*Mohammad Ali Jinnah University*

# Software Design Specification (SDS)

For

# Advisify


## Members

Ali Raza Bugti

Muhammad Ahmed

Arfat Ayub

## Supervisor

Syed Haider Imam Jaffery


| | |
|---|---|
| *Project Code* | *FA25-FYP-31* |
| *Supervisor* | *Syed Haider Imam Jaffery* |
| *Project Team* | *Ali Raza Bugti FA22-BSCS-0136*<br>*Muhammad Ahmed FA22-BSCS-0210*<br>*Arfat Ayub FA22-BSCS-0138* |
| *Submission Date* | *20/12/2025* |

**Table of Contents**

# 1. Introduction

## 1.1 Purpose of the Document

The purpose of this Software Design Specification (SDS) is to describe the detailed design of the proposed Final Year Project titled **Advisify – AI- Assisted Course Registration System**. This document provides a clear understanding of the system architecture, database design, software components, workflows, and user interface. The SDS serves as a technical reference for developers, supervisors, and evaluators.

## 1.2 Scope of the System

Advisify is an admin-only, AI- assisted academic system designed to automate the course registration and advisory process for BSCS students. The system processes academic transcripts, detects advisory cases, allocates semester wise courses based on academic rules, generates timetables, and prepares database-ready registration data.

The system eliminates manual advisory processes, reduces workload on faculty advisors, and ensures rule - based, consistent, and efficient course registration.

## 1.3 Definitions, Acronyms, and Abbreviations

- **SDS**: Software Design Specification
- **ERD**: Entity Relationship Diagram
- **CSV**: Comma Separated Values
- **PDF**: Portable Document Format
- **AI**: Artificial Intelligence
- **FYP**: Final Year Project

# 2. System Overview

The system is designed as a multi-tier application consisting of a frontend interface, a Node.js gateway backend, a FastAPI based multi agent processing layer, and a PostgreSQL database. The system takes academic transcript PDFs as input and produces automated course registration and timetable outputs.

## 2.1 High Level Workflow

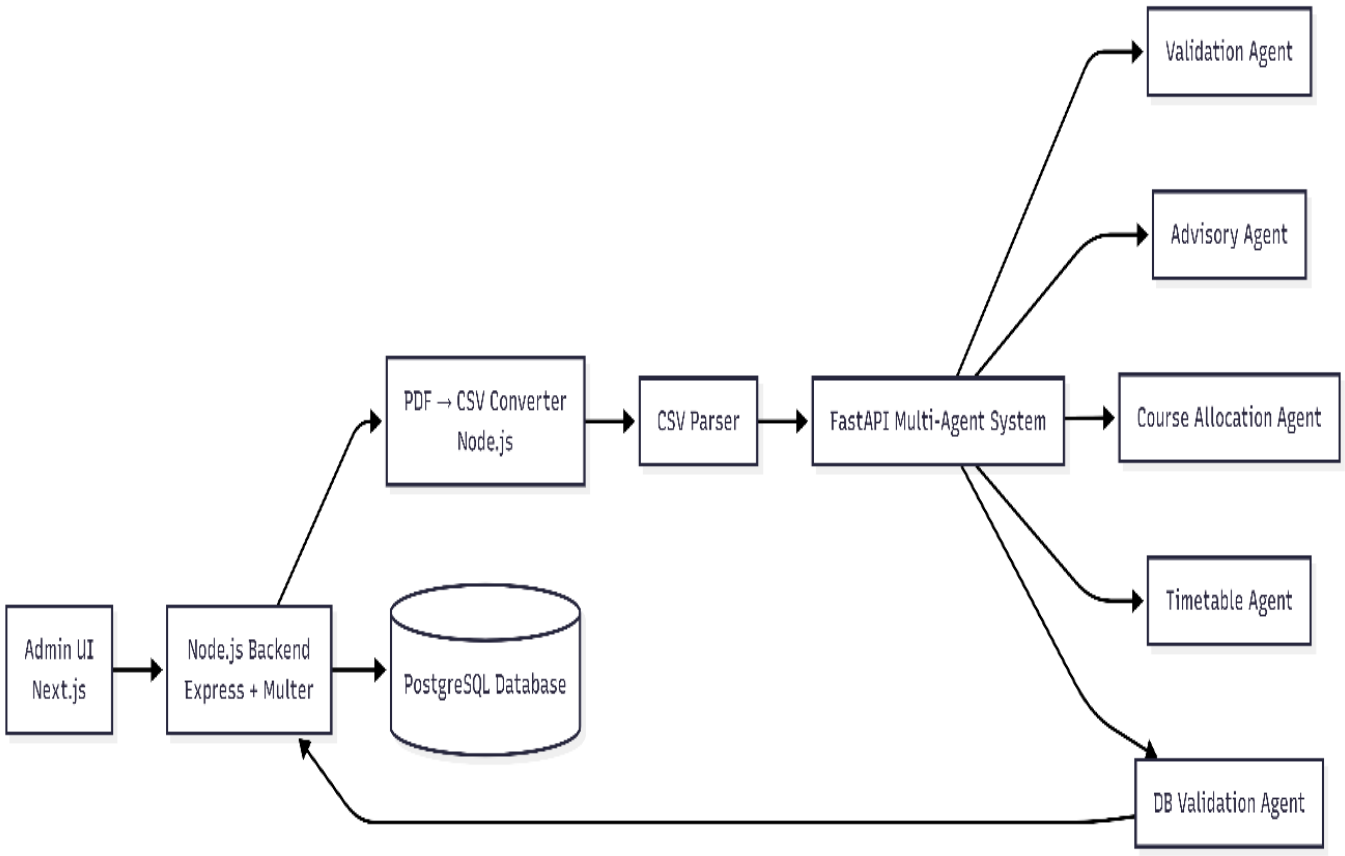
1. Admin uploads student transcript PDFs.
2. Node.js backend converts PDFs into standardized CSV format.
3. CSV data is parsed and transformed into structured JSON.
4. Data is processed by a FastAPI based multi- agent system.
5. Final validated data is returned to Node.js.
6. Node.js writes the data into the PostgreSQL database.

