# Course Recommendation System Technical Report

## 1. Title Page

### Intelligent Course Recommendation System

**Declaration**: We, TeamML Excellence, confirm that this work submitted for assessment is our own and is expressed in our own words. Any uses made within it of the works of any other author, in any form (ideas, equations, figures, texts, tables, programs), are properly acknowledged at the point of use. A list of the references used is included.

**From**: -

* Daljeet Singh Bhinder
* Gautam Khatri
* Ricardo Schwarz Costamilan

**Course**: AI Capstone (COMP 385 - Section 002)

**Date**: March 21, 2025

## 2. Executive Summary

Students often struggle to select appropriate courses aligned with their career goals and academic strengths, leading to suboptimal academic outcomes. Our intelligent course recommendation system benefits students and academic advisors by providing personalized course suggestions based on individual student profiles and predicted likelihood of success. We solved this problem using machine learning techniques, specifically logistic regression, with Python’s scikit-learn library to analyze student data including GPA, attendance, extracurricular activities, and career goals. Our system achieved an accuracy rate of over 80% in predicting student success and effectively matched career goals with relevant courses. Unlike other recommendation systems that rely solely on historical enrollment data, our solution incorporates individual student characteristics and achievement inputs for tailored recommendations. We recommend expanding the system with additional data sources and implementing a real-time feedback loop to continually improve recommendation quality as students progress through their academic careers.

## 3. Table of Contents

1. Title Page
2. Executive Summary
3. Table of Contents
4. List of Illustrations
5. Introduction
6. Methodology
7. Results/Data/Analysis
8. Conclusions
9. Recommendations
10. Bibliography (Technical References)
11. Appendices
    * Appendix 1: Stakeholder Register
    * Appendix 2: System Specification (Requirements Model)
    * Appendix 3: System Design
    * Appendix 4: Project Plan (Gantt Chart)
    * Appendix 5: Test Plan and Test Cases
    * Appendix 6: UI/UX Design
    * Appendix 7: Deployment Strategy
    * Appendix 8: High-level MLOps Description
12. List of Acronyms
13. Glossary of Terms

## 4. List of Illustrations

**Figures**: - Figure 1: Boxplot of GPA Distribution - Figure 2: Boxplot of Attendance Distribution - Figure 3: Correlation Heatmap of Numeric Variables - Figure 4: Distribution of Career Goals - Figure 5: Confusion Matrix of Model Predictions - Figure 6: Use Case Diagram for Course Recommendation System - Figure 7: AI Capability Architecture Diagram - Figure 8: Full Stack Architecture Diagram - Figure 9: Class Diagram - Figure 10: Component Diagram - Figure 11: Success Prediction Sequence Diagram - Figure 12: Course Recommendation Sequence Diagram - Figure 13: Project Gantt Chart - Figure 14: Student Profile Input Screen Mockup - Figure 15: Course Recommendation Results Screen Mockup - Figure 16: MLOps Pipeline Diagram

**Tables**: - Table 1: Classification Report for Success Prediction Model - Table 2: Stakeholder Register - Table 3: Use Case List - Table 4: Functional Requirements - Table 5: Non-functional Requirements - Table 6: Technology Stack - Table 7: Test Cases

## 5. Introduction

### Background

The higher education landscape presents students with increasingly complex course selection decisions that significantly impact their academic success and career trajectories. Traditional course selection methods often rely on generic program requirements and limited advisor input, failing to account for individual student characteristics, learning patterns, and career aspirations. Previous research in educational data mining has shown strong correlations between appropriate course selection and academic success, with a 15-30% improvement in completion rates and grade performance when students take courses aligned with their strengths and goals.

### Problem Statement

Current course selection processes lack personalization and data-driven insights, resulting in suboptimal academic outcomes. Many students select courses without clear understanding of how these align with their career goals, learning styles, or academic strengths. This leads to increased dropout rates, course withdrawals, and extended graduation timelines, all of which have significant financial and psychological impacts on students and resource implications for educational institutions.

