Software Requirements Specification

for

Automated Timetable Generator (ATG)

Version 1.0 approved

Prepared by : Muhammad Shaheer Alam 2019369
Muhammad Afzal 2019279
Mohammad Arsalan 2019306
Maida Shafaq 2019222
Ramsha Suhail 2019435

Organization: Ghulam Ishaq Khan Institute

Date Created: December 25th 2021

Submitted to: Sir Ahsan Shah

Table of Contents

Name	Date	Reason for changes	Version

1. INTRODUCTION

1.1. Purpose

Automated Timetable Generator (ATG) is a software that is used to generate timetables automatically. The timetable is currently scheduled manually which requires the knowledge of each faculty member and student's presence without clashes which in turn demands significant time and effort. It is an extremely cumbersome, time taking process that is very error-prone too. The classes clash either in an identical room or with the same teacher having more than one class at a time.

Our aim here is to develop a simple, easily understandable, and efficient application that could automatically generate timetables within seconds.

The main objectives of ATG are:

- To avoid clashes that occur between classes and staff.
- To manage all the periods automatically.
- To avoid the complexity of setting and managing timetables manually.
- To generate timetable automatically for classes (students) and faculty.
- To reduce time utilization and effort and improve efficiency.
- To view the timetable easily.
- To focus on optimization of resources i.e. teachers, labs, classrooms, etc.

It is a comprehensive solution for all higher education institutes but particularly designed while keeping GIK Institute and its administrative realities in mind. It will help to overcome the challenges faced in setting the timetable manually.

1.2. Document Conventions

No such abbreviations or typographical conventions have been used while writing this SRS document. There are no priorities assigned to pieces of text depending on their thickness or font style. Every requirement is independent and has its priority. This document was created based on the IEEE template for System Requirement Specification Documents as provided by our instructor.

1.3. Intended Audience and Reading Suggestions

- Any individual belonging to the organization needs to design the timetable for staff and classes.
- Both faculty members and students want to view their respective timetables.
- Admin can view all the content i.e. all the timetables. Admin has the authority to manage the database which feeds the data to the solution.

1.4. Product Scope

Admin, in every institute, looks for ways to schedule timetables easily without wanting to face any difficulty. Once a clash occurs when implemented manually, correcting that clash may lead to timetable rescheduling which can become a repeated process leading to multiple revisions. Creating such timetables manually is complex and time-consuming, cumbersome and a complete mess. By automating this process with a computer-assisted timetable generator can save a lot of time for administrators who are involved in creating and managing course timetables. Therefore, Automated Timetable Generator (ATG) is designed to generate a timetable for each faculty member and class, keeping in view the availability of teachers and the physical resources i.e. classrooms, labs. Most importantly, it helps in the improvement of resource utilization and optimization as the big problematic task which occupied so many individuals will be handled by one machine.

1.5. References

IEEE template for System Requirement Specification Documents: https://goo.gl/nsUFwy

2. Overall Description

2.1. Product Perspective

The automated timetable generator is supposed to be an open-source web application. This application can be used by the administration of the universities. This will help the users to generate a clash-free timetable that will avoid any clashes and save a lot of time that is normally spent in maintaining the paperwork. The system traces its origins from the difficulties faced by the GIK Institute's students and administration as part of the scheduling of their semester classes. This software system is targeted to eliminate multiple repetitions of the timetable, with clashes still to be resolved by the respective faculties and instructors.

2.2. Product Functions

The following functions are included in the project:

- Input Courses
- Input Instructors
- Input Lectures
- Input groups for specific courses
- Generate a timetable that ensures no clashes.
- Try its level best to schedule corresponding labs before classes.
- Display the final timetable

The main function of the timetable generator is to get the appropriate input fields from the user and display the corresponding timetable. The user must provide the necessary inputs required to generate an acceptable timetable

2.3. User Classes and Characteristics

The timetable generator is built for the following types of users:

- Administration
  Each year, numerous students are enrolled in different departments of the same university. Managing them through paperwork is not only difficult but also time-consuming, so this application serves perfectly for them to easily create a timetable that is acceptable to everyone and is much less effort and much less error-prone.
- Faculty
  When the students are enrolled in the same courses but different sections, it is problematic for the faculty to keep track of when and where is the next class scheduled for them. This leads to some confusion among them. Therefore, this app eradicates this problem.
- Students
  A timetable with clashes is the most problematic for this user class as they are the ones who get penalized with absents and mental stress. Moreover, a timetable which is scattered throughout the week with long breaks in the middle of the day is not at all liked by the students as they prefer having continuous classes and the day to get over as soon as possible or the first class to start as late as possible. This system also targets this as a tertiary problem to solve.

