College of Innovative Management & Science
Approved by University of Lucknow, Lucknow
Located Near Ambedkar Park, Vipul Khand - VI, Gomti Nagar.

A SUMMER TRAINING PROJECT REPORT

ON

"Python Full Stack Development "

Submitted in Partial Fulfilment of the Requirements for the

BCA ,2nd year –Summer Training Report on Data Structures and Algorithms (DSA) using java

TRAINING ORGANIZATION

IT Consulting Private Limited
Research | Development | Training
INDIA'S NO.1 TRAINING COMPANY

22-Aug 2025

Submitted To:

Prof.Chitranjan Singh

Submitted by:

Sanchita Singh

BCA(3nd year)

Acknowledgement

I would like to express my sincere gratitude to all those who supported and guided me throughout the successful completion of my summer training in **Python Full Stack Development**. First and foremost, I am thankful to **OJD IT Consulting (P) Limited** for providing me with the opportunity to undergo a **45 Days Summer Training Program** and for offering a well-structured learning environment that helped me enhance my technical and practical skills.

I am grateful to the training team and staff members of **OJD IT Consulting (P) Limited** for their continuous support, guidance, and cooperation throughout the training period, which played an important role in the successful completion of this project.

Lastly, I would like to thank my family and friends for their constant encouragement, motivation, and moral support, which helped me complete this training successfully.

Abstract

This report presents a comprehensive overview of the **Python Full Stack Development** training undertaken as part of the **45 Days Summer Training Program**. The primary objective of the training was to gain practical knowledge of both front-end and back-end web development technologies and to understand how they work together to build complete web applications.

The training covered front-end technologies such as **HTML, CSS, and Bootstrap** for designing responsive and user-friendly interfaces, along with back-end development using **Python** and the **Django framework**. Database concepts were also introduced to manage and store application data efficiently. Emphasis was placed on hands-on learning through practical exercises, coding tasks, and project-based implementation.

The methodology involved understanding theoretical concepts followed by real-time application through coding assignments and project development. This approach helped in developing a clear understanding of server-side logic, database connectivity, and integration between the front-end and back-end components of a web application.

The outcomes of the training demonstrate the ability to design, develop, and implement a complete web-based application using Python Full Stack Development principles. In conclusion, this training enhanced technical skills, problem-solving abilities, and provided a strong foundation for advanced web development and software engineering concepts.

Certificate

Govt. of India Regd.

Reg. ID : 5280

Certificate No : 1801259

OJD IT COUNSULTING [P] LIMITED

An Autonomous Institution Regd. Under Sec 23 Rac Act 1956. Regd. MSME, NITI Aayog Government Of India.

Certificate of Completion

This is to certify that

Sanchita Singh

has successfully completed the **45 Days**

Summer Training in Python Full Stack Development

(HTML CSS Bootstrap, Python, Database, Django Framework, Project)

The candidate has successfully completed the program training from OJD on **22-Aug-25** and is awarded **A** Grade.

Authorized Signatory

Issued by: OJD IT CONSULTING PVT. LTD. Having its Registered Office at 2nd Floor, Sahara Plaza, Patrakarpuram, Gomti Nagar, Lucknow - 10 (UP)

Head Office: A-15, Ashoka Appartment, NR Tinu Public School Devli, Sainik Farms, Delhi (India) - 110062, www.OJDITC.com

An Autonomous Institution (Regd. Under Society Act, 1860) For Industrial And Vocational Education Incorporated Under Ministry Of Corporate Affairs Government Of India.

***This is a computer generated digitally signed certificate issued as per the particulars filed by the candidate

Table of Contents

1. Acknowledgement

2.Abstract

3.Certificate

4.Table of Contents

6.Methodology

8. Conclusion

7. Results & Discussions

9. References

Introduction

Python Full Stack Development forms an essential part of modern web application development, enabling the creation of dynamic, scalable, and user-friendly software solutions. This summer training program focused on understanding and implementing full stack development concepts using Python, along with front-end and back-end technologies commonly used in the software industry.

The primary objective of this training was to enhance practical development skills by working with front-end technologies such as **HTML, CSS, and Bootstrap**, and back-end development using **Python and the Django framework**. The training also covered database concepts to manage and retrieve data efficiently. Through hands-on exercises, assignments, and project-based learning, the program provided real-world experience in building complete web applications.

This report describes the methodology adopted during the training, key learning outcomes, and results achieved. The structured and practical approach ensured a clear understanding of full stack development, laying a strong foundation for advanced web technologies and future opportunities in the IT industry.

.

Methodology

The training program followed a structured and systematic approach that combined theoretical learning with practical implementation. The primary objective was to gain hands-on experience in Python Full Stack Development by understanding both front-end and back-end technologies and their integration in real-world web applications.