## 3. System Architecture Design

The system follows a modular and service oriented architecture to ensure scalability, maintainability, and separation of concerns.

### 3.1 Architectural Components

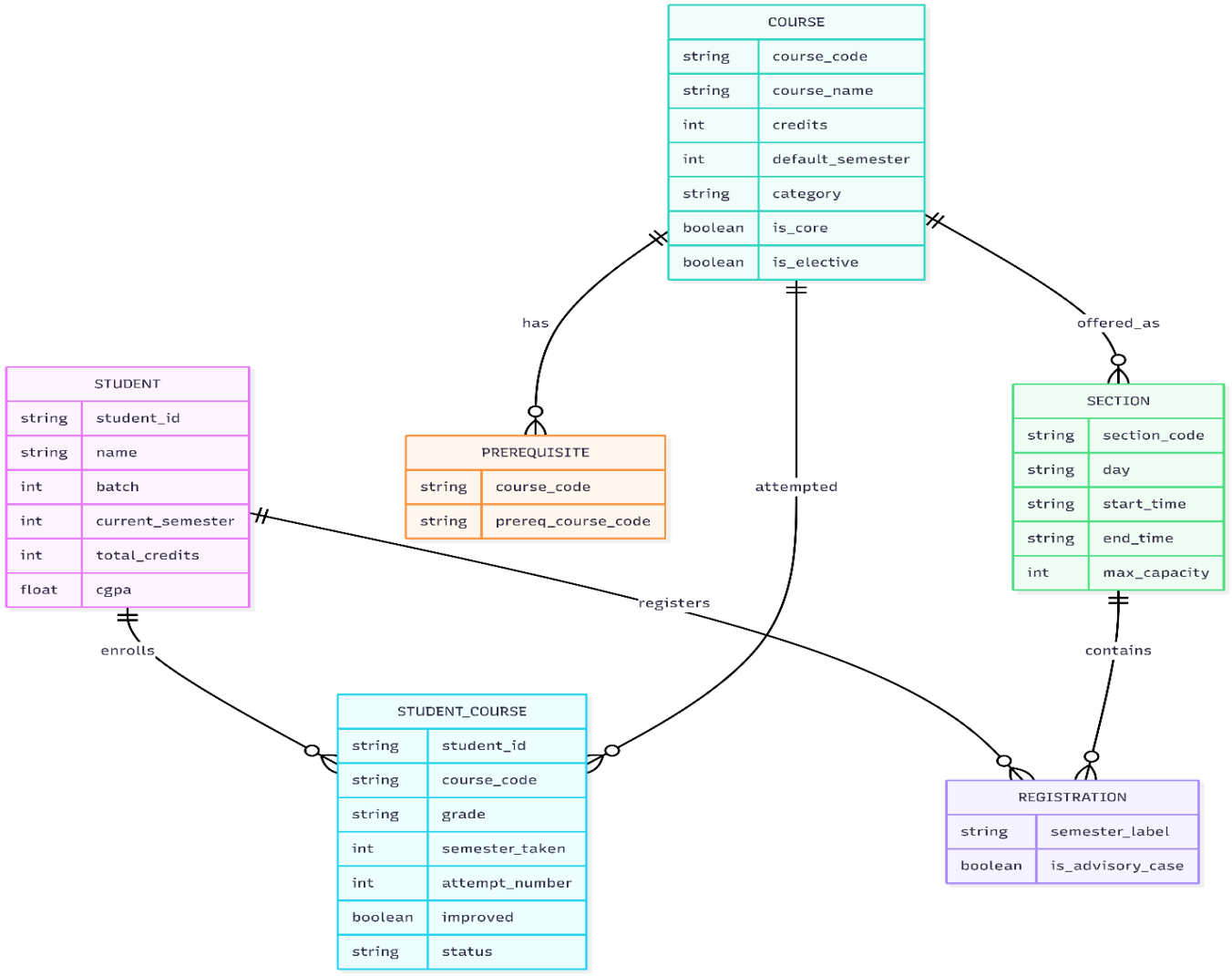
- **Frontend (Next.js)**: Admin dashboard for file upload automation and result visualization.
- **Backend Gateway (Node.js)**: Handles file uploads, PDF to CSV conversion, CSV parsing, and database operations.
- **AI Processing Layer (FastAPI)**: Executes academic logic using multiple intelligent agents.
- **Database (PostgreSQL)**: Stores students, courses, registrations, and timetable data.



