Cover Page

Project Name: Student Lunch Payment Application

Developer: Amstrong Akendung

Submission Date: 11/01/2024

Table of Contents

1. Introduction

1.2 Purpose:

This document outlines the system requirements for the Online Student Lunch Payment System. This system aims to replace traditional cash and check-based lunch payment methods with a more efficient, secure, and convenient online solution for parents, students, and school administrators.

1.3 Scope:

The Online Student Lunch Payment System will provide the following functionalities:

- Secure online payments for student meals
- Account management for parents (add students, view balances, transaction history)
- Automated payment options
- Low-balance alerts
- Student balance checks
- Meal pre-ordering (optional)
- Reporting and administrative functions

2. Customer Problem Statements and System Requirements

2.2 Customer Problem Statement:

Schools, parents, and students face inefficiencies in managing lunch payments due to the lack of a centralized, digital payment system. Current lunch payment systems often involve cash or checks, leading to:

- **Inefficiency:** Time-consuming for school staff.
- **Security Risks:** Potential for errors and theft.
- **Inconvenience:** Difficult for parents to manage payments.
- **Lack of Visibility:** Limited access to real-time balance information.

2.3 System Requirements:

The Student Lunch Payment Application will address these problems by providing:

- Secure online payment options for parents.
- Easy account creation and management for parents and students.
- Allow parents to make lunch payments to their students' accounts.
- Provide real-time balance updates for student accounts.
- Send low balance alerts to parents.
- Track transaction history for transparency and accountability.
- Support multi-student management for families with more than one child.

3. Functional Requirements Specification

3.2 Stakeholders:

- **Parents:** Primary users of the system, interested in convenient payment options, account management, and balance tracking.
- **Students:** Users who need to view their balances and potentially pre-order meals.
- **School Administrators:** Managed accounts, generated reports, and configured system settings.
- **Cafeteria Staff:** Users who process payments, view student balances, and potentially manage meal pre-orders.
- **School District IT Staff:** Responsible for system installation, maintenance, and integration with existing school systems.
- **School District Finance Department:** Oversees financial aspects, including payment processing and reconciliation.

3.3 Actors and Goals:

| Actor	Type	Goal
Parent	Human	Manage accounts, make payments, track balances, set up alerts.
Student	Human	View balance, pre-order meals (optional).
Administrator	Human	Manage accounts, generate reports, configure settings.
Cafeteria Staff	Human	Process payments, view balances, manage pre-orders (optional).
System	Software	Process payments, manage data, generate reports, send notifications.

3.4 Use Cases:

Here's a breakdown of the use cases based on your functional requirements, along with estimated effort:

Use Case ID	Use Case Name	Description	Estimated Effort (Points)
UC-1	Create Account	Parent creates a new account with personal information and login credentials.	2
UC-2	Manage Account	Parent updates account information (e.g., contact details, password).	1
UC-3	Add Student	Parent adds a student to their account, linking them to their profile.	1
UC-4	Make Payment	Parent makes a payment to a student's account using various payment methods (credit/debit card, electronic check, etc.).	3
UC-5	View Transaction History	Parent views a history of transactions (payments and purchases) for a specific student.	2
UC-6	View Account Balance	Parent views the current lunch balance for a specific student.	1
UC-7	Set Up Automatic Payments	Parent configures automatic recurring payments to a student's account.	3
UC-8	Receive Low Balance Alerts	System sends an alert (email or SMS) to the parent when a student's balance falls below a defined threshold.	2
UC-9	View Student Balance (Student)	Student views their current lunch balance.	1
UC-10	Pre-order Meal (Student)	Student (or parent) pre-orders a meal for a future date (optional).	2
UC-11	Manage Student	Administrator creates, modifies, or deletes student	2

