Subscription Management System
CIS 344 – Group Project Report

By: Austin Iglesias

Instructor: Yanilda Peralta Ramos
Institution: Lehman College
Course Section: CIS 344
Semester: Fall 2025
GitHub Repository:
https://github.com/austinI8911/subscription-management-system

Introduction

The Subscription Management System is a database-driven web application built with PHP and MySQL. Its purpose is to help users or administrators keep track of active subscriptions, payment records, and any changes made over time. The system allows for adding, updating, canceling, and auditing subscriptions, while maintaining a record of payments associated with each user plan.

This project was designed to demonstrate practical skills in relational-database design, secure web programming, and full-stack integration using HTML, PHP, and MySQL. By completing this project, the "team" gained experience in connecting backend databases to dynamic web pages, enforcing data integrity through foreign keys, and applying CRUD operations in a secure, structured environment.

Objectives

The main objectives of the Subscription Management System project are:

1. **Database Integration:** Design and implement a normalized MySQL database with multiple related tables that use foreign keys and joins.

2. **Functionality:** Build working PHP pages to create, read, update, and delete (or cancel) subscription records.

3. **Payment Tracking:** Automatically record payments whenever a new subscription is added and allow manual entry of future payments.

4. **Audit Logging:** Maintain a clear history of subscription activity—creation, updates, and cancellations—in an audit table.

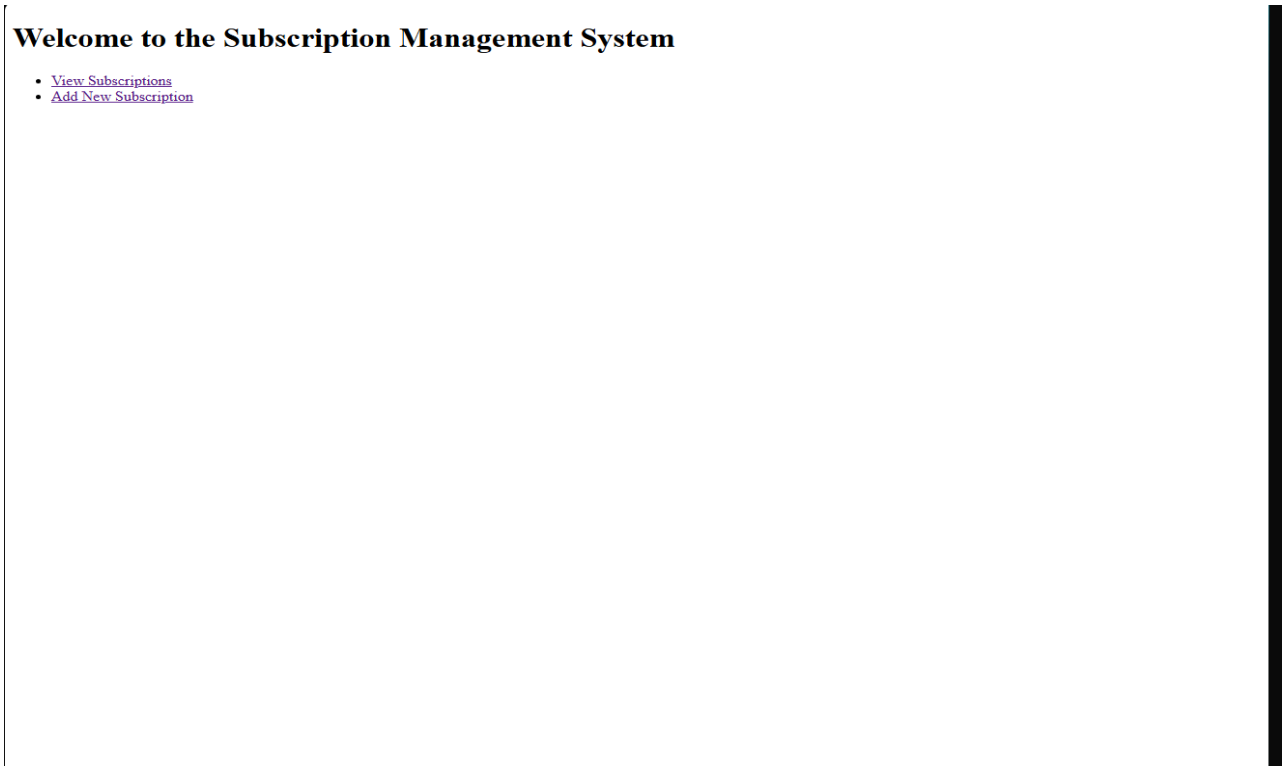
5. **Data Security:** Use prepared statements to prevent SQL-injection attacks and preserve data accuracy.

