PSG college of Technology
Department of Computer Applications
23MX18 Web Application Development

06.11.2025 Worksheet 14 : (To be written in the observation NoteBook)

Due Date of Submission : 14th November 2025

Concept to use: Pattern Extraction using JavaScript, CRUD operation with MongoDB, UI development, Client server communication using Node, Express and any Middleware/body-parser.

You are to develop a mini application to maintain the skillset details of students of your class.

Through the backend create or open the database schema to add details like rollnumber, name, Guardian_Phone number (optional could be null also indicating the student is a day's scholar) , skill (could be one or many of dance, drawing, painting, singing, storyWrting,chess, cricket, soccer, orchestra etc.,)

Develop a simple UI to add student details, delete students , update skillset of student details by roll number.

Also give provision to display a list of all students having particular skill in the next page in a neat format. For searching for the skill , use Patterns in JavaScript.

Design the UI with proper form validation at the client side with HTML Patterns or ejs regEx scripts. Use proper controls like multiple selection checkbox for skill, phone number with mobile icon etc., Use suitable favicon for your website.

Your application should have routes like home, addStudent, deleteStudent, updateStudent,displayStudent(by rollNo) , skillStudent(search by Skillset) , displayAll (displays all day's scholar students)

Developing a user- friendly UI is appreciated.

Note : Students could use any css frameworks like Bootstrap.
Either you could develop this application using ejs or plain HTML/CSS/JS connect to the backend.
Use of React/Angular/Flask is not allowed.

CODE:

student.js

```
const mongoose = require('mongoose');

// Student Schema
const studentSchema = new mongoose.Schema({
  rollNumber: {
    type: Number,
    required: [true, 'Roll number is required'],
    unique: true,
    validate: {
      validator: function(v) {
        return Number.isInteger(v) && v > 0;
      },
      message: 'Roll number must be a positive integer'
    }
  },
  name: {
    type: String,
    required: [true, 'Name is required'],
    trim: true,
    minlength: [2, 'Name must be at least 2 characters long'],
    maxlength: [100, 'Name cannot exceed 100 characters']
  },
  guardianPhone: {
    type: String,
    default: null,
    validate: {
      validator: function(v) {
        // Indian mobile number format: 10 digits starting with 6-9
        if (!v || v === null || v === '') return true; // Optional field
        return /^[6-9]\d{9}$/.test(v);
      },
      message: 'Phone number must be a valid 10-digit Indian mobile
number'
    }
  },
  skills: {
    type: [String],
```

```
    required: true,
    validate: {
      validator: function(v) {
        const allowedSkills = ['dance', 'drawing', 'painting', 'singing',
'storyWriting', 'chess', 'cricket', 'soccer', 'orchestra'];
        if (!Array.isArray(v) || v.length === 0) return false;
        return v.every(skill => allowedSkills.includes(skill));
      },
      message: 'Skills must be selected from the allowed list'
    }
  }
}, {
  timestamps: true
});

// Index for faster queries
studentSchema.index({ rollNumber: 1 });
studentSchema.index({ skills: 1 });

const Student = mongoose.model('Student', studentSchema);

module.exports = Student;
```

Style.css:

```
/* Custom Styles for Student Skillset Manager */

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #f8f9fa;
    min-height: 100vh;
    display: flex;
    flex-direction: column;
}

main {
    flex: 1;
}
```

```css
.navbar-brand {
    font-weight: 600;
    font-size: 1.25rem;
}

.card {
    border: none;
    border-radius: 10px;
}

.card-header {
    border-radius: 10px 10px 0 0 !important;
    font-weight: 600;
}

.shadow {
    box-shadow: 0 0.125rem 0.25rem rgba(0, 0, 0, 0.075) !important;
}

.shadow-sm {
    box-shadow: 0 0.125rem 0.25rem rgba(0, 0, 0, 0.075) !important;
}

.form-control:focus {
    border-color: #0d6efd;
    box-shadow: 0 0 0 0.2rem rgba(13, 110, 253, 0.25);
}

.btn {
    border-radius: 5px;
    font-weight: 500;
}

.badge {
    font-size: 0.875rem;
    padding: 0.35em 0.65em;
}

.table {
```

```
    border-radius: 5px;
    overflow: hidden;
}

.table thead th {
    border-bottom: 2px solid #dee2e6;
    font-weight: 600;
}

.table-hover tbody tr:hover {
    background-color: rgba(0, 0, 0, 0.02);
}

.alert {
    border-radius: 5px;
    border: none;
}

footer {
    margin-top: auto;
}

/* Custom checkbox styling */
.form-check-input:checked {
    background-color: #0d6efd;
    border-color: #0d6efd;
}

.form-check-input:focus {
    border-color: #86b7fe;
    outline: 0;
    box-shadow: 0 0 0 0.25rem rgba(13, 110, 253, 0.25);
}

/* Phone icon styling */
.input-group-text {
    background-color: #e9ecef;
    border-right: none;
}
```

```css
.input-group .form-control {
    border-left: none;
}

.input-group .form-control:focus {
    border-left: 1px solid #ced4da;
}

/* Highlight matching text in search results */
mark {
    background-color: #ffc107;
    padding: 0;
    font-weight: 600;
}

/* Responsive adjustments */
@media (max-width: 768px) {
    .display-4 {
        font-size: 2rem;
    }

    .card {
        margin: 0 10px;
    }
}
```

