GALGOTIAS UNIVERSITY

NAAC GRADE A+ Accredited University

Plot No. 2, Sector 17-A Yamuna Expressway, Greater Noida, Gautam Buddha Nagar, Uttar Pradesh, India.

Campus Sync

CAMPUS SYNC CONNECTS STUDENTS, FACULTY, & CAMPUS SERVICES IN ONE PLATFORM

"A Java-Based Web Application with JDBC, Servlets & MySQL Integration"

COURSE CODE: (R1UC304C)

B.TECH. THIRD SEM (2025 -2026)

AMAN PATEL – aman.24scse1260003@galgotiasuniversity.ac.in

SHOBHIT TIWARI – shobhit.24scse1290006@galgotiasuniversity.ac.in

ALOK SHAW – alok.24scse1290023@galgotiasuniversity.ac.in

Mentor: Prof. Dr. Jitender Tanwar

Java MySQL Apache Tomcat

index

PROJECT OVERVIEW

CampusSync is a web-based college management system developed using Java Servlets, JSP, and JDBC. The purpose of this platform is to simplify and automate the essential academic and administrative processes within an educational institution. The system provides three major user roles—Admin, Faculty, and Student—each with dedicated features designed to enhance efficiency and reduce manual workload. Administrators can manage students, faculty members, and publish official notices. Faculty members can mark attendance, upload assignments, and enter academic marks. Students can log in to view their attendance records, assignment updates, and marks in a centralized manner. CampusSync uses a structured MVC architecture, where Servlets serve as controllers, JSP pages act as the view layer, and DAO classes handle database interactions, ensuring a clean, scalable, and efficient codebase.

THE SYSTEM SUPPORTS:

- Student attendance tracking
- Marks management
- Assignment upload & distribution
- Notice publishing
- Student & faculty management
- Centralized authentication system

❖ Tech Stack Used:

Java • JSP • Servlets • JDBC • MySQL • Tomcat • Maven

PROBLEM AND SOLUTION

● PROBLEM UNDERSTANDING

Most colleges still rely on manual or semi-digital systems for managing attendance, marks, assignments, and communication. This leads to delays, errors, and inefficiency.

● PROPOSED SOLUTION

CampusSync processes with a web-driven architecture.

The solution ensures:

- Faster data access
- Automated workflows
- Accurate records
- Secure login-based operations
- Centralized database for all academic data

● SOLUTION DESIGN SUMMARY

1. Admin Module – manages students, faculty, notices.
2. Faculty Module – attendance, marks, assignments.
3. Student Module – view marks, attendance, and notices.
4. MySQL Database – stores all system data.
5. Servlet Controllers – manage request-response cycles.
6. DAO Layer – handles all database interactions.
7. JSP Views – present the UI.

CORE JAVA CONCEPTS USED

This project strongly demonstrates Java fundamentals:

✔ **Object-Oriented Programming**

- **Encapsulation:** All models (Student, Faculty, Admin, Assignment) are POJOs.
- **Abstraction:** DAO classes hide JDBC logic from servlets.
- **Polymorphism:** Servlet doGet/doPost overriding.
- **Constructor Overloading** in model classes.

✔ **Exception Handling**

- try/catch
- throws keyword
- ServletException handling for DB and IO exceptions

✔ **Collections Framework**

- **List<Student>, List<Faculty>** used in DAO layer

✔ **Clean, Modular Code**

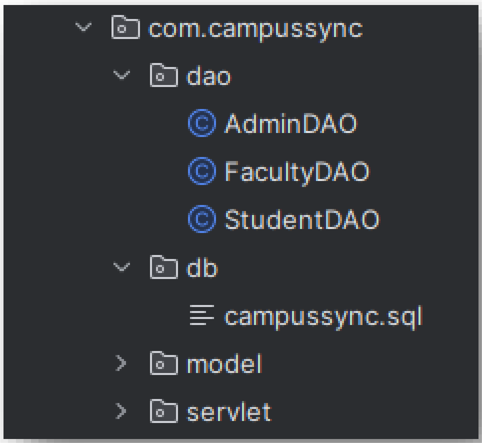
- Separate packages:
  - com.campussync.servlet
  - com.campussync.dao
  - com.campussync.model
  - com.campussync.util

✓ JDBC + Java Integration

- PreparedStatement
- ResultSet
- Connection pooling concept (explained in documentation).

CampusSync demonstrates strong use of core Java concepts. The project makes extensive use of **Object-Oriented Programming**, including encapsulation through POJO model classes such as Student, Faculty, and Admin. Each model class securely stores user-related attributes with getter and setter methods. Abstraction is implemented via DAO (Data Access Object) classes, which hide JDBC logic from the Servlets, ensuring separation of concerns and cleaner code. Method overriding is used in Servlets, especially through the `doGet()` and `doPost()` methods, highlighting polymorphism.

Exception handling is consistently applied throughout the system using try–catch blocks and `throws` declarations, especially in database operations where SQL Exceptions must be safely handled. Java Collections like `List<Student>` and `List<Faculty>` are used in DAO classes to store and return structured data. The entire project follows clean, readable coding standards, maintaining modularity through package separation—`servlet`, `dao`, `model`, and `util`.


```
com.campussync
  dao
    AdminDAO
    FacultyDAO
    StudentDAO
  db
    campussync.sql
  model
  servlet
```

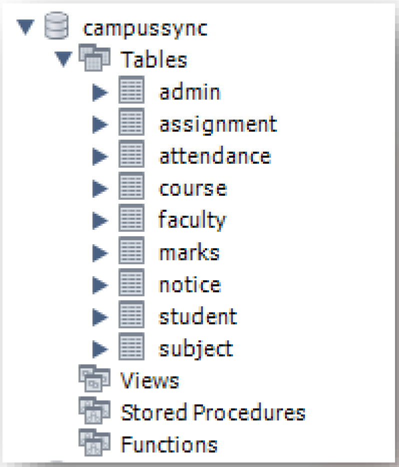
DATABASE INTEGRATION

Database connectivity plays a crucial role in CampusSync. The project uses JDBC to connect the application with a MySQL database. A dedicated DBConnection class is responsible for loading the MySQL driver and establishing connections, ensuring code reusability and centralized configuration. PreparedStatement is used across all DAO classes, preventing SQL injection and improving database security. This also enhances performance by precompiling SQL queries.

"JDBC is used extensively for all operations."

✓ How JDBC Works in This Project:

1. **DBConnection class** loads MySQL driver.
2. **DAO classes** use PreparedStatement for secure queries.
3. **CRUD operations** performed on all tables.
4. **ResultSet** returns data to JSP via servlets.
   - SQL schema includes:
     - student
     - faculty
     - admin
     - subject
     - attendance
     - assignment
     - marks
     - notice



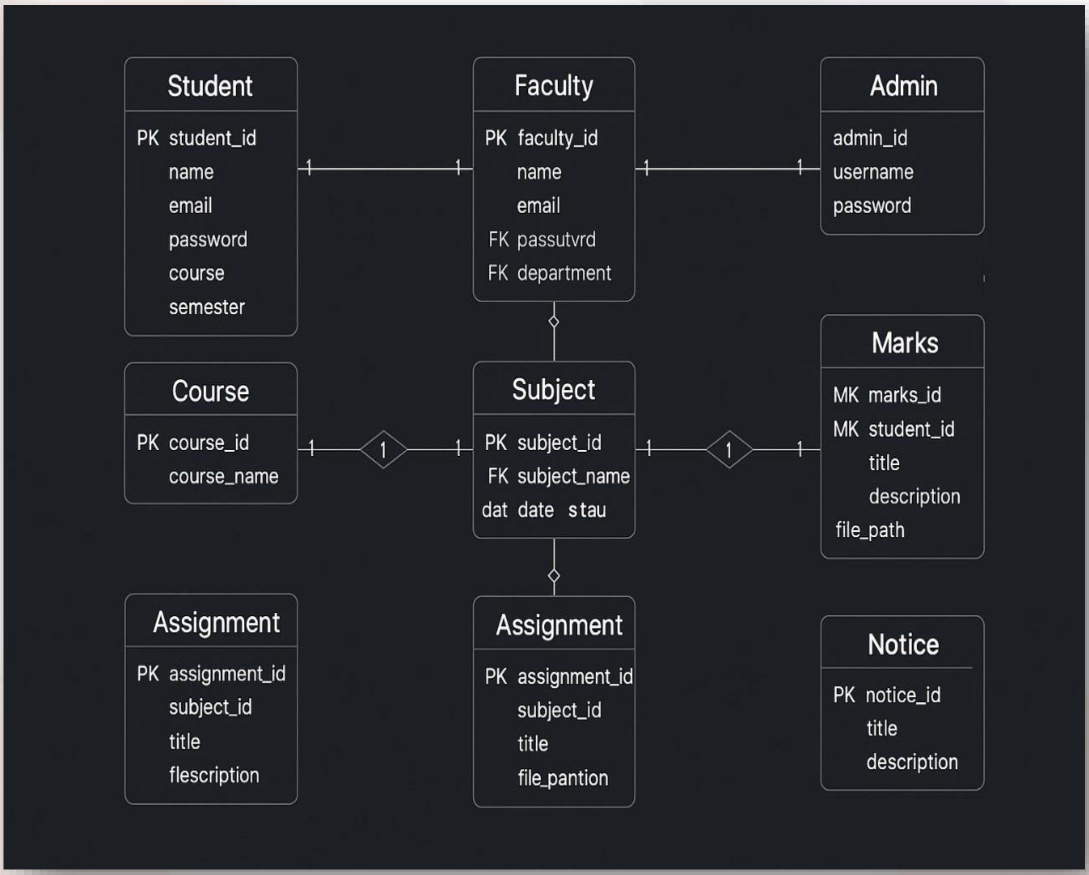
```
▼ campussync
  ▼ Tables
    ▶ admin
    ▶ assignment
    ▶ attendance
    ▶ course
    ▶ faculty
    ▶ marks
    ▶ notice
    ▶ student
    ▶ subject
    Views
    Stored Procedures
    Functions
```

✔ Security

- PreparedStatement prevents SQL Injection.
- try-with-resources ensures no memory leaks.

