UPR Enrollment System Project Final Phase Report

INSO 4101: Introduction to Software Engineering
Prof: Marko Schütz-Schmuck
Team: Los Reyes de la Punta
Members: Hermes Colón Rosado, Abdiel Cortés Cortés, Jaime Ginorio Ramírez, Isaac López Díaz, Christian Robles Rodríguez, Fabiola Robles Vega

I. Informative Part

A. Name, Place, Date:

UPR Enrollment System, Mayagüez, Puerto Rico, 27/November/2020

B. Partners:

a. Developers:

i. Front-end Developers:

1. Role: The front-end developers are in charge of the part of the project that runs on the browser. They are responsible for the user-facing part of the website, i.e. the elements the user sees and interacts with.
2. Members:
   a. Fabiola Robles (Front-End Team Leader)
   b. Abdiel Cortés
   c. Christian Robles

ii. Back-End Developers:

1. Role: The back-end developers are in charge of the logic in the web services, cloud infrastructure, Databases and APIs.
2. Members:
   a. Jaime Ginorio (Back-End Team Leader)
   b. Isaac López
   c. Hermes Colón
   d. Christian Robles (Helper Developer)

b. Clients:

i. UPR institution (University of Puerto Rico)
The system proposed in this document will be substituting the current enrollment system that the institution utilizes for course enrollment each semester, that is causing several problems to the institution and its members (later explained in this document).

c. Domain Experts:

i. CTI UPRM personnel:
The CTI employees are the current group in charge of the university's systems infrastructure and are the ones who, currently, deal with the process and protocol of the enrollment. (CTI page)

C. Current Situation:

The current enrolment system consists of an SSH protocol that must be accessed using a terminal, where students navigate through the system by inputting commands in the keyboard. To access the system, third party software must be installed if the device that they're using doesn't come with a terminal with SSH capabilities. The system has constant server issues, it slows down when there is a large amount of students accessing it, and on multiple occasions the server crashes; causing the university to have to reschedule the enrollment dates.

D. Needs

The current enrollment system is outdated and in need of replacement with something that's more reliable and easier to use. Therefore, there is a need to create a new enrollment system that allows students to access it without installing complicated third-party software, that has a simple interface to use, and that is stable enough to withstand the usage demands without slowing down or crashing while under heavy loads. For this a redesign of the architecture is needed in order to provide a smooth and painless enrollment process to the audience, with a plus of a new user interface.

E. Ideas

We will develop a web application where students can log in with their university accounts and enroll in their courses. Students will be able to search for courses by using the course code, course name, or by searching through the department that offers the course. The system will automatically detect if a student has the requirements that a course demands and allows, or denies, them enrollment. The system will use a serverless architecture that ensures stability and speed when facing heavy workloads. With this new design of a serverless architecture we will target the problem of the system constantly crashing due to heavy loads (later explained).

F. Scope

To improve the overall experience in the UPR enrollment process for both, the students, and the administration.

G. Span

To improve the UPR enrollment process by making it a less time-consuming process and providing a user-friendly experience for the students. At the same time, improve the stability of this process so that it can be done on time, avoiding any need of rescheduling the process.

H. Synopsis

The UPR enrollment system constantly fails, causing a bad experience to the students and the administration; most semesters, especially at the beginning, experience a delay because of this system. As part of our domain, we have an institution, in this case the university. This institution is divided into departments with several degrees. A student selects one of those degrees to pursue and in order to achieve the goal, he/she needs to complete the required courses of the degree. In order to take those courses, the students need to enroll in them each semester. There is a need to create a new enrollment system that allows students to access it without installing complicated third-party software and that is stable enough to avoid crashing while under heavy loads. We want to provide the UPR students with a web application that is easy to use for them, and at the same time minimize the constant failures that the current system faces. Our project addresses the need of a renewal in the enrollment system, specifically in terms of a system architecture that supports a heavy user flow without crashing and secondly a new user interface that is understandable to first-time users. The current enrollment system consists of the student login in through the terminal and using a SSH protocol with his/her credentials. After that, the student chooses the option to proceed with the enrollment process, which has many unintuitive features, mainly because it is a command-line-interface application. Besides this, if the students need more information about courses (what prerequisites might the course have) they need to visit another platform. We want to provide an intuitive system that provides the student with every information they might need. Also, it should tolerate a heavy flow of users in order to provide a smooth experience without interruptions, or any need of rescheduling the process. Because of these requirements, we are proposing an easy-to-understand webpage with a serverless cloud infrastructure in order to manage the heavy flow of users without any problems. On this webpage, the student will login with the university credentials and then proceed to an informative page where they will be able to see the availability of next semester's courses, and, at the time to enroll, they will have this option. The enrollment process will be with a search option by course name or codification in an elegant graphical interface, this way the student will be able to enroll the course by just clicking buttons and typing the course name or codification. As part of our requirements, we want a system that is safe and protects the students' information all the way. We will take this in consideration of our implementation and testing will be done to corroborate this and all the other system requirements. Further development could establish and integrate into the system support for students that are not able to find the courses they want to enroll and the waiting list for courses that are already full.

