Team members:
Félix Dasta
Armando León
Emmanuel Hernandez

Student Academic Virtual Assistant

1 Introduction

1.1 Background and Motivation

Given how broad the academic field is, it is highly likely that students will have doubts in multiple aspects of academic life. From the perspective of the student, these aspects include, but are not limited to:

- What courses have I taken?
- Which courses am I missing to complete my degree?
- What GPA do I currently have?
- How does my academic performance compare to that of other students in the same curriculum as me, or those who have taken the course under the same conditions as me?
- Which professors will teach the course that I want to take next semester?
- What feedback have students who have taken the course left to help me achieve my academic goals?
- I want to see the course syllabus beforehand, but I don't want to search for those individually one by one. Can I extract it directly from my curriculum?

These needs raised by students are our motivation to develop a platform that helps them to plan in a better way their academic career.

1.2 Justification

There exist similar platforms, however, they are found on separate web pages and some also carry a cost to use the same. Given that all the previous elements mentioned are related, considering a single platform to attend the needs of the students will be a good idea. This, and taking into consideration the tool that will be capable of comparing the student performance relative to the performance of their peers based on historical data, will make the product unique.

2 Problem Statement

2.1 Problem Definition

Some universities don't have tools to help students to plan their academic life in a better way. Of course, there are numerous free tools that perform tasks such as GPA calculation, but they lack functionality beyond that, such as storing courses that have been taken and displaying them in an organized manner that can allow the user to see their academic progress at a glance. There exist fewer possibilities where the user might be enrolled in a second major, or they might be undertaking a minor degree. These less common situations are not covered by free existing applications. There are also tools to see feedback from professors with whom the students would like to take a class, but these tools often do not provide specific feedback based on a certain course and section. Finally, many students would like to compare their performance relative to that of the peers in their class, to know if they are on the right track throughout their academic career. However, there is no public platform that allows us to extrapolate data anonymously from people that have taken the same courses as the student.

2.2 Target Domain

Our target domain is academic improvement for both the student and the professor. From the student's perspective, if the student notices that they performed below their peers in most of the courses, then the student will have in mind that they will have to make some adjustments to their study habits. Peers who give feedback on the section of courses they have previously taken will be able to tell which tactics worked and which did not in order to pass the course successfully, so that future students who will be taking the course will keep this in mind. This feedback can be filtered so that the students can search for a section that best suits the criteria they desire for the course they will be taking, as there will be variations according to the course modality and evaluation techniques decided upon in a given semester.

From the professor’s perspective, based on feedback from their students, they will be able to consider what aspects they can improve in their courses for future occasions, and what teaching tactics they should continue to use according to the students. These feedbacks can be filtered under different criteria, such as courses and semesters, or you can also search for feedback for a specific section if you wish. All feedback will be anonymous, however, it will be possible to see which department the student who left said feedback belongs to. In essence, this can be a useful piece of information to keep in mind when looking at feedback from any section, as there are some courses whose requirements may vary a bit according to the student's curriculum, and thus different gaps can be identified according to the feedback from these students from different departments.

2.3 Proposed Innovation

The solution we propose to materialize these perspectives, features, and ideas is to provide a digital platform in the form of a website. This website will provide its users with the tools they need to fulfill and track their academic progress and improve upon it. However, it is characterized as a digital platform primarily because a key feature is its anonymized feedback system where users can share their knowledge and experience with courses they have previously taken. Consequently, this will stimulate a positive feedback loop where the community of students can help themselves, each other, and their professors with the aim of improving the overall university experience.

3 Project Objectives

3.1 SMART Objectives

- As of February 17, 2022, we expect to have the project proposal completed. In this project proposal, the problem statement, objective, solutions, technical description and project plan will be established.

Those details will aim to facilitate the development process of the project, which will be started by February 18, 2022. Technical description such as system architecture, modules, ER Diagrams, Wireframe for UI, flowchart for algorithms and engineering constraints will be taken into account for the development process.

- The main objective of this project is to develop an application that will provide a unique user system, a curriculum viewer, a performance reviewer and course details by May 6th, 2022. Our team will achieve this set goal by operating in an Agile manner that will allow us to work on pre-defined issues, which will be agreed upon by the team in a meeting. By dividing the project into stories, we will be able to create shares of equal work that will allow us to meet our goals. The sprint period we will use will be two weeks of length.

3.2 Task Completion Definition

The team will ensure that the stories created will be clearly defined, thus allowing us to use stories as a task completion metric. Stories will be created using the Trello platform, given that it allows us to keep track of who has been working with which tasks, provide comments on some specific tasks (to have those for future reference), and to keep track of the amount of work which each of the members of the team has accomplished, to ensure that everyone is being productive.

3.3 Success Definition

Since most of these stories will encapsulate features of the platform, once finished we will also be able to test said features. This will allow the team to measure the success of development. Tangible goals will grant the team a sense of fulfillment once a task is finished. As we are aware of this, productive planning sessions are important to help meet the deadline established. By successfully completing this project, we will produce an application that will allow students to further improve their academic performance and allow them to fulfill their academic goals more efficiently.

4 Solution Approach

4.1 Solution Proposed

Based on the academic characteristics that our university community requires so that they can be better equipped to have a successful college experience, it was determined that the following features will be necessary for the system-to-be:

- **Unique user registration:** We want to limit the amount of spammers in the system, given that we want the users statistics to be as accurate as possible. Hence, we will be limiting registration to users with an academic institution address, and they will have to verify their account via an email that will be sent to them.

- **Student curriculum viewer:** Student curriculum is used so that the students can benefit from access to their GPA calculation by semesters or academic year. Through predefined curriculum templates provided by different universities, students will be able to choose the curriculum that corresponds to them so that they can start entering their records. If the university is not in the system, we will allow the student to create the curriculum based on the one provided by their university. However, if the student wishes to benefit from all the app features, by using a form, students will need to notify the administrators of the page which university they wish to add to the system and their corresponding curriculum. While the students are looking at their academic curriculum, they will be able to see the details of the courses from their curriculum.

- **Academic minor feasibility:** It is likely that many students want to complement their academic major with a minor, therefore the students will be able to see the academic minors that are available at the university. When the student selects the desired minor, they will see which courses they need to complete, and have already completed, to obtain the minor degree.

- **Course details:** The details of the course will include the syllabus, the professors who have taught that course and any other relevant information. To clarify, our purpose is to provide constructive criticism, so that the student can learn from the professor's teaching tactics beforehand. The students will be able to see advice from other students in order to succeed in the course.

- **Student performance reviewer:** When students get their course grades and enter them into the system, they will be able to compare their performance relative to that of other students in the course anonymously. This will allow them to see if they are on the right track, academically, relative to their classmates. They will be able to evaluate the performance based on different criterias, such as, for example, extrapolating all the historical data anonymously of all the students who have taken that course previously or making a comparison based on the students who took that course in the same semester, since it would adapt more to the circumstances of that moment.

4.2 Gap With Existing Solutions

Similar market offerings lack most of the capabilities that our platform will offer. Academic organizers in the market provide the user with capabilities such as class tracking but the user must manually enter the course, so in essence it is like a note taking application. Our solution will provide the user, given that the institution they are a member of is participating in the platform, with a more intuitive solution. The user will be able to see their curriculum and decide which classes they have taken. Our solution will also provide the user with a ranking system, comparing their academic performance to that of their peers. This is not a feature that any current market offering has.

A feature that is present in other market offerings is the course feedback system. We will differentiate our offering with that of the market by emphasizing on the lack of censorship and the lack of validation that existing solutions suffer from. We will implement a censorship system to ensure that user experience is not hindered and an environment of respect is maintained. We will also ensure that every reviewer is a valid user by requiring institutional credentials, such as their email address.

4.3 Value Proposition

This project will provide users with an at-a-glance overview of their academic careers, as well as useful information about their study programs. All this information will be offered in the convenience of a single platform. We believe that providing students with information on how other students have performed in a specific course will give them an extra confidence boost to tackle and progress through their academic paths. If a user is ever interacting with a potential employer or a recruiter, and they are not able to remember information about a specific course they have taken, our easy to access curriculum viewer and progress tracker will allow them to quickly verify as well as provide them with other information, such as their GPA. Aside from benefiting students, professors will also be able to benefit from the platform. The feedback system will allow professors to see what different types of teaching styles work better for students. This extra information may allow them to more easily reach a decision on changing or modifying their teaching style. Most importantly, the user can feel sure of the fact that the feedback is valid due to the strict user registration that our application will have.

4.4 Benefits and Limitations

Due to the nature of some of the platform's features, some limitations may be encountered if there is not an established user base. One of these features would be the feedback system. If there are not enough users participating, there might be a case where a specific professor or section might not have any feedback. Another feature that could be affected by lack of an established user base would be the student ranking system. Without enough students participating, it would be difficult to produce an accurate ranking, thus affecting the functionality.

If a consistent user base can be created, then the platform would be able to work to its full potential. This would allow users to be able to take advantage of accurately working features. Users will benefit from the platform by being able to track their performance relative to their classmates, view school curriculums, quickly access and learn about what a professor's teaching style is like to better choose a section, and in the case of a professor, they would be able to have access to the constructive feedback left by students.

