

UPR Grader Project Phase 2

ICOM 4009/INSO 4101: Introduction to Software Engineering

Prof. Marko Schütz-Schmuck

Team: DLMS

I. Informative Part

Group Members:

José F. Rivera Rivera

Jaime L. Miranda Ramírez

Joey R. Hernández Perez

Génesis M. Torres Pinto

Gabriela Cardona Blas

Darielis M. Morales Rodríguez

Group Name: DLMS

Location: University of Puerto Rico at Mayagüez

Date: January 2021 – onward

Partners:

- **Front-end developers:** Front-end web development is the practice of converting data to a graphical interface so that users can view and interact with that data.
 - Jaime L. Miranda Ramírez
 - Génesis M. Torres Pinto
 - Gabriela Cardona Blas
- **Back-end developers:** A back-end web developer is responsible for server-side web application logic and integration of the work front-end developers do. Back-end developers usually write the web services and APIs used by front-end developers and mobile application developers.

- José F. Rivera Rivera
- Joey R. Hernández Perez
- Darielis M. Morales Rodríguez

Current Situation

Any UPR student that wishes to know their current GPA needs to access their respective campus's server through the SSH protocol in some platform like Putty. Students like to view their GPA for many reasons that include knowing if they qualify for federal aid, updating their resume, and/or applying for internships, graduate programs, or scholarships. Through this system, you can see information like the enrolled classes, the GPA, the classes taken or to take and several other functions supported like class enrollment. Although the University provides these functions, the system is usually quite unstable and is one in which you sometimes must wait a long time for maintenance/updates among other things to be able to observe your GPA at the end of a semester. At the moment there is no platform where you can place your grades by yourself and get the GPA either for a specific semester or the general one. These delays and inconvenients can cause students to miss opportunities and be misinformed about their current standing as a student.

Needs

There is a need to optimize and provide an efficient service to the university's student community for accessing information relevant to their respective courses. Essentially, a need for an efficient and easy-to-use service without the need to install complicated third-party software, navigate through an SSH server, or relying on the assistance of University staff like department counselors. The aforementioned processes waste both students and University staff valuable time, and can cause unnecessary traffic on the University's servers.

Ideas

The main idea is based on meeting the needs of the university's student community and in turn making this procedure more bearable. We are going to develop a web-based application where students can "log in" with a username and password so that they can perform various

functions in a faster and more accessible way. The students will be able to choose their curriculum from a list so that they can see all the courses they need to take while having the option to create their own semester by manually adding the different courses they complete. Students will be able to see what classes they already have taken, and which ones they are missing, they will be able to observe and add grades to their classes and in this way be able to see their GPA for a particular semester, general GPA, or major GPA. Also, they will be able to see UPR Campuses where these courses were taken.

Scope

Meet the needs of the university's student community in a fast, effective and user-friendly way. Also, improve the overall experience in the process for all the university's student community.

Span

To improve the process by making it a user-friendly experience and less-time consuming process for all the university's student community.

Workflow

The team follows an agile development process, where we iterate through each of the steps of the software development process every two weeks. The implementation workflow is as follows. Every requirement will have an issue created on GitHub respective to that requirement. A project was created on the GitHub repository, following a kanban template. On the project kanban, different cards are presented, such as Backlog, This Sprint, In progress, and Done. Initially, all the issues created were put in the backlog, and as a sprint begins, issues were moved from the project backlog to the This Sprint card. When a developer starts working on an issue they would move the issue to the In Progress card. Also, for every issue, the developer that will be working on the issue creates a new branch so that the main branch is always up and running. Once the developer finalizes the feature, they need to create a Pull Request and assign other team members as reviewers so that the majority of the team approves the changes. After the Pull Request is approved, the branch is merged to main, and the issue is closed and moved to the done card.