II. Descriptive Part

A. Rough sketch domain description:

a. Concepts forming a Student Enrollment System: University, Credentials (name and lastname, SSN, student ID, PIN, DoB), Student, Semester, Program (bachillerato), Curriculum, Courses (Course Name/Code, # of Credits, Professor, Capacity, Schedule, Pre/Co-requisites), Navigation, Enrollment/Withdrawal, Payment.

b. For students to attend an educational institution they are required to enroll first. This type of process can vary along institutions, some of them prefer to do the enrollment with no student influence at all, others provide an enrollment system so that the students can do the enrollment process by themselves. There are many types of enrollment systems, going all the way from pencil-and-paper to terminal-based to more updated systems, but all of them have the same purpose: give the students some tools to manage their enrollment.
In general, no matter if it is outdated or updated, all the systems wrap around the same steps in order to achieve the enrollment process. First, the systems need to identify the student, which could be done using their name, email, or some sort of username. It is essential for the systems to ensure that the students are who they claim to be, because of this it is important to require some personal information that just the student should know, for example, password, social security number, student number, 4-digit pin, birth date, or any combination of these. Once the students' identities are certified, in older systems, they would proceed straightly to select their classes, in more newer ones, students are probably taken to some place from where they can perform a series of actions depending on what they want to do, and on occasions the courses are also displayed there too. Some of these actions may be enroll a course, drop a course, see the student information, see the schedule, search for available sections of courses, confirm the enrollment, and many others. What the students can or cannot do next depends on the actions available on the screen, which obviously depend on how the systems were built by the institutions. While some of these actions like enroll a course, drop a course, and search for available courses might be essential for the systems, others are just important to guarantee a better overall user experience.
Most of these actions have their rules or limitations, for example, if enrolling a course is the action, what courses can the students enroll depend on their major, which also depends on what department of the institution the major belongs to. Another limitation could be that there is an established date by the institution for the students to enroll or drop a course, and to confirm the enrollment. If this is not achieved everything about the enrollment will be lost.

c. There exists an institution, in this case a university, which is composed of different departments. These departments consist of several degrees available to students to pursue and achieve a professional career. In order to complete a degree, students must approve all the courses the degree contains. These courses, depending on the quantity of material and lecture hours, have a measure of credits. Each student has a maximum load of credits per semester depending on the degree.

d. A course is a collection of lectures and activities where a professor teaches certain topics and skills to an audience of students. The professor is a person who has certain academic preparations and knowledge relating to the subject of the course. A course belongs to a department that relates to the subject taught in that course. Each course has a measure of credits, where the number of credits indicates the amount of hours per week required for that course. A course can contain multiple sections where each section is taught by a professor, in an assigned classroom, at a specific time, on certain days of the week. Attending an in-person course is defined by the students going to the classroom lectures, and in an online course, the professor can decide how they will measure attendance.

e. Enrolling a course is the action of a student adding a course to his/her tuition. Dropping a course is the action of a student removing a course from his/her tuition. Searching for a course is when a student navigates through the course catalog. While inquiring about a course is when a student searches for a course to get information about it.

B. Domain Narrative

Educational institutions may vary between school, universities, college, and many others. All of these institutions are composed by smaller units like departments, which at the same time have majors, and so on. This is done, to keep an order at some level of abstraction by grouping people, students, with the same interests. Each of the majors or programs of the departments consists of a series of courses that will shape the students according to what they are studying. Students have to enroll in order to be part of a program, for this the institutions usually provide an enrollment system, which can vary depending on the institution, and also depending on when the system was built. Although they can vary, the major purpose is always the same, which is to provide the students with some tools so they can enroll successfully.

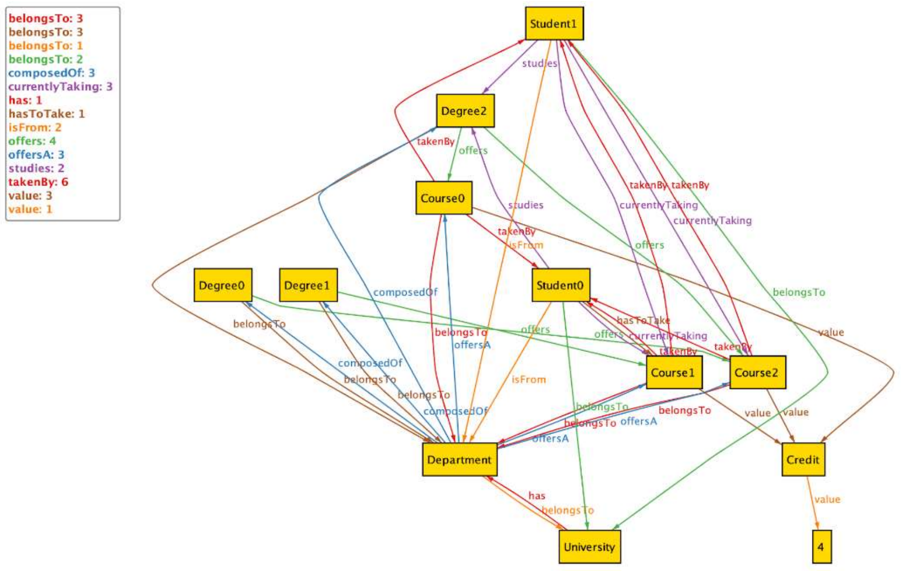
The enrollment systems usually consist of the same basic things, no matter how different they are from one another, the basis is the same. In order to complete the enrollment process successfully, each semester, users (students) have to use the course enrollment system provided by the institution. A course is a collection of lectures and activities where a professor teaches certain topics and skills to an audience of students. Each course has a measure of credits, where the number of credits indicates the amount of hours per week required for that course. Students have a maximum amount of credits per semester, which they need to have in consideration at the time of planning their schedule for next semester. A course can contain multiple sections, of which a student can only enroll one, where each section is taught by a professor, in an assigned classroom, at a specific time, on certain days of the week. A professor is a person who has the academic knowledge and preparation to provide a certain course.

