ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERST

COLLEGE OF ENGINEERING
SOFTWARE COMPONENT DESIGN

Name **Id**

Dawit Berihun ets0361/13

Agile-for prototype Software Development: A Case Study of the school management system Website Project

Introduction:

The School Management Website is a comprehensive solution designed to streamline administrative and academic processes within educational institutions. It offers role-based access for administrators, teachers, students, and parents, enabling efficient management of tasks such as attendance tracking, fee payments, class scheduling, and communication. By leveraging Agile development and prototyping, this project ensures iterative progress and stakeholder involvement, delivering a user-friendly and functional system.

1. Project Setup

- **Repository Initialization:**
  1. Create a GitHub repository named school-management-website.
  2. Set up the repository with branch protection rules to avoid direct commits to the main branch.
  3. Configure my development environment with necessary tools and frameworks.
- **Branching Strategy:**
  1. **Main Branch**: For stable and tested code ready for deployment.
  2. **Feature Branches**: One branch per feature (e.g., feature/login-system, feature/attendance-module).

3. **Bugfix Branches**: For resolving issues (e.g., bugfix/form-validation-error).

2. Agile Development Workflow

- **Sprint Planning:**

    1. Break the project into manageable sprints (e.g., 2 weeks per sprint).
    2. Define the SMSP features for each sprint, such as:

        1. Sprint 1: User authentication and database setup.
        2. Sprint 2: Student management and attendance modules.
        3. Sprint 3: teacher management and attendance modules.
        4. Sprint 4: admin management.
        5. Sprint 3: Fee payment and reporting dashboards.
        6. Sprint 4: Testing, deployment, and feedback integration.

Sprint	**Task**	**Deliverable**
Sprint 1	Design UI, Authentication System	Login Page Prototype
Sprint 2	Develop Dashboard, Database Schema	Admin Dashboard, Basic
Sprint 3	Add Attendance, Fee Management Modules	Functional Modules
Sprint 4	Testing and Deployment	Fully Functional Website

- **Daily Standups:**

Each participant typically answers **three key questions**:

1. **What did I accomplish yesterday?**
2. **What am I planning to do today?**
3. **Are there any blockers or challenges?**

The meeting should last **15 minutes or less** and is usually conducted standing (to encourage brevity).

Example Scenarios for Daily Standups

Scenario 1: Sprint Progress Tracking

Team Member 1 (Frontend Developer):

- ✓ *Yesterday*: Completed the login page UI and integrated basic validation.
- ✓ *Today*: Start working on the admin dashboard layout.
- ✓ *Blocker*: Need clarification on the exact sections for the admin dashboard.

Team Member 2 (Backend Developer):

- ✓ *Yesterday*: Set up the database schema for users and roles.
- ✓ *Today*: Implement authentication endpoints.
- ✓ *Blocker*: Waiting for the frontend team to confirm field requirements for user roles.

Scrum Master: Helps clarify the dashboard requirements and coordinates a quick follow-up meeting if needed.

Scenario 2: Mid-Sprint Adjustments

Team Member 1 (Tester):

- ➢ *Yesterday*: Tested login functionality and found two major bugs related to password validation.
- ➢ *Today*: Retest after fixes and start testing the attendance module.
- ➢ *Blocker*: Lack of test data for the attendance module.

Team Member 2 (Backend Developer):

- ➢ *Yesterday*: Fixed reported bugs in the login functionality.

- *Today*: Work on integrating attendance submission with the database.
- *Blocker*: None.

Scrum Master:Assigns a team member to generate test data for the attendance module and ensures testing proceeds without delays.

Scenario 3: Addressing Challenges

Team Member 1 (Parent Portal Developer):

a. *Yesterday*: Developed the basic structure of the parent portal UI.
b. *Today*: Add functionality for viewing student reports.
c. *Blocker*: Unsure how to structure the API calls for fetching student data.

Team Member 2 (API Developer):

d. *Yesterday*: Completed the student report API endpoint.
e. *Today*: Support the frontend team in integrating the API.
f. *Blocker*: None.

Scrum Master: Connects the frontend and backend developers to resolve API integration issues.

for Effective Daily rule :

Be Time-Conscious:Stick to the 15-minute limit. Avoid long discussions; save detailed problem-solving for separate meetings.

Use Task Tracking Tools: Reference tools like Jira, Trello, or GitHub project boards during the standup to show progress.

Encourage Participation: Ensure every team member gets a chance to share updates. Rotate the order of speaking to maintain engagement.

Focus on Outcomes: Emphasize what will be accomplished by the end of the day.

Address Blockers: resolving blockers immediately after the standup.

- **Sprint Review and Retrospective:**
    1. Review completed features at the end of each sprint.
    2. Incorporate feedback into the next sprint.

3. Prototyping

I use **Figma** to design and iterate on prototypes. Focus on:

Purpose: Authenticate users and provide role-based access.