4.5 Commercial Potential

Although the desired model would be a completely free to use model, overhead costs such as hosting will require a monetary cost to be able to upkeep the platform. Luckily, revenue producing techniques such as ad spaces would allow us to meet such costs and keep the platform hosted. The downsides to this would be that user experience would be affected by said ads. An alternative would be to implement a premium user account, which would allow the user to pay a one time fee to remove advertisements from their viewing experience permanently or for a defined time period. This would be completely optional and would not offer any other advantage or features over non-premium users, aside from no advertisements.

4.6 Required Resources

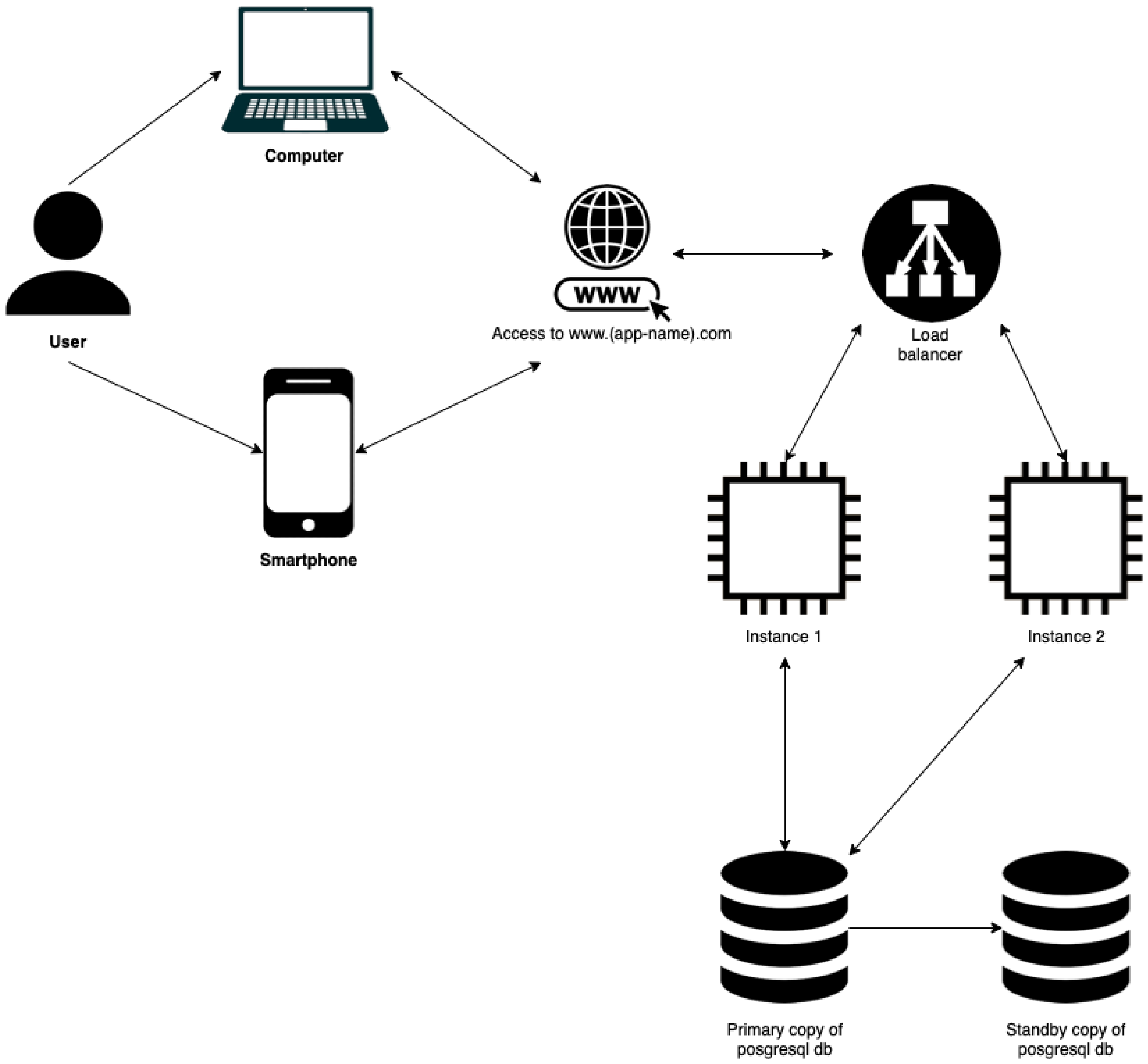
The only monetary resources required for the planned product would be to cover overhead costs such as hosting. Aside from this, we will also need the curriculum files for academic programs, so that they can be added into the platform. The labor resources would be fulfilled by the team, so nothing else would be required.

4.7 Intellectual Property

Our platform will not make use of any intellectual property. The data that we will be leveraging for statistics is available for the public domain at no cost. Any logo or art design that the team decides to use, will be compliant with trademark and copyright laws. After researching for trademarked applications that are similar in scope to our platform, we were not able to find any that matched the criteria.

5 Technical Description

5.1 Define the system architecture


Computer
User
Access to www.(app-name).com
Load balancer
Smartphone
Instance 1
Instance 2
Primary copy of posgresql db
Standby copy of posgresql db

In order for the user to be able to use our application, what will be required from them is a device that is capable of connecting to the internet and that has a browser to access the web. However, from the perspective of the owner, since the expectation of it is that it can be used for multiple universities, it is possible that there are many users connected simultaneously. Thus, the server will need to be capable of handling load balancing. Based on our research, AWS seems to be a viable option for us, given that the hosting cost will be proportional to the amount of use and they have servers in multiple regions around the world, making it feasible to the user so that they can connect to the closest available zone. Also, in order to improve application performance and save money, most accessed data will be stored on the server by using caching techniques (such as Redis) provided by AWS, so that it is not necessary to have to look up the information all the time in the database. When a user connects to the application and they get redirected to the website by connecting to the closest availability zone, it is highly likely that some data related to their university is already in the server cache due to the fact that users from a certain geographical location will be more interested in knowing information about a university that is closer to them.

5.2 Modules To Implement

User Interface Component Modules

- **Navbar:** Gives the user easy access to the 4 main views supported by the application: Dashboard, Professor Feedback, Course Feedback, and Curriculum.

- **Search:** Gives the user the choice to quickly search for a specific professor or course instead of searching for them within their curriculum.

- **Curriculum:** This component will contain the list of courses that pertain to a given semester that the user has completed or is currently enrolled in. Each course listing will have relevant information and will allow the user to directly enter that course's feedback page or it's professor's feedback page.

- **Stat Widgets:** There are various components across all views supported in the web application which require the displaying of statistics in the form of graphs and numbers; however, they all share similar attributes and structure. For example, the grades obtained by the students in a section will be represented as a pie chart.

- **Featured information:** In both feedback views there are components that show selected feedback which are either popular or recent and therefore relevant at a first glance for the user.

- **Feedback:** For both feedback views, this component will be used to represent the feedback that a student has left.

- **Filter:** Various views, notably the curriculum, make use of this family of components which are used to limit the scope of the pool of information for the ease-of-use of the user.

User Interface Pages Modules

- **Home Page:** The home page serves the purpose of introducing a potential user to the intended goal of our web application. There is an image carousel which has rotating views of the system as well as brief descriptions of their purpose. Additionally, there is a short demonstration video that will showcase use cases and key features of the web application. Finally, there is an information section which explains more in depth the motivation behind the app's creation as well as an introduction of our team.

- **Student Dashboard:** The main face of the application, the dashboard's purpose is to provide the user with all the information they need quick access to. This includes their academic profile information as well as various important stats such as GPA and missing credits. Right in the center is the user's current curriculum with interactive elements that display any chosen course along with some brief information about it and it's professor.

- **Professor Feedback:** This page serves more as a collection of summaries which can allow a student to understand how it is that a given professor teaches. It includes a course picker, where students can see feedback for sections of specific courses that the given professor teaches. Additionally, it has a "quick facts" section which will provide information such as average student grades as well as featured feedback.

- **Course Feedback:** This page is a key feature of our platform where students can help their professors out by giving feedback about the techniques they use and what they think of the class they took. Of critical note is the course rating which allows a user to understand the difficulty level of the course compared to the past. The user can view the feedback and ratings left for all professors that have given the chosen course across all time ranges (as is feasible). Even more, the user can access the course syllabus (if available) as well as view average grades and featured feedback for each section.

- **Curriculum:** The last of the supported views, this page allows the user to access the curriculum they are currently enrolled in. Information pertaining to curriculum specifics such as department and credits is present on the bottom panel of the page. The curriculum can be filtered by year and semester and is accompanied by a side panel that shows the user which courses they've yet to take. Furthermore, these missing courses can be filtered by all sorts of categories so that future course planning is streamlined and simplified.

Object Relational Model Modules

An object relational mapper will be used to manipulate data from the database using an object-oriented paradigm. By using an ORM, this will make the code easier to update, maintain and reuse, given that we will manipulate data as objects. Also, it will shield our application from "SQL Injection Attacks". The object parameters and relationship with other entities that will be used in the ORM can be seen in section 5.3, where the ER Diagram is being described.

Algorithm Utils Modules

- **Get general GPA:** This function will get all the courses that the user has registered in the database, and it will calculate the general GPA.

- **Get departamental ranking:** Retrieve every student stats that are available at the student's departament, and get ranking based on other students' that belong to the same curriculum.