Matriculation is the process of students committing to the University by enrolling in a set of courses and paying a tuition fee to the University. Enrolling courses is the process of a student browsing through the course catalog and selecting which he/her needs according to his/her curriculum. The course catalog is the set of courses being offered in a semester by the University. Dropping from a course is the opposite of enrolling a course, it is the process where a student removes a course that he/she already enrolled in. Meaning that users have to go to the system and login using their institutional credentials in order to receive access. Inside the system can be found many things**.**

The user will be able to search for the desired courses and enroll in them once the enrollment shift begins. There, in the enrollment screen, users can either add or remove courses, from their current enrollment and according to the user's curriculum based on their Program. In this screen can be found the schedule, credits, professor, and codification of the courses currently enrolled, if any. From this screen users can also navigate through courses using its codification or name, for whatever reason they want to. It could be to explore the available sections, the capacity available, which sections are reserved, which professor is giving what course, and many other reasons. It is common that before a semester starts, the student confirms their enrollment and pays the tuition according to the enrolled courses which prices may vary according to the credit value of a course, whether the course is a laboratory, etc.

C. Domain Models

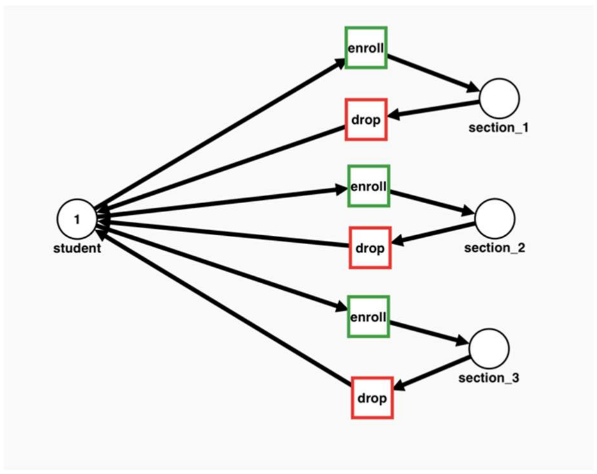
1. <u>Alloy Model</u>:

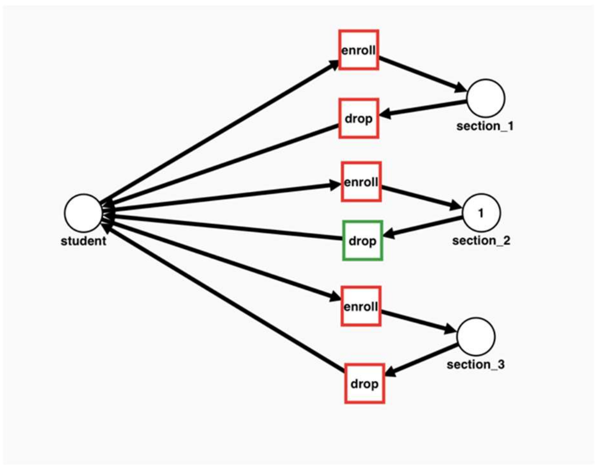


To build this model we used Alloy and followed the idea on how to build models presented by the paper "Alloy as a Language for Domain Modeling" by Asoudeh Nesa and Khosravi Ramtin. This model represents the relationship between a University, a Department, 2 Degrees, 3 Courses (belonging to the same Department but not the same Degrees), a Credit of 4 making it the value of each Course, and 2 Students that belong to the same Department but are pursuing different Degrees. The model describes that the relationship between most is either many-to-one or one-to-many. For example, the relationship between Student and Department is many-to-one since there could be many students belonging to the same Department but a Student belongs to only one Department. In our next phases we should tackle special cases for Students that pursue multiple Degrees making them belong to more than one Department, however, these cases are the minimum.

2. Petri Nets:

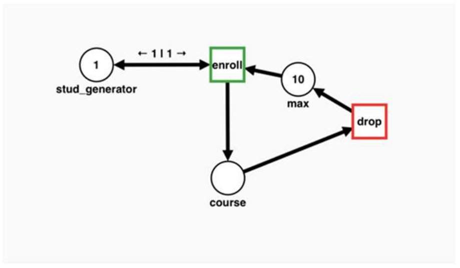
- One Section per Student Petri Net

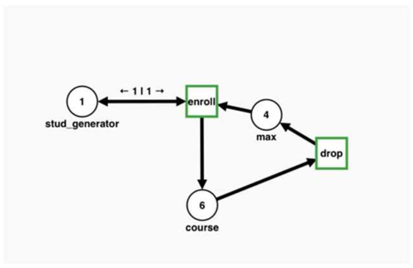


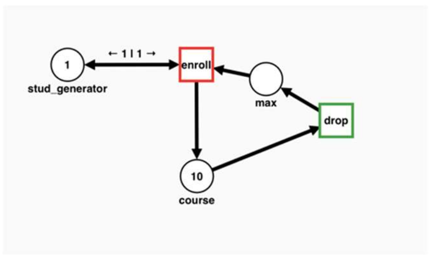


This model represents the process where one student enrolls a course section, and if the student wants to enroll another section of the same course, dropping from the previous section is required. No student can enroll in more than one section from the same course.

