Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

VI CompeTe

BCSE203E - WEB PROGRAMMING

PROJECT DOCUMENT

Winter Semester 2024 - 2025

24BCI0080
Kshitij Rahul Manikshete

UNDER THE GUIDENCE OF
Dr. KAUSER AHMED P SCOPE

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

DECLARATION

I/We hereby declare that the project entitled “VI CompeTe” submitted by me/us to the School of Computer Science and Engineering, VIT University, Vellore-14 in partial fulfillment of the requirements for the Web Programming (BCSE203E) course, is a record of bonafide work carried out by me under the supervision of Dr.Kauser Ahmed, Assistant Professor Senior. I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other course or degree or diploma of this institute or of any other institute or university.

4.2.2. Sample code

4.2.3. Sample Screenshot

4.3. Etc..,

5. Screenshots

Selected frontend design - excluding recent technology and existing problem solution screenshots.

6. Implementation Code
7. Conclusion

Mark Distribution

Topics Covers	Weightage
dopting Recent Technologies (Minimum 2)	2.5
Finding Solution for existing problems (Minimum 2)	2.5
HTML5 / CSS/ Java Script	5
Demo, Viva & Report	10
Total	20

Minimum 30 Pages

1. Introduction

1.1. Aim

To create a community-driven platform, **VI compeTe**, that fosters healthy academic competition among VIT students by enabling performance tracking, group-based learning, and data-driven insights.

1.2. Objective

- Facilitate study groups for collaborative learning.

- Provide statistical comparisons (class toppers, branch rankings, expected performance).

- Motivate students through predictive analytics (CGPA forecasts, S-grade probability).

2. System Requirements (Software Requirements)

- Basic Browser such as Internet Explorer, Chrome and Firefox.
- Animation on Safari might not be as smooth.
- Basic requirements such as support for HTML, CSS and Javascript.

3. Recent Technologies Adapted

3.1 Technology-1:

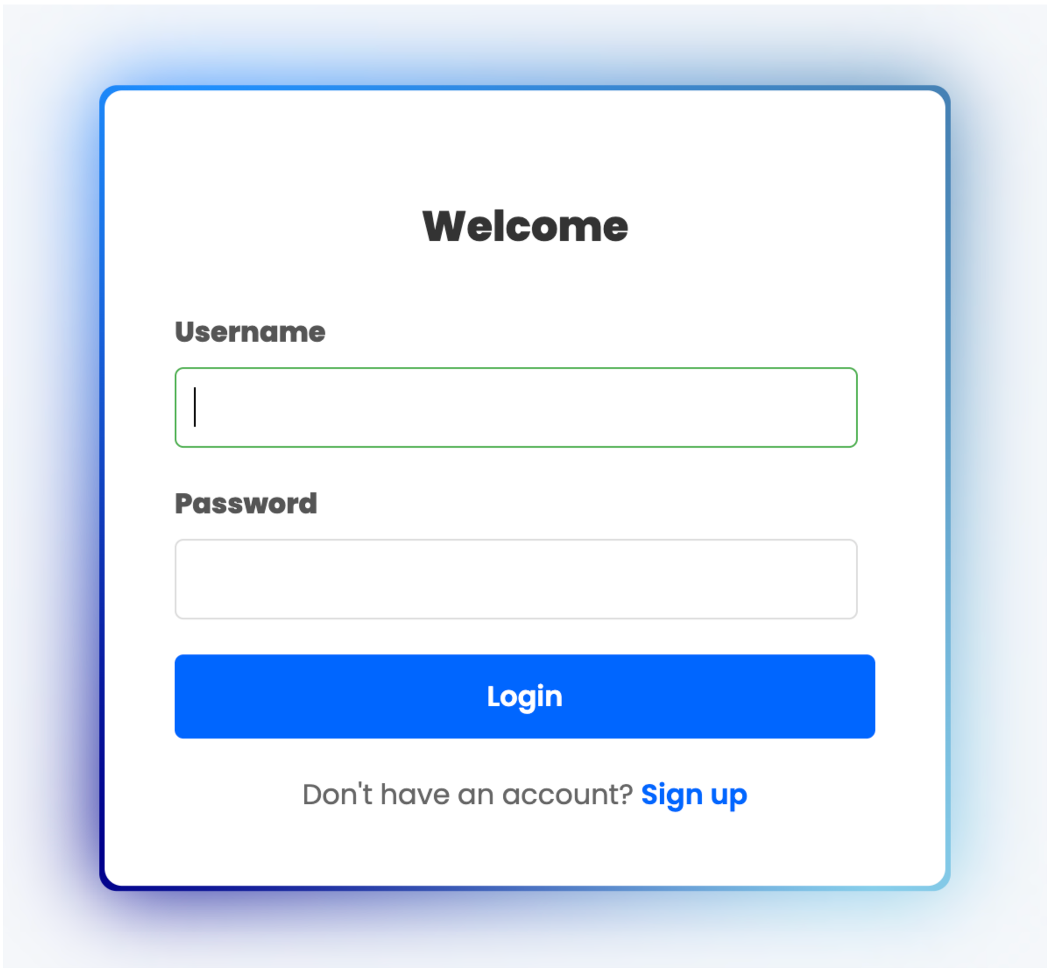
3.1.1 Description – Advanced CSS to create Animation

One of the recent technology used in this project is in CSS the use of @property rule and its implementation with conical gradient and pseudo element to create the animation present on the login page where we use a pseudo element of the login container called **::after** and **::before** and we use them to implement the **conical gradient** background and them we use the **@property** rule to declare the **angle(--angle)** variable and make it variable with the help of keyframes that are used in animation.

3.1.2 Sample Code : -

```css
@property --angle{
    syntax: "<angle>";
    initial-value: 0deg;
    inherits: false;
}
.login-container::after {
    content: "";
    height: 100%;
    width: 100%;
    position: absolute;
    top: 50%;
    left: 50%;
    border-radius: 10px;
    translate: -50% -50%;
    z-index: -2;
    padding: 3px;
    background-image: conic-gradient(from var(--angle), #1E90FF,
#4682B4, #87CEEB, #00008B, #1E90FF);
    animation: 4s spin linear infinite;
    border-radius: 10px;
}
.login-container::before {
    content: "";
    height: 100%;
    width: 100%;
position: absolute;
    top: 50%;
    left: 50%;
    translate: -50% -50%;
    z-index: -3;
    background-image: conic-gradient(from var(--angle), #1E90FF,
#4682B4, #87CEEB, #00008B, #1E90FF);
    animation: 4s spin linear infinite;
    border-radius: 10px;
    filter: blur(1.5rem);
    opacity: 1;
    padding: 3px;
}
@keyframes spin {
    from{
        --angle: 0deg;
    }
    to{
        --angle: 360deg;
    }
}
```

3.1.3 Sample Screenshot:



3.2 Technology – 2

3.2.1 Description – Various Advanced JavaScript ES6+ features

In this webpage, I implemented several modern JavaScript technologies to create an interactive user experience. The application uses the Web Storage API (**local Storage**) to persist user data between sessions, eliminating the need for constant server requests.

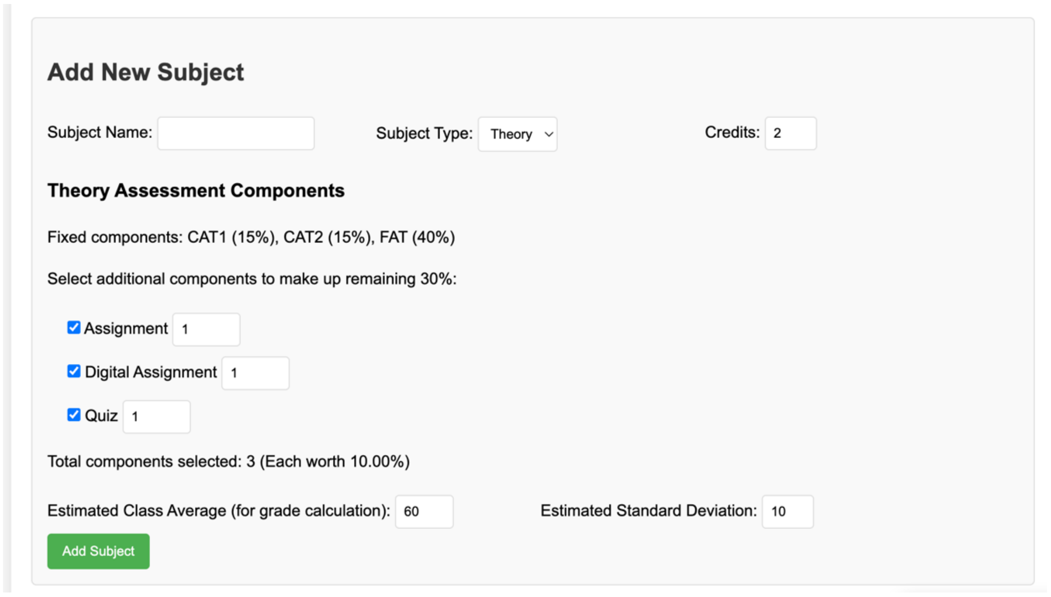
I employed **ES6+ features** like **arrow functions**, template literals, and destructuring to write cleaner, more maintainable code. The DOM is manipulated dynamically based on user actions and stored data, with event listeners handling all user interactions.

The architecture follows single-page application principles, with JavaScript routing determining which content to display based on the URL path. Data validation ensures only valid information is stored, and complex operations like grade calculations are handled client-side for immediate feedback."