SCHEMA’S ER DIAGRAM:

- Student ↔ Attendance ↔ Subject
- Faculty ↔ Subject
- Student ↔ Marks
- Admin publishes Notice
- Assignment belongs to Subject



SERVLET and Web

✓ Servlets Implemented

1. AuthServlet

- Single login system for all roles
- Session management
- Logout handler

2. AdminServlet

- Manage Students
- Manage Faculty
- Publish Notices
- Load admin dashboard

3. FacultyServlet

- Mark attendance
- Upload assignments
- Enter marks
- Faculty dashboard

4. StudentServlet

- View attendance
- View marks
- Student dashboard
- Only student can view marks/attendance

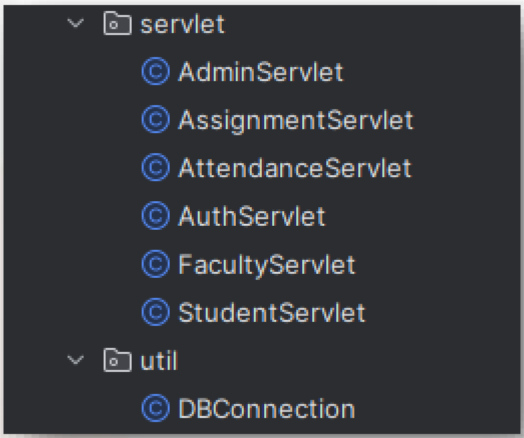
✓ Web Integration

- JSP acts as the View
- Servlets act as Controller
- DAO + DB act as Model

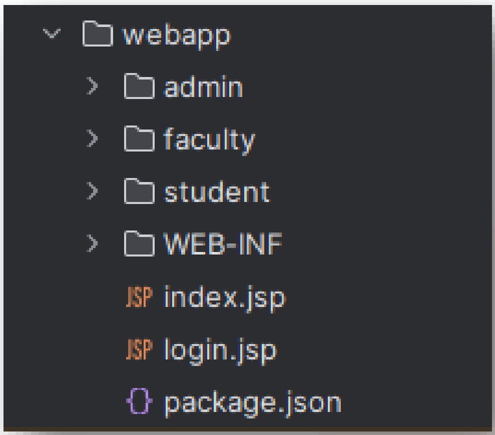
Session validation ensures secure access:

- Only admin can access admin pages
- Only faculty can mark attendance
- Only student can view marks/attendance

CampusSync uses Servlets as the core controllers for handling HTTP requests and coordinating different modules. Each role has its own dedicated servlet. The AuthServlet manages the authentication system for all three roles and securely maintains sessions. The AdminServlet handles administrative operations like adding students, adding faculty, and publishing notices. The FacultyServlet manages all academic operations including attendance marking, assignment uploading, and entering marks. The StudentServlet retrieves attendance and marks records for display on the student dashboard.



```
servlet
    AdminServlet
    AssignmentServlet
    AttendanceServlet
    AuthServlet
    FacultyServlet
    StudentServlet
util
    DBConnection
```

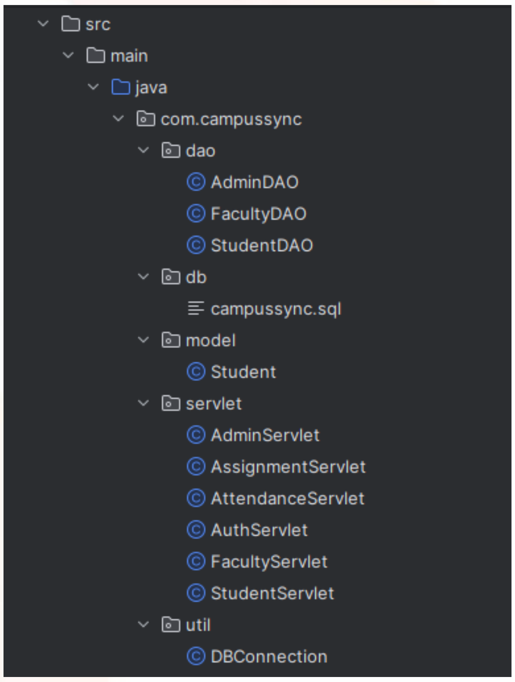


```
webapp
    admin
    faculty
    student
    WEB-INF
    index.jsp
    login.jsp
    package.json
```

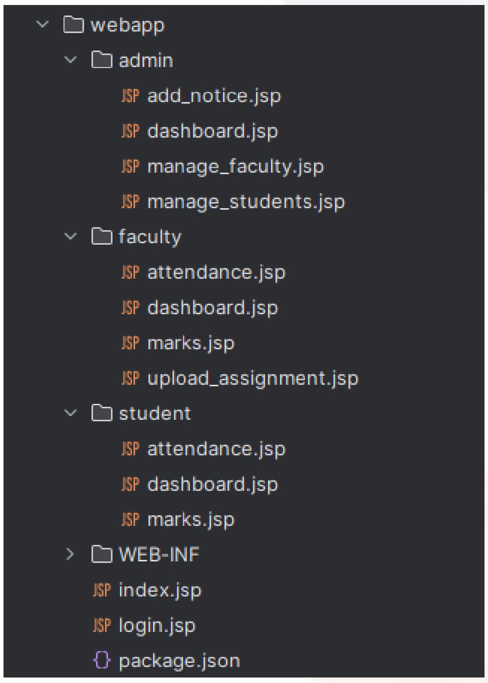
MVC ARCHITECTURE

The architecture of CampusSync follows a multi-layered structure. The presentation layer consists of JSP pages responsible for interacting with users. When a user performs an action, the request is sent to a Servlet, which functions as the controller. The Servlet communicates with DAO classes to execute the necessary database operations. DAO classes use JDBC to interact with the MySQL database. Once the data is processed, it is forwarded back to the JSP page for display. This architecture ensures loose coupling, reusability, and ease of maintenance.

Client → Servlet → DAO → MySQL → JSP response



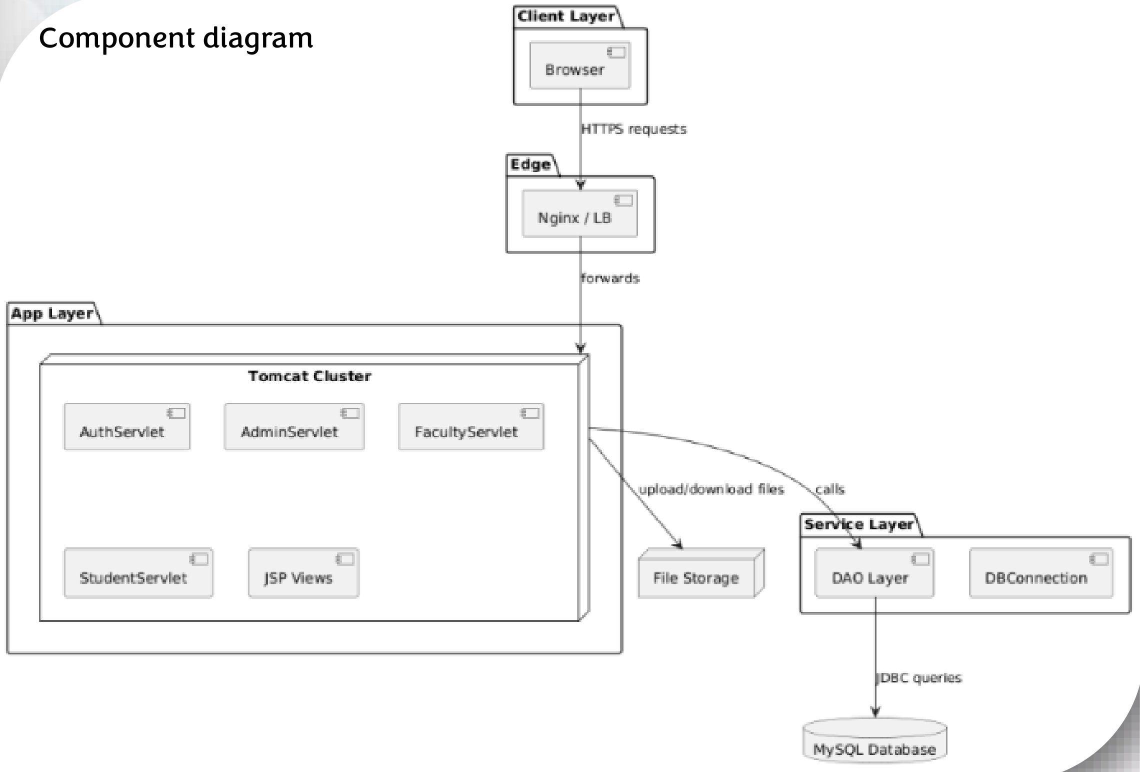
```
src
  main
    java
      com.campussync
        dao
          AdminDAO
          FacultyDAO
          StudentDAO
        db
          campussync.sql
        model
          Student
        servlet
          AdminServlet
          AssignmentServlet
          AttendanceServlet
          AuthServlet
          FacultyServlet
          StudentServlet
        util
          DBConnection
```