- Course Capacity Petri Net



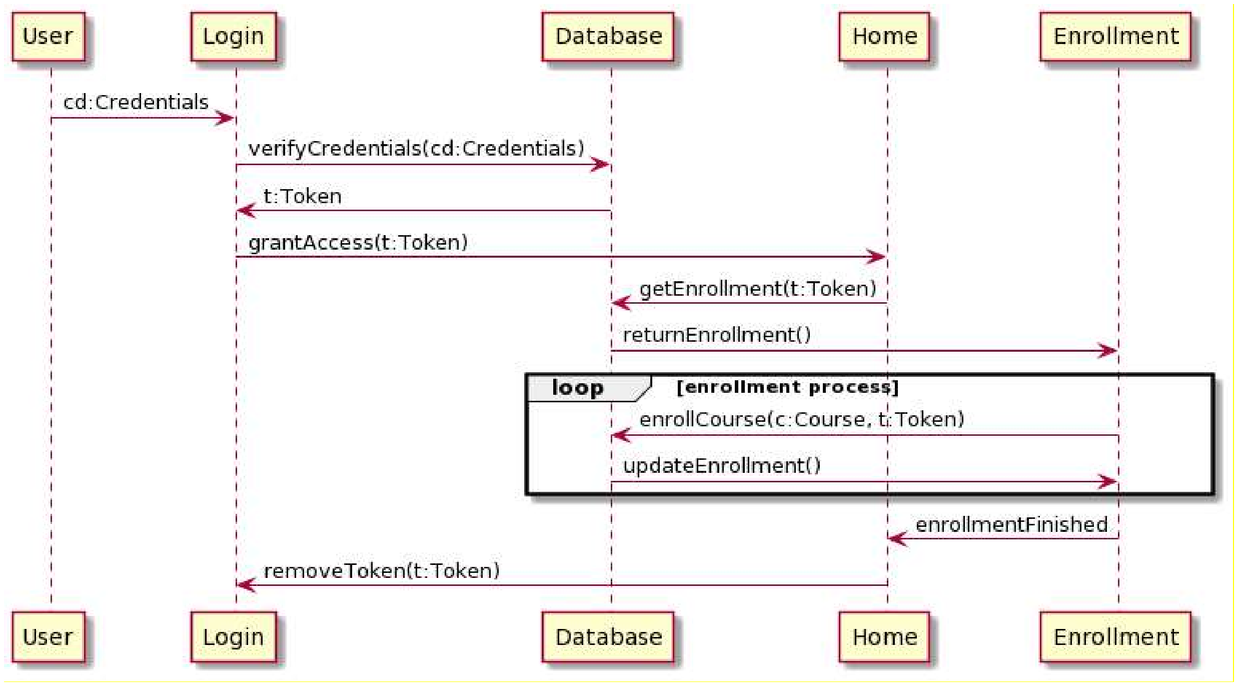




This diagram models the capacity attribute of the courses. Students can enroll courses until the capacity is reached, if that is the case, the only way to enroll that course would be to wait for students to drop out of the course.

3. Plant UML:

- Enrollment Process UML



The state diagram above illustrates the undergoing process in order to enroll courses in our system.

N. Rough sketch requirements as user stories

As a student, I hope that I can enroll in my distance courses quickly and effortlessly, and painlessly through enrolling my preferred courses. Also, I want my data to be protected confidentially. Upon entering the enrollment system, it is expected to be easy to use and, above all, to be stable to enroll in my courses effectively and without interruptions. A brief explanation of the course is expected to arouse students' interest, and they can reinforce knowledge to succeed in class. The user needs to know which classes he is going to enroll in and the prerequisites to enroll them in the system.

As a professor, I want students to be able to find the courses that I offer to attract interested students. It is expected that the professor is offering the class which was offered above and that it is by the number of students you can handle and the availability of seats in the classroom. Finally, the teacher must know which class he is willing to give and his availability of hours and days for this course.

O. Requirements

a. End Goal

As a student of the University of Puerto Rico I want to be able to create my semester tuition using a reliable and user-friendly platform without the need of downloading third party softwares. In addition, where I do not need to deal with any sort of technical issues like; not being able to connect to the platform or not being able to enroll in a course. I want a smooth and fast process where I can enroll my desired courses without any delay problem of the system.

P. Software Design & Implementation

a. Before we start, here's some background information: In order to create a realistic and viable software design we need to take into consideration some of the points of the root cause analysis. Is easy to just propose a solution that states "Create a user friendly application", but as we stated on our A3 the problem is not only the UI/UX of our current course enrollment application but the reliability of the system when a lot of users are connected to it.

This is why the core part of our software design is the cloud infrastructure. We as students and citizens of Puerto Rico know the situation that the UPR is going through. So at the end of the day the students from UPR and the faculty want to choose where they spend their funds(money) wisley. This is why we decided to design and develop in a serverless environment.