3.2.2 Sample Code

```
const handleLogin = (e) => {
    e.preventDefault();

    // Destructuring form values
    const { value: username } = document.getElementById('username');
    const { value: password } = document.getElementById('password');

    if (!username || !password) {
        alert('Please enter both username and password');
        return;
    }

    // Object shorthand property names
    userData.currentUser = { username, program: 'BTech', batch: '2023',
class: 'CSE-A' };

    // localStorage with error handling
    try {
        localStorage.setItem('userData', JSON.stringify({
            ...userData, // Spread existing data
            lastLogin: new Date().toISOString() // Add new property
        }));
        window.location.href = 'dashboard.html';
    } catch (e) {
        console.error('Storage failed:', e);
        alert('Login failed - please try again');
    }
};


document.getElementById('loginForm')?.addEventListener('submit',
handleLogin);
```

3.2.3 Sample Screenshot



Add New Subject

Subject Name:

Subject Type: Theory

Credits: 2

Theory Assessment Components

Fixed components: CAT1 (15%), CAT2 (15%), FAT (40%)

Select additional components to make up remaining 30%:

☑ Assignment 1

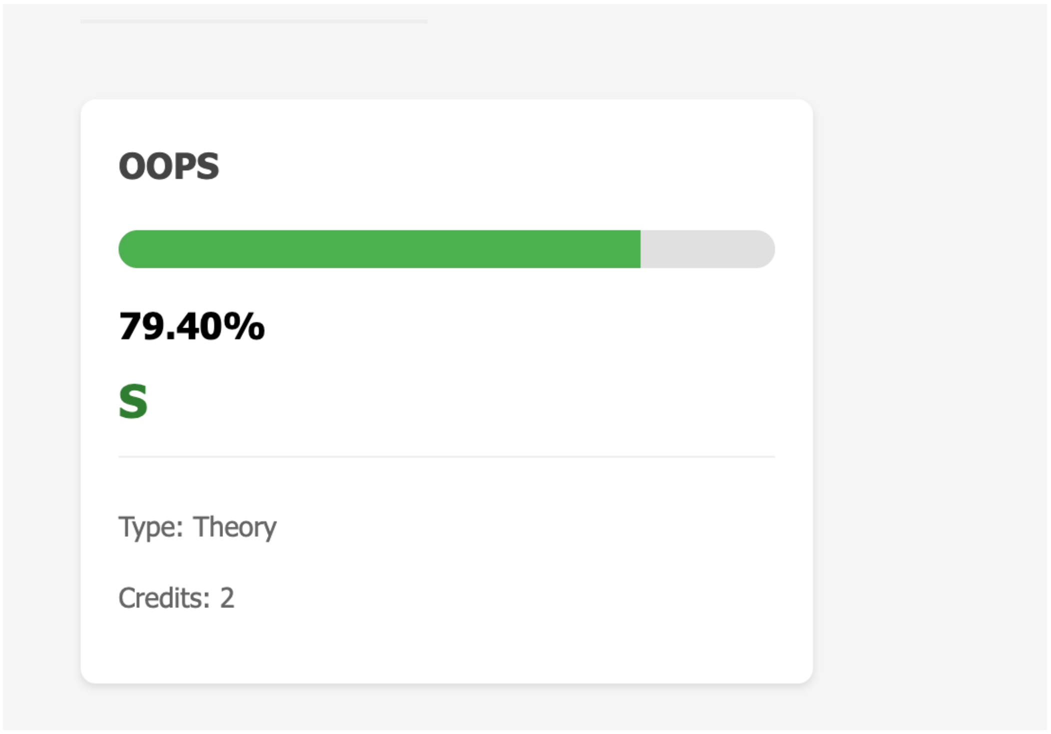
☑ Digital Assignment 1

☑ Quiz 1

Total components selected: 3 (Each worth 10.00%)

Estimated Class Average (for grade calculation): 60

Estimated Standard Deviation: 10



OOPS

79.40%

S

Type: Theory

Credits: 2

4. Solution for Existing problems

4.1 Solution -1

4.1.1 Problem and Solution Description

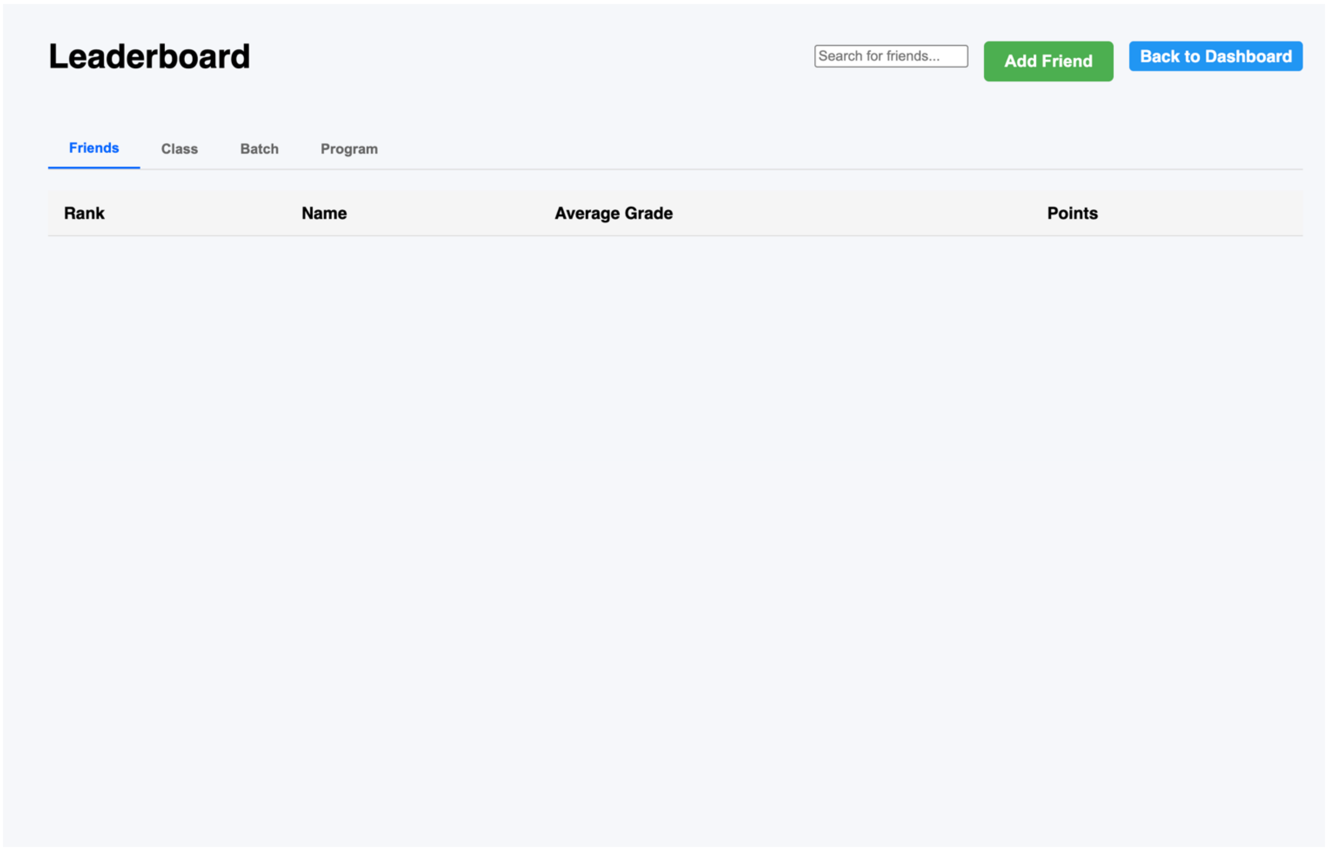
The problem that a lot of students have today is they cannot focus on their studies properly and they get distracted very easily and tend to push the deadline to the very last minute this website is where people can see their friends do progress in their academics and then get motivated to do the same and this will improve the performance of students and will be a motivation factor and a platform for friendly competitions. Therefore the leaderboard feature in the website will be a great addition because it will make the students to compete with their friends in VIT and that's where the name **"VI compeTe"** came from it means **"We compete in VIT".**

4.1.2 Sample Code

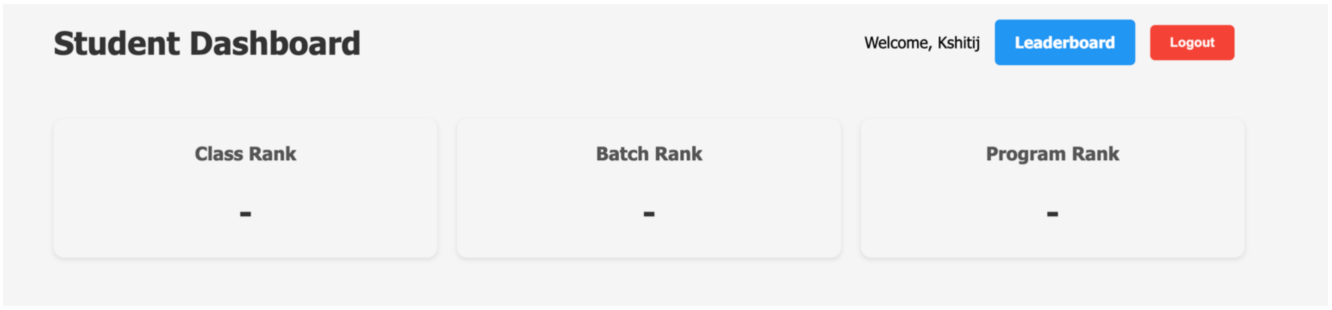
```
                    <!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Leaderboard</title>
    <link rel="stylesheet" href="styles/main.css">
</head>

<body>
    <div class="leaderboard-container">
        <header class="leaderboard-header">
            <h1>Leaderboard</h1>
            <div class="leaderboard-controls">
                <input type="text" id="search-friend"
placeholder="Search for friends...">
                <button id="add-friend-btn" class="primary-btn">Add
Friend</button>
                <a href="dashboard.html" class="back-btn">Back to
Dashboard</a>
            </div>
        </header>

        <div class="tabs">
            <button class="tab-btn active" data-
tab="friends">Friends</button>
```

```html
            <button class="tab-btn" data-tab="class">Class</button>
            <button class="tab-btn" data-tab="batch">Batch</button>
            <button class="tab-btn" data-tab="program">Program</button>
        </div>

        <div class="leaderboard-content">
            <div class="tab-content active" id="friends-tab">
                <table>
                    <thead>
                        <tr>
                            <th>Rank</th>
                            <th>Name</th>
                            <th>Average Grade</th>
                            <th>Points</th>
                        </tr>
                    </thead>
                    <tbody id="friends-list">

                    </tbody>
                </table>
            </div>

        </div>
    </div>

    <script src="js/main.js"></script>
</body>

</html>
```

