Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur

Students Kit

Objective

These guidelines are for the student to adopt to make progress in the project.
Given below are the templates for the documents related to the project. These are just guidelines only. The team can improve these.

Requirements Specification (RS)

Following is a template for the RS document. Some example requirements are entered in to it to show how to use the template. Make sure that you enter even the smallest/most trivial requirements also. That would help in validating the system during testing.

No.	**Requirement**	**Essential/ Desirable**	**Description of the Requirement**	**Remarks**
RS1	The system should have a login	Essential	A Welcome Page should appear when the URL is invoked. The welcome page should have a login.	This will ensure that only authorized users can access the school administration system.
RS2	The system should provide help screens	Essential	Help screens should provide details about system features in a Q&A format.	This should include guidelines for system usage and common questions related to student and teacher management.
RS3	The system should 'lock' the	Desirable	This feature will	Since the system

	login ID after 3 failed attempts		increase security by locking out users after three consecutive wrong password attempts.	will primarily be used by school staff, this feature is desirable for additional security but not critical.
RS4	The system should support client-side validation	Essential	Form validations such as checking mandatory fields and data format should occur on the client side.	This will reduce unnecessary server requests and improve user experience by providing immediate feedback on form input errors.
RS5	The system should allow student registration	Essential	The system should have a feature to register new students, including input for personal and academic details.	This is the core function of the school admin system, enabling the school to maintain a detailed database of all students.
RS6	The system should allow teacher assignment to classes	Essential	There should be a feature to assign teachers to specific classes based on subjects taught.	This will allow the school administration to manage and organize classes effectively.
RS7	The system should generate student progress reports	Desirable	A feature that generates progress reports for students based on grades and attendance.	This feature will make it easier for teachers to evaluate students' academic performance but can be

				implemented after basic system functionality is completed.
RS8	The system should support multi-user access control	Essential	Multiple roles (admin, teacher, parent, student) should have different access levels to the system.	This will ensure that only authorized users have access to the parts of the system relevant to their role, improving both security and functionality.
RS9	The system should provide attendance management	Essential	A module for recording and tracking student attendance should be included.	This will help the school manage attendance records efficiently, with daily input and reporting features.
RS10	The system should have a fee management feature	Desirable	The system should allow the administration to manage and track student fee payments.	This feature will simplify the financial tracking process, but it can be considered for future versions if time is limited in the initial development phase.

Database Fields Specification

Student Id is the key of the students database. Teachers Id is the key of the Teachers database. When a new student registers with the portal, he will get a student Id. His activities will be tracked in the student database. The quantity of each student (like dob, class, address etc) will be stored in the products database.

Students Database

No.	Field Name	Range of Valid Values for the Field	Remarks
1	Student Id	1 to 10000	This is the unique key field for identifying a student.
2	First Name	Up to 30 characters in length	First name of the student.
3	Last Name	Up to 30 characters in length	Last name of the student.
4	Date of Birth	Date format (YYYY-MM-DD)	Date of birth of the student.
5	Class	Numeric (1-12)	The class/grade the student is enrolled in.
6	Address	Up to 100 characters in length	The address of the student.
7	Contact Number	Numeric (10-15 digits)	Parent/guardian's contact number.
8	Enrollment Date	Date format (YYYY-MM-DD)	The date the student was enrolled in the school.

Teachers Database

No.	Field Name	Range of Valid Values for the Field	Remarks
1	Teacher Id	1 to 10000	This is the unique key field for identifying a student.
2	First Name	Up to 30 characters in length	First name of the student.
3	Last Name	Up to 30 characters in length	Last name of the student.

4	Subject	Up to 50 characters in length	The subject(s) that the teacher teaches.
5	Contact Number	Numeric (10-15 digits)	Contact number of the teacher.
6	Hire Date	Date format (YYYY-MM-DD)	The date the teacher was hired.