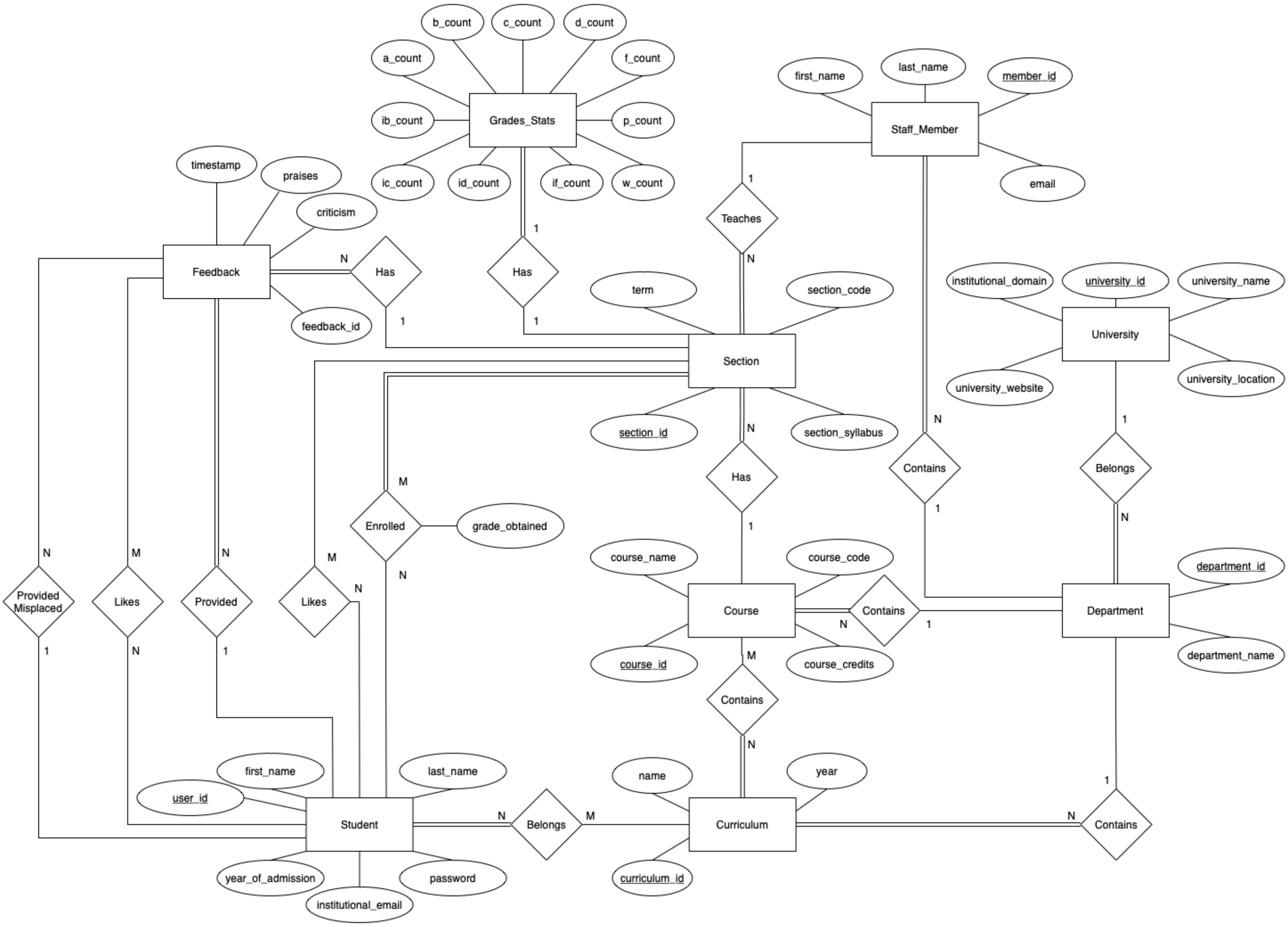
- **Get average course GPA:** Based on historical data from students that have taken a course, average course GPA will be retrieved. Letter grade counts will also be retrieved from the courses.

- **Modify semester GPA:** If we want to add a course to a specific semester, the course will be added and the GPA for that semester will be modified.

- **Map course grade stats:** Course grade stats will be mapped to the course object. Grade stats will be retrieved from real grade stats data if it is available, otherwise, it will be interpolated from the registered students of the system.

- **Verify profanity on feedback:** Based on the desired feedback that will be published, checked if the text may contain some profanity. If it doesn't pass the profanity checker filter, mark the comment as misplaced.

5.3 ER Diagram and Description



Relationships:

1) **Student and curriculum:** It is possible that the student belongs to multiple curriculums (example: student is taking a second bachelor degree or is complementing their major degree with a minor degree), and it is possible that the academic program to which the curriculum belongs to contains multiple students (typically, that is always the case). Also, all elements from the student side must have a curriculum.

2) **Department and curriculum:** It is possible that a single department has multiple curriculums, however, every curriculum must belong to only one department.

3) **Curriculum and course:** A curriculum should contain multiple courses, hence, the courses that the curriculum must contain should be created before the curriculum. Also, a course can belong to multiple curriculums (example: calculus courses).

4) **University and department:** A university contains multiple departments and a department belongs only to one university (every department in each university is unique, each with its corresponding staff members and courses). Also, a department can't exist without the university, given that the university is responsible for having multiple departments for its function.

5) **Department and staff member:** Every department can have multiple staff members but every staff member must belong to only one department. Note that there are cases when a professor can give courses that belong to different departments, however, under the typical university rules, they are an official member of one department only (example, a professor of electrical engineering can be teaching a computer science course).

6) **Course and section:** Every course can have multiple sections, however, every section should belong to only one course.

7) **Department and course:** A department can contain multiple courses, however, each course should belong to one department only. It is possible that some newly created departments don't have any courses yet.

8) **Staff member and section:** A single staff member (professor or teaching assistant) can give multiple sections, but each section must be given by one staff member only. Note that some courses can be provided by multiple people at the same time (that is, a professor and teaching assistants for instance), but under the system, only the course manager (the instructor) is the one that is registered.

9) **Section and grade stats:** A section can have records for grade stats, and each record belongs to one section only. Grade stats can't exist without a section. Note that it might be possible that some universities don't have public records of grade stats, hence, it is not necessary for a section to have grade stats.

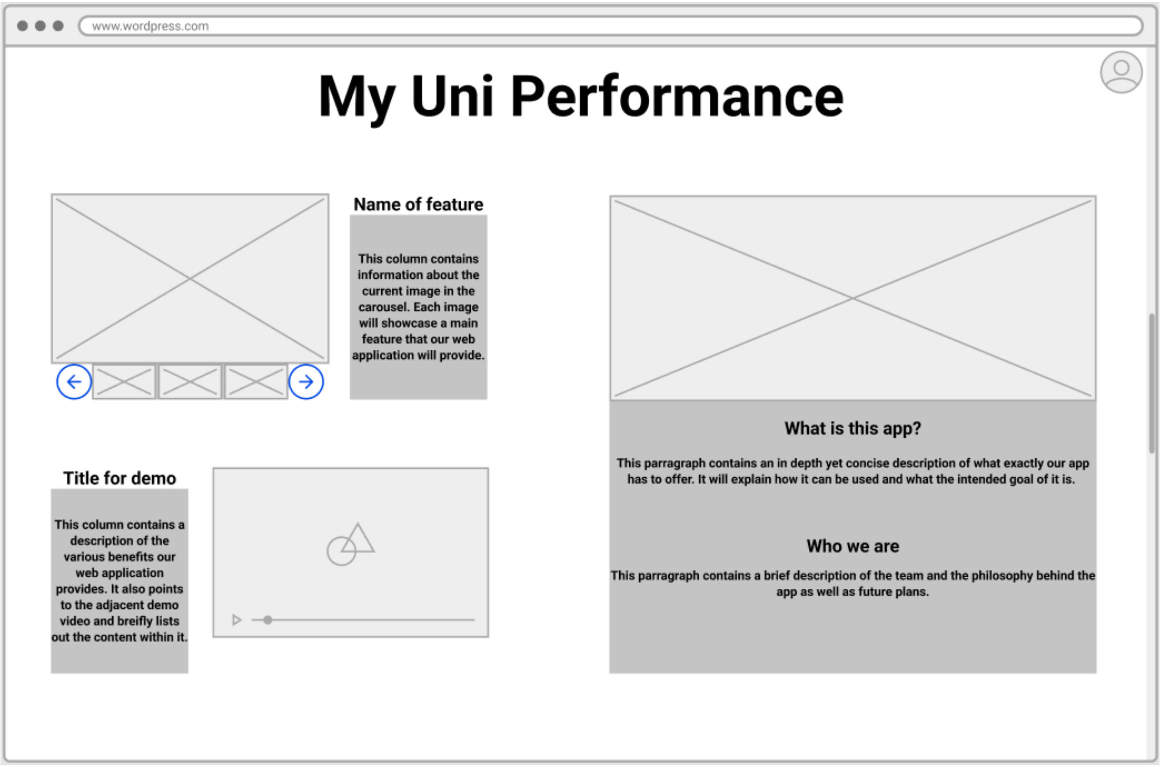
10) **Feedback relationship between student and section:** A student can leave multiple feedbacks, either to one section only or for more than one section. Also, a section can have multiple feedbacks, either from one student only or more than one student. But each feedback is left by one student only and corresponds to one section only. The student can thumbs up many feedbacks

and a feedback can contain many thumbs up from different students. Thumbs up represents if the student has liked the feedback. Note that a provided feedback can be inappropriate, and it might be possible that a student provides one or more inappropriate feedback, but each inappropriate feedback belongs to one student only. Inappropriate feedback won't be shown to the public, however, those will be stored in the database so that the administrative staff can determine if the user deserves the suspension based on those comments that may contain profanity.