The training began with an introduction to front-end development, covering basic concepts of HTML, CSS, and Bootstrap to design responsive and user-friendly web interfaces. After understanding the front-end fundamentals, the focus shifted to back-end development using Python and the Django framework, where concepts such as URL routing, views, models, and templates were explored. Database concepts were also included as part of the methodology to understand data storage, retrieval, and management in web applications. Practical exercises were carried out to connect the database with the back-end and perform basic operations such as data insertion, updating, and deletion.

A major emphasis of the training was on hands-on coding and project-based learning. Regular assignments and practice tasks were provided to strengthen conceptual understanding and improve coding skills. The methodology also included testing and debugging techniques to ensure the proper functioning of web applications.

Overall, the training methodology ensured a balanced blend of theory and practice, enabling the development of a complete web application. This structured approach helped in gaining a clear understanding of Python Full Stack Development and its application in building scalable and efficient web-based solutions.

Results & Discussions

The summer training program in Python Full Stack Development significantly enhanced my understanding of web application development and strengthened my practical programming skills. The training provided a balanced combination of front-end and back-end development concepts, enabling the successful implementation of real-world web applications. The key outcomes and results of the training are summarized below:

1. Key Achievements

Gained a strong understanding of front-end technologies such as HTML, CSS, and Bootstrap for designing responsive and user-friendly interfaces.

Developed back-end functionality using Python and the Django framework, including URL routing, views, models, and templates.

Successfully implemented database connectivity to perform Create, Read, Update, and Delete (CRUD) operations efficiently.

Built a functional web-based application by integrating front-end, back-end, and database components.

2. Practical Implementation

Worked on hands-on coding exercises and mini-projects to understand the complete development lifecycle of a web application.

Implemented user authentication, form handling, and basic validation techniques.

Applied testing and debugging methods to ensure smooth functionality and error-free execution of the application.

3. Challenges Faced

Initially faced challenges in understanding the integration between front-end and back-end components. However, continuous practice and project-based learning helped in overcoming these difficulties.

Managing database connectivity and handling server-side logic required careful implementation, which was improved through practical exercises and debugging.

4. Learning Outcomes and Insights

The training highlighted the importance of structured coding practices and modular design in web development.

Gained insight into how full stack development enables efficient and scalable web applications.

Developed problem-solving skills and a better understanding of real-world application development using Python-based technologies.

Conclusion

In conclusion, the summer training program in **Python Full Stack Development** has been a valuable and enriching learning experience. The training provided a strong understanding of both front-end and back-end web development concepts and offered practical exposure to building complete web-based applications.

Throughout the training, I gained hands-on experience in using front-end technologies such as **HTML, CSS, and Bootstrap**, along with back-end development using **Python and the Django framework**. Working with databases and integrating them with server-side logic enhanced my understanding of how real-world web applications are developed and maintained.

During the course of the training, several challenges were encountered, particularly in integrating different components of a web application and ensuring smooth data handling. Overcoming these challenges improved my problem-solving abilities, debugging skills, and understanding of structured coding practices.

The training not only strengthened my technical skills but also helped me develop a systematic and professional approach to web development. It has prepared me to explore advanced web technologies and apply Python Full Stack Development concepts effectively in academic projects and future professional opportunities.

Overall, this summer training has laid a strong foundation for my career in software development and has motivated me to continue learning and growing in the field of full stack web development.