Classes Database

No.	Field Name	Range of Valid Values for the Field	Remarks
1	Class Id	1 to 12	This is the unique key field for identifying a class.
2	Class Name	Up to 50 characters in length	Name of the class (e.g., 5th Grade, 6th Grade).
3	Teacher Id	1 to 1000	The unique identifier of the teacher assigned to this class.
4	Room Number	Numeric (1-100)	The room number where the class is conducted.
5	Schedule	Up to 50 characters in length	The schedule or timing for the class

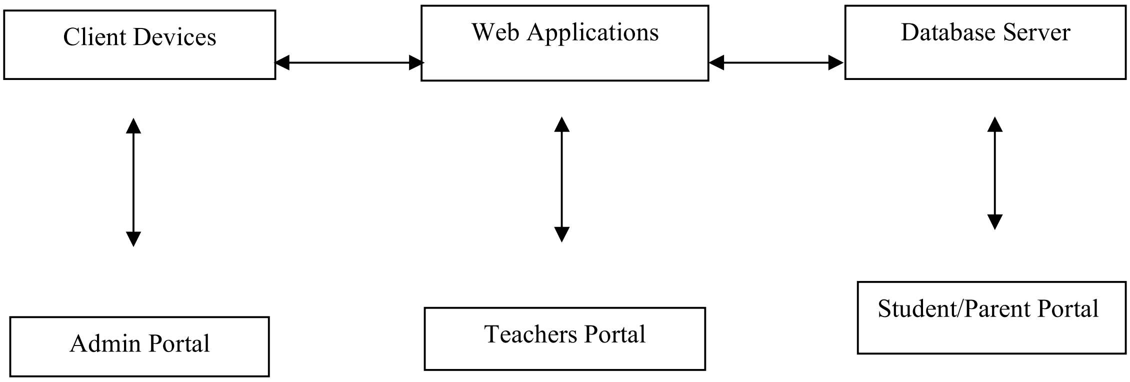
Attendance Database

No.	Field Name	Range of Valid Values for the Field	Remarks
1	Attendance Id	1 to 100000	Unique identifier for the attendance record.
2	Student Id	1 to 10000	The unique identifier for the student.
3	Class Id	1 to 12	The class/grade the student is in.
4	Date	Date format (YYYY-MM-DD)	The date for which attendance is recorded.
5	Status	Date format (YYYY-MM-DD)	Whether the student was present or absent on the given date.

High Level/Detailed Design (HLD/DD)

Overview of the system

Provide a block diagram depicting where the database will be located, where the application will run etc. Also, provide details about the database server that is going to be used etc.



1. **Client Devices**:
   - This includes any device that can access the system using a browser or mobile app (laptops, smartphones, tablets).
   - Users include administrators, teachers, students, and parents.
2. **Web Application**:
   - **Frontend**: Built using **React.js**, this will be the user interface for the system, allowing users to perform various tasks such as viewing reports, managing attendance, and updating student/teacher records.
   - **Backend**: **Node.js** will handle server-side processing, routing requests from the frontend to the database, and ensuring role-based access control.

3. **Database Server**:
    - A relational database like **MongoDB** will store all the relevant data for students, teachers, classes, attendance, and fees.
    - The database will ensure data integrity and enforce relationships between the different tables (e.g., linking students to their classes and attendance records).

Design Components

Split the system into its design components. In this case, the components would User verification, user registration, attendance management, etc. For each of the components, provide information in the following format.

Component one

User-verification

Purpose

This component will verify if the user (admin, teacher, student, or parent) attempting to access the system is a valid registered user.

Pseudocode

Pseudocode is written to get more clarity on the component so that the actual implementation is made easier. For the user-verification component :

```
Bool verify_user(user_id, password1)
{
  % Get the user ID and password from the user.
  Get_login_and_password();
```

```
% Verify if the user ID is valid.
If !login_id_valid(user_id)
{
  report_error('Invalid login ID');
  return false;
}

% Retrieve the database entry for the user ID.
if get_database_entry(user_id, database_entry)
{
  % Get the encrypted password stored in the database.
  Get_encrypted_password(user_id, password2);

  % Decrypt the stored password.
  Decrypt_password(password2, password3);

  % Compare the entered password with the decrypted password.
  If compare_passwords(password1, password3)
  {
    % Successfully verified, allow entry.
    Enter_system();
  }
  else
  {
    % Password comparison failed.
    Report_error('Incorrect password. Try again.');
  }
}
else
{
    % Invalid login.
    report_error('User not found in database');
  }
}
```

Component two

User-Registration

Purpose

This component registers new users (admins, teachers, students, and parents) into the system, storing their details for future logins and role-based access.

