1. Table of Contents

2. Table of Figures

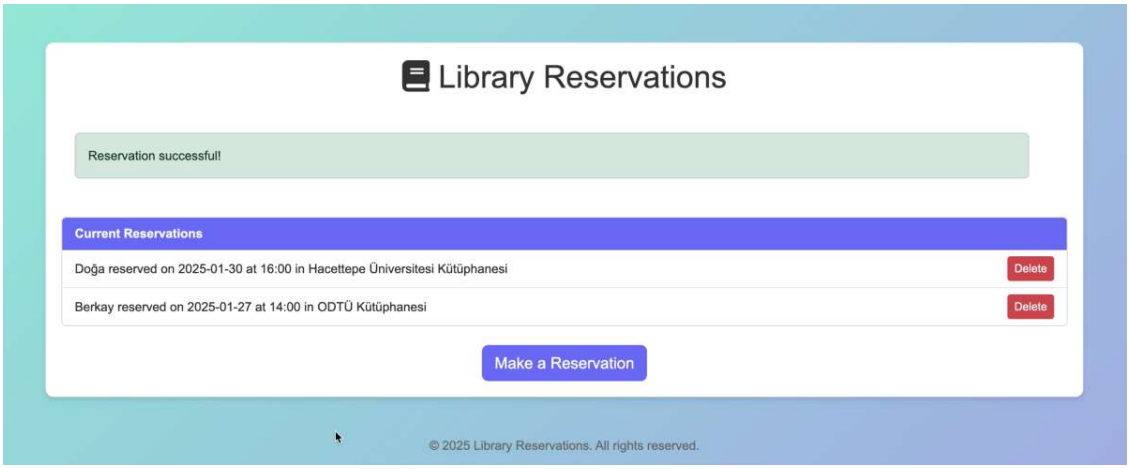

Figure.1 (Main Page)

Figure.2 (Reservation Form Date)

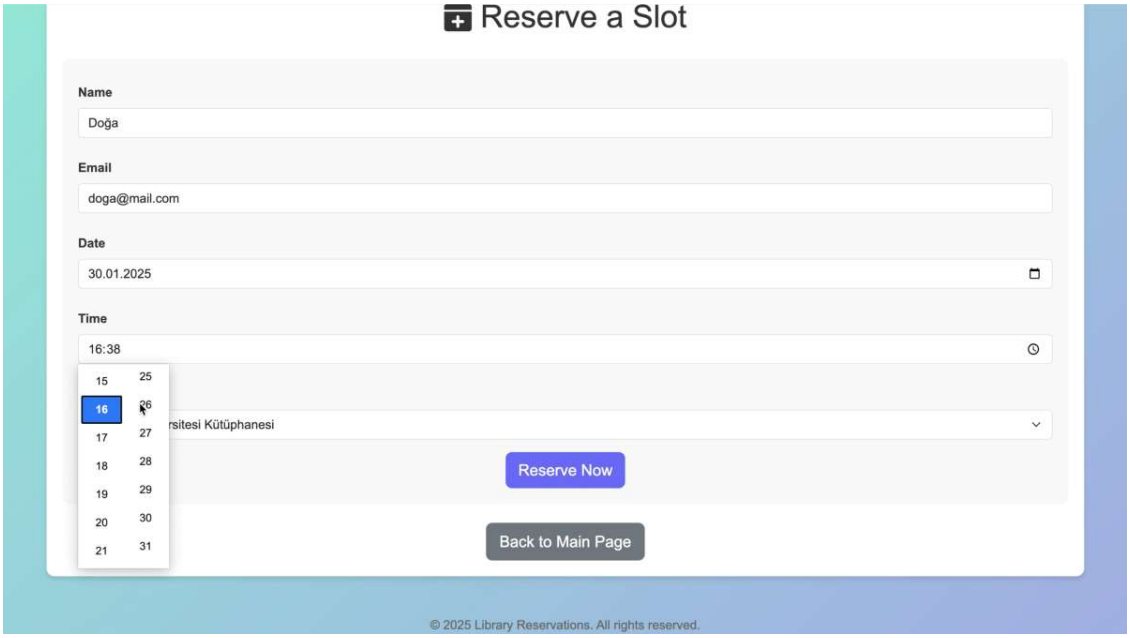

Figure.3 (Reservation Form Time)