Sample Source Code

```
<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Library Management System</title>
    <style>
        :root {
            --color-primary: #208090;
            --color-primary-hover: #1a6b78;
            --color-primary-active: #1a5b68;
            --color-bg: #fcfcf9;
            --color-surface: #ffffff;
            --color-text: #134252;
            --color-text-secondary: #626c71;
            --color-border: #dfe4e5;
            --color-success: #22c55e;
            --color-error: #c01530;
            --color-warning: #a84b2f;
        }

        * {
            box-sizing: border-box;
        }

        body {
            margin: 0;
            padding: 0;
            font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', sans-
serif;
            background-color: var(--color-bg);
            color: var(--color-text);
            line-height: 1.5;
        }

        .container {
            max-width: 1200px;
            margin: 0 auto;
            padding: 20px;
        }

        header {
            background: linear-gradient(135deg, var(--color-primary) 0%, var(--
color-primary-hover) 100%);
            color: white;
```

```
        padding: 30px 0;
        margin-bottom: 30px;
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    }

    header h1 {
        margin: 0;
        font-size: 28px;
        font-weight: 600;
    }

    header p {
        margin: 8px 0 0 0;
        opacity: 0.9;
        font-size: 14px;
    }

    .auth-container {
        background: var(--color-surface);
        border: 1px solid var(--color-border);
        border-radius: 12px;
        padding: 40px;
        max-width: 400px;
        margin: 40px auto;
        box-shadow: 0 2px 8px rgba(0, 0, 0, 0.05);
    }

    .auth-container h2 {
        margin: 0 0 30px 0;
        color: var(--color-text);
        font-size: 24px;
        text-align: center;
    }

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

    label {
        display: block;
        margin-bottom: 8px;
        font-weight: 500;
        font-size: 14px;
        color: var(--color-text);
    }

    input, select, textarea {
        width: 100%;
        padding: 12px;
```

```
            border: 1px solid var(--color-border);
            border-radius: 8px;
            font-size: 14px;
            font-family: inherit;
            transition: border-color 0.2s, box-shadow 0.2s;
        }

        input:focus, select:focus, textarea:focus {
            outline: none;
            border-color: var(--color-primary);
            box-shadow: 0 0 0 3px rgba(32, 128, 144, 0.1);
        }

        .form-row {
            display: grid;
            grid-template-columns: 1fr 1fr;
            gap: 15px;
        }

        .form-row.full {
            grid-template-columns: 1fr;
        }

        button {
            padding: 12px 24px;
            border: none;
            border-radius: 8px;
            font-size: 14px;
            font-weight: 600;
            cursor: pointer;
            transition: all 0.2s;
            font-family: inherit;
        }

        .btn-primary {
            background-color: var(--color-primary);
            color: white;
            width: 100%;
        }

        .btn-primary:hover {
            background-color: var(--color-primary-hover);
            box-shadow: 0 4px 12px rgba(32, 128, 144, 0.2);
        }

        .btn-primary:active {
            background-color: var(--color-primary-active);
        }
```

```css
.btn-secondary {
    background-color: transparent;
    color: var(--color-primary);
    border: 1px solid var(--color-primary);
    padding: 10px 20px;
    font-size: 13px;
}

.btn-secondary:hover {
    background-color: rgba(32, 128, 144, 0.05);
}

.btn-danger {
    background-color: var(--color-error);
    color: white;
    padding: 8px 16px;
    font-size: 13px;
}

.btn-danger:hover {
    background-color: #b01228;
}

.btn-success {
    background-color: var(--color-success);
    color: white;
    padding: 8px 16px;
    font-size: 13px;
}

.btn-success:hover {
    background-color: #22b350;
}

.dashboard {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    gap: 20px;
    margin-bottom: 30px;
}

.stat-card {
    background: var(--color-surface);
    border: 1px solid var(--color-border);
    border-radius: 12px;
    padding: 20px;
    text-align: center;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.05);
}
```

```css
.stat-card h3 {
    margin: 0 0 10px 0;
    font-size: 14px;
    color: var(--color-text-secondary);
    font-weight: 500;
}

.stat-card .number {
    font-size: 32px;
    font-weight: 700;
    color: var(--color-primary);
    margin: 0;
}

.tabs {
    display: flex;
    gap: 10px;
    margin-bottom: 20px;
    border-bottom: 2px solid var(--color-border);
    overflow-x: auto;
    padding-bottom: 10px;
}

.tab-button {
    padding: 12px 20px;
    border: none;
    background: none;
    color: var(--color-text-secondary);
    cursor: pointer;
    font-size: 14px;
    font-weight: 500;
    border-bottom: 2px solid transparent;
    margin-bottom: -12px;
    transition: all 0.2s;
    white-space: nowrap;
}

.tab-button.active {
    color: var(--color-primary);
    border-bottom-color: var(--color-primary);
}

.tab-content {
    display: none;
}

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

```
}

.card {
    background: var(--color-surface);
    border: 1px solid var(--color-border);
    border-radius: 12px;
    padding: 20px;
    margin-bottom: 20px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.05);
}

.card h3 {
    margin: 0 0 15px 0;
    font-size: 18px;
    color: var(--color-text);
}

.book-list {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
    gap: 20px;
}

.book-card {
    background: var(--color-surface);
    border: 1px solid var(--color-border);
    border-radius: 12px;
    padding: 20px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.05);
    transition: all 0.3s;
}

.book-card:hover {
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.1);
    transform: translateY(-2px);
}

.book-card h4 {
    margin: 0 0 8px 0;
    font-size: 16px;
    color: var(--color-text);
}

.book-card p {
    margin: 8px 0;
    font-size: 13px;
    color: var(--color-text-secondary);
}
```

```css
.book-card .author {
    font-weight: 500;
    color: var(--color-primary);
}

.book-meta {
    display: flex;
    justify-content: space-between;
    margin: 15px 0;
    padding: 10px 0;
    border-top: 1px solid var(--color-border);
    border-bottom: 1px solid var(--color-border);
}

.book-meta div {
    font-size: 13px;
}

.book-meta label {
    margin: 0;
    font-weight: 600;
    color: var(--color-text-secondary);
    font-size: 11px;
    text-transform: uppercase;
}