4.1.3 Sample Screenshot



Rank	Name	Average Grade	Points



4.2 Solution – 2

4.2.1 Problem and Solution Description

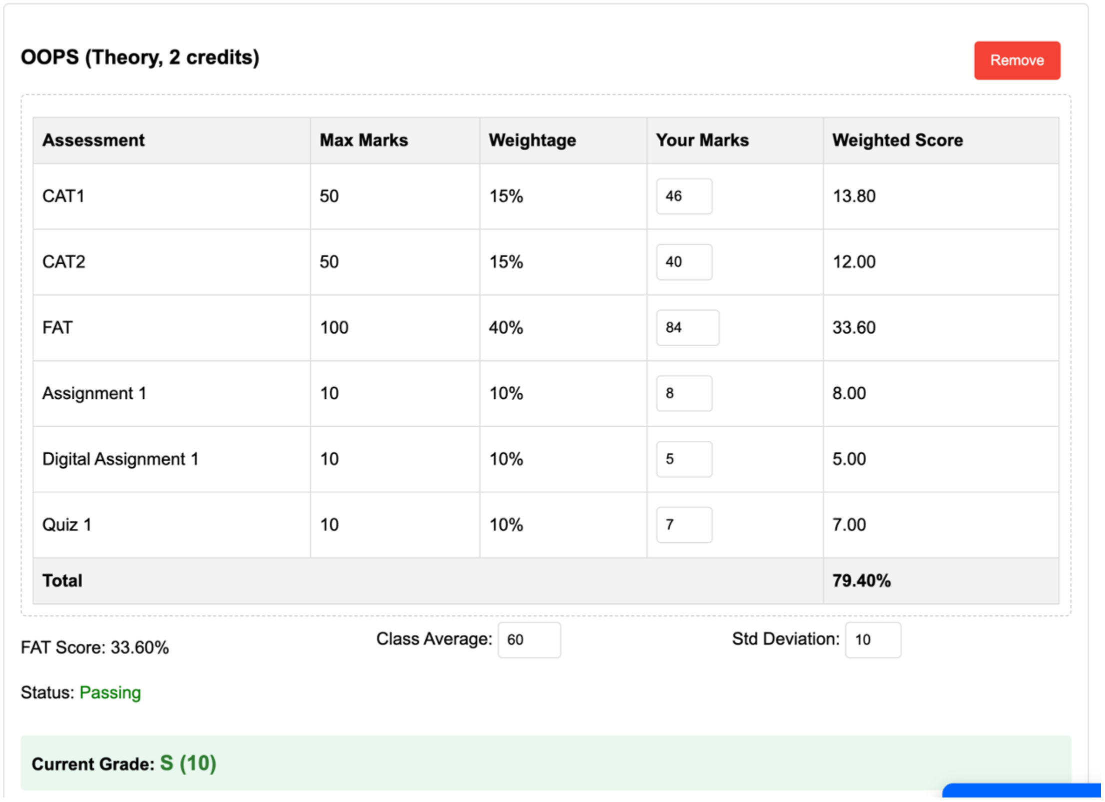
Many students especially in their first year and first semester are confused about the grading system and are just not familiar until the first semester get by. Therefore the students need a better visualization of the grades and what are the components that the subjects is requiring to get a particular grade in a subject. This is why at VI compeTe I have established a **CGPA and Grade calculator** which helps students understand the assessment techniques used in VIT and I have also coded **algorithm** for **standard deviation** for the Theory components as there the **class average** is taken and then the grades are given based upon the standard deviation of the marks of the students.

4.2.2 Sample Code

```
function calculateGrade(subject, totalWeighted) {
            if (subject.type === 'lab') {

                if (totalWeighted >= 90) return 'S';
                if (totalWeighted >= 80) return 'A';
                if (totalWeighted >= 70) return 'B';
                if (totalWeighted >= 60) return 'C';
                if (totalWeighted >= 55) return 'D';
                if (totalWeighted >= 50) return 'E';
                return 'F';
            } else {

                if (!subject.classAverage || !subject.stdDev) return
'N/A';

                const deviation = (totalWeighted - subject.classAverage)
/ subject.stdDev;

                if (deviation >= 1.5) return 'S';
                if (deviation >= 0.5) return 'A';
                if (deviation >= -0.5) return 'B';
                if (deviation >= -1.0) return 'C';
                if (deviation >= -1.5) return 'D';
                if (deviation >= -2.0) return 'E';
                return 'F';
            }
        }
```

```
function getGradePoint(grade) {
    switch (grade) {
        case 'S': return '10';
        case 'A': return '9';
        case 'B': return '8';
        case 'C': return '7';
        case 'D': return '6';
        case 'E': return '5';
        case 'F': return '0';
        default: return 'N/A';
    }
}
```

4.2.3 Screenshot



OOPS (Theory, 2 credits)

Assessment	Max Marks	Weightage	Your Marks	Weighted Score
CAT1	50	15%	46	13.80
CAT2	50	15%	40	12.00
FAT	100	40%	84	33.60
Assignment 1	10	10%	8	8.00
Digital Assignment 1	10	10%	5	5.00
Quiz 1	10	10%	7	7.00
Total				**79.40%**

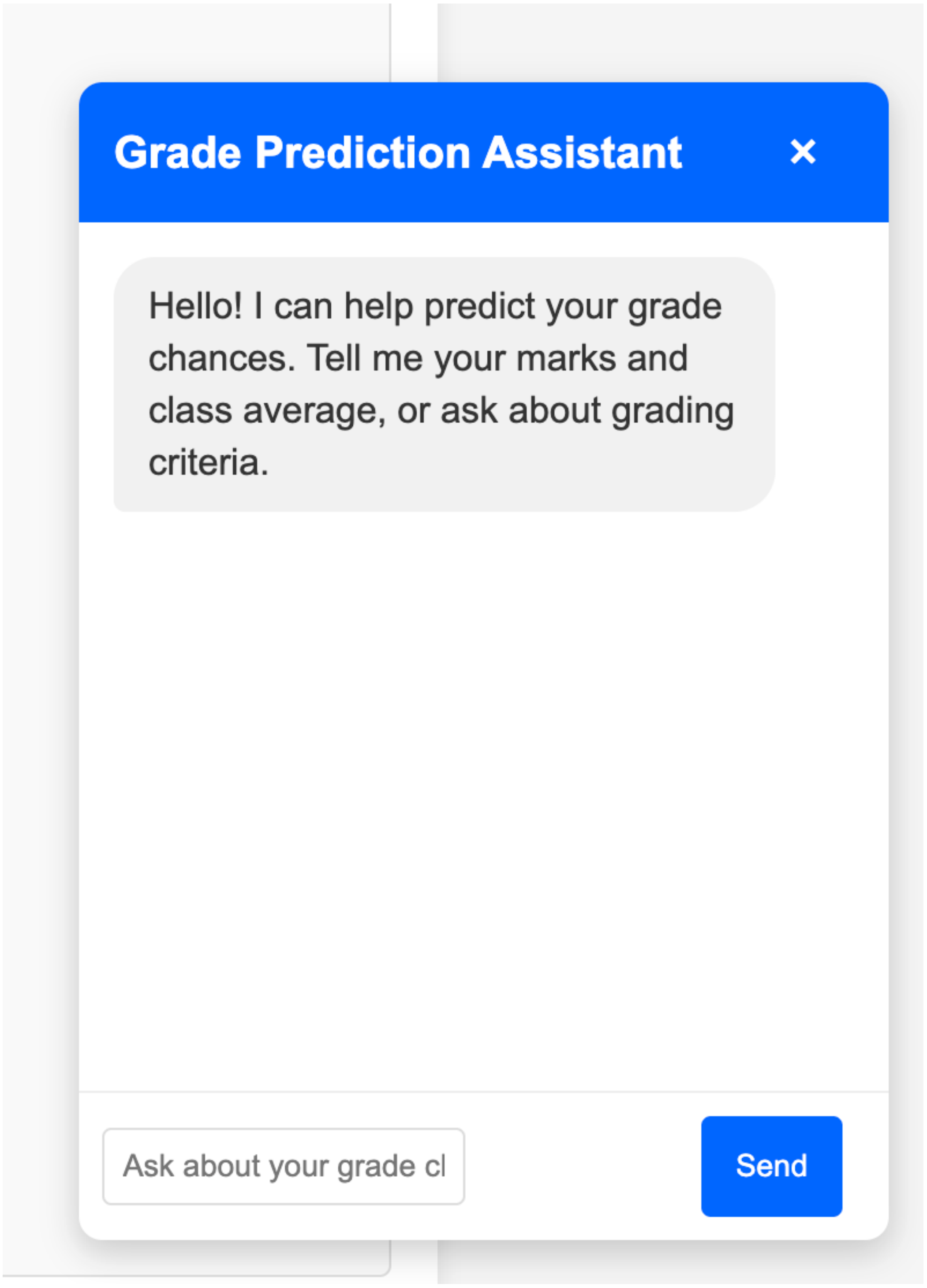
FAT Score: 33.60%   Class Average: 60   Std Deviation: 10

Status: Passing

Current Grade: S (10)

5. Screenshots

This is the Chatbot which is implemented in the website to make it more user friendly and more understandable for new users and it can also tell your grade in a specific subject if you tell it your marks.



Grade Prediction Assistant

Hello! I can help predict your grade chances. Tell me your marks and class average, or ask about grading criteria.