# 4. Data Design

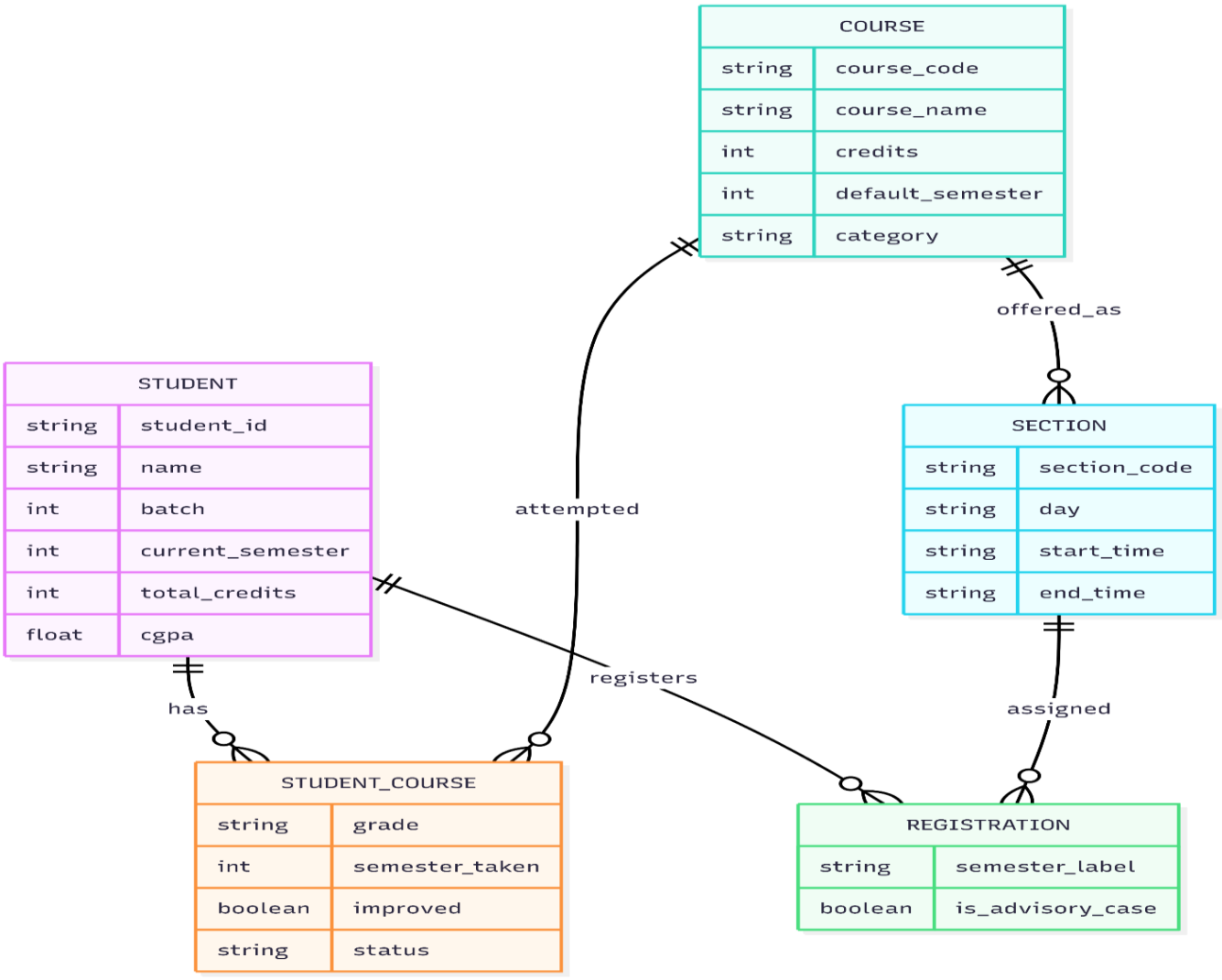
## 4.1 Entity Relationship Design

The system database is designed to store academic and registration-related data in a structured manner. Key entities include Student, Course, Student Course Attempts, Prerequisites, Sections, and Registrations. Relationships between these entities ensure data integrity and support complex academic queries.



## 4.2 Database Design

The database schema is implemented in PostgreSQL and includes normalized tables with primary and foreign key constraints. The design supports multiple course attempts, advisory case tracking, timetable sections, and semester wise registrations.