validation.js:

```javascript
// Client-side form validation using JavaScript RegEx

(function() {
    'use strict';

    // Get all forms that need validation
    const forms = document.querySelectorAll('form[id$="Form"]');

    // Validate each form
```

```
    Array.from(forms).forEach(form => {
        form.addEventListener('submit', function(event) {
            if (!form.checkValidity()) {
                event.preventDefault();
                event.stopPropagation();
            }

            // Custom validation for skills checkboxes
            if (form.id === 'addStudentForm' || form.id ===
'updateStudentForm') {
                const skillsCheckboxes =
form.querySelectorAll('input[name="skills"]:checked');
                const skillsError = form.querySelector('#skillsError');

                if (skillsCheckboxes.length === 0) {
                    event.preventDefault();
                    event.stopPropagation();
                    if (skillsError) {
                        skillsError.style.display = 'block';
                    }
                } else {
                    if (skillsError) {
                        skillsError.style.display = 'none';
                    }
                }
            }

            // Custom validation for phone number (Indian format)
            const phoneInput =
form.querySelector('input[name="guardianPhone"]');
            if (phoneInput && phoneInput.value.trim() !== '') {
                const phonePattern = /^[6-9]\d{9}$/;
                if (!phonePattern.test(phoneInput.value.trim())) {
                    event.preventDefault();
                    event.stopPropagation();
                    phoneInput.setCustomValidity('Please enter a valid
10-digit Indian mobile number (starting with 6-9)');
                    phoneInput.classList.add('is-invalid');
                } else {
                    phoneInput.setCustomValidity('');
```

```
                    phoneInput.classList.remove('is-invalid');
                }
            }

            form.classList.add('was-validated');
        }, false);

        // Real-time validation for phone number
        const phoneInput =
form.querySelector('input[name="guardianPhone"]');
        if (phoneInput) {
            phoneInput.addEventListener('input', function() {
                if (this.value.trim() === '') {
                    this.setCustomValidity('');
                    this.classList.remove('is-invalid');
                } else {
                    const phonePattern = /^[6-9]\d{9}$/;
                    if (phonePattern.test(this.value.trim())) {
                        this.setCustomValidity('');
                        this.classList.remove('is-invalid');
                        this.classList.add('is-valid');
                    } else {
                        this.setCustomValidity('Please enter a valid
10-digit Indian mobile number (starting with 6-9)');
                        this.classList.remove('is-valid');
                        this.classList.add('is-invalid');
                    }
                }
            });

            // Only allow numeric input for phone
            phoneInput.addEventListener('keypress', function(e) {
                if (!/[0-9]/.test(e.key) && !['Backspace', 'Delete',
'Tab', 'Escape', 'Enter'].includes(e.key)) {
                    e.preventDefault();
                }
            });
        }

        // Real-time validation for roll number
```

```
        const rollNumberInput =
form.querySelector('input[name="rollNumber"]');
        if (rollNumberInput) {
            rollNumberInput.addEventListener('input', function() {
                const rollPattern = /^[1-9]\d*$/;
                if (this.value === '' || rollPattern.test(this.value)) {
                    this.setCustomValidity('');
                    this.classList.remove('is-invalid');
                } else {
                    this.setCustomValidity('Roll number must be a positive
integer');
                    this.classList.add('is-invalid');
                }
            });
        }

        // Real-time validation for name
        const nameInput = form.querySelector('input[name="name"]');
        if (nameInput) {
            nameInput.addEventListener('input', function() {
                const namePattern = /^[A-Za-z\s]{2,100}$/;
                if (this.value === '' || namePattern.test(this.value)) {
                    this.setCustomValidity('');
                    this.classList.remove('is-invalid');
                } else {
                    this.setCustomValidity('Name must be 2-100 characters
and contain only letters and spaces');
                    this.classList.add('is-invalid');
                }
            });
        }

        // Skills checkbox validation
        if (form.id === 'addStudentForm' || form.id ===
'updateStudentForm') {
            const skillsCheckboxes =
form.querySelectorAll('input[name="skills"]');
            const skillsError = form.querySelector('#skillsError');

            skillsCheckboxes.forEach(checkbox => {
```

```
                checkbox.addEventListener('change', function() {
                    const checkedSkills =
form.querySelectorAll('input[name="skills"]:checked');
                    if (checkedSkills.length > 0 && skillsError) {
                        skillsError.style.display = 'none';
                    }
                });
            });
        }
    });

    // Pattern-based skill search validation (client-side regex preview)
    const skillPatternInput =
document.querySelector('input[name="skillPattern"]');
    if (skillPatternInput) {
        skillPatternInput.addEventListener('input', function() {
            const pattern = this.value.trim();
            if (pattern) {
                try {
                    // Test if the pattern is a valid regex
                    new RegExp(pattern, 'i');
                    this.setCustomValidity('');
                    this.classList.remove('is-invalid');
                } catch (e) {
                    // Invalid regex pattern
                    this.setCustomValidity('Invalid pattern');
                    this.classList.add('is-invalid');
                }
            } else {
                this.setCustomValidity('');
                this.classList.remove('is-invalid');
            }
        });
    }

})();
```

