CASE STUDY 3

Online Movie Ticket Booking REST API – Development & Testing

Problem Statement

A cinema chain wants to build a RESTful backend service for an **Online Movie Ticket Booking System**.
The system should allow users to view movies, check show timings, and book tickets.
All APIs must be **manually tested** and **automated**.

Objectives

Students should be able to:

- Understand REST API architecture
- Build REST APIs using Flask
- Design CRUD-based endpoints
- Test APIs using Postman
- Automate API tests using `requests` & `pytest`
- Apply unit testing and TDD concepts

Functional Requirements

1. REST API Design

Follow REST principles:

- Stateless communication
- Resource-based URLs

- JSON request/response
- Proper HTTP status codes

---

2. API Endpoints to Implement (Flask)

HTTP Verb	Endpoint	Description
GET	/api/movies	Retrieve all movies
GET	/api/movies/	Get movie by ID
POST	/api/movies	Add a new movie
PUT	/api/movies/	Update movie details
DELETE	/api/movies/	Delete a movie
POST	/api/bookings	Book movie tickets

---

3. Sample Movie Data (JSON)

```
{
  "id": 101,
  "movie_name": "Interstellar",
  "language": "English",
  "duration": "2h 49m",
  "price": 250
}
```

Testing Requirements

🔹 Manual API Testing (Postman)

Students must:

- Perform GET, POST, PUT, DELETE requests
- Use headers: `Content-Type: application/json`
- Validate:
  - o Status codes (200, 201, 400, 404)
  - o Response payload
  - o Booking failure scenarios

🔹 Automated API Testing (Python)

Use:

- `requests` library
- `pytest`

Test Scenarios:

- Fetch all movies
- Add a new movie
- Book tickets
- Validate JSON responses
- Assert HTTP status codes

Automation Framework Concepts Applied

- Pytest test discovery
- Fixtures for setup/teardown
- Parameterized tests
- Assertions & exception handling
- Command-line execution
- HTML test reports

CASE STUDY 4

Hospital Management System – Web, API & Robot Framework Automation

---

Problem Statement

A hospital wants to automate testing for its **Hospital Management System**, which includes:

- A web portal for patient registration
- REST APIs for patient records
- Automated testing using Pytest and Robot Framework

Objectives

Students should:

- Work with REST APIs and Web applications
- Automate API tests using `requests`
- Parse HTML content
- Perform JSON serialization/deserialization
- Implement Robot Framework automation
- Execute tests via command line

System Components

---

1. Patient REST API (Flask)

HTTP Verb	Endpoint	Description
GET	/api/patients	Fetch all patients
POST	/api/patients	Register a patient
GET	/api/patients/	Get patient details
PUT	/api/patients/	Update patient info

---

2. Web Page (HTML)

Patient Registration Form:

- Patient Name
- Age
- Gender
- Contact Number
- Disease
- Doctor Assigned

Form submits data to backend REST API.

Automation Tasks

🔹 API Automation (Python + requests)

Students must:

- Send GET and POST requests
- Pass headers and request body
- Validate API responses
- Deserialize JSON responses
- Handle negative test cases

🔹 **Web Scraping Task**

Use:

- BeautifulSoup / lxml

Extract from patient list page:

- Patient name
- Age
- Disease
- Assigned doctor

Pytest Automation Requirements

- Unit tests for API endpoints
- API fixtures
- `conftest.py` usage
- Parameterized tests for multiple patients
- Skip and xfail tests
- Parallel execution (`pytest-xdist`)
- Generate HTML reports

Robot Framework Automation

Tasks to Perform:

- Install Robot Framework & Selenium Library
- Write test cases using keyword-driven approach
- Implement data-driven testing for patients
- Automate:
    - Browser launch
    - Text field input
    - Radio button & checkbox selection

    - Dropdown handling
- Implement suite setup & teardown
- Execute tests via command line