### Significance

An intelligent course recommendation system addresses these challenges by leveraging student data and machine learning to generate personalized suggestions. This approach has several significant benefits: 1. Improved student success rates and timely completion 2. Enhanced alignment between academic choices and career goals 3. More efficient use of institutional resources by reducing course withdrawals and retakes 4. Data-driven support for academic advisors 5. Potential for scalable, consistent advising across large student populations

### Literature Review Summary

Our literature review examined 25 recent papers on educational data mining and course recommendation systems. Key findings include: - Machine learning approaches consistently outperform rule-based systems for academic prediction - Logistic regression and ensemble methods have shown strong performance in educational contexts - Student GPA and attendance patterns are high-value predictors across multiple studies - Career goal alignment is often overlooked in existing recommendation systems - Few existing systems incorporate both predictive analytics and personalized input

### Exploratory Data Analysis Summary

Our initial data exploration revealed several important insights: - Strong positive correlation (r=0.72) between GPA and success rates - Significant correlation (r=0.64) between attendance and success - Extracurricular activities showed moderate correlation with success (r=0.38) - Career goals demonstrated clustering patterns, with STEM-oriented goals showing distinct patterns - Data distributions for GPA and attendance showed acceptable normality after preprocessing

## 6. Methodology

Our methodology followed a structured approach to develop the course recommendation system:

1. **Data Collection and Preparation**
   * Gathered student data including GPA, attendance, extracurricular activity participation, and career goals
   * Collected course information including names, difficulty levels, and prerequisites
   * Cleaned and preprocessed data, addressing missing values and outliers
   * Split data into training (80%) and testing (20%) sets using stratified sampling to maintain class balance
2. **Feature Engineering and Selection**
   * Identified key predictive features: GPA, attendance, extracurricular activities, and career goals
   * Applied StandardScaler to normalize numeric features (GPA, attendance)
   * Implemented OneHotEncoder for categorical features (career goals)
   * Created a preprocessing pipeline to ensure consistent transformation
3. **Model Development**
   * Selected logistic regression as the primary classification algorithm based on literature review
   * Configured the model with appropriate hyperparameters (max\_iter=500)
   * Integrated preprocessing and model components into a scikit-learn Pipeline
   * Trained the model on the prepared training dataset
4. **Evaluation Framework**
   * Established evaluation metrics: accuracy, F1-score, and classification report
   * Created visualization for model performance using confusion matrix
   * Validated model performance using stratified k-fold cross-validation
5. **Recommendation System Development**
   * Developed algorithm to filter courses based on predicted success and career goals
   * Implemented functionality to incorporate student achievements and interests
   * Created fallback mechanisms for cases with no specific matches
6. **System Integration**
   * Combined prediction model and recommendation logic
   * Developed interfaces for student input and recommendation display
   * Implemented model persistence using joblib for deployment
7. **Testing and Validation**
   * Performed unit testing on individual components
   * Conducted integration testing of the complete system
   * Validated recommendations against expert academic advisor input

This methodology aligns with our project plan (see Appendix 4) and follows standard practices in machine learning system development while incorporating domain-specific considerations for educational applications.

## 7. Results/Data/Analysis

### Data Visualization Results

Our exploratory data analysis revealed important patterns in the student dataset:

1. **GPA Distribution**: The boxplot analysis showed a median GPA of 3.2 with an interquartile range of 2.8-3.6, indicating a slight negative skew in the distribution. This suggests most students in our dataset maintain above-average academic performance.
2. **Attendance Patterns**: Attendance data showed a wider distribution with a median of 85% and interquartile range of 75-92%. Notable outliers were present in the lower range, representing students with significant attendance issues.
3. **Correlation Analysis**: The correlation heatmap revealed:
   * Strong positive correlation (0.72) between GPA and success
   * Moderate to strong correlation (0.64) between attendance and success
   * Weaker but significant correlation (0.38) between extracurricular activities and success
   * Minimal correlation between GPA and attendance (0.31), suggesting these are largely independent predictors
4. **Career Goal Distribution**: The career goal distribution showed Computer Science, Data Science, and Business Administration as the most common aspirations, accounting for approximately 65% of the dataset.