Figure.4 (Reservation Form Library)

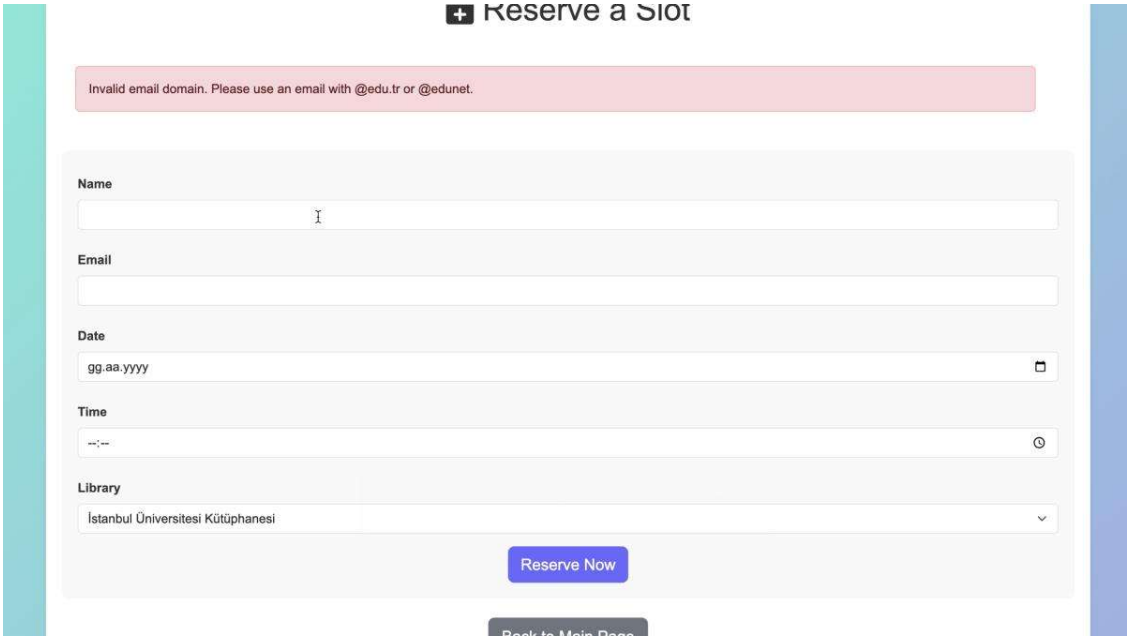

Figure.5 (Error Notification Invalid Domain)

