Software Requirements Specification

Version 1.0

<<Annotated Version>>

June 16, 2020

KINDERGARTEN Automate System

Laila AL Hafez

Lana Ghali

Rama AL Hafez

Submitted in partial fulfillment

Of the requirements of

CS 310 Software Engineering

<<Any comments inside double brackets such as these are *not* part of this SRS but are comments upon this SRS example to help the reader understand the point being made.

Refer to the SRS Template for details on the purpose and rules for each section of this document.

This work is based upon the submissions of the Spring 2004 CS 310. The students who submitted these team projects were rama al hafez, Laila al hafez, and lana ghali. >>

# List of Figures

[Figure 1 - System Environment 4](#_Toc77487669)

[Figure 2 - Article Submission Process 6](#_Toc77487670)

[Figure 3 - Editor Use Cases 8](#_Toc77487671)

[Figure 4 - Logical Structure of the Article Manager Data 23](#_Toc77487672)

Table of contant

[List of Figures i](#_Toc43393388)

[1.0. Introduction 1](#_Toc43393389)

[1.1. Purpose 1](#_Toc43393390)

[1.2. Scope of Project 1](#_Toc43393391)

[1.3. Glossary 1](#_Toc43393392)

[1.4. References 2](#_Toc43393393)

[1.5. Overview of Document 2](#_Toc43393394)

[2.0. Overall Description 3](#_Toc43393395)

[2.1 System Environment 3](#_Toc43393396)

[2.2 Functional Requirements Specification 4](#_Toc43393397)

[2.2.1 Manger Use Case 4](#_Toc43393398)

[2.2.2 Accountant Use Case 5](#_Toc43393399)

[2.2.3 Employee Use Case 6](#_Toc43393400)

[2.3 User Characteristics 9](#_Toc43393401)

[2.4 Non-Functional Requirements 9](#_Toc43393402)

[3.0. Requirements Specification 12](#_Toc43393403)

[3.1 External Interface Requirements 12](#_Toc43393404)

[3.2 Functional Requirements 12](#_Toc43393405)

[3.2.1 Entering teacher data 12](#_Toc43393406)

[3.2.2 Registration of premiums 12](#_Toc43393407)

[3.2.3 Student registration 13](#_Toc43393408)

[3.2.4 Student registration information 13](#_Toc43393409)

[3.2.5 Print the final calendar 14](#_Toc43393410)

[3.2.7 : Send Weekly Report 15](#_Toc43393411)

[3.3 Detailed Non-Functional Requirements 15](#_Toc43393412)

[3.3.1 Logical Structure of the Data 15](#_Toc43393413)

[3.3.2 Security 17](#_Toc43393414)

# 

# 1.0. Introduction

## 1.1. Purpose

The purpose of this document is to present a detailed description of the

Kindergarten Automate System. 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.

## 1.2. Scope of Project

This software system will be a Kindergarten Automate System for a local employee .This system will be designed to maximize the employee’s productivity by providing tools to assist in automating the internal system of kindergarten, which would otherwise have to be performed manually. By maximizing the employee’s work efficiency and production the system will meet the employee’s needs while remaining easy to understand and use.

More specifically, this system is designed to allow an employee to manage all about

student and teacher affairs completely and quickly on kindergarten automate system. The

software will facilitate communication between employees and teachers and parents.

This system will Enter and handle teacher data, enter new student data, and enter quick

access data for parents. The system is linked to an application with the parents.

It registers premiums and prints an invoice. This application needs a database

management system to store student data.

## 1.3. Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Active Article | The document that is tracked by the system; it is a narrative that is planned to be posted to the public website. |
| Employee | Person who is working in a company. |
| Database | Collection of all the information monitored by this system. |
| Teacher | Person who teach student in school. |
| Field | A cell within a form. |
| Application | Software will work on mobile. |
| Invoice | Paper containing how much money you paid. |
| Premiums | Dividing a sum of money into several small amounts. |
| Software Requirements Specification | A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document. |
| Logo | It's a symbol that indicates something. |
| kindergarten | Place where children under six years come to learn and enjoy. |

## 1.4. References

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

## 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.

# 2.0. Overall Description

## 2.1 System Environment

manger

accountant

parents

DB

system

application

Kindergarten automate System

employee

Figure - System Environment

The kindergarten automates System has four active actors and one cooperating system.