studentroutes.js:

```
const express = require('express');
const router = express.Router();
const Student = require('../models/Student');

// GET / - Home page
router.get('/', (req, res) => {
  res.render('home', { title: 'Student Skillset Manager - Home' });
});

// GET /addStudent - Form to add a student
router.get('/addStudent', (req, res) => {
  res.render('addStudent', {
    title: 'Add Student',
    error: null,
    success: null
  });
});

// POST /addStudent - Insert student into DB
router.post('/addStudent', async (req, res) => {
  try {
    const { rollNumber, name, guardianPhone, skills } = req.body;

    // Convert rollNumber to number
    const rollNum = parseInt(rollNumber);

    // Handle skills - can be string or array
    let skillsArray = [];
    if (Array.isArray(skills)) {
      skillsArray = skills;
    } else if (typeof skills === 'string') {
      skillsArray = [skills];
    }

    // Handle guardianPhone - empty string becomes null
    const phone = guardianPhone && guardianPhone.trim() !== '' ?
guardianPhone.trim() : null;
```

```
    // Create new student
    const student = new Student({
      rollNumber: rollNum,
      name: name.trim(),
      guardianPhone: phone,
      skills: skillsArray
    });

    await student.save();

    res.render('addStudent', {
      title: 'Add Student',
      error: null,
      success: `Student with roll number ${rollNum} added successfully!`
    });
  } catch (error) {
    let errorMessage = 'Failed to add student. ';
    if (error.code === 11000) {
      errorMessage += 'Roll number already exists!';
    } else if (error.errors) {
      errorMessage += Object.values(error.errors).map(e =>
e.message).join(', ');
    } else {
      errorMessage += error.message;
    }

    res.render('addStudent', {
      title: 'Add Student',
      error: errorMessage,
      success: null
    });
  }
});

// GET /deleteStudent - Page with roll number input
router.get('/deleteStudent', (req, res) => {
  res.render('deleteStudent', {
    title: 'Delete Student',
    error: null,
    success: null
```

```
  });
});

// POST /deleteStudent - Delete by roll number
router.post('/deleteStudent', async (req, res) => {
  try {
    const { rollNumber } = req.body;
    const rollNum = parseInt(rollNumber);

    const student = await Student.findOneAndDelete({ rollNumber: rollNum
});

    if (!student) {
      return res.render('deleteStudent', {
        title: 'Delete Student',
        error: `Student with roll number ${rollNum} not found!`,
        success: null
      });
    }

    res.render('deleteStudent', {
      title: 'Delete Student',
      error: null,
      success: `Student with roll number ${rollNum} deleted successfully!`
    });
  } catch (error) {
    res.render('deleteStudent', {
      title: 'Delete Student',
      error: 'Failed to delete student: ' + error.message,
      success: null
    });
  }
});

// GET /updateStudent - Page to update skillset by roll number
router.get('/updateStudent', (req, res) => {
  res.render('updateStudent', {
    title: 'Update Student Skills',
    error: null,
    success: null,
```

```
    student: null
  });
});

// POST /updateStudent - Update skills
router.post('/updateStudent', async (req, res) => {
  try {
    const { rollNumber, skills } = req.body;
    const rollNum = parseInt(rollNumber);

    // Handle skills - can be string or array
    let skillsArray = [];
    if (Array.isArray(skills)) {
      skillsArray = skills;
    } else if (typeof skills === 'string') {
      skillsArray = [skills];
    }

    const student = await Student.findOneAndUpdate(
      { rollNumber: rollNum },
      { skills: skillsArray },
      { new: true, runValidators: true }
    );

    if (!student) {
      return res.render('updateStudent', {
        title: 'Update Student Skills',
        error: `Student with roll number ${rollNum} not found!`,
        success: null,
        student: null
      });
    }

    res.render('updateStudent', {
      title: 'Update Student Skills',
      error: null,
      success: `Skills updated successfully for roll number ${rollNum}!`,
      student: student
    });
  } catch (error) {
```

```
    let errorMessage = 'Failed to update student. ';
    if (error.errors) {
      errorMessage += Object.values(error.errors).map(e =>
e.message).join(', ');
    } else {
      errorMessage += error.message;
    }

    res.render('updateStudent', {
      title: 'Update Student Skills',
      error: errorMessage,
      success: null,
      student: null
    });
  }
});

// GET /displayStudent - Display student by roll number
router.get('/displayStudent', async (req, res) => {
  try {
    const { rollNumber } = req.query;

    if (!rollNumber) {
      return res.render('displayStudent', {
        title: 'Display Student',
        student: null,
        error: null
      });
    }

    const rollNum = parseInt(rollNumber);
    const student = await Student.findOne({ rollNumber: rollNum });

    if (!student) {
      return res.render('displayStudent', {
        title: 'Display Student',
        student: null,
        error: `Student with roll number ${rollNum} not found!`
      });
    }
```

```
    res.render('displayStudent', {
      title: 'Display Student',
      student: student,
      error: null
    });
  } catch (error) {
    res.render('displayStudent', {
      title: 'Display Student',
      student: null,
      error: 'Error: ' + error.message
    });
  }
});

// GET /skillStudent - Search page for skill (pattern search)
router.get('/skillStudent', (req, res) => {
  res.render('skillStudent', {
    title: 'Search by Skill',
    students: null,
    searchTerm: null,
    error: null
  });
});

// POST /skillStudent - Show all students having a skill (regex based)
router.post('/skillStudent', async (req, res) => {
  try {
    const { skillPattern } = req.body;

    if (!skillPattern || skillPattern.trim() === '') {
      return res.render('skillStudent', {
        title: 'Search by Skill',
        students: null,
        searchTerm: null,
        error: 'Please enter a skill pattern to search'
      });
    }

    // Create regex pattern for case-insensitive partial matching
```

```
    const regexPattern = new RegExp(skillPattern.trim(), 'i');

    // Find all students whose skills array contains a skill matching the
pattern
    const students = await Student.find({
      skills: { $regex: regexPattern }
    }).sort({ rollNumber: 1 });

    res.render('skillStudent', {
      title: 'Search by Skill',
      students: students,
      searchTerm: skillPattern.trim(),
      error: null
    });
  } catch (error) {
    res.render('skillStudent', {
      title: 'Search by Skill',
      students: null,
      searchTerm: null,
      error: 'Error searching: ' + error.message
    });
  }
});

// GET /displayAll - Display all day scholars (guardianPhone = null)
router.get('/displayAll', async (req, res) => {
  try {
    const students = await Student.find({ guardianPhone: null })
      .sort({ rollNumber: 1 });

    res.render('displayAll', {
      title: 'Day Scholars (No Guardian Phone)',
      students: students,
      error: null
    });
  } catch (error) {
    res.render('displayAll', {
      title: 'Day Scholars (No Guardian Phone)',
      students: [],
      error: 'Error fetching students: ' + error.message
```

```
    });
  }
});

module.exports = router;
```