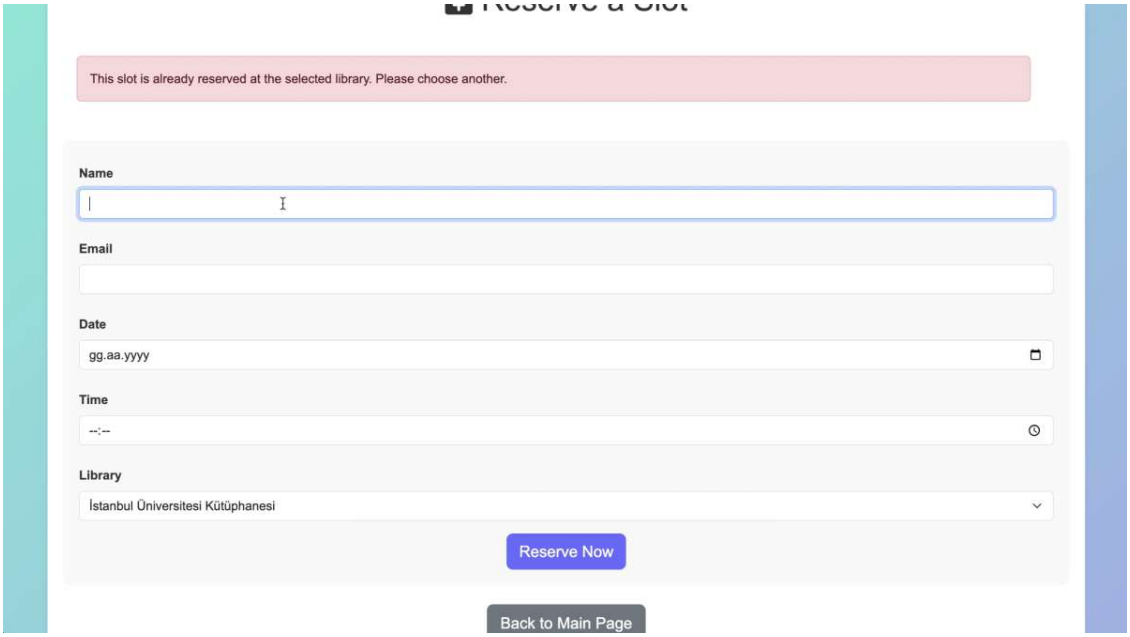

Figure.6 (Error Notification Not Available)

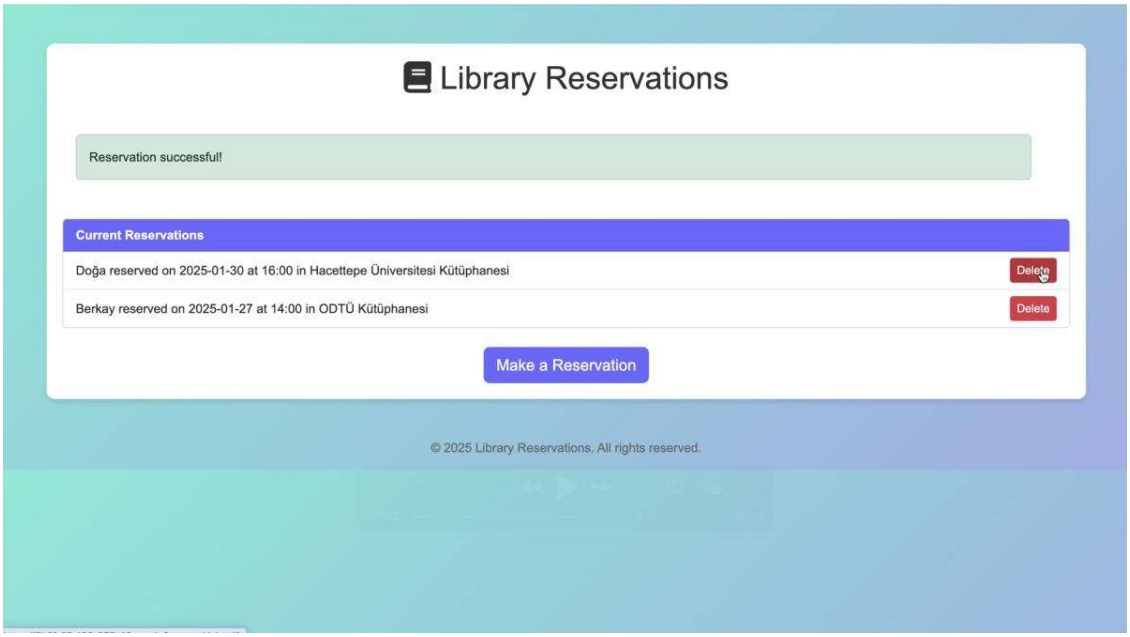

Figure.7 (Reservation Succesful Notification)