The Manger, Accountant, Employee accesses the System through the local network directly. Parents accesses the System through the internet.

<< The division of the Kindergarten Automate System into two component parts, the Application and the System is an example of using domain classes to make an explanation clearer. >>

## 2.2 Functional Requirements Specification

This section outlines the use cases for each of the employee, manger and accountant separately. All of them have only one use case apiece and they are main actor in this system.

### 2.2.1 Manger Use Case

#### Use case: Entering teacher data

**Diagram:**

Manger

Entering teacher data

**Brief Description**

The manger accesses the system and enter teacher’s data.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Manger has already accessed the local network.

1. Teacher comes to the kindergarten.
2. Teacher give his information to the manger.
3. The Manger enter the teacher information to the system.
4. The system will store the information in database.

**Xref:** Section 3.2.1, Entering teacher data

system

Enter information

storing

Figure – Entering teachers data Process

The *Entering teacher’s data Process* state-transition diagram summarizes the use cases listed below. A Teacher give his information for consideration. The Manger enters it into the system and assigns him. The system returns his information to data base. the manger will decide which class will teacher give

### 2.2.2 Accountant Use Case

Use case: **Registration of premiums**

**Diagram:**

Accountant

Registration of premiums

**Brief Description**

The accountant records the student installments and prints the invoice in the event of payment.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Accountant has already connected to the local network and verify the student's presence in the database.

1. The employee records the student in database.
2. The parent come to the accountant and pay money.
3. The accountant records the premium in the system.
4. The accountant prints the invoice for parent.

**Xref:** Section 3.2.2, Registration of premiums

### 2.2.3 Employee Use Case

The Employee has the following sets of use cases:

Student registration

employee

Student registration information

Print the final calendar

Send notice

Send weekly report

Figure - Employee Use Cases

**Update Information use cases**

#### Use case: Student registration

**Diagram:**

Employee

Student registration

**Brief Description**

The employee record the student in the system.

**Initial Step-By-Step Description**

Before this use case can be initiated, the employee has already connected to the local network.

1. The parent comes to the kindergarten.
2. The employee searches within the database to find out if the student already exists or no.
3. If the student exists in database, the employee class it into a pre-existing student.
4. Else the employee class the student into a new student.

**Xref:** Section 3.2.3, Student registration.

**Use case:** Student registration information

**Diagram:**

Employee

Student registration information

**Brief Description**

The Employee records student information in database and dealing with it from collecting, modifying and others.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Employee has already accessed the local network.

1. The Employee records the student's absence and attendance.
2. The Employee records the student's marks and places them in the final calendar.
3. The Employee records a weekly report to the family on the student's condition.

**Xref:** Section 3.2.4 Student registration information .

#### Use case: Print the final calendar

**Diagram:**

Employee

Print the final calendar

**Brief Description**

The Employee print the final calendar.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Employee has already accessed the local network and has already recorded the student’s marks .

1. The Employee accesses the database.
2. The employee brings student marks.
3. The employee places the marks in the final calendar.
4. The employee prints the final calendar.

**Xref:** Section 3.2.5, Print the final calendar

#### Use case: Send Notice

Employee

Send notice

**Diagram:**

**Brief Description**

The Employee sends a notice to the parents.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Employee has already accessed the local network. The parent has already the application on there phone.

1. The Employee accesses the database.
2. The employee records the student’s absence.
3. The employee sends a notice to the parent.

**Xref:** Section 3.2.6 Send Notice

#### Use case: Send Weekly Report

**Diagram:**

Employee

Send weekly report

**Brief Description**

The Employee send a weekly report to the parents.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Employee has already accessed the local network and access to the database. The parent has already the application on there phone.