### Model Performance

The logistic regression model demonstrated promising performance on the test dataset:

1. **Accuracy**: The model achieved an overall accuracy of 83.6%, significantly outperforming the baseline accuracy of 58.2% (proportion of the majority class).
2. **F1-Score**: The weighted F1-score of 0.82 indicates good balance between precision and recall across both success and failure predictions.
3. **Confusion Matrix Analysis**: The confusion matrix revealed:
   * 147 true positives (correctly predicted successes)
   * 89 true negatives (correctly predicted failures)
   * 26 false positives (incorrectly predicted successes)
   * 21 false negatives (incorrectly predicted failures)
   * Higher precision for success prediction (0.85) than for failure prediction (0.81)
4. **Classification Report**: The detailed classification report showed:

| Class | Precision | Recall | F1-Score | Support |
| --- | --- | --- | --- | --- |
| Failure | 0.81 | 0.77 | 0.79 | 115 |
| Success | 0.85 | 0.88 | 0.86 | 168 |
| Weighted Avg | 0.83 | 0.84 | 0.83 | 283 |

1. **Cross-Validation**: Five-fold cross-validation confirmed the model’s stability with a mean accuracy of 82.7% and standard deviation of 2.1%.

### Recommendation System Effectiveness

To evaluate the recommendation system component, we conducted a qualitative assessment with a panel of academic advisors:

1. **Relevance Rating**: On a scale of 1-5, the average relevance rating for course recommendations was 4.2, indicating high alignment with student profiles.
2. **Career Alignment**: 87% of recommended courses were judged to be well-aligned with specified career goals.
3. **Personalization**: Incorporating student achievements and interests improved perceived recommendation quality by an average of 25% compared to using only the prediction model.
4. **Edge Cases**: The fallback mechanism for students with no specific matches successfully provided appropriate beginner courses in all test cases.

These results demonstrate that the integrated approach of combining machine learning predictions with personalized inputs produces effective course recommendations that are both data-driven and tailored to individual needs.

## 8. Conclusions

Based on the implemented course recommendation system and its evaluation, we conclude that:

* The logistic regression model serves as an effective predictor of student success with an accuracy of 83.6% and F1-score of 0.82, confirming the viability of machine learning for educational recommendation systems.
* Student GPA and attendance are the strongest indicators of academic success, with correlation coefficients of 0.72 and 0.64 respectively, suggesting these should be core features in any academic success prediction model.
* The preprocessing pipeline successfully handles both numerical scaling and categorical encoding, ensuring optimal model performance across diverse student profiles.
* Career goals play a significant role in determining appropriate course recommendations, with 87% of recommended courses being well-aligned with specified career objectives.
* The integrated approach of combining machine learning predictions with personalized inputs (achievements and interests) provides 25% improvement in recommendation quality compared to using only the prediction model.
* The confusion matrix indicates the model performs better at predicting success (0.88 recall) than failure (0.77 recall), suggesting potential for targeted improvements in identifying at-risk students.
* The system’s ability to recommend beginner courses when no specific matches are found provides a reliable fallback mechanism for students with unique or evolving career goals.
* Cross-validation results with a standard deviation of 2.1% demonstrate the model’s stability across different data subsets, confirming the robustness of our approach.

## 9. Recommendations

For future improvements and enhancements to the course recommendation system, we recommend:

1. **Multi-platform Development**: Extend the current implementation to a web-based interface and mobile applications to increase accessibility. This would allow students to access recommendations anywhere and integrate with existing university systems.
2. **Additional Data Sources**: Incorporate more diverse student data such as previous course history, learning styles, and peer performance to enhance prediction accuracy. Particularly, integrating historical grade distributions for specific courses could refine success predictions.
3. **Algorithm Diversification**: Test and implement ensemble methods or neural networks to potentially improve the prediction accuracy beyond the current 83.6%. Random Forest and Gradient Boosting algorithms show particular promise for educational data.
4. **Real-time Feedback System**: Implement a feedback loop where student performance in recommended courses updates the model to improve future recommendations, creating a continuously learning system.
5. **Expanded Course Database**: Integrate with online learning platforms to include a wider range of courses beyond the current dataset, potentially including MOOCs and industry certifications.
6. **Natural Language Processing**: Implement NLP techniques to better match student interests and achievements with course descriptions, enabling more nuanced content-based filtering.
7. **Explainable AI Components**: Add features that explain why specific courses are being recommended to increase student trust and engagement with the system recommendations.
8. **Internationalization**: Adapt the system to account for different educational systems and career paths across countries, making it applicable to global educational contexts.
9. **Longitudinal Tracking**: Develop capabilities to track student progress over multiple semesters to refine long-term course planning and create multi-semester recommendation paths.
10. **Privacy Enhancements**: Strengthen data anonymization and security protocols to protect sensitive student information while maintaining prediction accuracy.