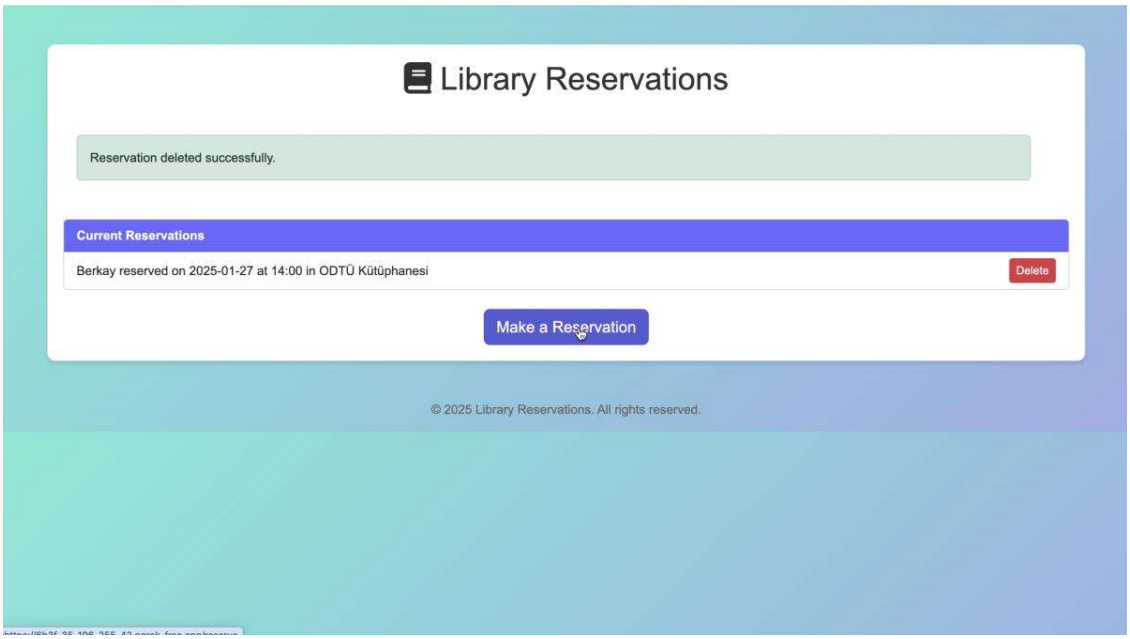

Figure.8 (Reservation Deletion Succesful Notification)

3. Introduction

The Library Reservation System (LRS) aims to enhance the reservation experience for university students by providing an efficient and user-friendly platform. This system ensures reliable management of library capacities, enabling students to reserve study spaces with ease.

Project Objective:

To develop a reservation system that:

- Validates users based on university email domains.
- Tracks and enforces library capacity limits.
- Provides accurate reservation validations and user notifications.

Scope:

The project covers critical functionalities outlined in the Software Requirements Specification (SRS), focusing on:

- User authentication via email validation.
- Reservation management with capacity tracking.
- A responsive user interface for seamless access.

4. Preliminary User Guide

Login and Registration:

- Only valid university email domains (e.g., `@edu.tr`, `@edunet`) can be used for login.
- Invalid email domains trigger an error message and deny access.

Making Reservations:

1. Navigate to the reservation page.
2. Provide the required details:
    - Name
    - Email
    - Reservation date and time
    - Library selection
3. Submit the form. A success or error notification will be displayed.

Viewing and Deleting Reservations:

- View current reservations on the main page.
- Delete reservations by clicking the associated “Delete” button.

Error Handling:

- **Invalid Date:** Users cannot reserve for past dates.
- **Full Capacity:** Notifications indicate fully booked libraries.
- **Invalid Email:** Users are alerted if their email domain is not accepted.

5. User Interfaces with Proof of Requirements

Screenshots:

- **Main Page:** Lists current reservations with user options for deletion (Figure.1)
- **Reservation Form:** Accepts inputs for name, email, date, time, and library (Figure.2, Figure.3, Figure.4)
- **Error Notifications:** Displays real-time feedback for invalid inputs (Figure.5, Figure.6, Figure.7, Figure.8)

Requirements Validation:

- **Email Validation:** Allows only approved domains (`@edu.tr`, `@edunet`).
- **Capacity Management:** Blocks reservations when the library reaches its capacity.
- **Time-Slot Enforcement:** Prevents duplicate reservations for the same slot.