.status-badge {
    display: inline-block;
    padding: 6px 12px;
    border-radius: 6px;
    font-size: 12px;
    font-weight: 600;
}

.status-available {
    background-color: rgba(34, 197, 94, 0.1);
    color: var(--color-success);
}

.status-issued {
    background-color: rgba(192, 21, 47, 0.1);
    color: var(--color-error);
}

.actions {
    display: flex;
    gap: 10px;
    margin-top: 15px;
}
```

```css
.actions button {
    flex: 1;
    padding: 10px;
    font-size: 13px;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin: 15px 0;
}

th, td {
    padding: 12px;
    text-align: left;
    border-bottom: 1px solid var(--color-border);
    font-size: 14px;
}

th {
    background-color: rgba(32, 128, 144, 0.05);
    color: var(--color-text);
    font-weight: 600;
}

tr:hover {
    background-color: rgba(32, 128, 144, 0.02);
}

.alert {
    padding: 12px 16px;
    border-radius: 8px;
    margin-bottom: 20px;
    font-size: 14px;
    animation: slideIn 0.3s ease;
}

@keyframes slideIn {
    from {
        opacity: 0;
        transform: translateY(-10px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}
```

```css
.alert-success {
    background-color: rgba(34, 197, 94, 0.1);
    color: var(--color-success);
    border: 1px solid var(--color-success);
}

.alert-error {
    background-color: rgba(192, 21, 47, 0.1);
    color: var(--color-error);
    border: 1px solid var(--color-error);
}

.alert-info {
    background-color: rgba(32, 128, 144, 0.1);
    color: var(--color-primary);
    border: 1px solid var(--color-primary);
}

.user-info {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
    padding: 15px;
    background: rgba(32, 128, 144, 0.05);
    border-radius: 8px;
    border: 1px solid var(--color-border);
}

.user-info span {
    font-weight: 500;
    color: var(--color-primary);
}

.search-box {
    display: flex;
    gap: 10px;
    margin-bottom: 20px;
}

.search-box input {
    flex: 1;
}

.search-box button {
    padding: 12px 24px;
}

.empty-state {
```

```
            text-align: center;
            padding: 40px;
            color: var(--color-text-secondary);
        }

        .empty-state h3 {
            margin: 0 0 10px 0;
            color: var(--color-text);
        }

        @media (max-width: 768px) {
            .dashboard {
                grid-template-columns: 1fr;
            }

            .form-row {
                grid-template-columns: 1fr;
            }

            .book-list {
                grid-template-columns: 1fr;
            }

            .tabs {
                flex-wrap: wrap;
            }
        }
    </style>
</head>
<body>
    <header>
        <div class="container">
            <h1>📚 Library Management System</h1>
            <p>Manage books, track borrowing, and maintain inventory
efficiently</p>
        </div>
    </header>

    <div class="container">
        <div id="app"></div>
    </div>

    <script>
        // State management
        const app = {
            currentUser: null,
            userRole: null,
            books: [
```

```
                { id: 1, title: 'Introduction to Algorithms', author: 'Cormen, Leiserson, Rivest, Stein', isbn: '978-0262033848', copies: 5, available: 3, category: 'Computer Science', issued_to: null },
                { id: 2, title: 'Clean Code', author: 'Robert C. Martin', isbn: '978-0132350884', copies: 4, available: 2, category: 'Programming', issued_to: null },
                { id: 3, title: 'Design Patterns', author: 'Gang of Four', isbn: '978-0201633610', copies: 3, available: 1, category: 'Software Design', issued_to: null },
                { id: 4, title: 'The Pragmatic Programmer', author: 'Hunt & Thomas', isbn: '978-0201616224', copies: 6, available: 4, category: 'Programming', issued_to: null },
                { id: 5, title: 'Fluent Python', author: 'Luciano Ramalho', isbn: '978-1491946008', copies: 2, available: 1, category: 'Python Programming', issued_to: null }
            ],
            members: [
                { id: 1, name: 'Sanchita Singh', email: 'sanchitasingh@9696.com', password: 'singh9696', role: 'librarian', books_issued: 0 },
                { id: 2, name: 'Nidhi Singh', email: 'singhnidhi@123.com', password: 'nidhi123', role: 'member', books_issued: 2 },
                { id: 3, name: 'Sarthak Singh', email: 'sarthak@0987.com', password: 'singh0987', role: 'member', books_issued: 1 }
            ],
            transactions: [
                { id: 1, member_id: 2, book_id: 1, issue_date: '2024-12-10', return_date: null, status: 'issued' },
                { id: 2, member_id: 2, book_id: 4, issue_date: '2024-12-05', return_date: '2024-12-15', status: 'returned' },
                { id: 3, member_id: 3, book_id: 3, issue_date: '2024-12-12', return_date: null, status: 'issued' }
            ]
        };

