SOFTWARE REQUIREMENTS SPECIFICATION

for

Automatic Timetable Generator

Version 1.0 approved

Hex Group

March 18, 2019

Contents

Revision History

Date	Author	Reason For Changes	Version
18-03-2019	Kumbong Hermann N	First Draft	1.0

1 Introduction

1.1 Purpose

The aim of this document is to delineate the software requirements specification for an automatic timetable generation software. The initial version of the software will be aimed at scheduling courses at the university level and will serve as a course completion requirement for the COE 356 Software engineering course at KNUST. Future versions(not described in this document) will generalize the scheduling problem to cater for scheduling in non academic environments like hospitals, jobs etc.

1.2 Document Conventions and Terminology

This document follows MLA Format. Bold-faced text has been used to emphasize section and sub-section headings. Highlighting is to point out words in the glossary and italicized text is used to label and recognize diagrams. In addition the following jargon is used throughout the text:

Term	Meaning
GUI	The graphical user interface is a form of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.

1.3 Intended Audience and Reading Suggestions

The primary audience for this document consists of all group members of Hex(project manager, developers, section heads and analysts) together with the project supervisor. The information provided could also be of benefit to product users and the prospective users on whose suggestions this document heavily relies. The document includes but is not limited to: an overall description of the the project, a discussion on the interface requirements, an analysis of system features and a description of other non functional requirements of the system. The document follows the sequence just described and is meant to be followed in that order. However, a busy reader may omit the last section on non functional requirements. The section on external interface requirements is directed primarily at the front end department while the overall description of the project will most suit the needs of a prospective client or user. It is the duty of the project manager to peruse this document and to enforce its usage and distribution of tasks described.

1.4 Project Scope

The current goal of this project is to develop a software for automatically scheduling courses in the university milieu. Currently most universities use a manual or a semi manual system for generating timetables. Most software available works for high schools and primary schools and is not easily adapted to the university setting. The process which can last for even weeks necessitates a lot of effort in resolving clashes and adjusting the timetable to meet specific institutional needs or those of lectures and students. Our objectives in designing our own system to solve this problem are as follows:

- Since at this moment we are unable to determine with certainty if the process can be completely automated, our goal is to automate the process of generating a timetable as much as possible providing the user of the system with little manual work to complete the timetable.
- To reduce the effort and time that are currently expended in constructing a timetable by atleast 70 %.
- To generate schedules that can
- To design a system that can be customized to meet the specific scheduling needs of any institution.
- To develop a simple and intuitive user oriented system that requires as little user training as possible.
- To provide an solution for maintaining, publishing and sharing timetables to users.

1.5 References

1. IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

2 Overall Description

2.1 Product Perspective

The Automatic Timetable Generation System is a closed source system comprising of 3 main components. The three components are as follows:

The Desktop platform is a stand alone program, which is a component of a larger timetable management system. The desktop component allows the user(administrator) to create and maintain a new timetable or to maintain a timetable already generated using the web component. This component is synchronized with the web app component which we describe next. The web app component provides a similar role to the desktop component. In addition to this it provides the means for managing user accounts. The mobile app component is meant for the end users of the timetable. (students, lecturers or other stakeholders). It provides them with a means of accessing their schedules and keeps them aware of any updates. It also gives them the opportunity to request any changes in their schedules from the administrator.

2.2 Product Functions

The timetable management system alongside its subsystems shall perform the following features:

- Provide the administrator with a means of creating a timetable template that satisfies the particular requirements(constraints) of his/her institution.
- Provide the administrator with a feature to automatically generate a timetable based on constraints they impose.
- Allow the administrator to apply manual changes to the timetable.
- **When applying manual changes to the timetable, provide the administrator with real time alerts and information when constraints are not met.** (This feature is prime in importance)
- Provide the administrator on hints or information on available resources or solutions to resolving clashes.
- End users (students and lecturers) should have a module to provide the administrator with information required to generate schedules for them.

- End users (students and lecturers) should have a module to request changes in their schedules from the administrator.
- Provide the administrator or other stakeholders with an easy interface to import their data or to enter new data required for timetable generation and management.
- End users should have a module that informs them of their timetable and any changes that have been made to it. (This could be through the mobile or the web app component)

2.3 User Classes and Characteristics

The users of the system can be broadly classified into two major categories. Users can fall into only one category only or in both categories. However a user can only have one type of account. In a a case where a user can be in anyone category, they have to create an account for each new category. The categories are as follows:

1. **Administrators**
   Administrators consist of anyone who are involved in the timetable generation process. More succinctly, administrators have the following privileges and roles:
   - Can initiate the process of creating a new timetable.
   - Has access to and can modify any timetable generated.
   - Publishes the timetable and is responsible for responding to user requests and performing updates on the timetable.
   - Fills the database with relevant information about resources such as (courses, lecturers, class size)
   - Creates the atomic timetable template (for example is the timetable from monday to saturday or just weekdays, the number of working hours and all related information.)
   - Decides on and implements scheduling and constraints and enforces that they are strictly followed.
   - Assigns priorities to different users, tasks, resources.
2. **End Users** End users are the consumers of the timetable schedule. For instance a lecturer could have a schedule, a student may have a schedule, a particular class group have a schedule and so on. (Due to elective courses a student's schedule is not necessarily the same as that of their class. The discrepancy increases in a liberal art system). The end users have the following roles and privileges with respect to the system:
   - Can view their personal schedules or those of any group they belong to.
   - Can request changes to their schedules or that of any group they belong to(it is left to the administrator to decide whether or not to implement such changes.)

2.4 Operating Environment

Our system is cross platform and does not depend on the hardware or architecture of the host system The web component is browser independent. In particular we support the particular operating systems.

1. **Mobile**

    a) Android OS

    b) ios

2. **Desktop**

    a) Windows 7 and above

    b) Mac

    c) Linux

    d) Solaris

2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

2.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are

already documented elsewhere (for example, in the vision and scope document or the project plan).>

3 External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4 System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1: REQ-2:

4.2 System Feature 2 (and so on)

5 Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

6 Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

6.1 Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

6.2 Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

6.3 Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>