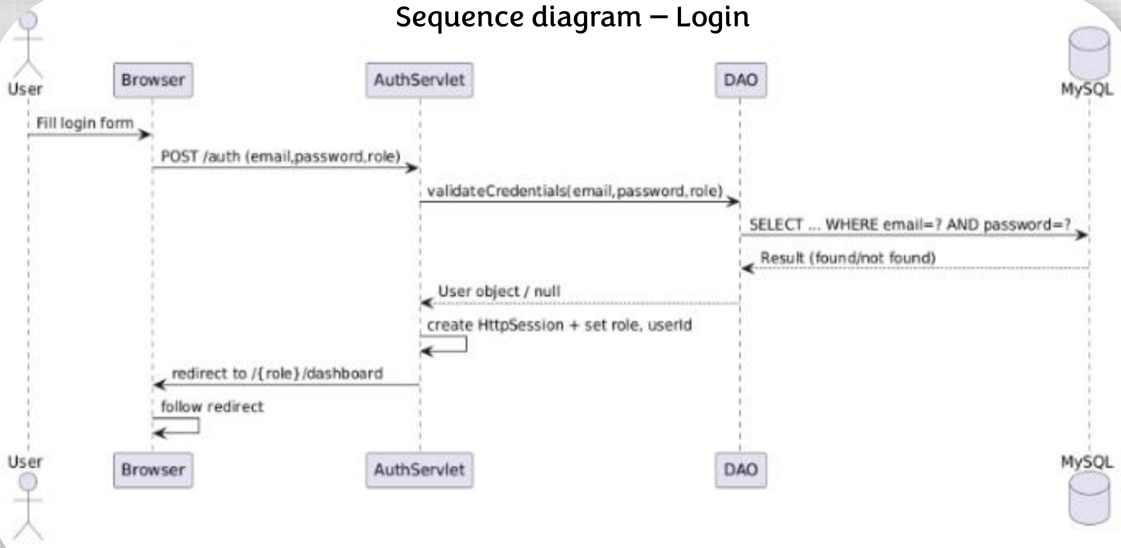
```
webapp
  admin
    add_notice.jsp
    dashboard.jsp
    manage_faculty.jsp
    manage_students.jsp
  faculty
    attendance.jsp
    dashboard.jsp
    marks.jsp
    upload_assignment.jsp
  student
    attendance.jsp
    dashboard.jsp
    marks.jsp
  WEB-INF
  index.jsp
  login.jsp
  package.json
```

SYSTEM ARCHITECTURE DESIGN

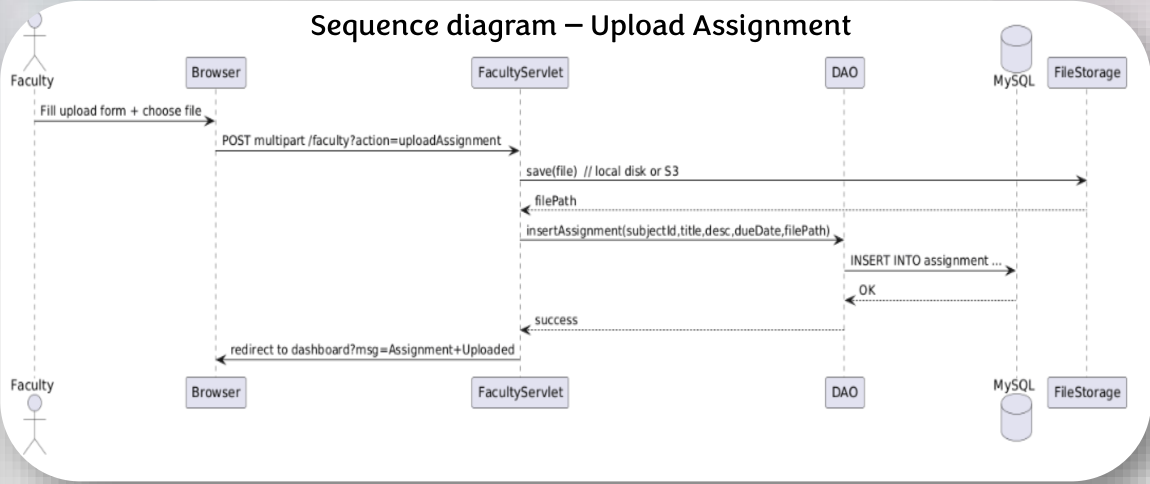
Component diagram

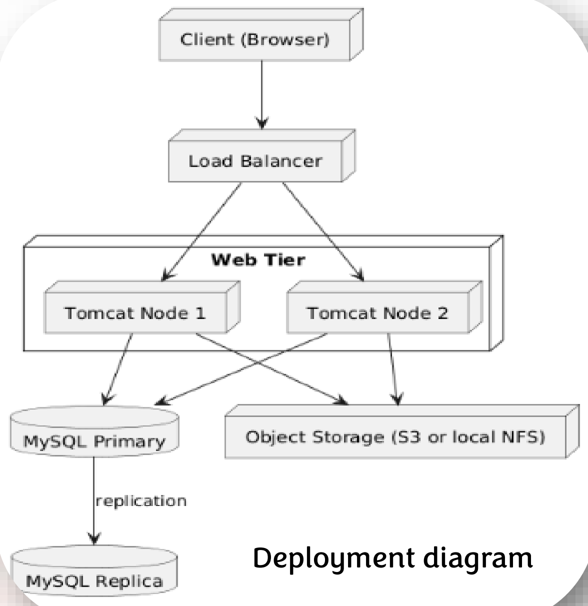


Sequence diagram – Login

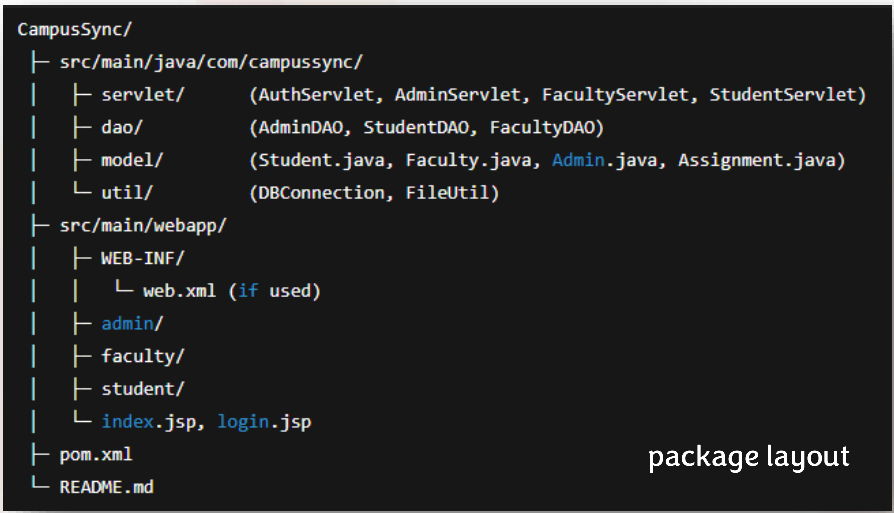

User
Browser
AuthServlet
DAO
MySQL
Fill login form
POST /auth (email,password,role)
validateCredentials(email,password,role)
SELECT ... WHERE email=? AND password=?
Result (found/not found)
User object / null
create HttpSession + set role, userId
redirect to /{role}/dashboard
follow redirect
User
Browser
AuthServlet
DAO
MySQL

Sequence diagram – Upload Assignment


Faculty
Browser
FacultyServlet
DAO
MySQL
FileStorage
Fill upload form + choose file
POST multipart /faculty?action=uploadAssignment
save(file) // local disk or S3
filePath
insertAssignment(subjectId,title,desc,dueDate,filePath)
INSERT INTO assignment ...
OK
success
redirect to dashboard?msg=Assignment+Uploaded
Faculty
Browser
FacultyServlet
DAO
MySQL
FileStorage



Deployment diagram

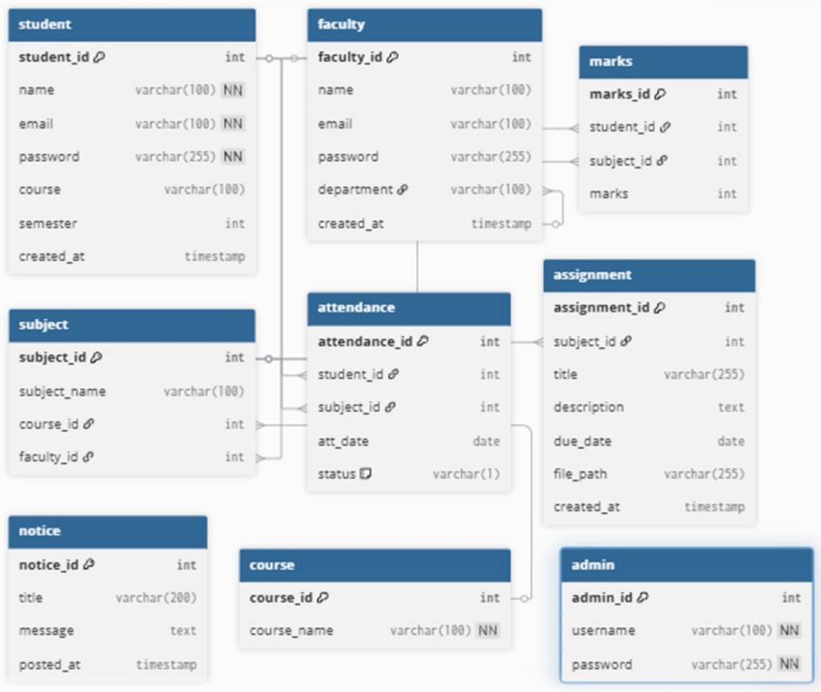


```
CampusSync/
 ├ src/main/java/com/campussync/
 │   ├ servlet/      (AuthServlet, AdminServlet, FacultyServlet, StudentServlet)
 │   ├ dao/          (AdminDAO, StudentDAO, FacultyDAO)
 │   ├ model/        (Student.java, Faculty.java, Admin.java, Assignment.java)
 │   └ util/         (DBConnection, FileUtil)
 ├ src/main/webapp/
 │   ├ WEB-INF/
 │   │   └ web.xml (if used)
 │   ├ admin/
 │   ├ faculty/
 │   ├ student/
 │   └ index.jsp, login.jsp
 ├ pom.xml
 └ README.md
```