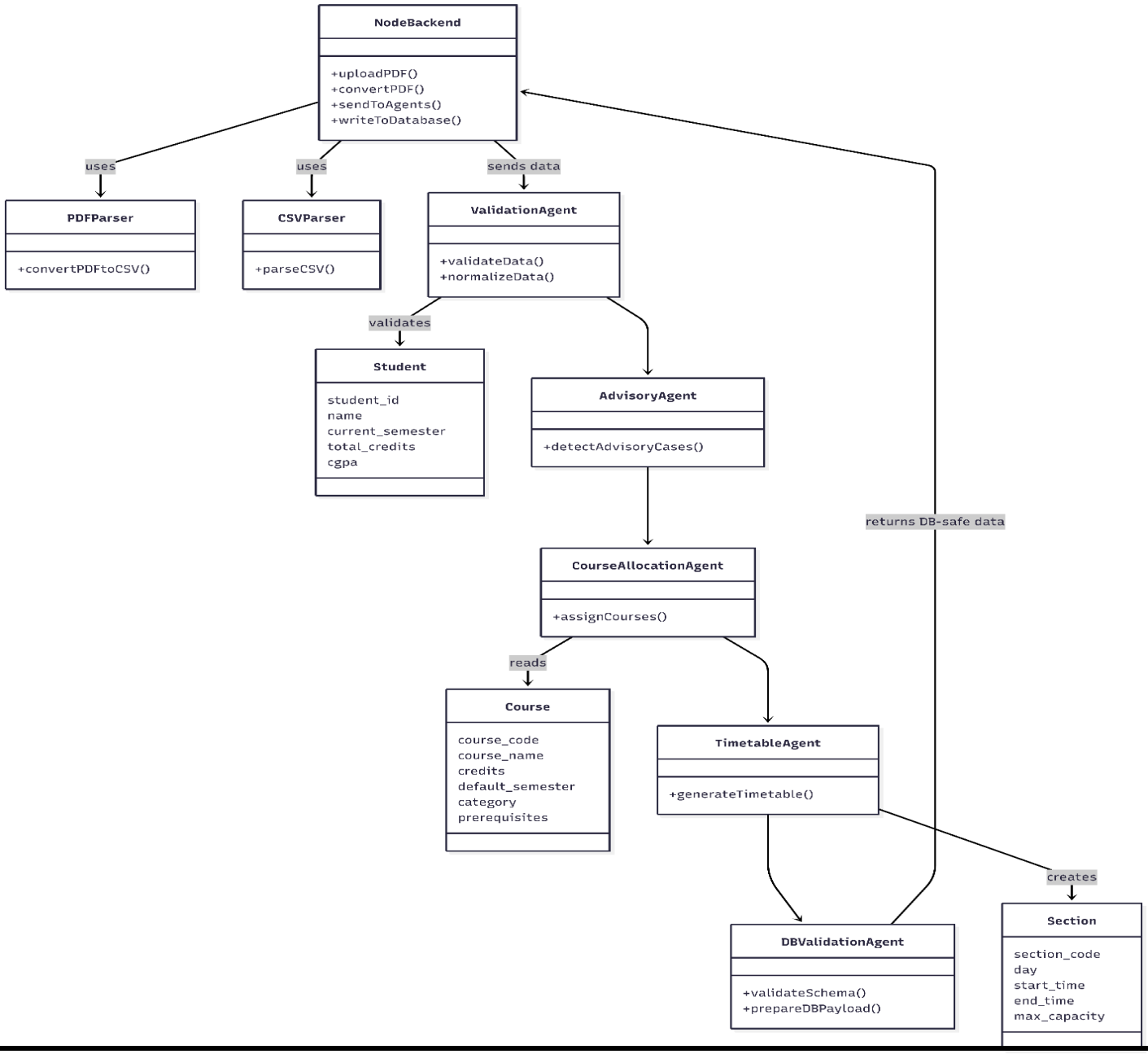


# 5. Component Design (Class Diagram)

The system logic is implemented using a modular, agent- based approach. Each agent has a specific responsibility and processes data sequentially.

## 5.1 Core Classes / Agents

- **PDFParser**: Converts transcript PDFs into CSV format.
- **ValidationAgent**: Validates and normalizes academic data.
- **AdvisoryAgent**: Detects advisory cases based on failed or missing courses.
- **CourseAllocationAgent**: Allocates semester - wise courses using academic rules.
- **TimetableAgent**: Generates clash- free timetables and assigns sections.
- **DBValidationAgent**: Validates final data for database insertion.

