Software Requirements Specification

For

Online Parent Teacher Association

Prepared by:

Abhishek Kumar (M140363CA)

Jaigish (M140407CA)

Vivek P. (M140413CA)

National Institute Of Technology, Calicut

Submitted On: January 21, 2016

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the Online Parent Teacher Association. 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.

1.2 Product Scope

This system will be used by teachers, parents and students to be in contact with each other. This system will be intended to increase communication between the parents and the teachers. Increased interactions between teachers and parents will be helpful to enhance ward's capabilities. It is not intended but this system can also be used as messenger in closed user group, as it will have messaging feature inbuilt. This system will also help parents to stay active for their wards, as they will be able to track when their wards will leave for school or college and won't reach there by time.

Initially this system will be created for single institutions, and can be deployed as separate domains or as a sub domain of existing website of institutions. Further this system will also be able to enhance to connect multiple colleges in a circle, like city, state, country or global level.

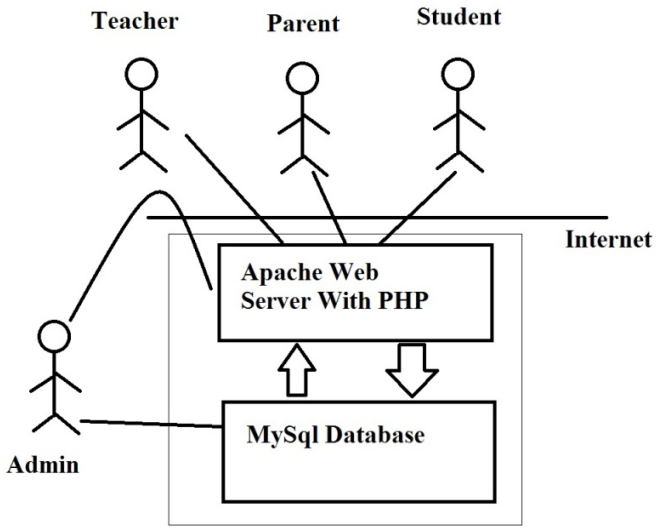
1.3 Intended Audience and Reading Suggestions

This Software Requirements Specification is intended to developers. Hence this SRS is written prior to implementation, therefore it can’t be considered as complete. While implementing this SRS, some functionality may seem not to be really necessary, and some functionality may seem to be missing. Final application based on this SRS may slightly deviate from the concept implemented in this SRS. While reading this SRS, it will be helpful to read in sequence. Because, there are many sections which can’t be understood without knowing the previous sections.

2. Overall Description

2.1 Product Perspective

The online Parent Teacher Association has four active actors and one cooperating system. The Parent, Teacher or Student can access the system through the internet only. The Admin can access the system directly as well as via internet.



2.2 Product Functions

2.2.1 User: Admin

2.2.1.1 Use Case: Login

Admin logs in the system using registered login id and password.

2.2.1.2 Use Case: Register user

The admin registers the user by filling the details of requested user information.

2.2.1.3 Use Case: Delete User

The admin marks a user a user as inactive which is now not active.

2.2.1.4 Use Case: Restore User

The admin restores the user details that is accidently archived.

2.2.1.5 Use Case: Send registration mail

The admin sends mail to a user on creation or deletion of account.

2.2.1.6 Use Case: View users

The admin view the list of all users.

2.2.1.7 Use Case: View requests

The admin view the requests submitted for registration.

2.2.1.8 Use Case: Approve request

The admin approve requests from the request list.

2.2.1.9 Use case: Log out

Admin logs out of the system.

2.2.2 User: Teacher

2.2.2.1 Use Case: Login

Teacher needs to login using registered id and password.

2.2.2.2 Use Case: Chose course

Teacher chooses course from the list.

2.2.2.3 Use Case: See enrolled students

Teacher sees the list of students enrolled in the course.

2.2.2.4 Use Case: Enter grade

Teacher enters grades for each student in the enrolled course.

2.2.2.5 Use case: Enter attendance

Teacher enters attendance for each student in the chosen course.