6. **User Interface:** Provide a clean, easy-to-navigate interface for viewing and managing all information.

7. **Documentation:** Produce a detailed report, EER diagram, and SQL export so the system can be reproduced and understood by others.

System Features

The Subscription Management System provides a collection of integrated features that work together to manage users, plans, subscriptions, payments, and activity tracking. Each feature was designed to demonstrate both technical understanding and functional usability.

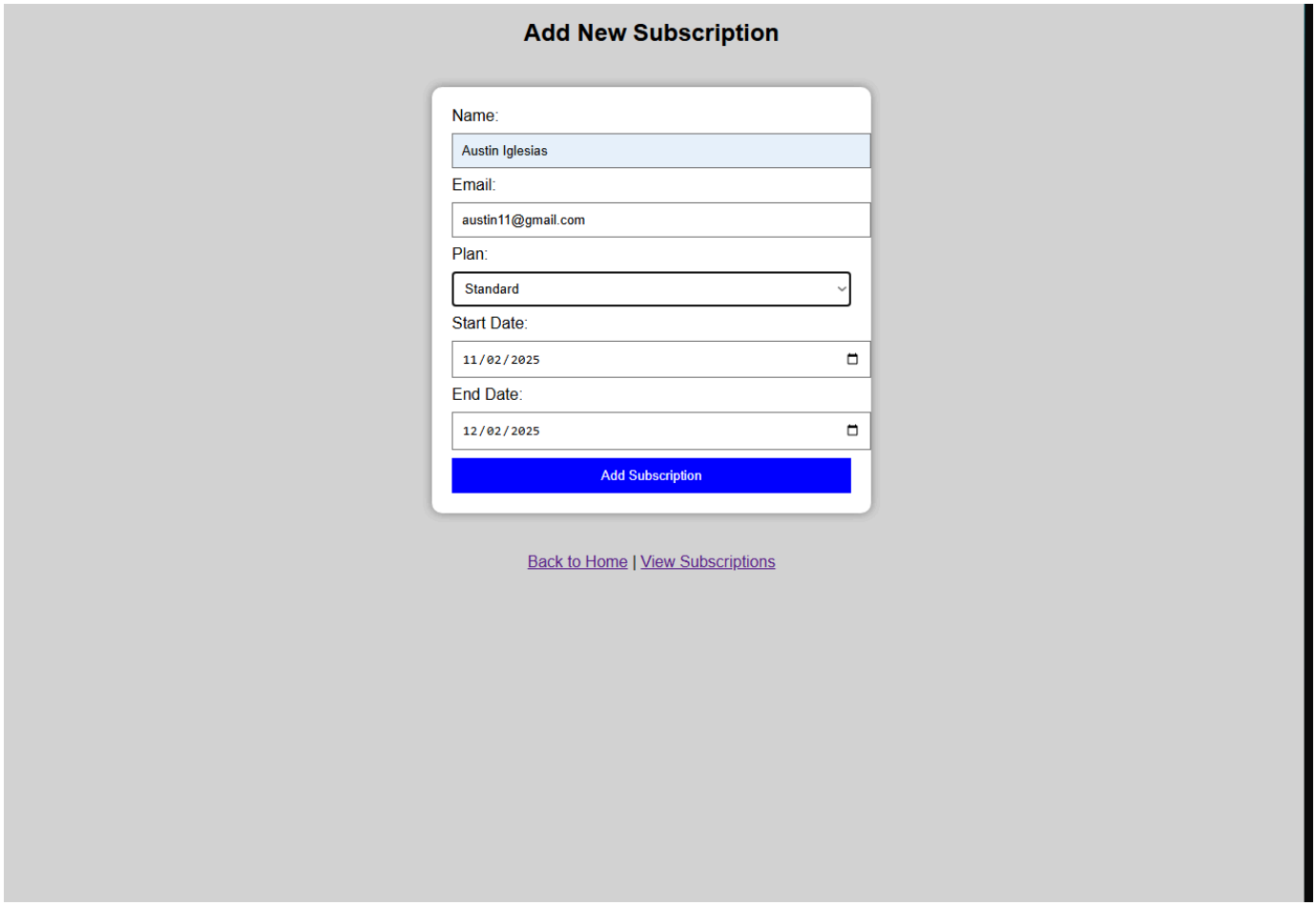


1. Add Subscription

Users can create new subscriptions by selecting an existing user and a subscription plan (Basic, Standard, or Premium). When a new subscription is added:

- The system automatically links the user to a plan.
- A payment record is created automatically based on the plan's price.
- A log entry is recorded in the audit table showing that a new subscription was created.

This feature demonstrates the **Create** operation of CRUD and uses prepared statements to securely insert data into multiple tables.