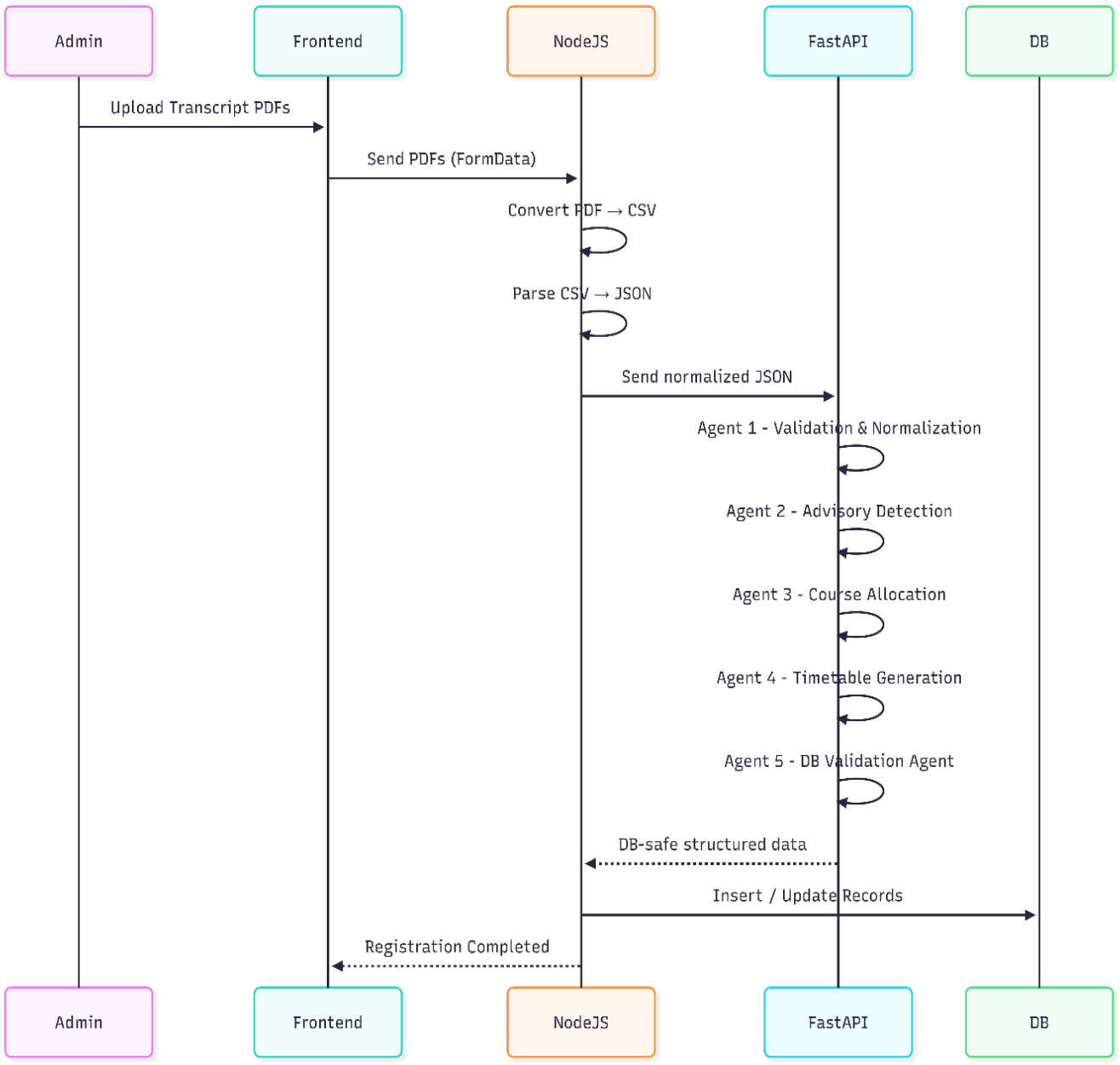


# 6. Sequence Design

The sequence design describes the interaction between system components during the execution of the registration process.

## 6.1 Registration Flow

- Admin initiates the process by uploading transcript PDFs.
- Node.js processes files and forwards data to FastAPI agents.
- Agents execute validation, advisory detection, allocation, and timetable generation.
- Final validated data is returned to Node.js for database storage.

