[1. Introduction](#h.ijxfblb4p34q)

[1.1 Purpose](#h.wb4ats9pjl7p)

[1.2 Scope of Project](#h.bwnbccduun4x)

[1.3 Glossary](#h.x5y4wypp2q1f)

[1.4 References](#h.i1nb5wkc7aca)

[1.5 Overview of Document](#h.asui6wuccury)

[2. Overall Description](#h.vzacybik0ox3)

[2.1 Product perspective](#h.jlveoxoxgke7)

[2.2 Product functions](#h.8uj8fgp4fpqc)

[2.3 User characteristics](#h.srihyy8sl2b)

[2.4 General Constraints](#h.y6c6snmz7vcj)

[2.5 Assumptions and dependencies](#h.x6ls08c5nnkt)

[2.6 Apportioning of requirements](#h.5puc28pnm2pn)

[3. Specific Requirements](#h.jqbdjtbzzhlo)

[3.1 External Interface Requirements](#h.nes17u6s2tte)

[3.1.1 User Interfaces](#h.x4ytaieaqugw)

[3.1.2 Hardware Interfaces](#h.nhue22oaqjck)

[3.1.3 Software Interfaces](#h.mamfe5sn2uyi)

[3.1.4 Communications Interfaces](#h.52l01lde1jn)

[3.2 Functional Requirements](#h.9ujq4og1ez4v)

[3.2.1 <Functional Requirement or Feature #1>](#h.vc94gds6mp97)

[3.2.2 <Functional Requirement or Feature #2>](#h.lgmccoz7puoj)

[3.3 Use Cases](#h.9c4ncyopm1kp)

[3.3.1 Use Case #1](#h.kcxe8h5k1i79)

[3.3.2 Use Case #2](#h.a3y4perrm5of)

[3.4 Classes / Objects](#h.a4pmcpldvux3)

[3.4.1 <Class / Object #1>](#h.rc804hvr4jt4)

[3.4.2 <Class / Object #2>](#h.3nx3sok4f4y7)

[3.5 Non-Functional Requirements](#h.b4jdh8kqn37k)

[3.5.1 Performance](#h.l8nemkfu0p58)

[3.5.3 Availability](#h.brnhgwuy6iss)

[3.5.4 Security](#h.3jhmnv70gbxv)

[3.5.5 Maintainability](#h.9pzjdmvg3npp)

[3.5.6 Portability](#h.9j9x9gbbzj1t)

[3.6 Inverse Requirements](#h.rb23gq9bsbmt)

[3.7 Design Constraints](#h.b2tq1zj2mh9u)

[3.8 Logical Database Requirements](#h.20xj30tqmppo)

[3.9 Other Requirements](#h.5fox2j6wdlny)

[4. Analysis Models](#h.xi22tq760en9)

[4.1 Sequence Diagrams](#h.2exffxg1c1pw)

[4.3 Data Flow Diagrams (DFD)](#h.749fxadx14lw)

[4.2 State-Transition Diagrams (STD)](#h.x6dtqfmwe84c)

[5. Change Management Process](#h.35nb43693ix4)

[A. Appendices](#h.yf016ehskay1)

[A.1 Appendix 1](#h.9rna28gtq466)

[A.2 Appendix 2](#h.5wcprt5lov3g)

# **1. Introduction**

This chapter gives a scope description and overview of everything included in this SRS document. Also, the purpose for this document is described and a list of abbreviations and definitions is provided.

## 1.1 Purpose

The purpose of this document is to present a detailed description of the “Syllabus Manager And Report Tracking” (SMART) software. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to a customer for its approval.

## 1.2 Scope of Project

All instructors deal with penning out the estimates of completing course syllabus for their classes on a regular basis. As they progress through their classes the process of keeping a track becomes difficult and cumbersome. The “Syllabus Manager And Report Tracking” will be a local Windows Operating System based software which will maximize the instructor’s productivity by providing tools to assist in semi-automating and digitizing the syllabus planning and progress tracking process. By maximizing the instructor’s work efficiency and production the system will meet the instructor’s needs while remaining easy to understand and use.

