**STUDENT MANAGEMENT SYSTEM**



Web Technologies Course Project

in partial fulfillment of the degree

**Bachelor of Technology**

in

**Computer Science & Engineering**

**By**

2103A51L12 I.Sai Rajeshwara Chary

Under the Guidance of

**Mr. K. Ravi Chythanya**

Assistant professor

**Submitted to**



**SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE**

# SR UNIVERSITY, ANANTHASAGAR, WARANGAL

**April, 2023.**



**SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE**

**CERTIFICATE**

This is to certify that this project entitled **“STUDENT MANAGEMENT SYSTEM USING ANGULAR”** is the bonafied work carried out by **I.SAI RAJESHWARACHARY** as a Web Technologies Course Project in the partial fulfillment to award the degree **Bachelor of Technology** in **Computer Science & Engineering** during the academic year 2022-2023 under our guidance and Supervision.

**Supervisor Head of the Department**

**ACKNOWLEDGEMENT**

I take this opportunity to express my profound gratitude and deep regards to our guide **Mr.Kanegonda Ravi Chythanya** sir HOD **Dr M.Sheshikala** mam who gave me the opportunity to do this wonderful project on the topic Student Management System using Angular. I thank my guide for his exemplary guidance, monitoring and constant encouragement throughout the course of this thesis. The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life on which I was about to embark.

The project helped me learn how to do proper Research and I learned about many new things while doing the project.

**ABSTRACT**

This project involves building a web-based student management system using Angular, a widely used JavaScript framework. The system will enable educational institutions to manage various aspects of student data, including enrollment, attendance, grades, and personal information. By leveraging Angular's component-based architecture, data binding, and dependency injection, the system will be highly scalable and robust. Additionally, user authentication and authorization will be incorporated to ensure data privacy and security. The system will streamline administrative tasks for educational institutions while also improving communication with students and parents. Ultimately, this project will provide a modern solution for educational institutions to manage student data more effectively and efficiently.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S. No** | **Title** | **Page. no** |
| **1** | **Introduction** | **6** |
| **2** | **Objective** | **7** |
| **3** | **Problem Statement** | **8** |
| **4** | **Features** | **10** |
| **5** | **Modules** | **11** |
| **6** | **Methodology** | **12** |
| **7** | **Requirement Specifications** | **13** |
| **8** | **Implementation** | **14-26** |
| **9** | **Results** | **27-29** |
| **10** | **Conclusion** | **30** |
| **11** | **Future Scope** | **31** |
| **12** | **References** | **32** |

**1. INTRODUCTION**

The goal of the project is to leverage the Angular framework to create a web-based student management system. In order to handle many parts of student data, including enrollment, attendance, grades, and personal information, educational institutions need to use a student management system. Traditional paper-based ways of maintaining student data are no longer effective and error-prone due to technological advancements. A contemporary, scalable, and reliable system that can efficiently manage student data is thus required.

Angular, a potent framework with several benefits, including a component-based architecture, data binding, and dependency injection, will be used to construct the system. Because of the component-based architecture, it is simple to maintain the programme because developers may create reusable components. Data synchronisation between the model and view is facilitated by data binding, while the management of dependencies and increased modularity of the code are achieved via dependency injection.

To guarantee the security and privacy of the data, the system will include user authentication and permission. Depending on their jobs, users will have access at varying levels, such as administrator, instructor, and student. Additionally, the system will offer an easy-to-use interface for administrators, teachers, and students to access and manage their data.

In conclusion, the project intends to offer educational institutions a cutting-edge, effective, and dependable student management system. The system will provide a complete solution to handle student data, streamline administrative processes, and enhance stakeholder communication by utilising Angular's features and including user authentication and authorization.

**2. OBJECTIVE**

The "Student Management System using Angular JS" project's goal is to create a thorough web-based application that will let educational institutions handle student data effectively. The main aim of the project is to provide a modern solution to replace traditional paper-based methods of managing student data. This system will be created using Angular JS, a potent and well-liked JavaScript framework that comes with a number of benefits like dependency injection, component-based architecture, and data binding.

The project's primary goal is to build a dependable and expandable system that can effectively manage massive amounts of student data. Angular JS will be used to construct reusable components that make it easier to maintain the application. The system will also incorporate user authentication and authorization to ensure data privacy and security. Angular's built-in security capabilities such as guards and interceptors will be utilised to manage user authentication and authorisation.

The second objective is to develop a user-friendly interface for students, teachers, and administrators to view and manage their data. Angular's data binding capability will be utilised to create a responsive and dynamic user interface, allowing users to input and manage data effortlessly. Administrative functions including as enrollment, attendance, and grading will be automated to speed processes and decrease errors. In order to help educational institutions make data-driven decisions and raise the standard of instruction overall, real-time analytics and reporting will be made available.

In conclusion, the Student Management System using Angular JS project seeks to deliver a modern, effective, and trustworthy solution to manage student data, expedite administrative duties, and improve communication with students and parents. The system will allow educational institutions to make data-driven decisions that improve the quality of education because it is adaptable, simple to use, and provides real-time analytics and reporting.

**3. PROBLEM STATEMENT**

Traditional paper-based techniques or outmoded software programmes that are challenging to use and deficient in the capabilities required for effective management of massive volumes of student data are frequently used in existing solutions for managing student data. These systems frequently involve manual data entry, which takes time, is prone to error, and makes it challenging to maintain records current.