## 10. Bibliography (Technical References)

1. Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749.
2. Baker, R. S. (2019). Challenges for the future of educational data mining: The Baker learning analytics prizes. Journal of Educational Data Mining, 11(1), 1-17.
3. Chatti, M. A., Dyckhoff, A. L., Schroeder, U., & Thüs, H. (2012). A reference model for learning analytics. International Journal of Technology Enhanced Learning, 4(5-6), 318-331.
4. Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O’Reilly Media.
5. Iatrellis, O., Kameas, A., & Fitsilis, P. (2021). Academic advising systems: A systematic literature review of empirical evidence. Education Sciences, 11(1), 1-29.
6. Kassambara, A. (2017). Practical guide to cluster analysis in R: Unsupervised machine learning. STHDA.
7. McKinney, W. (2012). Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. O’Reilly Media.
8. Moreno-Marcos, P. M., Alario-Hoyos, C., Muñoz-Merino, P. J., & Kloos, C. D. (2019). Prediction in MOOCs: A review and future research directions. IEEE Transactions on Learning Technologies, 12(3), 384-401.
9. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
10. Thai-Nghe, N., Drumond, L., Krohn-Grimberghe, A., & Schmidt-Thieme, L. (2010). Recommender system for predicting student performance. Procedia Computer Science, 1(2), 2811-2819.

## 11. Appendices

### Appendix 1: Stakeholder Register

**Table 2: Stakeholder Register**

| Stakeholder | Role | Interests | Influence | Engagement Strategy |
| --- | --- | --- | --- | --- |
| Students | End Users | Course success, career preparation, graduation timeline | High | Regular surveys, user testing sessions, feedback forms |
| Academic Advisors | System Users | Student guidance, success rates, efficient advising | High | Training workshops, system demonstrations, involvement in testing |
| Faculty | Content Providers | Course enrollment, student preparation, teaching effectiveness | Medium | Regular updates, content validation requests, performance feedback |
| Administration | Resource Allocators | Resource utilization, student outcomes, retention rates | Medium | Executive summaries, ROI reports, system performance analytics |
| IT Department | System Maintainers | System reliability, maintenance, integration | Medium | Technical documentation, maintenance schedules, upgrade planning |
| Career Services | Feedback Providers | Career alignment, industry needs, placement rates | Low | Periodic consultations, career trend reports, outcome sharing |
| Alumni | Indirect Beneficiaries | Program improvement, professional network | Low | Success stories, occasional feedback requests |
| Industry Partners | External Stakeholders | Graduate quality, skill alignment | Low | Annual briefings, skill-need surveys |

### Appendix 2: System Specification (Requirements Model)

#### i. Use Case List

**Table 3: Use Case List**

| Use Case Name | Actors | Description |
| --- | --- | --- |
| Student Profile Creation | Students, Academic Advisors | Create or update student profiles with GPA, attendance, extracurricular activities, and career goals. Includes data validation and profile storage. |
| Success Prediction | Academic Advisors, System | Process student data through the machine learning model to predict likelihood of success in courses. Includes feature preprocessing and model application. |
| Course Recommendation | Students, Academic Advisors | Generate personalized course recommendations based on student profile, prediction results, and additional inputs. Includes filtering by career goals and difficulty matching. |
| Model Training | System Administrator | Update the prediction model with new student data to improve accuracy and account for changing patterns. Includes cross-validation and performance evaluation. |
| Student Achievement Input | Students | Students can input achievements and interests to refine recommendations beyond the base prediction. Includes keyword matching and preference weighting. |
| Course Database Management | Faculty, System Administrator | Manage the course database with updated information about prerequisites, difficulty, and content. Includes version control and consistency checks. |

#### ii. Use Case Diagram

[Figure 6: Use Case Diagram for Course Recommendation System]

#### iii. Use Case Detailed Descriptions (AI-related)

**Use Case: Success Prediction**

* **Primary Actor**: Academic Advisor
* **Stakeholders and Interests**:
  + Academic Advisor: Needs accurate predictions to guide students
  + Students: Benefit from appropriate course recommendations
  + Administration: Interested in improving success rates
* **Preconditions**: Student profile with required data exists in the system
* **Success Guarantee**: System generates accurate prediction of student success
* **Main Success Scenario**:
  1. Advisor selects student profile for analysis
  2. System extracts relevant features (GPA, attendance, etc.)
  3. System preprocesses data (scaling numeric values, encoding categories)
  4. System applies trained model to predict success likelihood
  5. System displays prediction result with confidence score
  6. System stores prediction for use in course recommendations
* **Extensions**:
  + Data missing: System prompts for complete information
  + Low confidence prediction: System flags for additional review
* **Technology & Data Variations**: Model accepts both numeric and categorical inputs
* **Frequency**: Performed at beginning of each academic term or upon student request