Serverless Computing: Serverless computing is a cloud computing execution model in which the cloud provider runs the server, and dynamically manages the allocation of machine resources. **Pricing is based on the actual amount of resources consumed by an application, rather than on pre-purchased units of capacity.**
Reference: https://g.co/kgs/MVUawB

As mentioned above, when using serverless computing we don't need to worry about managing servers, load balancing, etc. Because our cloud provider is the one that is responsible for that. As stated on the AWS Serverless Computing Webpage: "Serverless enables you to build modern applications with increased agility and lower total cost of ownership. Building serverless applications means that your developers can focus on their core product instead of worrying about managing and operating servers or runtimes, either in the cloud or on-premises. This reduced overhead lets developers reclaim time and energy that can be spent on developing great products which scale and that are reliable".
Reference: Serverless Computing – Amazon Web Services

b. These are the tools from AWS that we are going to use:

i. **VPC(Virtual Private Cloud)**

1. "Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways. You can use both IPv4 and IPv6 in your VPC for secure and easy access to resources and applications.

   You can easily customize the network configuration of your Amazon VPC. For example, you can create a public-facing subnet for your web servers that have access to the internet. You can also place your backend systems, such as databases or application servers, in a private-facing subnet with no internet access. You can use multiple layers of security, including security groups and network access control lists, to help control access to Amazon EC2 instances in each subnet."

Reference: Amazon Virtual Private Cloud (VPC)

2. In our VPC we will deploy our backend services(Database, Lambda Functions) in the private subnet. This will help us in the Networking aspect with things such as security and management.

3. **PRICING:** As mentioned earlier, pricing is important to our software design. We won't be charged for creating and deploying lambda functions and DB instances on a VPC. However, you will get charged if you allocate IP addresses for public instances and don't use them.The price is $0.01/hr for each IP that we reserve and do not use.

ii. **Lambda**

1. "AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of Amazon Web Services. It is a computing service that runs code in response to events and automatically manages the computing resources required by that code."

Reference: https://g.co/kgs/yKNvXr

2. We will be using Lambda to deploy our API and Web Application.

3. **PRICING:** We are going to be charged for the number of requests of functions received and the duration it takes them to be fully executed.

For example if we allocate 512 MB of memory to our function and are executed 3 million times in one month and it runs for 1 second each time, the total will be $18.34 for that month. So take into consideration that the course enrollment application is overloaded by students only 2-5 times per year. We don't get charged during the months the functions are not invoked.

4. More pricing info here: AWS Lambda – Pricing

iii. Amazon Aurora Serverless

1. “Amazon Aurora Serverless is an on-demand, auto-scaling configuration for Amazon Aurora (MySQL-compatible and PostgreSQL-compatible editions), where the database will automatically start up, shut down, and scale capacity up or down based on your application's needs. It enables you to run your database in the cloud without managing any database instances. It's a simple, cost-effective option for infrequent, intermittent, or unpredictable workloads.

Manually managing database capacity can take up valuable time and can lead to inefficient use of database resources. With Aurora Serverless, you simply create a database endpoint, optionally specify the desired database capacity range, and connect your applications. You pay on a per-second basis for the database capacity you use when the database is active, and migrate between standard and serverless configurations with a few clicks in the Amazon RDS Management Console.”

Reference: Amazon Aurora Serverless | MySQL PostgreSQL Relational Database | Amazon Web Services

2. We are going to use a relational database in order to maintain consistency on the data and also based on how old the current UPR course enrollment application is, the probability of them using a relational database is very high compared to a non relational database. But if they happen to be using a non relational database there is also the same services for non relational databases on AWS.

3. **PRICING:** As mentioned above the Aurora Serverless DB can be shut down automatically and you only will be charged for the data inside the DB. But realistically talking, we don’t recommend doing that because if the DB is shut down and then starts up again I might take a minute to load completely resulting into a timeout on the client side. So we

recommend having only one Aurora Capacity Unit(AUC) on. One ACU has approximately 2 GB of memory with corresponding CPU and networking, similar to what is used in Aurora user-provisioned instances. The price per AUC is $0.06/hr. That's equal to $43 monthly when only one unit is on. But again, the point of Aurora Serverless is to handle occasional spikes in traffic which result in a cheaper alternative than paying for multiple user provisioned instances.

4. More pricing info here: [Amazon Aurora Pricing | MySQL PostgreSQL Relational Database | Amazon Web Services]

iv. NAT INSTANCE

1. When Lambda functions are deployed into a VPC they automatically lose outbound internet connection. Since our application needs to send confirmation emails to the students when they make a change on their semester tuition we need to give these lambda functions internet access by deploying a ec2 instance with the NAT instance image and configuring it as a NAT gateway. We will deploy this on the public subnet and this will cost $0.0116/hr resulting in approximately $8.35 monthly and you can shut it down whenever you want in contrast to giving access by using a NAT gateway which will cost approximately $38 monthly and you can NOT shut it down.

c. Design Pattern Justification Reasons:

i. We've decided to go with a serverless implementation due to the current problem that we are trying to solve. As mentioned earlier the UPR course enrollment application crashes due to overload. But, this kind of traffic is only received approximately three-five times per year. Thus, it is not cost-efficient to buy a more expensive/powerful server instance since these types of spikes only occur three-fives times per year and we will be paying for resources that we don't use. Serverless is designed to handle that type of unusual traffic spikes while at the same time lowering costs.

ii. Another reason that we've selected serverless is because of the simplicity of the scenario that we are in. If we go about implementing actual servers we will need to create multiple load balancers in addition to the servers. This is not a bad approach but is not an ideal one for our scenario.

