



GradeUP – Final Project Report



Table of Contents

1. Introduction.....	3
2. Team Contributions	3
Backend:	3
Designers:	3
Scrum Master:.....	3
3. Features Implemented.....	3
5. Development Process.....	5
6. Screenshots.....	7
7. Results and Challenges	7
8. Conclusion.....	9

1. Introduction

GradeUP is a Flask-based web application built to help students evaluate universities and academic programs through surveys and data collection. It serves as a feedback and analysis platform for students, administrators, and researchers to make informed decisions and improve educational offerings.

2. Team Contributions

Each team member contributed to different modules of the platform:

- Frontend: UI/UX design and Jinja templating
- Backend: Flask architecture, SQLAlchemy models, survey logic
- Admin Panel: Management tools for users, programs, and universities
- Survey System: Question types, response handling, progress tracking
- Documentation and Testing: User stories, ER diagram, and bug fixes

Backend:

[KRRusev21](#) | [TPIvanov21](#)

Designers:

[PSDineva21](#) | [GYFilipov21](#)

Scrum Master:

[SGMikov21](#)

3. Features Implemented

- Secure user authentication with role-based access
- University and program listings with search and filters
- Custom survey creation and results analysis
- Admin dashboard for managing users, content, and data
- Data export functionality
- Responsive, modern UI with cosmic design theme

3.1 AI Module Integration

An advanced AI recommendation system was developed, integrating two key modules into the GradeUP platform:

- AdvancedPredictionSystem: Generates program predictions using confidence scoring.
- RecommendationEngine: Provides personalized university and program recommendations based on survey responses, user interests, and academic profiles.

The AI modules are fully integrated into the Flask backend, directly interacting with SQLAlchemy models. They provide predictions through REST API endpoints:

- `/api/predict` → `predict_with_confidence()`
- `/api/recommend` → `recommend_programs()`

All predictions and recommendation results are stored in the database for analytics and future personalization. The AI engine enhances decision-making for students by delivering data-driven and personalized academic recommendations.

4. Code Documentation

The codebase (showed below) follows Python documentation standards with docstrings and inline comments.

```
"""
Data Collection Models

Database models for survey data collection and storage functionality.
"""

import ...

class SurveyData(db.Model): 20 usages ▲ TPivanov21+1
    """Extended survey data storage for analytics and reporting."""
    __tablename__ = 'survey_data'

    id = db.Column(db.Integer, primary_key=True)
"""

Data Export Module

Handles exporting survey data in various formats (CSV, JSON, Excel).
"""

import ...

class DataExporter: 1 usage ▲ TPivanov21+1
    """Handles data export operations in various formats."""

    @staticmethod 1 usage ▲ TPivanov21
    def export_to_csv(survey_data: List, filename: str = None) -> str
        """Export survey data to CSV format."""
        if not filename:
            filename = f"survey_data_{datetime.now().strftime('%Y%m%")}
```

```
@bp.route('/status') ▲ KRRusev21
def simple_status():
    """Simple status check that always works"""
    return jsonify({
        "status": "app_running",
        "message": "Flask app is running successfully",
        "timestamp": datetime.now().isoformat()
    })

@bp.route('/health') ▲ KRRusev21
def health_check():
    """Health check endpoint for debugging"""
    try:
        # Check if DATABASE_URL is set
        db_url = os.environ.get('DATABASE_URL')
    except:
        raise ValueError('File size must be less than 2MB')

    if len(form_picture.read()) > 2 * 1024 * 1024:
        form_picture.seek(0)
        raise ValueError('File size must be less than 2MB')

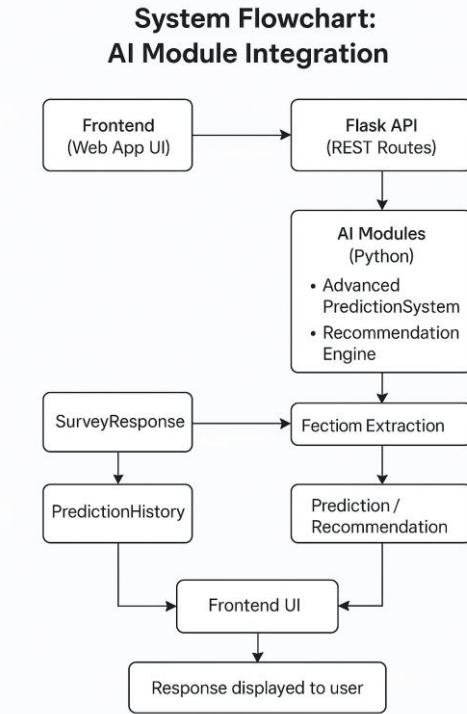
    form_picture.seek(0)

    random_hex = os.urandom(8).hex()
    _, f_ext = os.path.splitext(form_picture.filename)
    picture_fn = random_hex + f_ext
```

4.1 System Architecture Diagram

The diagram below illustrates the full integration between the frontend, Flask backend, AI modules, and database layers:

- The frontend collects user survey data.
- Flask backend routes data to the AI modules.
- The AI modules process the data and return predictions.
- Results are stored in the database and presented to the user.



