School Management System Documentation

Made By: Yousif Adel

2025

Table of Contents

1. Preface

1.1 Expected Readership

This documentation is intended for:

- **Developers (primarily the creator):**
  To provide a detailed understanding of the system's architecture, design, and implementation, aiding in future improvements, maintenance, and debugging. Since I am the sole developer, this section will serve as a personal guide to track the project's evolution.
- **System Administrators:**
  To assist in the deployment and management of the system. As I continue to improve and refine the system, this documentation will also help with maintenance and updates.
- **End Users (Students, Teachers, Supervisors, Parents**):
  To familiarize them with the features and functionality of the system and guide them in interacting with it based on their respective roles.
- **Future Collaborators or Stakeholders:**
  In case the project expands, and additional developers or stakeholders are involved, this documentation will serve as an introduction to the project, its objectives, and its development.

1.2 Version History

- **Version 1.0 (Release Date: May 2023)**
  - Initial release of the School Management System documentation.
  - Registration page.
  - Improved UI responsiveness.
- **Version 2.0 (Release Date: January 2025)**
  - Full role-based access control (Student, Teacher, Supervisor, Parent).
  - Supervisor can now add, update, or remove students, teachers, and courses.
  - Improved registration system with admission tracking.
  - Add Login Enhanced security features for user login.
  - Minor bug fixes related to the UI and form validation.
  - Improved language-switching functionality (Arabic and English).
  - Refined user role permissions, ensuring supervisors could manage students, teachers, and courses.
  - Light and dark modes are available for UI customization.
  - Toast notifications for login success and other actions.

1.3 Rationale for New Versions

New versions aim to:

- **Enhance User Experience:** Improve navigation and features based on feedback to ensure a user-friendly system.
- **Ensure Scalability:** Optimize performance to support more users and features like course management and role-based access.
- **Improve Security:** Strengthen security with measures like better password management and role-based access to protect user data.
- **Implement New Features:** Add evolving features like registration tracking and advanced notifications to meet project needs.

1.4 Summary of Changes in Version 2.0

- **Role-Based Access Control:** Different user roles (Student, Teacher, Supervisor, Parent) can access specific pages. Supervisors manage students, teachers, and courses.
- **Admission & Registration Tracking:** Supervisors can manage and track admissions.
- **UI Improvements:** Light/dark modes and responsive design for better usability.
- **Toast Notifications:** Inform users of successful actions (login, updates, etc.).
- **Multilingual Support:** English and Arabic languages supported for broader accessibility.

2. Introduction

2.1 Introduction to the Need for the School Management System

- **Purpose**: The School Management System (SMS) was created to streamline and optimize the management of educational institutions.
- **Problem Addressed**: As the school grows, manual processes for managing students, teachers, and courses become inefficient and error prone.
- **Centralized Solution**: The SMS automates tasks such as student registration, course management, and communication among stakeholders (students, teachers, supervisors, parents).
- **Benefits**:
  - **Enhanced Efficiency**: Streamlines administrative tasks, saving time and reducing errors.
  - **Improved Collaboration**: Facilitates seamless communication between all users (students, teachers, supervisors, parents).
  - **Better Data Tracking**: Provides a digital platform for easy tracking of academic and administrative data.

2.2 Overview of Features (Registration, Login, Role-based Access, Courses)

The SMS offers several core features:

- **Registration:** A streamlined process for adding new students to the system, allowing them to enroll in courses and be assigned to relevant user roles.
- **Login:** Secure user authentication with separate login systems for different roles (students, teachers, supervisors, and parents), ensuring privacy and controlled access to the system.
- **Role-Based Access:** Implemented to restrict access to specific pages and features based on user roles. Students can view courses and teachers, teachers can access their courses and students, and supervisors have full control over managing users and courses.
- **Courses Management:** Allows for the creation, viewing, and assignment of courses. Supervisors can add, update, or delete courses as needed.

