CS/SE/CE 3354 Software Engineering Final

Project Deliverable 2

ProfFinder

Group Members:
Alejandra De Alba Ortiz, Benjamin Garcia, Guderian Raborg, Janet Bui, Kaavya Jampana, Matthew Bierie, Ozair Kamran, Sophia O'Malley, Srivishnu Jegan Babu

Delegation of Tasks [Individual Members]

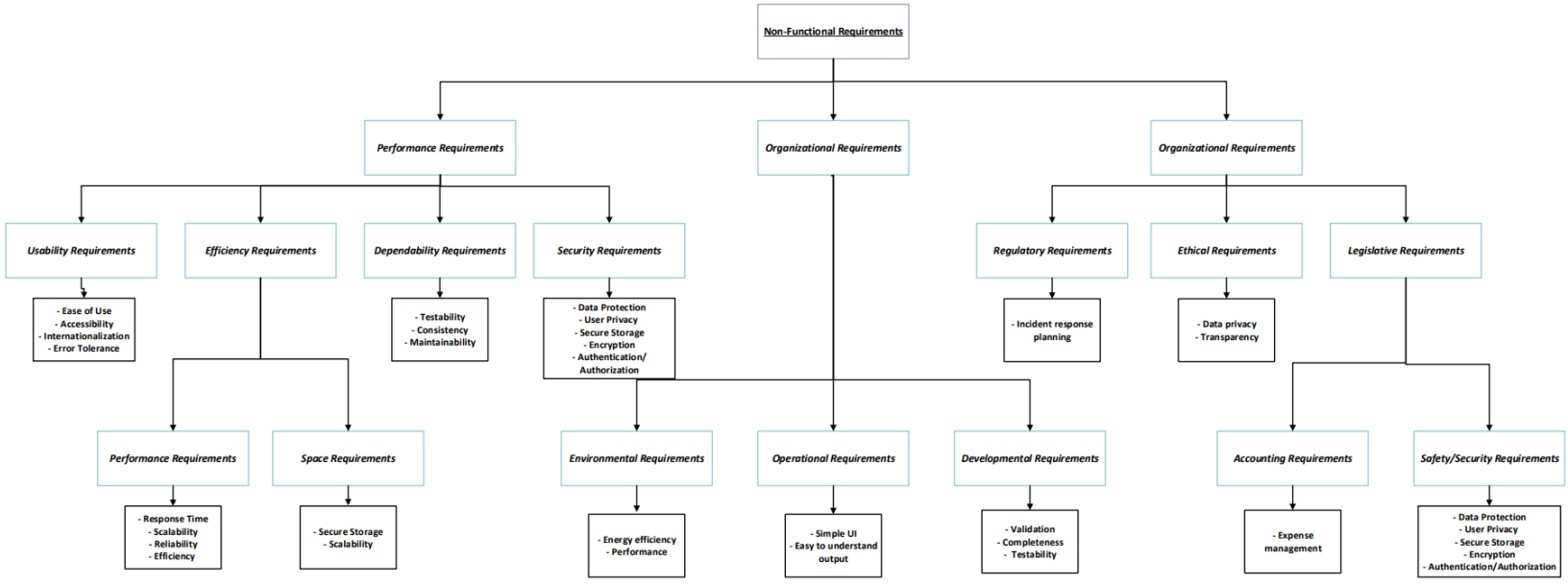
- **Kaavya Jampana** – Non-Functional Requirements, Presentation
- **Srivishnu Jegan Babu** – Use Case Diagram, Build the code for test planning
- **Benjamin Garcia** – Functional Requirements, About ProfFinder Section, Project Scope File, Project Scheduling, document formatting
- **Ozair Kamran** – Architectural Design, Build Code for Test Planning, README.md Update
- **Matthew Bierie** – Software Process Model, Comparison of work
- **Sophia O'Malley** – Requirements and Sequence Diagram, Conclusion
- **Alejandra De Alba Ortiz** – Sequence Diagram, Class Diagram, Project Scope File, Cost of Products and Personnel, Cost of Software
- **Guderian Raborg** – Architectural Design, Estimate Cost Effort and Pricing, Help Build Code for Test Planning
- **Janet Bui** – Use Case Diagram, Class Diagram, GitHub Creation, README.md, Project Scheduling, Estimate Cost Effort and Pricing

Project Deliverable 1 Content

Functional Requirements

1. A user shall be able to search for multiple professors by specifying the university and desired course.
2. A user shall be able to enter a Rate My Professor (RMP) link and receive a summary of the professor.
3. The system shall scrape data from RMP using the specified requirements.
4. Students of any university shall be capable of using this system.
5. The system will generate a summary consisting of the professors pros and cons.
6. If multiple professors are searched, the system shall only provide the information of the top ranked ones.

