
CAREER
RECOMMNDATION SYSTEM
Unlock Your Future. Find Your Path

1. Introduction

- Python-based Career Advisor System for students
- Analyzes academic performance and suggests career paths
- Uses data visualization for better insights
- Simple, interactive desktop application
- Combines education analysis with career guidance

2. Problem Statement

- Students lack clear connection between academics and careers
- Manual career counseling is inaccessible/expensive
- No instant tools for subject-strength-based career mapping
- Need for visual representation of academic performance
- Gap in automated career guidance systems

3. Functional Requirements

- Input Management: Accept marks for 5 predefined subjects
- Data Analysis: Calculate strongest subject using maximum marks
- Career Mapping: Suggest careers based on subject excellence
- Performance Classification: Categorize marks into performance levels
- Visualization: Generate bar chart and pie chart
- Results Display: Show careers with salary estimates

4. Non-functional Requirements

- Performance: Response time < 2 seconds for all operations
- Usability: Simple interface with clear prompts
- Reliability: Error-free execution with input validation
- Maintainability: Clean, documented code structure
- Portability: Cross-platform compatibility (Windows/Mac/Linux)

5. System Architecture

- Single-tier architecture: Standalone desktop application
- Modular design: Separate components for input, processing, output
- Linear workflow: Sequential execution from input to visualization
- Libraries: NumPy (computation), Matplotlib (visualization)

6. Design Diagrams

○ Use Case Diagram

- Actor: Student/User
- Use Cases:
  - Enter subject marks
  - View career recommendations
  - See performance analysis
  - View visual charts

○ **Workflow Diagram**

1. Input marks → 2. Process data → 3. Analyze performance → 4. Generate recommendations → 5. Display charts

Sequence Diagram

- User → System: Input marks
- System → NumPy: Calculate strongest subject
- System → Career Map: Get career suggestions
- System → Matplotlib: Generate charts
- System → User: Display results

○ **Class/Component Diagram**

- Main Program (orchestrator)
- Data Processor (NumPy operations)
- Career Mapper (Dictionary-based)
- Visualizer (Matplotlib charts)

○ ER Diagram : Not applicable (no database storage)

7. Design Decisions & Rationale

- Predefined subjects: Ensures consistent analysis across users
- NumPy for calculations: Efficient array operations for performance
- Matplotlib for visualization: Industry-standard plotting library
- In-memory data storage: Simple implementation, no persistence needed

- Hardcoded career mapping: Reliable, predictable recommendations

8. Implementation Details

- Programming Language: Python 3.x

- Key Libraries:

  - NumPy: `argmax()` for strongest subject

  - Matplotlib: `bar()` and `pie()` for charts

  - Random: Salary generation

- Data Structures: Lists, Dictionaries, NumPy arrays

- Algorithms: Maximum value finding, classification logic

9. Screenshots / Results

- Console output showing strongest subject and careers

- Bar chart with subject marks (strongest highlighted in gold)

- Pie chart showing performance distribution

- Career list with randomized salary ranges

- Performance classification (Excellent/Good/Average/Needs Improvement)

10. Testing Approach

- Unit Testing: Verify strongest subject calculation

- Integration Testing: End-to-end workflow validation

- Input Testing: Boundary values (0, 100), invalid inputs

- Visual Testing: Chart generation and display
- User Acceptance: Simple interface evaluation

11. Challenges Faced

- Matplotlib figure sizing and layout management
- Handling empty data segments in pie chart
- Ensuring cross-platform compatibility
- Balancing simplicity with functionality
- Input validation without complex error handling

12. Learnings & Key Takeaways

- Effective use of NumPy for data analysis
- Matplotlib chart customization techniques
- Importance of user-friendly input/output design
- Value of visual data representation in educational tools
- Trade-offs between simplicity and feature richness

13. Future Enhancement

- Database integration for persistent user profiles
- More subjects with customizable options
- Advanced ML algorithms for better predictions
- Web application version for broader accessibility

- User authentication and progress tracking
- Export functionality for reports and charts
- Integration with career APIs for real-time job market data
- Mobile app version for on-the-go access