2.3 Integration with Other Systems (e.g., MySQL Database)

- **Database Integration:** The SMS utilizes a MySQL database for managing crucial data such as user details, courses, and registrations**.**
- **Data Accessibility:** MySQL ensures that data is consistently stored and easily retrieved for efficient system operations.
- **Scalability:** The database is designed to handle the growing volume of data as the institution expands, ensuring smooth management of large datasets.
- **Security and Integrity:** Integration maintains data integrity and security, safeguarding sensitive information.
- **PHP Interaction:** The system uses PHP to interact with the MySQL database, handling user actions and storing data effectively.

2.4 Alignment with Business or Strategic Objectives

The development of the School Management System aligns with key strategic objectives, including:

- **Efficiency:** Automating administrative processes to reduce time spent on manual tasks and allow staff to focus on other priorities.
- **Scalability:** Ensuring the system can handle growing numbers of students, teachers, and courses without performance degradation.
- **Security:** Implementing role-based access control to protect sensitive data and ensure that only authorized users can make changes or access specific information.
- **User Satisfaction:** Providing a user-friendly interface with multilingual support (Arabic and English) and customizable themes (light/dark mode) to ensure a seamless experience for all users.

3. Glossary

3.1 Student, Teacher, Supervisor, Parent

- **Student: A user who can view courses and teachers' information.**
- **Teacher:** A user who can view courses, students, and other teachers' information.
- **Supervisor:** A user with administrative privileges to manage students, teachers, and courses, including adding, updating, and deleting records.
- **Parent:** A user who can view courses and teachers' information related to their child

3.2 Registration and Admission Tables CRUD Operations

The tables in the database store information related to user registration (student, teacher) and admissions, including personal and academic details.

3.3 Role-based Access Control

Stands for Create, Read, Update, and Delete operations, representing the basic functions used to manipulate data within the system (e.g., adding students, viewing course details, updating teacher information, and deleting records).

3.4 Language Handling (Arabic/English)

A security model that restricts access to system resources based on the user's role (student, teacher, supervisor, parent). Each role has specific permissions and access to pages and functionalities.

3.5 Toast Notifications

Small pop-up messages are used to notify users of system actions or events, such as successful logins, registration, or updates. These notifications are temporary and disappear after a short period.

4. User Requirements Definition

4.1 User Services (What each role can do)

- **Student:** View Courses and Teacher Details
- **Teacher:** View courses they teach, and students enrolled.
- **Supervisor:**
  - Add, update, and delete students, teachers, and courses.
  - Manage student registrations and admissions.
- **Parent:** View Courses, Teachers

4.2 Functional Requirements for Students, Teachers, Supervisors, and Parents

- **Students:**
  - Register for courses.
  - View course schedules and teachers.
  - Receive notifications of course updates.
- **Teachers:**
  - Manage courses they are teaching.
  - View student progress and attendance.
  - Add or update course materials.
- **Supervisors:**
  - Oversee the entire system, including managing user roles and permissions.
  - Handle course and user management.
  - Monitor and generate reports for all users.
- **Parents:**
  - Access the courses and teacher information.
  - Receive notifications regarding their child's academic status.

4.3 Non-functional System Requirements (Usability, Security, Performance)

- **Usability:**
  - The system should be easy to use with a clean, intuitive interface.
  - Multi-language support (Arabic and English).
- **Security:**
  - Role-based access control.
  - Secure password management and user data protection.
- **Performance:**
  - Fast load times for all pages.
  - Optimized for scalability as the institution grows.

4.4 Product and Process Standards

- **System should adhere to web development best practices.**
- **Use responsive design for compatibility across devices.**
- **Maintain clean, modular code and follow secure coding practices.**

5. System Architecture

5.1 System Overview (Frontend, Backend, Database)

The system is a web-based application with a frontend built using HTML, CSS, and React. The backend is powered by PHP and MySQL to handle server-side logic and database operations. Data is stored in a MySQL database, which integrates with both the front end and backend.

5.2 Architecture of the Frontend (HTML, CSS, React)

The front end is designed with HTML5 and CSS3 for structure and styling. React is used for building dynamic and interactive user interfaces, improving the overall user experience with seamless navigation and faster page rendering.

5.3 Backend Architecture (PHP, MySQL, Role Management)

The backend is built using PHP, which handles business logic and connects to the MySQL database to manage data. Role management ensures users can only access the resources relevant to their role, enforcing security and access control.