5. Development Process

We followed an agile development cycle with user stories, sprints, and regular team check-ins. Progress was tracked using GitHub Projects boards. Sprint goals were defined weekly, and each iteration focused on building, testing, and refining a specific feature set.

Link to the project's User Stories: [UserStories_GradeUp.docx](#)

Screenshots of the GitHub Projects board would be included here:

The image displays two separate GitHub Projects boards side-by-side. Each board consists of four columns: Todo, In Progress, Testing, and Done.

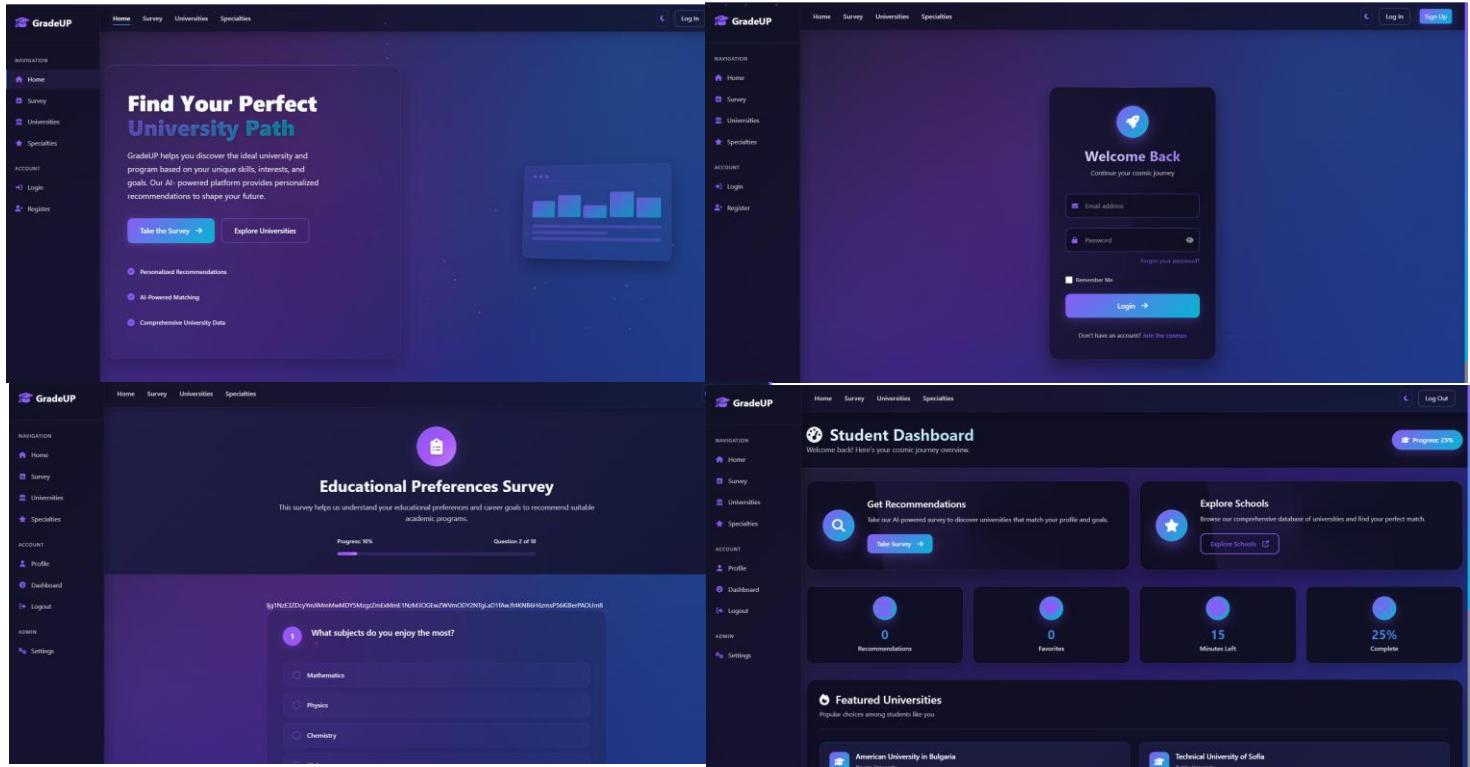
- Initial Project Setup:**
 - Todo:** 0 / 8 Estimate: 0. Items include: This item hasn't been started, Create README.md, Setup Intake Project, Create assignment group #10, and Add assignment group #11.
 - In Progress:** 0 / 8 Estimate: 0. Items include: This is activity being worked on, Implement assignment group #10, and Implement assignment group #11.
 - Testing:** 0 / 0 Estimate: 0.
 - Done:** 0 / 0 Estimate: 0.
- User Profiles & Authentication:**
 - Todo:** 0 / 5 Estimate: 0. Items include: This item hasn't been started, Create student profile #1, and Create student profile #2.
 - In Progress:** 0 / 5 Estimate: 0. Items include: This is activity being worked on, Implement assignment group #10, and Implement assignment group #11.
 - Code Review:** 0 / 5 Estimate: 0. Items include: Review assignment group #10, Review assignment group #11, and Review assignment group #12.
 - Done:** 0 / 5 Estimate: 0. Items include: This has been completed, Create student profile #1, Create student profile #2, Set up user roles, and Set up user roles.