11) **Student and section:** A student can be enrolled in multiple sections, and a section can have multiple students, but a section can't exist without students. Also, optionally the students may be able to thumbs up one or more sections if they want, and each section can contain many thumbs up from different students.

5.4 Describe Wireframe for UI

Home page



My Uni Performance

Name of feature

This column contains information about the current image in the carousel. Each image will showcase a main feature that our web application will provide.

What is this app?

This parragraph contains an in depth yet concise description of what exactly our app has to offer. It will explain how it can be used and what the intended goal of it is.

Who we are

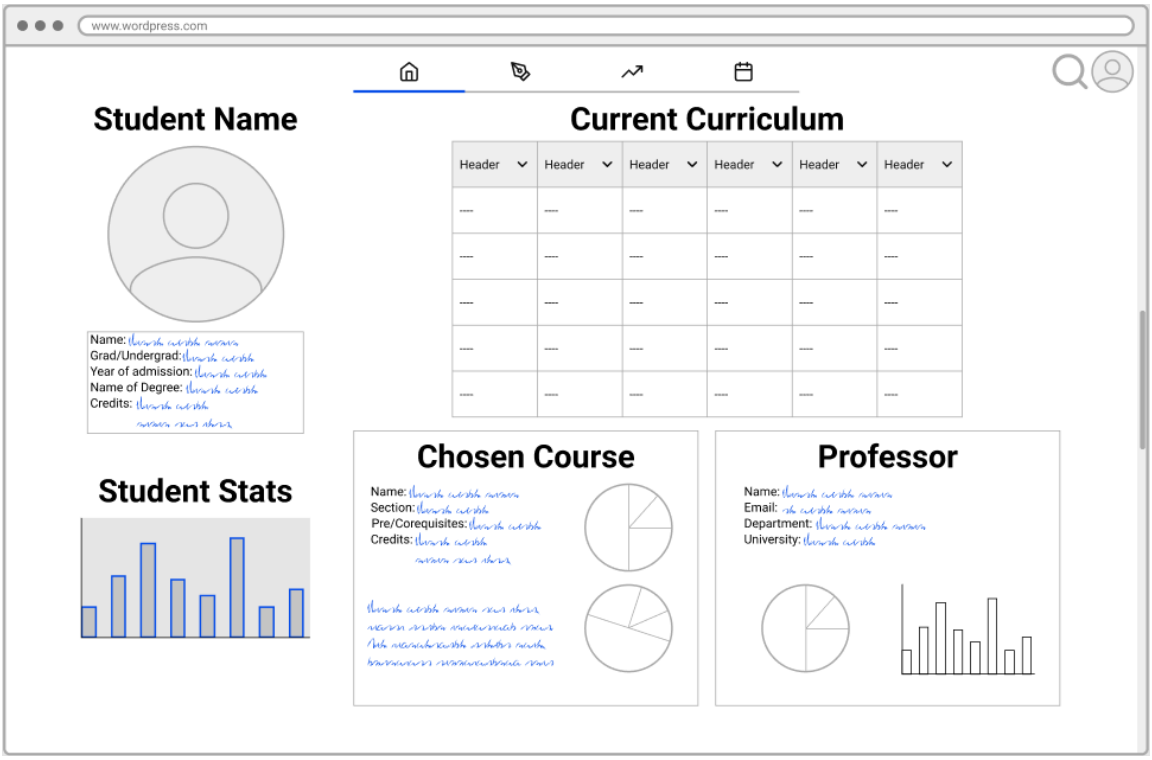
This parragraph contains a brief description of the team and the philosophy behind the app as well as future plans.

Title for demo

This column contains a description of the various benefits our web application provides. It also points to the adjacent demo video and breifly lists out the content within it.

This is the application landing page, where general information about the application and its purpose can be seen.

Student dashboard



Header	Header	Header	Header	Header	Header
----	----	----	----	----	----
----	----	----	----	----	----
----	----	----	----	----	----
----	----	----	----	----	----
----	----	----	----	----	----

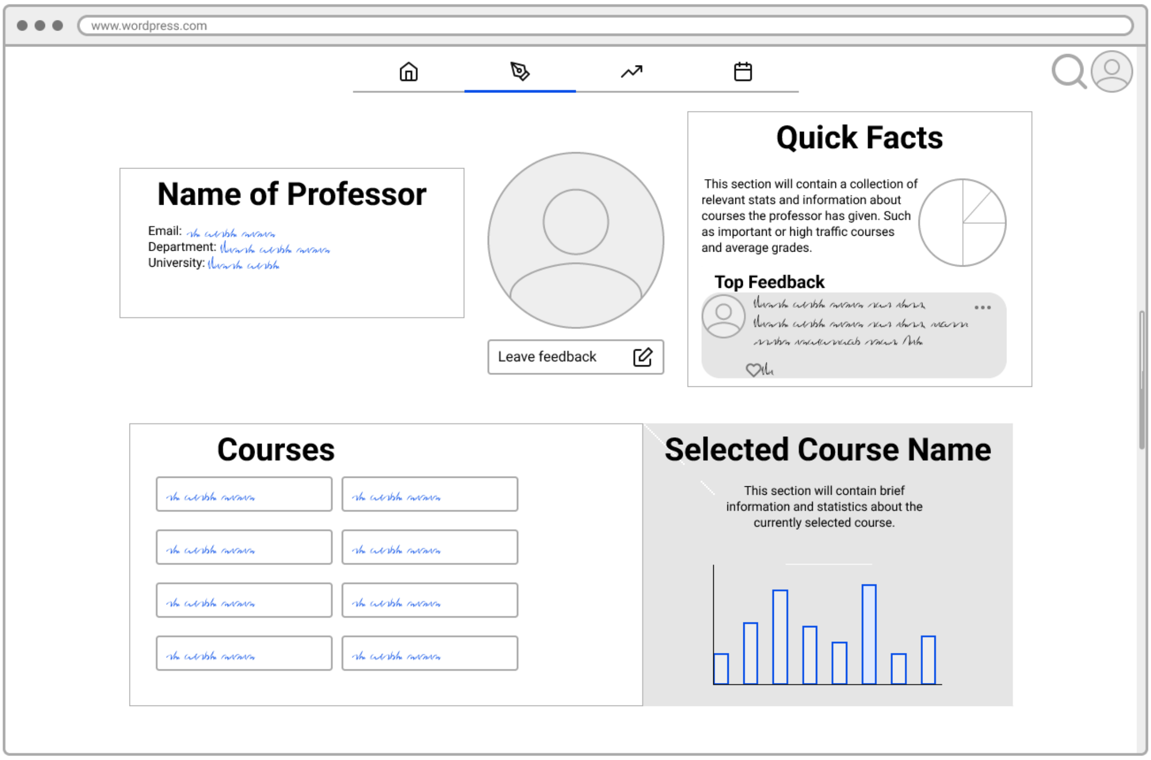
Name:
Grad/Undergrad:
Year of admission:
Name of Degree:
Credits:

Name:
Section:
Pre/Corequisites:
Credits:

Name:
Email:
Department:
University:

The student dashboard allows the student to have quick and easy access to all relevant information they need, such as courses taken and their statistics.

Professor feedback



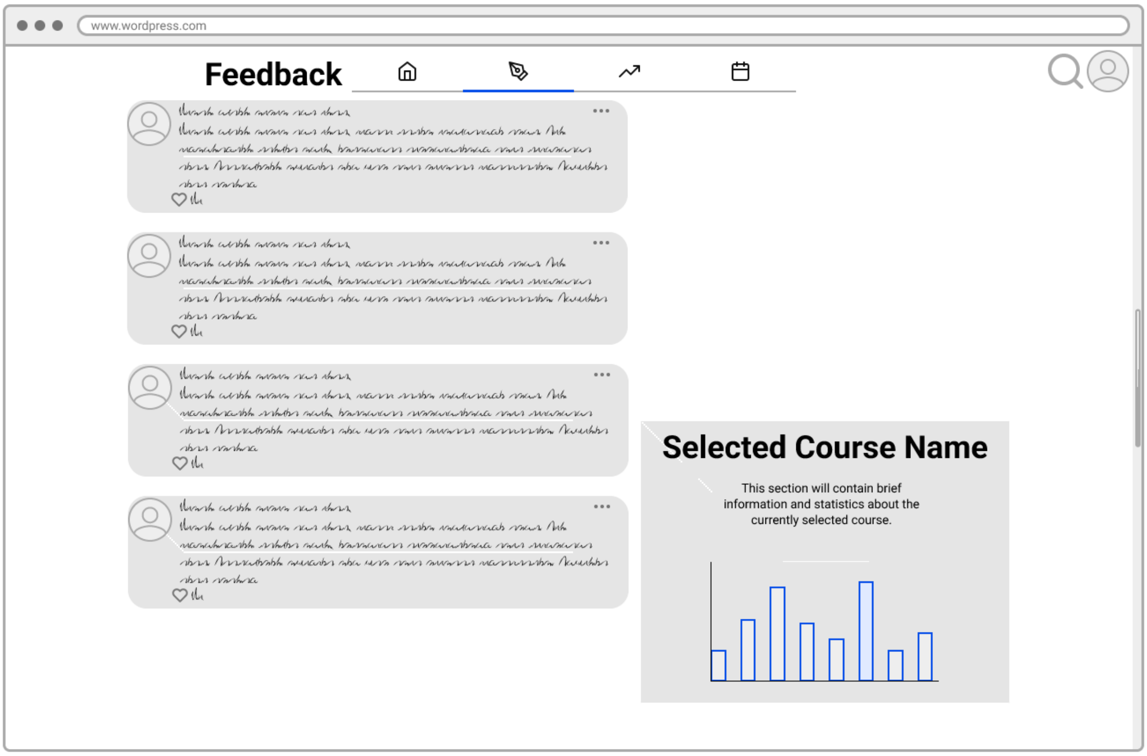
Email:
Department:
University:

This section will contain a collection of relevant stats and information about courses the professor has given. Such as important or high traffic courses and average grades.