package layout

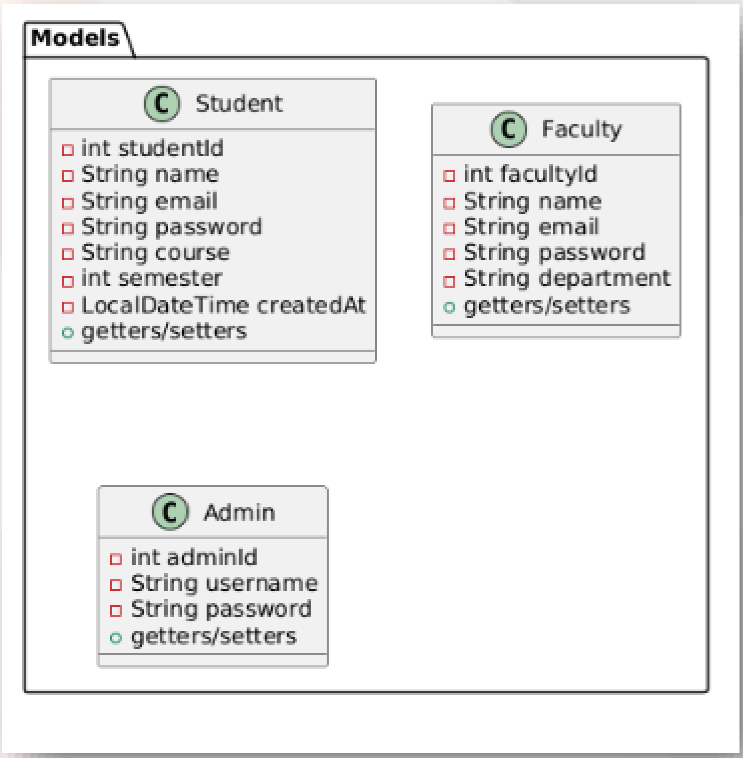
ER DIAGRAM

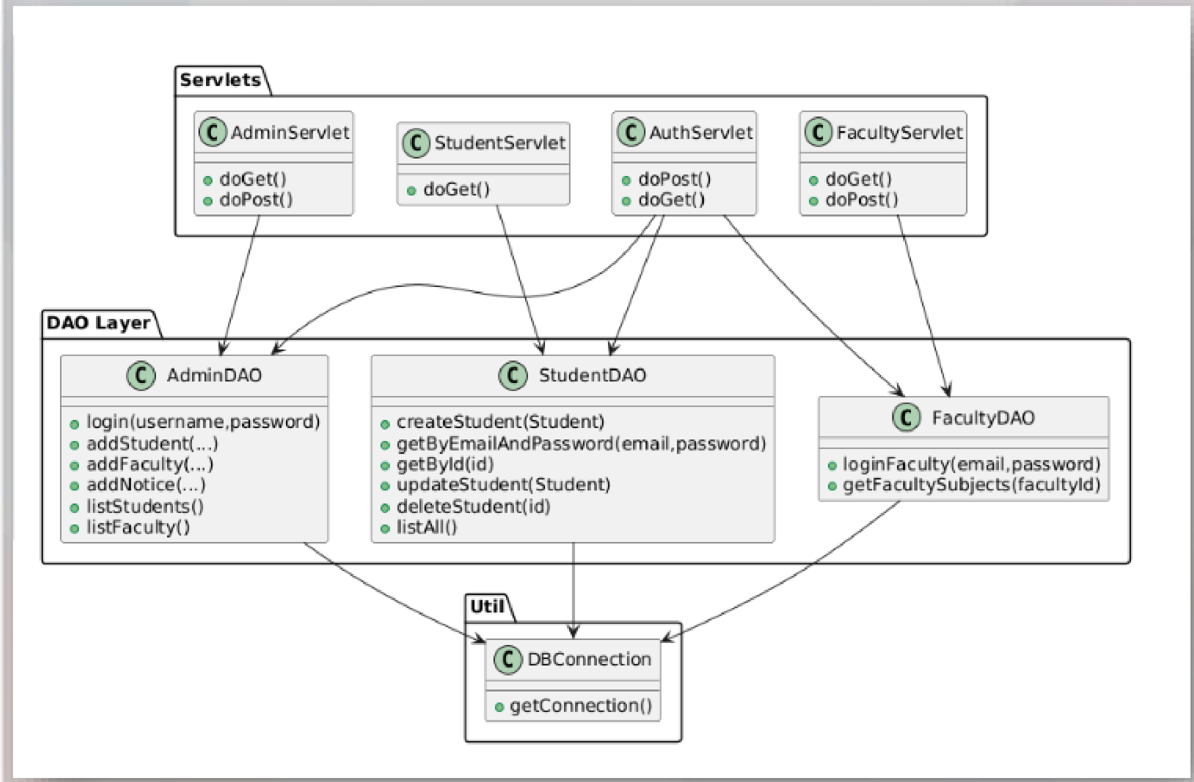
The Entity-Relationship diagram for CampusSync consists of multiple interconnected tables. The student table stores basic student information, the faculty table stores faculty details, and the subject table connects subjects with faculty. Attendance records link students to specific subjects along with dates. Assignment and marks tables store academic submissions and performance metrics. Admins manage notices stored in the notice table. All relationships are designed to ensure data consistency and support smooth academic workflows.



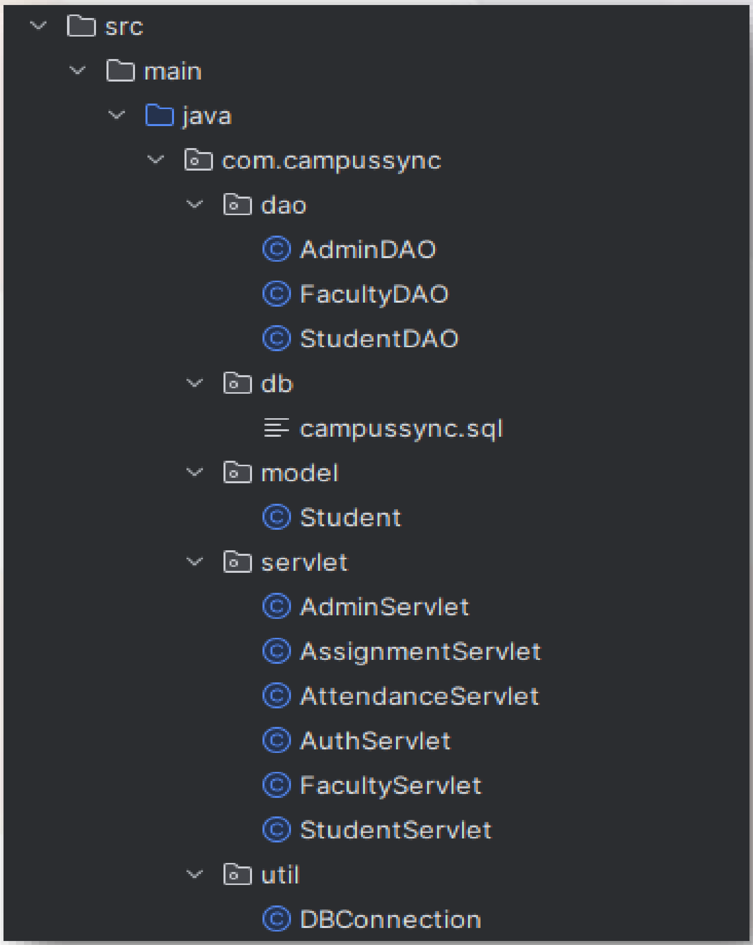
Class Diagram

The class diagram highlights the object-oriented structure of the project. Model classes such as Student, Faculty, and Admin encapsulate user data. DAO classes like StudentDAO, FacultyDAO, and AdminDAO implement business logic and handle all database interactions. Servlets serve as controllers managing user requests. The DBConnection class acts as a utility class ensuring unified database connectivity. This structure demonstrates clean OOP implementation.





Servlets
AdminServlet
doGet()
doPost()
StudentServlet
doGet()
AuthServlet
doPost()
doGet()
FacultyServlet
doGet()
doPost()
DAO Layer
AdminDAO
login(username,password)
addStudent(...)
addFaculty(...)
addNotice(...)
listStudents()
listFaculty()
StudentDAO
createStudent(Student)
getByEmailAndPassword(email,password)
getById(id)
updateStudent(Student)
deleteStudent(id)
listAll()
FacultyDAO
loginFaculty(email,password)
getFacultySubjects(facultyId)
Util
DBConnection
getConnection()



src
main
java
com.campussync
dao
AdminDAO
FacultyDAO
StudentDAO
db
campussync.sql
model
Student
servlet
AdminServlet
AssignmentServlet
AttendanceServlet
AuthServlet
FacultyServlet
StudentServlet
util
DBConnection

SCREENSHOTS

CAMPUSSYNC — CORE CODE SNIPPETS

1. Problem Understanding & Solution Design

CampusSync aims to provide a centralized system for student, faculty, and admin interactions:

- Admin manages students, faculty, subjects, courses.

- Faculty manages attendance, assignments, and marks.

- Students view attendance, marks, and assignments.