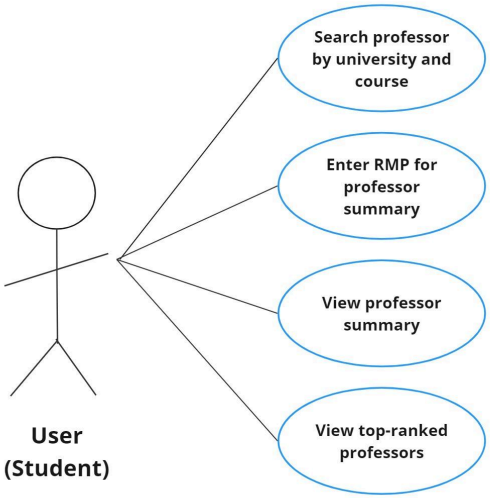
Non-functional Requirements



Use Case Chart/Diagram

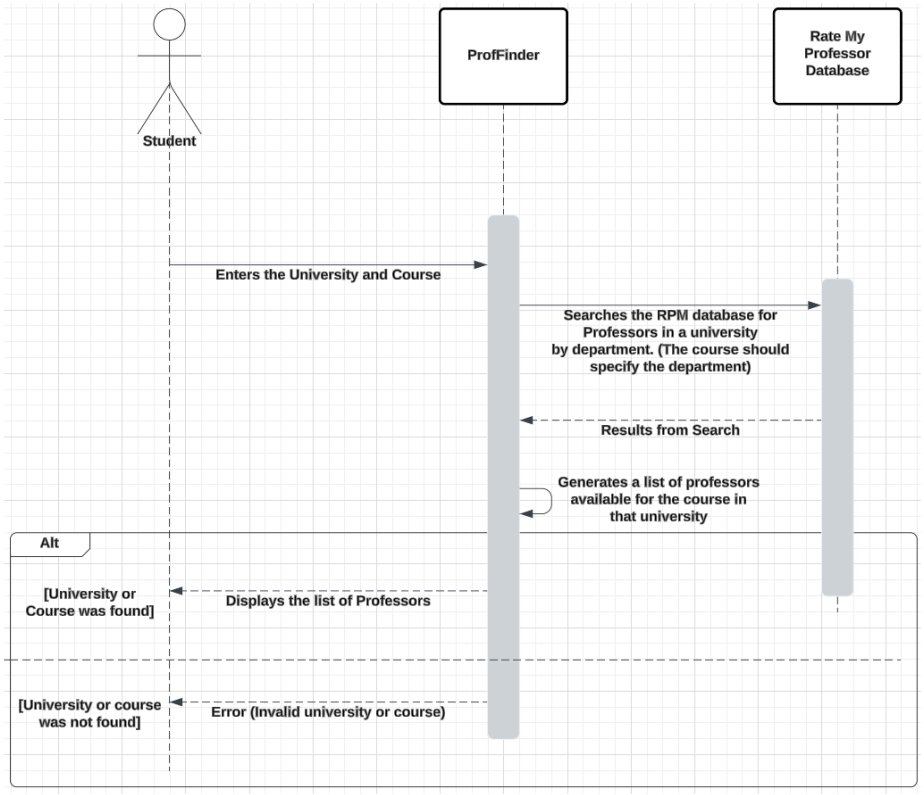
ProfFinder - View Professor Summary and Top-Ranked Professors	
Actors	Student (primary actor), Chatbot System
Description	A student inputs a Rate My Professor (RMP) link or specifies a course and university to receive professor evaluations. The system scrapes RMP data to provide summaries of pros and cons or ranks top professors
Data	RMP links, course information
Stimulus	User command issued by the student
Response	Summary of professor's pros and cons or list of top-ranked professors
Comments	The system must handle multiple requests and ensure data accuracy from RMP.