addStudent.ejs:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title><%= title %></title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.
css" rel="stylesheet">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.0/font/bootstrap-i
cons.css">
    <link rel="icon" type="image/svg+xml" href="/images/favicon.svg">
    <link rel="stylesheet" href="/css/style.css">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <div class="container">
            <a class="navbar-brand" href="/">
                <i class="bi bi-mortarboard-fill"></i> Student Skillset
Manager
            </a>
            <button class="navbar-toggler" type="button"
data-bs-toggle="collapse" data-bs-target="#navbarNav">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="/">Home</a>
```

```
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" href="/addStudent">Add
Student</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/displayStudent">Display
Student</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/updateStudent">Update
Student</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/deleteStudent">Delete
Student</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/skillStudent">Search by
Skill</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/displayAll">Day
Scholars</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>

    <div class="container mt-5">
        <div class="row justify-content-center">
            <div class="col-md-8">
                <div class="card shadow">
                    <div class="card-header bg-primary text-white">
                        <h4 class="mb-0"><i class="bi
bi-person-plus-fill"></i> Add New Student</h4>
                    </div>
                    <div class="card-body">
                        <% if (error) { %>
```

```
                        <div class="alert alert-danger
alert-dismissible fade show" role="alert">
                            <i class="bi
bi-exclamation-triangle-fill"></i> <%= error %>
                            <button type="button" class="btn-close"
data-bs-dismiss="alert"></button>
                        </div>
                    <% } %>
                    <% if (success) { %>
                        <div class="alert alert-success
alert-dismissible fade show" role="alert">
                            <i class="bi bi-check-circle-fill"></i>
<%= success %>
                            <button type="button" class="btn-close"
data-bs-dismiss="alert"></button>
                        </div>
                    <% } %>

                    <form action="/addStudent" method="POST"
id="addStudentForm" novalidate>
                        <div class="mb-3">
                            <label for="rollNumber"
class="form-label">Roll Number <span class="text-danger">*</span></label>
                            <input type="number"
                                   class="form-control"
                                   id="rollNumber"
                                   name="rollNumber"
                                   required
                                   min="1"
                                   pattern="[0-9]+"
                                   placeholder="Enter roll number">
                            <div class="invalid-feedback">Please enter
a valid roll number (positive integer).</div>
                        </div>