Pseudocode

Pseudocode is written to get more clarity on the component so that the actual implementation is made easier. For the user-registration component :

```
Bool register_user(user_details)
{
  % Validate the details entered by the user.
  If !validate_user_details(user_details)
  {
    report_error('Invalid user details. Please check and try again.');
    return false;
  }

  % Check if the user already exists in the system.
  if user_exists(user_details.user_id)
  {
    report_error('User already exists. Please log in.');
    return false;
  }
  % Encrypt the password for storage.
  Encrypt_password(user_details.password, encrypted_password);

  % Store user details in the database.
  if !store_user_details(user_details.user_id, encrypted_password, user_details)
  {
    report_error('Error in registration. Please try again.');
    return false;
  }

  % Registration successful.
  Report_success('Registration successful! Please log in.');
  return true;
}
```

Component three

Attendance Management

Purpose

This component manages student attendance records, allowing teachers to input and track daily attendance for their classes.

Pseudocode

Pseudocode is written to get more clarity on the component so that the actual implementation is made easier. For the attendance managemnt component :

```
Bool record_attendance(class_id, student_id, status)
{
  % Check if the class and student exist in the system.
  if ! class_exists(class_id) || !student_exists(student_id)
  {
    report_error('Class or Student not found');
    return false;
  }

  % Get the current date.
  current_date = Get_current_date();

  % Check if attendance for this student on the current date already exists.
  if attendance_exists(class_id, student_id, current_date)
  {
    report_error('Attendance already recorded for today');
    return false;
  }

  % Record the attendance in the database.
  if !store_attendance(class_id, student_id, current_date, status)
  {
    report_error('Error recording attendance');
    return false;
  }

  % Attendance successfully recorded.
  Report_success('Attendance recorded successfully');
  return true;
}
```

Component Four

Grade Management

Purpose

This component allows teachers to input, update, and manage grades for students in their respective classes.

Pseudocode

Pseudocode is written to get more clarity on the component so that the actual implementation is made easier. For the grade management component :

```
Bool record_grade(class_id, student_id, subject, grade)
{
  % Check if the class, student, and subject exist.
  if !class_exists(class_id) || !student_exists(student_id) || !subject_exists(subject)
  {
    report_error('Class, Student, or Subject not found');
    return false;
  }

  % Check if the grade is valid (e.g., A to F or numeric).
  if !validate_grade(grade)
  {
    report_error('Invalid grade entered');
    return false;
  }

  % Store the grade in the database.
  if !store_grade(class_id, student_id, subject, grade)
  {
    report_error('Error recording grade');
    return false;
  }

  % Grade successfully recorded.
  Report_success('Grade recorded successfully');
  return true;
}
```

Component Five

Fee Management

Purpose

This component allows administrators to manage student fee records, including tracking payments and generating outstanding fee reports.

Pseudocode

Pseudocode is written to get more clarity on the component so that the actual implementation is made easier. For the fee management component :

```
Bool record_fee_payment(student_id, amount)
{
  % Check if the student exists in the system.
  if !student_exists(student_id)
  {
    report_error('Student not found');
    return false;
  }

  % Retrieve the student's current outstanding fee amount.
  outstanding_fee = get_outstanding_fee(student_id);

  % If the amount paid is more than the outstanding fee.
  if amount > outstanding_fee
  {
    report_error('Amount exceeds outstanding fee');
    return false;
  }

  % Update the fee record.
  if !update_fee_record(student_id, amount)
  {
    report_error('Error updating fee record');
    return false;
  }

  % Fee payment successfully recorded.
  Report_success('Fee payment recorded successfully');
  return true;
}
```

Component Six

Class Scheduling

Purpose

This component helps administrators schedule classes, assign teachers, and update the class schedule for a school term.