ProfFinder: Use Case for "Student"

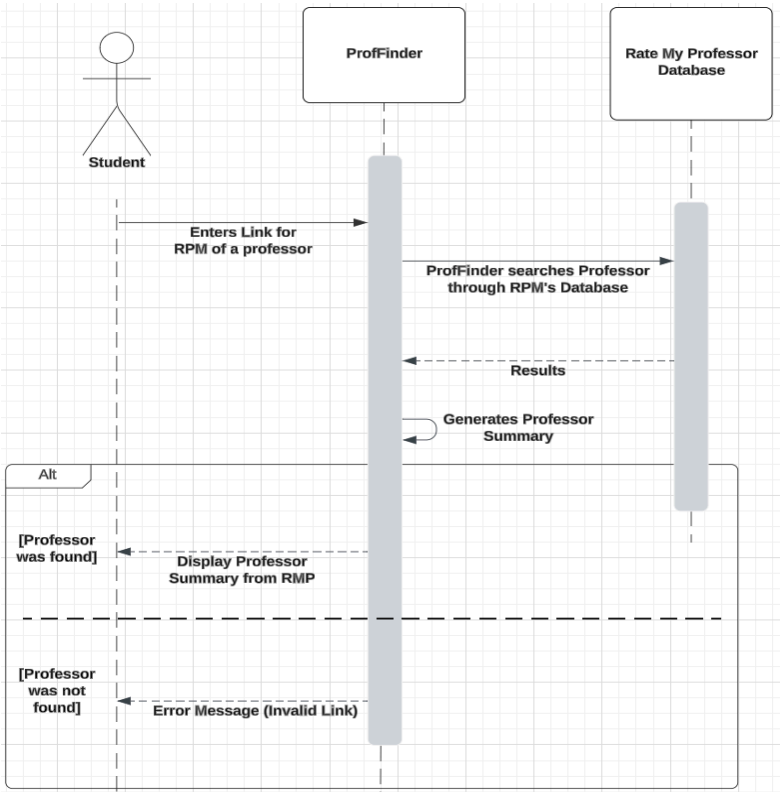


Sequence Diagrams

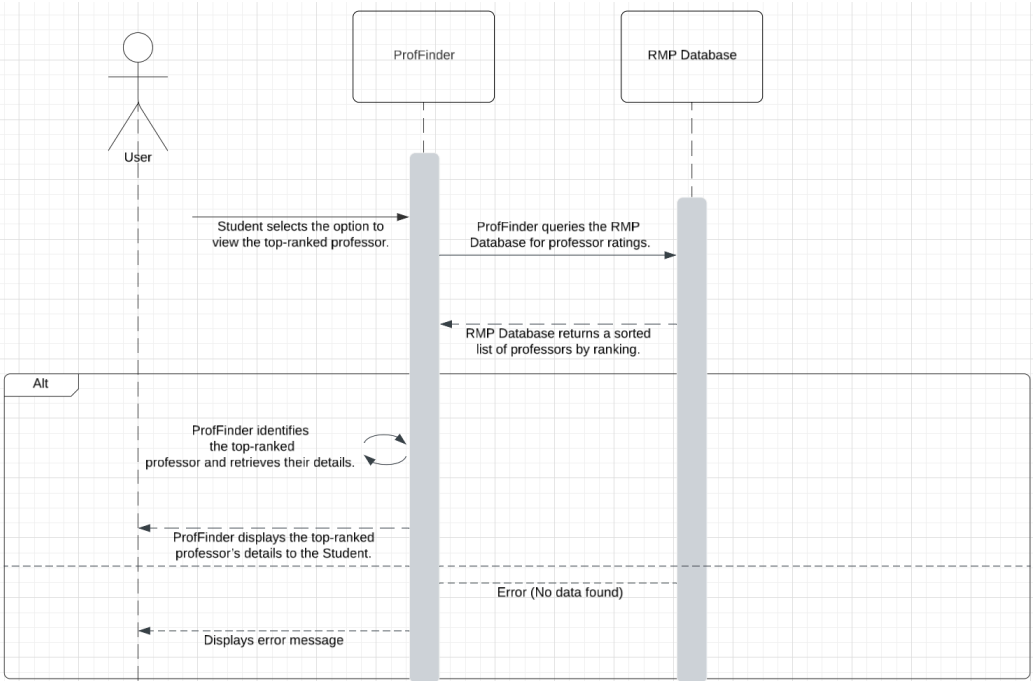
Use Case: Search Professor by University and Course