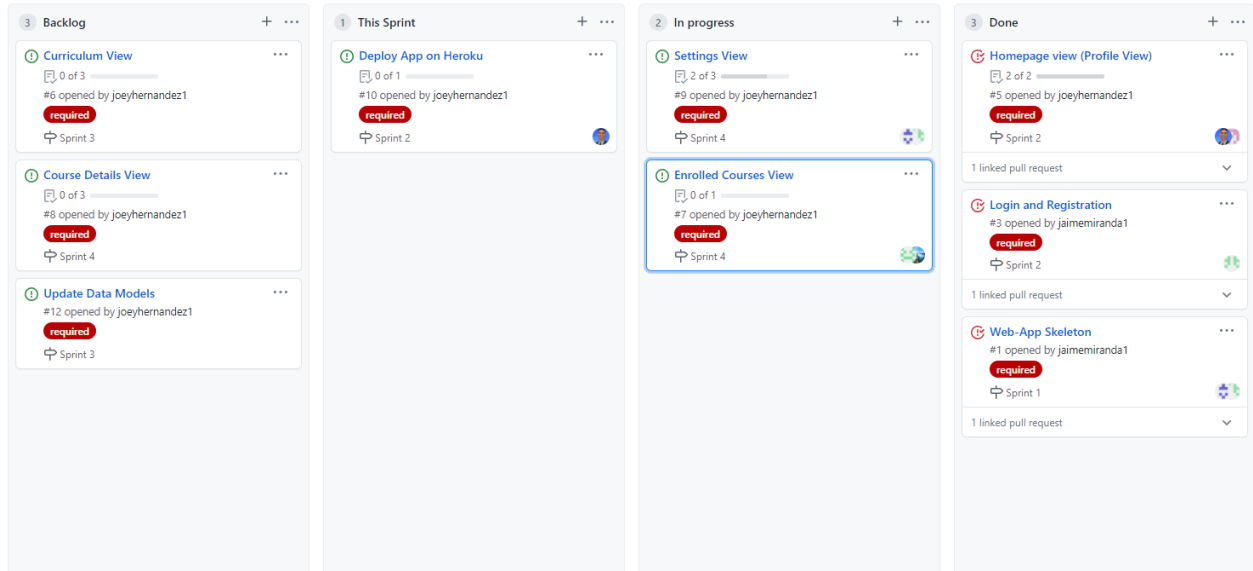


Image of kanban board being used for this sprint

Synopsis

Currently, a system exists that is used by students in order to access information about the courses in their curriculum. We are working in a space that only allows access to the information one page at a time through an SSH client. When you have an archaic, slow and quite unstable system, the user experience during the process becomes a bad and undesirable one. For this reason, there is a need to optimize the system, improving and providing an efficient service for the university's student community. In our case, it will provide a better experience through a web-based application. The website will allow the student to choose a curriculum so they can keep track of their academic progress. Also, the website will facilitate the process of showing students their GPA, either for a specific semester, the general one, or major GPA. Also, it will be able to post your grades to the corresponding classes without having to wait for a system to place it for you long after you have finished your course or without having periods that deny access to the information. On the other hand, you will be able to know which classes you have and have not taken as long as the student keeps updating the web-page. The software architecture for this web-page will be based on python, django, and postgresSQL. A way to verify the proper functionality of the system could be that a student can, for example, login and logout, pick his academic program, among others functions.

II. Descriptive Part

Rough Domain Sketch of UPR GPA and Class Grades Viewer

Students from all of the University of Puerto Rico campuses have complained when using Putty to access valuable information from the university's server. Some of these tasks include: student enrollment, viewing their enrolled classes and the information pertaining to these enrolled classes, viewing their GPA and classes taken, or to take, and many more. Now, the problem may not be necessarily Putty, but at times the system is under maintenance, or when student enrollment time comes, it is under heavy usage and it can be real slow or even crash many times. For a student that just wants to view their current GPA, courses enrolled, or courses taken and their respective grades, this might be a problem or just too much trouble. The current process for a student to view their GPA has too many steps in students' opinions. They would need to use Putty, or an equivalent program, to access the university's server, and log in as a student. Then, a selection would need to be made depending on the student's particular need at the time. In the case of viewing the GPA, the student selects "View other information". Once view other information is selected, the student would select the curriculum option. After pressing this option, the student enters personal information that includes student ID, permanent access code, social security number, and their birth date. If the information entered is correct, the student will see their curriculum courses and the grades obtained for the courses that have been taken. More information about credits completed, credits missing, general GPA, major GPA, and more, can also be found in that screen.

While brainstorming on what falls under our domain some key aspects came up. Currently, students can perform different actions such as:

- A. Searching for their GPA.
- B. Searching for grades on particular classes.
- C. View their curriculum with taken courses and courses to be taken.
- D. UPR Campuses where these courses were taken.

Domain Narrative

Every University uses some kind of metric to evaluate their students. At the University of Puerto Rico the Grade-Point Average (GPA) 4.00 scale is used for those purposes. What this means is that the highest GPA possible for a student to have is 4.0. Each student will have their

own GPA based on the grades obtained in the courses they have taken. The grades that are used in the UPR system are A, B, C, D, and F. In order for students to view their current GPA, they need to either order a transcript from the Registrar's office, remember the grades obtained from previous courses and calculate it themselves, or go into Putty and enter the SSH command followed by the university's respective campus server address. From there, students can do all sorts of things; including viewing their GPA and their grades. Depending on the UPR campus, students can be enrolled in undergraduate or graduate programs.

Domain Terminology

Entities

1. A *University* is a higher education and research institution in which students receive academic degrees from the university in different academic disciplines. A university can have different campuses.

Type: University

2. A *Student* is the entity enrolled in some academic program offered by the university. The student can be considered an undergraduate or graduate depending on the academic program in which the student is currently enrolled. This entity is capable of looking into its current GPA, list of courses taken with their corresponding credit quantity and grade obtained, calculating possible GPAs by adding possible courses and their respective outcomes, and updating their GPA by adding new courses and their final outcome.

Type: Student

3. An *Academic Program* is defined as a combination of requirements and courses leading to a degree to be obtained by the student. Each academic program is composed of Major courses and General courses.

Type: Academic_Program

4. An *Undergraduate Student* is a student that is enrolled in a higher education degree program, either a Bachelor's degree program or an Associate's degree program.
5. A *Graduate Student* is someone who has previously completed a Bachelor's degree program and is currently enrolled in a, more specialized, graduate program.
6. The *Grade Point Average (GPA)* is a number that represents the students performance on the courses the student has taken. The GPA number ranges from 0.00 to 4.00, where 4.00 is the highest score to achieve. This number is calculated using the number of credits

corresponding to the approved courses taken by the student and the honor points from the approved courses resulting grades. There are two types of GPA, general GPA and major GPA.

7. *Major GPA* corresponds to the GPA calculated using only the approved courses that correspond to the student's major.
8. *General GPA* corresponds to the GPA calculated using all the courses approved by the student.
9. A *Credit* is the unit of time for a course where the total amount of credits for a course correspond to the total number of contact hours taken in class.
10. A *Grade* corresponds to the letter grade obtained at the end of the semester for a respective course. Grades translate into honor points which are then used to calculate the student's GPA. The letter grades are A, B, C, D, and F. Each grade has a corresponding weight associated with it (4, 3, 2, 1, 0), respectively, which will be later used to calculate honor points.
11. Honor points are the result of multiplying the weight corresponding to the course's final grade and the amount of credits of the respective course.
12. A *course* corresponds to a class to be taken by the student. Each course has a specific ID formed by concatenating the department's ID and the course code, a specific number of credits, and results in a grade for the student.

Type: Course

13. *Courses Taken* are courses that a specific student has taken. This entity includes course details, grade obtained, and term taken.