HIGH-LEVEL ARCHITECTURE

Browser (JSP)
↓
Servlets (Controller Layer)
↓
DAO Layer (Database Logic)
↓
MySQL Database

The JSP pages act as the user interface where students, faculty, and admins submit actions like login or attendance. These requests are sent to Servlets, which work as the controller layer—processing input, applying logic, managing sessions, and deciding what to do next. When data needs to be saved or fetched, the servlets call the DAO layer, which contains all the database code using JDBC. DAO interacts directly with the MySQL database by running SQL queries and returning results back to the servlet. Finally, the servlet forwards the processed data back to a JSP page to display it to the user.

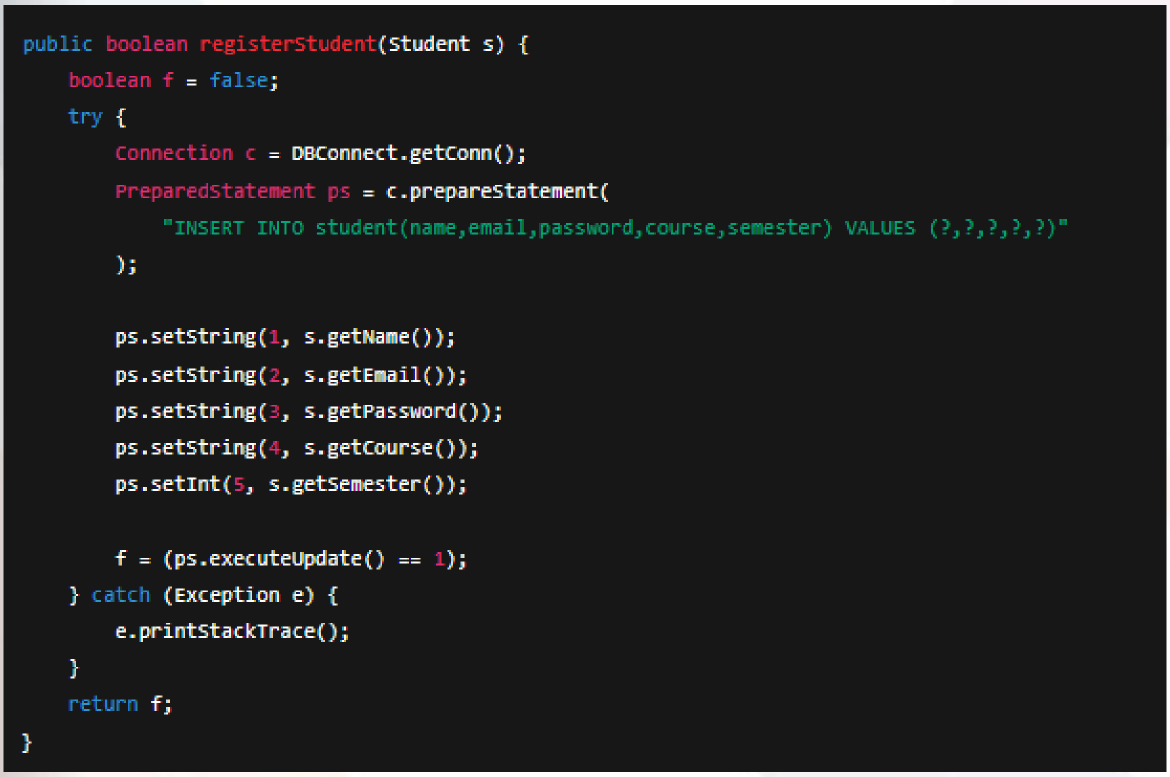
Core Java Concepts (OOP + Encapsulation + Constructors)

```java
## Student Model (OOP Example)

public class Student {
    private int student_id;
    private String name;
    private String email;
    private String password;
    private String course;
    private int semester;

    public Student() {}

    public Student(String name, String email, String password, String course, int semester) {
        this.name = name;
        this.email = email;
        this.password = password;
        this.course = course;
        this.semester = semester;
    }

    public int getStudent_id() { return student_id; }
    public String getName() { return name; }
    public String getEmail() { return email; }
    public String getCourse() { return course; }
}
```

JDBC Integration Snippet (Database Layer)

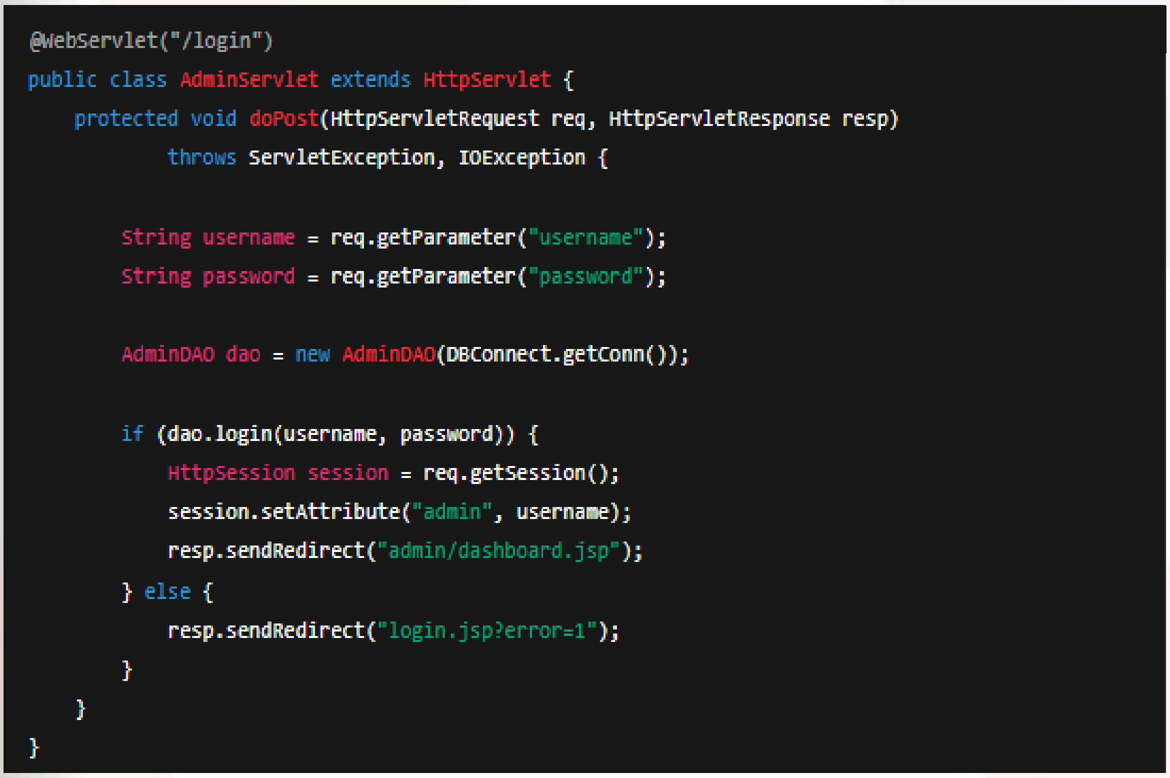
```java
public class DBConnect {
    private static Connection con;

    public static Connection getConn() {
        try {
            if (con == null) {
                Class.forName("com.mysql.cj.jdbc.Driver");
                con = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/campussync", "root", "password");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return con;
    }
}
```

StudentDAO – Insert Operation (Prepared Statement)



```java
public boolean registerStudent(Student s) {
    boolean f = false;
    try {
        Connection c = DBConnect.getConn();
        PreparedStatement ps = c.prepareStatement(
            "INSERT INTO student(name,email,password,course,semester) VALUES (?,?,?,?,?)"
        );

        ps.setString(1, s.getName());
        ps.setString(2, s.getEmail());
        ps.setString(3, s.getPassword());
        ps.setString(4, s.getCourse());
        ps.setInt(5, s.getSemester());

        f = (ps.executeUpdate() == 1);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return f;
}
```

Login Servlet (HTTP + Session + Controller Logic)



```java
@WebServlet("/login")
public class AdminServlet extends HttpServlet {
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {

        String username = req.getParameter("username");
        String password = req.getParameter("password");

        AdminDAO dao = new AdminDAO(DBConnect.getConn());

        if (dao.login(username, password)) {
            HttpSession session = req.getSession();
            session.setAttribute("admin", username);
            resp.sendRedirect("admin/dashboard.jsp");
        } else {
            resp.sendRedirect("login.jsp?error=1");
        }
    }
}
```

Login Servlet (HTTP + Session + Controller Logic)

```java
@WebServlet("/markAttendance")
public class AttendanceServlet extends HttpServlet {
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {

        int studentId = Integer.parseInt(req.getParameter("student_id"));
        int subjectId = Integer.parseInt(req.getParameter("subject_id"));
        String status = req.getParameter("status");

        AttendanceDAO dao = new AttendanceDAO(DBConnect.getConn());
        dao.markAttendance(studentId, subjectId, status);

        resp.sendRedirect("faculty/attendance.jsp");
    }
}
```