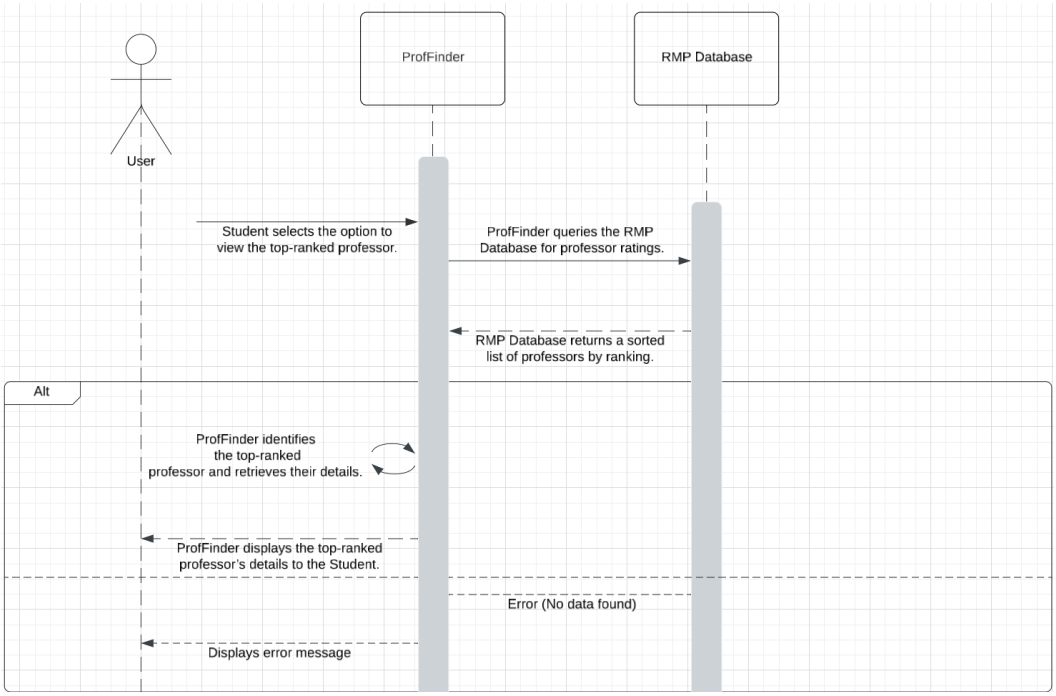
Use Case: Enter RMP Link for Professor Summary