Pseudocode

Pseudocode is written to get more clarity on the component so that the actual implementation is made easier. For the class scheduling component :

```
Bool schedule_class(class_id, teacher_id, subject, time_slot, room_number)
{
  % Check if the class, teacher, and room are valid.
  if !class_exists(class_id) || !teacher_exists(teacher_id) ||
!room_exists(room_number)
```

```
  {
    report_error('Class, Teacher, or Room not found');
    return false;
  }

  % Check for scheduling conflicts (e.g., room or teacher already assigned).
  if scheduling_conflict(teacher_id, time_slot) || room_conflict(room_number, time_slot)
  {
    report_error('Scheduling conflict detected');
    return false;
  }

  % Store the schedule in the system.
  if !store_schedule(class_id, teacher_id, subject, time_slot, room_number)
  {
    report_error('Error scheduling class');
    return false;
  }

  % Class successfully scheduled.
  Report_success('Class scheduled successfully');
  return true;
}
```

Test-Plan (TP)

The test-plan is basically a list of testcases that need to be run on the system. Some of the testcases can be run independently for some components (report generation from the database, for example, can be tested independently) and some of the testcases require the whole system to be ready for their execution. It is better to test each component as and when it is ready before integrating the components.

It is important to note that the testcases cover all the aspects of the system (ie, all the requirements stated in the RS document).

No.	Testcase Title	Description	Expected Outcome	The requirement in RS that is being tested	Result
1	Successful User Registration	The user should be able to register with valid details and a correct password	Registration should be successful, and the user should be able to log in	RS1	Passed
2	Unsuccessful User Registration due to existing ID	Try registering with an ID that already exists in the system	Registration should fail with an error "User ID already exists"	RS1	Passed
3	Successful User Login	Log in to the system using a valid user ID and correct password	Login should be successful, and the user should access the system	RS1	Passed
4	Unsuccessful User Login due to wrong password	Attempt login with a valid user ID but wrong password	Login should fail with an error "Invalid Password"	RS1	Passed
5	Unsuccessful User Login due to invalid user ID	Attempt login with an invalid user ID	Login should fail with an error "Invalid	RS1	Passed

			User ID"		
6	Account Lockout after 3 Wrong Password Attempts	Enter an invalid password 3 times consecutively during login	Account should be locked, and a message should display "Account locked"	RS3	Passed
7	Add Student Information	Admin should be able to successfully add student details to the database	Student details should be successfully saved in the database	RS1	Passed
8	Add Teacher Information	Admin should be able to add teacher details	Teacher details should be saved and available in the system	RS1	Passed
9	Record Student Attendance	Teacher should be able to record student attendance for a class	Attendance should be successfully saved in the database	RS3	Passed
10	Prevent Duplicate Attendance Entries	Attempt to record attendance for a student who already has attendance recorded for the day	System should prevent duplicate attendance entries	RS3	Passed
11	Generate Attendance Report	Generate an attendance report for a specific class	The attendance report should be generated with correct data	RS3	Passed

12	Add Grades for Students	Teachers should be able to enter grades for students	Grades should be successfully saved in the database	RS4	Passed
13	Generate Grade Report	Generate a grade report for a specific student	The grade report should display accurate information	RS4	Passed
14	Add Fee Payment Record	Admin should be able to record fee payments for students	Fee payment should be updated in the student's record	RS5	Passed
15	Generate Outstanding Fees Report	Generate a report listing students with outstanding fees	The report should display all students with unpaid fees	RS5	Passed
16	Add Class Schedule	Admin should be able to schedule classes with teacher assignments	Class schedule should be saved and retrievable	RS6	Passed
17	Prevent Scheduling Conflict for Teachers	Attempt to schedule a teacher for multiple classes in the same time slot	The system should display an error message for scheduling conflicts	RS6	Passed
18	Generate Student Progress Report	Generate a student progress report that includes attendance, grades, and fees	The report should combine all relevant student data	RS7	Passed

			accurately		
19	Display Help Screens	Help screens should provide detailed information about system features	Help information should be displayed correctly in a Q&A format	RS2	Passed