**SRS FOR AUTOMATED MARKS UPLOADING SYSTEM**

**Table of Contents**

**1. Introduction**

1.1 Purpose

1.2 Scope

1.3 Overview

**2. General Description**

**3. Functional specifications**

3.1 Login

3.2 Validation

3.3 Select type of exam

3.4 Attach answer sheet

3.5 Recognise the desired data from the sheet

3.6 Verify and upload data

**4. Interface Requirements**

4.1 GUI

4.2 Hardware Interface

4.3 Software Interface

**5. Performance Requirements**

**6. Constraints**

**7. Performance**

7.1 Security

7.2 Reliability

7.3 Availability

7.4 Maintainability

7.5 Reusability

**1. Introduction**

This document gives details about the various functional and non-functional requirements for automated marks uploading system. This software will read marks of a student from the first page of answer sheets and automatically upload it on the database.

**1.1 Purpose**

Traditional way of uploading marks of a student in database requires some user (operator/teacher) to manually map marks of each student against his/her account. In this process, user requires to first open the portal for a particular student, then read marks from the answer sheet of that student, then map the marks into the field of that subject and then finally upload it. This long process may be prone to errors like incorrect marks are read from the sheet, some typing error may occur while uploading marks, marks of any other student could be mapped against any other student or marks of any other subject could be mapped against any other subject by mistake. So, this all makes us feel the need of an automated system that should be effective enough to eliminate all of these manual tasks and do all of this on our behalf precisely and automatically. In this project I will try to implement the same.

**1.2 Scope**

This project will automate the process of uploading marks by reading the scanned image of first page of an answer sheet and then recognise roll number, subject code and marks obtained by using some machine learning algorithms and then map those marks against that subject for the student whose roll number was recognised. Then it will ask user to verify the data before uploading it to the database for two-factor verification.

**1.3 Overview**

The system will provide easy solution to college management.

The SRS will include two sections, namely:

Overall Description: This section will describe major components of the system, interconnections, and external interfaces.

Specific Requirements: This section will describe the functions of actors, their roles in the system and the constraints faced by system.

**2. General description**

#### 2.1 Product Perspective:

Software will have an interface that will enable user to upload the front page of answer sheet and then interact with the college database to update the marks of a student. Before uploading, marks will also undergo several computations as per the weight-age of those marks in final marks of student for the subject. The user will land to this interface by logging in to the system and if his login credentials are valid.

**2.2 Software Interface:**

* Front End: The system is a web based software.
* Web Server: The web application will be hosted by using HelioHost testing server.
* Back End: We may use back-end as MY SQL database .

**3. Functional Specifications**

This section provides the functional overview of the product. The project will require the C# as a front end and at the back end MYSQL database will be running. Various functional modules that can be implemented by the software will be

1. Login
2. Validation
3. Select type of exam
4. Attach answer sheet
5. Recognise the desired data from the sheet
6. Verify and upload data

**3.1 Login**

User logs in through this module to the software by using the employee ID and password provided by college administration.

**3.2 Validation**

This module checks whether the person trying to login is authorised by college administration or not. It matches for the Employee ID in college database that carries list of authorised users and it checks whether the password is correct or not. Finally if both the credentials are valid, it lets user to login to the system.

**3.3 Select type of exam**

Here user will select the type/name of exam (MTE or ETE). On the basis of this selection computations will be done over the marks and then it will be uploaded accordingly.

**3.4 Attach answer sheet**

Here user will simply select the image file of an answer sheet that is already scanned and residing into the user’s computer.

**3.5 Recognise the desired data from the sheet**

At this stage, software will recognise the earlier mentioned data from the attached image file using some ML algorithms.

**3.6 Verify and upload data**

Finally, user will be asked to verify whether the data recognised, matches with actual data on the sheet or not. After user’s approval, concerned computations will be done over the marks and then marks will be uploaded to the database.

1. **Interface Requirements**
   1. **GUI**

* GUI of the software should be simple enough for users, so that he/she doesn’t faces any issues in understanding it while working on it. It should be free from conflicts and un-clarity. It should be interactive because there will be no one to assist the user about its working.
* Also the password confidentiality should be maintained, this can be done by using asterisks at the password panel.
* Proper security messages should be displayed where-ever required.

**4.2 Hardware Interface**

Various interfaces for the product could be:

1. Monitor
2. Keyboard
3. Scanner which can scan the answer sheet’ first page
4. Interface that connects the scanner with the system.

**4.3 Software Interface**

1. Any windows operating system.
2. Software will be developed on Microsoft Visual Studio 2017 Community Edition.
3. C# language will be used for development purpose.
4. For the database handling MYSQL will be used & that will be hosted by HelioHost.
5. Python should be installed on the machine or should be able to be integrated with platform used for development for ML related tasks.
6. The final application must be packaged in a set up program, so that the products can be easily installed on machines.
7. **Performance Requirements**

The machine should be compatible enough to install and run the software. There shouldn’t be any heavy system requirements to run the software.

1. **Constraints**

* The information of all the authorised users must be stored in the database so as to give the access to the software.
* Every subjects taught in the university should must stored in a separate database along with their subject codes.
* Details of every student should must be stored on the database along with their roll numbers.
* The users access the software from any computer that is connected to the internet.
* The users should have their correct IDs and passwords to enter into the login page.

1. **Performance**

**7.1 Security**

The banking system must be fully accessible to only authentic user.

**7.2 Reliability**

The application should be highly reliable and it should recognise all the data precisely.

**7.3 Availability**

Any information about the student and subject should be quickly available from the database to the authorized user.

**7.4 Maintainability**

The software should be maintainable in such a manner that if any data is recognised wrong, then it should be easily correctable.

**7.5 Portability**

The application should be portable on any windows based system. It should not be machine specific.