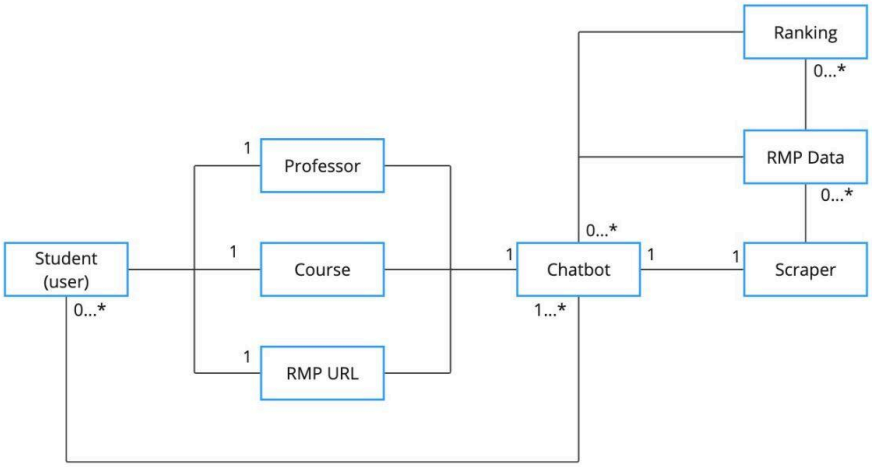
Use Case: Top-Ranked Professors



Use Case: View Professor Summary



Class Diagrams

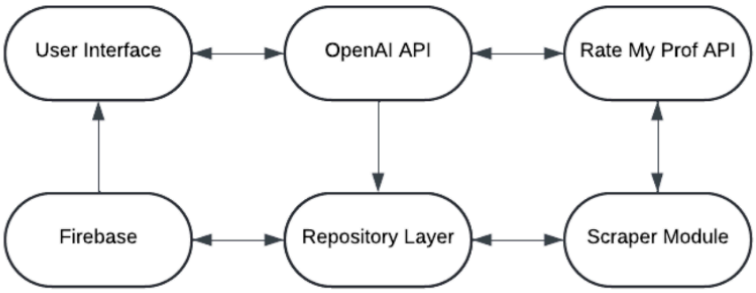


Student	Professor	Course	RMP URL
username email university	name department average_rating	course_name course_number professors_list	url
search_professor(course) view_summary(professor) rate_professor(professor, rating)	get_rmp_data() generate_summary() update_rating()	get_professors() add_professor() remove_professor()	fetch_data() validate_url()

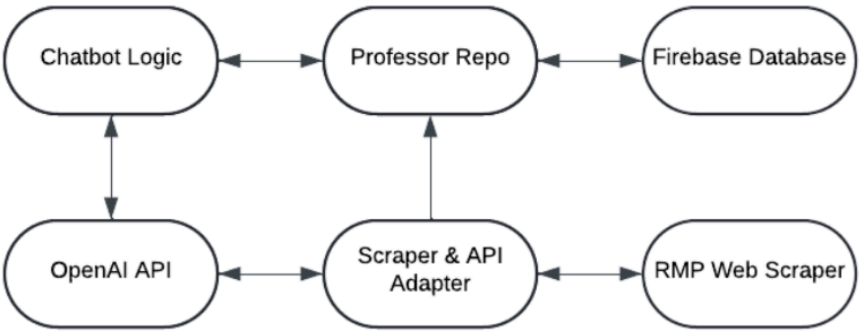
Chatbot	Scraper	RMP Data	Ranking
api_key response_history current_query	url html_content parsed_data	summary possitive negative	course_id professor_rankings criteria_weights
process_input() generate_response() integrate_with_openai()	scrape_data() parse_html() extract_information()	analyze_data() generate_summary() generate_ranking()	calculate_ranking() update_criteria() display_top_professors()