Type: List of courses

14. A *department* offers different academic programs that are related to a topic or domain.
15. A *semester* is a half-year term in university, typically lasting fifteen to eighteen weeks.

Functions and Behaviors:

Function:

- add_to_courses_taken(Student, Course, Grade): Student
- get_courses_to_take(Student) : Set Courses
- get_courses_taken(Student) : Set Courses
- get_academic_program(Student): Academic Program

- `get_program_courses(Academic Program): Set Courses`
- `calculate_gpa(Set courses_taken): Float`
- `calculate_major_gpa(Set courses_taken): Float`
- `get_major_courses_taken(Student): Set Courses`
- `get_credits(Set Courses): Integer`
- `get_missing_credits(Student): Integer`
- `get_semester_courses(Set Courses, Semester): Set Courses`
- `separate_courses_by_semester(Set Courses): Dictionary{Semester : Set Courses}`

Behavior:

- **GPA Calculation:** Performs `get_courses_taken(Student)` action and applies the `calculate_gpa` action to the set of courses returned by the `coursesTaken` function.
- **Major GPA Calculation:** Performs `get_major_courses_taken(Student)` action and applies the `calculate_major_gpa` action to the set of courses returned by the `get_major_courses_taken` function.
- **View Courses By Semester:** Perform `get_academic_program` and use its return value to perform `get_program_courses`, and finally the returned set of courses is used to perform `separate_courses_by_semester`.
- **View Courses Taken By Semester:** Perform `get_courses_taken`, then group them by semester using the `separate_courses_by_semester`.

All previous functions and behaviors were derived from the domain description.

Rough sketch requirements as user stories

As a UPR student, I want to be able to look for my current GPA without having to go through the problems that using Putty and accessing the university's server might cause. To avoid these problems, I want to be able to see my current GPA instantly without using Putty, and calculate possible GPAs by adding more courses and their respective credits and grades to the GPA calculation. Also, I would want to see the grades obtained in each of the courses previously taken.

Requirements

1. The system must provide means for the system components to be made to interface with domain entities, such as students, courses, academic programs, users, courses enrolled, etc.

2. The system must provide means for capable students to register for and account.
3. The system must provide means for users to sign in once an account was created.
4. The system shall log in users once they have created an account.
5. The system shall give the user the option of keeping their session alive even when exiting the application.
6. If the user decides to keep their session alive, the system shall be able to log in the user when entering the application without any sign in required.
7. The system must provide means for existing users to navigate through the different views of the application.
8. The system must provide a home view where users shall see their name, current general and major GPAs, academic program, and UPR campus.
9. The system must provide a curriculum view where the user shall see all the courses respective to their academic program and add the obtained grade for courses already taken.
10. The system shall update the user's general and/or major GPAs once grades are assigned to a course on the curriculum's view.
11. The system shall provide means to modify interchangeable courses in student curriculums, such as the ones respective to free, socio humanistic, and/or technical electives, depending on the user's academic program.
12. The system must provide an enrolled courses view where the user is able to add their current courses with their respective information, such as, but not limited to, course code, course name, course credits, etc.
13. The system must provide means to modify existing course information on the enrolled courses view, as well as deleting courses entirely.
14. The system must provide a settings view where users are able to modify account information such as UPR campus and academic program, and delete their account.
15. The system must provide means for users to log out.

Machine Requirements

Some of the requirements required for the machine to have are:

1. The machine must provide one of the latest versions of one of the main operating systems (Windows, MacOS, Linux, etc)

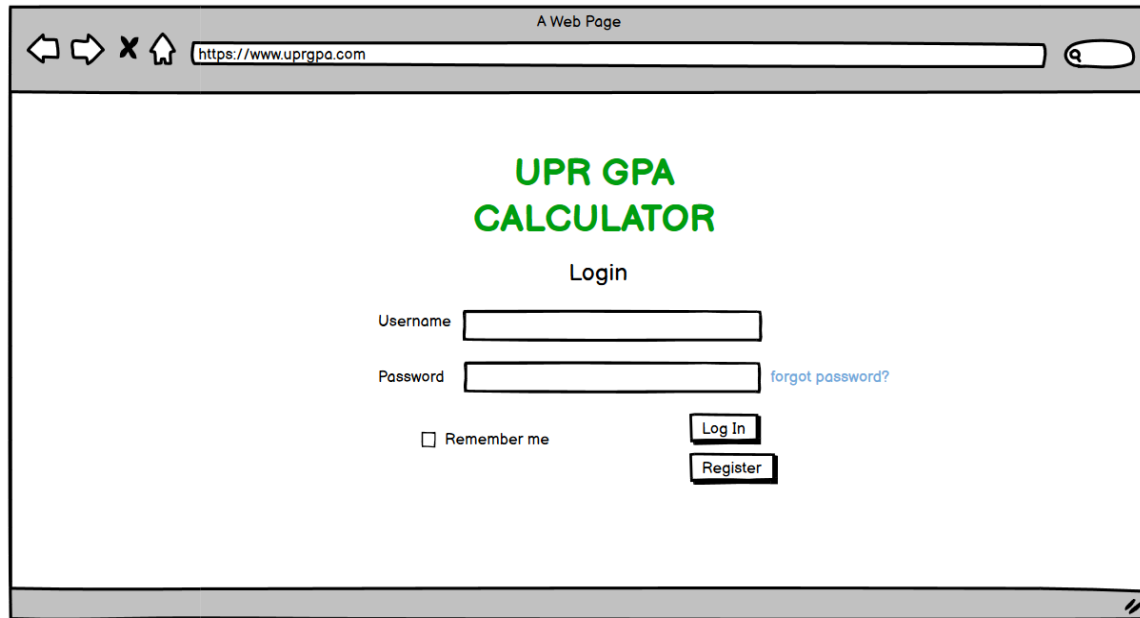
2. The machine must provide means to access the internet.
3. The machine must provide means to run Python, Django, PostgreSQL, Heroku CLI, and Git.