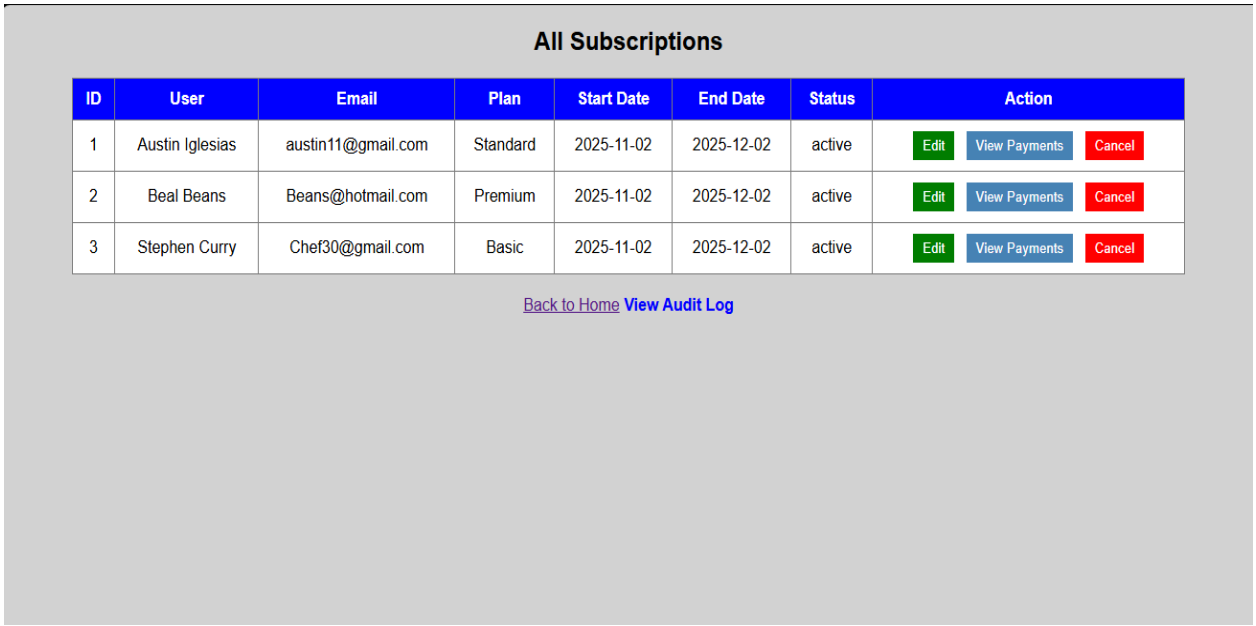


2. View Subscriptions

The subscriptions page displays all active and canceled subscriptions. Data is retrieved using **SQL JOINs** across the **users, plans,** and **subscriptions** tables.
Each row includes:

- User information
- Plan name
- Start and end dates
- Status (active, expired, or canceled)
- Action buttons for **Edit**, **View Payments**, and **Cancel**

This feature shows how PHP can dynamically generate tables using joined query results.