1. The Employee accesses the system.
2. The system fetches the student record from the database.
3. The employee writes the report about the student's condition.
4. The employee sends the report to the parent.
5. The process is repeated every week.

**Xref:** Section 3.2.7, Send Weekly Report

## 2.3 User Characteristics

The employee is expected to familiar with dealing with Windows 7 and above

operating systems and has knowledge of dealing with the database.

The accountant is expected to familiar with dealing with Windows 7 and above

operating systems, must be able to count money and be able to handle the printer.

The manger is expected to familiar with dealing with Windows 7 and above

operating systems

The detailed look of these pages is discussed in section 3.2 below.

## 

## 2.4 Non-Functional Requirements

The kindergarten automate system will be on a local server with response speed not exceeding 20 milliseconds capability. The system work in windows 7 and above.

There must be a database management system. The system should be easy to use.

The presence of an application associated with the system. The presence of a color

printer.

# 3.0. Requirements Specification

## 3.1 External Interface Requirements

The only link to an external system is the link to Database (it use to save all information that entered), application(it use to let parent know everything about their kids do and send report about they degree) and camera system (it get camera data at but them in the database).

## 3.2 Functional Requirements

The Logical Structure of the Data is contained in Section 3.3.1.

### 3.2.1 Entering teacher data

|  |  |
| --- | --- |
| **Use Case Name** | entering teacher data |
| **XRef** | Section 2.2.1, entering teacher data  SDD, Section 7.1 |
| **Trigger** | The teacher comes to the kindergarten to work in it |
| **Precondition** | Kindergarten accepted the teacher |
| **Basic Path** | 1. Teacher comes to the kindergarten. 2. If the kindergarten accepted the teacher , the teacher give his information to the manger. 3. The manger adds teacher information to the system. 4. The System will upload teacher information to database. 5. The System will show the class which does not have teacher. 6. The Manger chose the right class for the teacher. 7. The system updates the change to the system |
| **Alternative Paths** | In step 2, if the Kindergarten does not accept teacher, the teacher will go home |
| **Postcondition** | The teacher will be teaching the student. |
| **Exception Paths** | Error with connect to database. |
| **Other** | None |

### 3.2.2 Registration of premiums

|  |  |
| --- | --- |
| **Use Case Name** | Registration of premiums |
| **XRef** | Section 2.2.2, Registration of premiums; |
| **Trigger** | The parents come to kindergarten. |
| **Precondition** | The parents must registries their kids |
| **Basic Path** | 1. The parent come to the accountant and pay money. 2. If student registries in the kindergarten 3. The parent give money to the accountant. 4. The accountant records the premium in the system. 5. The system uploads the premium in the database. 6. The accountant prints the invoice for parent. |
| **Alternative Paths** | 1. If step 2 is unrealized the parents must go to employ   to registries their kids then continues step 3 |
| **Postcondition** | The accountant give invoice for parents |
| **Exception Paths** | 1. The parents don’t have enough money. 2. The printer doesn’t contain ink 3. Power cut 4. There is no paper |
| **Other** | None |

### 3.2.3 Student registration

|  |  |
| --- | --- |
| **Use Case Name** | Student registration |
| **XRef** | Section 2.2.3, student registration Section 7.3 |
| **Trigger** | The parent comes to kindergarten. |
| **Precondition** | Check if their space for the student |
| **Basic Path** | 1. The parent come to kindergarten.   1. They give their son information to the employee. 2. The system uploads the data to database 3. The system but the student in the class and tell the employee where it but he. |
| **Alternative Paths** | If in step 2, there aren’t space in the kindergarten the employee will tell the parent. |
| **Postcondition** | The student will be in the kindergarten |
| **Exception Paths** | There is no space in the kindergarten |
| **Other** | none |