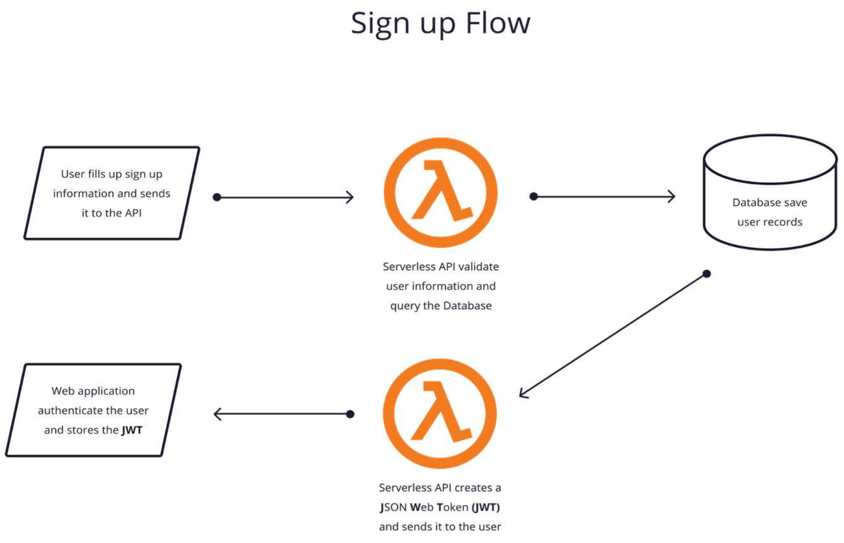
iii. On the other hand, one of our team members (Jaime Ginorio) has experience with the same exact technology stack that is being proposed. This is a huge advantage since one of the members is already familiarized with this, he can teach the other members and advance in the project on a fast phase.

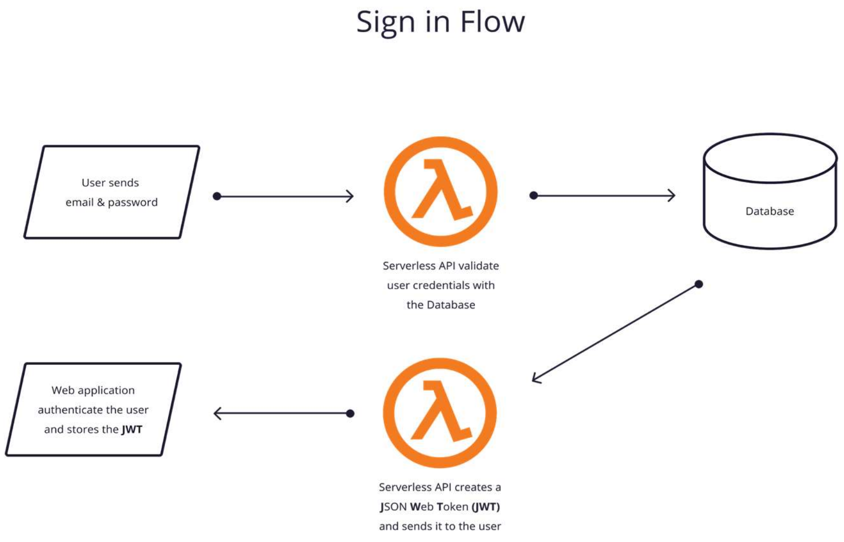
d. Phase 1- Implementation Progress (7/October/2020:

i. As a team we decided to create a separate repository for the backend and the frontend during this first phase, to keep organization in our code and avoid merge problems within the team. Even so the communication of the team is constant and we held weekly meetings in order to be on the same page and give updates. Following will be presented the progress on implementation of both teams.

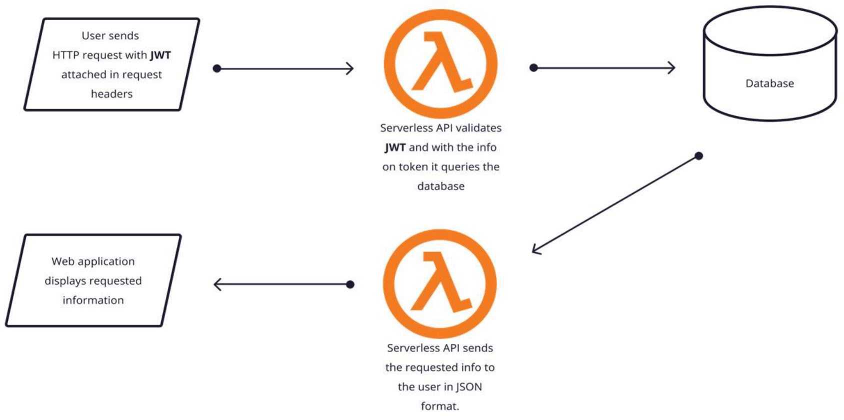
ii. BACKEND:

1. Repository link: https://github.com/fabiolarobles1/los-reyes-de-la-punta-backend
2. Diagrams:
   a. The diagrams below represent the current routes that are implemented and the current Database schema.