Architectural Design

High-Level System Architecture Diagram



Pipe and Filter Pattern Architecture Diagram

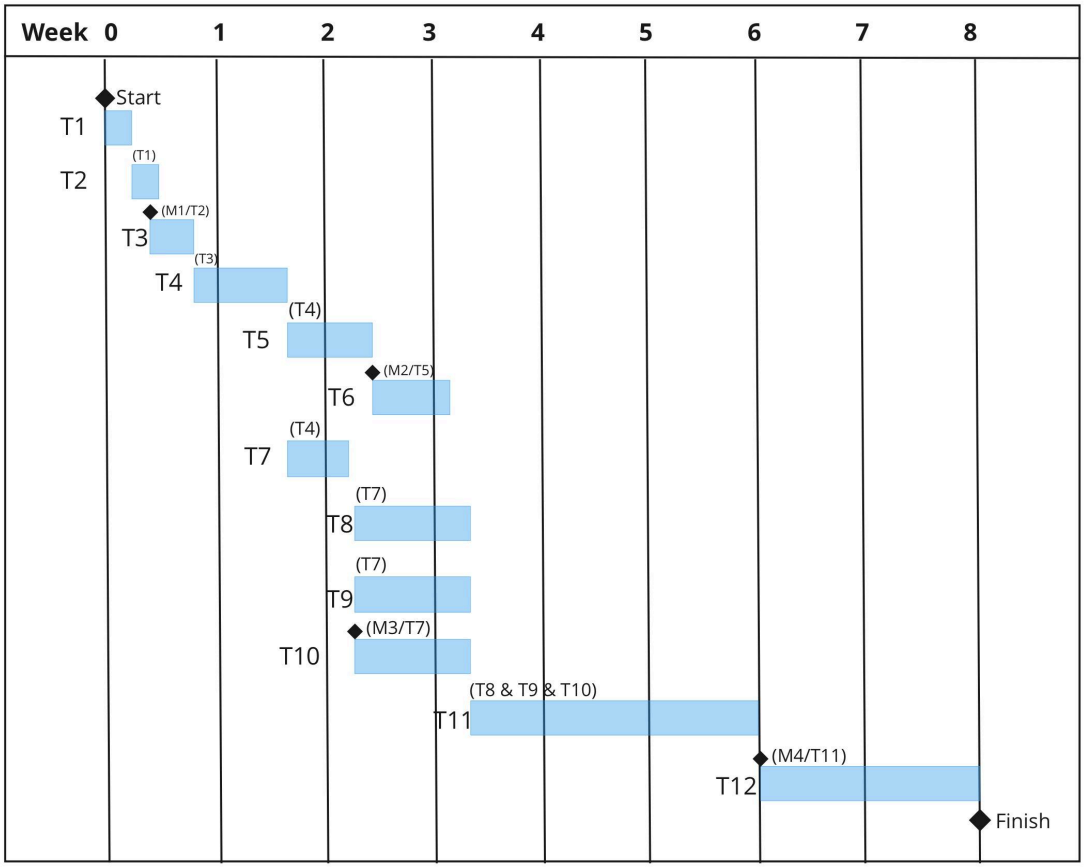


Project Scheduling

Project Task Scheduling Table

Task	Effort (person-days)	Duration (days)	Dependencies	Start date
T1: Define Project Scope	9	1		01/06/2025
T2: Team Coordination	9	1	T1: Define Project Scope	01/07/2025
T3: Set Up Development Environment	18	2	T2: Team Coordination (Milestone 1)	01/08/2025
T4: Integrate Firebase and OpenAI API	10	5	T3: Set Up Development Environment	01/10/2025
T5: Develop Web Scraping Module	15	4	T4: Integrate Firebase and OpenAI API	01/16/2025
T6: Implement Data Cleaning Functions	8	3	T5: Develop Web Scraping Module (Milestone 2)	01/22/2025
T7: Build Basic Chatbot Framework	15	3	T4: Integrate Firebase and OpenAI API	01/16/2025
T8: Develop Summary Generation Algorithm	15	5	T7: Build Basic Chatbot Framework	01/21/2025
T9: Implement Ranking System	15	5	T7: Build Basic Chatbot Framework	01/21/2025
T10: Integrate Firebase for Data Storage	10	5	T7: Build Basic Chatbot Framework (Milestone 3)	01/21/2025
T11: Conduct Testing of Core Features	27	14	T8, T9, T10	01/28/2025
T12: Prepare Document and User Manual	27	10	T11: Conduct Testing of Core Features (Milestone 4)	02/17/2025

Activity Bar Chart



Scheduling:

- Work Monday through Friday, with weekends off
- Start date: 01/06/2025
- End date: 03/03/2025

Justification:

- A team of 9 members
- The project will take approximately 8 weeks (40 days) to finish

Overall Cost, Effort, and Pricing

For the ProfFinder project, we will use the **Application Composition Model** [1] to estimate cost, effort, and pricing. This model is ideal for projects built from existing components, like ProfFinder, which integrates the OpenAI API and Firebase. By calculating "Application Points" (Object Points) for each functional element - screens, reports, modules - and weighting them by complexity, we can determine the project's total application points (NAP). The model also adjusts for code reuse and developer productivity, giving us a tailored estimate of the cost and effort for ProfFinder.

1. **Identify Application Points**
    - **Screens:** 4 screens

    - Input Screen for RMP Links
    - Input Screen for Course and University Selection
    - Output Screen for Professor Summary
    - Output Screen for Ranked Professors
- **Reports:**
    - Professor Summary Report
    - Professor Ranking Report #1 - which is based on overall rating, rating for specific attributes.
    - Professor Ranking Report #2 - which is based on users' filters (e.g., department, course, etc.), that adds complexity

Component	**Count**	**Complexity**	**Application Points**
Screens	4	Simple	4
Simple Report (Professor Summary)	1	Simple	2
Complex Reports (Rankings)	2	Complex	16