- Final UI And Auth Enhancements:**
 - Todo:** 0 / 5 Estimate: 0. Items include: This item hasn't been started, Implement global cosmic design theme and animations, and Update homepage with hero section and cosmic layout.
 - In Progress:** 0 / 5 Estimate: 0. Items include: This is activity being worked on, Implement assignment group #10, and Implement assignment group #11.
 - Code Review:** 0 / 5 Estimate: 0. Items include: Review assignment group #10, Review assignment group #11, and Review assignment group #12.
 - Done:** 0 / 5 Estimate: 0. Items include: This has been completed, Implement global cosmic design theme and animations, Update homepage with hero section and cosmic layout, Redesign base template with cosmic navigation, and Finalize configuration for auth forms.
- Data Collection Module:**
 - Todo:** 0 / 5 Estimate: 0. Items include: This item hasn't been started, Implement data collection module, and Set up data collection module.
 - In Progress:** 0 / 5 Estimate: 0. Items include: This is activity being worked on, Implement data storage for selected data sources, and Verify setbacks to include data collection module.
 - Code Review:** 0 / 5 Estimate: 0. Items include: Review assignment group #10, Review assignment group #11, and Review assignment group #12.
 - Done:** 0 / 5 Estimate: 0. Items include: This has been completed, Implement data storage for selected data sources, Implement data storage for selected data sources, Verify setbacks to include data collection module, Host the web app on GitHub Codespaces, Add fallback to SQLite if DATABASE_URL is not set, Fix PostgreSQL connection config for Azure, and Add status endpoint for basic uptime check.