	Accounts (Admin)	accounts.	
UC-12	Manage Parent Accounts (Admin)	Administrator creates, modifies, or deletes parent accounts, potentially linking/unlinking them to student accounts.	2
UC-13	Generate Reports (Admin)	Administrator generates various reports (payment summaries, low balance reports, meal counts, etc.) for specific periods.	4
UC-14	Configure System Settings (Admin)	Administrator configures system-wide settings (e.g., low balance thresholds, payment options, meal pricing).	3
UC-15	Process Payment (Cafeteria Staff)	Cafeteria staff processes a student's payment (online or offline).	2
UC-16	View Student Balance (Cafeteria)	Cafeteria staff views a student's current balance at the point of sale.	1
UC-17	Manage Meal Pre-orders (Cafeteria)	Cafeteria staff views and manages meal pre-orders for students (optional).	2

3.5 Use Case Diagram:
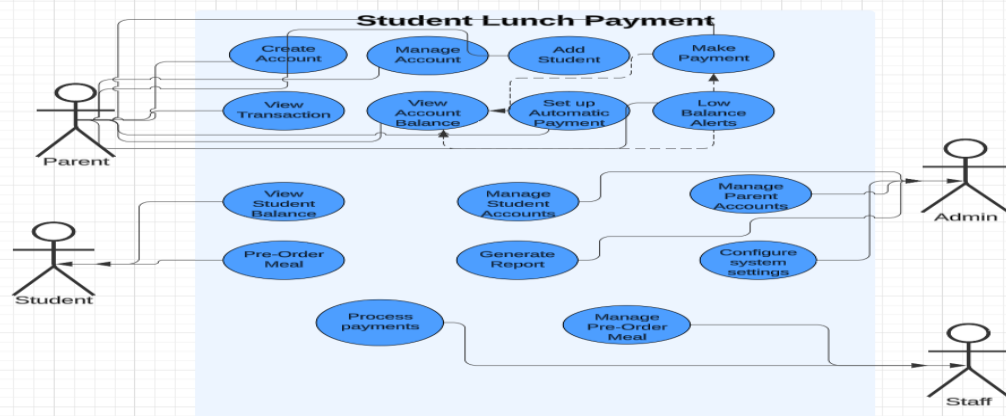
## 3.6 Use Case Description

- **Make a Payment to a Student's Account**
    - Description: Allows parents to top up their child's lunch account.
    - Trigger: Parent initiates payment request on the app.
- **View Account Balance**
    - Description: Displays real-time balance information for each student account.
    - Trigger: User logs into their account.
- **Set Low Balance Alert**
    - Description: Notifies parents when the account balance falls below a defined threshold.
    - Trigger: Parent sets a threshold level.
- **View Transaction History**
    - *Description*: Provides access to all past transactions, including payment dates, amounts, and statuses.
    - *Trigger*: User navigates to the transaction history section.
- **Manage Multiple Student Accounts**
    - *Description*: Allows parents to manage lunch balances and payments for multiple students from a single account.

4. System Sequence Diagram

4.2 Make Payment:

The below Steps explain what happens when a Parent makes a payment to a student's lunch account:

- The Parent actor sends a "Make Payment Request" message to the System, providing the studentID, amount, and paymentMethod.
- The System then sends a message to the Account object to retrieve the account associated with the studentID.
- The System creates a Payment object with the provided amount and paymentMethod.
- The System sends a validatePayment() message to the Payment object to ensure the payment details are valid.
- Alternative flows based on validation:
  - **Valid Payment:**
    - The Payment object interacts with an External Payment Gateway to process the payment.
    - If successful, the Payment object is applied to the Account, a Transaction record is created, and a success message is returned to the Parent.
  - **Invalid Payment:**
    - If the payment validation fails, a failure message with the reason is returned to the Parent.
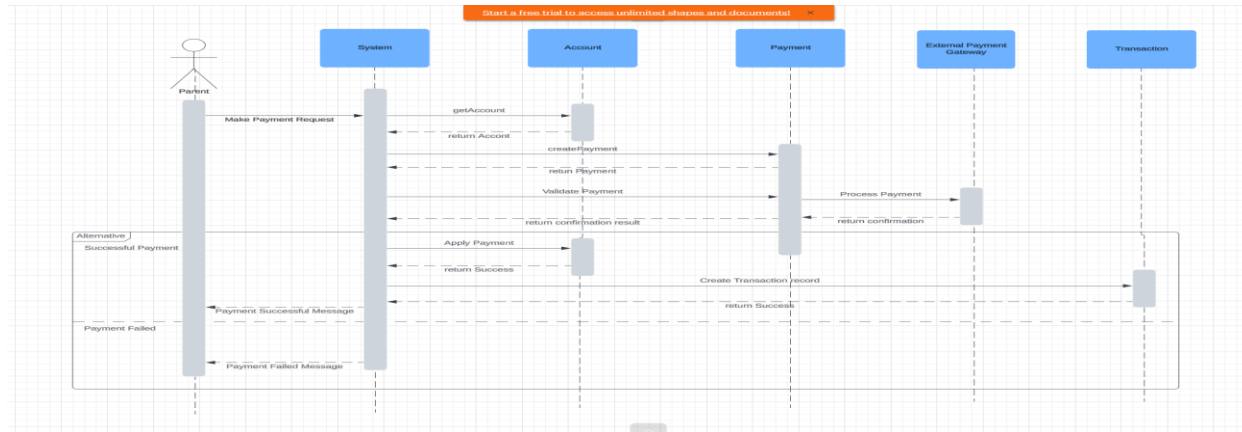
4.3 Generate Low Balance Report