More specifically, this system is designed to allow an instructor, manage and track his/her course syllabus completion plan for each of their class.

## 1.3 Glossary

### Table 1 - Definitions

|  |  |
| --- | --- |
|  |  |

## 1.4 References

1. IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.
2. Software Requirements Specification Amazing Lunch Indicator.
3. Software Requirements Specification for SplitPay.

## 1.5 Overview of Document

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

# 

# 

# **2. Overall Description**

This chapter will give an overview of the whole system. The system will be explained in its context to show how the system interacts with user and introduce the basic functionality of it. It will also describe what type of stakeholders that will use the system and what functionality is available for each type. At last, the constraints and assumptions for the system will be presented.

## 2.1 Product perspective

The software system will consist of a single package.

The users first create their individual accounts storing all their course and class information. For each of their class and course the software maintains topics and subtopics retrieved from the users input. Once that is done, the software facilitate in making tentative schedules that they desire to follow. The registered users get progress reports of how they are doing in a particular course or class by giving syllabus realization details to the software.

The software can generate informative reports on each stage which can be used to further optimize, make changes to their syllabus completion planning process and report keeping.

## 2.2 Product functions

1. The software allows the users to make individual password protected accounts with information about their classes and courses.
2. There are three cases:

* users may teach multiple courses to same class.
* users may teach same course to multiple classes.
* users may teach multiple courses to multiple classes.

all of which can be easily accommodated and accounted for.

1. To preserve confidentiality no user can view the progress of other user.
2. Topic and subtopics names are shared among users undertaking the same courses. This will liberate them from re-entering those details; saving time and effort.
3. Progress tracking and report generation is possible at all stages.
4. Both soft-copies and hard-copies of the reports can be generated.

## 2.3 User characteristics

The user is expected to be a Windows OS Environment literate and be able to use selectors, buttons, input fields, menus, and similar tools.

## 2.4 General Constraints

The main component of the project is the software application, which will be limited to the Windows operating system (specifically Windows 7 and above). The application is not resource or graphics intensive, so there are no practical hardware constraints. However the software will rely on several functionalities built into the Windows OS.

## 2.5 Assumptions and dependencies

The software assumes that the user has a list of all working days and topics names available to them during the sign-up process. Since the application depends heavily on the data provided by the user it can become a daunting task to do and requires patience. Furthermore the software assumes that the input data is correct.

## 2.6 Apportioning of requirements

In the case that the project is delayed, there are some requirements that could be transferred to the next version of the application.

# **3. Specific Requirements**

This chapter contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

## 3.1 External Interface Requirements

This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.

### 3.1.1 User Interfaces

A first-time user of the application should see the log-in page when they open the application, see Figure 2. If the user has not registered, they should be able to do that on the log-in page.

If the user is not a first-time user, they should be able to see the dashboard directly when the application is opened, see Figure 3. Here the user can view each of their class and course.Clicking on a particular class or course display additional details, see Figure 4.

Every user should have a profile page where they can edit their personal details and password, see Figure 5.

### 3.1.2 Hardware Interfaces

The software does not require any specific direct hardware interfaces.

Input and output devices like the mouse, keyboard and monitor, printer are managed by the underlying Device Drivers of the operating system.

### 3.1.3 Software Interfaces

### 3.1.4 Communications Interfaces

The communication between the different parts of the system is important since they depend on each other. However, in what way the communication is achieved is not important for the system and is therefore handled by the underlying operating system.

## 3.2 Functional Requirements

### 3.2.1 <Functional Requirement or Feature #1>

3.2.1.1 Introduction

3.2.1.2 Inputs

3.2.1.3 Processing

3.2.1.4 Outputs

3.2.1.5 Error Handling

### 3.2.2 <Functional Requirement or Feature #2>

…

## 3.3 Use Cases

### 3.3.1 Use Case #1

### 3.3.2 Use Case #2

…

## 3.4 Classes / Objects

### 3.4.1 <Class / Object #1>

3.4.1.1 Attributes

3.4.1.2 Functions

<Reference to functional requirements and/or use cases>

### 3.4.2 <Class / Object #2>

…

## 3.5 Non-Functional Requirements

The requirements in this section provide detailed view of the measurements placed on the system performance as well as the required reliability, availability, security, maintainability and portability of the software system.