This section will contain brief information and statistics about the currently selected course.

This is the professor feedback page, it allows the student to view how other students have rated a professor before, and it allows the student to leave their own feedback.

Professor feedback feed

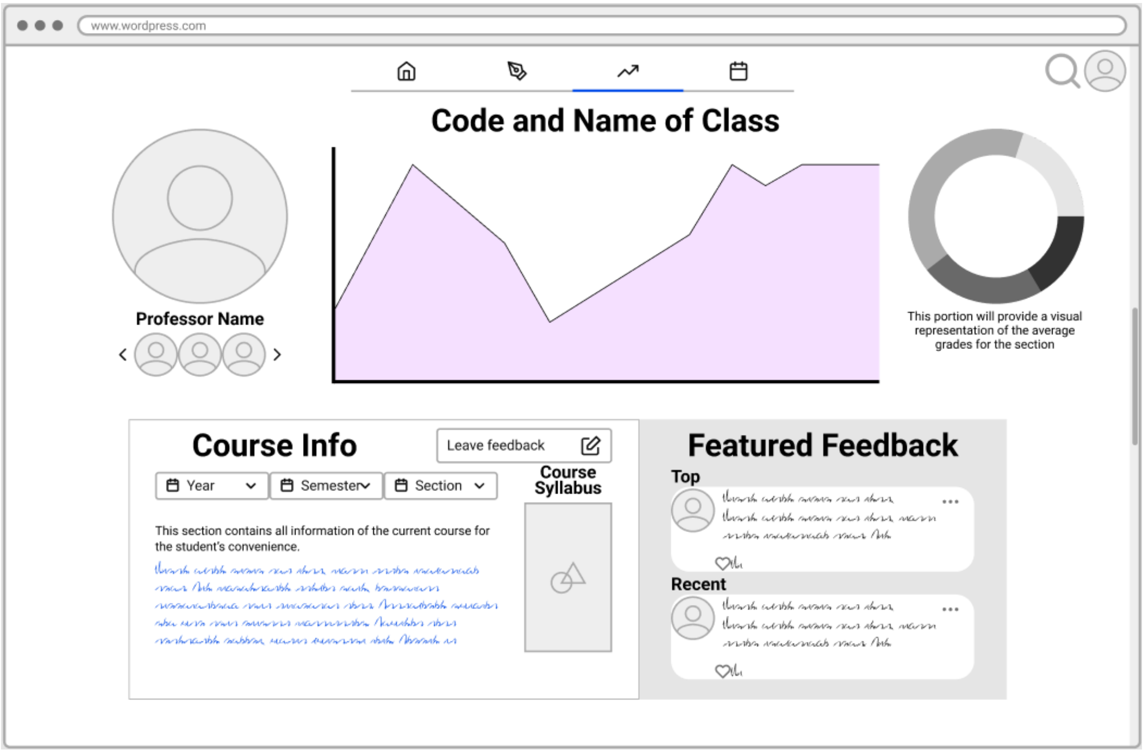


Selected Course Name

This section will contain brief information and statistics about the currently selected course.

This is a view of the feedback feed for a given professor based on the sections they have teached.

Course feedback



Code and Name of Class

This portion will provide a visual representation of the average grades for the section

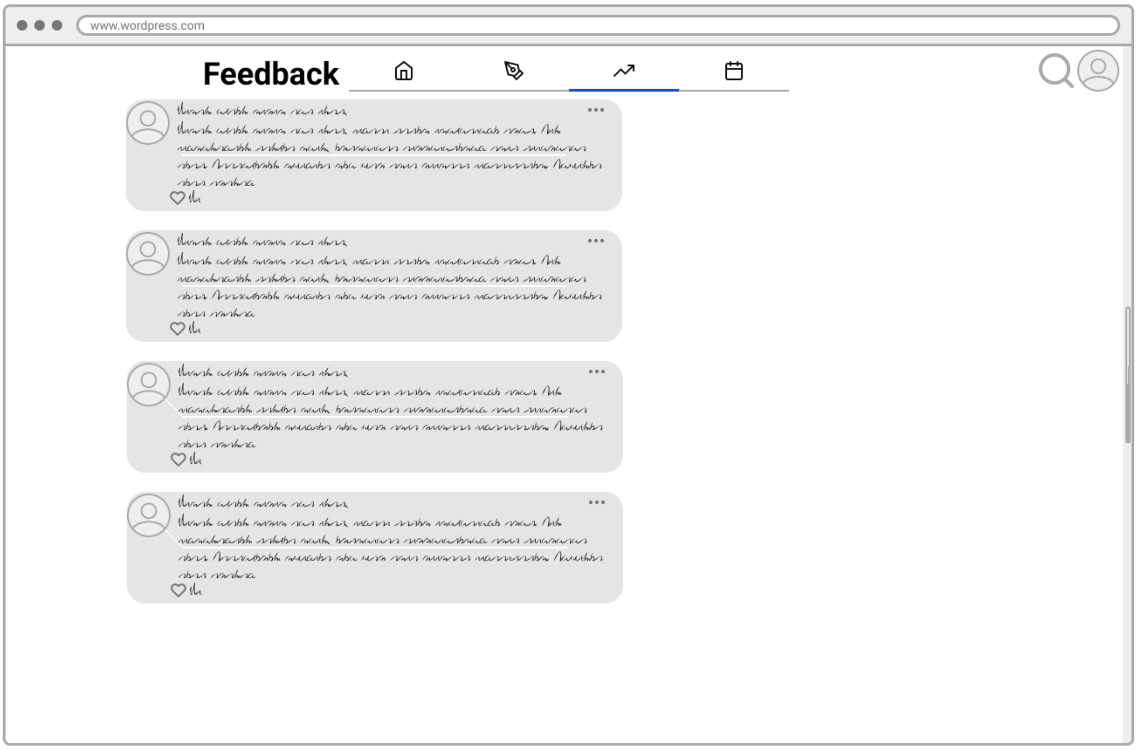
Course Info

This section contains all information of the current course for the student's convenience.

Featured Feedback

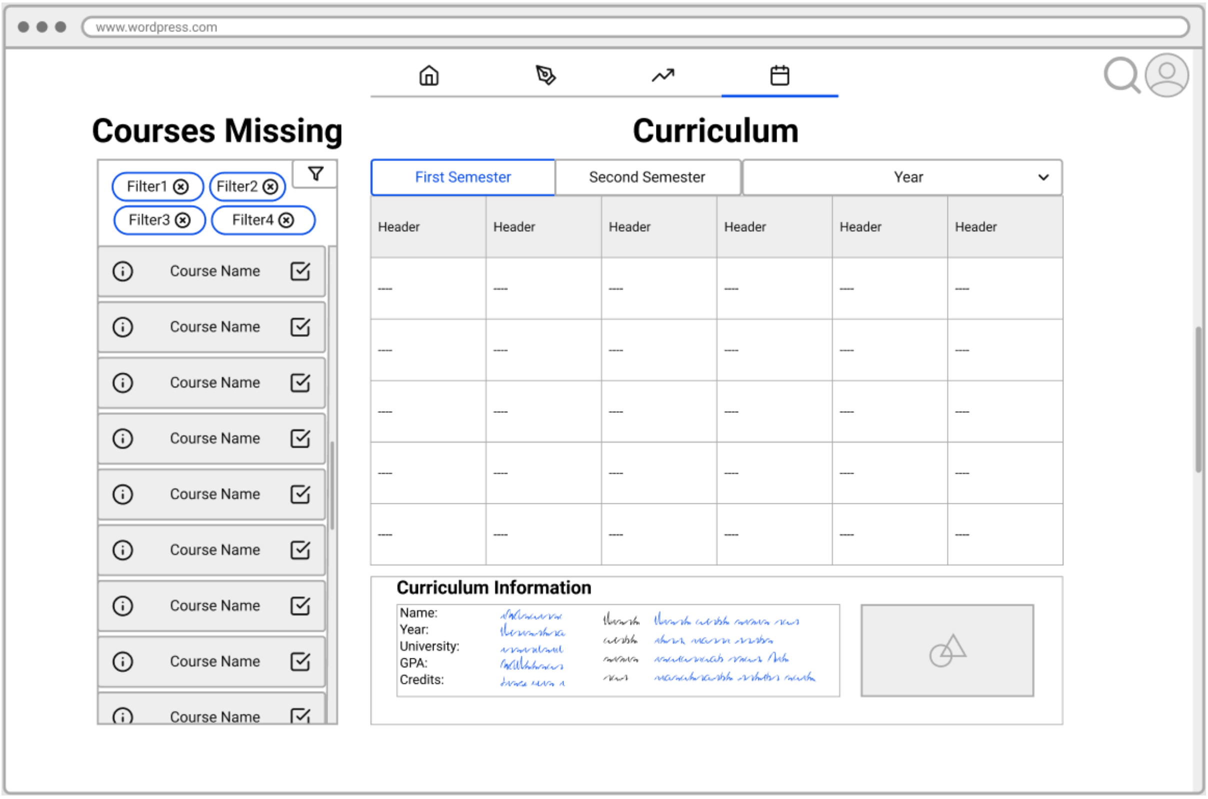
This is a view of the course feedback page, it allows the student to gauge how a course has been rated in the past and relevant statistics that pertain to it.