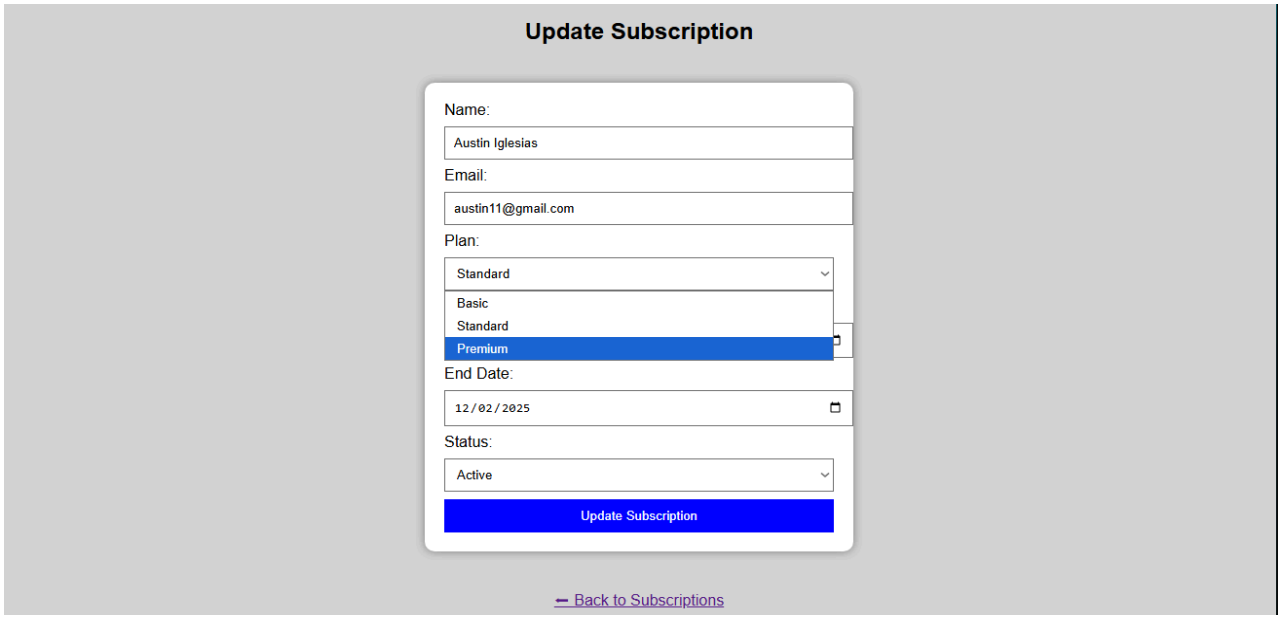


3 Update/Edit Subscription

The update page allows changes to an existing subscription, such as changing the plan, adjusting start or end dates, or modifying its status.
When updates are made:

- Changes are written back to the database using secure prepared statements.
- A record is added to the audit table marking the subscription as “updated.”

This demonstrates the **Update** portion of CRUD functionality.

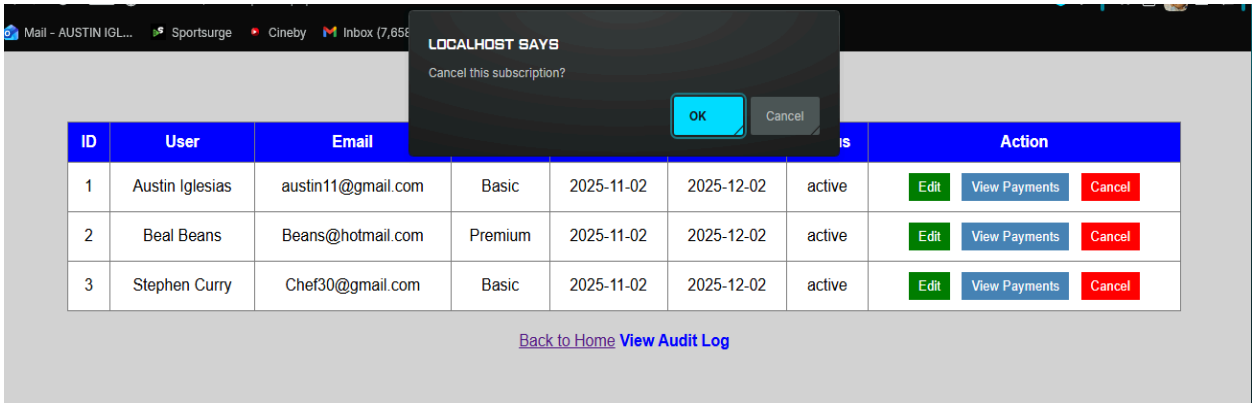


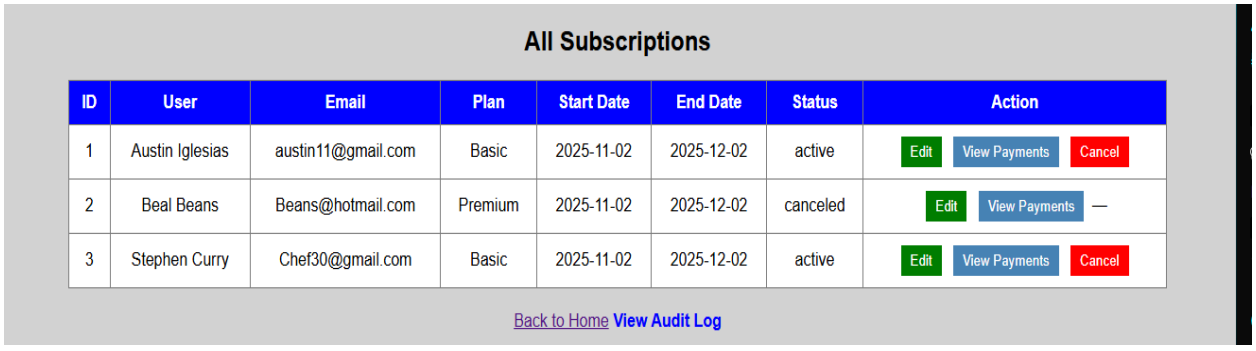
4. Cancel Subscription

Users can cancel subscriptions through the main subscription list.
When a cancel action is triggered:

- The system changes the **status** field to **canceled.**
- An audit entry is recorded to preserve the cancellation event.
- A confirmation message is displayed to the user.