6. Implementation Code

Index.html –

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>VI compeTe Logo</title>
    <link rel="stylesheet" href="styles/main.css">
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600;800&
display=swap" rel="stylesheet">
</head>

<body>
    <div class="logo-container">
        <div class="vit-part">VI</div>
        <div class="competext">&nbsp;compe<span class="vit-
part">T</span>e</div>
    </div>

    <div class="login-container">
        <div class="login-form">
            <h2 class="box">Welcome</h2>
            <form id="loginForm">
                <div class="form-group">
                    <label for="username">Username</label>
                    <input type="text" id="username" name="username"
required>
                </div>
                <div class="form-group">
                    <label for="password">Password</label>
                    <input type="password" id="password" name="password"
required>
                </div>
                <button type="submit" class="login-btn">Login</button>
                <div class="signup-link">
                    Don't have an account? <a href="#">Sign up</a>
                </div>
            </form>
        </div>
    </div>
    <script src="js/main.js"></script>
</body>

</html>
```

dashboard.html –

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Student Dashboard</title>
    <link rel="stylesheet" href="styles/main.css">
    <link rel="stylesheet" href="styles/calculator.css">
</head>

<body>
    <div class="dashboard-container">
        <header class="dashboard-header">
            <h1>Student Dashboard</h1>
            <div class="user-controls">
                <span id="username-display">Welcome, User</span>
                <a href="leaderboard.html" class="leaderboard-
btn">Leaderboard</a>
                <a href="index.html"><button id="logout-btn"
class="logout-btn">Logout</button></a>
            </div>
        </header>

        <div class="stats-container">
            <div class="stat-card">
                <h3>Class Rank</h3>
                <p id="class-rank">-</p>
            </div>
            <div class="stat-card">
                <h3>Batch Rank</h3>
                <p id="batch-rank">-</p>
            </div>
            <div class="stat-card">
                <h3>Program Rank</h3>
                <p id="program-rank">-</p>
            </div>
        </div>
```

```html
        <section class="grades-section">
            <div class="section-header">
                <h2>Current Grades</h2>
                <a href="calculator.html"><button id="add-subject-btn"
class="primary-btn">Add New Subject</button></a>
            </div>
            <div class="grades-container" id="grades-container">
                <
            </div>
        </section>

        <section class="recent-section">
            <h2>Recent Assessments</h2>
            <div class="marks-container" id="recent-assessments">

            </div>
        </section>
    </div>

    <div id="chatbot-container" class="chatbot-container">
        <div class="chatbot-header">
            <h3>Grade Prediction Assistant</h3>
            <button id="chatbot-toggle" class="chatbot-
toggle">×</button>
        </div>
        <div id="chatbot-messages" class="chatbot-messages"></div>
        <div class="chatbot-input">
            <input type="text" id="chatbot-input-field" placeholder="Ask
about your grade chances...">
            <button id="chatbot-send">Send</button>
        </div>
    </div>
    <button id="chatbot-launch" class="chatbot-launch">Grade
Help</button>

    <script src="js/main.js"></script>
    <script src="js/calculator.js"></script>
</body>

</html>
```

Leaderboard.html –

```html
<!DOCTYPE html>
<html lang="en">

<head>
```

```html
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Leaderboard</title>
    <link rel="stylesheet" href="styles/main.css">
</head>

<body>
    <div class="leaderboard-container">
        <header class="leaderboard-header">
            <h1>Leaderboard</h1>
            <div class="leaderboard-controls">
                <input type="text" id="search-friend"
placeholder="Search for friends...">
                <button id="add-friend-btn" class="primary-btn">Add
Friend</button>
                <a href="dashboard.html" class="back-btn">Back to
Dashboard</a>
            </div>
        </header>

        <div class="tabs">
            <button class="tab-btn active" data-
tab="friends">Friends</button>
            <button class="tab-btn" data-tab="class">Class</button>
            <button class="tab-btn" data-tab="batch">Batch</button>
            <button class="tab-btn" data-tab="program">Program</button>
        </div>

        <div class="leaderboard-content">
            <div class="tab-content active" id="friends-tab">
                <table>
                    <thead>
                        <tr>
                            <th>Rank</th>
                            <th>Name</th>
                            <th>Average Grade</th>
                            <th>Points</th>
                        </tr>
                    </thead>
                    <tbody id="friends-list">

                    </tbody>
                </table>
            </div>

        </div>
    </div>
    <script src="js/main.js"></script>
</body>

</html>
```

Main.css –

```
body {
  margin: 0;
  padding: 0;
  font-family: 'Poppins', sans-serif;
  overflow-x: hidden;
  background-color: #f5f7fa;
}

/* Logo styles */
.logo-container {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  display: flex;
  align-items: center;
  transition: all 1s ease;
  z-index: 10;
}

.logo-container.move-up {
  top: 20px;
  left: 50%;
  transform: translateX(-50%);
}

.vit-part {
  font-size: 4rem;
  font-weight: 800;
  color: #0066ff;
}

.competext {
  font-size: 4rem;
  font-weight: 800;
  color: #000000;
  overflow: hidden;
  white-space: nowrap;
```

```css
  width: 0;
  animation: extend 1.5s ease-out forwards;
  animation-delay: 0.5s;
}

@keyframes extend {
  0% { width: 0; opacity: 0; }
  100% { width: 340px; opacity: 1; }
}

/* Login form styles */
.login-container {
  display: none;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: 100%;
  max-width: 400px;
  padding: 40px;
  background: white;
  border-radius: 10px;
  z-index: 5;
  opacity: 0;
  transition: opacity 0.5s ease;
}

@property --angle{
    syntax: "<angle>";
    initial-value: 0deg;
    inherits: false;
}

.login-container::after {
    content: "";
    height: 100%;
    width: 100%;
    position: absolute;
    top: 50%;
    left: 50%;
    border-radius: 10px;
    translate: -50% -50%;
    z-index: -2;
    padding: 3px;
    background-image: conic-gradient(from var(--angle), #1E90FF,
#4682B4, #87CEEB, #00008B, #1E90FF);
    animation: 4s spin linear infinite;
    border-radius: 10px;
```

```css
}

@keyframes spin {
    from{
        --angle: 0deg;
    }
    to{
        --angle: 360deg;
    }
}

.box::after {
    content: "";
    height: 100%;
    width: 100%;
    position: absolute;
    top: 50%;
    left: 50%;
    translate: -50% -50%;
    background-color: white;
    z-index: -1;
    border-radius: 10px;
}

.login-container::before {
    content: "";
    height: 100%;
    width: 100%;
    position: absolute;
    top: 50%;
    left: 50%;
    translate: -50% -50%;
    z-index: -3;
    background-image: conic-gradient(from var(--angle), #1E90FF,
#4682B4, #87CEEB, #00008B, #1E90FF);
    animation: 4s spin linear infinite;
    border-radius: 10px;
    filter: blur(1.5rem);
    opacity: 1;
    padding: 3px;
}

.login-container.show {
  display: block;
  opacity: 1;
  background-color: white;

}
```

```css
.login-form h2 {
  margin-bottom: 30px;
  color: #333;
  text-align: center;
}

.form-group {
  margin-bottom: 20px;
  width: 92%;
  max-width: 400px;
}

.form-group label {
  display: block;
  margin-bottom: 8px;
  color: #555;
  font-weight: 600;
}

.form-group input {
  width: 100%;
  padding: 12px 15px;
  border: 1px solid #ddd;
  border-radius: 5px;
  font-family: 'Poppins', sans-serif;
  font-size: 14px;
  transition: border 0.3s;
}

.form-group input:focus {
  border-color: #0066ff;
  outline: none;
}

.login-btn {
  width: 100%;
  padding: 12px;
  background-color: #0066ff;
  color: white;
  border: none;
  border-radius: 5px;
  font-family: 'Poppins', sans-serif;
  font-size: 16px;
  font-weight: 600;
  cursor: pointer;
  transition: background-color 0.3s;
}
```

```css
.login-btn:hover {
  background-color: #0052cc;
}

.signup-link {
  margin-top: 20px;
  text-align: center;
  color: #666;
}

.signup-link a {
  color: #0066ff;
  text-decoration: none;
  font-weight: 600;
}

.signup-link a:hover {
  text-decoration: underline;
}
/* Dashboard Styles */
.dashboard-container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

.dashboard-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 30px;
}

.logout-btn {
    background-color: #f44336;
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
    font-weight: bold;
}

.stats-container {
    display: flex;
    justify-content: space-between;
    margin-bottom: 30px;
    gap: 20px;
}

.stat-card {
    flex: 1;
    background-color: #f5f5f5;
    padding: 20px;
    border-radius: 10px;
    text-align: center;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}

.stat-card h3 {
    color: #555;
    margin-top: 0;
}

.stat-card p {
    font-size: 2rem;
    font-weight: bold;
    color: #333;
    margin: 10px 0 0;
}

.marks-section, .grades-section {
    margin-bottom: 30px;
}

.marks-section h2, .grades-section h2 {
    color: #333;
    border-bottom: 2px solid #eee;
    padding-bottom: 10px;
}

.marks-container {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
    gap: 20px;
    margin: 20px 0;
}

.marks-card {
    background-color: white;
    border-radius: 8px;
    padding: 15px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}
```

```css
.marks-card h3 {
    margin-top: 0;
    color: #444;
}

.marks-card .marks {
    font-size: 1.5rem;
    font-weight: bold;
    color: #2e7d32;
    margin: 10px 0 0;
}

.grades-container {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
    gap: 20px;
    margin-top: 20px;
}

.grade-card {
    background-color: white;
    border-radius: 8px;
    padding: 20px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}

.grade-card h3 {
    margin-top: 0;
    color: #444;
}

.grade-progress {
    height: 20px;
    background-color: #e0e0e0;
    border-radius: 10px;
    margin: 15px 0;
    overflow: hidden;
}

.progress-bar {
    height: 100%;
    background-color: #4caf50;
}

.grade-score {
    font-size: 1.2rem;
    font-weight: bold;
    margin: 5px 0;
}

.grade-letter {
    font-size: 1.5rem;
    font-weight: bold;
    color: #2e7d32;
    margin: 5px 0 0;
}

.primary-btn {
    background-color: #4caf50;
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
    font-weight: bold;
    font-size: 1rem;
    margin-top: 10px;
}

/* Marks Form Styles */
.marks-form-container {
    max-width: 600px;
    margin: 0 auto;
    padding: 20px;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

.form-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 30px;
}

.back-btn {
    background-color: #2196F3;
    color: white;
    text-decoration: none;
    padding: 10px 20px;
    border-radius: 5px;
    font-weight: bold;
}

#marks-form {
    background-color: white;
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}

.form-group {
    margin-bottom: 20px;
}

.form-group label {
    display: block;
    margin-bottom: 8px;
    font-weight: bold;
    color: #555;
}

.form-group input,
.form-group select {
    width: 100%;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 5px;
    font-size: 1rem;
}

.form-group input:focus,
.form-group select:focus {
    outline: none;
    border-color: #4caf50;
}

@media (max-width: 768px) {
    .stats-container {
        flex-direction: column;
    }

    .marks-container, .grades-container {
        grid-template-columns: 1fr;
    }
}
/* Add these to your existing styles.css */

/* Leaderboard Styles */
.leaderboard-container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
}
```

```css
.leaderboard-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 30px;
}

.leaderboard-controls {
    display: flex;
    gap: 15px;
    align-items: center;
}

.tabs {
    display: flex;
    border-bottom: 1px solid #ddd;
    margin-bottom: 20px;
}

.tab-btn {
    padding: 10px 20px;
    background: none;
    border: none;
    cursor: pointer;
    font-weight: bold;
    color: #666;
}

.tab-btn.active {
    color: #0066ff;
    border-bottom: 2px solid #0066ff;
}

.tab-content {
    display: none;
}

.tab-content.active {
    display: block;
}

table {
    width: 100%;
    border-collapse: collapse;
}

th, td {
    padding: 12px 15px;
    text-align: left;
    border-bottom: 1px solid #ddd;
}

th {
    background-color: #f5f5f5;
    font-weight: bold;
}

/* Dashboard Additions */
.user-controls {
    display: flex;
    align-items: center;
    gap: 15px;
}

.leaderboard-btn {
    background-color: #2196F3;
    color: white;
    text-decoration: none;
    padding: 10px 20px;
    border-radius: 5px;
    font-weight: bold;
}

.section-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
}

.calculator-header {

    align-items: center;
    padding: 15px 20px;
    background-color: #f5f7fa;
    border-bottom: 1px solid #ddd;
}

.calculator-header h1 {
    margin: 0 auto;
    color: #0066ff;
    margin-left: 40%;
}

.back-btn {
    color: #ffffff;
    text-decoration: none;
    font-weight: bold;
    padding: 5px 10px;
    border-radius: 4px;
}

.back-btn:hover {
    background-color: #e6f0ff;
}

/* already added i think grade css*/
/* Add these styles to your main.css */
.grades-container {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
    gap: 20px;
    margin-top: 20px;
}

.grade-card {
    background: white;
    border-radius: 8px;
    padding: 20px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.subject-details {
    margin-top: 10px;
    font-size: 0.9em;
    color: #666;
    border-top: 1px solid #eee;
    padding-top: 10px;
}

.grade-progress {
    height: 20px;
    background-color: #e0e0e0;
    border-radius: 10px;
    margin: 15px 0;
    overflow: hidden;
}

.progress-bar {
    height: 100%;
    background-color: #4caf50;
    transition: width 0.3s ease;
}
```

```
.grade-score {
    font-weight: bold;
    margin: 5px 0;
}

.grade-letter {
    font-size: 1.5rem;
    font-weight: bold;
    color: #2e7d32;
    margin: 5px 0 0;
}
```