The image shows a single GitHub Projects board with four columns: Todo, In Progress, testing, and Done.

- Deployment & Diagnostics:**
 - Todo:** 0 / 5 Estimate: 0. Items include: This item hasn't been started.
 - In Progress:** 0 / 5 Estimate: 0. Items include: This is actively being worked on.
 - testing:** 0 / 0 Estimate: 0.
 - Done:** 0 / 6 Estimate: 0. Items include: Host the web app on GitHub Codespaces, Add fallback to SQLite if DATABASE_URL is not set, Fix PostgreSQL connection config for Azure, and Add status endpoint for basic uptime check.
- Documentation:**
 - Todo:** 0 / 5 Estimate: 0. Items include: This item hasn't been started.
 - In Progress:** 0 / 5 Estimate: 0. Items include: This is actively being worked on.
 - Done:** 8 / 8 Estimate: 0. Items include: Add Entity Relationship Diagram (ERD) to the documentation, Add ERD to the README.md, and Add user stories for GradeUP functionality.

6. Screenshots

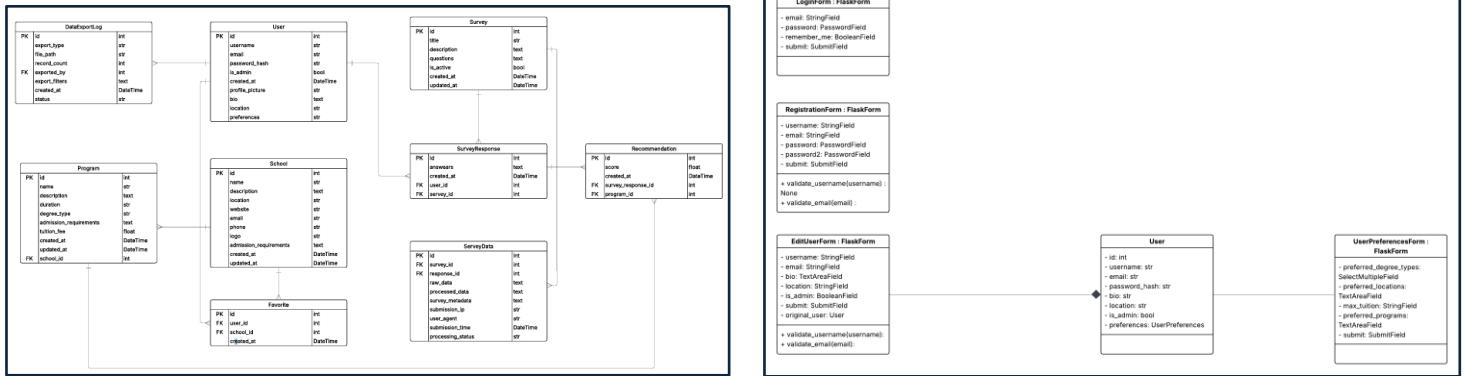
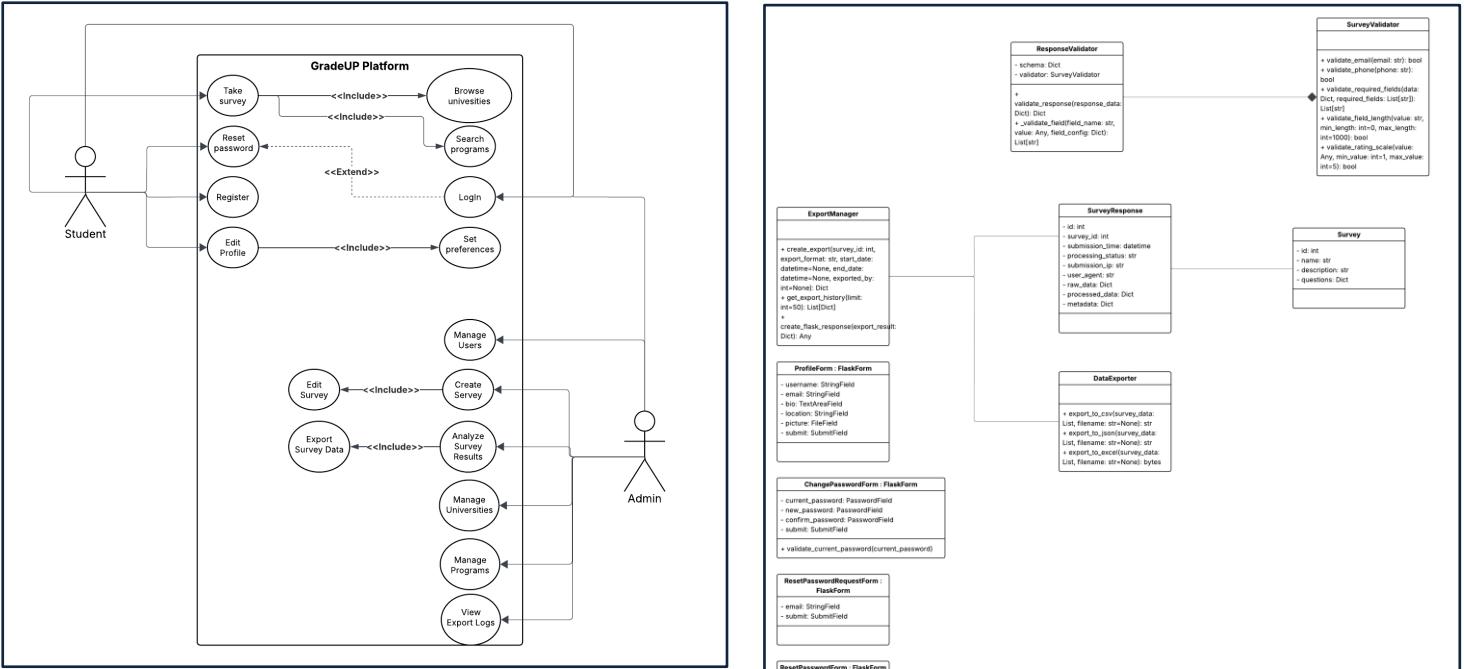
Screenshots of the UI and core features would be included here:



7. Results and Challenges

GradeUP successfully implements a complete evaluation workflow. Key challenges included managing dynamic survey structures and fixing infinite loading issues in the details modal. These were addressed through better data handling and optimized frontend rendering.

Screenshots of the UML Diagrams (Use Case, Class diagram and ERD) would be included here:



8. Conclusion

GradeUP is a functional and scalable tool for academic feedback and evaluation. It combines modern web technology with user-centric design to create value for students, educators, and researchers.