A web-based student management system built using Angular JS is the suggested solution to the problem statement and has various advantages over other options. The suggested system is adaptable, user-friendly, and includes security safeguards to guarantee data confidentiality and privacy. Large volumes of student data may be managed effectively, allowing educational institutions to automate administrative processes like registration, attendance, and grading. Real-time monitoring and analytics will offer insightful information about students' development and help educational institutions to make data-driven decisions that raise the standard of instruction.

The proposed system will be created using Angular JS, a potent and well-liked JavaScript framework that comes with a number of benefits like dependency injection, component-based architecture, and data binding. The system will be created with an intuitive user interface that makes it simple for students, teachers, and administrators to enter and manage data. Additionally, it will be adaptable, enabling educational institutions to customise the system to their own requirements. In order to protect the security and privacy of the data, the system will also include security elements such user authentication and authorization.

Overall, the suggested Student Management System using Angular JS provides a cutting-edge, effective, and dependable solution to manage student data, restructure administrative duties, and enhance communication between students, teachers, and administrators. The technology will offer insightful information about students' development and allow educational institutions to make data-driven decisions that raise the standard of instruction.

**4. FEATURES**

A number of features will be included in the suggested remedy for the Student Management System utilising Angular JS project to deliver a cutting-edge, effective, and dependable remedy for managing student data. The following are some of the main aspects of the suggested remedy:

User-friendly interface: The system will include an intuitive user interface that makes it simple for administrators, teachers, and students to enter and manage data.

Security features: To guarantee data privacy and security, the system will have security measures including user authentication and authorization.

Automated administrative tasks: The system will automate administrative processes including registration, attendance, and grading, easing the burden on educational institutions and eliminating errors.

Real-time analytics and reporting: The system will give educational institutions access to real-time analytics and reporting.

Customizable: The system will be able to be modified to meet the specific requirements of educational institutions.

Scalability: The system will be reliable and scalable and capable of effectively handling significant volumes of student data.

Data binding: To construct a responsive and dynamic user interface that will make it simple for users to enter and manage data, Angular's data binding capability will be employed.

Component-based architecture: Reusable components will be created using Angular's component-based architecture to make application maintenance simple.

The system will be responsive to mobile devices and optimised for them, enabling administrators, teachers, and students to access information at any time and from any location.

Overall, the proposed Student Management System using Angular JS project would deliver an extensive web-based application that effectively manages student data, automates administrative duties, and improves educational quality by offering real-time analytics and reporting. The system will be a dependable option for educational institutions because it is user-friendly, adaptable, secure, scalable, and mobile device-optimized.

**5. MODULES**

StudentApp, StudentService, StudentSearch, and StudentDisplay are the four unique AngularJS modules that are used in the provided code.

studentApp module is created and used in app.js and StudentController.js files. It has no dependencies.

StudentService module is defined in studentservice.js file. It depends on Angular's $http and $q services.

studentSearch module is defined in studentdirective.js file. It is a directive that implements a student search box. It depends on the StudentService module.

studentDisplay module is defined in studentdirective.js file. It is a directive that displays the student data. It also depends on the StudentService module.

**6. METHODOLOGY**

Methodology for executing the Student Management System using the AngularJS.

Define the system's requirements and scope: You must first specify the functions and features that the system should offer. To do this, it will be necessary to determine the needs and requirements of many stakeholders, including students, teachers, and administrators.

Plan the architecture and design: Once you are certain of the system's needs and scope, you can begin planning the system's architecture and design. As part of this, the data model will be established, the user interface will be created, and the general structure of the programme will be planned.

Create the backend: Now that the architecture and design are in place, you can begin creating the system's backend. This will entail developing the RESTful API, constructing the database schema, and putting any required business logic into place.

Build the frontend: After the backend is finished, you can begin using AngularJS to build the system's frontend. In order to do this, views, components, and services must be developed and integrated with the backend API.

When the frontend and backend are finished, you can begin testing the system to make sure it is operating as planned. Unit testing, integration testing, end-to-end testing, and problem-solving will all be part of this.

Deploy and maintain: After testing is over, the system can be deployed to production and maintained. This will entail keeping an eye out for problems, correcting bugs, and updating the system as needed to adapt to shifting demands and specifications.

**7. REQUIREMENT SPECIFICATION**

The system requirements for the project student management system using Angular.js can include:

1. Operating System: Windows 7 or above, macOS 10.10 or above, or Linux distributions supported by Angular.js.

2. Browser: Any modern web browser that supports Angular.js and its dependencies, such as Chrome, Firefox, Safari, or Edge.

3. Hardware: The hardware requirements will depend on the size and complexity of the system. However, a minimum of 2GB RAM and 2GHz processor should be sufficient for a basic implementation.

4. Development tools: A text editor or Integrated Development Environment (IDE) such as Visual Studio Code or WebStorm, Git for version control, Node.js and NPM package manager to install and manage dependencies.

5. Database: A database management system such as MySQL, PostgreSQL, MongoDB, or Firebase for storing student data.

6. Framework: Angular.js framework version 1.x for front-end development.

7. Libraries and dependencies: Various libraries and dependencies like Angular UI-Router, Angular Material, Moment.js, etc. that can be installed using the NPM package manager.

8. Security: The system should have appropriate security measures to protect student data and prevent unauthorized access.

9. Performance: The system should be designed to handle a large number of concurrent users and provide optimal performance.

10. Accessibility: The system should be accessible to users with disabilities and should comply with accessibility standards such as WCAG 2.0.