**Use Case: Course Recommendation**

* **Primary Actor**: Student
* **Stakeholders and Interests**:
  + Student: Receives personalized course recommendations
  + Academic Advisor: Monitors recommendations for appropriateness
  + Faculty: Interested in proper student preparation for courses
* **Preconditions**: Student success prediction has been completed
* **Success Guarantee**: Student receives relevant course recommendations matching career goals
* **Main Success Scenario**:
  1. Student requests course recommendations
  2. System retrieves student career goal and success prediction
  3. Student inputs achievements and interests (optional)
  4. System filters course database using career goal and success prediction
  5. System refines recommendations based on achievements and interests
  6. System displays recommended courses with difficulty levels
* **Extensions**:
  + No matching courses: System recommends beginner courses
  + Multiple career interests: System provides recommendations across areas
* **Technology & Data Variations**: Can incorporate text matching for interests
* **Frequency**: Typically performed prior to course registration periods

#### iv. List of Functional Requirements

**Table 4: Functional Requirements**

| ID | Requirement | Associated Use Case |
| --- | --- | --- |
| FR-01 | The system shall accept and store student profile data including GPA, attendance, extracurricular activities, and career goals | Student Profile Creation |
| FR-02 | The system shall preprocess student data using standardization for numeric features and encoding for categorical features | Success Prediction |
| FR-03 | The system shall predict student success using a trained machine learning model | Success Prediction |
| FR-04 | The system shall recommend courses based on career goals, achievements, and interests | Course Recommendation |
| FR-05 | The system shall filter course recommendations based on predicted success likelihood | Course Recommendation |
| FR-06 | The system shall allow for model training and updates with new student data | Model Training |
| FR-07 | The system shall provide visualization of student data and model performance | Success Prediction |
| FR-08 | The system shall export and save trained models for deployment | Model Training |
| FR-09 | The system shall handle cases where no specific course recommendations are found | Course Recommendation |
| FR-10 | The system shall allow students to input additional information for recommendation refinement | Student Achievement Input |
| FR-11 | The system shall display course recommendations with relevant metadata (name, difficulty) | Course Recommendation |
| FR-12 | The system shall maintain a database of courses with attributes including prerequisites and difficulty | Course Database Management |

#### v. List of Non-functional Requirements

**Table 5: Non-functional Requirements**

| ID | Requirement | Metric |
| --- | --- | --- |
| NFR-01 | Accuracy | The prediction model shall achieve a minimum of 80% accuracy on test data |
| NFR-02 | Performance | The system shall generate recommendations within 3 seconds of request |
| NFR-03 | Scalability | The system shall support a minimum of 10,000 student profiles |
| NFR-04 | Usability | The course recommendation interface shall require no more than 3 steps to generate recommendations |
| NFR-05 | Reliability | The system shall have 99.5% uptime during registration periods |
| NFR-06 | Security | All student data shall be encrypted with AES-256 and anonymized for model training |
| NFR-07 | Maintainability | The system shall support model updates without service interruption |
| NFR-08 | Compatibility | The system shall function with standard CSV data exports from university systems |
| NFR-09 | Regulatory Compliance | The system shall comply with FERPA and relevant educational data privacy regulations |
| NFR-10 | Documentation | All AI components shall be fully documented with implementation details |
| NFR-11 | Transparency | The system shall provide confidence scores for all predictions |
| NFR-12 | Robustness | The system shall gracefully handle missing data with appropriate fallback mechanisms |

### Appendix 3: System Design

#### i. List of Technologies

**Table 6: Technology Stack**

**Hardware:** - Server: Cloud-based virtual machines for model training and deployment (AWS EC2 or Azure VM) - Database Server: Dedicated server with minimum 16GB RAM for storing student and course data - Client Devices: Standard computers/mobile devices for accessing the system interface - Development Workstations: 16GB RAM, modern CPU for model development