Below, the Use case diagram visually represents the interactions that happen when an Administrator generates a report that lists all Student accounts with low balances.
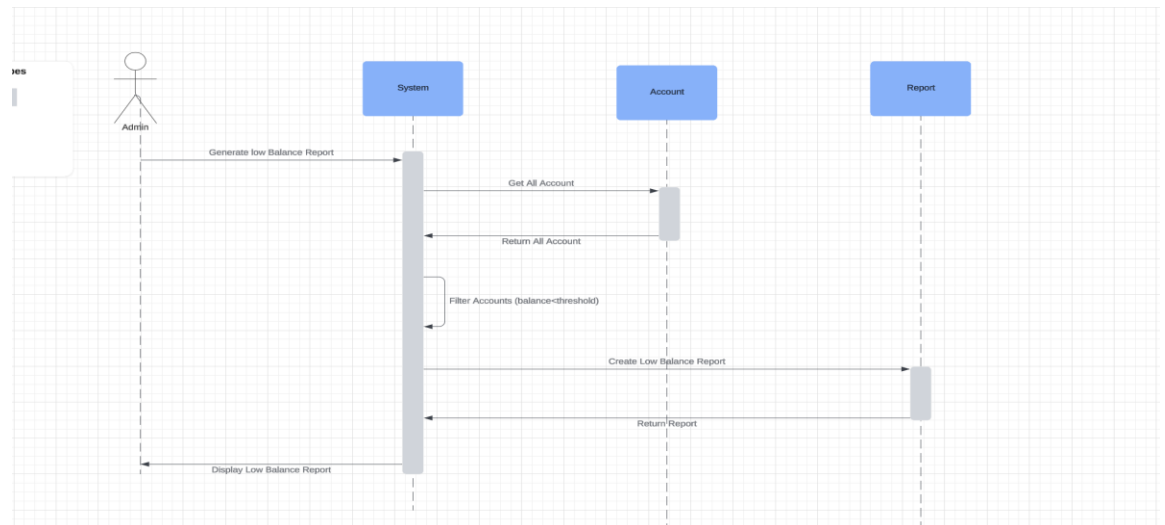
5. Activity Diagrams:

5.2 Adding Student:

This use case describes the process of a parent adding a new student to their account.

States:

- Initial State: The parent is logged in and can access their account
- Final State: System Add a student to the Parent Account