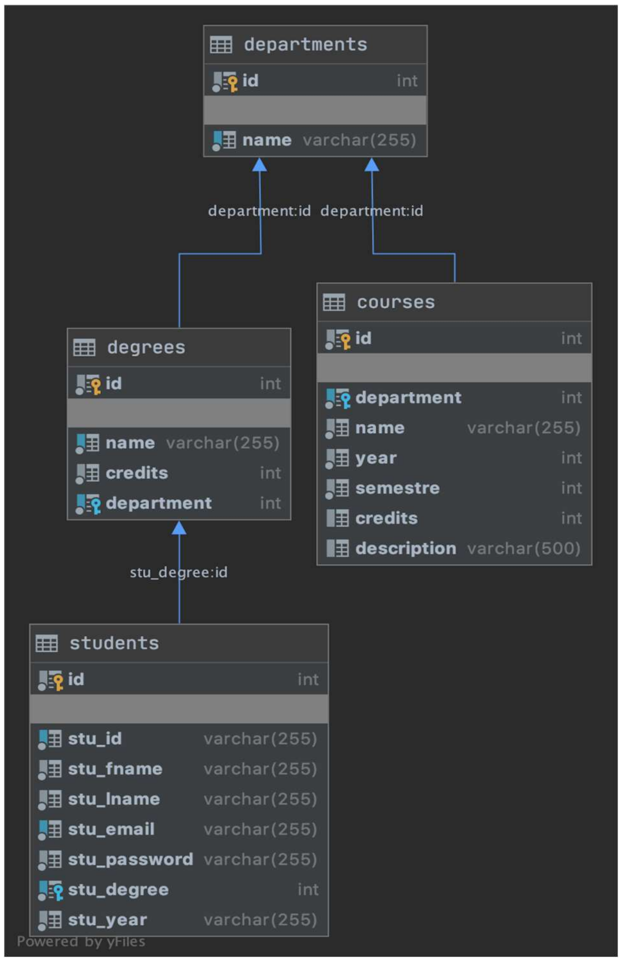

Sign up Flow
User fills up sign up information and sends it to the API
Serverless API validate user information and query the Database
Database save user records
Web application authenticate the user and stores the JWT
Serverless API creates a JSON Web Token (JWT) and sends it to the user


Sign in Flow
User sends email & password
Serverless API validate user credentials with the Database
Database
Web application authenticate the user and stores the JWT
Serverless API creates a JSON Web Token (JWT) and sends it to the user