This maintains data integrity without deleting records, which supports proper history tracking.



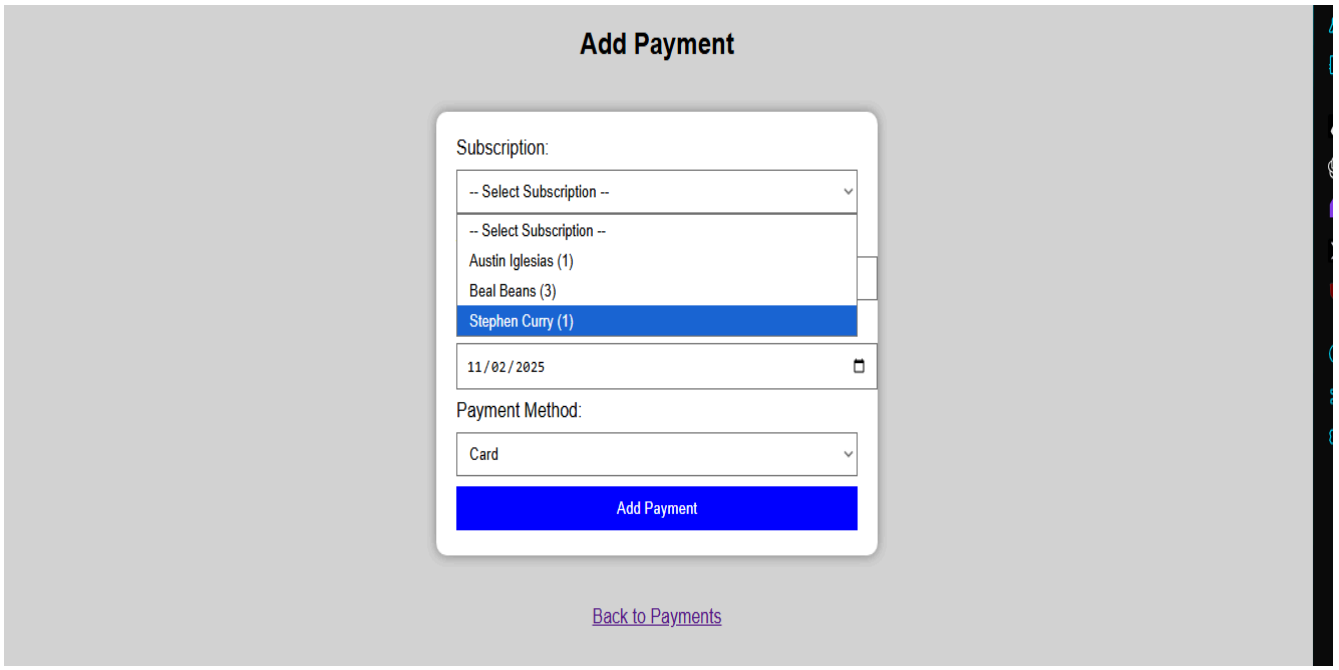


5. Add Payment

Payments can be manually added through a dedicated form.
The form includes:

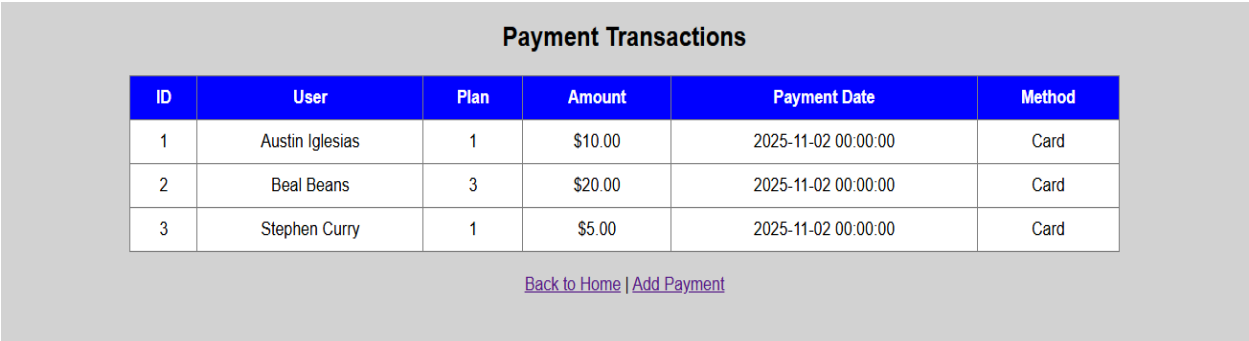
- Dropdown selection of existing subscriptions
- Payment amount, date, and method (Card, Cash, or PayPal)

Each payment record is linked to a specific subscription using a **foreign key**. This ensures traceability between subscriptions and their transactions.