5.4 System Interfaces (Frontend-Backend Interaction

The front end interacts with the backend via PHP-based API endpoints, which process requests (like data retrieval, form submission) and return the necessary data. The backend ensures proper data management and communicates with the database to store or retrieve information based on the frontend's requests.

6. System Requirements Specification

6.1 Functional Requirements (Registration, Login, Role-based Permissions)

- **Registration:** Users (students, teachers, supervisors, parents) must be able to register by providing necessary details, with role-based access.
- **Login:** Secure login functionality for users, validating credentials against the database and granting access according to user roles.
- **Role-based Permissions:** Define specific actions for each role (e.g., students can view courses, supervisors can manage users), ensuring proper access control.

6.2 Non-Functional Requirements (Performance, Security, Usability)

- **Performance:** The system should be optimized for fast loading times and smooth navigation.
- **Security:** Secure login, password encryption, and role-based access control to protect sensitive data.
- **Usability:** The interface should be user-friendly and intuitive, providing easy navigation for different user roles.

6.3 Database Requirements (Logical Organization, Data Storage)

- **Logical Organization:** The database must be organized with separate tables for users, roles, courses, and other necessary data to maintain clear and efficient data management.
- **Data Storage:** Use MySQL to store user information, courses, roles, and system logs, ensuring data integrity and security.

6.4 User Interface Design (Light/Dark Modes, Responsiveness)

- The system should offer light and dark mode options, allowing users to choose their preferred display.
- The design must be responsive, adapting to different screen sizes and devices for a seamless user experience.

6.5 Multilingual Support (Arabic, English)

Provide the option for users to switch between Arabic and English to ensure the system is accessible to a diverse user base.

7. System Model

7.1 UI Design (Pages: Home, Registration, Login, Dashboard)

- **Home Page**: A welcoming page displaying key information and easy navigation to other sections.
- **Registration Page:** A form to register users, allowing them to create accounts with necessary details.
- **Login Page:** A secure login form for authenticating users based on their credentials.
- **Dashboard:** A central hub for users to manage their actions based on roles (view courses, manage students, etc.).

7.2 Database Design (Tables: Users, Roles, Courses)

- **Users Table:** Stores user information such as username, email, password, and role.
- **Roles Table**: Defines different user roles (e.g., student, teacher, supervisor) and their permissions.
- **Courses Table:** Stores course details, including course name, description, and teacher information.

7.3 Authentication and Authorization Flow

Describes the process of user authentication (verifying user credentials) and authorization (granting access based on user roles) to ensure secure access to the system.

7.4 Security Design (Login Verification, Role Protection)

- Describes how the login system verifies user credentials and ensures secure access.
- Role protection ensures that each user can only access pages and actions appropriate for their role, preventing unauthorized access to sensitive data.

8. Implementation

8.1 Frontend Implementation (HTML, CSS, React, React-Bootstrap)

- **HTML:** Used for the structure of the web pages.
- CSS: For styling and responsive design.
- **React:** JavaScript library for building dynamic and interactive user interfaces.
- **React-Bootstrap:** A library for responsive components to enhance UI.
- **Functions**

`fetchData:`

Description:

Fetches data asynchronously from the provided API endpoint and updates the UI based on the result. Displays a success or error message depending on the outcome.

Parameters:

- `setData:` Updates the state with the fetched data.
- `setMsg`: Displays a success message when the operation succeeds.
- `setError`: Sets an error message when an issue occurs.
- `METHOD`: The HTTP method (e.g., GET, POST) used for the request.
- `URL`: The API endpoint to fetch data from.
- `dataLanguage`: Determines the language for messages ("ar" for Arabic, "en-US" for English).

`getRoleNameFromID:`

Description:

Returns the role name corresponding to a given role ID. If the ID is not recognized, it defaults to "Unknown"

Parameters:

`roleID`: A string representing the role ID (e.g., "1", "2", "3").

Returns:

- `"Student"` for role ID "1".
- `"Teaching-staff"` for role ID "2".
- `"Parent"` for role ID "3".
- `"Unknown"` if the role ID is not recognized.

`handleSelectChange:`

Description:

Updates the selected value in a dropdown when the user selects.