2.2.2.6 Use case: See message

Teacher sees the messages in his inbox.

2.2.2.7 Use case: Send message

Teacher sends message to student or parent.

2.2.2.8 Use case: Upload class notes

Teacher uploads class notes to system.

2.2.2.9 Use case: Update home work

Teacher updates homework to system.

2.2.2.10 Use case: See the discussion forum

Teacher sees the discussion forum.

2.2.2.11 Use case: Post in discussion forum

Teacher posts his message in discussion in the discussion forum.

2.2.2.12 Use case: Log out

Teacher logs out of the system.

2.2.3 User: Parent

2.2.3.1 Use case: Login

Parent logs in the system using the registered user id and password.

2.2.3.2 Use Case: See attendance

Parent sees the attendance of his/her ward.

2.2.3.3 Use Case: See message

Parent sees the message in his inbox.

2.2.3.4 Use case: Send message

Parent sends message to a teacher or his ward.

2.2.3.5 Use case: See progress report

Parent sees the progress report of the student.

2.2.3.6 Use case: See the discussion forum

Parent sees the messages in discussion forum.

2.2.3.7 Use case: Post in discussion forum

Parent posts a message in the discussion forum.

2.2.3.8 Use Case: Log Out

Parent logs out of the system.

2.2.4 User: Student

2.2.4.1 Use case: Login

Student logs in the system using the registered user id and password.

2.2.4.2 Use case: See the class notes

Student sees the class notes uploaded in the system

2.2.4.3 Use case: See the home work

Student sees the home work uploaded in the system.

2.2.4.4 Use Case: See attendance

Student sees his attendance in the system.

2.2.4.5 Use Case: See Message

Student sends the message to the teachers or other students.

2.2.4.6 Use case: Send message

Student sends message to a teacher or his parent.

2.2.4.7 Use case: See progress report

Student sees his progress report.

2.2.4.8 Use case: Log out

Students logs out of the system.

3. Functional Requirements

Use Case Name	Login
Trigger	Any user of the system logs in the system
Precondition	User has opened the login page.
Basic Path	1. User inputs the user id and password in the fields. 2. System searches stored password for the given id. 3. Since password is encrypted, system encrypts the password user entered. 4. System matches the both encrypted passwords. 5. If the passwords match, system creates the cookies for user, and redirects to the home page.
Alternative Paths	If the provided password is incorrect, system loopback to login page.
Post-condition	User logs in the system.
Exception Paths	None
Other	None

Use Case Name	Log out
Trigger	Any user of the system clicks **log out** link
Precondition	User is logged in.
Basic Path	1. System deletes all the cookies intended for the particular user. 2. System pushes login page
Alternative Paths	None
Post-condition	User is logged of the system.
Exception Paths	None
Other	None

Use Case Name	Register User
Trigger	Admin registers a user manually.
Precondition	1. Admin is logged in the system.
Basic Path	1. User enters the details of the user. 2. System enters details of the users in the database. 3. System pushes an html page with user id of registered user.
Alternative Paths	If new user is already registered, system pushes an error message.
Post-condition	New user registered.
Exception Paths	If some mandatory field is missing, system loop back to same page again.
Other	None

Use Case Name	Delete User
Trigger	Admin deletes a user.
Precondition	1. Admin is logged in the system.
Basic Path	1. User enters the id of the user. 2. System marks active field of the user as false in the database. 3. System pushes an html page with success message.
Alternative Paths	If user doesn't exist or already deleted, system pushes an error message.
Post-condition	User deleted.
Exception Paths	None
Other	None

Use Case Name	Restore User
Trigger	Admin restores a user.
Precondition	1. Admin is logged in the system.
Basic Path	1. User enters the id of the user. 2. System marks active field of the user as true in the database. 3. System pushes an html page with success message.
Alternative Paths	If user doesn't exist or already active, system pushes an error message.
Post-condition	User restored.
Exception Paths	None
Other	None