                        <div class="mb-3">
                            <label for="name" class="form-label">Name
<span class="text-danger">*</span></label>
                            <input type="text"
                                   class="form-control"
```

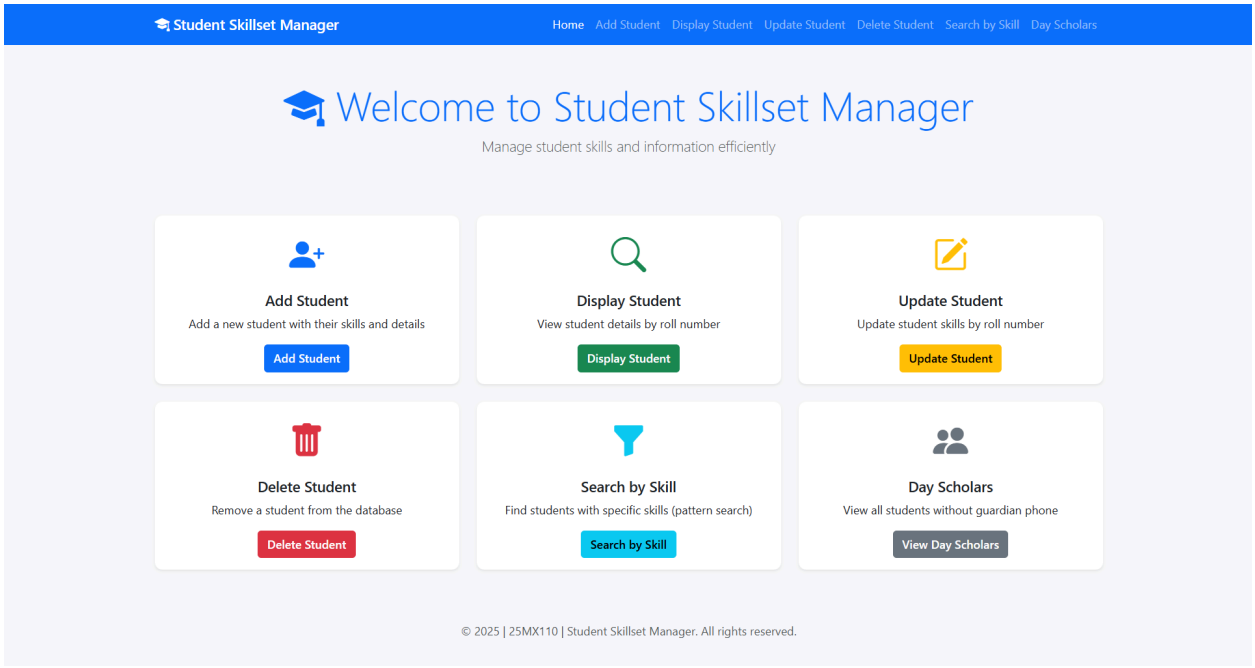
```
                                    id="name"
                                    name="name"
                                    required
                                    pattern="[A-Za-z\s]{2,100}"
                                    minlength="2"
                                    maxlength="100"
                                    placeholder="Enter student name">
                                <div class="invalid-feedback">Name must be
2-100 characters and contain only letters and spaces.</div>
                            </div>

                            <div class="mb-3">
                                <label for="guardianPhone"
class="form-label">
                                    <i class="bi bi-telephone-fill"></i>
Guardian Phone (Optional)
                                </label>
                                <div class="input-group">
                                    <span class="input-group-text"><i
class="bi bi-telephone-fill"></i></span>
                                    <input type="tel"
                                           class="form-control"
                                           id="guardianPhone"
                                           name="guardianPhone"
                                           pattern="[6-9][0-9]{9}"
                                           placeholder="10-digit Indian
mobile number"
                                           maxlength="10">
                                </div>
                                <small class="form-text
text-muted">10-digit Indian mobile number (starting with 6-9). Leave empty
for day scholars.</small>
                                <div class="invalid-feedback">Please enter
a valid 10-digit Indian mobile number (starting with 6-9).</div>
                            </div>