Software Design

a. Software Architecture

- i. The aim of the software to be developed is to solve the problem that arises from students having to use Putty and connect to the university's server to view their GPA. With that in mind, the team has decided to build a Web application that will allow students to view their GPA, add courses and their respective grades, view their grades obtained from added courses, and basically view their academic program's curriculum graded as they take courses and add them to their website account.
- ii. To that aim, and given the skills and knowledge that each team member has, the team decided to use the Python programming language and Django as the Web framework. Python was chosen as the programming language since most team members had some knowledge on Python already. On the other hand, Django offered some great documentation and tutorials, while also a team member had some previous experience using the tool as well. This Web framework provided some tutorials on connecting with databases as well, and since the system to be built needs to store user data, it only boosted its chances to get used. Since user data needs to be stored, a database management system needed to be added to our software architecture. To that end, the team decided to use PostgreSQL, since some members of the team had already used the mentioned DBMS. A question arises, and is that where does the Web application store user data. The most viable solution is using serverless computing, in which the cloud provider runs the server and manages the server resource allocation. The cloud provider to be used has not yet been determined.
- iii. Components for the software architecture include:
 1. Views: Python function that takes a Web request and returns a Web response. The view itself contains whatever arbitrary logic is necessary to return that response. These views encompass all the requirements in which

the system needs a user to see something, such as the enrolled courses view, curriculum view, home view, and settings view. Each view is represented in Django as a function written in the views.py file, and returns the render request of the view. UI sketches for the views are shown below.



A Web Page

https://www.uprgpa.com

UPR GPA CALCULATOR

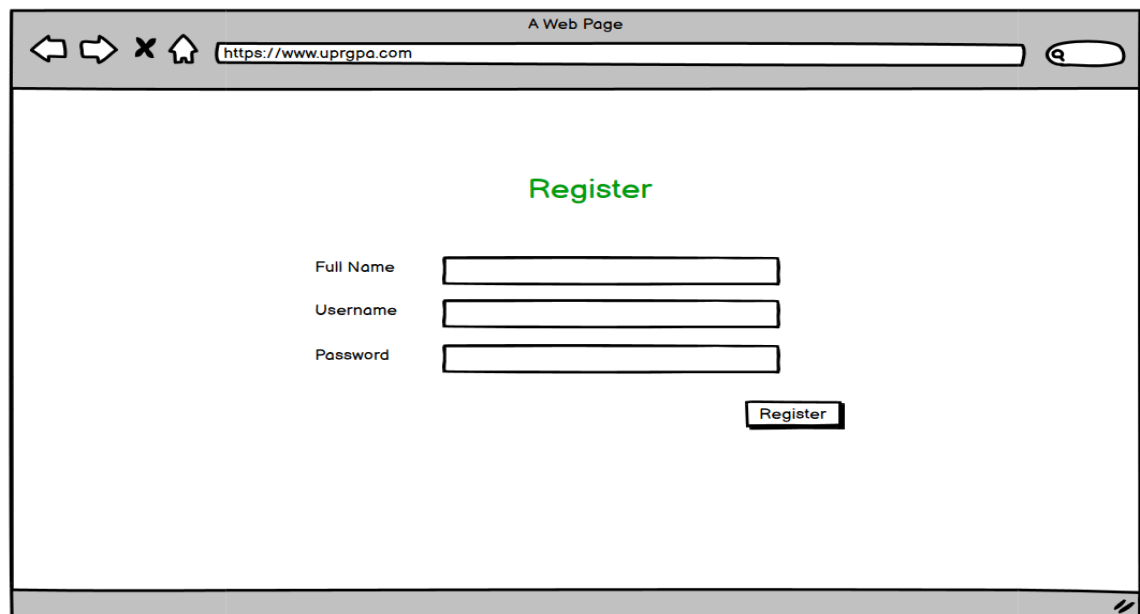
Login

Username

Password [forgot password?](#)

☐ Remember me

Image 1. Login View - User logs in if account has been previously created, if not, selects register to create a new account.



A Web Page

https://www.uprgpa.com

Register

Full Name

Username

Password

Image 2. Register View - User creates new account to be able to use the Web-Application.