6. Infrastructure Description and Technology Stack

1. Application Layer

- **Framework:** Flask, a lightweight web framework in Python, is used to develop the server-side logic and handle HTTP requests and responses.
- **Routes:** Flask routes manage user requests, including viewing reservations, creating new ones, and deleting existing ones.

2. Database Layer

- **Database:** Parsed database from recollected data.
    - **Tables:**
        - `Library`: Stores information about libraries, including name and capacity.

        - `Reservation`: Stores user reservations with attributes like name, email, date, time, and associated library.
    - **Data Flow:** The application interacts with the database using SQLAlchemy, an Object-Relational Mapping (ORM) tool.

3. Frontend Layer

- **Technologies:** HTML, CSS, and Bootstrap are used to create a responsive and user-friendly interface.
- **Components:**
    - **Reservation Form:** Allows users to input details for their reservations.
    - **Main Page:** Displays current reservations in a tabular format with action buttons for deletion.

4. Logging and Monitoring

- **Tool:** File-based logging is implemented using Python's `logging` module.
    - **Purpose:** Tracks key actions such as reservation creation and deletion.
    - **File:** Logs are saved in `reservations.log` for debugging and performance monitoring.

5. Hosting and Environment

- **Local Development Environment:** The system is developed and tested locally using Visual Studio Code.
- **Server Requirements:** Flask's built-in development server is used for local testing. For production deployment, a WSGI server like Gunicorn can be used.

6. API and Integration

- **Internal APIs:** Flask endpoints handle data flow between the frontend and backend, processing JSON data for efficient communication.
- **Integration Points:** Seamless communication between the application and the database is ensured through SQLAlchemy.

7. Scalability and Performance

- **Designed for Growth:** While SQLite is used for simplicity in this phase, the application can easily migrate to more robust databases (e.g., PostgreSQL or MySQL) for higher scalability.
- **Efficient Queries:** The use of ORM ensures efficient database queries, minimizing response times.