Main.js –

```
// Global data structure
const userData = {
    currentUser: null,
    subjects: [],
    friends: [],
    rankings: {
        class: [],
        batch: [],
        program: []
    },
    studentMarks: []
};

document.addEventListener('DOMContentLoaded', function() {
    const path = window.location.pathname;

    if (path.endsWith('index.html') || path === '/') {
        handleLoginPage();
    } else if (path.includes('dashboard.html')) {
        handleDashboardPage();
    } else if (path.includes('marks.html')) {
        handleMarksPage();
    } else if (path.includes('leaderboard.html')) {
        handleLeaderboardPage();
    } else if (path.includes('calculator.html')) {
    }
});

// LOGIN PAGE
function handleLoginPage() {
    // Animation handling
    setTimeout(function() {
        const logoContainer = document.querySelector('.logo-container');
        if (logoContainer) logoContainer.classList.add('move-up');

        setTimeout(function() {
            const loginContainer = document.querySelector('.login-
container');
            if (loginContainer) loginContainer.classList.add('show');
        }, 1000);
    }, 2500);

    // Form submission
    const loginForm = document.getElementById('loginForm');
    if (loginForm) {
        loginForm.addEventListener('submit', function(e) {
            e.preventDefault();

            const username = document.getElementById('username').value;
            const password = document.getElementById('password').value;

            if (!username || !password) {
                alert('Please enter both username and password');
                return;
            }

            // Set user data
            userData.currentUser = {
                username: username,
                program: 'BTech',
                batch: '2023',
                class: 'CSE-A'
            };

            // Save to storage
            localStorage.setItem('loggedIn', 'true');
            localStorage.setItem('username', username);
            localStorage.setItem('userData', JSON.stringify(userData));
            window.location.href = 'dashboard.html';
        });
    }
}

// DASHBOARD PAGE
function handleDashboardPage() {
    // Check login first
    if (!localStorage.getItem('loggedIn')) {
        window.location.href = 'index.html';
        return;
    }
    // Load fresh data from localStorage
    const savedData = localStorage.getItem('userData');
    if (savedData) {
        try {
            Object.assign(userData, JSON.parse(savedData));
        } catch (e) {
            console.error("Error parsing user data:", e);
        }
    }

    // Setup UI and load content
    setupDashboardUI();
    loadDashboardContent();
}

function loadUserData() {
    try {
        const savedData = localStorage.getItem('userData');
        if (savedData) {
            const parsed = JSON.parse(savedData);
            Object.assign(userData, parsed);

            userData.subjects = userData.subjects || [];
            userData.studentMarks = userData.studentMarks || [];
        }
    } catch (e) {
        console.error("Error loading user data:", e);
        userData.subjects = [];
        userData.studentMarks = [];
    }
}

function setupDashboardUI() {
    // Welcome message
    const usernameElement = document.getElementById('username-display');
    if (usernameElement) {
        usernameElement.textContent = `Welcome,
${userData.currentUser.username}`;
    }

    // Logout button
    const logoutBtn = document.getElementById('logout-btn');
    if (logoutBtn) {
        logoutBtn.addEventListener('click', function() {
            localStorage.clear();
            window.location.href = 'index.html';
        });
    }
    // Calculator button
    const calculatorBtn = document.getElementById('calculator-btn');
    if (calculatorBtn) {
        calculatorBtn.addEventListener('click', function() {
            window.location.href = 'calculator.html';
        });
    }

    // Leaderboard button
    const leaderboardBtn = document.getElementById('leaderboard-btn');
    if (leaderboardBtn) {
        leaderboardBtn.addEventListener('click', function() {
            window.location.href = 'leaderboard.html';
        });
    }
    const addSubjectBtn = document.getElementById('add-subject-btn');
    if (addSubjectBtn) {
        addSubjectBtn.addEventListener('click', function() {
            window.location.href = 'calculator.html';
        });
    }
}

function loadDashboardContent() {

    const savedData = localStorage.getItem('userData');
    if (savedData) {
        Object.assign(userData, JSON.parse(savedData));
    }
    if (userData.studentMarks.length > 0) {
        loadMarksData();
    }

    // Load subjects from calculator
    if (userData.subjects.length > 0) {
        loadSubjectGrades();
    }

    // Update rankings
    updateRankings();
    checkForNewSubjects();
}

function loadMarksData() {
    const marksContainer = document.getElementById('recent-
assessments');
    if (!marksContainer) return;
    marksContainer.innerHTML = '';
    const recentMarks = userData.studentMarks.slice(-4).reverse();

    recentMarks.forEach(mark => {
        const card = document.createElement('div');
        card.className = 'marks-card';
        card.innerHTML = `
            <h3>${capitalizeFirstLetter(mark.subject)}
${capitalizeFirstLetter(mark.assessmentType)}
${mark.assessmentNumber}</h3>
            <p
class="marks">${mark.marksObtained}/${mark.totalMarks}</p>
            <p class="marks-date">${new
Date(mark.date).toLocaleDateString()}</p>
        `;
        marksContainer.appendChild(card);
    });
}

function loadSubjectGrades() {
    const gradesContainer = document.getElementById('grades-container');
    if (!gradesContainer) return;

    // Clear existing content
    gradesContainer.innerHTML = '';

    // Check if we have subjects to display
    if (!userData.subjects || userData.subjects.length === 0) {
        gradesContainer.innerHTML = '<p>No subjects added yet. Use the
calculator to add subjects.</p>';
        return;
    }

    // Create grade cards for each subject
    userData.subjects.forEach(subject => {
        // Calculate total weighted score
        const totalWeighted = subject.assessments.reduce((sum,
assessment) => {
            const marks = assessment.marks || 0;
            return sum + (marks / assessment.maxMarks) *
assessment.weightage;
        }, 0);

        // Create the grade card element
        const gradeCard = document.createElement('div');
        gradeCard.className = 'grade-card';
        gradeCard.innerHTML = `
```

```
            <h3>${subject.name}</h3>
            <div class="grade-progress">
                <div class="progress-bar" style="width:
${totalWeighted}%"></div>
            </div>
            <p class="grade-score">${totalWeighted.toFixed(2)}%</p>
            <p class="grade-letter">${calculateGrade(subject,
totalWeighted)}</p>
            <div class="subject-details">
                <p>Type: ${subject.type === 'theory' ? 'Theory' :
'Lab'}</p>
                <p>Credits: ${subject.credit}</p>
            </div>
        `;

        gradesContainer.appendChild(gradeCard);
    });
}

//  MARKS PAGE
function handleMarksPage() {
    // Check login
    if (!localStorage.getItem('loggedIn')) {
        window.location.href = 'index.html';
        return;
    }

    // Form submission
    const marksForm = document.getElementById('marks-form');
    if (marksForm) {
        marksForm.addEventListener('submit', function(e) {
            e.preventDefault();

            const subject = document.getElementById('subject').value;
            const assessmentType = document.getElementById('assessment-
type').value;
            const assessmentNumber =
document.getElementById('assessment-number').value;
            const marksObtained =
parseFloat(document.getElementById('marks-obtained').value);
            const totalMarks =
parseFloat(document.getElementById('total-marks').value);

            if (marksObtained > totalMarks) {
                alert('Marks obtained cannot be greater than total
marks');
                return;
            }
            const percentage = (marksObtained / totalMarks) * 100;

            userData.studentMarks.push({
                subject,
                assessmentType,
                assessmentNumber,
                marksObtained,
                totalMarks,
                percentage,
                date: new Date().toISOString()
            });

            localStorage.setItem('userData', JSON.stringify(userData));
            window.location.href = 'dashboard.html';
        });
    }
}

// LEADERBOARD PAGE
function handleLeaderboardPage() {
    // Check login
    if (!localStorage.getItem('loggedIn')) {
        window.location.href = 'index.html';
        return;
    }

    // Load data
    loadUserData();

    // Setup UI
    setupLeaderboardUI();

    // Load content
    loadLeaderboardContent();
}

function setupLeaderboardUI() {
    // Tab switching
    document.querySelectorAll('.tab-btn').forEach(btn => {
        btn.addEventListener('click', function() {
            const tabId = this.getAttribute('data-tab');
            switchTab(tabId);
        });
    });

    // Friend search
    const searchInput = document.getElementById('search-friend');
    if (searchInput) {
        searchInput.addEventListener('input', function() {
            searchFriends(this.value);
        });
    }

    // Back button
    const backBtn = document.querySelector('.back-btn');
    if (backBtn) {
        backBtn.addEventListener('click', function() {
            window.location.href = 'dashboard.html';
        });
    }
}

function loadLeaderboardContent() {
    if (userData.rankings.class.length === 0) {
        updateRankings();
    }

    switchTab('friends');
}

function switchTab(tabId) {
    // Hide all tabs
    document.querySelectorAll('.tab-content').forEach(tab => {
        tab.classList.remove('active');
    });

    // Deactivate all buttons
    document.querySelectorAll('.tab-btn').forEach(btn => {
        btn.classList.remove('active');
    });

    // Activate selected tab
    const activeTab = document.getElementById(`${tabId}-tab`);
    if (activeTab) activeTab.classList.add('active');

    // Activate button
    const activeBtn = document.querySelector(`.tab-btn[data-
tab="${tabId}"]`);
    if (activeBtn) activeBtn.classList.add('active');

    // Load content for tab
    if (tabId === 'friends') {
        loadFriendsList();
    } else {
        loadRankingList(tabId);
    }
}

// HELPER FUNCTIONS
function updateRankings() {
    userData.rankings = {
        class: generateRankingData(30, userData.currentUser.username),
        batch: generateRankingData(100, userData.currentUser.username),
        program: generateRankingData(500, userData.currentUser.username)
    };

    localStorage.setItem('userData', JSON.stringify(userData));
}

function generateRankingData(count, currentUsername) {
    const rankings = [];
    for (let i = 1; i <= count; i++) {
        rankings.push({
            name: i === Math.floor(count/2) ? currentUsername :
`Student${i}`,
            score: Math.floor(Math.random() * 100),
            rank: i
        });
    }
    return rankings.sort((a, b) => b.score - a.score);
}

function calculateGrade(subject, totalWeighted) {
    if (subject.type === 'lab') {
        if (totalWeighted >= 90) return 'S';
        if (totalWeighted >= 80) return 'A';
        if (totalWeighted >= 70) return 'B';
        if (totalWeighted >= 60) return 'C';
        if (totalWeighted >= 55) return 'D';
        if (totalWeighted >= 50) return 'E';
        return 'F';
    } else {
        if (!subject.classAverage || !subject.stdDev) return 'N/A';

        const deviation = (totalWeighted - subject.classAverage) /
subject.stdDev;

        if (deviation >= 1.5) return 'S';
        if (deviation >= 0.5) return 'A';
        if (deviation >= -0.5) return 'B';
        if (deviation >= -1.0) return 'C';
        if (deviation >= -1.5) return 'D';
        if (deviation >= -2.0) return 'E';
        return 'F';
    }
}

function capitalizeFirstLetter(string) {
    return string.charAt(0).toUpperCase() + string.slice(1);
}

function getGradeLetter(percentage) {
    if (percentage >= 90) return 'A';
    if (percentage >= 80) return 'B';
    if (percentage >= 70) return 'C';
    if (percentage >= 60) return 'D';
    return 'F';
}
function checkForNewSubjects() {
    const calculatorSubjects =
JSON.parse(localStorage.getItem('calculatorSubjects')) || [];
    if (calculatorSubjects.length > 0) {
        userData.subjects = [...userData.subjects,
...calculatorSubjects];
        localStorage.setItem('userData', JSON.stringify(userData));
        localStorage.removeItem('calculatorSubjects');
        loadSubjectGrades();
    }
}
```