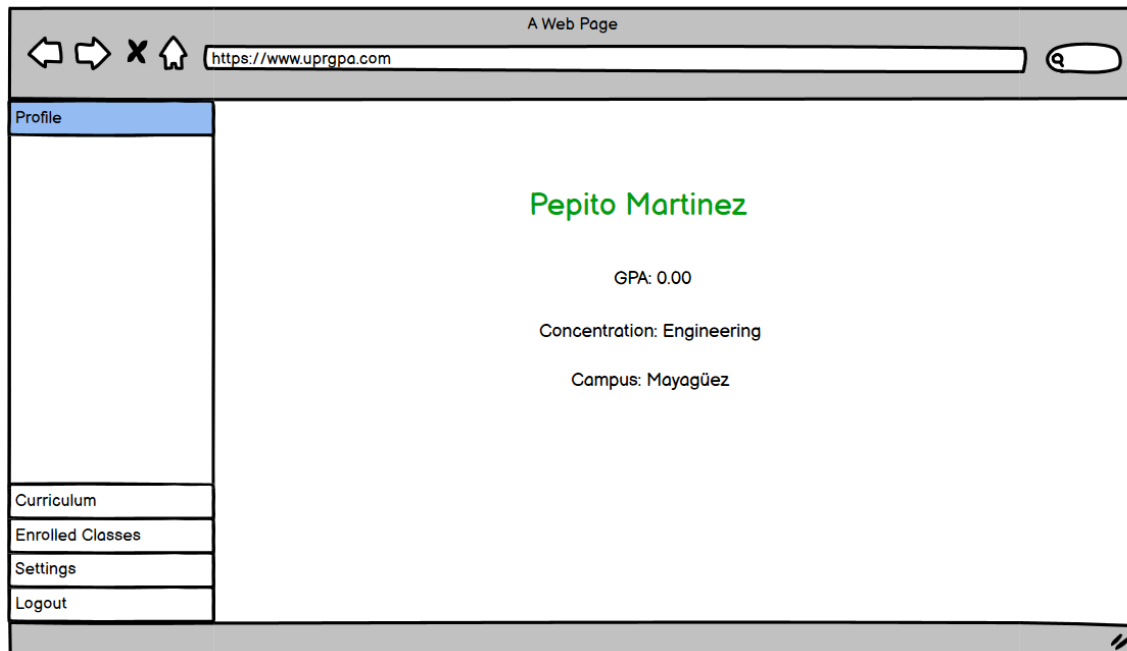


Image 3. Profile View - User sees details about his profile.

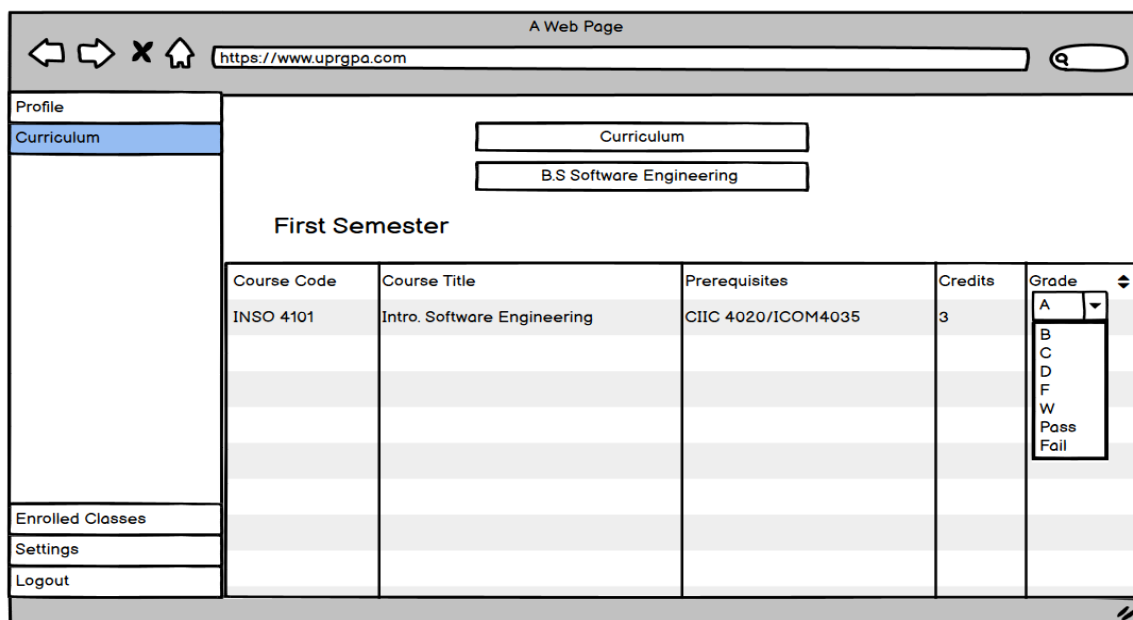


Image 4. Curriculum View - Users add, modify, delete, and see their courses taken and their resulting grade.

A Web Page

https://www.uprgpa.com

Profile

Curriculum

Enrolled Classes

Enrolled Classes Fall Semester 2021

Course Code	Course Title	Credits
INSO 4101	Intro. Software Engineering	3

Settings

Logout

Image 5. Enrolled Courses View - Users can see, add, modify, and delete courses currently enrolled in.

A Web Page

https://www.uprgpa.com

Profile

Curriculum

Enrolled Classes

INSO 4101

Exams	Projects	Assistance	Quizzes
1 A	1 A	1 Present	1 A
2 B	2 B	2 Late	2 B
3 C	3 C	3 Absent	3 C
4 D	4 D		4 D

Settings

Logout

Image 6. Course Details View - Users can see a detailed view of a particular course and add evaluation results obtained in that course.