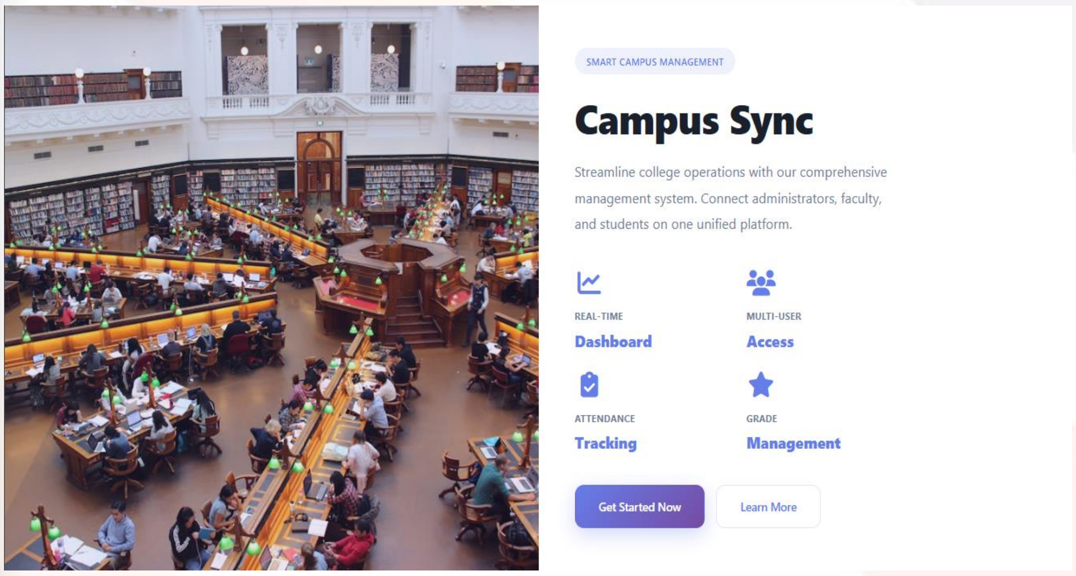
Attendance Servlet (Faculty Example)

```jsp
<h2>Welcome, <%= session.getAttribute("student") %></h2>

<h3>Your Attendance:</h3>
<table>
<tr>
    <th>Subject</th>
    <th>Status</th>
</tr>

<%
    AttendanceDAO dao = new AttendanceDAO(DBConnect.getConn());
    List<Attendance> list = dao.getAttendance(studentId);
    for (Attendance a : list) {
%>
<tr>
    <td><%= a.getSubjectName() %></td>
    <td><%= a.getStatus() %></td>
</tr>
<% } %>
</table>
```

Assignment Upload (File Handling Logic)

```java
@WebServlet("/uploadAssignment")
@MultipartConfig
public class AssignmentServlet extends HttpServlet {

    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {

        Part file = req.getPart("file");
        String fileName = file.getSubmittedFileName();

        String path = getServletContext().getRealPath("") + "files/" + fileName;
        file.write(path);

        int subjectId = Integer.parseInt(req.getParameter("subject_id"));

        AssignmentDAO dao = new AssignmentDAO(DBConnect.getConn());
        dao.saveAssignment(subjectId, fileName, path);

        resp.sendRedirect("faculty/upload_assignment.jsp");
    }
}
```

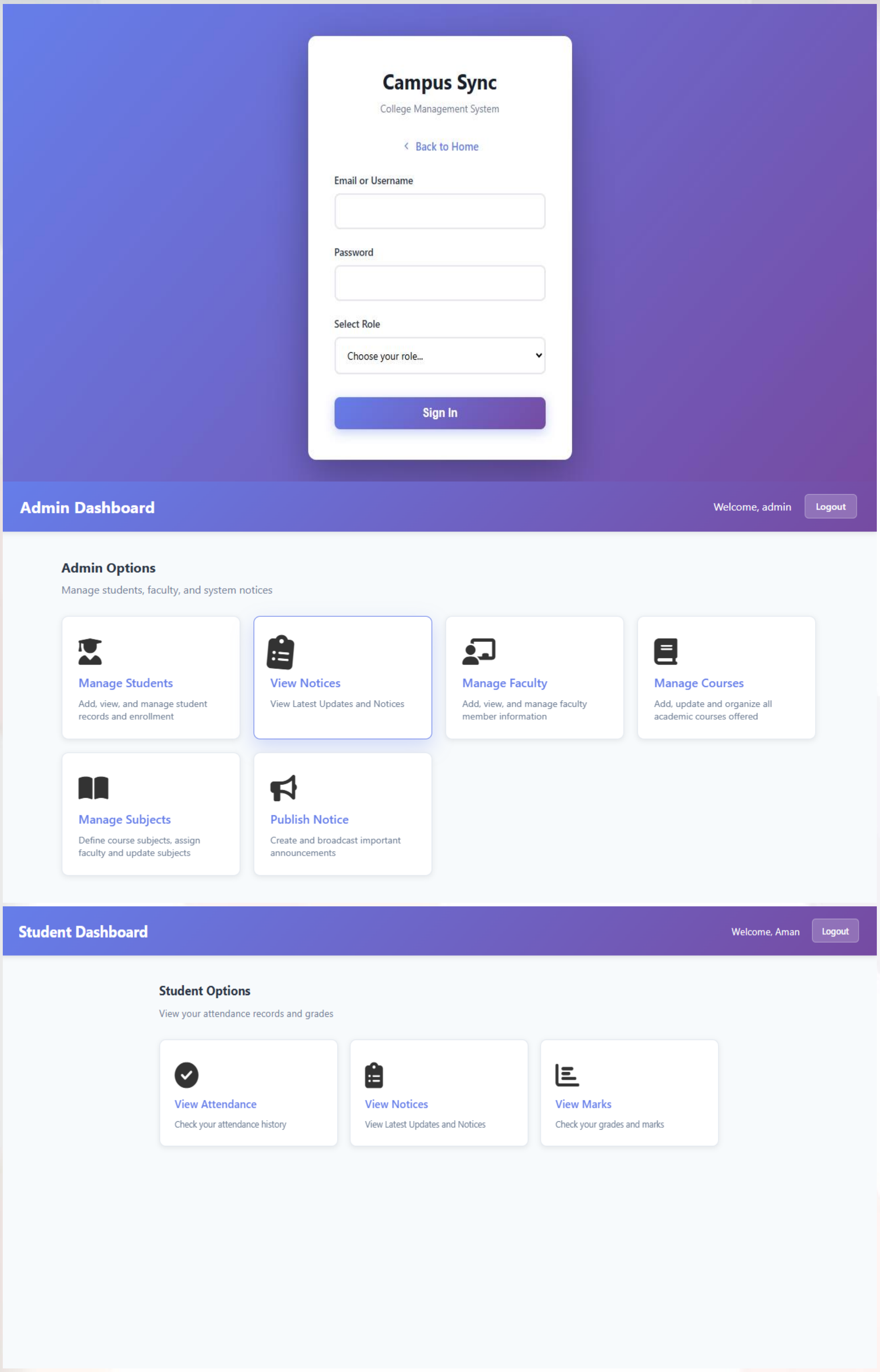
SCRCEENSHOTS FROM PROJECT WEBPAGES



SMART CAMPUS MANAGEMENT

Campus Sync

Streamline college operations with our comprehensive management system. Connect administrators, faculty, and students on one unified platform.



Campus Sync

College Management System

< Back to Home

Email or Username

Password

Select Role

Choose your role...