Parameters:

- `e`: The event object from the select input
- `setSelect`: Function to update the selected value.

handleInputChange:

Description:

Handles the input field changes and validates the input based on its type (e.g., name, email, password). Displays appropriate error messages if validation fails

Parameters:

- `e`: The event object from the input field.
- `type`: A string representing the type of input (e.g., "name", "email", "password").
- `setError`: Function to update error messages.
- `setData`: Function to update the input data value.
- `dataLanguage`: The language preference for displaying error messages
- `password (Optional)`: The password value used for validation when checking confirmation passwords.

Validation Logic:

- Ensures required fields are not empty.
- Validates email format.
- Enforces password length and special character requirements.
- Confirms password matches during registration.
- Ensures salary is a positive value and above a minimum threshold.

handleDelete:

Description:

Handles the deletion of an item by sending a delete request to the server, updating the UI, and displaying notifications based on the result.

Parameters:

- `e`: The event object triggered by the delete action
- `ID`: The ID of the item to be deleted.
- `type`: The type of item being deleted.
- `DELETED_URL`: The API endpoint for the delete request.
- `setData`: Function to update the state with fetched data after deletion.
- `setMsg`: Function to update the message state.
- `setError`: Function to handle and update error state.
- `METHOD`: The HTTP method for the delete request.
- `VIEW_URL`: The API endpoint for fetching the updated data after deletion.
- `dataLanguage`: The language preference for messages and notifications.

Functionality:

- Prevents the default event behavior.
- Sends a delete request to the specified API endpoint.
- Displays toast notifications based on the success or failure of the operation.
- Fetches updated data after successful deletion.
- Handles network or unexpected errors and logs relevant messages.Ensures salary is a positive value and above a minimum threshold.

handleNavigateUpdate:

Description:

Handles navigation to the update page for a specific item using React Router.

Parameters:

- `e`: The event object triggered by the navigation action.
- `navigate`: The navigation function from React Router to change the route.
- `type`: The type of item to be updated.
- `ID`: The ID of the item to be updated.

Functionality:

- Prevents the default behavior of the event.
- Navigates to the update page for the specified item using its type and ID.

handleSubmitFunction:

Description:

Manages form submission, including payload validation, API request, and user redirection based on the response.

Parameters:

- `e:` The event object triggered by form submission.
- `setMsg:` Function to set success or informational messages.
- `setError:` Function to set error messages.
- `dataLanguage:` Language preference for messages.
- `navigate:` Function for route navigation.
- `URL:` The API endpoint for the request.
- `METHOD:` The HTTP method for the request (e.g., "POST", "PUT").
- `payload:` The data to be sent in the request body.
- `type:` The type of operation (e.g., login, register).
- `role:` The user role for redirection.

Functionality:

- Notification Handling (use react-toastify)
- From Validation
  - Prevents default form submission behavior.
  - Checks if all fields in the payload are filled.
- API Request Handling
  - Sends an API request with the specified method and payload.
  - Captures and logs the raw response for debugging.
  - Tries to parse the response into JSON.
- Response Handling
  - If the request succeeds:
    - For login:
      - Saves user role and type in localStorage.
      - Displays success message.
      - Redirects based on role after 3 seconds.
    - For registration:
      - Displays success message.
      - Redirects to login after 3 seconds.
    - For other operations:
      - Displays success message.
      - Redirects to a specific page after a delay.
  - If the request fails, sets an error message.
- Error Handling

- Handles API errors and JSON parsing failures.
- Displays appropriate messages for network or request failures.
- Shows an error message if required fields are missing.

`useLanguageDirection (custom Hook):`

Description:

Automatically updates the document's language and text direction (LTR or RTL) based on the user's selected language from the Redux store.