Total Application Points = 4 + 2 + 16 = 22

2. **Adjust for Reuse**

   We will adjust for reuse as we integrate existing APIs (Firebase and OpenAI). Let's assume 20% reuse

   New Object Points (NOP) = 22 (1 - $\frac{20}{100}$) = 17.6

3. **Determine Productivity (PROD)**

Experience	Low	Nominal
ICASE Maturity	Low	Nominal
Productivity (PROD)	7	13

   PROD = $\frac{7+13}{2}$= 10 objects point per person month

4. **Calculate Effort**

   PM = $\frac{NOP}{PROD}$ = $\frac{17.6}{10}$ = 1.76 person months

   Assume 5 working days a week and 4 weeks a month, so there are 20 working days per month.

Effort in person days = PM x 20 = 35.2 person-days

5. **Estimate Labor Cost:**
   Assume a developer's salary is approximately $7,000 per month.
   Cost per day = $\frac{7000}{20}$ = $350

   Total labor cost = $350 * 35.2 = $12,320

6. **Software & Hardware Costs:**
   - OpenAI API costs [3]: Estimated at around $100 per month for usage during development
     - OpenAI API cost for 2 months: $100 * 2 = $200
   - Firebase Hosting costs [2]: Estimated at $25 per month
     - Firebase costs for 2 months: $25 * 2 = $50
     - Server maintenance cost for 2 months: $100 * 2 = $200
   - Miscellaneous costs (tools, hosting, etc.):
     - Development Tools (IDE, libraries) estimated cost: $500
     - Licensed Software estimated cost: $400

   Total cost = $12,320 + $200 + $50 + $200 + $500 + $400 = $13,470

7. **Conclusion:**
   Total estimated effort: 35.2 person-days
   Total estimated cost: $13,470

Cost of Personnel

For this project, we have calculated the costs of 9 members; 9 personnel. According to the ACM above, we assume a developer's salary is approximately $7,000 for 2 months. However, this can be further expanded by examining the skill level of each personnel.

- The project will need a Project manager to oversee the application development and make sure the team is meeting deadlines and timelines. According to Heap [4], the average salary of a PM is $108,992, or about $52 per hour, assuming they work 5 days a week and 40 hours a week. The total cost for a PM for this project would be $14,643.20 for 2 months.
- The project will also need frontend and backend developers. According to Panchuk [5], the average salary of a frontend engineer is about $78,000, and for a backend engineer, $90,000. Assuming they work 5 days a week for 40 hours per week, that comes to $34 and $43 per hour, respectively.
- There will be 1 PM, 4 frontend, and 4 backend software engineers delegating tasks, which concludes to a total of $101,375.80 in cost for personnel, assuming

they will work 5 days a week, 40 hours per week, for 2 months.

Cost of Software:

We have calculated the total cost of software by using the ACM above. By not including the cost of labor, we can assume that the total cost of software comes to $1,350. This includes the cost for OpenAI API, Firebase Hosting, and miscellaneous (IDE, Modeling software, libraries).

Test Plan

The test plan for **`getProfessorData`** focuses on validating its ability to fetch and parse data from a professor's RateMyProfessor page. It includes two key test cases: successful data retrieval with expected properties (`name`, `rating`, `comments`), and error handling when the page cannot be accessed. **Using Jest**, we ensure that **`getProfessorData`** reliably returns expected data or raises an error if the fetch fails. This plan verifies the function's core functionality, providing a stable foundation for other components that rely on this data.

For example, the **`getProfessorData`** function was tested with two cases: successful data retrieval and error handling. In the first case, a URL returned expected professor data (`name`, `rating`, `comments`), and the function passed by returning the correct object.

It returned:
Dr. Example
4.5
Great teacher!

In the second case, an invalid URL triggered an error, and the function correctly threw an error message "Failed to fetch professor data."

It throws an error message:
"Failed to fetch professor data."

Both tests passed, confirming the function's reliability. The test code (`index.test.js`) is included in this zip file on submission or in its own zip file.