7. Fully Documented Program Code

```
from flask import Flask, render_template_string, request, redirect, url_for, flash
from pyngrok import ngrok
import json
from datetime import datetime
import os

app = Flask(__name__)
app.config['SECRET_KEY'] = '07cec91437b3a028905bf1c61342e0005d2cb5382b81277df50ae7c734127f19'

# File to store reservations
RESERVATION_FILE = "/content/reservations.json"

# In-memory storage for reservations and libraries
reservations = []
libraries = ["İstanbul Üniversitesi Kütüphanesi", "Hacettepe Üniversitesi Kütüphanesi",
             "Boğaziçi Üniversitesi Kütüphanesi", "ODTÜ Kütüphanesi", "Ege Üniversitesi Kütüphanesi"]

# Ensure the file exists and load reservations
def initialize_reservations():
    if not os.path.exists(RESERVATION_FILE):
        with open(RESERVATION_FILE, "w") as file:
            json.dump([], file)  # Create an empty JSON array
    else:
        return load_reservations()

# Load reservations from file
def load_reservations():
    with open(RESERVATION_FILE, "r") as file:
        return json.load(file)

# Save reservations to file
def save_reservations():
    with open(RESERVATION_FILE, "w") as file:
        json.dump(reservations, file)

# Initialize reservations at startup
reservations = initialize_reservations() or []
```

```
• 
• # HTML Templates
• index_html = '''
• <!DOCTYPE html>
• <html lang="en">
• <head>
•     <meta charset="UTF-8">
•     <meta name="viewport" content="width=device-width,
  initial-scale=1.0">
•     <title>Library Reservations</title>
•     <link
  href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bo
  otstrap.min.css" rel="stylesheet">
•     <link rel="stylesheet"
  href="https://cdnjs.cloudflare.com/ajax/libs/font-
  awesome/6.0.0/css/all.min.css">
•     <style>
•         body {
•             background: linear-gradient(135deg, #74ebd5,
  #9face6);
•             font-family: 'Arial', sans-serif;
•             color: #333;
•         }
•         .container {
•             margin-top: 50px;
•             background: white;
•             border-radius: 10px;
•             padding: 20px;
•             box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
•         }
•         .btn-primary {
•             background-color: #6c63ff;
•             border: none;
•             transition: background-color 0.3s;
•         }
•         .btn-primary:hover {
•             background-color: #5a54d4;
•         }
•         .header {
•             text-align: center;
•             margin-bottom: 30px;
•         }
•         .header h1 {
•             font-size: 2.5rem;
•             color: #333;
```

```
        }
        .card-header {
            background-color: #6c63ff;
            color: white;
            font-weight: bold;
        }
        .list-group-item {
            display: flex;
            justify-content: space-between;
            align-items: center;
        }
        .list-group-item .btn-danger {
            font-size: 0.9rem;
        }
        .form-container {
            padding: 20px;
            background: #f9f9f9;
            border-radius: 10px;
        }
        footer {
            margin-top: 50px;
            text-align: center;
            color: #666;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1><i class="fa-solid fa-book"></i> Library
Reservations</h1>
        </div>

        <!-- Flash Messages -->
        {% with messages =
get_flashed_messages(with_categories=true) %}
            {% if messages %}
                <div class="alert alert-dismissible fade show"
role="alert">
                    {% for category, message in messages %}
                        <div class="alert alert-{{ category
}}" role="alert">
                            {{ message }}
                        </div>
                    {% endfor %}
```

```
                </div>
            {% endif %}
        {% endwith %}

        <!-- Reservation List -->
        <div class="card">
            <div class="card-header">
                Current Reservations
            </div>
            <ul class="list-group list-group-flush">
                {% for res in reservations %}
                <li class="list-group-item">
                    {{ res['name'] }} reserved on {{ res['date'] }} at {{ res['time'] }} in {{ res['library'] }}
                    <a href="/delete/{{ loop.index0 }}" class="btn btn-danger btn-sm">Delete</a>
                </li>
                {% endfor %}
            </ul>
        </div>

        <!-- Reserve Button -->
        <div class="text-center mt-4">
            <a href="/reserve" class="btn btn-primary btn-lg">Make a Reservation</a>
        </div>
    </div>

    <footer>
        <p>&copy; 2025 Library Reservations. All rights reserved.</p>
    </footer>

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

reserve_html = '''
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Reserve a Slot</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bo
otstrap.min.css" rel="stylesheet">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css">
    <style>
        body {
            background: linear-gradient(135deg, #74ebd5,
#9face6);
            font-family: 'Arial', sans-serif;
            color: #333;
        }
        .container {
            margin-top: 50px;
            background: white;
            border-radius: 10px;
            padding: 20px;
            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
        }
        .btn-primary {
            background-color: #6c63ff;
            border: none;
            transition: background-color 0.3s;
        }
        .btn-primary:hover {
            background-color: #5a54d4;
        }
        .header {
            text-align: center;