Common Form Fields:

1. Login Page:

1. **Username/Email**:placeholder: *Enter your username or email* Input type: Text
2. **Password**: Placeholder: *Enter your password* Input type: Password
3. **Role Selection Dropdown**: Options: Admin, Teacher, Student, Parent
4. **Buttons**: Login: Submit the form. Forgot Password: Redirect to password recovery.

Additional Notes:

- ✓ I add validation (e.g., *Username is required*, *Password must be at least 8 characters*).
- ✓ Use placeholders and error messages for user feedback.

2. Admin Dashboard

Purpose: Manage users, track data, and generate reports.

Common Forms:

Add Student/Teacher:

- ✓ Full Name (Text input).
- ✓ Role (Dropdown: Student/Teacher).
- ✓ Grade/Class (Dropdown for students, Text input for teachers).
- ✓ Contact Information (Email/Phone number).
- ✓ Submit Button.

Generate Report:

- ✓ Report Type (Dropdown: Attendance, Fees, Grades).
- ✓ Time Period (Date picker).
- ✓ Export Options (Checkbox: PDF, Excel).
- ✓ Generate Button.

Class Scheduling:

- ✓ Class Name (Text input).
- ✓ Teacher Assignment (Dropdown of teacher names).
- ✓ Time Slot (Dropdown or time picker).
- ✓ Submit Button.

3. Attendance Module

Purpose: Simplify the attendance marking process for teachers.

Common Form Fields:

1. **Select Class**:Drop-down to choose the class.
2. **Date**: Date picker for attendance date.
3. **Attendance List**: Student Names (Auto-populated). Status (Checkbox or Radio Button for *Present*, *Absent*, *Late*).
4. **Save Button**: Save the attendance record in the database.

Additional Notes: Include bulk actions (e.g., *Mark all Present*).

4. Fee Management

Purpose: Allow parents to view and pay fees.

Common Form Fields:

1. **Student Information**: Auto-populate fields (e.g., Student Name, Class, Outstanding Balance).
2. **Payment Options**: Dropdown for Payment Method (e.g., Card, Bank Transfer).
3. **Card Details Form (if applicable)**:

    - ✓ Cardholder Name (Text input).
    - ✓ Card Number (Number input).
    - ✓ Expiry Date (MM/YY).
    - ✓ CVV (Number input).

4. **Payment Button**: Trigger the payment gateway.

5. Parent Portal

Purpose: Provide parents access to their child's progress and communication tools.

Common Forms:

Student Progress Report:

- ✓ Dropdown for selecting a student (if parent has multiple children).
- ✓ View Button to display grades, attendance, or feedback.

Messaging Form:

- ✓ Teacher Selection (Dropdown of assigned teachers).
- ✓ Message Content (Text area).
- ✓ Attachments (Optional file upload).
- ✓ Send Button.

Additional Notes: Add a history section where parents can see sent and received messages.

Form Design rule

1. **Consistency**: Use uniform spacing, font sizes, and button styles across forms.
2. **Validation Feedback**: Show error messages immediately for invalid inputs.
3. **Responsive Design**: Ensure forms adapt to different devices (desktop, tablet, mobile).
4. **Usability**:
    - Use auto-fill for repetitive fields (e.g., parent name when logging in).
    - Group related inputs together for better flow.

4. Database Design

Design the following tables to support the functionalities:

1. **Users**: Manages login credentials and roles.
2. **Students**: Stores student details.
3. **Teachers**: Stores teacher information.
4. **Classes**: Manages class schedules and teacher assignments.
5. **Attendance**: Tracks daily attendance status.
6. **Fees**: Records fee transactions and due payments.
7. **Messages**: Enables communication between parents and teachers.

5. Development Workflow

1. **Collaborative Coding**:
    1. Use GitHub for version control.
    2. Regularly pull the latest code (git pull origin main).
    3. Push feature branches for review (git push origin feature/<branch-name>).
2. **Pull Requests**:
    1. Ensure code reviews before merging into the main branch.

    2. Run automated tests for new features.

6. Continuous Integration (CI)

1. Set up CI pipelines to automatically:
    1. Run unit tests.
    2. Check code quality.
    3. Deploy tested code to a staging environment.

7. Testing

1. **Unit Testing**: Test individual components (e.g., login system, attendance module).
2. **Integration Testing**: Verify that modules work together (e.g., attendance updates in dashboards).
3. **User Testing**: Have students, parents, and teachers test the system for usability.

8. Deployment

1. Use platforms like AWS, Heroku, or Google Cloud to host the website.
2. Configure continuous deployment for easy updates.

9. Documentation

1. **Technical Documentation**:
    1. System architecture.
    2. Database schema.
    3. API endpoints.
2. **User Manuals**: istructions for using the admin dashboard, attendance module, and parent portal.

10. Presentation

1. Demonstrate:

    1. User authentication flow.
    2. Admin features (student and teacher management).
    3. Key modules (attendance, fee tracking, reporting).

2. Highlight:

    1. How Agile and GitHub workflows enhanced development.
    2. Design iterations through prototypes.