                            <div class="mb-3">
                                <label class="form-label">Skills <span
class="text-danger">*</span></label>
                                <div class="row">
```

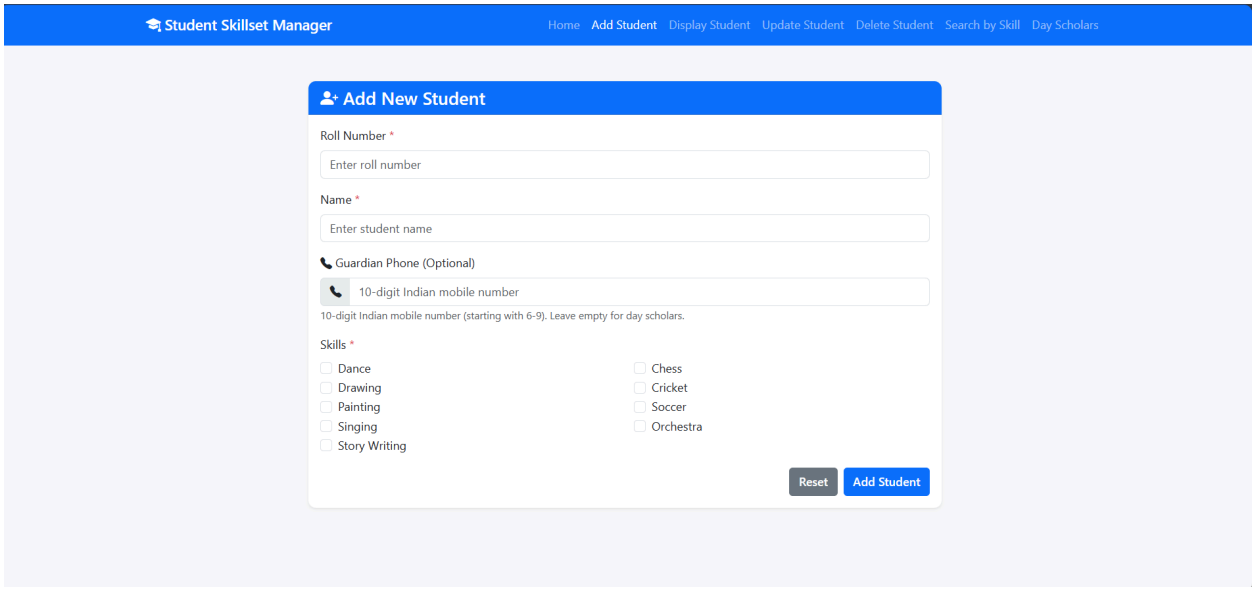
```
                                    <div class="col-md-6">
                                        <div class="form-check">
                                            <input
class="form-check-input" type="checkbox" name="skills" value="dance"
id="skillDance">
                                            <label
class="form-check-label" for="skillDance">Dance</label>
                                        </div>
                                        <div class="form-check">
                                            <input
class="form-check-input" type="checkbox" name="skills" value="drawing"
id="skillDrawing">
                                            <label
class="form-check-label" for="skillDrawing">Drawing</label>
                                        </div>
                                        <div class="form-check">
                                            <input
class="form-check-input" type="checkbox" name="skills" value="painting"
id="skillPainting">
                                            <label
class="form-check-label" for="skillPainting">Painting</label>
                                        </div>
                                        <div class="form-check">
                                            <input
class="form-check-input" type="checkbox" name="skills" value="singing"
id="skillSinging">
                                            <label
class="form-check-label" for="skillSinging">Singing</label>
                                        </div>
                                        <div class="form-check">
                                            <input
class="form-check-input" type="checkbox" name="skills"
value="storyWriting" id="skillStoryWriting">
                                            <label
class="form-check-label" for="skillStoryWriting">Story Writing</label>
                                        </div>
                                    </div>
                                    <div class="col-md-6">
                                        <div class="form-check">
```

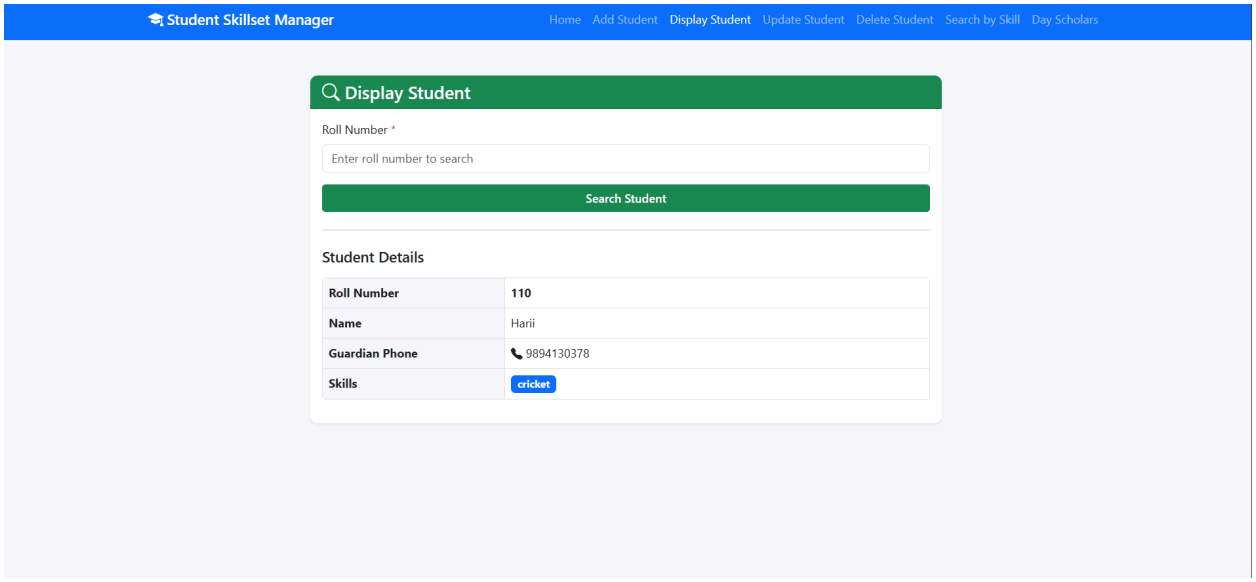
```
                                        <input
class="form-check-input" type="checkbox" name="skills" value="chess"
id="skillChess">
                                        <label
class="form-check-label" for="skillChess">Chess</label>
                                    </div>
                                    <div class="form-check">
                                        <input
class="form-check-input" type="checkbox" name="skills" value="cricket"
id="skillCricket">
                                        <label
class="form-check-label" for="skillCricket">Cricket</label>
                                    </div>
                                    <div class="form-check">
                                        <input
class="form-check-input" type="checkbox" name="skills" value="soccer"
id="skillSoccer">
                                        <label
class="form-check-label" for="skillSoccer">Soccer</label>
                                    </div>
                                    <div class="form-check">
                                        <input
class="form-check-input" type="checkbox" name="skills" value="orchestra"
id="skillOrchestra">
                                        <label
class="form-check-label" for="skillOrchestra">Orchestra</label>
                                    </div>
                                </div>
                            </div>
                            <div class="text-danger mt-2"
id="skillsError" style="display: none;">
                                Please select at least one skill.
                            </div>
                        </div>