**Software:** - Programming Language: Python 3.8+ - Machine Learning Libraries: Scikit-learn 1.0.2, Pandas 1.3.5, NumPy 1.21.5 - Visualization Tools: Matplotlib 3.5.1, Seaborn 0.11.2 - Development Environment: Google Colab (development), PyCharm (code editing), Production Server (deployment) - Database Management System: PostgreSQL 14.0 - Web Framework: Flask 2.0.1 for API services - Front-end: HTML5, CSS3, JavaScript with Vue.js 3.0 - Version Control: Git 2.35.1 - Model Persistence: Joblib 1.1.0 - Containerization: Docker 20.10.12 - CI/CD Pipeline: GitHub Actions

#### ii. AI Capability Description Document with Data Descriptions

**AI Capabilities:** - Supervised Learning: Binary classification of student success using logistic regression - Feature Preprocessing: Numeric scaling (StandardScaler) and categorical encoding (OneHotEncoder) - Recommendation Generation: Rule-based filtering combined with ML predictions and user inputs - Model Evaluation: Performance assessment using accuracy, F1-score, and confusion matrix - Model Persistence: Storage and loading of trained models via joblib - Cross-validation: K-fold validation to ensure model robustness

**Data Descriptions:**

*Student Data:* - student\_id: Unique identifier for each student (integer) - gpa: Grade Point Average on 4.0 scale (float, range: 0.0-4.0) - attendance: Percentage of classes attended (float, range: 0-100) - extracurricular: Participation in extracurricular activities (binary: 0=No, 1=Yes) - career\_goal: Career aspiration of the student (categorical string) - success: Target variable indicating previous academic success (binary: 0=No, 1=Yes)

*Course Data:* - course\_id: Unique identifier for each course (integer) - course\_name: Name of the course (string) - difficulty: Difficulty level of the course (categorical: ‘Beginner’, ‘Intermediate’, ‘Advanced’) - prerequisites: Required prerequisite courses, if any (string, comma-separated list) - keywords: Related topics and skills (string, comma-separated list)

*Model Output:* - success\_prediction: Binary prediction of student success (0=Failure, 1=Success) - confidence\_score: Probability score associated with prediction (float, range: 0.0-1.0)

*Recommendation Output:* - course\_list: List of recommended courses with metadata (JSON array) - match\_score: Relevance score for each recommendation (float, range: 0.0-1.0)

#### iii. Architecture Design (AI Capability)

Figure 7: AI Capability Architecture Diagram

A diagram of a process flow

Description automatically generated

#### iv. Architecture Design (Full Stack)

[Figure 8: Full Stack Architecture Diagram]

A diagram of a software structure

Description automatically generated with medium confidence

#### v. Class Diagram(s)

Figure 9: Class Diagram

A diagram of a computer

Description automatically generated with medium confidence

#### vi. List of Components and Component Diagram

**Components:** 1. Data Collection Module: Handles student data input and storage 2. Data Preprocessing Pipeline: Transforms raw data for model consumption 3. Machine Learning Model: Implements logistic regression for success prediction 4. Course Recommendation Engine: Filters and ranks courses based on student profile 5. Visualization Component: Generates data visualizations and model performance metrics 6. Model Persistence Service: Saves and loads trained models 7. User Interface: Provides front-end for student and administrator interaction 8. Database Interface: Manages data storage and retrieval operations 9. API Service: Exposes system functionality for integration

Figure 10: Component Diagram

#### A diagram of a machine learning process Description automatically generated vii. Interaction Diagrams for AI -related Use Cases

**Success Prediction Interaction:**

Figure 11: Success Prediction Sequence Diagram

A diagram of a process flow

Description automatically generated

**Course Recommendation Interaction:**

Figure 12: Course Recommendation Sequence Diagram

A screenshot of a computer

Description automatically generated

### Appendix 4: Project Plan (Gantt Chart)

[Figure 13: Project Gantt Chart]

### Appendix 5: Test Plan and Test Cases

#### Test Plan

**Unit Tests:** 1. Data Preprocessing Component Tests - Test scaling of numeric features - Test encoding of categorical features - Test handling of missing values

1. Model Training Tests
   * Test model initialization
   * Test model fitting process
   * Test prediction functionality
2. Recommendation Engine Tests
   * Test course filtering by career goal
   * Test integration of achievement and interest inputs
   * Test fallback to beginner courses

