# FINAL MILESTONE REPORT — SMARTLEARN

---

## SMARTLEARN – FINAL MILESTONE PROJECT REPORT

**Name:** *Amirthavarshini S*
**Program:** *Full Stack MERN*
**Date:** *29 NOVEMBER 2025*

---

# 1. Introduction

This report presents the implementation of the Final Milestone Project titled **"SmartLearn – Online Course Catalog & Enrollment System"** using the **MERN Stack** (MongoDB, Express.js, React.js, Node.js).

The goal of this application is to:

- Display a course catalog
- Allow users to enroll in a course
- Prevent duplicate enrollments
- Validate API inputs
- Implement proper error handling
- Write automated tests (Mocha + Chai + SuperTest)
- Build a working frontend that interacts with backend APIs

This report includes the architecture, database schema, API endpoints, frontend integration, testing results, and screenshots.

---

# 2. Project Architecture

The project follows a **modular MERN architecture**, with clear separation of:

- **Backend** (Node.js + Express + MongoDB)
- **Frontend** (React.js)
- **Test Suite** (Mocha + Chai + SuperTest)
- **Documentation + Screenshots**

### Folder Structure

```
smartlearn-milestone/
├─ backend/
│  ├─ models/
│  ├─ routes/
│  ├─ middleware/
│  ├─ seed/
│  ├─ test/
│  ├─ server.js
│  └─ .env
├─ frontend/
├─ screenshots/
├─ README.md
└─ report.pdf
```

---

# 3. Backend Implementation

## 3.1 Database Schema (MongoDB + Mongoose)

### Course Schema

Fields:

- courseId *(String, unique, required)*
- title *(String, required)*

- category *(String, required)*
- price *(Number, required, >= 0)*
- createdAt *(auto timestamp)*

### Enrollment Schema

Fields:

- userId *(String, required)*
- courseId *(String, required)*
- enrolledOn *(Date)*
- Unique constraint: *(userId, courseId)* → prevents duplicate enrollments

---

## 3.2 API Endpoints

### POST /api/courses

Creates a new course
Includes field validation using *express-validator*

### GET /api/courses

Fetches all available courses

### POST /api/enroll

Allows a user to enroll in a course
Validations:

- Required fields
- Course existence check
- Duplicate enrollment prevention

---

## 3.3 Error Handling Middleware

A global middleware catches all server errors and returns responses in the format:

```
{
  "success": false,
  "data": null,
  "message": "Error message"
}
```

---

## 3.4 Input Validation

Validation is applied using **express-validator** for:

- Creating a course
- Enrolling in a course

Ensures clean and secure data.

---

# 4. Frontend Implementation (React.js)

The frontend is built using Create React App.
It performs:

### ✔ Fetch and display course catalog

### ✔ Show loading / error states

### ✔ Provide “Enroll Now” button

### ✔ Show successful & duplicate enrollment messages

### ✔ Maintain enrolled courses list

Key components:

### CourseList.js

- Fetches all courses from backend
- Displays course details
- Handles enrollment actions

### api.js Service

Central place for calling backend APIs using fetch()

---

# 5. Testing (Mocha + Chai + SuperTest)

The testing suite validates API behavior:

### Test Cases

1. Should return **201** on successful enrollment
2. Should return **400** on duplicate enrollment

The test automates API calls and asserts correct responses.

### Expected Output

```
2 passing
```

This confirms backend functionality is correct.

---

# 6. Seeding the Database

A seed script (`seedCourses.js`) is created to insert initial courses:

- React for Beginners
- Node & Express Mastery

This ensures the React UI displays data immediately.

---

# 7. Screenshots Included

The screenshots folder contains:

1. **backend_get_courses.png** – GET /api/courses response
2. **backend_post_course.png** – POST /api/courses success
3. **tests_pass.png** – Mocha passing tests
4. **frontend_course_list.png** – UI course list
5. **frontend_enroll_success.png** – Enrollment success UI

These screenshots demonstrate the working functionality.

---

# 8. Conclusion

The Final Milestone Project successfully meets all required objectives:

- Complete MERN stack implementation
- Validations and error handling
- Database schema with unique constraints
- Automated testing
- Fully functional React frontend
- Clean folder structure and documentation
- Screenshots for verification

This project demonstrates the ability to design, build, test, and document a full-stack MERN application