Project Background:

The Academy of Motion Picture Arts and Sciences (AMPAS) has decided to open the process of voting for academy awards to the world. Any individual from any country can vote on a movie to be nominated in almost any category. Once the nominated movies are chosen, anyone can vote on who should win the award. The qualifications for a person being able to vote will be for them to view a set number of previous Oscar winning movies.

Product Wants:

AMPAS wants you to build an application that will allow users to search any previously nominated and/or winning Oscar movies by category of award. They want to make it easy for people to view those movies. So, in the search results, they would like to include a link to the movie's website page at some provider of movie viewing choices such as IMDB, Hulu etc. They'd like for people to be able to view what critics and others said about the movie at the time of its release - from sources like movie review sites or IMDB rankings. Most importantly, they want to expose some functionality as an API so that they can sell access to other developers who can build from the product and create new interfaces.

A failed version of this project was attempted several years back and an AMPAS insider has managed to exhume a few snippets of the SRS draft from that attempt. You may use this artifact to provide greater insight into the project.

Project Grading:

It is conceivable that challenges will occur which may affect the breadth/depth of your final product. Please always keep in mind that I am more concerned about your process than I am about the final product.

Minimum Project Deliverables:

A project which includes all minimum project deliverables will receive at least a 75% score on the project.

1. Product
    a. Must provide a REST endpoint that delivers a collection resource in JSON.
    b. Must provide a REST endpoint that delivers a singleton resource in JSON.
    c. Must provide a REST endpoint that allows search of 1 Oscar category and returns results containing the nominees in JSON.
    d. Must provide a minimum level of documentation regarding access, input and output to your API endpoints.
2. Process
    a. Must have a product vision
    b. Must have one persona
    c. Must store source code in a repository such as Github.com or Gitlab.ecs.csus.edu
        i. Instructor must receive invite to/location of repository.
        ii. There must be evidence of ongoing repository activity for each sprint.
    d. Must use Flying Donut to track and adhere to Scrum process.
        i. Instructor must receive invite to/location of project.
    e. Must implement AT LEAST one user story per sprint.
    f. Must incorporate unit testing for a minimum of 2 classes.
3. Presentation
    a. Presentation details will be discussed toward the end of the semester however all team members must participate in the oral project presentation.
        i. Slides
        ii. Discuss project and or process
        iii. Demo a working product

Non-Minimum Project Deliverables:

Each non-minimum item will add points to your projects score. The total number of points for a project implementing all non-minimum deliverable items is above 100%. Therefore, it is not expected that any project will include ALL non-minimum deliverables.

1. Product

   a. The results returned contain data which correlates the Oscar category results with additional data from an outside source such as OMDB, TMDB, TVDB etc. by providing a link to an external site for the movie.
   b. The search feature allows limiting results a to date range.
   c. More than one category can be searched.
   d. More than one endpoint that delivers a collection resource.
   e. More than one endpoint that delivers a singleton resource.
   f. Graphical user interface (GUI) for the product (or portions thereof).
   g. Well-designed HTML page documenting API endpoints and example inputs/outputs.
2. Process
   a. Incorporate test-driven development practice for one Sprint
      i. As evidenced by timestamps on check in for test code vs. check in for implementation code.
   b. Refactor code in a significant manner with a clear and stated goal for refactoring.
      i. As evidenced by an implemented story from the Product Backlog with corresponding work in repository.
   c. Incorporate Contextual Inquiry/Elicitation techniques to create visual persona(s) and develop additional personas.
      i. As evidenced by notes taken from the inquiries and creation of additional visual persona(s) beyond the minimum(part of deliverable 2).
   d. Create mockups for a proposed GUI (even if not implemented).
   e. Incorporate pair programming during one Sprint.
      i. As evidenced by a 1 minute clip of a recorded in-person or Use Together session with link included in final deliverables.
   f. Incorporate one or more design patterns.
      i. Indicate in user story which design pattern for which class(es) and provide comments in code (checked in to repository).
   g. Adopt a coding standard and follow it
      i. As evidence by its inclusion in the repository
   h. Use a database to store and retrieve Oscar data using queries.
      i. Note: It is not expected that you will use this option. The architecture required to implement this is more complex and students may not have taken CSC134. However, this option is meant to provide credit to any team whose members have experience with and would like to utilize a database.
   i. Analyze Code using SonarQube or similar tool.
      i. As evidenced by screen shot of SonarQube analysis
   j. Explain and document obstacles encountered during the project and how those obstacles were handled.

There may be other opportunities, beyond those outlined above, to raise your project above the minimum 75% grade evaluation level. If you have any ideas, feel free to let me know before enacting them so that you don't waste your efforts.