**8. IMPLEMENTATION**

**Code:**

**index.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>StudentManagement</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular-route.js"></script>

<script type="text/javascript" src="app.js"></script>

<script type="text/javascript" src="studentcontroller.js"></script>

<script type="text/javascript" src="studentservice.js"></script>

<script type="text/javascript" src="studentdirective.js"></script>

</head>

<body ng-app = "studentApp">

<div class="container">

<h2 align="center"><b>Student Management System</b></h2>

<div ng-controller = "StudentController as sc">

<h2 align="center"><b>Search Students</b></h2>

<div student-search></div>

<h3 align="center"><b>Student Details</b></h3>

<div student-display></div>

</div>

</div>

</body>

</html>

**app.js**

angular

.module("studentApp", []);

**addStudent.html**

<div class="row" align="center">

<div class="col-sm-3"></div>

<div class="col-sm-4">

<div id="add" class="collapse">

<form class="form-horizontal">

<div class="form-group">

<label for="name" class="col-sm-5 control-label">Name</label>

<div class="col-sm-7">

<input type="text" class="form-control" id="name" placeholder="Name" ng-model = "sc.sname"/>

</div>

</div>

<div class="form-group">

<label for="address" class="col-sm-5 control-label">Address</label>

<div class="col-sm-7">

<input type="text" class="form-control" id="address" placeholder="Address" ng-model = "sc.saddress"/>

</div>

</div>

<!-- <div class="form-group">

<label for="class" class="col-sm-5 control-label">Class</label>

<div class="col-sm-7">

<input type="text" id="class" class="form-control" placeholder="class">

</div>

</div> -->

<div class="form-group">

<label for="class" class="col-sm-5 control-label">Class</label>

<div class="col-sm-7">

<select class="form-control" id="class" ng-model = "sc.sclass">

<option>1</option>

<option>2</option>

<option>3</option>

<option>4</option>

<option>5</option>

<option>6</option>

<option>7</option>

<option>8</option>

<option>8</option>

<option>10</option>

</select>

</div>

</div>

<div class="form-group">

<label for="mobile" class="col-sm-5 control-label">Mobile</label>

<div class="col-sm-7">

<input type="text" id="mobile" class="form-control" placeholder="9\*\*\*-\*\*\*\*\*\*" ng-model = "sc.smobile"/>

</div>

</div>

<button type="Reset" class="btn btn-success" ng-click = "sc.addStudent()">Add</button>

<button type="Reset" class="btn btn-danger">Reset</button>

</form>

</div>

</div>

<div class="col-sm-5"></div>

</div>

**studentcontroller.js**

angular

.module("studentApp")

.controller("StudentController", StudentController);

StudentController.$inject[

'StudentService'

];

function StudentController(StudentService){

var vm = this;

vm.students = StudentService.getStudents();

vm.orderBy = orderBy;

vm.deleteStudent = deleteStudent;

vm.showStatus = false;

vm.doShow = doShow;

vm.addStudent = addStudent;

vm.editStudent = editStudent;

vm.editorEnabled = [];

vm.enableEditor = enableEditor;

vm.disableEditor = disableEditor;

function orderBy(myOrder){

vm.ordering = myOrder;

}

function deleteStudent(id){

StudentService.deleteStudent(id);

}

function doShow(){

vm.showStatus = true;

}

function addStudent(){

var maxId = StudentService.maxId();

vm.student = {

id: maxId + 1,

name: vm.sname,

address: vm.saddress,

class: vm.sclass,

mobile: vm.smobile

};

StudentService.addStudent(vm.student);

}

function editStudent($index, editId, editName, editAddress, editClass, editMobile){

vm.student = {

id: editId,

name: editName,

address: editAddress,

class: editClass,

mobile: editMobile

};

StudentService.editStudent(vm.student);

vm.disableEditor($index);

}

function enableEditor(index){

vm.editorEnabled[index] = true;

}

function disableEditor(index){

vm.editorEnabled[index] = false;

}

}

**studentdirective.js**

var app = angular.module("studentApp");

app.directive("studentSearch", studentSearch);

app.directive("studentDisplay", studentDisplay);

app.directive("addStudent", addStudent);

function studentSearch(){

return {

templateUrl: "studentsearch.htm"

};

}

function studentDisplay(){

return {

templateUrl: "studentdisplay.htm"

};

}

function addStudent(){

return {

templateUrl: "addStudent.htm"

};

}

**studentdisplay.html**

<div class="row" align="center">

<button type="button" class="btn btn-default btn-sm" data-toggle="collapse" data-target="#add">

<span class="glyphicon glyphicon-plus"></span>Add-User

</button>

<div add-student></div>

<!-- <div id="add" class="collapse">

<p>This is simple collaspe</p>

</div> -->

</div>

<div align="center" class="row">

<div class="col-sm-12">

<table class="table table-stripted table-bordered">

<thead>

<th ng-click = "sc.orderBy('id')" ng-dblclick = "sc.orderBy('-id')"><strong>Id</strong></th>

<th ng-click = "sc.orderBy('name')" ng-dblclick = "sc.orderBy('-name')"><strong>Name</strong></th>

<th ng-click = "sc.orderBy('address')" ng-dblclick = "sc.orderBy('-address')"><strong>Address</strong></th>

<th ng-click = "sc.orderBy('class')" ng-dblclick = "sc.orderBy('-class')"><strong>Class</strong></th>

<th ng-click = "sc.orderBy('mobile')" ng-dblclick = "sc.orderBy('-mobile')"><strong>Mobile</strong></th>

<th><strong>Action</strong></th>

</thead>

<tr ng-repeat = "s in sc.students | orderBy : sc.ordering | filter : {id : sc.id, name : sc.name, address : sc.address, class : sc.class, mobile : sc.mobile}">