6. View Payments

The payments page displays all recorded payments using a joined query across the **payments, subscriptions, plans,** and **users** tables.
There is also a **View Payments** button under each subscription that filters and displays only that user's transactions.



Payment Transactions

ID	User	Plan	Amount	Payment Date	Method
1	Austin Iglesias	1	$10.00	2025-11-02 00:00:00	Card
2	Beal Beans	3	$20.00	2025-11-02 00:00:00	Card
3	Stephen Curry	1	$5.00	2025-11-02 00:00:00	Card

Back to Home | Add Payment

7. Audit Log

The audit log automatically records every major subscription action (create, update, cancel).
This log stores:

- The subscription ID
- Action type
- Timestamp
- Details about what changed

The audit log provides accountability and transparency in how records are modified, fulfilling the requirement to show historical data tracking.

Subscription Audit Log

Audit ID	User	Plan	Action	Date	Notes
10	Beal Beans	3	Canceled	2025-11-02 13:48:15	Subscription was canceled by user
9	Austin Iglesias	1	Updated	2025-11-02 13:44:14	Subscription details modified
8	Stephen Curry	1	Created	2025-11-02 13:30:34	Subscription created with plan ID 1
7	Beal Beans	3	Created	2025-11-02 13:30:12	Subscription created with plan ID 3
6	Austin Iglesias	1	Created	2025-11-02 13:28:46	Subscription created with plan ID 2
5	Stephen Curry	1	Created	2025-11-01 22:51:54	Subscription created with plan ID 2
4	Austin Iglesias	1	Canceled	2025-11-01 21:45:33	Subscription was canceled by user
3	Beal Beans	3	Canceled	2025-11-01 21:45:26	Subscription was canceled by user
2	Austin Iglesias	1	Updated	2025-11-01 21:11:25	Subscription details modified
1	Beal Beans	3	Created	2025-11-01 21:09:44	Subscription created with plan ID 1

Back to Home

Technical Implementation

The Subscription Management System was implemented using a combination of PHP, MySQL, and HTML, all running locally within the XAMPP environment. This section explains how the backend, frontend, and database components work together to deliver the full functionality of the project.

Development Environment

- **Software Used:** XAMPP (Apache, PHP, MySQL)
- **Database Management:** MySQL Workbench
- **Programming Languages:** PHP, HTML
- **Operating System:** Windows 10
- **Web Browser:** Google Chrome

The XAMPP package provided an integrated environment for both the web server (Apache) and database server (MySQL), allowing the application to run locally on http://localhost/Subscription_System_php/.

Backend (Server-Side)

The backend was built entirely in PHP. Each page performs specific CRUD (Create, Read, Update, Delete/Cancel) operations connected to the MySQL database.

Key Backend Operations:

- **Database Connection:** Each PHP file creates a new connection to the MySQL database using the mysqli extension.
- **Data Handling:** All insert, update, and delete operations are executed using prepared statements to prevent SQL injection and ensure data integrity.
- **Automatic Payments:** When a new subscription is created, a corresponding payment is automatically added to the payments table based on the selected plan's price.
- **Audit Logging:** Whenever a subscription is created, updated, or canceled, an entry is automatically added to the subscription_audit table with the action details and timestamp.

This setup ensures that the application handles all interactions securely and maintains accurate data relationships.

Frontend (Client-Side)

The frontend was created using HTML for simplicity and clarity. The main design goals were usability and straightforward navigation.

Key Frontend Components:

- **Navigation Links:** Buttons on the home page link to all functional pages (Add Subscription, View Subscriptions, Add Payment, View Payments, Audit Log).
- **Forms:** Each form is built using standard HTML input types with clear labels and required validation attributes.
- **Tables:** All display pages use HTML tables with alternating colors and readable headers to display query results.

Security Measures

- **Prepared Statements:** Every SQL query uses prepare() and bind_param() methods to prevent injection.
- **Validation:** Basic validation ensures that empty or invalid fields cannot be submitted.
- **Restricted Input:** Users cannot directly alter URL parameters to affect unintended records due to query constraints.
- **Safe Updates:** The system updates records without deleting them, preserving historical data and audit consistency.

Application Flow

1. **User opens the home page:** They can choose to add a subscription, view records, or manage payments.
2. **Add Subscription:** A form collects user and plan information; the system inserts a new record into subscriptions, adds an entry in payments, and records the event in subscription_audit.
3. **View and Manage Subscriptions:** Subscriptions are displayed using joined queries between users, plans, and subscriptions. Users can update or cancel directly from the table.
4. **Payments and Audit Logs:** The user can view payment history and the full activity log from any page. All tables dynamically pull data using JOINs to show full contextual information.