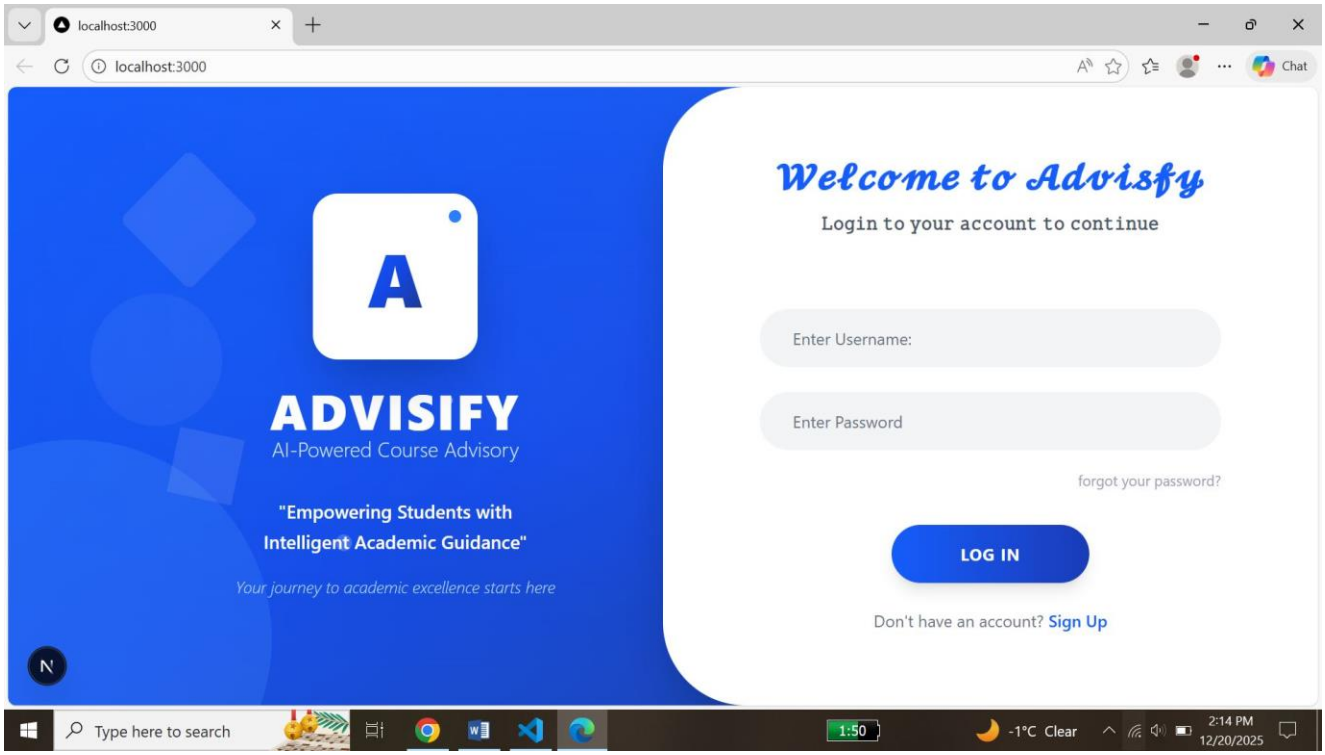


# 7. Interface Design

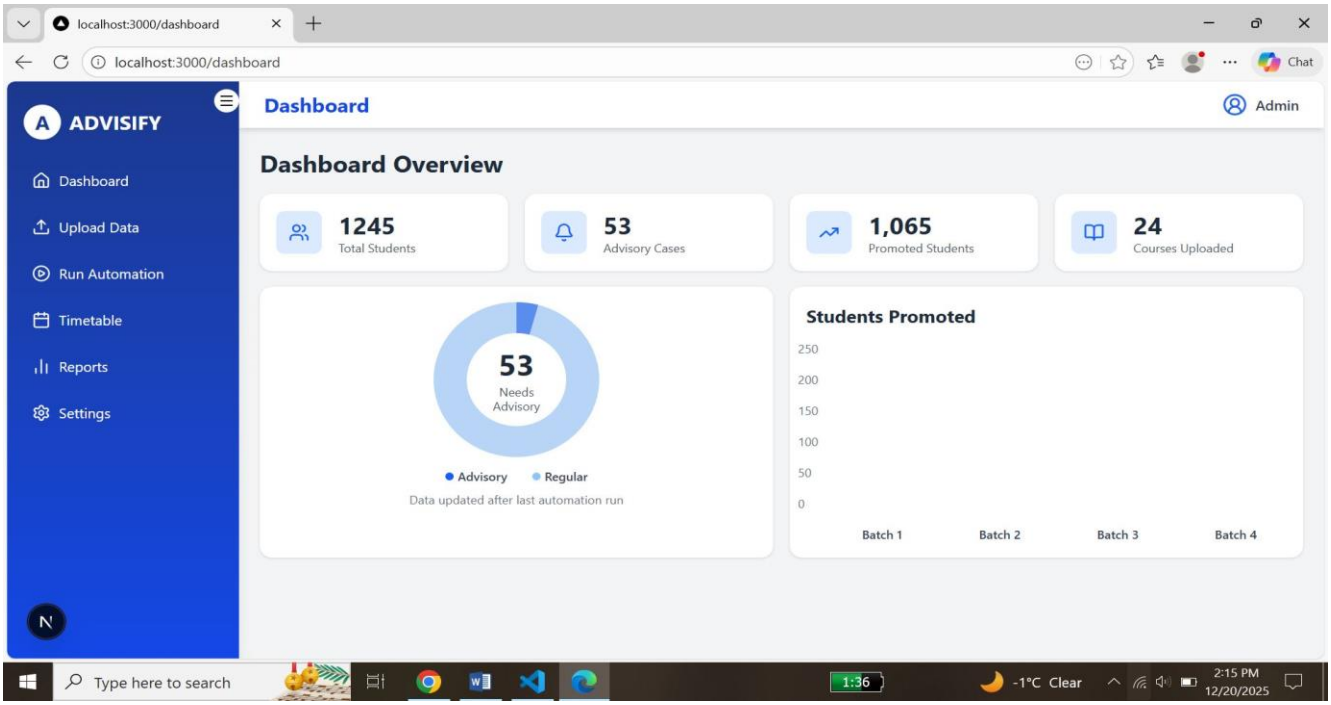
The system provides a simple and intuitive admin- only user interface.