Course feedback feed



This is a view of the feedback feed for a given course.

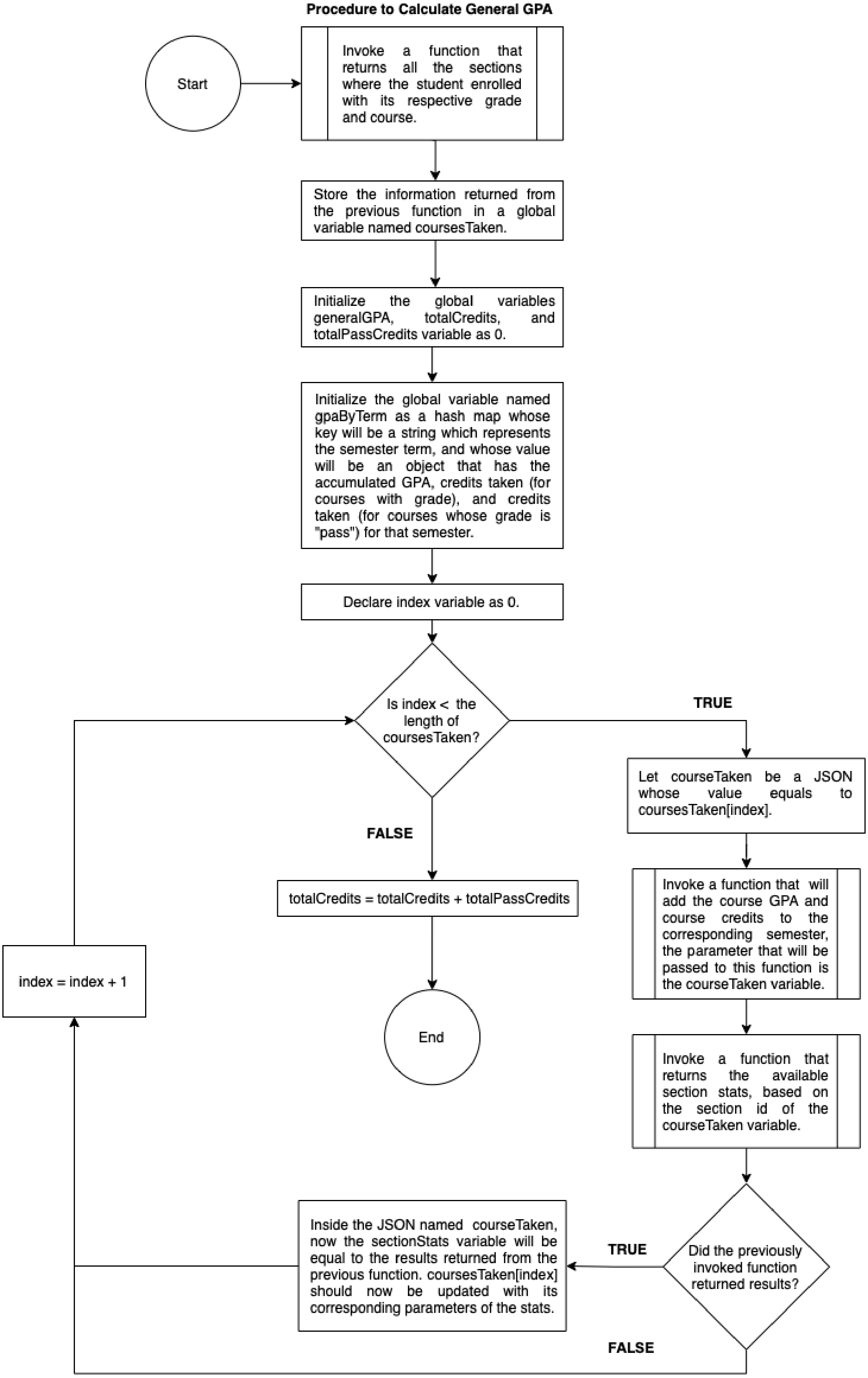
Student curriculum



Header	Header	Header	Header	Header	Header
----	----	----	----	----	----
----	----	----	----	----	----
----	----	----	----	----	----
----	----	----	----	----	----
----	----	----	----	----	----

This is a view of the student curriculum page, it allows the student to see where they are academically.

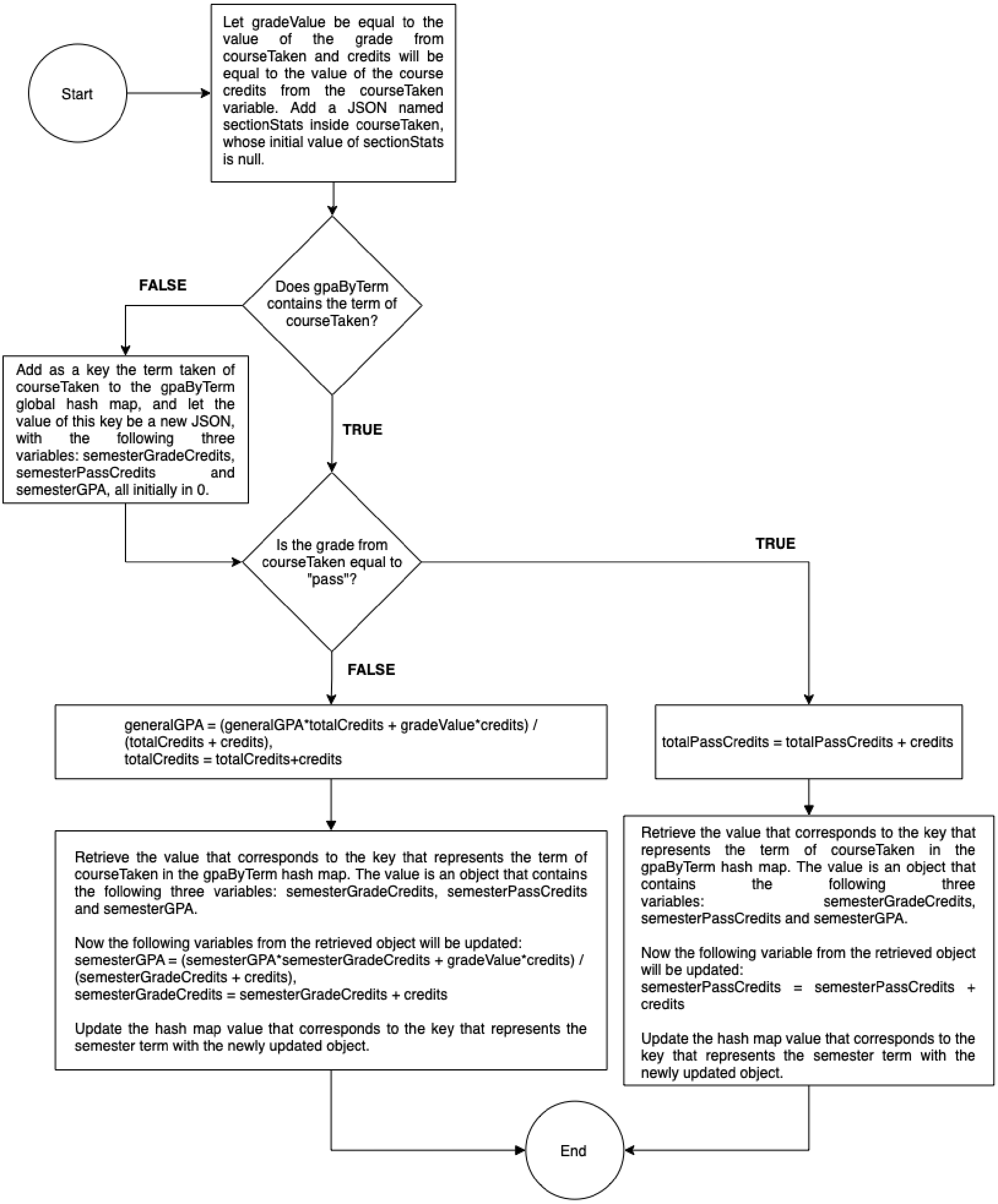
5.5 Describe Flowcharts for algorithms


Procedure to Calculate General GPA
Start
Invoke a function that returns all the sections where the student enrolled with its respective grade and course.
Store the information returned from the previous function in a global variable named coursesTaken.
Initialize the global variables generalGPA, totalCredits, and totalPassCredits variable as 0.
Initialize the global variable named gpaByTerm as a hash map whose key will be a string which represents the semester term, and whose value will be an object that has the accumulated GPA, credits taken (for courses with grade), and credits taken (for courses whose grade is "pass") for that semester.
Declare index variable as 0.
Is index < the length of coursesTaken?
TRUE
Let courseTaken be a JSON whose value equals to coursesTaken[index].
Invoke a function that will add the course GPA and course credits to the corresponding semester, the parameter that will be passed to this function is the courseTaken variable.
Invoke a function that returns the available section stats, based on the section id of the courseTaken variable.
Did the previously invoked function returned results?
TRUE
Inside the JSON named courseTaken, now the sectionStats variable will be equal to the results returned from the previous function. coursesTaken[index] should now be updated with its corresponding parameters of the stats.
FALSE
index = index + 1
FALSE
totalCredits = totalCredits + totalPassCredits
End

Procedure to Calculate General GPA

The previous flowchart describes the algorithm that takes into consideration all the courses that the student has registered in the system, and will calculate the student general GPA based on the amount of credits and honor points accumulated on all the courses. The courses that instead of having a traditional grade, have a “pass” grade will not be taken in consideration for the GPA calculation. Also, additional functions are invoked inside this procedure to calculate the GPA by semester, and to recollect each of the sections statistics. The functions procedures will be seen in the following two flowcharts.