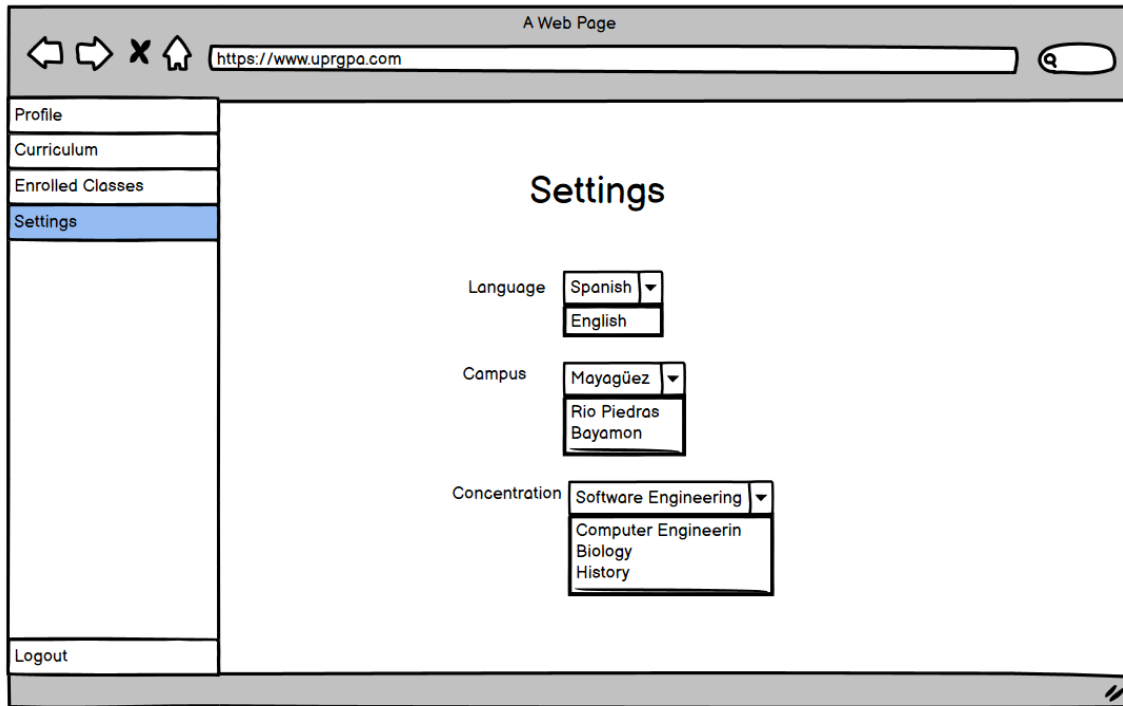


Image 7. Settings View - Users can see current settings and modify them.

2. **Models:** Data models describe entities and relationships to be used throughout the application to store the relevant data pertaining to the system. The diagram below shows the relationships between the different data models used in the system. The current data models include, but are not limited to: auth_user, students, programs, program_courses, courses, student_courses. Each data model can be represented as a table in the database. These models are represented in the Django Framework as classes, and translated by Django's ORM to tables in our PostgreSQL database. These tables were designed with the current domain description, keep in mind that the following tables are updated as the project progresses. The current model relationships allow simple access of the information needed in the application.

public. auth_user		
id	serial	PK
username	char(150)	
first_name	char(150)	
last_name	char(150)	
email	char(150)	
password	char(150)	
groups	ManyToMany	
user_permissions	ManyToMany	
is_staff	boolean	
is_active	boolean	
is_superuser	boolean	
last_login	datetime	
date_joined	datetime	

This table contains all the information related to user authentication. Each user is binded to one student.

public. students		
student_id	serial	PK
student_campus	char(50)	
student_gpa	char(50)	
student_major_gpa	char(50)	
student_user	integer	FK
student_program	char(50)	FK

The Students table represents the student entity. This entity contains fields like their respective campus, major and general GPA, a relation to their specific academic program, and a relation to their user.

public. programs		
program_id	char(50)	PK
program_name	char(150)	
program_courses	integer	
program_credits	integer	
program_department	char(150)	

This table represents the different Academic Program entities available to each student. It contains information like the program name, amount of courses and credits to take, and to which department the program belongs.

public. courses		
course_id	integer	PK
course_credits	integer	
course_name	char(50)	

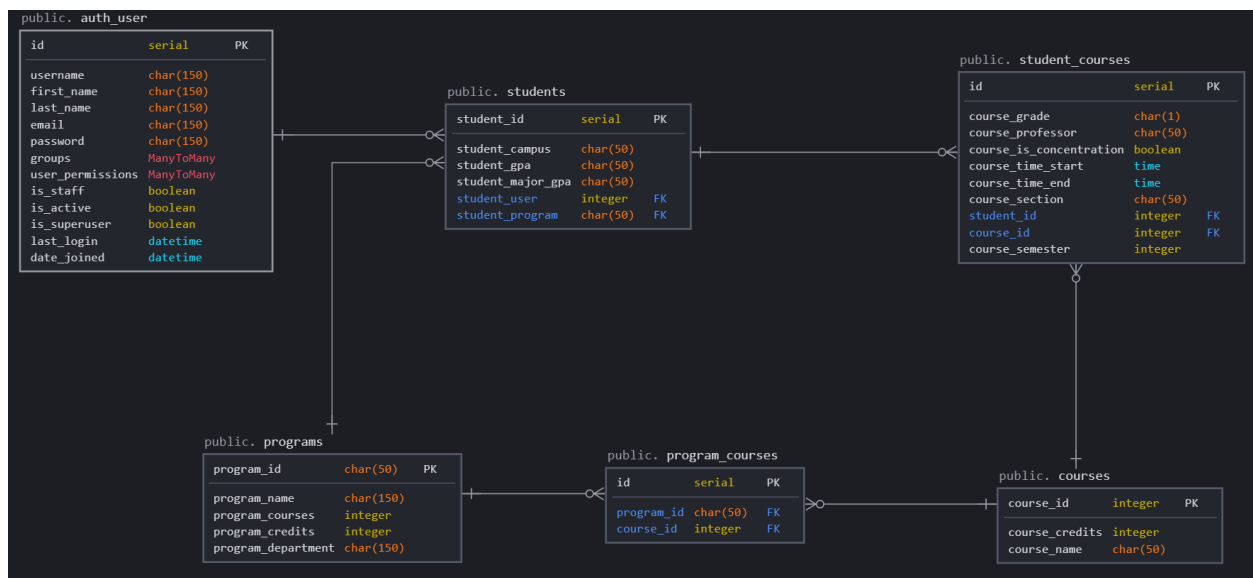
This table contains all the courses offered by the university throughout the academic year for students to take. These courses hold information like the amount of credits and the course name.

public. program_courses		
id	serial	PK
program_id	char(50)	FK
course_id	integer	FK

This table represents the relationship between academic programs and their respective courses.

public. student_courses		
id	serial	PK
course_grade	char(1)	
course_professor	char(50)	
course_is_concentration	boolean	
course_time_start	time	
course_time_end	time	
course_section	char(50)	
student_id	integer	FK
course_id	integer	FK
course_semester	integer	

This table contains information regarding all the courses a student has taken. It represents the relationship between student and course tables, and contains information such as the resulting grade, the professor that taught the course, if the course is a concentration course, start and end time, course section and semester, and the respective course and student ID.