Use Case Name	View requests.
Trigger	Admin clicks on view users link.
Precondition	1. Admin is logged in the system.
Basic Path	1. System retrieves all the tuples from the requests table. 2. System pushes an html page with details of each request. 3. There is a checkbox corresponding to each request.
Alternative Paths	None.
Post-condition	Request displayed
Exception Paths	None
Other	None

Use Case Name	Approve requests.
Trigger	Admin clicks on approve users button.
Precondition	1. Admin is logged in the system.
Basic Path	1. System retrieves all the tuples from the requests table. 2. System pushes an html page with details of each request. 3. There is a checkbox corresponding to each request. 4. User checks all the check buttons corresponding to each request which is to be approved. 5. User presses the submit button. 6. System copies the selected details from requests table to user table. 7. System removes the copied tuples from the requests table. 8. System pushes all generated ids on new html page.
Alternative Paths	If some users in approval list are already registered, their tuples will be deleted but not copied.
Post-condition	Request Approved
Exception Paths	None
Other	None

Use Case Name	Send Mail
Trigger	Admin Send mail to approved user.
Precondition	1. Admin is logged in the system.
Basic Path	1. Each time a user is registered, system generates a user id and password for new user. 2. System provides a button to send mail to user. 3. Admin clicks the send mail button. 4. System retrieves user id, mail id and initial password. 5. Then the system sends the mail to the user with user id and password.
Alternative Paths	None
Post-condition	Mail Sent
Exception Paths	None
Other	None

Use Case Name	Choose course
Trigger	Teacher clicks the choose course button.
Precondition	1. Teacher is logged in the system.
Basic Path	1. Teacher clicks on the choose course link. 2. System retrieves the courses for the teacher registered in the current semester, and pushes on an html page. 3. Teacher clicks the desired course. 4. System pushes various options on an html page.
Alternative Paths	If teacher is not registered for any course in the current semester, system will generate a customized error page.
Post-condition	Course page is open.
Exception Paths	None
Other	Options on the course page are grading, list of enrolled students, attendance, class notes, home work and syllabus.

Use Case Name	See enrolled students
Trigger	Teacher clicks the see enrolled students link.
Precondition	1. Teacher is logged in the system. 2. Course page is already open.
Basic Path	1. System retrieves list of all students from the database. 2. System pushes the retrieved data on an html page.
Alternative Paths	If no student is registered for any course in the current semester, system will generate a customized error page.
Post-condition	Enrolled list is displayed.
Exception Paths	None
Other	None

Use Case Name	Enter grade
Trigger	Teacher clicks the enter grades link.
Precondition	1. Teacher is logged in the system. 2. Course page is already open.
Basic Path	1. System retrieves list of all students from the database. 2. System pushes the retrieved data on an html page with text input field in front of each student to enter grade. 3. System pushes an extra field to select exam name. 4. Teacher inputs each field and then submits. 5. System inputs each student grade in the database.
Alternative Paths	none.
Post-condition	Grading is uploaded
Exception Paths	None
Other	None

Use Case Name	Enter Attendance
Trigger	Teacher clicks the attendance link.
Precondition	1. Teacher is logged in the system. 2. Course page is already open.
Basic Path	1. System retrieves list of all students from the database. 2. System pushes the retrieved data on an html page with checkbox field in front of each student to check attendance. 3. Teacher checks checkboxes of present students and then submits. 4. System inputs attendance with date and course of each student in the database.
Alternative Paths	none.
Post-condition	Attendance is uploaded
Exception Paths	None
Other	None

Use Case Name	Upload class notes
Trigger	Teacher clicks the class notes link.
Precondition	1. Teacher is logged in the system. 2. Course page is already open.
Basic Path	1. System pushes a page with previous notes of selected subject with a field to upload new file. 2. Teacher selects a file and submits. 3. System adds the file in file server and stores the name in database.
Alternative Paths	none.
Post-condition	Class note is uploaded.
Exception Paths	None
Other	None