**Integration Tests:** 1. End-to-End Recommendation Process - Test complete flow from student data to course recommendations - Test model persistence and loading - Test integration of all components

1. User Input Integration
   * Test handling of user-provided achievements
   * Test handling of user-provided interests

**Table 7: Test Cases**

| Test ID | Description | Input | Expected Output | Pass/Fail Criteria |
| --- | --- | --- | --- | --- |
| UT-01 | Numeric Scaling | Student GPA: 3.5 | Scaled value within model input range | Output value between -3 and 3 |
| UT-02 | Career Goal Encoding | Career Goal: “Data Science” | One-hot encoded vector | Correct binary vector with 1 in appropriate position |
| UT-03 | Success Prediction | Complete student profile | Binary success prediction | Prediction matches expected value for test cases |
| UT-04 | Missing Data Handling | Student profile with missing attendance | Preprocessed data with imputed values | No errors, reasonable imputation |
| UT-05 | Course Filtering | Career goal: “Computer Science” | Filtered course list | Only Computer Science related courses returned |
| UT-06 | Model Training | Training dataset | Trained model object | Model accuracy above 75% on validation set |
| IT-01 | Complete Recommendation | Student ID: 1, Achievements: “Python” | List of relevant courses | Non-empty list matching career goal and achievement |
| IT-02 | Fallback Recommendation | Student ID with no matches | List of beginner courses | Non-empty list of entry-level courses |
| IT-03 | Model Persistence | Trained model object | Saved model file | Model can be reloaded with same performance |
| IT-04 | UI Display Test | Recommendation data | Formatted recommendation display | All course details correctly rendered |

### Appendix 6: UI/UX Design

Figure 14: Student Profile Input Screen Mockup

A screen shot of a computer

Description automatically generated

Figure 15: Course Recommendation Results Screen MockupA screenshot of a course

Description automatically generated

### Appendix 7: Deployment Strategy

1. **Development Environment**:
   * Initial development in Google Colab
   * Version control through GitHub repository
   * Regular code reviews and unit testing
   * Development branch for ongoing enhancements
2. **Testing Environment**:
   * Dedicated test server with replica database
   * Integration testing with sample dataset
   * Performance testing and optimization
   * User acceptance testing with academic advisors
   * Staging environment that mirrors production
3. **Production Deployment**:
   * Model export using Joblib
   * Server preparation with required dependencies
   * Database initialization with production data
   * Containerization with Docker for consistency
   * Phased rollout to user groups:
     + Phase 1: Selected academic departments (20%)
     + Phase 2: Expanded to additional departments (50%)
     + Phase 3: Full university deployment (100%)
4. **Maintenance Plan**:
   * Scheduled model retraining with new student data (once per semester)
   * Performance monitoring and logging
   * Regular backups of model and data
   * Versioning of deployed models
   * Automated alerts for system issues
   * Patch deployment process for critical issues
5. **Scaling Strategy**:
   * Horizontal scaling with load balancing for increased user load
   * Database sharding for growing data volume
   * Caching layer for frequently requested recommendations

### Appendix 8: High-level MLOps Description

The MLOps pipeline for the course recommendation system consists of:

1. **Data Management**:
   * Data collection from student records
   * Data validation and cleaning
   * Version control of datasets using DVC (Data Version Control)
   * Feature engineering pipeline
   * Data drift detection
2. **Model Development**:
   * Experiment tracking with MLflow
   * Hyperparameter optimization with Grid Search
   * Cross-validation for model selection
   * Model performance comparison
   * Model documentation
3. **Model Deployment**:
   * Model packaging with Joblib
   * Environment configuration with Docker
   * API development with Flask
   * Containerization for consistent deployment
   * Blue-green deployment for zero downtime updates
4. **Monitoring and Maintenance**:
   * Performance metrics tracking
   * Drift detection for data and predictions
   * Automated retraining triggers
   * A/B testing for model improvements
   * Alerting system for model degradation

[Figure 16: MLOps Pipeline Diagram]

## 12. List of Acronyms

* AI: Artificial Intelligence
* API: Application Programming Interface
* AWS: Amazon Web Services
* CI/CD: Continuous Integration/Continuous Deployment
* CSV: Comma-Separated Values
* DVC: Data Version Control
* EC2: Elastic Compute Cloud
* FERPA: Family Educational Rights and Privacy Act
* GPA: Grade Point Average
* HTML: Hypertext Markup Language