                        <div class="d-grid gap-2 d-md-flex
justify-content-md-end">
                            <button type="reset" class="btn
btn-secondary">Reset</button>
```

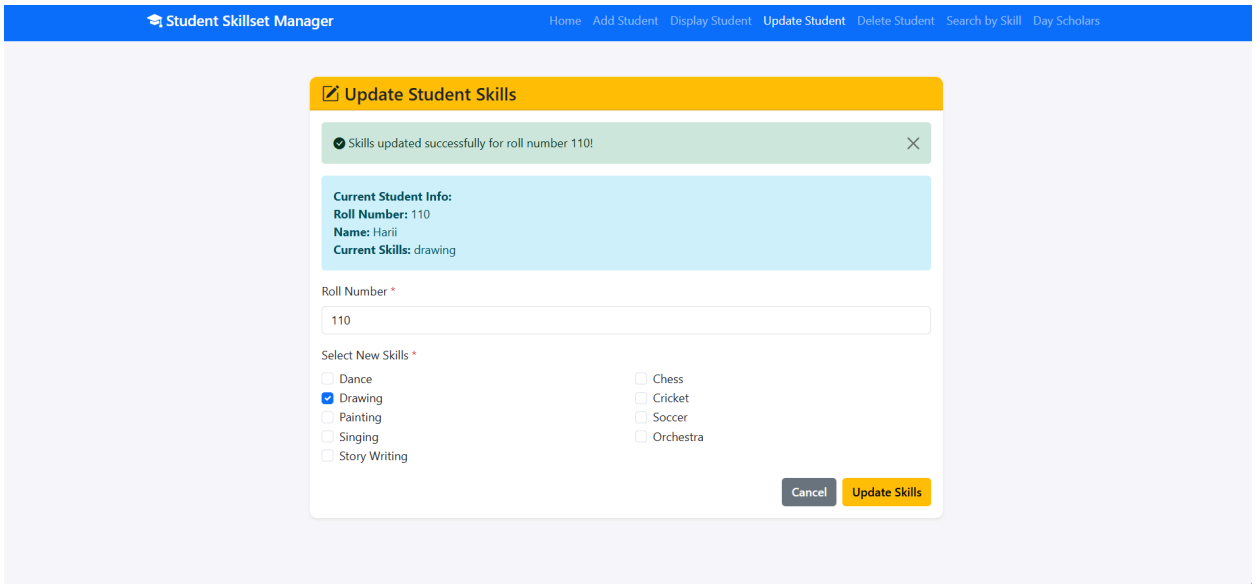
```
                                <button type="submit" class="btn
btn-primary">Add Student</button>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle
.min.js"></script>
    <script src="/scripts/validation.js"></script>
</body>
</html>
```

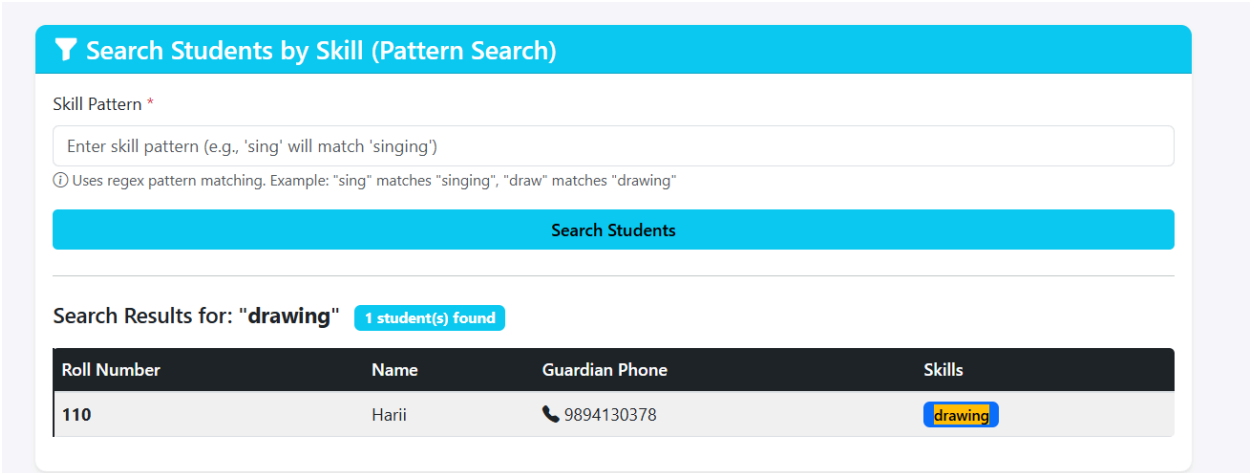
OUTPUT:

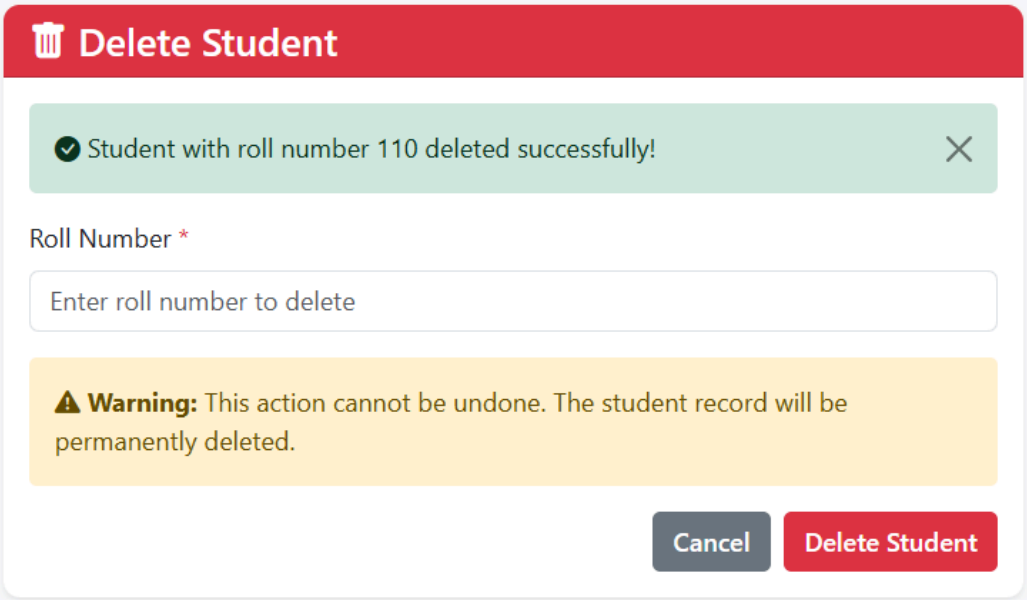

Student Skillset Manager
Home
Add Student
Display Student
Update Student
Delete Student
Search by Skill
Day Scholars
Welcome to Student Skillset Manager
Manage student skills and information efficiently
Add Student
Add a new student with their skills and details
Add Student
Display Student
View student details by roll number
Display Student
Update Student
Update student skills by roll number
Update Student
Delete Student
Remove a student from the database
Delete Student
Search by Skill
Find students with specific skills (pattern search)
Search by Skill
Day Scholars
View all students without guardian phone
View Day Scholars
© 2025 | 25MX110 | Student Skillset Manager. All rights reserved.

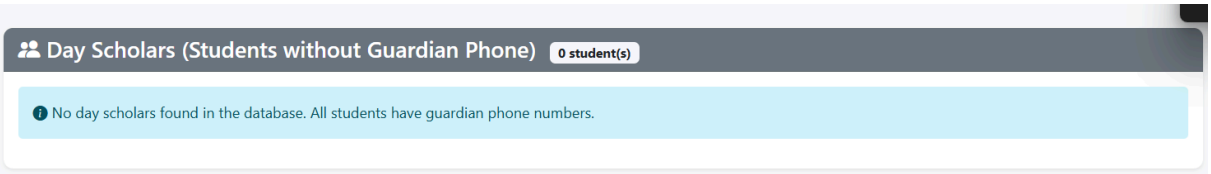

Student Skillset Manager
Home
Add Student
Display Student
Update Student
Delete Student
Search by Skill
Day Scholars
Add New Student
Roll Number *
Enter roll number
Name *
Enter student name
Guardian Phone (Optional)
10-digit Indian mobile number
