

**Software Architecture and Design LAB**

**Project Report**

|  |  |
| --- | --- |
| Group Members: | Muhammad Safiullah Qazi |
|  | Muhammad Ibrahim Shaikh  Ameer Hamza Laghari |
| Program: | BSSE |
| Section: | 4-A |

**Lab Instructor:**Danish Shahzad

**Project Title: University Admin Dashboard**

**Project Description:**

A responsive web-based admin dashboard built using **HTML, CSS, JavaScript (Chart.js)** for the frontend and **PHP with MySQL** for the backend. The panel allows an admin to view dynamic data about students, teachers, employees, and earnings.

**Note:** All data is entered **manually using phpMyAdmin** into a **MySQL database** running on **XAMPP**.

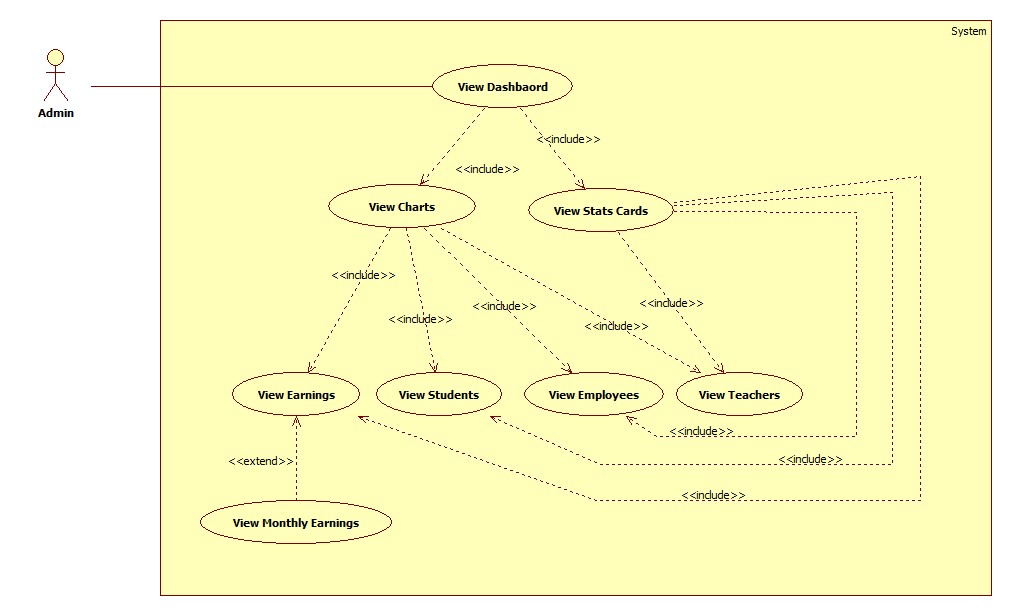
**1. Use Case Diagram**

**Actors:**

* Admin (Primary User)

**Use Cases:**

* View Dashboard
* View Stats Cards
  + <> View Students
  + <> View Teachers
  + <> View Employees
  + <> View Earnings
* View Charts
  + <> View Earnings
  + <> View Monthly Earnings
  + <> View Students
  + <> View Teachers
  + <> View Employees



**Explanation:**

* Admin accesses the dashboard.
* Dashboard provides separate metrics and chart views.
* Monthly earnings chart is shown when user interacts with the line chart.