2.4. Operating Environment

This is a web-based app that uses the Django web framework. The app is to be built on a web server which is a stable platform for these types of applications. This application requires a fully functional database, which is provided by SQLite 360. For a stable working one will need a browser that supports Graphical User Interfaces, and Python 3.8 or above installed on the computer system. From a hardware perspective any device i.e. a laptop, PC, mobile phone with a stable internet connection can access this system software.

2.5. Design and Implementation Constraints

The main problem faced in this project is the automated deployment problem. Once the project is up and running, it should be made sure that the application is working automatically and without any problems. No matter how complex the algorithms are used, in some cases, it is just not practically possible to execute all of the soft requirements. There is a limitation as to the number of classes you can have before their corresponding labs due to the fewer physical labs present where the labs have to be executed. Similarly minimized empty times in the middle of the day for each group is also another problem which can rarely be fully satisfied and at some point in very complex cases will never be completely satisfied simultaneously for all groups. Hence, these are practical constraints that will in some cases have to be compromised upon. Secondly, regular backups should be made while working with Python and Django. If not, there is always a risk of losing important data that might be costly.

2.6. User Documentation

Since the application is user-friendly, the initial release of the application does not require a user manual or a tutorial to tell the users how to use the timetable generator. However, once the user feedback is also taken into consideration, a help section might be introduced later. We do look forward to recording our semester project presentation and attaching it as part of the system as that will be more than enough for any user to guide on how to get things done through the system.

2.7. Assumptions and Dependencies

The only assumption that the software system makes is that all the labs, instructors, and groups have no prior commitments and are available for the entire working hours in the week. If any of these assumptions are not true e.g. a Lab is busy with some research workshop then the timetable generated will be flawed. Apart from this, there are no other assumptions and dependencies whatsoever.

3. Specific Requirements

3.1. External Specific Requirements

3.1.1. User Interfaces

The system consists of about two user interfaces. One is the input user interface and the other is the output user interface. In the input, user interface the system demands input from the user through a database. This interface will be connected to the backend database and will by itself obtain data from it. The next is the code that runs on the data provided as part of the input. This code generates a timetable which is essentially the output of the system. This timetable will be saved as part of a text file for further manipulation or use. All of this will be guarded by a login page so that only the relevant people who have access to it will be able to run it.

3.1.2. Hardware Interfaces

There are no hardware interfaces whatsoever for the system as the system is purely software-oriented in nature. The system can operate on any modern-day computing device with enough memory and processing connectivity to run a web page given that a stable internet connection is provided.

3.1.3. Software Interfaces

The software is being developed on the latest version of Windows 10 Pro but is compatible with all devices. The first software dependency the system has is of Python 3.8 and onwards. The language is used to code the solution which generates the timetable and is also needed to design the Webpages using Django and its frameworks with which the user communicates with the code. With Python, we recommend using Anaconda Version 2020.11 for bringing in any Python Libraries as part to execute the code. Apart from this, the software system uses the Sqlite 360 Version 3.36 as the source of its data input. For the webpages, one may use any browser but we preferably recommend Google Chrome.

3.1.4. Communication Interfaces

There is no communication used by the system whatsoever with any external interfaces. The software as part of itself obtains data from the backend database and stores the output in a text file, but throughout the process does not communicate or obtain any message from any end.

3.2. Functional Requirements

The following functional requirements are to be achieved by the system. They will be discussed in greater depth in the sections below:

- The timetable generated shall have no clashes whatsoever
- The timetable generated should have classes scheduled before their corresponding labs so the course outlines can be synchronized.

3.3. Clash Avoidance

3.3.1. Description and Priority

Clash Avoidance is the highest priority requirement for this system. It is one of the reasons the system is being created. The system should avoid Clashes in the following particular respects:

- One group cannot have two lectures at the same time.
- One Lecture hall cannot have two lectures at the same time.
- One Instructor cannot have two lectures at the same time.

3.3.2. Response Sequences

The behavior cannot be stimulated by any of the user responses as this is to be handled by the code generating the timetable. Such a situation can only take place while the execution of the code schedules a class such that there is a clash but the code shall have a mechanism to avoid these clashes or fix them without generating any errors.