Calculator.js –

```
let subjects = [];

        function toggleAssessmentOptions() {
            const subjectType =
document.getElementById('subjectType').value;
            document.getElementById('theoryOptions').style.display =
subjectType === 'theory' ? 'block' : 'none';
            document.getElementById('labOptions').style.display =
subjectType === 'lab' ? 'block' : 'none';

            if (subjectType === 'theory') updateTheoryComponents();
            if (subjectType === 'lab') updateLabComponents();
        }

        function updateTheoryComponents() {
            const assignmentChecked =
document.getElementById('theoryAssignment').checked;
            const digitalChecked =
document.getElementById('theoryDigital').checked;
```

```
            const quizChecked =
document.getElementById('theoryQuiz').checked;

            document.getElementById('theoryAssignmentCount').disabled =
!assignmentChecked;
            document.getElementById('theoryDigitalCount').disabled =
!digitalChecked;
            document.getElementById('theoryQuizCount').disabled =
!quizChecked;

            const assignmentCount = assignmentChecked ?
parseInt(document.getElementById('theoryAssignmentCount').value) : 0;
            const digitalCount = digitalChecked ?
parseInt(document.getElementById('theoryDigitalCount').value) : 0;
            const quizCount = quizChecked ?
parseInt(document.getElementById('theoryQuizCount').value) : 0;

            const totalComponents = assignmentCount + digitalCount +
quizCount;
            const remainingWeight = 30;
            const weightPerComponent = totalComponents > 0 ?
remainingWeight / totalComponents : 0;

            let weightInfo = `Total components selected:
${totalComponents}`;
            if (totalComponents > 0) {
                weightInfo += ` (Each worth
${weightPerComponent.toFixed(2)}%)`;
            }

            document.getElementById('theoryWeightInfo').innerHTML =
weightInfo;
        }

        function updateLabComponents() {
            const assignmentChecked =
document.getElementById('labAssignment').checked;
            const digitalChecked =
document.getElementById('labDigital').checked;
            const quizChecked =
document.getElementById('labQuiz').checked;

            document.getElementById('labAssignmentCount').disabled =
!assignmentChecked;
            document.getElementById('labDigitalCount').disabled =
!digitalChecked;
            document.getElementById('labQuizCount').disabled =
!quizChecked;
```

```
            const assignmentCount = assignmentChecked ?
parseInt(document.getElementById('labAssignmentCount').value) : 0;
            const digitalCount = digitalChecked ?
parseInt(document.getElementById('labDigitalCount').value) : 0;
            const quizCount = quizChecked ?
parseInt(document.getElementById('labQuizCount').value) : 0;

            const totalComponents = assignmentCount + digitalCount +
quizCount;
            const remainingWeight = 60;
            const weightPerComponent = totalComponents > 0 ?
remainingWeight / totalComponents : 0;

            let weightInfo = `Total components selected:
${totalComponents}`;
            if (totalComponents > 0) {
                weightInfo += ` (Each worth
${weightPerComponent.toFixed(2)}%)`;
            }

            document.getElementById('labWeightInfo').innerHTML =
weightInfo;
        }

        function addSubject() {
            const name = document.getElementById('subjectName').value;
            const type = document.getElementById('subjectType').value;
            const credit =
parseInt(document.getElementById('subjectCredit').value);

            if (!name) {
                alert('Please enter a subject name');
                return;
            }

            const subject = {
                id: Date.now(),
                name,
                type,
                credit,
                assessments: [],
                classAverage: type === 'theory' ?
parseInt(document.getElementById('classAverage').value) : null,
                stdDev: type === 'theory' ?
parseInt(document.getElementById('stdDev').value) : null
            };
            if (type === 'theory') {
                subject.assessments.push({ type: 'CAT1', maxMarks: 50,
weightage: 15, marks: null });
                subject.assessments.push({ type: 'CAT2', maxMarks: 50,
weightage: 15, marks: null });
                subject.assessments.push({ type: 'FAT', maxMarks: 100,
weightage: 40, marks: null });

                const assignmentCount =
document.getElementById('theoryAssignment').checked ?

parseInt(document.getElementById('theoryAssignmentCount').value) : 0;
                const digitalCount =
document.getElementById('theoryDigital').checked ?

parseInt(document.getElementById('theoryDigitalCount').value) : 0;
                const quizCount =
document.getElementById('theoryQuiz').checked ?

parseInt(document.getElementById('theoryQuizCount').value) : 0;

                const totalComponents = assignmentCount + digitalCount +
quizCount;
                const weightPerComponent = totalComponents > 0 ? 30 /
totalComponents : 0;

                for (let i = 1; i <= assignmentCount; i++) {
                    subject.assessments.push({
                        type: `Assignment ${i}`,
                        maxMarks: 10,
                        weightage: weightPerComponent,
                        marks: null
                    });
                }

                for (let i = 1; i <= digitalCount; i++) {
                    subject.assessments.push({
                        type: `Digital Assignment ${i}`,
                        maxMarks: 10,
                        weightage: weightPerComponent,
                        marks: null
                    });
                }

                for (let i = 1; i <= quizCount; i++) {
                    subject.assessments.push({
                        type: `Quiz ${i}`,
                        maxMarks: 10,
                        weightage: weightPerComponent,
                        marks: null
                    });
                }
            } else { // lab
                subject.assessments.push({ type: 'FAT', maxMarks: 100,
weightage: 40, marks: null });
                const assignmentCount =
document.getElementById('labAssignment').checked ?

parseInt(document.getElementById('labAssignmentCount').value) : 0;
                const digitalCount =
document.getElementById('labDigital').checked ?

parseInt(document.getElementById('labDigitalCount').value) : 0;
                const quizCount =
document.getElementById('labQuiz').checked ?

parseInt(document.getElementById('labQuizCount').value) : 0;

                const totalComponents = assignmentCount + digitalCount +
quizCount;
                const weightPerComponent = totalComponents > 0 ? 60 /
totalComponents : 0;

                for (let i = 1; i <= assignmentCount; i++) {
                    subject.assessments.push({
                        type: `Assignment ${i}`,
                        maxMarks: 100,
                        weightage: weightPerComponent,
                        marks: null
                    });
                }

                for (let i = 1; i <= digitalCount; i++) {
                    subject.assessments.push({
                        type: `Digital Assignment ${i}`,
                        maxMarks: 100,
                        weightage: weightPerComponent,
                        marks: null
                    });
                }

                for (let i = 1; i <= quizCount; i++) {
                    subject.assessments.push({
                        type: `Quiz ${i}`,
                        maxMarks: 100,
                        weightage: weightPerComponent,
                        marks: null
                    });
                }
            }

            subjects.push(subject);
            renderSubjects();
            document.getElementById('subjectName').value = '';
            const userData =
JSON.parse(localStorage.getItem('userData')) || {
                currentUser:
JSON.parse(localStorage.getItem('userData')).currentUser,
                subjects: [],
                studentMarks: [],
                friends: [],
                rankings: { class: [], batch: [], program: [] }
            };

            userData.subjects.push(subject);
            localStorage.setItem('userData', JSON.stringify(userData));
            //window.location.href = 'dashboard.html';

        }

        function renderSubjects() {
            const container =
document.getElementById('subjectsContainer');
            container.innerHTML = '';

            if (subjects.length === 0) {
                container.innerHTML = '<p>No subjects added yet.</p>';
                return;
            }

            subjects.forEach(subject => {
                const subjectDiv = document.createElement('div');
                subjectDiv.className = 'subject-item';
                subjectDiv.id = `subject-${subject.id}`;

                let html = `
                    <h3>${subject.name}
(${subject.type.charAt(0).toUpperCase() + subject.type.slice(1)},
${subject.credit} credits)
                        <button class="remove-btn"
onclick="removeSubject(${subject.id})">Remove</button>
                    </h3>
                    <div class="assessment-section">
                        <table>
```

```
                            <thead>
                                <tr>
                                    <th>Assessment</th>
                                    <th>Max Marks</th>
                                    <th>Weightage</th>
                                    <th>Your Marks</th>
                                    <th>Weighted Score</th>
                                </tr>
                            </thead>
                            <tbody>
                `;

                let totalWeighted = 0;
                let fatScore = 0;

                subject.assessments.forEach(assessment => {
                    const weighted = assessment.marks !== null ?
                        (assessment.marks / assessment.maxMarks) *
assessment.weightage : 0;
                    totalWeighted += weighted;

                    if (assessment.type === 'FAT') {
                        fatScore = weighted;
                    }

                    html += `
                        <tr>
                            <td>${assessment.type}</td>
                            <td>${assessment.maxMarks}</td>
                            <td>${assessment.weightage}%</td>
                            <td>
                                <input type="number" min="0"
max="${assessment.maxMarks}"
                                    value="${assessment.marks !== null ?
assessment.marks : ''}"
                                    onchange="updateMarks(${subject.id},
'${assessment.type.replace(/ /g, '_')}', this.value)">
                            </td>
                            <td>${weighted.toFixed(2)}</td>
                        </tr>
                    `;
                });

                // Check pass criteria
                const passFat = subject.assessments.find(a => a.type ===
'FAT')?.marks !== null ?
```

```
                    (subject.assessments.find(a => a.type ===
'FAT').marks / subject.assessments.find(a => a.type === 'FAT').maxMarks)
>= 0.4 : false;

                const passed = passFat && totalWeighted >= 50;

                html += `
                            </tbody>
                            <tfoot>
                                <tr>
                                    <th colspan="4">Total</th>

<th>${totalWeighted.toFixed(2)}%</th>
                                </tr>
                            </tfoot>