Functionality

- **Retrieving the Language**
  Uses `useSelector` to access `state.language` from the Redux store.
- **Applying Language Direction**
  - Uses `useEffect` to update the document's `lang` and `dir` attributes whenever `dataLanguage` changes.
  - If `dataLanguage` is **not Arabic (`ar`)**, sets:
    ```
    document.documentElement.lang = "en"
    document.documentElement.dir = "ltr"
    ```
  - If `dataLanguage` **is Arabic (`ar`)**, sets:
    ```
    document.documentElement.lang = "ar"
    document.documentElement.dir = "rtl"
    ```

8.2 Backend Implementation (PHP, MySQL)

- **PHP:** The server-side scripting language used to handle user requests and communicate with the database.
- **MySQL:** Database used for storing and retrieving data, such as user information and course details.
- **PHP Functions**

`getAllData ($conn, $tableName, ...$attr):`

Description: Fetches all rows from a specified database table.

Parameters:

- `$conn`: The database connection object.
- `$tableName`: The name of the table to fetch data from.

Returns:

- An associative array with a success message and the data if successful.
- A failure message if no data is found.

`getData ($conn, $sql):`

Description: Executes a custom SQL query and fetches data.

Parameters:

- `$conn`: The database connection object.
- `$sql`: The SQL query to execute.

Returns:

- An associative array with the success message and fetched data
- A failure message if the query fails or returns no data.

`updateData ($conn, $table, $data, $condition):`

Description: Updates a record in each table based on the condition.

Parameters:

- `$conn`: The database connection object.
- `$table`: The name of the table to update.
- `$data`: An associative array of key-value pairs representing column names and new values.
- `$condition`: The condition to identify which record(s) to update.

Returns:

- A success message if the update is successful.
- A failure message with error details if the update fails.

`deleteData ($conn, $table, $id):`

Description: Deletes a record from a table based on the given ID.

Parameters:

- `$conn`: The database connection object.
- `$table`: The name of the table to delete from.

Returns:

- A success message if the delete operation is successful.
- A failure message if there's an error.

`insertData ($conn, $table, $data):`

Description: Inserts data into a specified table in the database.

Parameters:

- `$conn`: The database connection object.
- `$table`: The name of the table to insert data into.
- `$data`: An associative array where keys are column names and values are the corresponding values to insert.

Returns:

- A successful message if the insert operation is successful.
- A failure message with error details if the insert fails.

8.3 Toast Notifications Implementation

Implemented using JavaScript or React to notify users of actions like successful logins, course updates, or error messages, enhancing user experience.

8.4 Role-based Access Control Implementation

It is used to restrict access to certain parts of the system based on user roles, ensuring that users can only perform actions relevant to their roles (e.g., only supervisors can manage users).

8.5 Language Switching Implementation

A feature that allows users to switch between Arabic and English based on their preference, making the system accessible to a wider audience.

9. System Testing and Validation

9.1 Unit Testing (Frontend and Backend)

- **Frontend Testing**: Test individual components and pages for correct functionality and responsiveness, ensuring UI elements like forms and buttons work as expected.
- **Backend Testing**: Test individual backend modules, such as database queries, server-side logic, and API endpoints, to ensure proper handling of data and system functions.

9.2 Functional Testing (Registration, Login, Role-based Access)

- **Registration**: Test the registration process to ensure users can sign up with correct data and role assignments.
- **Login**: Test the login system to confirm users can access the system securely based on their credentials.
- **Role-based Access**: Test that users can only access features appropriate to their roles (e.g., students can only view courses, teachers can manage their courses).

9.3 Performance Testing (Speed, Load Time)

- **Speed**: Measure how quickly pages load under normal usage and test how the system handles data retrieval and page rendering efficiently.
- **Load Time**: Test the system's ability to perform under heavy traffic and usage, ensuring the system remains responsive.

9.4 Security Testing (Login Validation, Access Control)

- **Login Validation**: Test the security of login functionality, ensuring password encryption and proper authentication.
- **Access Control**: Test role-based access to make sure unauthorized users cannot access restricted pages or functions.

10. System Evolution

10.1 Anticipated Changes and Enhancements

- **User Interface**: Improvements to the user interface for a more modern, intuitive design.
- **Additional Roles**: Add more specialized user roles, such as admin and financial officers, to extend system capabilities.
- **Scalability**: Enhance the backend to accommodate a growing number of users and courses.

10.2 Planned Features (Advanced User Management, Reporting)

- **Advanced User Management**: Features for deeper user role customization and permissions, allowing administrators to manage and assign complex roles and tasks.
- **Reporting**: Generate detailed reports for students, teachers, and courses, including attendance, performance, and financial data.