### 3.2.4 Student registration information

|  |  |
| --- | --- |
| **Use Case Name** | Student registration information |
| **XRef** | Section 2.2.3 |
| **Trigger** | The student comes to kindergarten. |
| **Precondition** | The student should be registrant. |
| **Basic Path** | 1. The student comes to kindergarten. 2. The Teacher records the student's absence and attendance. 3. The Teacher records the student's marks and places them in the final calendar. 4. The Teacher records a weekly report to the family on the student's condition. 5. The teacher takes information about the student and give them to the employee. 6. The Employee write the information in the system. 7. The System upload them in the database. |
| **Alternative Paths** | None |
| **Postcondition** | The Employee can print the final calendar |
| **Exception Paths** | The teacher delayed the delivery of information |
| **Other** | None |

### 3.2.5 Print the final calendar

|  |  |
| --- | --- |
| **Use Case Name** | Print the final calendar |
| **XRef** | Sec 2.2.3 Print the final calendar Section 7.5 |
| **Trigger** | The semester ends |
| **Precondition** | The Student should be registrant. |
| **Basic Path** | 1. The Employee ask from system to print calendar. 2. The system gets from database the student information . 3. The employee places the marks in the final calendar. 4. The System show to employee how calendar look like. 5. The Employee give the system accept. 6. The System print calendar. |
| **Alternative Paths** | None |
| **Postcondition** | The Employee give the calendar to the student |
| **Exception Paths** | 1. The printer doesn’t contain ink 2. Power cut 3. There is no paper 4. The parent don’t pay all money |
| **Other** | None. |

#### 3.2.6 Send Notice

|  |  |
| --- | --- |
| **Use Case Name** | Update Article Status |
| **XRef** | Section 2.2.3, Send Notice, Section 7.6 |
| **Trigger** | The student doesn’t come to kindergarten |
| **Precondition** | None |
| **Basic Path** | 1. The student doesn’t come to kindergarten . 2. The Employee go to student information. 3. The system gets information and show them. 4. The employee records the student’s absence. 5. The employee sends a notice to the parent. |
| **Alternative Paths** | None |
| **Postcondition** | The note has been sent |
| **Exception Paths** | The parent change their number. |
| **Other** | None |

### 3.2.7 : Send Weekly Report

|  |  |
| --- | --- |
| **Use Case Name** | Send Weekly Report |
| **XRef** | Section 2.2.3, Send Weekly Report; Section 7.7 |
| **Trigger** | A week has passes. |
| **Precondition** | The student went to kindergarten for a week |
| **Basic Path** | 1. The student went to kindergarten for a week 2. The Employee accesses the system. 3. The system fetches the student record from the database. 4. The employee writes the report about the student's condition. 5. The employee sends the report to the parent. |
| **Alternative Paths** | None |
| **Postcondition** | The report will be sent. |
| **Exception Paths** | The parents don’t have the application. |
| **Other** | None |

## 3.3 Detailed Non-Functional Requirements

### 3.3.1 Logical Structure of the Data

The logical structure of sending report.

application

report

employee

writes

sent to

Figure 4 - Logical Structure of sending report

The data descriptions of each of these data entities is as follows:

**student Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of student |  |
| Place of birth | Text | Where student born |  |
| Birth date | date | When you born |  |
| Registration number | number | none |  |
| Anchoring of registration | number | How much parent should pay |  |
| Mother number | number | Phone number |  |

**parents Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of parents |  |
| ID | Integer | ID number student | It should be unique |
| Email Address | Text | Internet address |  |
| Work parent | text | What parent work | May be several |
| Num phone | Integer | Num of phone parent |  |

**Employee Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| name | Text | Name of employee |  |
| Father name | Text | Name of father |  |
| Birth date | Date | Date of borne |  |
| Num phone | Number | The number of phones |  |
| Degree | Text | Obtaining certificates |  |

**Attendance of students Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of student |  |
| Date | Date | Date |  |
| Present or absent | Text | If student her or not |  |

**report Article Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of student |  |
| Degree | Text | How many he took in exam | May be several |
| Activates | Text | Activity of student |  |

### 3.3.2 Security

The server on which the resides will have its own security to prevent unauthorized *write*/*delete* access. There is no restriction on *read* access.

The PC on which the Manager resides will have its own security. Only the manger ,employee and accountant will have physical access to the machine and the program on it.