3.3.3. Functional Requirements

Each class scheduled should be in such a manner that the following requirements are met:

- REQ 101: The Instructor taking the class shall not have another class at that time slot.
- REQ 102: The student group taking the class shall not have another class at that time slot.
- REQ 103: The Lecture Hall where the class is physically going to take place shall not have another class at that time slot.

The software must have a code section that schedules a class smartly so that all three of these clashes are avoided simultaneously. Just in case, the clash is generated it shall also have some form of algorithm or code to resolve the clash by rescheduling the class to another timeslot or lecture hall where there is no clash.

3.4. Classes before Labs

3.4.1. Description and Priority

Classes before Labs are a lower priority requirement for this system. It is one of the reasons we expect the system to facilitate the students. This requirement is placed so that the syllabi and learning outcomes are synchronized and maximum understanding of students.

3.4.2. Response Sequences

The behavior cannot be stimulated by any of the user responses as this is to be handled by the code generating the timetable. Such a situation can only take place while the execution of the code that it schedules a lab such that there is a class after it but the code shall have a mechanism to avoid these clashes or fix them without generating any errors.

3.4.3. Functional Requirements

Each class scheduled should be in such a manner that the following requirements are also ideally met:

- REQ 201: The Lab being scheduled shall be after the corresponding Class.

The software must have a code section that schedules a class smartly before the corresponding Lab. Just in case, the clash is generated it shall also have some form of algorithm or code to resolve the issue by rescheduling the class to another timeslot before the Lab. In case this is not possible the algorithm can leave the Class wherever it is scheduled as this is a tertiary requirement and in some cases, it might not even be possible to entertain this requirement due to the fewer number of labs physically available.

3.5. Consecutive Classes with minimal time waste

3.5.1. Description and Priority

Consecutive Classes are a lower priority requirement for this system. It is one of the reasons we expect the system to facilitate the students. This requirement is placed so that the students have minimal time to waste between the classes and maximize their productivity.

3.5.2. Response Sequences

The behavior cannot be stimulated by any of the user responses as this is to be handled by the code generating the timetable. Such a situation can only take place while the execution of the code schedules classes for each group such that there is a class after it.

3.5.3. Functional Requirements

Each class scheduled should be in such a manner that the following requirements are also ideally met:

- REQ 301: The Class/Lab being scheduled shall be immediately after any other Class/Lab.

The software must have a code section that schedules a class smartly after any other class. Just in case, a time gap is generated it shall also have some form of algorithm or code to resolve the issue by rescheduling the class to another timeslot favoring the students. In case this is not possible the algorithm can leave the Class wherever it is scheduled as this is a tertiary requirement and in some cases, it might not even be possible to entertain this requirement due to the unavailability of instructors or lecture halls.

4. Other Nonfunctional Requirements

4.1. Performance Requirements

Since the system is used to be run on a day to day machines, it would be advised to keep the performance optimal with optimal time and space complexities. It would be advised to not generate a codebase that takes multiple hours in execution as neither students nor the admin would want to run this program for days.

4.2. Safety Requirements

The system does not have any issues regarding safety, security, and privacy of data as no exchange of data of that personal level is involved. Since the product is to be used by educational institutions as a tool to help them through their administrative task there are no legislation or safety concerns to be taken care of in that regard.

4.3. Security Requirements

As a managerial tool, there are no security laws and issues that the system violates. However, to ensure that only the right people have access to it and only the relevant people from the administration have access to generate the timetable, it can be protected by a username and a password that authenticates and confirms the right user.

4.4. Software Quality Attributes

The software shall be 100% correct in the primary requirements placed on it, it shall be available, maintainable, portable, reliable, reusable, robust, easy to use, and tested until no clashes are seen.
The codebase i.e. the timetable generating algorithm should also be adaptable to any more constraints that any admin might want to customize depending on their personalized requirements.