11. Appendices

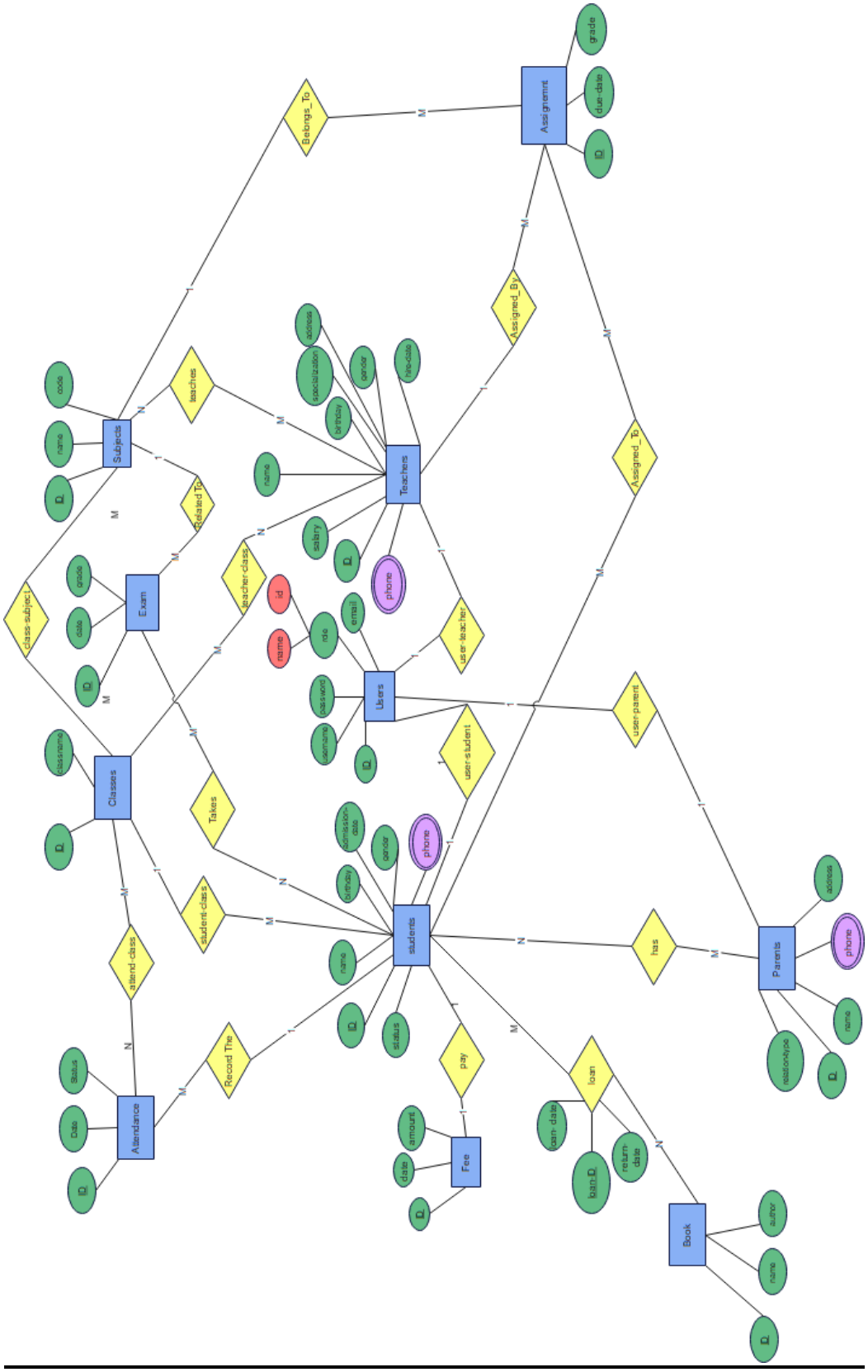
11.1 Hardware Requirements (Server, Local Machine)

- **Server**: Minimum hardware requirements for hosting the School Management System, including server specifications for optimal performance (e.g., CPU, RAM, storage).
- **Local Machine**: Hardware specs needed for developers or users running the system locally (e.g., OS, processor, RAM).