        function renderApp() {
            const appDiv = document.getElementById('app');

            if (!app.currentUser) {
                appDiv.innerHTML = renderLoginPage();
            } else if (app.userRole === 'librarian') {
                appDiv.innerHTML = renderLibrarianDashboard();
            } else {
                appDiv.innerHTML = renderMemberDashboard();
            }
        }

        function renderLoginPage() {
            return `
```

```
                <div class="auth-container">
                    <h2>Login</h2>
                    <form id="loginForm">
                        <div class="form-group">
                            <label for="email">Email</label>
                            <input type="email" id="email" required>
                        </div>
                        <div class="form-group">
                            <label for="password">Password</label>
                            <input type="password" id="password" required>
                        </div>
                        <button type="button" class="btn-primary"
onclick="handleLogin()">Login</button>
                    </form>
                    <div style="text-align: center; margin-top: 20px; font-
size: 13px; color: var(--color-text-secondary);">
                        <p>Demo Credentials:</p>
                        <p>Librarian: Sanchita Singh/ singh9696</p>
                        <p>Member: Nidhi Singh/ nidhi123</p>
                    </div>
                </div>
            `;
        }

        function handleLogin() {
            const email = document.getElementById('email').value;
            const password = document.getElementById('password').value;

            const member = app.members.find(m => m.email === email &&
m.password === password);

            if (member) {
                app.currentUser = member;
                app.userRole = member.role;
                renderApp();
            } else {
                alert('Invalid credentials!');
            }
        }

        function handleLogout() {
            app.currentUser = null;
            app.userRole = null;
            renderApp();
        }

        function renderLibrarianDashboard() {
            const totalBooks = app.books.reduce((sum, b) => sum + b.copies, 0);
```

```
            const availableBooks = app.books.reduce((sum, b) => sum +
b.available, 0);
            const issuedBooks = totalBooks - availableBooks;

            return `
                <div class="user-info">
                    <span>👤 Librarian: ${app.currentUser.name}</span>
                    <button class="btn-secondary"
onclick="handleLogout()">Logout</button>
                </div>

                <div class="dashboard">
                    <div class="stat-card">
                        <h3>Total Books</h3>
                        <p class="number">${totalBooks}</p>
                    </div>
                    <div class="stat-card">
                        <h3>Available</h3>
                        <p class="number">${availableBooks}</p>
                    </div>
                    <div class="stat-card">
                        <h3>Issued</h3>
                        <p class="number">${issuedBooks}</p>
                    </div>
                </div>

                <div class="tabs">
                    <button class="tab-button active"
onclick="switchTab('books')">📚 Books</button>
                    <button class="tab-button"
onclick="switchTab('members')">👥 Members</button>
                    <button class="tab-button"
onclick="switchTab('transactions')">📋 Transactions</button>
                    <button class="tab-button"
onclick="switchTab('addbook')">➕ Add Book</button>
                </div>

                <div id="books" class="tab-content active">
                    ${renderBooksList()}
                </div>

                <div id="members" class="tab-content">
                    ${renderMembersList()}
                </div>

                <div id="transactions" class="tab-content">
                    ${renderTransactionsList()}
                </div>
```

```
                <div id="addbook" class="tab-content">
                    ${renderAddBookForm()}
                </div>
            `;
        }

        function renderMemberDashboard() {
            const memberIssued = app.transactions.filter(t => t.member_id ===
app.currentUser.id && t.status === 'issued');

            return `
                <div class="user-info">
                    <span>👤 Member: ${app.currentUser.name}</span>
                    <button class="btn-secondary"
onclick="handleLogout()">Logout</button>
                </div>

                <div class="dashboard">
                    <div class="stat-card">
                        <h3>Books Issued</h3>
                        <p class="number">${memberIssued.length}</p>
                    </div>
                    <div class="stat-card">
                        <h3>Available Books</h3>
                        <p class="number">${app.books.reduce((sum, b) => sum +
b.available, 0)}</p>
                    </div>
                    <div class="stat-card">
                        <h3>Total Books</h3>
                        <p class="number">${app.books.length}</p>
                    </div>
                </div>

                <div class="tabs">
                    <button class="tab-button active"
onclick="switchTab('mybooks')">📚 Available Books</button>
                    <button class="tab-button"
onclick="switchTab('myissued')">📖 My Issued Books</button>
                    <button class="tab-button"
onclick="switchTab('myhistory')">📜 History</button>
                </div>

                <div id="mybooks" class="tab-content active">
                    ${renderBooksForMember()}
                </div>

                <div id="myissued" class="tab-content">
                    ${renderIssuedBooksForMember()}
                </div>
```