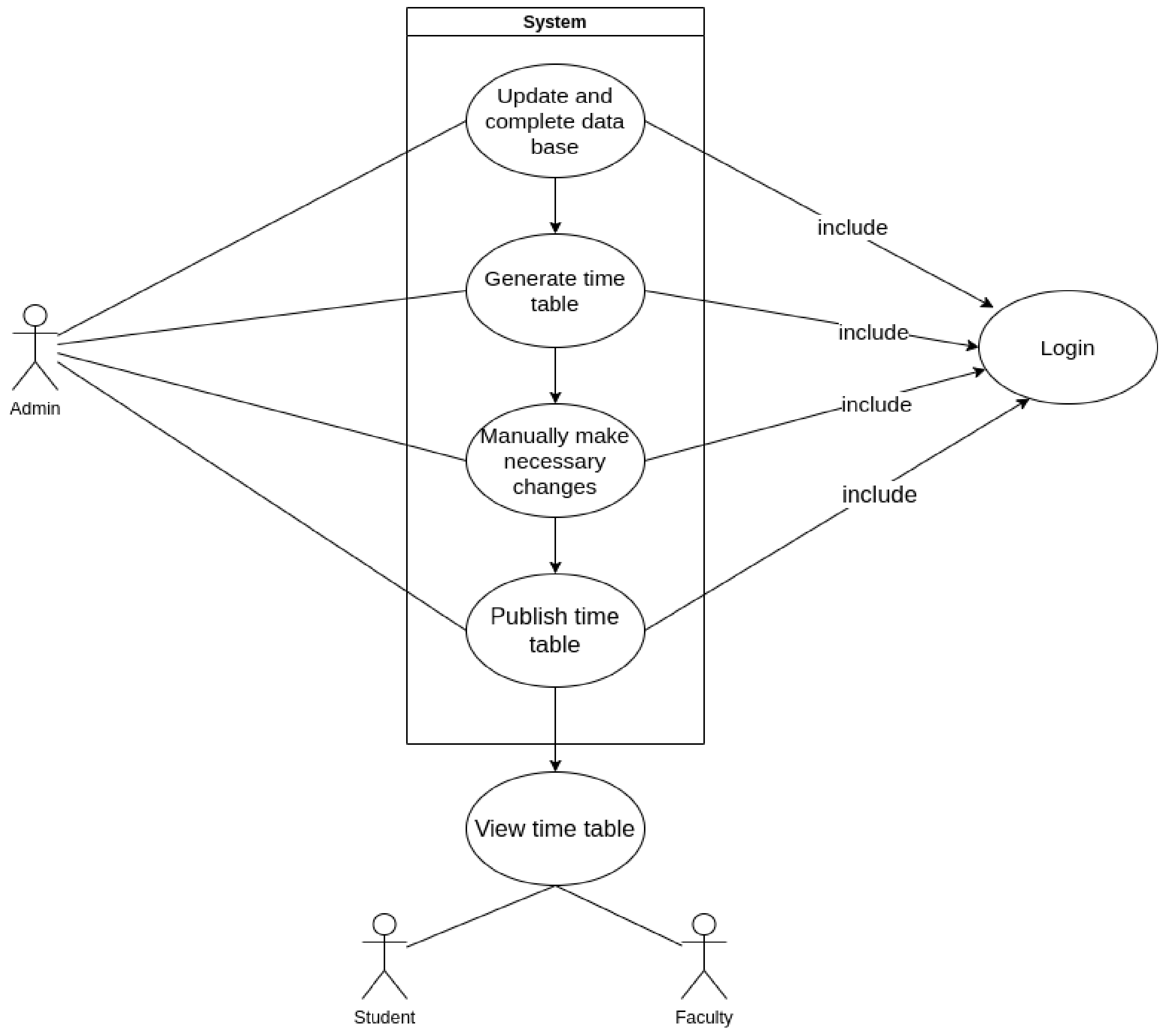
4.5. Business Rules

There is only one external constraint i.e. the timetable should only be generated once the back-end database has been completed and has no loopholes whatsoever.

5. Other Requirements

The system extracts data from a database built-in SQLite 360. The database the system uses shall have the appropriate tables designed in the schema and data dependencies are satisfied. A loophole in a database will eventually lead to a failure of the system itself. The system should also be reusable in its simplest terms. One should only update the data in the database and generate a new timetable and the system should be able to cater to that.

6. Use Cases


System
Update and complete data base
Generate time table
Manually make necessary changes
Publish time table
View time table
Login
include
include
include
include
Admin
Student
Faculty

7. Appendix A: Glossary

ATG: Automatic Timetable Generator is the name we have given to this system.
Instructor: Teacher/Faculty Member.
Lecture Hall: Classroom where the actual physical class takes place.
Group: A respective subsection of the students based on their faculty and batch no. e.g. Batch29-CS
Class: A unique combination of the instructor, Lecture Hall, Group, and timings.