<td>{{s.id}}</td>

<td>

<div ng-hide = "sc.editorEnabled[$index]">

{{s.name}}

</div>

<div ng-show = "sc.editorEnabled[$index]">

<input type="text" value="{{s.name}}" ng-model = "s.name"/>

</div>

</td>>

<td>

<div ng-hide = "sc.editorEnabled[$index]">

{{s.address}}

</div>

<div ng-show = "sc.editorEnabled[$index]">

<input type="text" value="{{s.address}}" ng-model = "s.address"/>

</div>

</td>

<td>

<div ng-hide = "sc.editorEnabled[$index]">

{{s.class}}

</div>

<div ng-show = "sc.editorEnabled[$index]">

<select class="form-control" id="class" ng-model = "s.class" ng-selected = "s.class">

<option>1</option>

<option>2</option>

<option>3</option>

<option>4</option>

<option>5</option>

<option>6</option>

<option>7</option>

<option>8</option>

<option>8</option>

<option>10</option>

</select>

</div>

</td>

<td>

<div ng-hide = "sc.editorEnabled[$index]">

{{s.mobile}}

</div>

<div ng-show = "sc.editorEnabled[$index]">

<input type="text" value="{{s.mobile}}" ng-model = "s.mobile"/>

</div>

</td>

<td>

<div ng-hide = "sc.editorEnabled[$index]">

<button class="btn btn-info" ng-click = "sc.enableEditor($index)">Edit</button>

<button class="btn btn-danger" ng-click = "sc.deleteStudent(s.id)">Delete</button>

</div>

<div ng-show = "sc.editorEnabled[$index]">

<button class="btn btn-success" ng-click = "sc.editStudent($index, s.id, s.name, s.address, s.class, s.mobile) ">Save</button>

<button class="btn btn-danger" ng-click = "sc.disableEditor($index)">Cancel</button>

</div>

</td>

</div>

</tr>

</table>

</div>

</div>

**studentsearch.html**

<div align="center" class="row">

<form class="form form-inline">

<div class="form-group">

<label>Search Basis</label>

<input type="text" ng-model = "sc.id" class="form-control" placeholder="id" />

<input type="text" ng-model = "sc.name" class="form-control" placeholder="name" />

<input type="text" ng-model = "sc.address" class="form-control" placeholder="address"/>

<input type="text" ng-model = "sc.class" class="form-control" placeholder="class"/>

<input type="text" ng-model = "sc.mobile" class="form-control" placeholder="mobile"/>

<button class="btn btn-success">Search</button>

</div>

</form>

</div>

**studentservice.js**

angular

.module("studentApp")

.factory("StudentService", StudentService)

function StudentService(){

var oStudent = {};

oStudent.getStudents = getStudents;

oStudent.addStudent = addStudent;

oStudent.editStudent = editStudent;

oStudent.deleteStudent = deleteStudent;

oStudent.maxId = maxId;

var students = [{

id: 1,

name: 'Resma',

address: 'Samakhusi',

class: 1,

mobile: '9840-123456'

},

{

id: 2,

name: 'Nabin',

address: 'Biratnagar',

class: 8,

mobile: '9840-343536'

},

{

id: 3,

name: 'Asha',

address: 'Baneshwor',

class: 6,

mobile: '9865-124324'

},

{

id: 4,

name: 'Ayush',

address: 'Dhumbarahi',

class: 7,

mobile: '9840-129474'

},

{

id: 5,

name: 'Arjun',

address: 'Dang',

class: 10,

mobile: '9841-124385'

}];

function getStudents(){

return students;

}

function addStudent(student){

students.push(student);

}

function editStudent(student){

debugger;

for(var i = 0; i< students.length; i++){

if(students[i].id == student.id){

students[i].name = student.name;

students[i].address = student.address;

students[i].class = student.class;

students[i].mobile = student.mobile;

break;

}

}

}

function deleteStudent(sid){

for(var i = 0; i< students.length; i++){

if(students[i].id == sid){

var deleteUser = students[i].name;

var isconfirm = confirm("Are you sure to delete " + deleteUser);

if(isconfirm){

students.splice(i, 1);

alert(deleteUser +" has been deleted successfully");

}

break;

}

}

}

function maxId(){

// var max = Math.max(...students.id);

var max = 0;

for(var i = 0; i< students.length; i++){

if(students[i].id > max){

max = students[i].id;

}

}

return max;