Database Design

The Subscription Management System database was built in MySQL and follows a relational model to ensure efficiency, consistency, and data integrity. The database contains five primary tables: users, plans, subscriptions, payments, and subscription_audit.

Each table is connected through foreign keys, and relationships were defined using referential constraints. The schema is designed to demonstrate SQL JOINs, foreign keys, and transactional consistency between records.

Tables Overview

1. Users Table

- **Primary Key:** user_id
- **Key Columns:** name, email
- **Description:** Stores the personal details of users who have subscriptions. Each user can have multiple subscriptions linked to their ID.

2. Plans Table

- **Primary Key:** plan_id
- **Key Columns:** plan_name, price
- **Description:** Contains all available subscription plans. The system currently includes three options: Basic, Standard, and Premium.

3. Subscriptions Table

- **Primary Key:** subscription_id
- **Key Columns:** user_id, plan_id, status, start_date, end_date

- **Description:** Connects users to their chosen plans. It tracks whether each subscription is active, expired, or canceled. This table forms the center of the database structure, linking users, plans, payments, and audit records.

4. Payments Table

- **Primary Key:** payment_id
- **Key Columns:** subscription_id, amount, method, payment_date
- **Description:** Records all payments associated with subscriptions. Each payment is tied to a specific subscription and includes the payment method, amount, and date.

5. Subscription_Audit Table

- **Primary Key:** audit_id
- **Key Columns:** subscription_id, action, action_date, details
- **Description:** Logs every important subscription event, including when a subscription is created, updated, or canceled. This table helps maintain a historical record of all activities for accountability and transparency.