### 3.5.1 Performance

3.5.1.1 Usage of the UI Element

ID: QR1

DESC: The User Interface element should be prominent and it should be evident that they are usable. Interacting with each such element should be only through the mouse and/or the keyboard.

RAT: In order to for a user to use and navigate the software easily.

DEP: none

3.5.1.2 Usage of the Dashboard view

ID: QR2

DESC: The displayed dashboard view should be user friendly and easy to understand. Selecting an element should only take one click.

RAT: In order to for a user to use the dashboard easily.

DEP: none

3.5.1.3 Response time

ID: QR3

TAG: ResponseTime

GIST: The fastness of report generation

METER: Measurements obtained from 10 report during testing.

MUST: No more than 5 seconds 100% of the time.

WISH: No more than 3 second 100% of the time.

3.5.1.4 System dependability

ID: QR4

TAG: SystemDependability

GIST: The fault tolerance of the system.

SCALE: If the system loses power, crashes or the system gets some strange input, the user should be informed.

METER: Measurements obtained from 2 hours of usage during testing.

MUST: 100% of the time.

3.5.2 Reliability

ID: QR5

TAG: SystemReliability

GIST: The reliability of the system.

SCALE: The reliability that the system gives the right result on reports.

METER: Measurements obtained from 10 reports during testing.

MUST: More than 98% of the reports.

PLAN: More than 99% of the reports.

WISH: 100% of the reports.

### 3.5.3 Availability

ID: QR6

TAG: SystemAvailability

GIST: The availability of the system when it is used.

SCALE: The average system availability (not considering OS or device failure).

METER: Measurements obtained from 10 hours of usage during testing.

MUST: More than 98% of the time.

PLAN: More than 99% of the time.

WISH: 100% of the time.

### 3.5.4 Security

ID: QR7

TAG: UserLoginAccountSecurity

GIST: Security of accounts.

SCALE: If an user tries to log in to the software with a non-existing account then the user should not be logged in. The user should be notified about log-in failure.

METER: 10 attempts to log-in with a non-existing user account during testing.

MUST: 100% of the time

ID: QR8

TAG: UserCreateAccountSecurity

GIST: The security of creating account for users of the system.

SCALE: If a user wants to create an account and the desired user name is occupied, the user should be asked to choose a different user name.

METER: Measurements obtained on 10 hours of usage during testing.

MUST: 100% of the time.

### 3.5.5 Maintainability

ID: QR9

TITLE: Application extendibility

DESC: The application should be easy to extend. The code should be written in a way that it favors implementation of new functions.

RAT: In order for future functions to be implemented easily to the application.

DEP: none

ID: QR10

TITLE: Application testability

DESC: Test environments should be built for the application to allow testing of the applications different functions.

RAT: In order to test the application.

DEP: none

### 3.5.6 Portability

ID: QR11

TITLE: Application portability

DESC: The application should be portable with Windows 7,8,8.1,10.

RAT: The adaptable platform for the application to run on.

DEP: none

## 3.6 Inverse Requirements

The software system works best when used on a single system, it can’t share resources over a network.

The software is not programmed to automatically populate the workdays and topics/subtopics names for a class and course.

The software is not intended to be used on a LINUX, UNIX based distribution.

## 3.7 Design Constraints

This section specifies design constraints imposed by other standards, company policies, hardware limitation, etc. that will impact this software project.

## 3.8 Logical Database Requirements

Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc.

## 3.9 Other Requirements

Catchall section for any additional requirements.

# **4. Analysis Models**

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS’s requirements.

## 4.1 Sequence Diagrams

## 4.3 Data Flow Diagrams (DFD)

## 4.2 State-Transition Diagrams (STD)

# **5. Change Management Process**

Identify and describe the process that will be used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved.

# **A. Appendices**

Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS’s overall set of requirements.

*Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.*

## A.1 Appendix 1

## A.2 Appendix 2