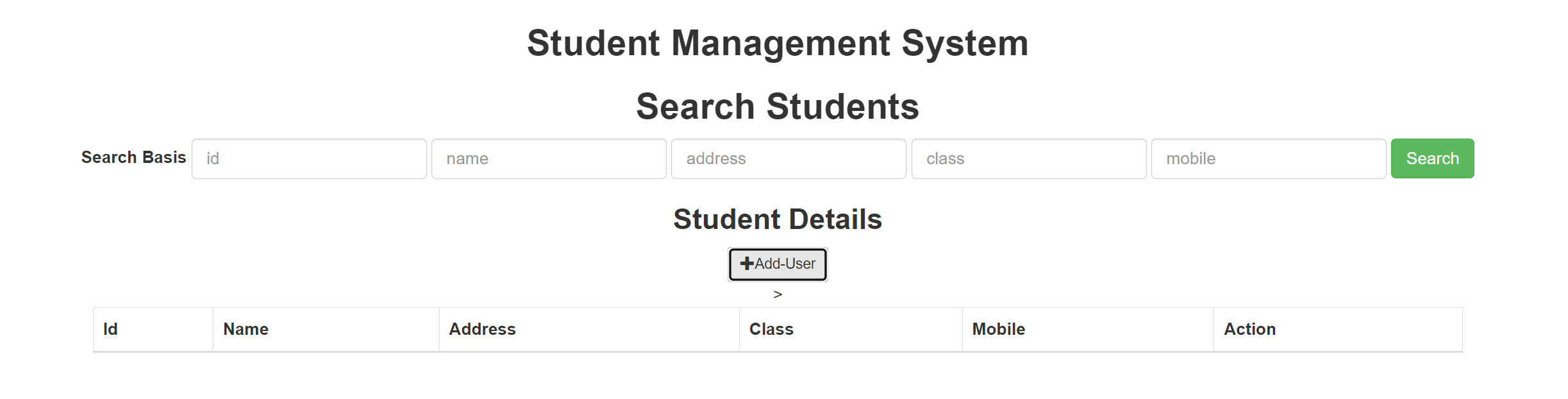
}

return Student;