10-digit Indian mobile number (starting with 6-9). Leave empty for day scholars.
Skills *
Dance
Drawing
Painting
Singing
Story Writing
Chess
Cricket
Soccer
Orchestra
Reset
Add Student


Student Skillset Manager
Home
Add Student
Display Student
Update Student
Delete Student
Search by Skill
Day Scholars
Display Student
Roll Number *
Enter roll number to search
Search Student
Student Details
Roll Number
110
Name
Harii
Guardian Phone
9894130378
Skills
cricket


Student Skillset Manager
Home
Add Student
Display Student
Update Student
Delete Student
Search by Skill
Day Scholars
Update Student Skills
Skills updated successfully for roll number 110!
Current Student Info:
Roll Number: 110
Name: Harii
Current Skills: drawing
Roll Number *
110
Select New Skills *
Dance
Drawing
Painting
Singing
Story Writing
Chess
Cricket
Soccer
Orchestra
Cancel
Update Skills


Search Students by Skill (Pattern Search)
Skill Pattern *
Enter skill pattern (e.g., 'sing' will match 'singing')
Uses regex pattern matching. Example: "sing" matches "singing", "draw" matches "drawing"
Search Students
Search Results for: "drawing"
1 student(s) found
Roll Number
Name
Guardian Phone
Skills
110
Harii
9894130378
drawing


Delete Student
Student with roll number 110 deleted successfully!
Roll Number *
Enter roll number to delete
Warning: This action cannot be undone. The student record will be permanently deleted.
Cancel
Delete Student


Day Scholars (Students without Guardian Phone)
0 student(s)
No day scholars found in the database. All students have guardian phone numbers.