            margin-bottom: 30px;
        }
        .header h1 {
            font-size: 2.5rem;
            color: #333;
        }
        .form-label {
            font-weight: bold;
            margin-top: 10px;
        }
        .form-control {
            border-radius: 5px;
```

```
        }
        .form-container {
            padding: 20px;
            background: #f9f9f9;
            border-radius: 10px;
        }
        footer {
            margin-top: 50px;
            text-align: center;
            color: #666;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1><i class="fa-solid fa-calendar-plus"></i>
Reserve a Slot</h1>
        </div>

        <!-- Flash Messages -->
        {% with messages =
get_flashed_messages(with_categories=true) %}
            {% if messages %}
                <div class="alert alert-dismissible fade show"
role="alert">
                    {% for category, message in messages %}
                        <div class="alert alert-{{ category
}}" role="alert">
                            {{ message }}
                        </div>
                    {% endfor %}
                </div>
            {% endif %}
        {% endwith %}

        <!-- Reservation Form -->
        <div class="form-container">
            <form method="POST">
                <div class="mb-3">
                    <label for="name" class="form-
label">Name</label>
                    <input type="text" class="form-control"
id="name" name="name" required>
                </div>
```

```
                <div class="mb-3">
                    <label for="email" class="form-
label">Email</label>
                    <input type="email" class="form-control"
id="email" name="email" required>
                </div>
                <div class="mb-3">
                    <label for="date" class="form-
label">Date</label>
                    <input type="date" class="form-control"
id="date" name="date" required>
                </div>
                <div class="mb-3">
                    <label for="time" class="form-
label">Time</label>
                    <input type="time" class="form-control"
id="time" name="time" required>
                </div>
                <div class="mb-3">
                    <label for="library" class="form-
label">Library</label>
                    <select class="form-select" id="library"
name="library" required>
                        {% for lib in libraries %}
                        <option value="{{ lib }}">{{ lib
}}</option>
                        {% endfor %}
                    </select>
                </div>
                <div class="text-center">
                    <button type="submit" class="btn btn-
primary btn-lg">Reserve Now</button>
                </div>
            </form>
        </div>

        <div class="text-center mt-4">
            <a href="/" class="btn btn-secondary btn-lg">Back
to Main Page</a>
        </div>
    </div>

    <footer>
        <p>&copy; 2025 Library Reservations. All rights
reserved.</p>
```

```
    </footer>

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

@app.route('/')
def index():
    return render_template_string(index_html, reservations=reservations)

@app.route('/reserve', methods=['GET', 'POST'])
def reserve():
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        date = request.form['date']
        time = request.form['time']
        library = request.form['library']

        # Check if the email domain is valid
        if not (email.endswith('@edu.tr') or email.endswith('@edunet')):
            flash('Invalid email domain. Please use an email with @edu.tr or @edunet.', 'danger')
            return render_template_string(reserve_html, libraries=libraries)

        # Check if the selected date is earlier than the current date
        today = datetime.today().date()
        selected_date = datetime.strptime(date, "%Y-%m-%d").date()
        if selected_date < today:
            flash('Reservations cannot be made for past dates.', 'danger')
            return render_template_string(reserve_html, libraries=libraries)

        # Check for existing reservations at the same time and library
        for res in reservations:
```

```
            if res['date'] == date and res['time'] == time and
res['library'] == library:
                flash('This slot is already reserved at the
selected library. Please choose another.', 'danger')
                return render_template_string(reserve_html,
libraries=libraries)

        # Add the reservation to the list
        reservations.append({'name': name, 'email': email,
'date': date, 'time': time, 'library': library})
        save_reservations()  # Save reservations to file
        flash('Reservation successful!', 'success')
        return redirect(url_for('index'))

    return render_template_string(reserve_html,
libraries=libraries)

@app.route('/delete/<int:index>')
def delete(index):
    if 0 <= index < len(reservations):
        reservations.pop(index)
        save_reservations()  # Save updated reservations to
file
        flash('Reservation deleted successfully.', 'success')
    return redirect(url_for('index'))

# Set up ngrok for public URL
public_url = ngrok.connect(5000)
print(f"Public URL: {public_url}")

# Start Flask app
app.run(port=5000)
```

8.Appendices and Glossary

Glossary:

- **Capacity:** Maximum reservations allowed per library.
- **Validation:** Ensures user input meets system criteria.
- **Flash Messages:** Notifications for users about their actions.

Library Capacities:

- İstanbul Üniversitesi Kütüphanesi
- Hacettepe Üniversitesi Kütüphanesi
- Boğaziçi Üniversitesi Kütüphanesi
- ODTÜ Kütüphanesi
- Ege Üniversitesi Kütüphanesi