                        </table>
                    </div>

                    <div class="flex-container">
                        <div class="flex-item">
                            <p>FAT Score: ${fatScore.toFixed(2)}%
${passFat ? '' : '<span class="warning">(Below 40%)</span>'}</p>
                            <p>Status: ${passed ? '<span
style="color:green">Passing</span>' : '<span
class="warning">Failing</span>'}</p>
                        </div>
                `;

                if (subject.type === 'theory') {
                    html += `
                        <div class="flex-item">
                            <label>Class Average:</label>
                            <input type="number" min="0" max="100"
value="${subject.classAverage}"

onchange="updateClassAverage(${subject.id}, this.value)">
                        </div>
                        <div class="flex-item">
                            <label>Std Deviation:</label>
                            <input type="number" min="1" max="20"
value="${subject.stdDev}"
                                onchange="updateStdDev(${subject.id},
this.value)">
                        </div>
                    `;
                }

                html += `</div>`;
                // Calculate and display grade
                const grade = calculateGrade(subject, totalWeighted);
                const gradePoint = getGradePoint(grade);

                html += `
                    <div class="result">
                        Current Grade: <span class="grade">${grade}
(${gradePoint})</span>
                    </div>

                    <!-- Add Save Marks button -->
                    <div class="subject-actions">
                        <button onclick="saveSubject(${subject.id})"
class="save-btn">Save Marks</button>
                    </div>
                `;

                subjectDiv.innerHTML = html;
                container.appendChild(subjectDiv);
            });
        }

        function updateMarks(subjectId, assessmentType, marks) {
            const subject = subjects.find(s => s.id === subjectId);
            if (!subject) return;

            const formattedType = assessmentType.replace(/_/g, ' ');

            const assessment = subject.assessments.find(a => a.type ===
formattedType);

            if (assessment) {
                assessment.marks = marks !== '' ? parseFloat(marks) :
null;
            }

            renderSubjects();
        }

        function updateClassAverage(subjectId, average) {
            const subject = subjects.find(s => s.id === subjectId);
            if (subject) {
                subject.classAverage = parseFloat(average);
                renderSubjects();
            }
        }
        function updateStdDev(subjectId, stdDev) {
            const subject = subjects.find(s => s.id === subjectId);
            if (subject) {
                subject.stdDev = parseFloat(stdDev);
                renderSubjects();
            }
        }

        function removeSubject(subjectId) {
            subjects = subjects.filter(s => s.id !== subjectId);
            renderSubjects();
        }

        function calculateGrade(subject, totalWeighted) {
            if (subject.type === 'lab') {
                // Absolute grading for lab
                if (totalWeighted >= 90) return 'S';
                if (totalWeighted >= 80) return 'A';
                if (totalWeighted >= 70) return 'B';
                if (totalWeighted >= 60) return 'C';
                if (totalWeighted >= 55) return 'D';
                if (totalWeighted >= 50) return 'E';
                return 'F';
            } else {
                // Relative grading for theory based on class average
and standard deviation
                if (!subject.classAverage || !subject.stdDev) return
'N/A';

                const deviation = (totalWeighted - subject.classAverage)
/ subject.stdDev;

                if (deviation >= 1.5) return 'S';
                if (deviation >= 0.5) return 'A';
                if (deviation >= -0.5) return 'B';
                if (deviation >= -1.0) return 'C';
                if (deviation >= -1.5) return 'D';
                if (deviation >= -2.0) return 'E';
                return 'F';
            }
        }

        function getGradePoint(grade) {
            switch (grade) {
                case 'S': return '10';
                case 'A': return '9';
                case 'B': return '8';
                case 'C': return '7';
                case 'D': return '6';
                case 'E': return '5';
                case 'F': return '0';
                default: return 'N/A';
            }
        }

function saveSubject(subjectId) {
    const subject = subjects.find(s => s.id === subjectId);
    if (!subject) return;

    const userData = JSON.parse(localStorage.getItem('userData')) || {
        currentUser:
JSON.parse(localStorage.getItem('userData')).currentUser,
        subjects: [],
        studentMarks: [],
        friends: [],
        rankings: { class: [], batch: [], program: [] }
    };

    const existingSubjectIndex = userData.subjects.findIndex(s => s.id
=== subjectId);
    if (existingSubjectIndex >= 0) {
        // Update existing subject
        userData.subjects[existingSubjectIndex] = subject;
    } else {
        userData.subjects.push(subject);
    }

    localStorage.setItem('userData', JSON.stringify(userData));

    alert('Marks saved successfully!');
}

// Chatbot functionality
document.addEventListener('DOMContentLoaded', function() {
    const chatbotContainer = document.getElementById('chatbot-
container');
    const chatbotToggle = document.getElementById('chatbot-toggle');
    const chatbotLaunch = document.getElementById('chatbot-launch');
    const chatbotMessages = document.getElementById('chatbot-messages');
    const chatbotInput = document.getElementById('chatbot-input-field');
    const chatbotSend = document.getElementById('chatbot-send');

    let isChatbotOpen = false;

    // Toggle chatbot visibility
    chatbotToggle.addEventListener('click', toggleChatbot);
    chatbotLaunch.addEventListener('click', toggleChatbot);

    // Send message when button clicked or Enter pressed
    chatbotSend.addEventListener('click', sendMessage);
    chatbotInput.addEventListener('keypress', function(e) {
        if (e.key === 'Enter') sendMessage();
    });

    function toggleChatbot() {
        isChatbotOpen = !isChatbotOpen;
        chatbotContainer.classList.toggle('open', isChatbotOpen);
        chatbotLaunch.style.display = isChatbotOpen ? 'none' : 'block';

        if (isChatbotOpen && chatbotMessages.children.length === 0) {
            addBotMessage("Hello! I can help predict your grade chances.
Tell me your marks and class average, or ask about grading criteria.");
        }
    }

    function sendMessage() {
        const message = chatbotInput.value.trim();
        if (!message) return;

        addUserMessage(message);
        chatbotInput.value = '';

        // Show typing indicator
        const typing = document.createElement('div');
        typing.className = 'typing-indicator bot-message';
        typing.innerHTML = '<span></span><span></span><span></span>';
        chatbotMessages.appendChild(typing);
        chatbotMessages.scrollTop = chatbotMessages.scrollHeight;

        // Process message and get response
        setTimeout(() => {
            chatbotMessages.removeChild(typing);
            const response = generateResponse(message);
            addBotMessage(response);
        }, 1000);
    }

    function addUserMessage(text) {
        const message = document.createElement('div');
        message.className = 'message user-message';
        message.textContent = text;
        chatbotMessages.appendChild(message);
        chatbotMessages.scrollTop = chatbotMessages.scrollHeight;
    }
    function addBotMessage(text) {
        const message = document.createElement('div');
        message.className = 'message bot-message';
        message.textContent = text;
        chatbotMessages.appendChild(message);
        chatbotMessages.scrollTop = chatbotMessages.scrollHeight;
    }

    function generateResponse(userMessage) {
        // Check for grade prediction request
        if (userMessage.toLowerCase().includes('grade') ||
            userMessage.toLowerCase().includes('predict') ||
            userMessage.toLowerCase().includes('chance')) {

            // Try to extract marks and average from message
            const marksMatch =
userMessage.match(/(\d+)\s*(marks|score|percent|%)/i);
            const avgMatch = userMessage.match(/(\d+)\s*(average|class
average|mean)/i);

            const marks = marksMatch ? parseInt(marksMatch[1]) : null;
            const average = avgMatch ? parseInt(avgMatch[1]) : null;

            if (marks && average) {
                // Simple prediction logic (can be enhanced)
                const deviation = ((marks - average) / average) * 100;

                if (deviation > 20) return "Based on your marks being
significantly above average, you have a very high chance of getting an S
grade!";
                if (deviation > 10) return "Your marks are well above
average - good chance for an A grade, possibly S if the distribution
favors it.";
                if (deviation > 0) return "You're above average - likely
looking at a B grade, with a chance for A if you're near the
threshold.";
                if (deviation > -10) return "You're around average -
probably a C grade, but could move to B with small improvements.";
                if (deviation > -20) return "Below average - likely a D
grade. Focus on key areas to improve.";
                return "Your marks are significantly below average - you
might get an E or F. Please consult with your instructor.";
            } else if (marks) {
                return `With ${marks}%, you would typically get a
${predictGradeFromMarks(marks)} grade. For a more accurate prediction,
please provide the class average.`;
            } else {
                return "To predict your grade, please tell me your marks
and the class average (e.g., 'I have 85 marks with class average of
65').";
            }
        }

        // Default responses
        const defaultResponses = [
            "I can help predict your grade based on your marks and class
average.",
            "The grading system considers both absolute marks and
relative performance.",
            "For theory subjects, grades are based on standard deviation
from the mean.",
            "For lab subjects, grades follow absolute percentage
thresholds.",
            "You can ask me things like: 'What grade will I get with 75
marks and 60 average?'"
        ];

        return defaultResponses[Math.floor(Math.random() *
defaultResponses.length)];
    }

    function predictGradeFromMarks(marks) {
        if (marks >= 90) return 'S';
        if (marks >= 80) return 'A';
        if (marks >= 70) return 'B';
        if (marks >= 60) return 'C';
        if (marks >= 55) return 'D';
        if (marks >= 50) return 'E';
        return 'F';
    }
});

document.addEventListener('DOMContentLoaded', function() {
    toggleAssessmentOptions();
    const savedData = localStorage.getItem('userData');
    if (savedData) {
        const userData = JSON.parse(savedData);
        subjects = userData.subjects || [];
        renderSubjects();
    }
    const theoryCheckboxes =
document.querySelectorAll('input[name="theoryComponents"]');
    theoryCheckboxes.forEach(checkbox => {
        checkbox.addEventListener('change', updateTheoryComponents);
    });
    const labCheckboxes =
document.querySelectorAll('input[name="labComponents"]');
    labCheckboxes.forEach(checkbox => {
        checkbox.addEventListener('change', updateLabComponents);
    });
});
```