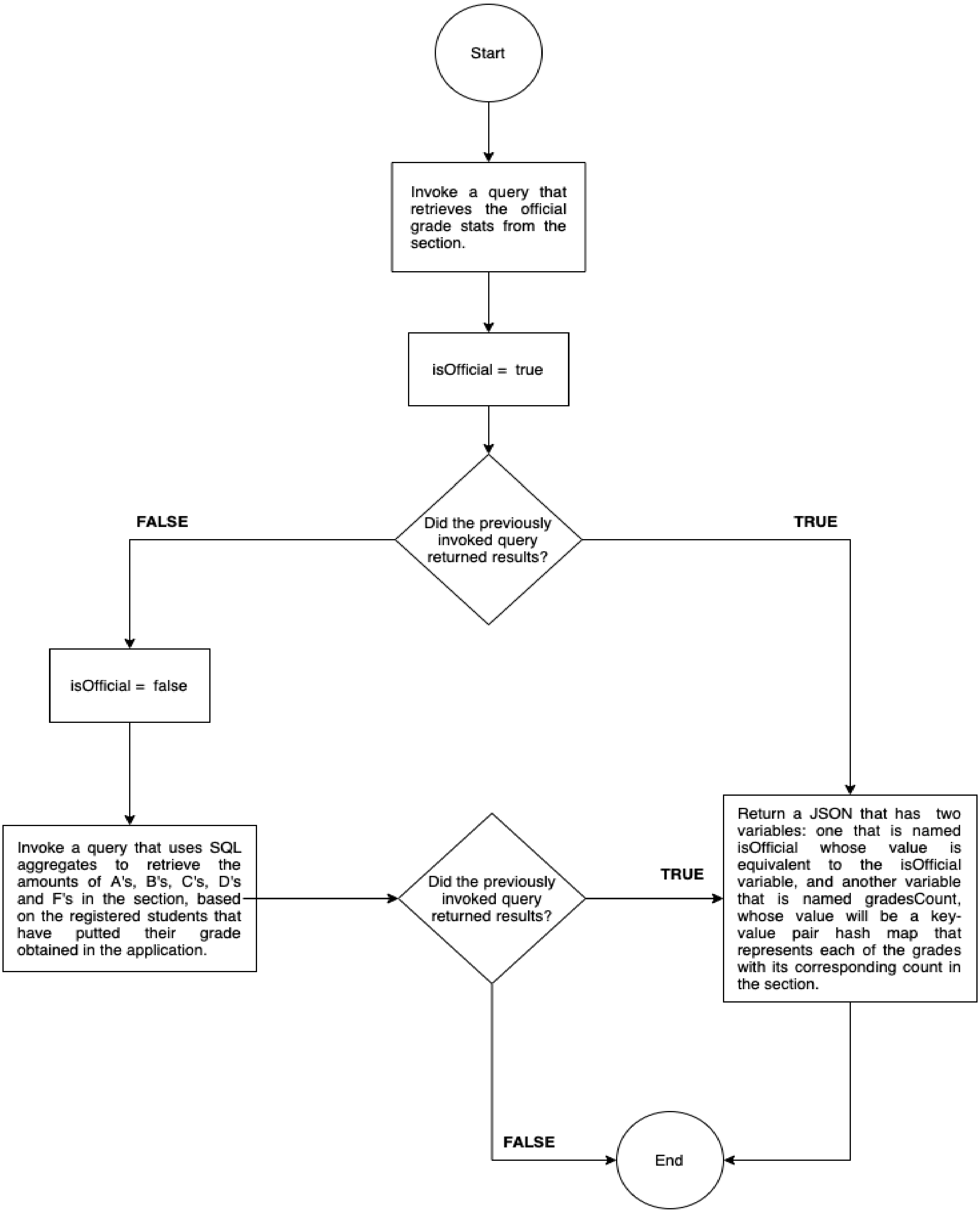
Procedure to Modify Semester GPA


Start
Let gradeValue be equal to the value of the grade from courseTaken and credits will be equal to the value of the course credits from the courseTaken variable. Add a JSON named sectionStats inside courseTaken, whose initial value of sectionStats is null.
Does gpaByTerm contains the term of courseTaken?
FALSE
Add as a key the term taken of courseTaken to the gpaByTerm global hash map, and let the value of this key be a new JSON, with the following three variables: semesterGradeCredits, semesterPassCredits and semesterGPA, all initially in 0.
TRUE
Is the grade from courseTaken equal to "pass"?
TRUE
FALSE
generalGPA = (generalGPA*totalCredits + gradeValue*credits) / (totalCredits + credits),
totalCredits = totalCredits+credits
totalPassCredits = totalPassCredits + credits
Retrieve the value that corresponds to the key that represents the term of courseTaken in the gpaByTerm hash map. The value is an object that contains the following three variables: semesterGradeCredits, semesterPassCredits and semesterGPA.
Now the following variables from the retrieved object will be updated:
semesterGPA = (semesterGPA*semesterGradeCredits + gradeValue*credits) / (semesterGradeCredits + credits),
semesterGradeCredits = semesterGradeCredits + credits
Update the hash map value that corresponds to the key that represents the semester term with the newly updated object.
Retrieve the value that corresponds to the key that represents the term of courseTaken in the gpaByTerm hash map. The value is an object that contains the following three variables: semesterGradeCredits, semesterPassCredits and semesterGPA.
Now the following variable from the retrieved object will be updated:
semesterPassCredits = semesterPassCredits + credits
Update the hash map value that corresponds to the key that represents the semester term with the newly updated object.
End

Procedure to Modify Semester GPA

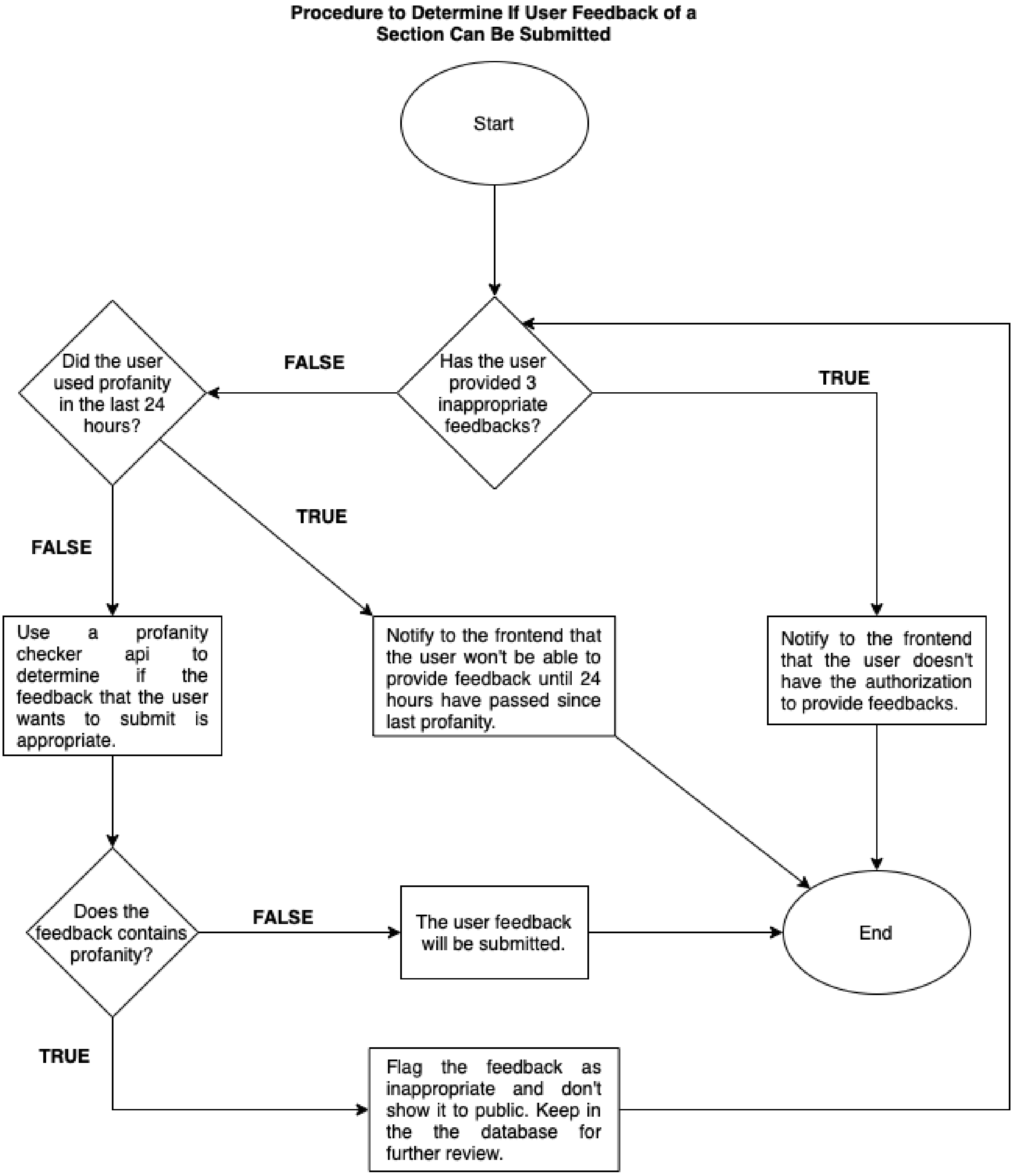
The previous flowchart describes the algorithm that will modify a specific semester GPA, based on the grade obtained on the recently added course for that semester. However, if it detects that the desired course to add to a specific semester will be the first course to be added for that desired semester, then the semester GPA will be initialized as the grade point obtained for that course.