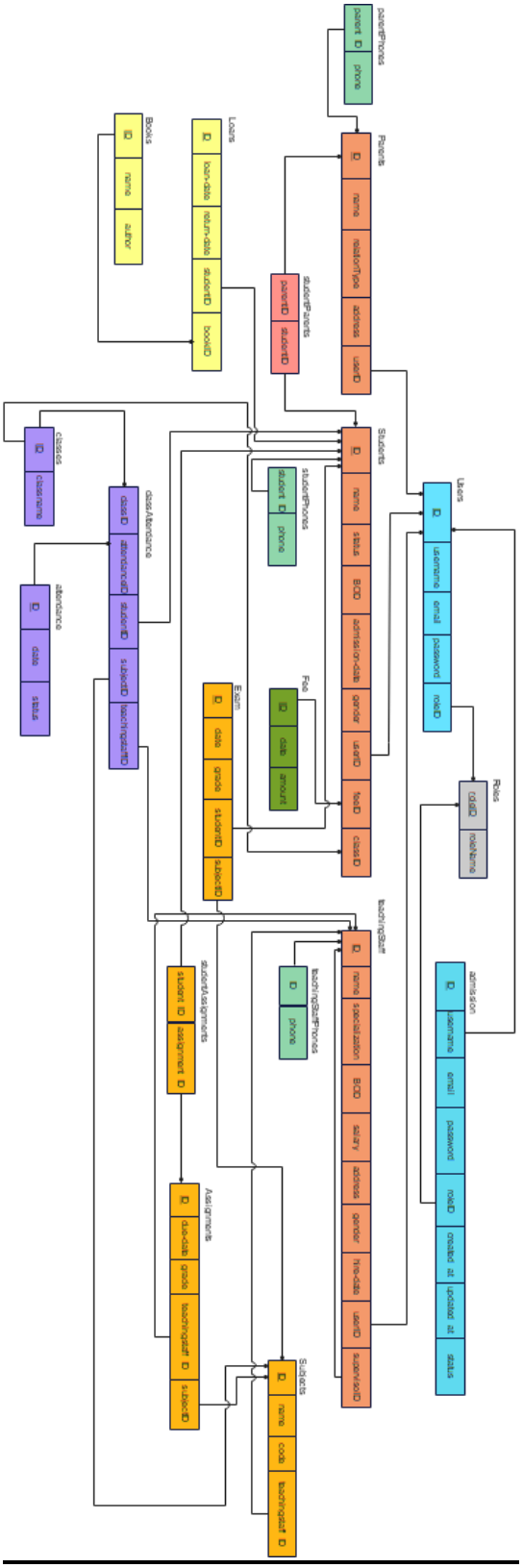
11.2 Database Schema (Tables, Keys)

A detailed diagram or description of the database schema, outlining the key tables (Users, Roles, Courses) and relationships between them, including primary and foreign keys.

ERD


Assignment
grade
due-date
ID
Belongs_To
M
1
Assigned_By
Assigned_To
Subjects
code
name
ID
teaches
related To
Teachers
access
socialization
gender
birth-date
birthday
name
salary
ID
phone
email
role
id
name
Exam
grade
date
ID
class-subject
teacher-class
user-teacher
Users
password
username
ID
user-student
user-parent
Classes
classname
ID
Takes
student-class
attend-class
students
admission-date
gender
birthday
phone
name
ID
status
has
Parents
access
phone
name
relationtype
ID
Attendance
Status
Date
ID
Record The
pay
Fee
amount
date
ID
loan
loan-date
loan ID
return-date
Book
autor
name
ID

Final Schema



11.3 API Documentation (If Applicable)

API endpoints (if any) used for integrating the SMS with other systems or services, including request formats, responses, and example use cases.

11.4 User Manual (Frontend, Backend, Role Instructions)

- **Frontend**: Instructions for users on how to navigate the system, register, log in, and perform basic tasks.
- **Backend**: Guide for administrators and supervisors on managing users, courses, and system settings.
- **Role Instructions**: Clear instructions for each user role (student, teacher, supervisor, parent) explaining their permissions and available actions.