Description**
Operating system	We have chosen Windows operating system for its best support and user-friendliness.
Database	To save the student, faculty and staff records we have chosen SQL+ database.
Java	To implement the project, we have chosen java language for its more interactive

3.4. Communications Interfaces

This project is a closed-circuit project and is only accessible by the admins and head of security. The guards can only use some of the features the project holds. It is not available online because of security reasons.

4. System Features

4.1. Functional Feature 1

4.1.1. Description and Priority

This system has a great impact in BRAC university as it holds the info of all the students and who are in university and when enter. This system has a higher priority as is ensure the security of the students as well as the university.

4.1.2. Response sequence

- Search for information of the students as they enter through the scanner
- Display the information of the students in the screen in front of the guards

4.1.3. Functional Requirements

- **User stories**
  Students of BRAC university faces many problems when they try to enter in university in a large number where guards take huge amount of time to check student's id one by one. So, the students want a system to be faster and safer (Source: Interview)
- **Guards**
  according to guards it a very hectic job to check every id of students when its class time and a huge number of students wants to enter all together.so they want a system which is quick, easy to handle and ensure safety. (Source: Interview)

4.2. Non-Functional Feature

4.2.1. Performance Requirement

- The system should accommodate high number of students, faculty and staff list names and ID without any fault
- Responses to view information shall take no longer than 5 seconds to appear on the screen.
  The steps involved to perform the implementation of database are as listed below.

A) E-R DIAGRAM

The E-R Diagram constitutes a technique for representing the logical structure of a database in a pictorial manner. This analysis is then used to organize data as a relation, normalizing relation and finally obtaining a relation database.

- ENTITIES: Which specify distinct real-world items in an application.
- PROPERTIES/ATTRIBUTES: Which specify properties of an entity and relationships.
- RELATIONSHIPS: Which connect entities and represent meaningful dependencies between them.

B) NORMALIZATION:

The basic objective of normalization is to reduce redundancy which means that information is to be stored only once. Storing information several times leads to wastage of storage space and increase in the total size of the data stored.

If a database is not properly designed it can give rise to modification anomalies. Modification anomalies arise when data is added to, changed or deleted from a database table. Similarly, in traditional databases as well as improperly designed relational databases, data redundancy can be a problem. These can be eliminated by normalizing a database.

Normalization is the process of breaking down a table into smaller tables. So that each table deals with a single theme. There are three different kinds of modifications of anomalies and formulated the first, second and third normal forms (3NF) is considered sufficient for most practical purposes. It should be considered only after a thorough analysis and complete understanding of its implications.

4.2.2. Safety Requirements

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed-up log, up to the time of failure.

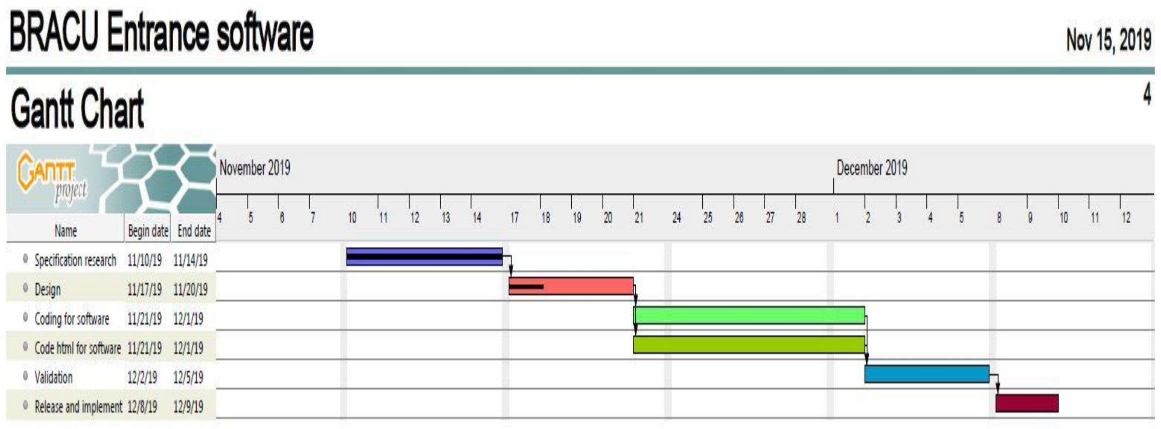
4.2.3. Security Requirements

- System will use secured database
- Normal users can just read information but they can’t edit or modify anything except their personal information.
- The admins and the head of security have access to information not available to the general body.

4.2.4. Software Quality attributes

- **VALIDITY:** The student, faculty and staff’s information and ID should be valid for successful authentication.
- **CORRECTNESS:** The system should record information without any wrong knowledge or based on any assumption.
- **MAINTAINABILITY:** The administrators and head of security should maintain correct information of the personnel.
- **USABILITY:** The information should satisfy the requirements needed for the people to enter the university premises.

4.2.5. Gantt chart



Name	Begin date	End date
Specification research	11/10/19	11/14/19
Design	11/17/19	11/20/19
Coding for software	11/21/19	12/1/19
Code html for software	11/21/19	12/1/19
Validation	12/2/19	12/5/19
Release and implement	12/8/19	12/9/19

5. Other Requirements

5.1. Testing and Results

The reason behind testing was to find errors. Every program or software has errors in it, against the common view that there are no errors in it if the program or software is working. Executing the programs with the intention of finding the errors in it is therefore testing; hence a successful test is one which finds errors. Testing is an activity; however, it is restricted to being performed after the development phase is complete, but is carried parallel with all stages of system development, starting with requirement specification.

Test cases were devised with a purpose in mind. A test case is a set of data that a system will process as normal input. The software units developed in the system are modules and routines that are assembled and integrated to perform the required function of the system. Test results once gathered and evaluated, provide a qualitative indication of the software quality and reliability and serve as the basis for design modification if required. In this phase, testing is done at different levels. Actually, testing phase of the implementations works accurately and efficiently before live operation commences.

5.2. Unit testing

Unit testing was done after the coding phase. The purpose of the unit testing was to locate errors in the current module, independent of the other modules. Some changes in the coding were done during the testing phase. Finally, all the modules were individually tested following bottom to top approach, starting with the smallest and lowest modules and then testing one at a time.

5.3. Black Box Testing

This method of software testing tests the functionality of an application as opposed to its internal structures or working (i.e. white box testing). Specific knowledge of the application's code/internal structure and programming knowledge, in general, is not required. Test cases are built to specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and design to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid inputs and determines the correct output. There is no knowledge of the test object's internal structure.

5.4. White Box Testing

This method of software testing tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing, an internal perspective of the system, as well as programming skills, are required and used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs.

5.5. Integration Testing

Once the unit was over, all the modules were integrated for integration testing. External and internal interfaces are implemented and work as per design, the performance of the module is not degraded.

5.6. Validation Testing

At the culmination of integration testing, the software is said to be completely assembled as a package; interfacing errors have been uncovered and corrected. Then as a final series of software test, validation tests were carried out.

5.7. Acceptance Testing

This is the final stage in the testing process before the system is accepted for operational use. Any requirement problem or requirement definition problem revealed from acceptance testing are considered and made error free.

6. Test case for BRACU entrance system

Test Scenario ID	Student 1
Test Case Description	Login – Positive test case
Pre-Requisite	A valid user account

S.No	**Action**	**Inputs**	**Expected Output**	**Test Result**	**Test Comments**
1	Should stand Infront of the camera	Image processed	verified		
2	Should swipe the id card in the scanner	Scan QR code	Login success		