```
                <div id="myhistory" class="tab-content">
                    ${renderMemberHistory()}
                </div>
            `;
        }

        function renderBooksList() {
            const bookCards = app.books.map(book => {
                const status = book.available > 0 ? 'available' : 'issued';
                return `
                    <div class="book-card">
                        <h4>${book.title}</h4>
                        <p><strong class="author">${book.author}</strong></p>
                        <p>ISBN: ${book.isbn}</p>
                        <p>Category: ${book.category}</p>
                        <div class="book-meta">
                            <div>
                                <label>Total</label><br>
                                ${book.copies}
                            </div>
                            <div>
                                <label>Available</label><br>
                                ${book.available}
                            </div>
                            <div>
                                <span class="status-badge status-${status}">
                                    ${status === 'available' ? '✓ Available' :
'✗ Limited'}
                                </span>
                            </div>
                        </div>
                        <div class="actions">
                            <button class="btn-secondary"
onclick="editBook(${book.id})">Edit</button>
                            <button class="btn-danger"
onclick="deleteBook(${book.id})">Delete</button>
                        </div>
                    </div>
                `;
            }).join('');

            return `<div class="card"><h3>Books Inventory</h3>${bookCards ?
`<div class="book-list">${bookCards}</div>` : '<div class="empty-state"><h3>No
books</h3></div>'}</div>`;
        }

        function renderBooksForMember() {
            const availableBooks = app.books.filter(b => b.available > 0);
```

```
            if (availableBooks.length === 0) {
                return '<div class="card"><h3>Available Books</h3><div
class="empty-state"><h3>No books available</h3><p>Check back
soon!</p></div></div>';
            }

            const bookCards = availableBooks.map(book => `
                <div class="book-card">
                    <h4>${book.title}</h4>
                    <p><strong class="author">${book.author}</strong></p>
                    <p>Category: ${book.category}</p>
                    <div class="book-meta">
                        <div><label>Available</label><br>${book.available}/${bo
ok.copies}</div>
                        <div><span class="status-badge status-available">✓
Available</span></div>
                    </div>
                    <div class="actions">
                        <button class="btn-primary"
onclick="issueBook(${book.id})">Issue Book</button>
                    </div>
                </div>
            `).join('');

            return `<div class="card"><h3>Available Books</h3><div class="book-
list">${bookCards}</div></div>`;
        }

        function renderIssuedBooksForMember() {
            const issued = app.transactions.filter(t => t.member_id ===
app.currentUser.id && t.status === 'issued');

            if (issued.length === 0) {
                return '<div class="card"><h3>My Issued Books</h3><div
class="empty-state"><h3>No books issued</h3></div></div>';
            }

            const bookCards = issued.map(tx => {
                const book = app.books.find(b => b.id === tx.book_id);
                return `
                    <div class="book-card">
                        <h4>${book.title}</h4>
                        <p><strong class="author">${book.author}</strong></p>
                        <p>Issued Date: ${tx.issue_date}</p>
                        <div class="actions">
                            <button class="btn-success"
onclick="returnBook(${tx.id})">Return Book</button>
                        </div>
```

```
                    </div>
                `;
            }).join('');

            return `<div class="card"><h3>My Issued Books</h3><div class="book-
list">${bookCards}</div></div>`;
        }

        function renderMemberHistory() {
            const history = app.transactions.filter(t => t.member_id ===
app.currentUser.id);

            if (history.length === 0) {
                return '<div class="card"><h3>History</h3><div class="empty-
state"><h3>No transactions</h3></div></div>';
            }

            const rows = history.map(tx => {
                const book = app.books.find(b => b.id === tx.book_id);
                return `
                    <tr>
                        <td>${book.title}</td>
                        <td>${tx.issue_date}</td>
                        <td>${tx.return_date || '-'}</td>
                        <td><span class="status-badge status-
${tx.status}">${tx.status}</span></td>
                    </tr>
                `;
            }).join('');

            return `
                <div class="card">
                    <h3>Transaction History</h3>
                    <table>
                        <thead>
                            <tr>
                                <th>Book</th>
                                <th>Issued Date</th>
                                <th>Return Date</th>
                                <th>Status</th>
                            </tr>
                        </thead>
                        <tbody>${rows}</tbody>
                    </table>
                </div>
            `;
        }

