EasyA SRS

Team: Git Good

- **Karma Woeser**
- **Jason Webster**
- **Raj Gill**
- **Giovanni Mendoza Celestino**

Problem Statement

Students at the University of Oregon don't have a platform to compare the grading distributions of instructors and courses. Sites like Rate My Professor help with the process of course selection but they depend on students writing reviews, which most students won't do. EasyA will allow students to view grading trends in courses and help them make an informed decision for their classes.

Task Requirements:

- Provide a searchable system to provide and compare grade distributions.
- Allow users to filter results based on department, instructor, course and class level.
- Allow users to view A,B,C,D, and F grades.
- Visualize data to help users compare courses.
- Provide an admin page that can update and maintain data.
- Implement a web scraper to collect faculty and grade distribution data.

Technologies Used:

- **Backend: Flask** for quick API development while also providing the tools needed for handling requests, managing routes, and connecting to a database.
- **Frontend: HTML, AJAX, JavaScript** (Chart.js for data visualization) for a simple frontend that will dynamically load without refreshing the page.
- **Database: MongoDB** is simple and good at handling structured data like grade distributions.
- **Web Scraping: Python's BeautifulSoup** to collect information from the University of Oregon's archives.
- **Deployment: Docker** for containerization and deployment which ensures consistency across environments and easy setup.

Description of Users

Primary Users:

- **Students**
    - Expectations: Quick and easy search and filtering with clear data visualization.
    - Expected Prior Knowledge: Familiar with search interfaces and understanding of the University of Oregon course naming structure.
    - Tech Usage Patterns: Most likely will be used during registration periods.

- **Administrators**
    - Expectations: An easy way to upload, merge, and delete datasets.
    - Expected Prior Knowledge: Familiar handling data files, and using admin panels.
    - Tech Usage Patterns: Updates data periodically, primarily before each term.

Scenarios or Use Cases

Use Case 1: Searching for a course's grade distribution

Scenario:

Alice, a CS student, wants to take CS 472 but has heard it's a difficult class, so she wants to look at the class history to see if this is really the case.

- Alice will select the CS department and 472.
- The system grabs grade distribution data of CS 472 and displays it on screen.
- Alice can apply filters like sorting by highest A, or C, or F.
- The graphs will dynamically update as other filters are applied, and she can decide wherever or not the class will be difficult or not.

Use Case 2: Comparing instructors grade distribution of a course

Scenario:

Bob is a CS student who wants to take CS 422 and has heard the current professor can be hard. So he uses EasyA to compare the current professor CS 422 for the current term to other professors that've taught CS 422.

- Bob selects CS 422 and all the instructors that've taught CS 422 grade distribution data will be displayed.
- The system generates a bar chart of all average grades of instructors of the course
- He can apply filters to see which one provided the most F's to avoid them, he can also choose the current professor to see if their reputation for bad grades is true.
- The student uses this information to choose an instructor.

Use Case 3: Admin uploading new data

Scenario:

Karma the admin wants to update the grading dataset for all courses. To make sure he can compare more grade distribution data.

- Karma can upload a link to the University of Oregon archives that is specified in the github and click the upload data button.
- The system validates the dataset and inserts it into the database.
- Karma is alerted with confirmation of a successful data upload.
- The updated database is shown in user filtering options and he can now look through more courses.

Detailed Description of Requirements

Functional Requirements:

Absolutely Required:

1. Users will be able to search for courses by course code and name.
2. Users will be able to filter results by department, instructor, and class levels.
3. Users will be able to filter between A,B,C,D, and F grade distributions
4. The system will display grade distributions using interactive graphs.
5. Users will be able to compare multiple instructors' grading trends.
6. Admins will be allowed to upload new JSON datasets for MongoDB.
7. Admins will be allowed to web scrape from UO's archive catalog
8. The system will retrieve and process data from MongoDB.
9. The system will validate the data before adding it to MongoDB.
10. The system will allow admins to merge faculty data with grade records.

11. The system will allow admins to delete records.
12. The system will allow for easy data retrieval.

Not Absolutely Required:

13. The system will alert errors for users when data processing fails.
14. The system will provide an error message for invalid searches.
15. The system will allow users to reset filters.
16. The system will update graphs dynamically .

Non-Functional Requirements:

Absolutely Required:

1. The system will ensure that response times to load the graphs aren't longer than 2 seconds.
2. The system will be intuitive and easy to use.
3. The system will be able to handle the upload of multiple datasets.

Not Absolutely Required:

4. The system will be compatible with different web browsers.
5. The system will be scalable in case there is an increase of site usage.
6. The system will be easy to track and have proper error handling

Excluded Features/Future Development

Some of the features we planned for originally were not included due to time constraints or technical complexity but could be implemented in the future:

- **Advanced Filtering**
    - User's can filter by course, instructor, and class level. Future versions could allow filtering by specific grade thresholds. Where it shows courses where only 20% of students got an A or show a course where 90% of the class passed.
    - This will allow for more specific filtering and allot students be make even better decisions on their courses

- **Mobile & Improved UI**

    - The current UI is simple and intuitive but could be improved to be more aesthetically pleasing. The bar graphs could be displayed in a different form that users may enjoy more. There is no mobile version of EasyA.
    - A more aesthetically pleasing site and mobile will reach more users and encourage people to use EasyA more.

- **Deployment**
    - Deploying EasyA will actually allow real users to use the site. We never got to deployment due to time constraints.