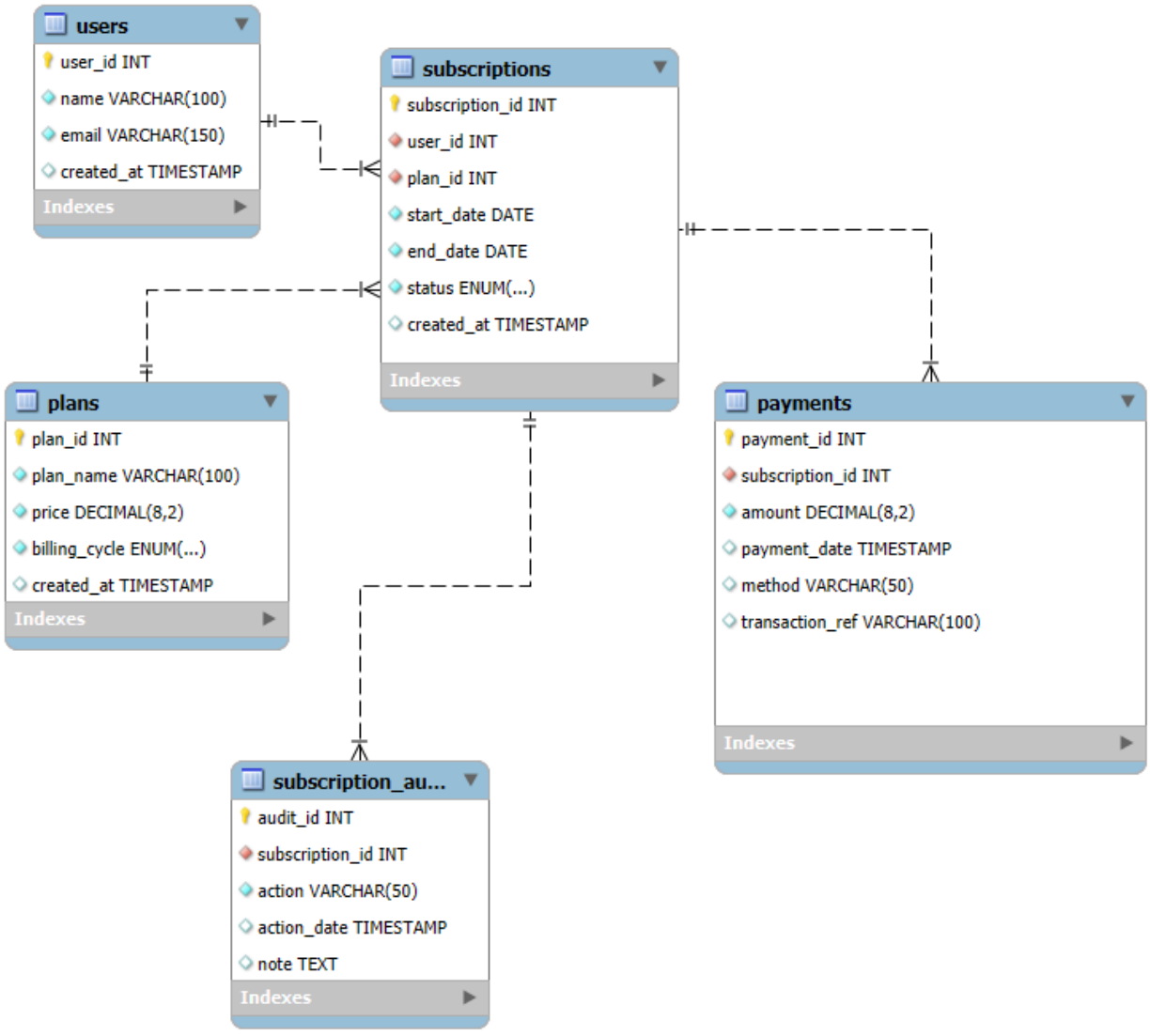
Relationships

The key relationships between tables are:

- **users → subscriptions:** One user can have many subscriptions.
- **plans → subscriptions:** Each subscription belongs to one plan.
- **subscriptions → payments:** A subscription can have multiple payments.
- **subscriptions → subscription_audit:** Each subscription can have multiple audit entries.

These relationships create a **one-to-many** structure that ensures normalized data storage and prevents duplication.

EER Diagram and SQL Database Structure



Subscription_serviceDB

Limit to 1000 rows

```
4
5
6     -- Users table
7   CREATE TABLE users (
8     user_id INT AUTO_INCREMENT PRIMARY KEY,
9     name VARCHAR(100) NOT NULL,
10    email VARCHAR(150) NOT NULL UNIQUE,
11    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
12  );
13
14    --  Plans table
15  CREATE TABLE plans (
16    plan_id INT AUTO_INCREMENT PRIMARY KEY,
17    plan_name VARCHAR(100) NOT NULL,
18    price DECIMAL(8,2) NOT NULL,
19    billing_cycle ENUM('monthly','yearly') NOT NULL DEFAULT 'monthly',
20    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
21  );
22
23    -- Plan names
24  INSERT INTO plans (plan_name, price, billing_cycle) VALUES
25    ('Basic', 5.00, 'monthly'),
26    ('Standard', 10.00, 'monthly'),
27    ('Premium', 20.00, 'monthly');
28
29
30    --  Subscriptions table
31  CREATE TABLE subscriptions (
32    subscription_id INT AUTO_INCREMENT PRIMARY KEY,
33    user_id INT NOT NULL,
34    plan_id INT NOT NULL,
35    start_date DATE NOT NULL,
36    end_date DATE NOT NULL,
37    status ENUM('active','expired','canceled') NOT NULL DEFAULT 'active',
38    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
39    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE,
40    FOREIGN KEY (plan_id) REFERENCES plans(plan_id) ON DELETE RESTRICT,
41    INDEX idx_user_status (user_id, status),
42    INDEX idx_end_date (end_date)
43  );
```

```sql
--  Payments table
CREATE TABLE payments (
  payment_id INT AUTO_INCREMENT PRIMARY KEY,
  subscription_id INT NOT NULL,
  amount DECIMAL(8,2) NOT NULL,
  payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  method VARCHAR(50),
  transaction_ref VARCHAR(100),
  FOREIGN KEY (subscription_id) REFERENCES subscriptions(subscription_id) ON DELETE CASCADE,
  INDEX idx_payment_subscription (subscription_id)
);

-- Audit table for subscription changes
CREATE TABLE subscription_audit (
  audit_id INT AUTO_INCREMENT PRIMARY KEY,
  subscription_id INT NOT NULL,
  action VARCHAR(50) NOT NULL, -- e.g., 'created', 'updated', 'canceled'
  action_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  note TEXT,
  FOREIGN KEY (subscription_id) REFERENCES subscriptions(subscription_id) ON DELETE CASCADE
);
```

Conclusion

The Subscription Management System project successfully demonstrates the practical integration of web development and database management concepts. By combining PHP, MySQL, and HTML, the system provides an efficient way to manage user subscriptions, track payments, and maintain a detailed audit trail of all activities. Throughout the development process, the “team” applied key programming and database principles, including foreign keys, SQL joins, and prepared statements for secure data handling. The project illustrates how relational databases can power dynamic web applications, and how user-friendly interfaces can interact seamlessly with backend logic. Testing confirmed that all major features, adding, updating, canceling, and auditing subscriptions function correctly, with automatic updates across all related tables. The audit log ensures full transparency of user actions, while the system’s design makes it easy to maintain and expand in the future. Overall, this project achieved all objectives outlined in the course guidelines. It not only strengthened technical skills in web programming and database design but also provided valuable hands-on experience in debugging, and system documentation.