        function renderMembersList() {
```

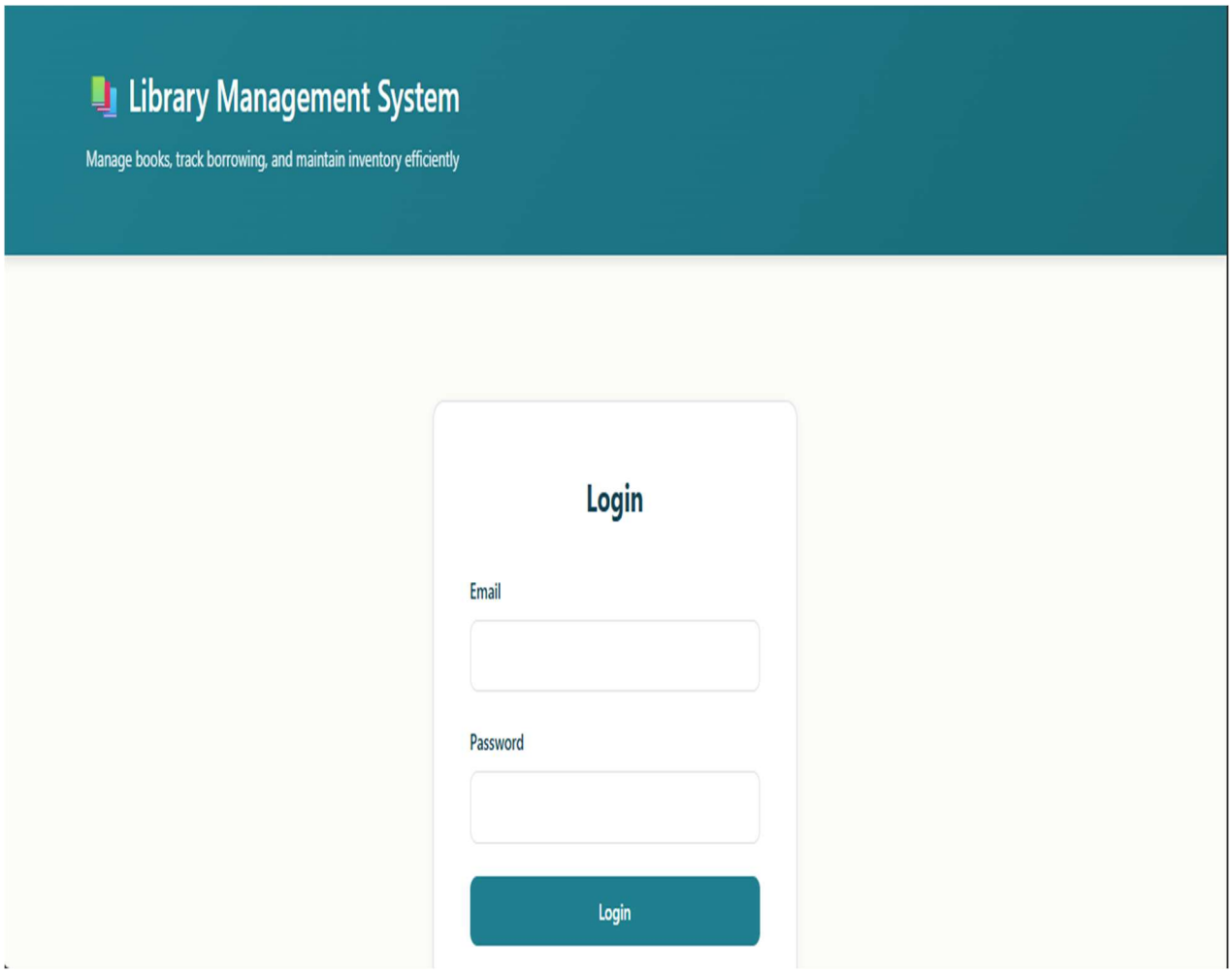
```
            const memberRows = app.members.filter(m => m.role ===
'member').map(member => `
                <tr>
                    <td>${member.name}</td>
                    <td>${member.email}</td>
                    <td>${member.books_issued}</td>
                    <td>
                        <button class="btn-danger" style="width: 80px;"
onclick="removeMember(${member.id})">Remove</button>
                    </td>
                </tr>
            `).join('');

            return `
                <div class="card">
                    <h3>Library Members</h3>
                    <table>
                        <thead>
                            <tr>
                                <th>Name</th>
                                <th>Email</th>
                                <th>Books Issued</th>
                                <th>Action</th>
                            </tr>
                        </thead>
                        <tbody>${memberRows}</tbody>
                    </table>
                </div>
            `;
        }

        function renderTransactionsList() {
            const transactionRows = app.transactions.map(tx => {
                const book = app.books.find(b => b.id === tx.book_id);
                const member = app.members.find(m => m.id === tx.member_id);
                return `
                    <tr>
                        <td>${member.name}</td>
                        <td>${book.title}</td>
                        <td>${tx.issue_date}</td>
                        <td>${tx.return_date || 'Not Returned'}</td>
                        <td><span class="status-badge status-
${tx.status}">${tx.status}</span></td>
                        ${tx.status === 'issued' ? `<td><button class="btn-
success" style="width: 80px;"
onclick="completeReturn(${tx.id})">Return</button></td>` : '<td>-</td>'}
                    </tr>
                `;
            }).join('');
```