Calculator.css –

```
body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            margin: 0;
            padding: 20px;
            background-color: #f5f5f5;
        }

        .container {
            max-width: 1000px;
            margin: 0 auto;
            background: white;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }

        h1,
        h2 {
            color: #333;
            /* margin-left: 50%;
            margin-right: 50%;
            white-space: nowrap;  */
        }

        .subject-form {
            margin-bottom: 20px;
            padding: 15px;
            border: 1px solid #ddd;
            border-radius: 5px;
            background-color: #f9f9f9;
        }

        .subject-list {
            margin-top: 30px;
        }
```

```
    .subject-item {
        margin-bottom: 20px;
        padding: 15px;
        border: 1px solid #ddd;
        border-radius: 5px;
        background-color: #fff;
    }

    .assessment-section {
        margin-top: 15px;
        padding: 10px;
        border: 1px dashed #ccc;
        border-radius: 5px;
    }

    .theory-options,
    .lab-options {
        display: none;
    }

    input,
    select {
        padding: 8px;
        margin: 5px 0;
        border: 1px solid #ddd;
        border-radius: 4px;
    }

    button {
        background-color: #4CAF50;
        color: white;
        padding: 10px 15px;
        border: none;
        border-radius: 4px;
        cursor: pointer;
        margin-right: 10px;
    }

    button:hover {
        background-color: #45a049;
    }

    .result {
        font-weight: bold;
        margin-top: 10px;
        padding: 10px;
        background-color: #e9f7ef;
        border-radius: 4px;
    }

    .grade {
        font-size: 1.2em;
        color: #2e7d32;
    }

    .remove-btn {
        background-color: #f44336;
        float: right;
    }

    .remove-btn:hover {
        background-color: #d32f2f;
    }

    table {
        width: 100%;
        border-collapse: collapse;
        margin-top: 10px;
    }

    th,
    td {
        border: 1px solid #ddd;
        padding: 8px;
        text-align: left;
    }

    th {
        background-color: #f2f2f2;
    }

    .warning {
        color: #d32f2f;
        font-weight: bold;
    }

    .component-controls {
        margin-left: 20px;
        margin-bottom: 10px;
    }

    .flex-container {
        display: flex;
        flex-wrap: wrap;
        gap: 15px;
```

```css
        }

        .flex-item {
            flex: 1;
            min-width: 200px;
        }
        .subject-actions {
    margin-top: 15px;
    text-align: right;
}

.save-btn {
    background-color: #2196F3;
    color: white;
    border: none;
    padding: 8px 16px;
    border-radius: 4px;
    cursor: pointer;
    font-weight: bold;
}

.save-btn:hover {
    background-color: #0b7dda;
}

/* Chatbot Styles */
.chatbot-container {
    position: fixed;
    bottom: 40px;
    right: 20px;
    width: 350px;
    height: 500px;
    background: white;
    border-radius: 10px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
    display: flex;
    flex-direction: column;
    overflow: hidden;
    z-index: 1000;
    transform: translateY(100%);
    transition: transform 0.3s ease;
}

.chatbot-container.open {
    transform: translateY(0);
}

.chatbot-header {
    background: #0066ff;
    color: white;
    padding: 15px;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.chatbot-header h3 {
    margin: 0;
    font-size: 1.2rem;
}

.chatbot-toggle {
    background: none;
    border: none;
    color: white;
    font-size: 1.5rem;
    cursor: pointer;
    padding: 0 5px;
}

.chatbot-messages {
    flex: 1;
    padding: 15px;
    overflow-y: auto;
    display: flex;
    flex-direction: column;
    gap: 10px;
}

.chatbot-input {
    display: flex;
    padding: 10px;
    border-top: 1px solid #eee;
}

.chatbot-input-field {
    flex: 1;
    flex-grow: 1;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
    margin-right: 10px;
}

.chatbot-input button {
    background: #0066ff;
    color: white;
    border: none;
    border-radius: 4px;
    padding: 0 15px;
    cursor: pointer;
    margin-left: auto;
}

.chatbot-launch {
    position: fixed;
    bottom: 20px;
    right: 20px;
    background: #0066ff;
    color: white;
    border: none;
    border-radius: 50%;
    width: 80px;
    height: 80px;
    font-size: 1rem;
    cursor: pointer;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.2);
    z-index: 999;
}

.message {
    max-width: 80%;
    padding: 10px 15px;
    border-radius: 18px;
    line-height: 1.4;
}

.user-message {
    align-self: flex-end;
    background: #0066ff;
    color: white;
    border-bottom-right-radius: 5px;
}

.bot-message {
    align-self: flex-start;
    background: #f1f1f1;
    color: #333;
    border-bottom-left-radius: 5px;
}

.typing-indicator {
    display: inline-block;
    padding: 10px 15px;
```

```css
    background: #f1f1f1;
    border-radius: 18px;
    border-bottom-left-radius: 5px;
}

.typing-indicator span {
    display: inline-block;
    width: 8px;
    height: 8px;
    background: #666;
    border-radius: 50%;
    margin: 0 2px;
    animation: typing 1s infinite;
}

.typing-indicator span:nth-child(2) {
    animation-delay: 0.2s;
}

.typing-indicator span:nth-child(3) {
    animation-delay: 0.4s;
}

@keyframes typing {
    0%, 100% { opacity: 0.4; }
    50% { opacity: 1; }
}

.heading-page {
    margin-left: 0%;
}
```