# Personalized Learning System - Project Documentation

## 1. Project Overview

This project implements a Personalized Learning System that helps students receive tailored recommendations based on their placement tests, learning styles, and academic performance. The system provides a comprehensive solution for managing student data, academic progress, and generating personalized recommendations.

## 2. System Architecture

### 2.1 Database Schema

The system uses MySQL with the following key entities: - Students - Departments - Teachers - Subjects - Courses - Learning Styles - Placement Tests - Recommendations - Assignments - Submissions - Exams

### 2.2 Backend (Node.js)

* Express.js server
* MySQL2 for database connectivity
* RESTful API endpoints for:
  + Student management
  + Subject management
  + Placement test processing
  + Recommendation retrieval

### 2.3 Frontend

* HTML/JavaScript-based UI
* Separate pages for:
  + Main dashboard
  + Recommendations view
  + Subjects proficiency view

## 3. Key Features

### 3.1 Student Management

* Student registration and authentication
* Department assignment
* Learning style tracking
* Academic progress monitoring

### 3.2 Placement Testing

* Multi-subject placement tests
* Automatic scoring and evaluation
* Performance tracking across subjects

### 3.3 Recommendation System

* Automated recommendation generation based on:
  + Placement test scores
  + Learning style preferences
  + Subject performance
* Detailed markdown-formatted reports including:
  + Performance summaries
  + Subject breakdowns
  + Personalized feedback
  + Growth recommendations
  + Department recommendations

### 3.4 Academic Progress Tracking

* Assignment management
* Submission tracking
* Grade recording
* Performance analytics

## 4. Database Design

### 4.1 Core Tables

* Students: Stores student personal information and credentials
* Departments: Manages academic departments
* Subjects: Contains subject information and teacher assignments
* Courses: Manages course offerings and department associations

### 4.2 Junction Tables

* Student\_department: Links students to departments
* Course\_Subjects: Maps subjects to courses
* Subjects\_good\_at: Tracks student proficiency levels

### 4.3 Performance Tables

* Placement\_tests: Records test scores
* Assignments: Manages student assignments
* Submissions: Tracks assignment submissions
* Exams: Stores exam results

## 5. API Endpoints

### 5.1 GET Endpoints

* /subjects: Retrieve all subjects
* /students: Get student list
* /recommendations/:studentId: Get student recommendations
* /subjects-good-at/:studentId: Get student subject proficiencies

### 5.2 POST Endpoints

* /placement-tests: Submit placement test scores

## 6. Database Operations

### 6.1 Triggers

* add\_subjects\_good\_at: Processes placement test results
* generate\_detailed\_recommendations: Creates personalized recommendations

### 6.2 Stored Procedures

* AddNewStudent: Student registration
* GetStudentRecommendations: Retrieves recommendations
* UpdateAssignmentGrade: Grade management
* GetStudentAssignments: Assignment retrieval
* authenticate: User authentication

### 6.3 Views

* student\_academic\_info: Academic information
* student\_assignment\_info: Assignment details
* student\_exam\_results: Exam performance
* Assignment\_Submissions\_View: Submission tracking
* Student\_Recommendations\_View: Recommendation summaries

## 7. Performance Optimization

### 7.1 Indexes

* Student indexes (email, department)
* Subject indexes (teacher)
* Course indexes (department)
* Performance indexes (student\_id, subject\_id)
* Recommendation indexes (student\_id)

## 8. Security Features

* Password-protected student accounts
* Session-based authentication
* Input validation and sanitization
* Prepared SQL statements

## 9. Frontend Components

### 9.1 Main Pages

* index.html: Main entry point
* recommendations.html: Recommendation display
* subjects-good-at.html: Subject proficiency view

### 9.2 JavaScript Modules

* main.js: Core functionality
* recommendations.js: Recommendation handling
* subjects-good-at.js: Subject proficiency display

## 10. Future Enhancements

1. Real-time progress tracking
2. Advanced analytics dashboard
3. Machine learning-based recommendations
4. Interactive learning path visualization
5. Mobile application support
6. Integration with external learning platforms

## 11. Installation and Setup

### Prerequisites

* Node.js v14+
* MySQL 8.0+
* Web browser with JavaScript enabled

### Setup Steps

1. Clone the repository
2. Install dependencies: npm install
3. Configure MySQL connection in server.js
4. Run SQL scripts in order:
   * schema.sql
   * indexes.sql
   * procedures.sql
   * triggers.sql
   * views.sql
   * insertions.sql
5. Start the server: node server.js

## 12. Testing

* Database integrity tests
* API endpoint testing
* Frontend unit tests
* Integration testing
* Performance testing

## 13. Maintenance

* Regular database backups
* Log monitoring
* Performance optimization
* Security updates
* Bug fixes and patches

## 14. Contact and Support

For technical support or contributions, please contact the development team.

*Last Updated: May 31, 2025*