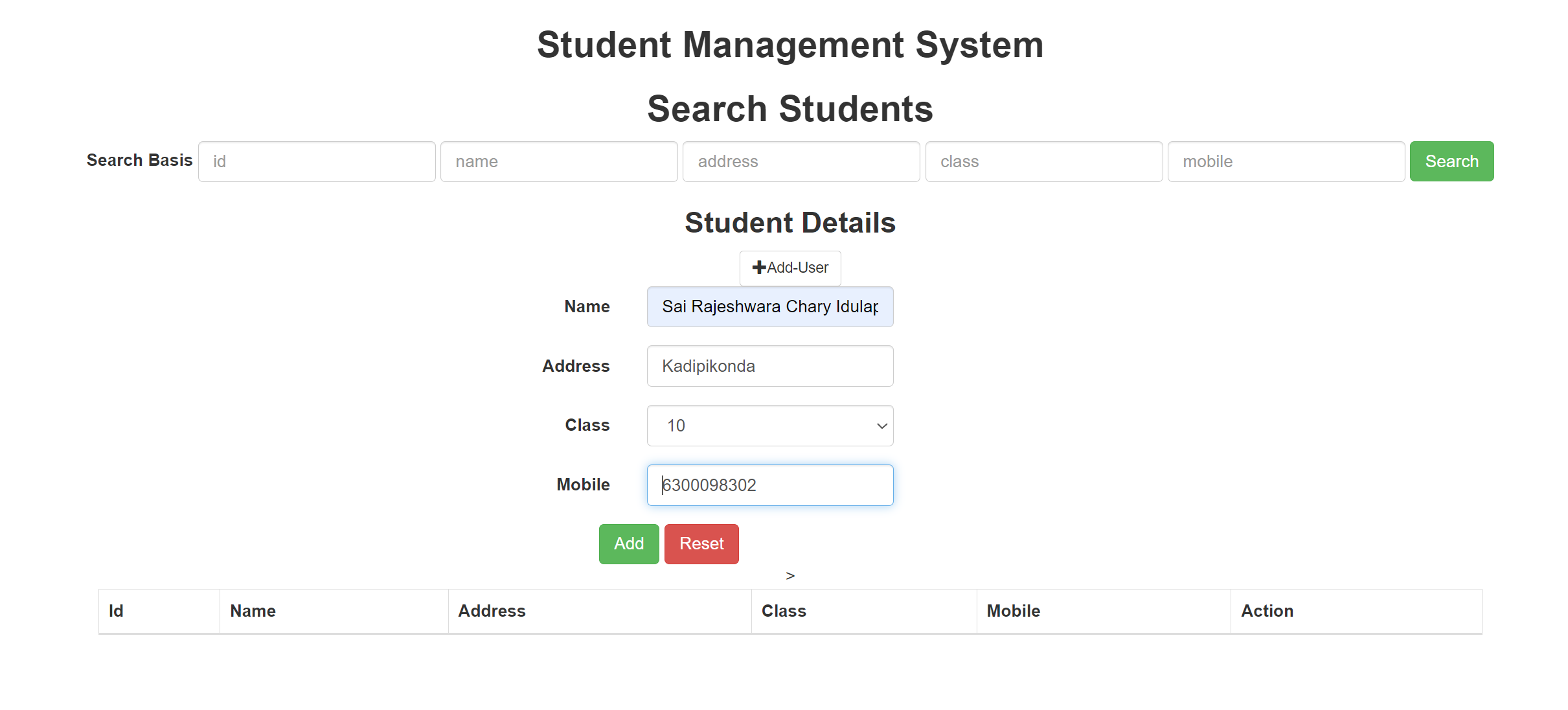
}

**9. RESULT**

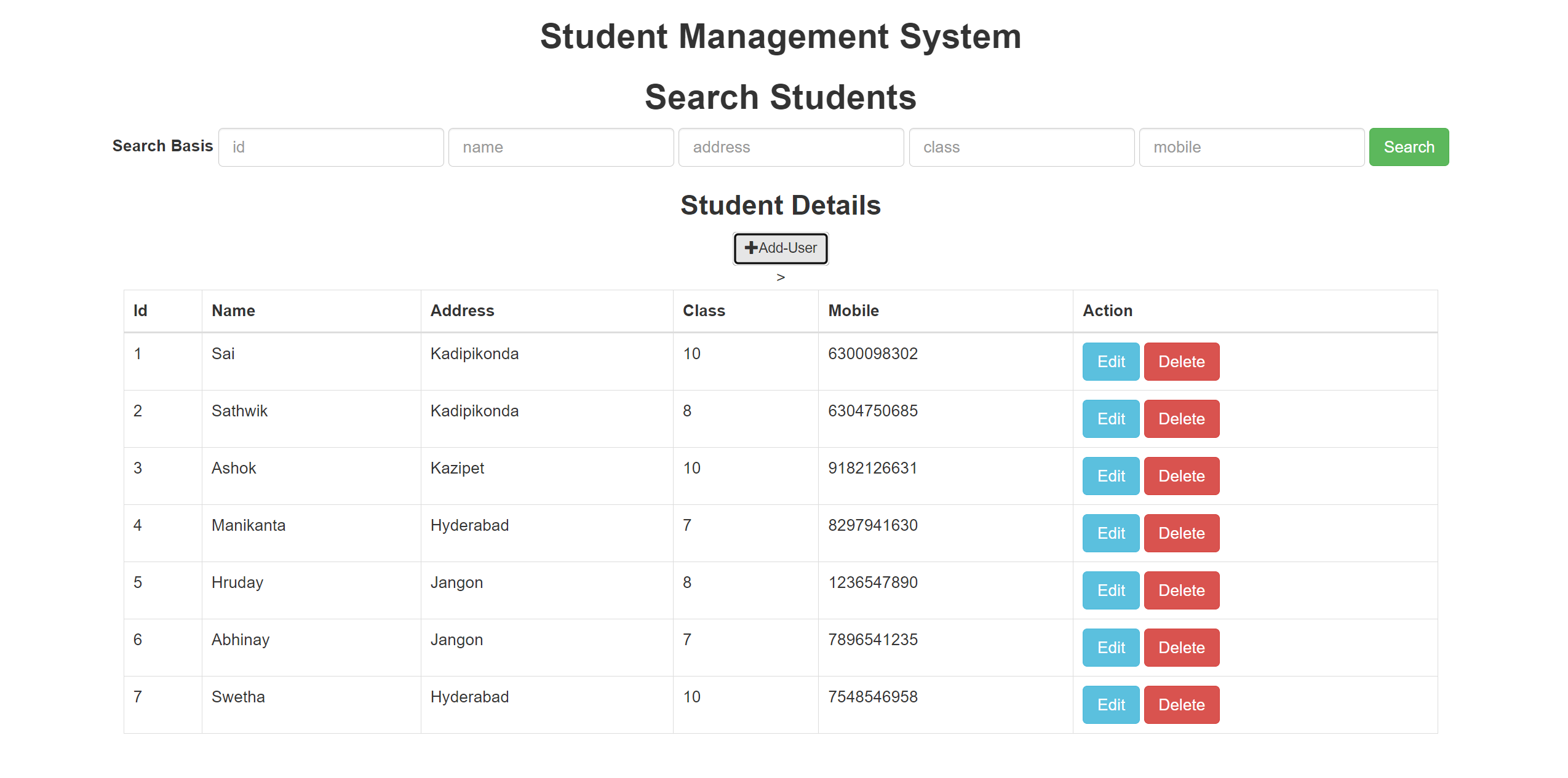
**Home Page:**

****

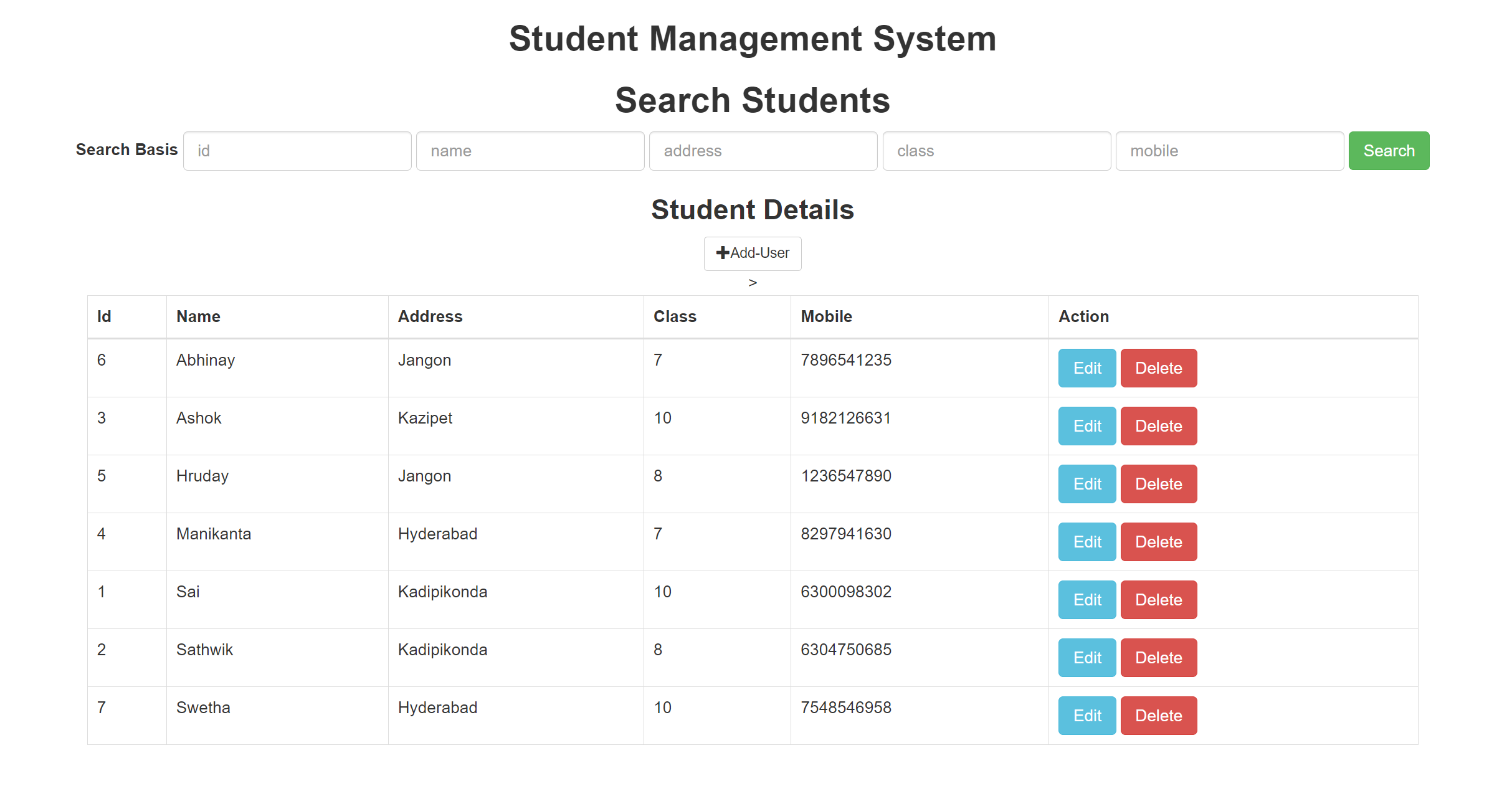
**Adding New Student Details:**

****

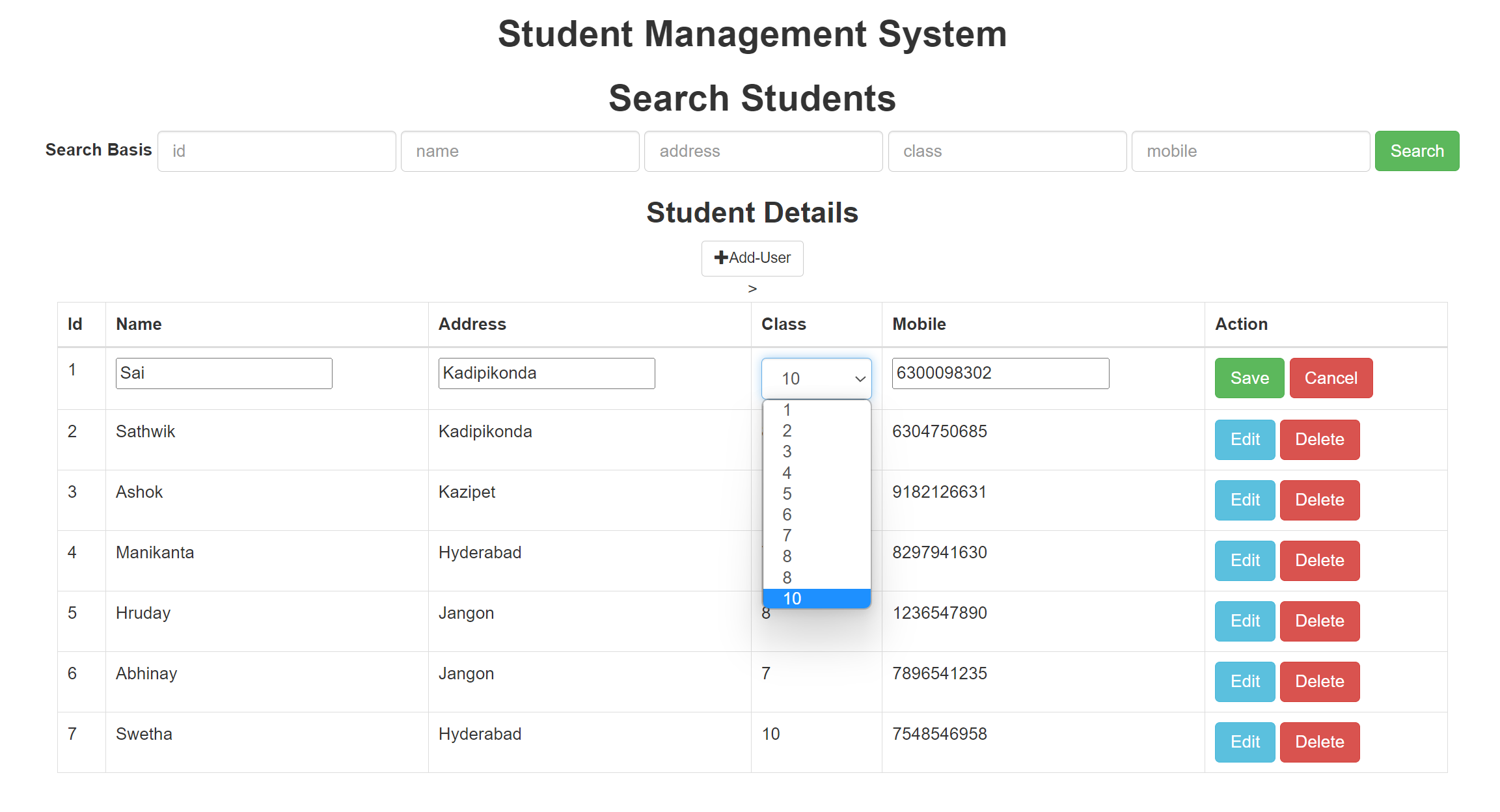
**Sort by Id:**

****

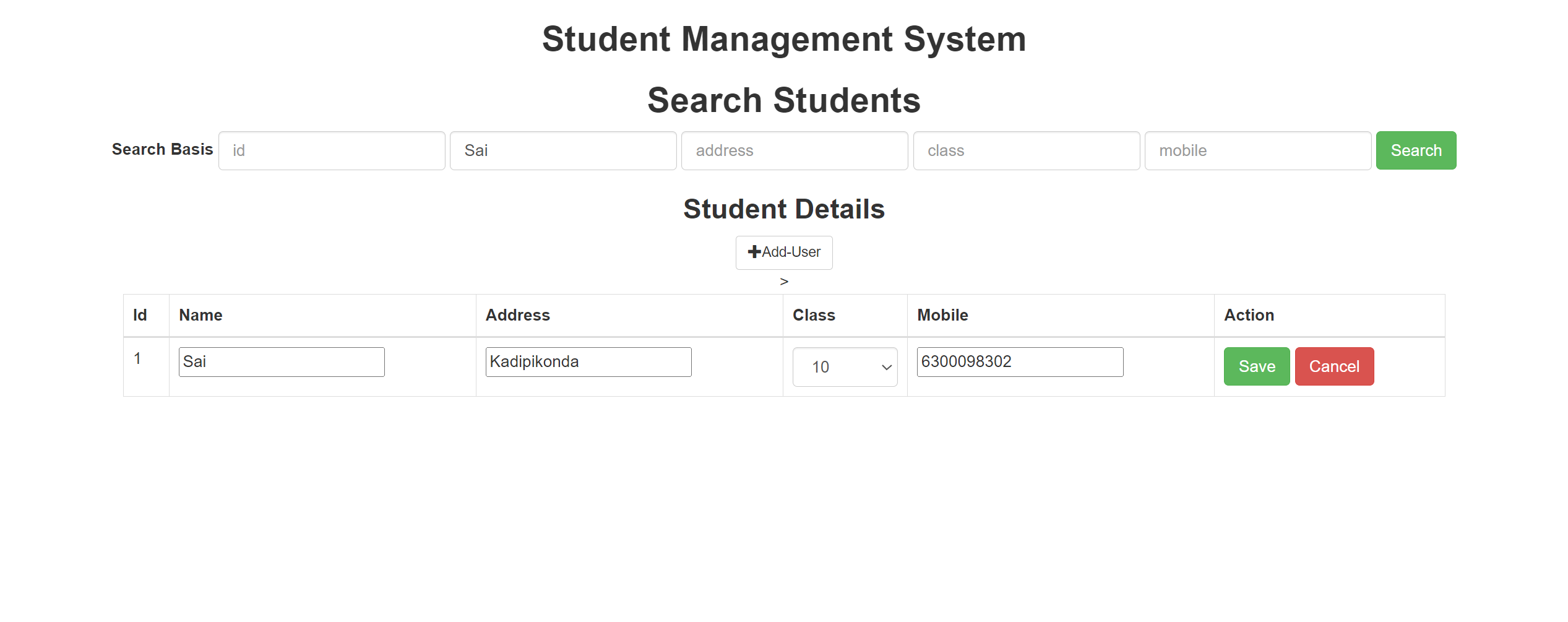
**Sort by Name:**

****

**Editing the Details of the student:**

****

**Searching with details:**

****

**10. CONCLUSION**

In conclusion, the AngularJS-based Student Management System project offers a complete management system for student records and academic activities. Administrators and educators may easily manage a variety of components of the academic process with the system's user-friendly interface, including student admittance, course registration, attendance tracking, grade management, and more.

This system's primary framework, AngularJS, offers a stable and scalable foundation for creating dynamic online applications with a rich user experience. With the help of AngularJS' strong data binding and dependency injection features, developers can create complex apps quickly and efficiently.