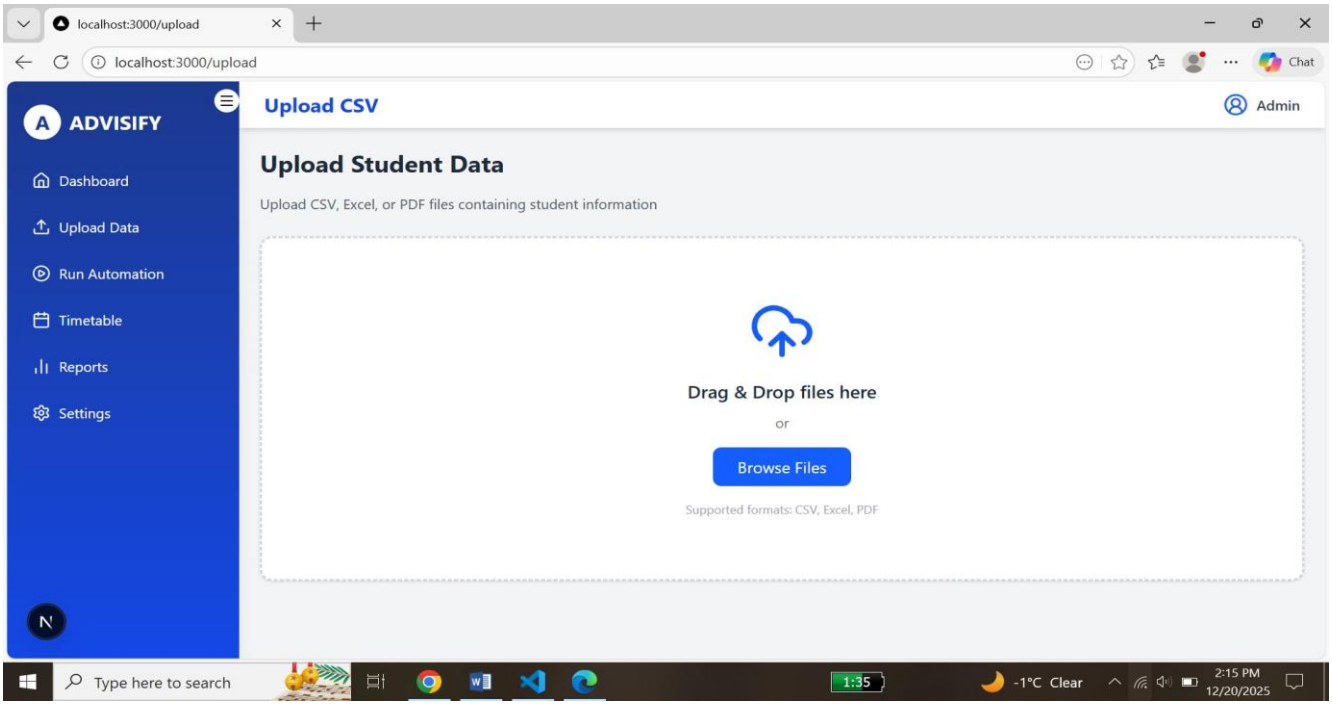
## 7.1 Admin Interface Features

- **Admin Authentication Page**

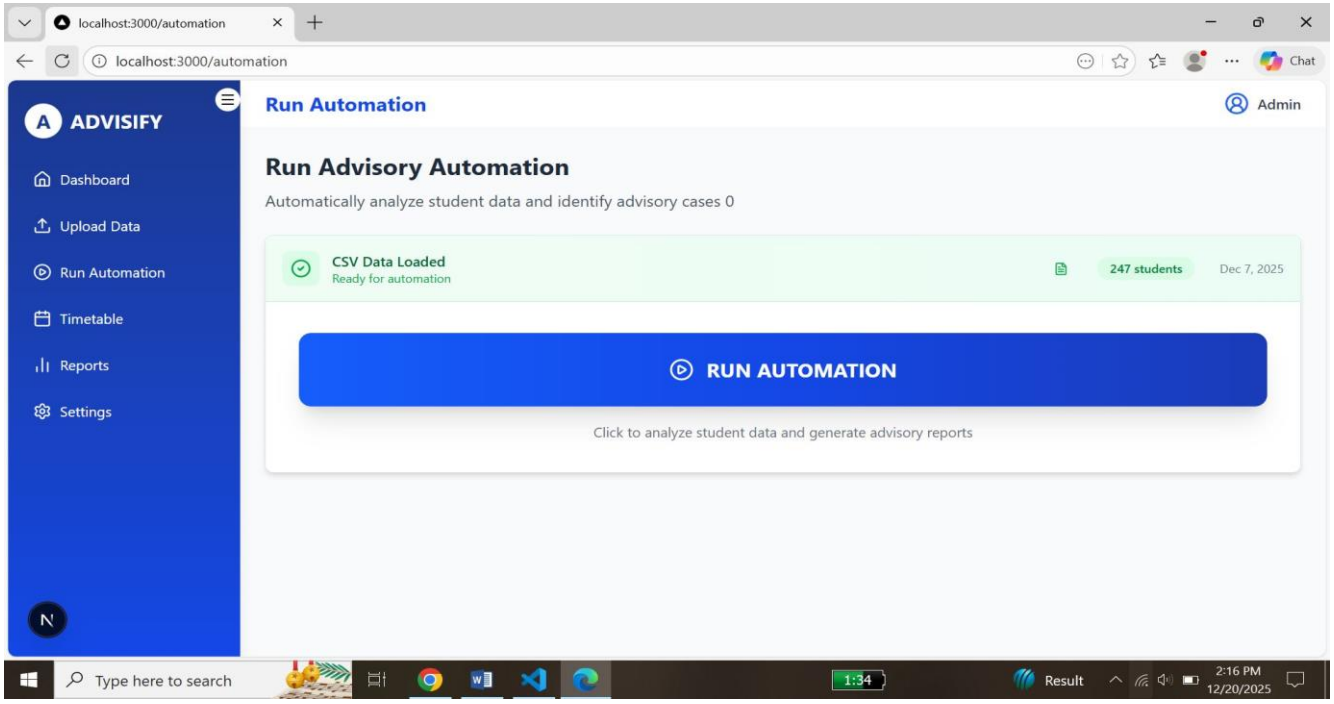



- **Dashboard**

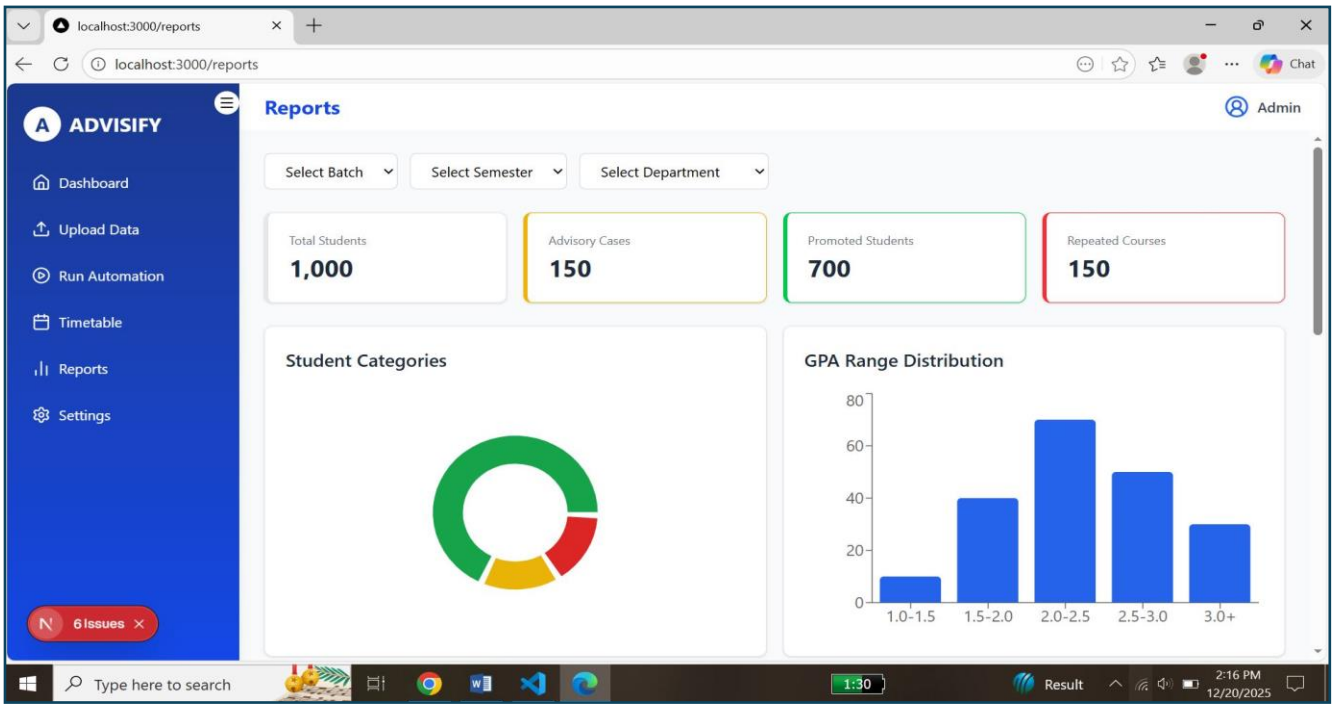





- **Run Automation page**




- **Reports Page**




## 8. Design Constraints and Assumptions

- The system is accessible only to administrative users.
- Course outlines and prerequisite rules are pre- stored in the database.
- Academic rules follow the official BSCS curriculum.
- Transcript PDFs follow a consistent university format.

## 9. Conclusion

This Software Design Specification provides a complete design overview of the Advisify system. The modular architecture, rule- based AI agents, and structured database design ensure scalability, accuracy, and efficiency in automating the academic course registration process. The SDS serves as a blueprint for system implementation and evaluation.