Sign In

Admin Dashboard

Welcome, admin

Logout

Admin Options

Manage students, faculty, and system notices

Manage Students

Add, view, and manage student records and enrollment

View Notices

View Latest Updates and Notices

Manage Faculty

Add, view, and manage faculty member information

Manage Courses

Add, update and organize all academic courses offered

Manage Subjects

Define course subjects, assign faculty and update subjects

Publish Notice

Create and broadcast important announcements

Student Dashboard

Welcome, Aman

Logout

Student Options

View your attendance records and grades

View Attendance

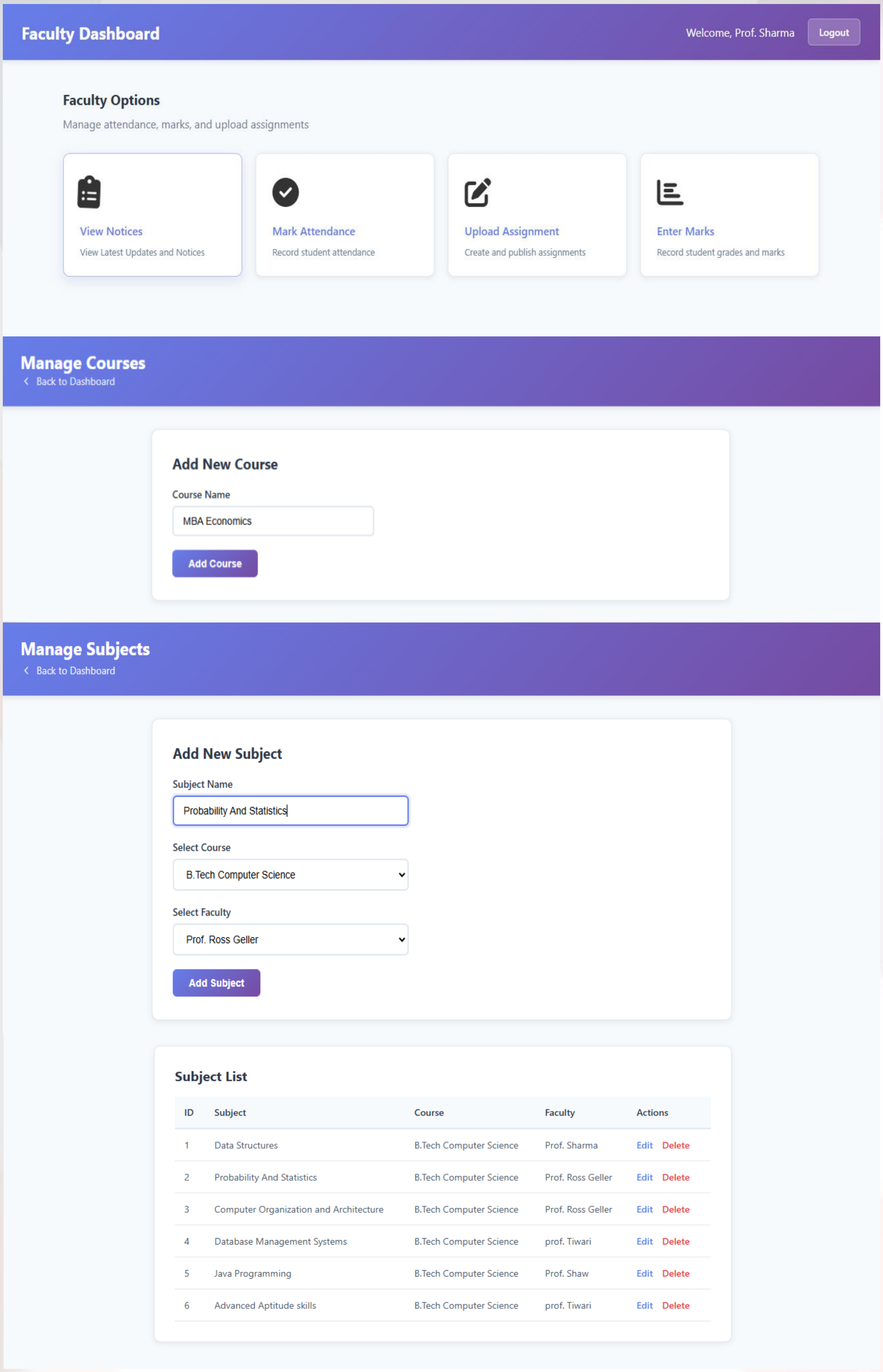
Check your attendance history

View Notices

View Latest Updates and Notices

View Marks

Check your grades and marks



Faculty Dashboard

Welcome, Prof. Sharma

Logout

Faculty Options

Manage attendance, marks, and upload assignments

View Notices
View Latest Updates and Notices

Mark Attendance
Record student attendance

Upload Assignment
Create and publish assignments

Enter Marks
Record student grades and marks

Manage Courses

‹ Back to Dashboard

Add New Course

Course Name: MBA Economics

Add Course

Manage Subjects

‹ Back to Dashboard

Add New Subject

Subject Name: Probability And Statistics

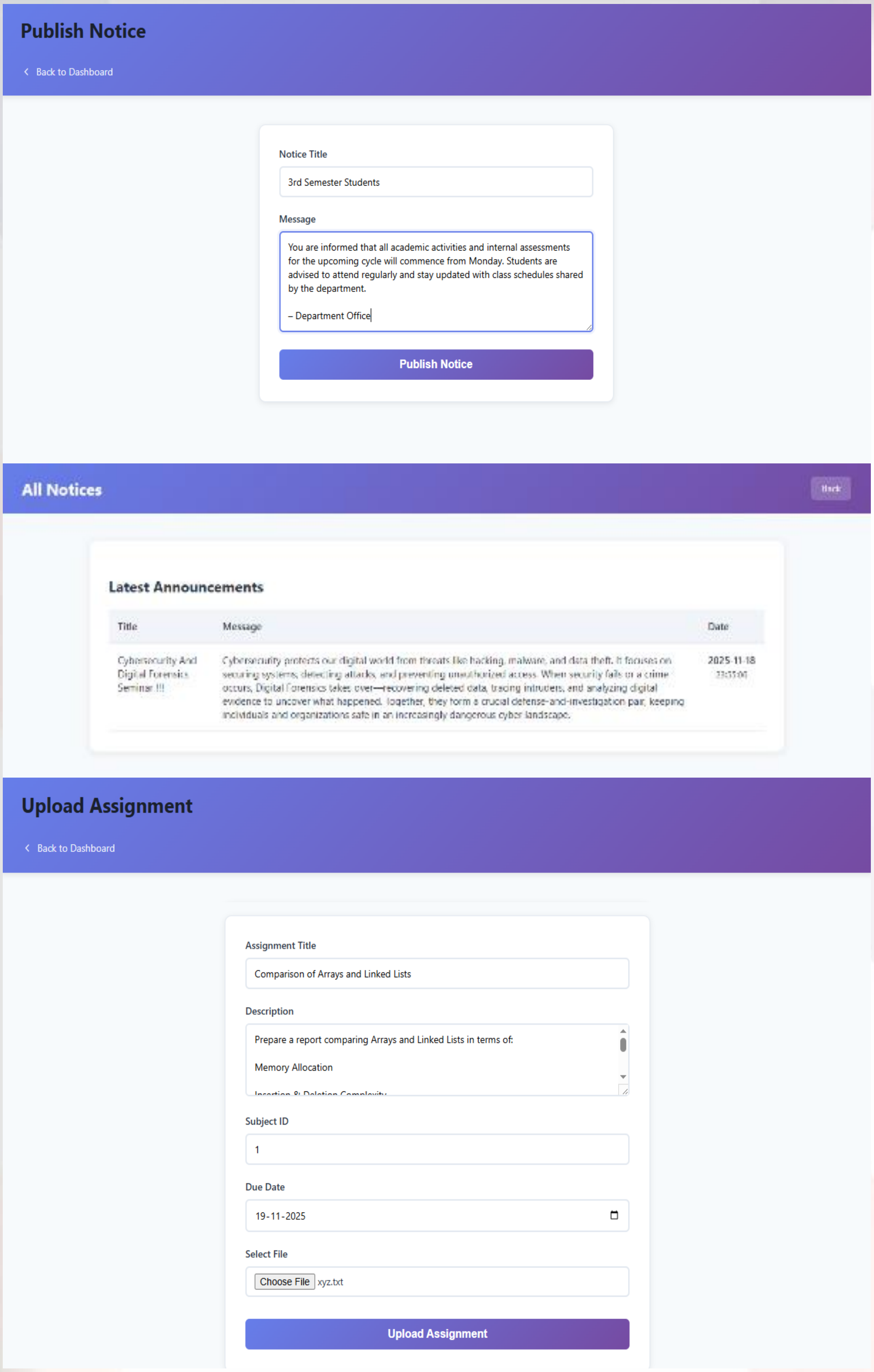
Select Course: B.Tech Computer Science

Select Faculty: Prof. Ross Geller

Add Subject

Subject List

ID	Subject	Course	Faculty	Actions
1	Data Structures	B.Tech Computer Science	Prof. Sharma	Edit Delete
2	Probability And Statistics	B.Tech Computer Science	Prof. Ross Geller	Edit Delete
3	Computer Organization and Architecture	B.Tech Computer Science	Prof. Ross Geller	Edit Delete
4	Database Management Systems	B.Tech Computer Science	prof. Tiwari	Edit Delete
5	Java Programming	B.Tech Computer Science	Prof. Shaw	Edit Delete
6	Advanced Aptitude skills	B.Tech Computer Science	prof. Tiwari	Edit Delete



Publish Notice

< Back to Dashboard

Notice Title

3rd Semester Students

Message

You are informed that all academic activities and internal assessments for the upcoming cycle will commence from Monday. Students are advised to attend regularly and stay updated with class schedules shared by the department.

– Department Office

All Notices

Latest Announcements

Title	Message	Date
Cybersecurity And Digital Forensics Seminar !!!	Cybersecurity protects our digital world from threats like hacking, malware, and data theft. It focuses on securing systems, detecting attacks, and preventing unauthorized access. When security fails or a crime occurs, Digital Forensics takes over—recovering deleted data, tracing intruders, and analyzing digital evidence to uncover what happened. Together, they form a crucial defense-and-investigation pair, keeping individuals and organizations safe in an increasingly dangerous cyber landscape.	2025-11-18 23:35:00

Upload Assignment

< Back to Dashboard

Assignment Title

Comparison of Arrays and Linked Lists

Description

Prepare a report comparing Arrays and Linked Lists in terms of:

Memory Allocation

Subject ID

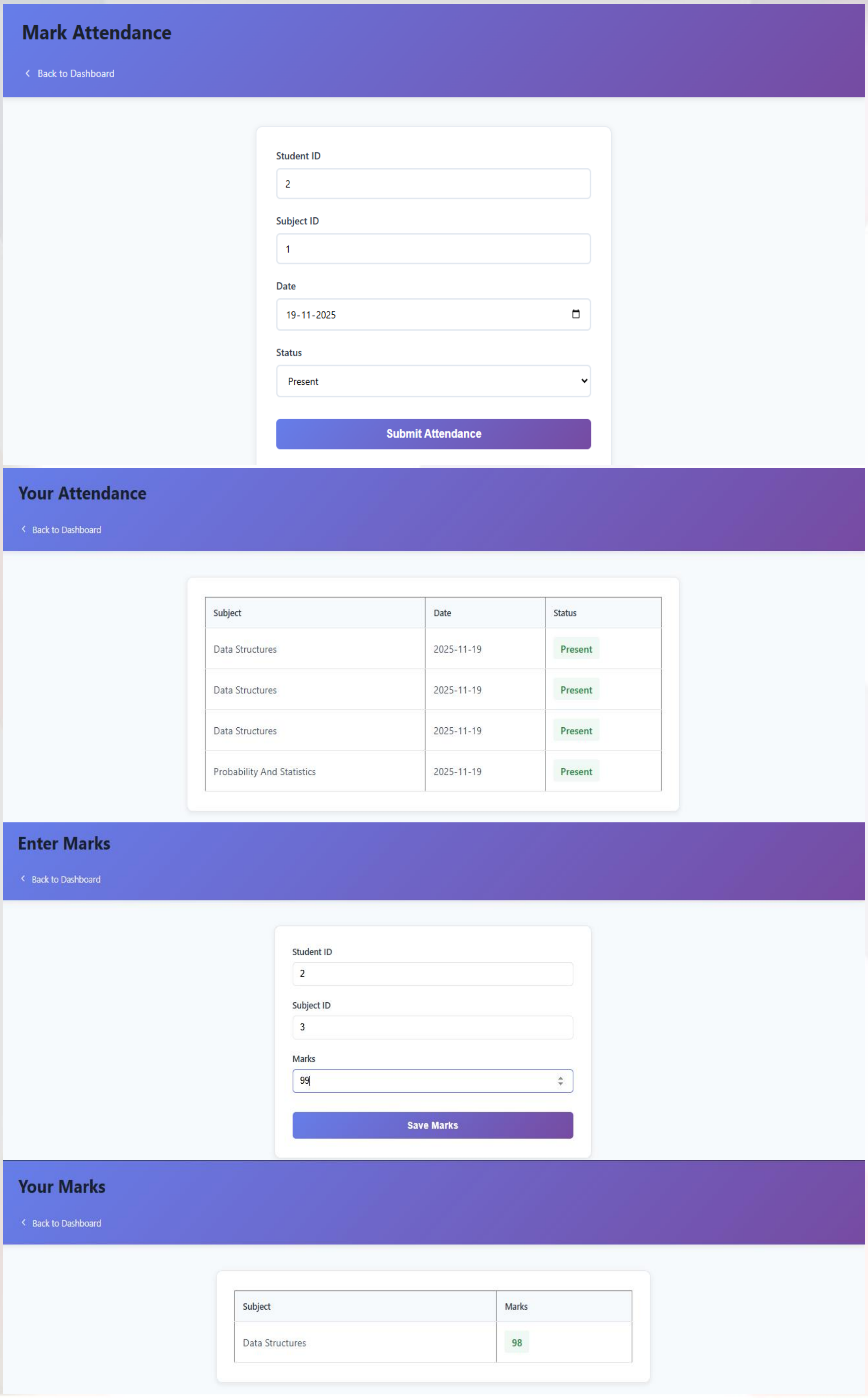
1

Due Date

19-11-2025

Select File

Choose File xyz.txt



Mark Attendance

‹ Back to Dashboard

Student ID	2
Subject ID	1
Date	19-11-2025
Status	Present

Your Attendance

‹ Back to Dashboard

Subject	Date	Status
Data Structures	2025-11-19	Present
Data Structures	2025-11-19	Present
Data Structures	2025-11-19	Present
Probability And Statistics	2025-11-19	Present

Enter Marks

‹ Back to Dashboard

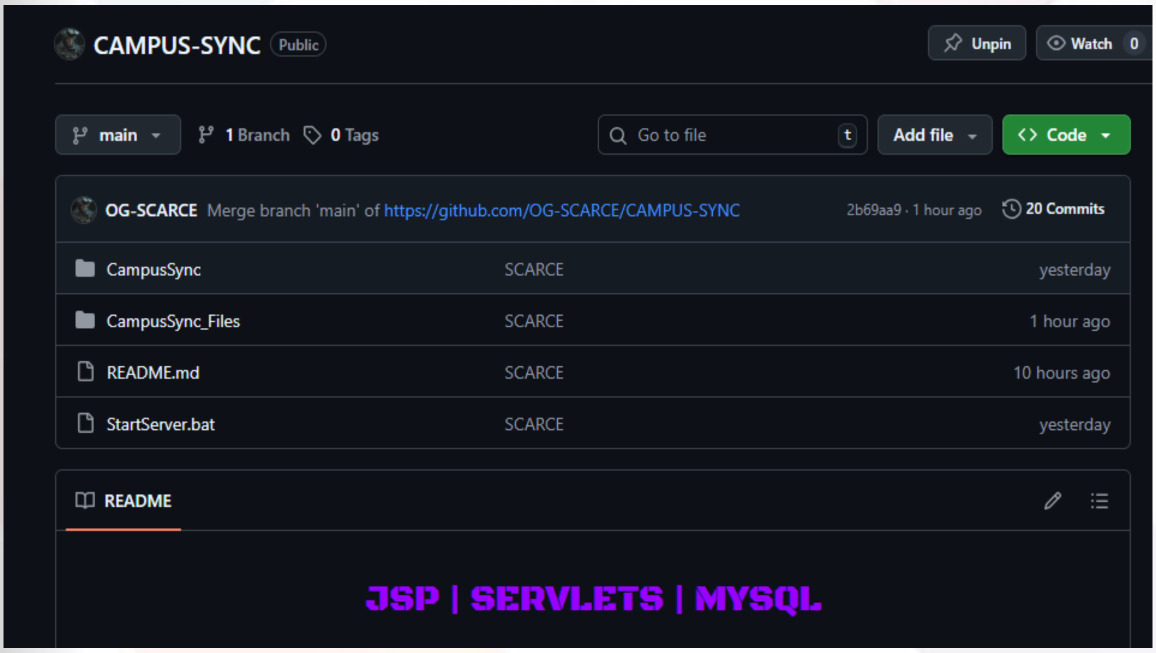
Student ID	2
Subject ID	3
Marks	99

Your Marks

‹ Back to Dashboard

Subject	Marks
Data Structures	98

GITHUB AND RUN

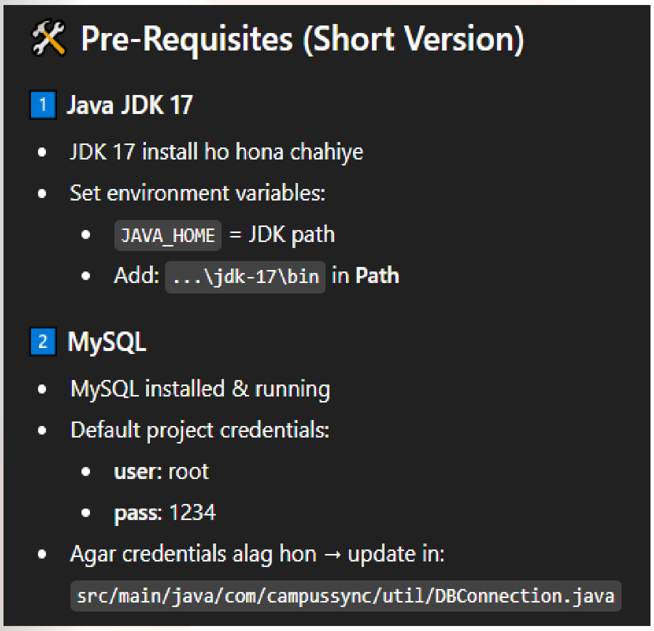


OG-SCARCE Merge branch 'main' of https://github.com/OG-SCARCE/CAMPUS-SYNC 2b69aa9 · 1 hour ago 20 Commits

CampusSync	SCARCE	yesterday
CampusSync_Files	SCARCE	1 hour ago
README.md	SCARCE	10 hours ago
StartServer.bat	SCARCE	yesterday

README

JSP | SERVLETS | MYSQL



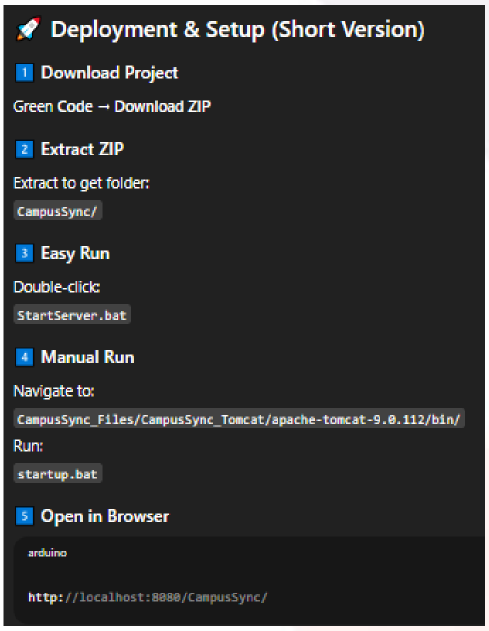
🛠️ Pre-Requisites (Short Version)

1️⃣ Java JDK 17

- JDK 17 install ho hona chahiye
- Set environment variables:
  - `JAVA_HOME` = JDK path
  - Add: `...\jdk-17\bin` in **Path**

2️⃣ MySQL

- MySQL installed & running
- Default project credentials:
  - **user**: root
  - **pass**: 1234
- Agar credentials alag hon → update in:
  `src/main/java/com/campussync/util/DBConnection.java`



🚀 Deployment & Setup (Short Version)

1️⃣ Download Project

Green Code → Download ZIP

2️⃣ Extract ZIP

Extract to get folder:
`CampusSync/`

3️⃣ Easy Run

Double-click:
`StartServer.bat`

4️⃣ Manual Run

Navigate to:
`CampusSync_Files/CampusSync_Tomcat/apache-tomcat-9.0.112/bin/`
Run:
`startup.bat`

5️⃣ Open in Browser

```arduino
http://localhost:8080/CampusSync/
```

Conclusion

The *Campus Sync* system successfully demonstrates a modern, structured, and efficient approach to campus management using JSP, Servlets, JDBC, and MySQL. By implementing the MVC architecture, the project ensures a clear separation of concerns, making the application easier to maintain, scale, and extend.

The system provides dedicated portals for administrators, faculty members, and students, each designed to streamline their respective academic operations. With features such as role-based authentication, student data handling, attendance management, assignment upload, marks handling, and notice broadcasting, the platform centralizes essential campus activities in a unified environment.

The use of JDBC-based DAO classes ensures secure and optimized communication with the database, while Tomcat serves as a reliable deployment server for dynamic web content. The folder structure is modular, organized, and aligned with industry standards, making the project suitable not only for academic submission but also real-world deployment with minor enhancements.

Overall, *Campus Sync* demonstrates strong understanding of Java web technologies and backend-driven application design. It stands as an effective, functional, and scalable solution for institutional workflow automation. With further improvements—such as integrating REST APIs, adding responsive UI, or implementing role-based analytics—the system can evolve into a complete enterprise-grade campus management platform.

thank you