Overall, this project shows how AngularJS can be used to create cutting-edge web apps for handling complicated data and business operations. Its adoption can be a great advantage for educational institutions, assisting them in streamlining their academic procedures, enhancing stakeholder communication, and improving the overall student experience.

**11. FUTURE SCOPE**

Future upgrades and modifications can be made in great detail to the AngularJS student management system project. There may be room for improvement in the following areas:

Integration with other systems: The system is compatible with other learning management systems, student information systems, and library management platforms that are used in online learning.

Adaptability to mobile devices: The system may be adapted to work with smartphones and tablets, giving students and teachers a more convenient method to access and manage their information.

Advanced analytics: The system can be improved to offer advanced analytics and reporting features, allowing instructors and administrators to spot trends and patterns in students' performance and behaviour and come to informed judgements.

Integration with online payment systems: The system can be integrated with online payment platforms, making it simpler for parents and children to pay for tuition and other costs.

Artificial intelligence and machine learning: By integrating artificial intelligence and machine learning algorithms, the system can be further enhanced so that it can offer individualised recommendations and insights based on student data.

Overall, the AngularJS student management system project has a bright future and can be continuously upgraded to satisfy the changing needs of stakeholders and educational institutions.

**12. REFERENCES**

<https://angular.io/tutorial/tour-of-heroes/toh-pt0>

<https://www.javatpoint.com/student-management-system>

<https://phpgurukul.com/student-management-system-using-php-and-mysql/>

<https://www.slideshare.net/GS274/student-management-system>

<https://www.freeprojectz.com/project-source-code-database-download/student-management-system-project>