Actions: Parent login and can see his or her account information. Parents click on Add Student, and the system displays the Add Student form. Parents enter student information. Parents submit the form, and the system validates student information and creates Student and Account objects. Display success or error message.
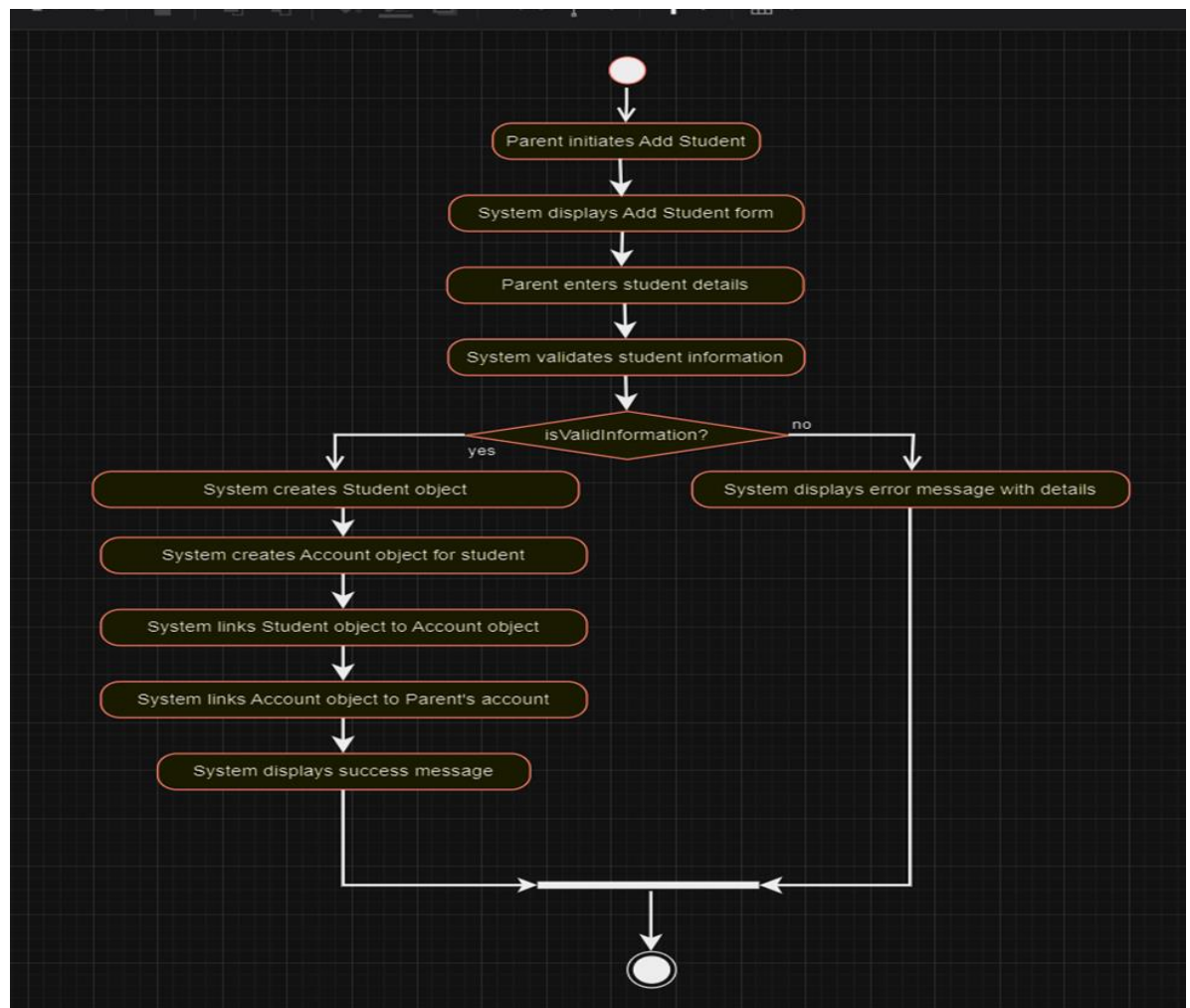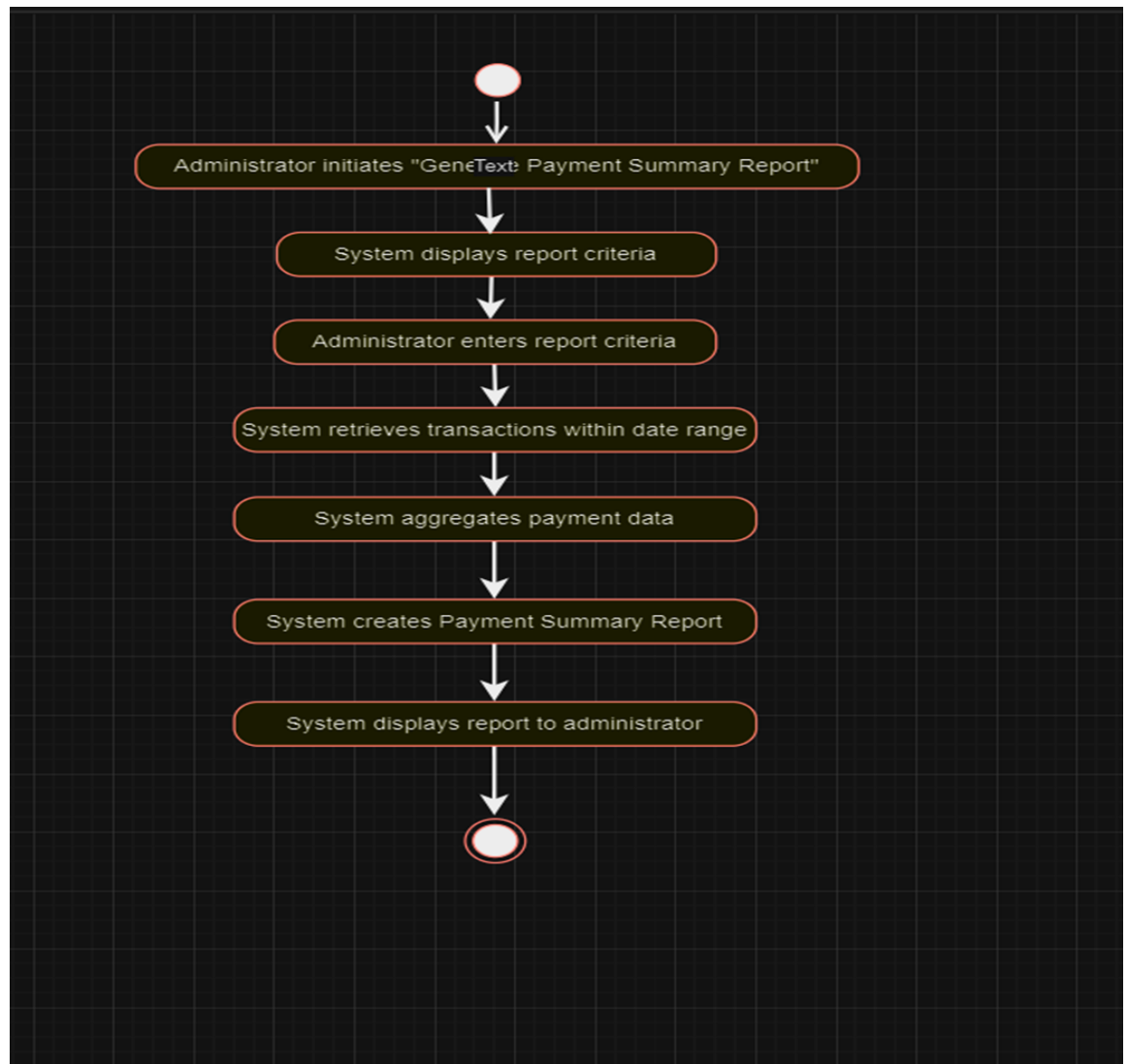
5.3 Generate Payment Summary Report:

States:

- Initial State: Administrator logged in and ready to generate a report.
- Final State: System Shows report to the administrator.

Actions: The administrator logs in and can see his or her account information. Enters report criteria, System generates the report, and Administrator views the generated report.

6. User Interface Specification

6.2 Make a Payment to a Student Account

- **Login Page**: Parent logs in securely.
- **Dashboard**: Shows all students, transaction history, and payment options.
- **Student Account**: Shows a Student Account
- **Payment Screen**: Input fields for amount and payment method, confirmation pop-up.
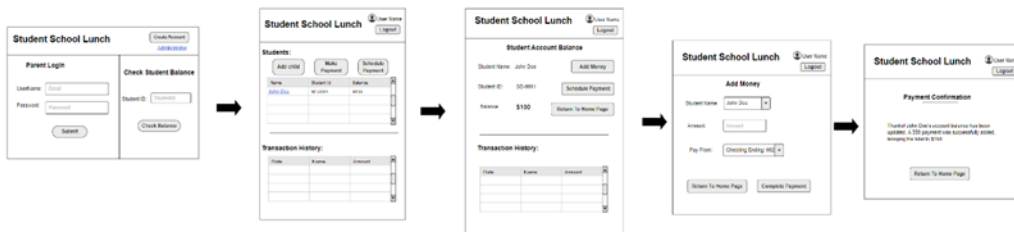
6.3 Views transactions History of a student.