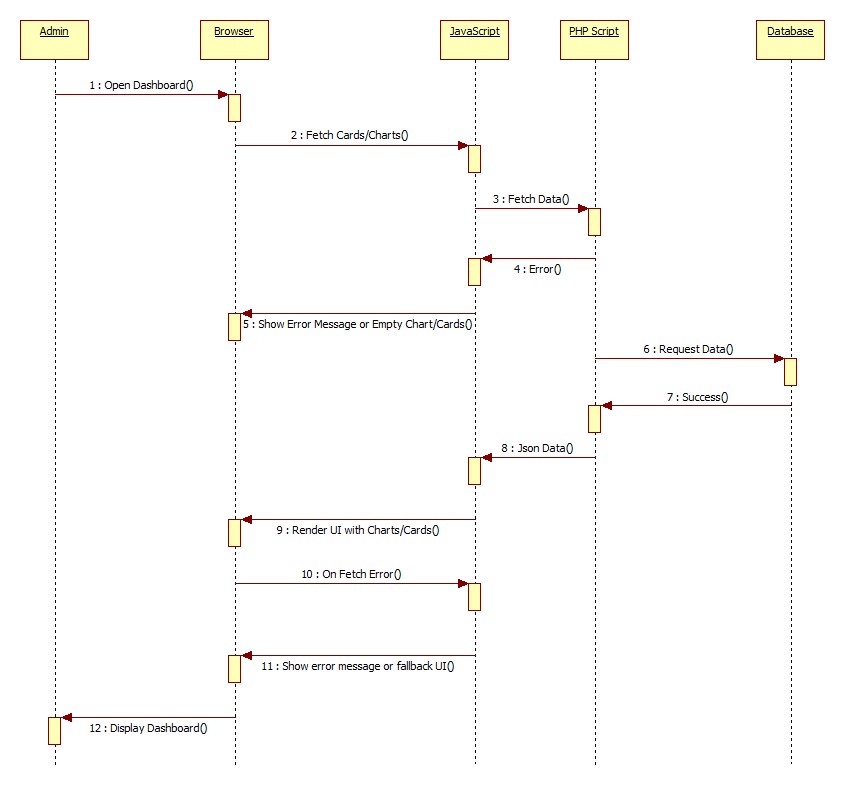
**2. Sequence Diagram**

**Scenario: Admin Opens Dashboard**

**Objects:** Admin → Browser → JS Scripts → PHP Scripts → MySQL DB

**Sequence Flow:**

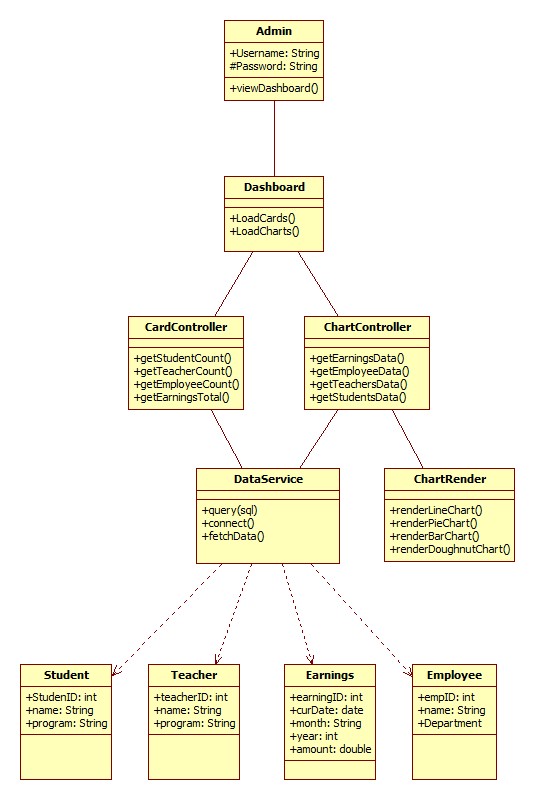
1. Admin opens dashboard (index.html).
2. JS (cards.js) sends fetch request to cards.php.
3. cards.php queries MySQL and returns JSON.
4. Data is inserted into dashboard cards.
5. Each chart script (chart1.js to chart4.js) fetches data via PHP.
6. PHP scripts query grouped data (e.g., by month, department).
7. Chart.js renders data into the dashboard.
8. If any PHP script fails:
   * JS catches error and logs it.
   * Chart does not render (failure handling).



**Explanation:**

* Emphasizes real-time dynamic data flow.
* Shows how frontend and backend interact via fetch API.

**3. Class Diagram**

****

**Explanation:**

* Clear layered architecture: UI → Controller → Logic → DB
* Admin only views data; backend handles DB queries
* JavaScript handles rendering logic via Chart.js

**4. Technologies Used:**

| **Technology** | **Purpose** |
| --- | --- |
| HTML/CSS | Frontend layout and styling |
| JavaScript | Dynamic data fetching & Chart.js |
| Chart.js | Data visualization (charts) |
| PHP | Backend scripts & DB querying |
| MySQL | Relational database |
| XAMPP | Local server environment |

**5. Database Tables:**

* students (s\_id, name, program)
* teachers (t\_id, name, designation)
* employees (emp\_id, name, department)
* earnings (er\_id, cur\_date, month, year, amount)

**6. Features:**

* Responsive Admin Dashboard
* Four dynamic cards:
  + Total Students
  + Total Teachers
  + Total Employees
  + Yearly Earnings
* Four charts (Chart.js):
  + Line chart: Monthly earnings
  + Pie chart: Students by program
  + Bar chart: Teachers by designation
  + Doughnut chart: Employees by department
* Uses Fetch API and PHP to fetch data in real-time

**7. Future Enhancements:**

* Add secure Admin Login system
* Add user-friendly data input UI (instead of manual entry)
* Export data to Excel/PDF
* Search and filter analytics
* Add CRUD functionality for all entities

**Conclusion:**

The University Admin Dashboard project is a lightweight dashboard application that visually tracks important university metrics using real-time data from MySQL. Designed using modern web tools and structured backend logic, the system is scalable and ready to evolve into a full ERP system.