Procedure to Retrieve The Section Stats


Start
Invoke a query that retrieves the official grade stats from the section.
isOfficial = true
Did the previously invoked query returned results?
FALSE
TRUE
isOfficial = false
Invoke a query that uses SQL aggregates to retrieve the amounts of A's, B's, C's, D's and F's in the section, based on the registered students that have putted their grade obtained in the application.
Did the previously invoked query returned results?
TRUE
Return a JSON that has two variables: one that is named isOfficial whose value is equivalent to the isOfficial variable, and another variable that is named gradesCount, whose value will be a key-value pair hash map that represents each of the grades with its corresponding count in the section.
FALSE
End

Procedure to Retrieve The Section Stats

The previous flowchart describes the algorithm that will retrieve stats from a section. It will first see if the official section stats provided by a university is available on the database. If that is not the case, then it will see which registered users on the system have inputted their obtained grade in that specific section and it will take a sampling based on those users. However, if no official stats are available, and the user that is using the application is the only student that registered their grade obtained in that section, then no section stats will be retrieved. Those are the statistics that will be used so that the student can compare their performance with other students.


Procedure to Determine If User Feedback of a Section Can Be Submitted
Start
Has the user provided 3 inappropriate feedbacks?
FALSE
TRUE
Did the user used profanity in the last 24 hours?
FALSE
TRUE
Use a profanity checker api to determine if the feedback that the user wants to submit is appropriate.
Notify to the frontend that the user won't be able to provide feedback until 24 hours have passed since last profanity.
Notify to the frontend that the user doesn't have the authorization to provide feedbacks.
Does the feedback contains profanity?
FALSE
The user feedback will be submitted.
End
TRUE
Flag the feedback as inappropriate and don't show it to public. Keep in the the database for further review.

Procedure to Determine User Feedback of a Section Can Be Submitted

The previous flowchart describes the algorithm that will determine if the recently provided feedback from a user can be published. Basically, if the user has used profanity at least three times, they won't be able to provide feedback. If they have tried to use profanity in the last 24 hours, they can't provide feedback until 24 hours have passed since the last offense. If the previous constraints aren't met, the feedback that the user will be trying to provide will be checked using a profanity verifier api. If the feedback passes this filter, it will be submitted, otherwise, the feedback will be flagged as inappropriate and they won't be able to provide feedback in 24 hours. However, even if the feedback passes the profanity verifier api filter, it doesn't necessarily imply that the feedback doesn't contain profanity. Hence, if a user reports a feedback that contains profanity or if we detect profanity in a feedback manually, the same penalties will be applied.

5.6 Identify realistic engineering constraints

We can anticipate that one of the constraints that will play a big role on this project would be the data extraction. Multiple universities will be involved, and inside those universities, there are multiple departments, professors and curriculums. Hence, thousands of records need to be added to the database in order for the application to work properly. Multiple additions can be automatized if we parse the strings that correspond to each record from multiple sources, such as universities courses, sections and grade stats. However, we don't know how to write a script that automatizes such tasks. This brings us to another point, which is the fact that the time is considered one of the biggest constraints, given that the project must be delivered by the beginning of may. In order to offset the constraint of time and data extraction, we plan to limit our scope to the UPR system in this project.

5.7 Describe engineering standards to be used

- ISO/IEC/IEEE 24774-2021 - Standard for Systems and software engineering — Life cycle management
    - Aims to provide a process for creating a software project life cycle.
    - Generic activities can be established in a process, such as preparing software system design and establishing designs related to each software system element.
    - Will be used due to the fact that the software is part of a larger system.
- IEEE 1012-2016 - Standard for System, Software, and Hardware Verification and Validation
    - Verification and validation processes are used to determine whether the development products of a given activity conform to the requirements of that activity and whether the product satisfies its intended use and user needs.
    - Given that we want to provide the best possible experience to the customers, our intention is to comply with this standard to make sure that the user is getting the experience they deserve.

6 Project Plan

6.1 Describe activities and steps that lead to the completion of the project

We will be working using the Agile methodology, given that this will allow us to work on pre-defined issues established by the team in a meeting. We will be using a sprint of two weeks, however, constant communication between all the members will be an emphasis for the project success.

The project is decomposed on multiple steps:

Step 1: Problem Statement and Project Objective

- In this step, we define the problem that the project focuses to solve with the proposed objective.

Step 2: Technical Description

- A system architecture, modules, ER Diagram, Wireframe for UI, Flowcharts for algorithms, engineering constraints and engineering standards will be defined to facilitate the development process of the project.

Step 3: Technical Implementation

- Based on the proposed technical description, the database tables will be defined according to the ER Diagram. After defining the database tables, we will be working in parallel with the front-end and backend, based on the established Wireframe for the UI and the Flowcharts of the established algorithms. Engineering constraints and standards will be kept in mind during the development. After completing the project, the system architecture will be kept in mind, especially for the deployment process given that we want the application to support load balancing and caching.

6.2 Provide a list of milestones and associated deliverables

Pre-Proposal - 01/25/2022:

- Describe project idea:
  - Motivation
  - Problem statement and challenges
  - Solution approach

Project Proposal - 02/17/2022:

- Contains the following project elements:
  - Introduction
  - Problem statement
  - Project objectives
  - Solution approach
  - Technical description
  - Project plan
  - References

First Project Implementation Progress - 03/10/2022:

- The most essential functionality of the project must be implemented

Progress Report - 03/17/2022:

- Reviews in the proposal must be addressed
- Design elements must be updated
- Preliminary performance must be discussed
- Project must have a web page and code repository

Second Project Implementation Progress - 04/29/2022:

- Bugs from the previous phase must be fixed, minor and major components must already be established in the project
- The project should already be available for public use

Final Report - 05/06/2022:

- Updated design elements
- Performance results

Final Demo - 05/06/2022:

- Fifteen minute demo will be provided via videoconference
- The following items must be available in the project web page:
  - Powerpoint presentation for the demo
  - Elevator pitch video

6.3 Define the specific role of team members

The team members of this project are Félix Dasta, Armando León and Emmanuel Hernandez, and all of them will be working as full stack developers. Each team member will be involved in the whole software development life cycle.

7 References

1. Managers.org.uk. 2022. *Setting smart objectives - chartered management institute. Chartered Management Institute*. [online] Available at: <https://www.managers.org.uk/wp-content/uploads/2020/03/CHK-231-Setting_Smart_Objectives.pdf> [Accessed 10 February 2022].Managers.org.uk. 2022. *Setting smart objectives - chartered management institute. Chartered Management Institute*. [online] Available at: <https://www.managers.org.uk/wp-content/uploads/2020/03/CHK-231-Setting_Smart_Objectives.pdf> [Accessed 10 February 2022].

2. En.wikipedia.org. n.d. *IEEE Standards Association - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/IEEE_Standards_Association#Standards> [Accessed 15 February 2022].

3. Ieeexplore.ieee.org. 2017. *1012-2016 - IEEE Standard for System, Software, and Hardware Verification and Validation - Redline*. [online] Available at: <https://ieeexplore.ieee.org/document/8356723> [Accessed 15 February 2022].

4. Cdn.standards.iteh.ai. 2021. *iTeh Standards*. [online] Available at: <https://cdn.standards.iteh.ai/samples/78981/5c068220670742f98a48d5cd7ad4d050/ISO-IEC-IEEE-24774-2021.pdf> [Accessed 15 February 2022].

5. YellowAnt. 2022. *ORM: Rethinking Data as Objects*. [online] Available at: <https://blog.yellowant.com/orm-rethinking-data-as-objects-8ddaa43b1410> [Accessed 16 February 2022].

6. Amazon Web Services, Inc. 2022. *What is Caching and How it Works | AWS*. [online] Available at: <https://aws.amazon.com/caching/> [Accessed 17 February 2022].