- **Login Page**: Parent logs in securely.
- **Dashboard**: shows all students, transaction history, and payment options.
- **Student Account**: shows a student account with a balance.
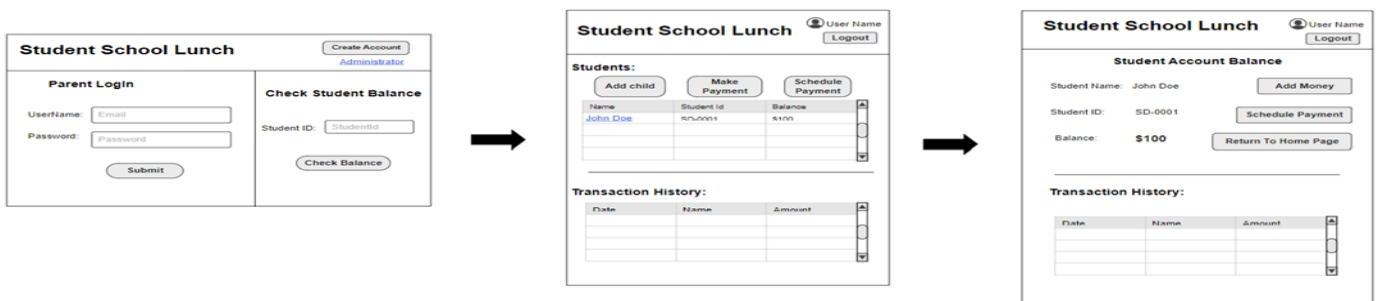
7. Project Plan

7.2 Software:

- **Node.js:** This is used to run JavaScript on the server side. It's the backbone for running an Express server.
- **Express:** A framework for Node.js that helps build RESTful APIs, handling routing and middleware for server-side operations.
- **React:** A JavaScript library for building the user interface, allowing for a dynamic and component-based front end.
- **MySQL:** A relational database management system (RDBMS) used to store and manage data such as student and parent accounts.
- **JWT (JSON web token):** A library for generating and validating JSON Web Tokens, essential for implementing authentication and authorization.
- **Cookie-parser:** A middleware for Express to handle cookies, enabling HttpOnly and Secure cookies to store tokens.
- **Bcrypt:** Used to hash and validate passwords securely.
- **Npm:** Package managers for installing dependencies and managing libraries in Node.js project.

7.3 Hardware:

- Web server: Hosting environment for Node.js backend and API
- Database server: XAMP provides an easy-to-set-up environment, allowing me to manage MySQL locally.

7.4 Network:

- Secure internet connection
- Firewall protection.

7.5 Framework

- **Backend:** Node.js with the Express framework for handling server-side routing and API endpoints.

- **Database:** MySQL for reliable data storage
- **Front-end:** React for building the user interface.

7.6 Timeline:

- **Phase 1: Requirements Gathering and Design (2 weeks)**
  - Gather detailed requirements from stakeholders (parents, school staff, administrators).
  - Design the system architecture, database schema, and user interface.
- **Phase 2: Development and Testing (6-8 weeks)**
  - Develop the core functionality (user accounts, payments, reporting).
  - Implement integration with existing school systems.
  - Conduct thorough testing to ensure functionality and security.
- **Phase 3: Deployment and Training (1-2 weeks)**
  - Deploy the system to a production environment.
  - Provide training to school staff and parents on how to use the new system.
- **Phase 4: Ongoing Maintenance and Support**
  - Monitor system performance and address any issues.
  - Provide ongoing support to users.
  - Implement enhancements based on feedback and evolving needs.

7.7 Deliverables: Detailed mock-ups, functional requirements document, system prototype, user feedback, final product, and user training materials.

8. References