```
        return `
            <div class="card">
                <h3>All Transactions</h3>
                <table>
                    <thead>
                        <tr>
                            <th>Member</th>
                            <th>Book</th>
                            <th>Issue Date</th>
                            <th>Return Date</th>
                            <th>Status</th>
                            <th>Action</th>
                        </tr>
                    </thead>
                    <tbody>${transactionRows}</tbody>
                </table>
            </div>
        `;
    }

    function renderAddBookForm() {
        return `
            <div class="card">
                <h3>Add New Book</h3>
                <form id="addBookForm">
                    <div class="form-row">
                        <div class="form-group">
                            <label for="title">Title *</label>
                            <input type="text" id="title" required>
                        </div>
                        <div class="form-group">
                            <label for="author">Author *</label>
                            <input type="text" id="author" required>
                        </div>
                    </div>
                    <div class="form-row">
                        <div class="form-group">
                            <label for="isbn">ISBN</label>
                            <input type="text" id="isbn">
                        </div>
                        <div class="form-group">
                            <label for="category">Category *</label>
                            <input type="text" id="category" required>
                        </div>
                    </div>
                    <div class="form-row">
                        <div class="form-group">
                            <label for="copies">Number of Copies *</label>
```

```
                                <input type="number" id="copies" min="1"
required>
                            </div>
                        </div>
                        <button type="button" class="btn-primary"
onclick="addBook()">Add Book</button>
                    </form>
                </div>
            `;
        }

        function addBook() {
            const title = document.getElementById('title').value;
            const author = document.getElementById('author').value;
            const isbn = document.getElementById('isbn').value;
            const category = document.getElementById('category').value;
            const copies = parseInt(document.getElementById('copies').value);

            if (!title || !author || !category || !copies) {
                alert('Please fill all required fields');
                return;
            }

            const newBook = {
                id: Math.max(...app.books.map(b => b.id), 0) + 1,
                title,
                author,
                isbn,
                copies,
                available: copies,
                category,
                issued_to: null
            };

            app.books.push(newBook);
            alert('Book added successfully!');
            document.getElementById('addBookForm').reset();
            switchTab('books');
            renderApp();
        }

        function issueBook(bookId) {
            const book = app.books.find(b => b.id === bookId);
            if (book.available <= 0) {
                alert('No copies available!');
                return;
            }

            book.available--;
```

```
            const tx = {
                id: Math.max(...app.transactions.map(t => t.id), 0) + 1,
                member_id: app.currentUser.id,
                book_id: bookId,
                issue_date: new Date().toISOString().split('T')[0],
                return_date: null,
                status: 'issued'
            };
            app.transactions.push(tx);
            alert('Book issued successfully!');
            renderApp();
        }

        function returnBook(transactionId) {
            const tx = app.transactions.find(t => t.id === transactionId);
            const book = app.books.find(b => b.id === tx.book_id);

            tx.return_date = new Date().toISOString().split('T')[0];
            tx.status = 'returned';
            book.available++;

            alert('Book returned successfully!');
            renderApp();
        }

        function completeReturn(transactionId) {
            returnBook(transactionId);
        }

        function editBook(bookId) {
            alert('Edit functionality can be added for book management');
        }

        function deleteBook(bookId) {
            if (confirm('Are you sure you want to delete this book?')) {
                app.books = app.books.filter(b => b.id !== bookId);
                renderApp();
            }
        }

        function removeMember(memberId) {
            if (confirm('Are you sure you want to remove this member?')) {
                app.members = app.members.filter(m => m.id !== memberId);
                renderApp();
            }
        }

        function switchTab(tabId) {
```

```
            document.querySelectorAll('.tab-content').forEach(el =>
el.classList.remove('active'));
            document.querySelectorAll('.tab-button').forEach(el =>
el.classList.remove('active'));

            document.getElementById(tabId).classList.add('active');
            event.target.classList.add('active');
        }

        // Initial render
        renderApp();
    </script>
</body>
</html>
```



References

[1] A. Downey, Think Python: How to Think Like a Computer Scientist, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2015.

[2] M. Lutz, Learning Python, 5th ed. Sebastopol, CA, USA: O'Reilly Media, 2013.

[3] D. Greenfeld and A. Roy, Two Scoops of Django: Best Practices for Django, 3rd ed. Greenfeld & Roy, 2020.

[4] Django Software Foundation, Django Documentation. [Online]. Available: https://docs.djangoproject.com/

[5] W3Schools, HTML, CSS, and Python Tutorials. [Online]. Available: https://www.w3schools.com/

[6] Mozilla Developer Network (MDN), Web Development Documentation. [Online]. Available: https://developer.mozilla.org/

[7] GeeksforGeeks, Python and Web Development. [Online]. Available: https://www.geeksforgeeks.org/python/