Complete Overview of Data Model Relations

b. Component Design

1. **Views** are held on the views.py. Each view is defined as a method on the file, and it is passed in a parameter for the request. The request parameter will hold information about what the user is doing. For example, below we can see the view

method for the home page. For the homepage, we would want to be able to access it if and only if a user is logged in. On the other hand, the user should be able to logout from any view, including the homepage view. A “POST” method would be sent in the request parameter if the user was in the homepage and clicked on the button to logout. Otherwise, send a dictionary variable (in the example below named context) with all the information needed to be displayed in the UI and redirect to the homepage link.

```
65
66 def home_page(request):
67     if not request.user.is_authenticated:
68         raise Exception(DisallowedRedirect)
69     if request.method == 'POST':
70
71         logout_request = request.POST.get('logout', None)
72         if request.user.is_authenticated and logout_request is not None:
73             logout(request)
74             return redirect('../')
75
76     name = request.user.first_name + ' ' + request.user.last_name
77     context = {'name': name,
78               'campus': request.user.students.student_campus,
79               'program': request.user.students.student_program,
80               'overall_gpa': request.user.students.student_gpa,
81               'major_gpa': request.user.students.student_major_gpa
82               }
83
84     return render(request, 'UPR_Grader/home.html', context)
```

Home View code snippet

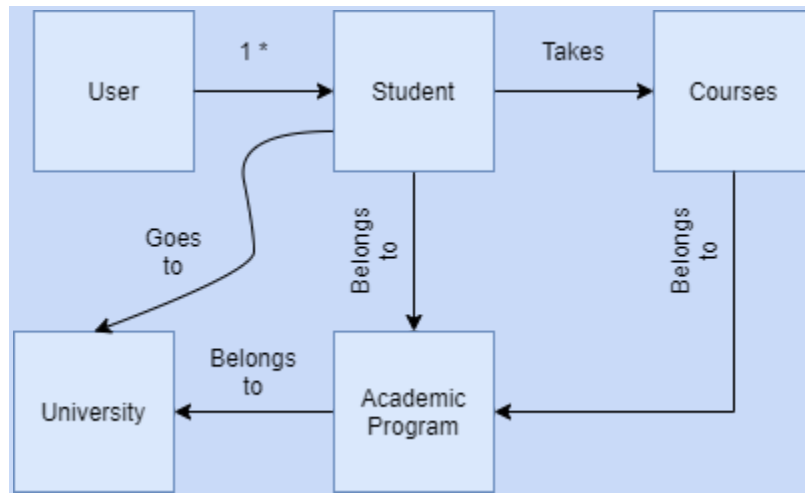
2. Models are held on the models.py file as classes, since as it was explained on the software architecture aspect, they are used by Django's ORM for the backend. All models created must inherit from Django's models.Model. Below, as an example, is the current model created for the academic programs that students are accepted in, which has fields like program name, course total, credits, and department.

```
class Program(models.Model):
    program_name = models.CharField(max_length=100, unique=True)
    program_course_total = models.IntegerField()
    program_credits = models.IntegerField()
    program_department = models.CharField(max_length=100)
```

Program Model code snippet

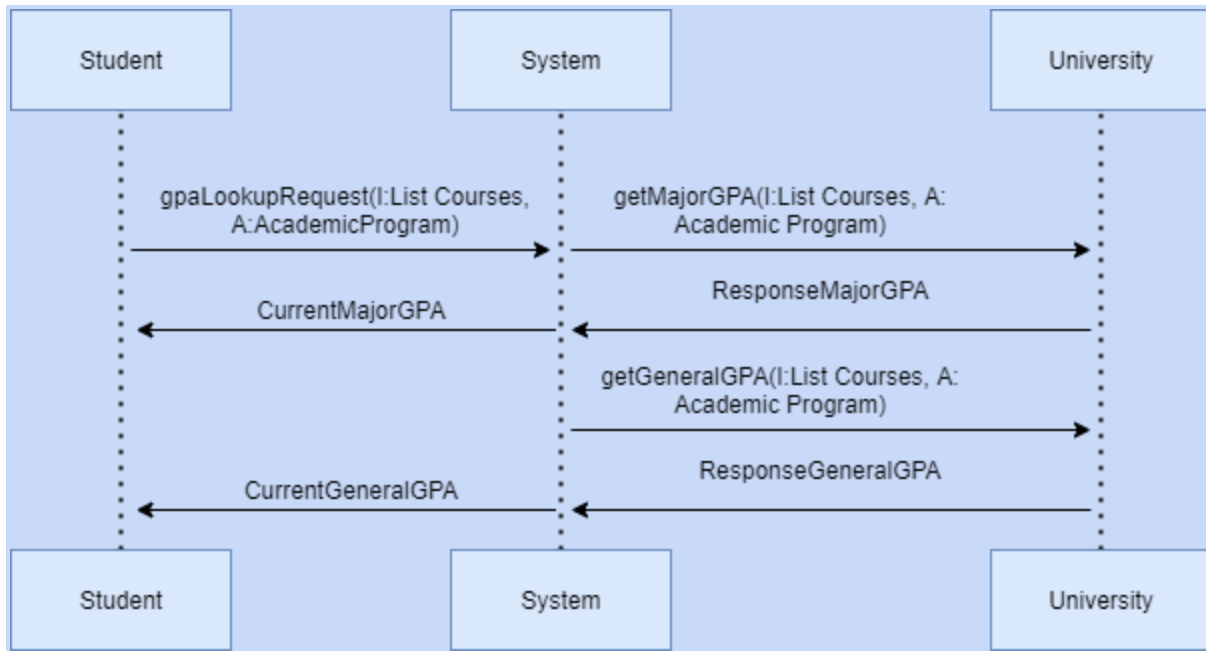
c. Diagrams

1. **Class Diagram:** Below is the class diagram for the system-to-be. It includes the users which are represented by a student that takes courses, goes to a university and belongs to an academic program in such university. Courses belong to an academic program as well.