Use Case Name	Upload Home Work
Trigger	Teacher clicks the Home work link.
Precondition	1. Teacher is logged in the system. 2. Course page is already open.
Basic Path	1. System pushes a page with previous home works of selected subject with a field to upload new home work. 2. Teacher enters new home work submits. 3. System adds the homework in database.
Alternative Paths	none.
Post-condition	Home work is uploaded.
Exception Paths	None
Other	None

Use Case Name	See attendance
Trigger	A parent or a student clicks the see attendance link
Precondition	1. Parent or student is logged in the system.
Basic Path	1. System retrieves the attendance details for the student and pushes on a html page
Alternative Paths	none.
Post-condition	Attendance is displayed.
Exception Paths	None
Other	None

Use Case Name	See progress report
Trigger	A parent or a student clicks the see progress report link
Precondition	1. Parent or student is logged in the system.
Basic Path	1. System retrieves the progress report details for the student and pushes on a html page
Alternative Paths	none.
Post-condition	Progress report details is displayed.
Exception Paths	None
Other	None

Use Case Name	See class notes
Trigger	A parent or a student clicks the see class notes link
Precondition	1. Parent or student is logged in the system.
Basic Path	1. System retrieves the class notes of all registered courses for the student and pushes on an html page. 2. System provide link to download each file.
Alternative Paths	none.
Post-condition	Class notes are displayed.
Exception Paths	None
Other	None

Use Case Name	See Home work
Trigger	A parent or a student clicks the see attendance link
Precondition	1. Parent or student is logged in the system.
Basic Path	1. System retrieves the home work of each registered course for the student and pushes on a html page
Alternative Paths	none.
Post-condition	Home work is displayed.
Exception Paths	None
Other	None

Use Case Name	See messages
Trigger	A parent, teacher or a student clicks the messages link.
Precondition	1. Parent, teacher or student is logged in the system.
Basic Path	1. System retrieves the messages related to the user and pushes on a html page
Alternative Paths	none.
Post-condition	Messages are displayed.
Exception Paths	None
Other	None

Use Case Name	Send message
Trigger	A parent, teacher or a student clicks the send message link or select a message to reply.
Precondition	1. Parent, teacher or student is logged in the system. 2. Messages page is already open.
Basic Path	1. User selects a message 2. System retrieves all the messages with user and pushes on an html page. 3. System provides a text box to enter new message. 4. User enter message and submits. 5. System saves message in database and refreshes the page.
Alternative Paths	none.
Post-condition	Message is sent
Exception Paths	None
Other	None

Use Case Name	Send and view message in discussion forum
Trigger	A parent or a teacher clicks the discussion forum link.
Precondition	1. Parent or teacher is logged in the system.
Basic Path	1. System retrieves all the messages in discussion form table and pushes on an html page. 2. System provides a text box to enter new message. 3. User enter message and submits. 4. System saves message in database and refreshes the page.
Alternative Paths	none.
Post-condition	Message is sent
Exception Paths	None
Other	None

4. Non-Functional Requirements

4.1 Performance Requirements

- Flexible service based architecture will be highly desirable for future extension.
- Some new functionality can be added during software modification.
- Server should be versatile.
- Server should be capable of handling enough requests.
- Server should be online 24x7.

4.2 Security Requirements

- Only software admin with valid user id and password can modify the entire coding and structure of the system.
- All the data on the server should be secured.
- Server should be resistant of security threats.

5. Hardware and Software requirements

5.1 Hardware requirements:

Any machine which is capable of running Apache-mysql-php server is suitable to run this system.

1. **Processor:** Intel Dual Core, (2 GHz)
2. **Ram:** 2 GB recommended, minimum 1 GB.
3. **Storage Capacity:** 10 GB minimum free space.
4. **Connectivity:** Ethernet

5.2 Software requirement:

Recommended OS: Linux Ubuntu, Windows XP, Windows 7 and later versions.

To implement this system a server with Apache installed in it with php5 and mysql is required on server side.

On the client side, any browser capable to run JavaScript program with HTML4.0 minimum is required to access this system.

Recommended Browsers: Mozilla Firefox, Google Chrome, Opera.