Code:

```
const axios = require("axios");
```

```
function parseProfessorData(data) {
  // Mock parse implementation: Adjust to parse actual HTML data as needed
  return { name: "Dr. Example", rating: 4.5, comments: ["Great teacher!,
Always is available to help other students!"] };
}

async function getProfessorData(url) {
  try {
    const response = await axios.get(url);
    const { data } = response;


    const professorData = parseProfessorData(data);
    return professorData;
  } catch (error) {
    throw new Error("Failed to fetch professor data");
  }
}


module.exports = { getProfessorData };
```

Unit Test Code(test.js file):

```
const { getProfessorData } = require("../src/index");
const axios = require("axios");


// Mock axios
jest.mock("axios");


// Test cases for getProfessorData
describe("getProfessorData", () => {
  it("should fetch and return professor data from RateMyProfessor URL",
async () => {
    const mockData = "<html>mocked data here</html>";
    axios.get.mockResolvedValue({ data: mockData });


    const result = await
getProfessorData("https://ratemyprofessor.com/professor-url");


    expect(result).toEqual({
      name: "Dr. Example",
      rating: 4.5,
      comments: ["Great teacher!, Always is available to help other
students!"]
    });
```

```
  });


  it("should throw an error if data fetch fails", async () => {
    axios.get.mockRejectedValue(new Error("Fetch error"));


    await
expect(getProfessorData("https://ratemyprofessor.com/professor-url")).reje
cts.toThrow(
      "Failed to fetch professor data"
    );
  });
});
```

Comparison to Other Works

There are other programs that are trying to achieve the same goal we are trying to achieve here. One of those is the RIT Rate my Professor extension on Google [6]. The problem with this program is it only works on the Rochester Institute of Technologies course book. We are designing our project to work on not only UTD's schedule planner, but also other schools as well. There is also another Google extension called Rate My Professor Google Chrome extension [7]. This doesn't have the same functionality as our project because it only shows a 1 through 5 rating of the professor and doesn't give the students detailed information about what kind of professor they are. There was also another program named Whiteboard that used to be made for UTD specifically, but it has been discontinued and removed from Google. Our project stands out because we are making a more general helper for students through the registration process that will not only help UTD students, but all students with detailed professor information.

Conclusion

ProfFinder showcases our dedication to developing a tool that is accessible, accurate, and user-friendly for students looking for professor evaluations at various universities. Through meticulous planning, teamwork, and thorough testing, our team has created a scalable system that utilizes both Rate My Professor data and tailored ranking algorithms to provide insights into professor quality. By incorporating Firebase and OpenAI APIs, we ensure reliable data management and quick responsiveness. Our architectural decisions, including the Pipe and Filter design, enhance modularity and allow for future growth. By addressing both functional and non-functional requirements, we have designed a flexible platform that distinguishes itself from existing tools, giving students a comprehensive and personalized experience when choosing their professors. The development of ProfFiner reflects our commitment to software engineering, balancing technical excellence with a focus on user needs to serve diverse student communities effectively.

References

[1] I. Somerville, *Software Engineering - Global Edition*, 10th ed., Pearson Education Limited, 2019.

[2] "Pricing plans," *Firebase*. Google, [Online]. Available: https://firebase.google.com/pricing. [Accessed: November 1, 2024].

[3] "Pricing" *OpenAI*. [Online]. Available: https://openai.com/api/pricing/. [Accessed: November 1, 2024].

[4] "What does a Product Manager *Really* Do & How is the Role Changing?," *Heap*. [Online]. Available: https://www.heap.io/topics/what-does-a-product-manager-do [Accessed: November 10, 2024].

[5] Maryna Panchuk, "Front End vs Back End Developer Salary: Comparing Across IT Companies," *Alcor BPO*, May 04, 2023. [Online]. Available: https://alcor-bpo.com/front-end-vs-back-end-developer-salary-comparing-across-it-companies/. [Accessed Nov. 11, 2024].

[6] *RIT Rate My Professor Extension*. Chrome Web Store. July 24, 2023. [Online]. Available: https://chromewebstore.google.com/detail/rit-rate-my-professor-ext/lcionigofpcbfpmnipnioapimoggnbda?pli=1 [Accessed: November 7, 2024].

[7] *Rate My Professor Extension*. Chrome Web Store. March 30, 2024. [Online]. Available: https://chromewebstore.google.com/detail/rate-my-professor-extensi/hgfogfefocfabnfjmjijfcjogeghmocn. [Accessed: November 7, 2024].