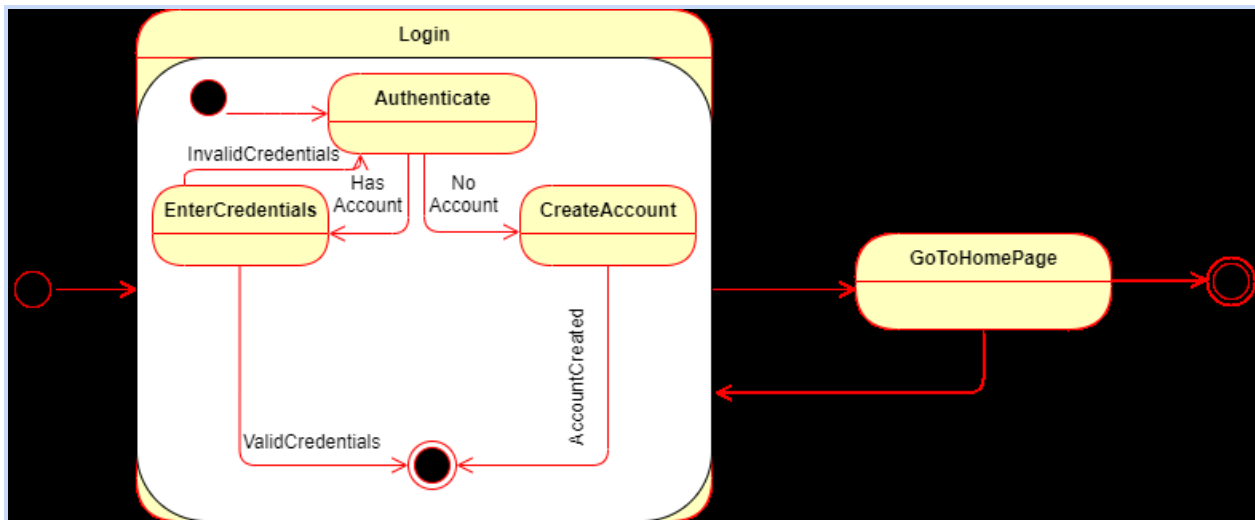
UPR-GRADER Class Diagram

2. **Sequence Diagram:** The diagram below contains the sequence of the student looking for both their major and general GPAs.



Student Major and General GPA Lookup Sequence

3. **State Chart:** The state chart below represents the necessary activity to be done in order to log in into our system.



UPR-GRADER Login State Chart

III. Analytic Part

Concept Analysis

A system that helps students keep track of their classes, grades, and GPA consists of a set of classes that correspond to a specific curriculum, a set of students that each have their own classes taken as well as their GPA, and a set of curriculums that have their own pertaining classes.

Verification

To verify that the team is getting the system-to-be right, the verification process will include different usage scenarios of the features as they are implemented. Usage scenarios will include:

1. Registering User: In this scenario a user must be registered successfully in our database with the credentials inputted by the user in the User Interface. This will be done by registering a user ourselves and looking at the database to see if both the user and student tables were updated with the right information.
2. Logging User: In this scenario a user must use previous credentials used for account creation in order to log in. Users must input the email address and password used when registering the account. This will be checked by registering for an account and then using those credentials to log in.
3. Home Page User Information: In this scenario the user is logged/registered and was redirected to the home page. Here the user should be able to see his name, and default values for overall and GPA, academic program, and campus. From there, the user should be able to logout and navigate to distinct views.
4. Settings View change of defaults: In this scenario the user should be able to change the default values for their campus and academic program and save the new ones. The new saved values should be displayed on the home page view in the next navigation. Also, the user should be able to logout and/or delete their account in this view. The database will be verified manually in order to check if the account was successfully deleted.

5. Enroll Courses View Course Modifications: In this scenario the user should be able to add courses, and modify or delete courses previously added. These courses would then be stored in the database and should be displayed in each user session.
6. Curriculum View Grade Edit: In this scenario the user should see a list of the courses pertaining to their respective academic program, and should be able to add a grade each course. After a grade is added, the respective user's/student's GPA should be updated in the database and displayed when they go to the Home Page View.

For each of these scenarios, the developer that is responsible for the development of the feature, also has the responsibility of running their respective usage scenario and demonstrating it to the team. Once it is demonstrated and each team member is satisfied with their work, the code should be pushed and reviewed before merging with the main branch, and eventually, production.

Validation

For validating the system, the team plans to deploy a beta version of the system in the future. This beta will be used by a limited number of users, in order to gather feedback on the system as a whole, and improve the system from there. Information will be collected from the beta version by having the users complete a questionnaire and/or survey about their overall experience, things that were great, things to improve, features to add, delete or modify, etc. Once in production and deployed formally, users will receive emails at the end of every semester after their account creation containing surveys, in order to gather feedback from users and improve the overall experience. Also, a contact us feature would be implemented where the team could be contacted directly.

Deployment

Deployment was done through a remote server on Heroku. There was a deployment for the Django project, and one for the PostgreSQL database, and later both were connected to each other. The current workflow is such that every time the team finishes a Milestone, or Sprint, the achieved results that were merged on to the main branch, are pushed to the heroku remote, or what we call production, as well. The idea is that we always have a running and working version on production, while development continues.

The working version can be accessed through the following link:
<https://upr-grader.herokuapp.com/>