Authentication Pattern

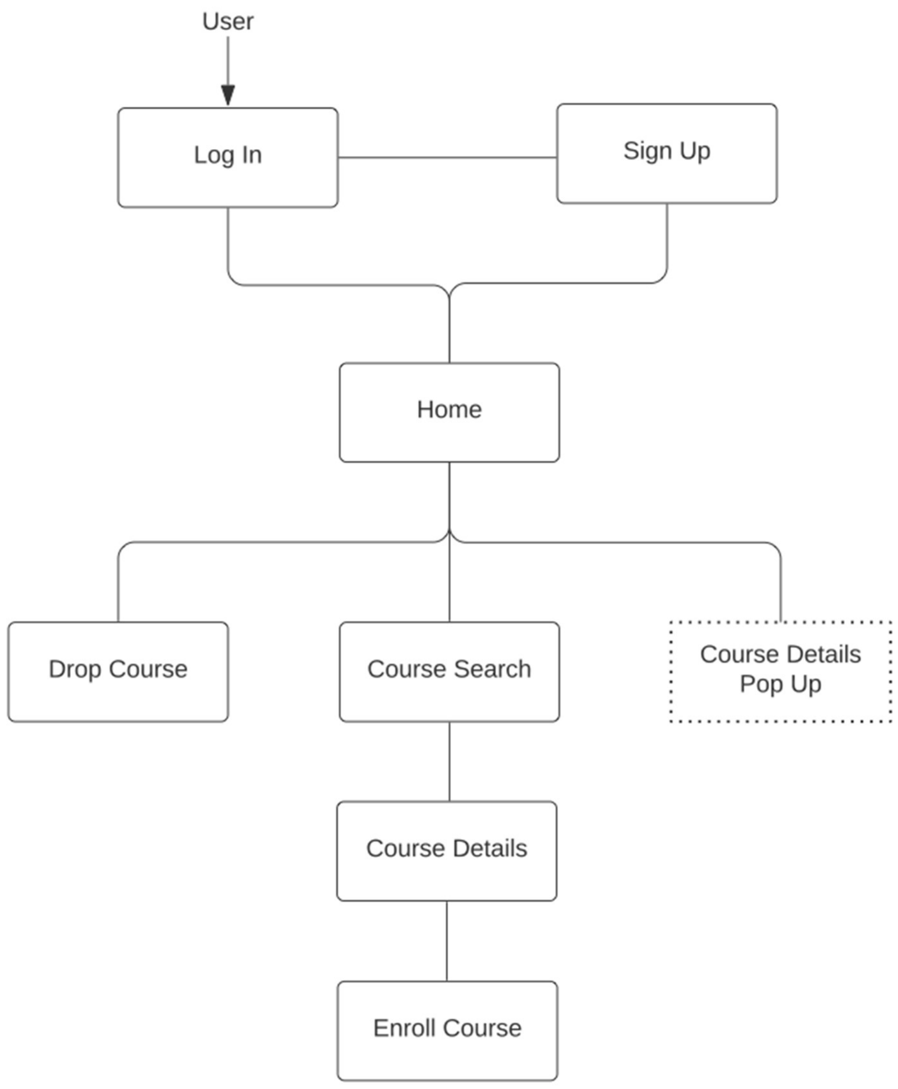

User sends HTTP request with **JWT** attached in request headers
Serverless API validates **JWT** and with the info on token it queries the database
Database
Serverless API sends the requested info to the user in JSON format.
Web application displays requested information

Current Database Schema

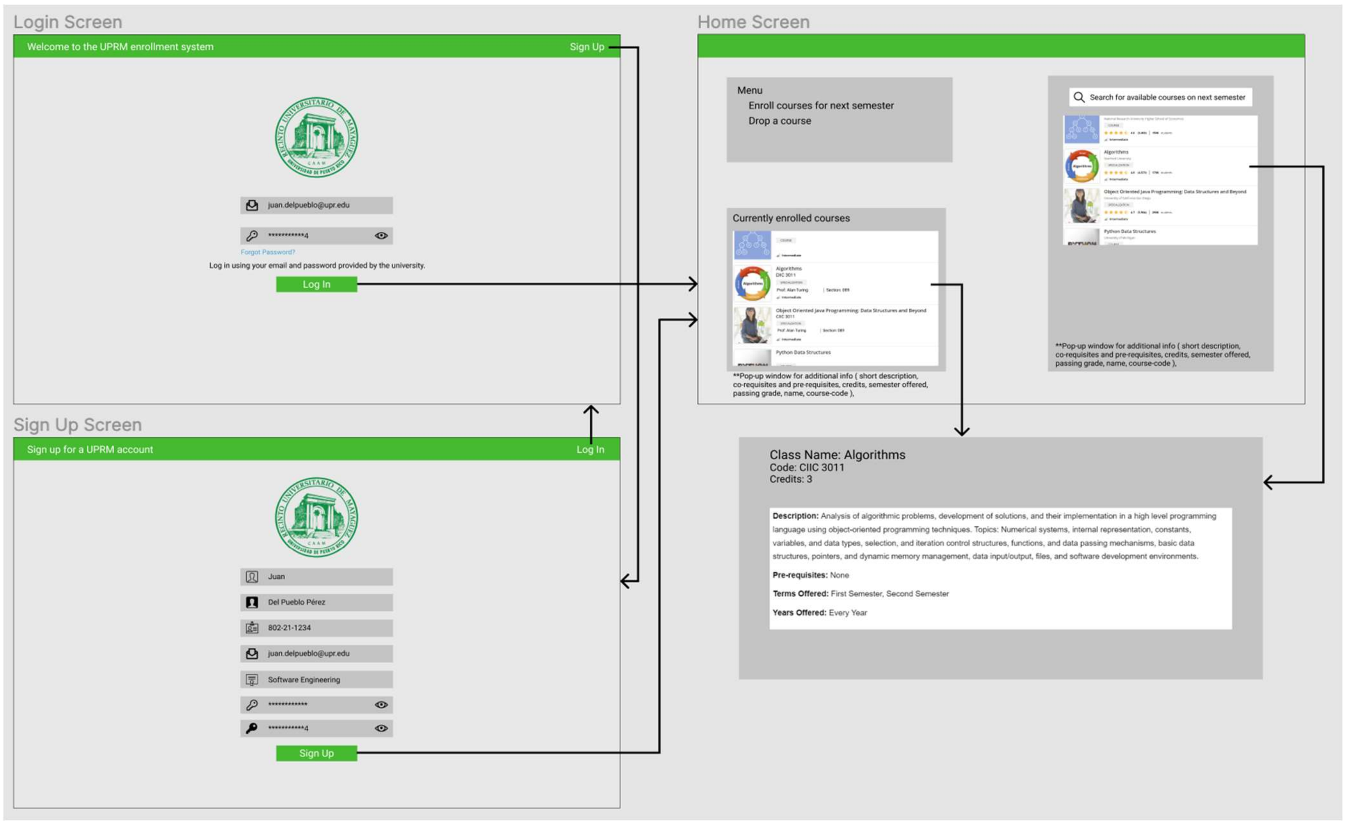


3. Demo Video:
    a. Authentication video, in this video it can be shown the authentication process explained above on the diagrams.

iii. FRONTEND:
1. Repository link: https://github.com/fabiolarobles1/los-reyes-de-la-punta-frontend
2. The front-end team first decided to create a site map and a mock up design of the pages on Figma. The following images display the design plan made so far. The arrows represent the flow the page will have for the user.



a. *Site Map:* The diagram above shows all the separate pages and the user flow of the website.


Login Screen
Welcome to the UPRM enrollment system
Sign Up
juan.delpueblo@upr.edu
Forgot Password?
Log in using your email and password provided by the university.
Log In
Sign Up Screen
Sign up for a UPRM account
Log In
Juan
Del Pueblo Pérez
802-21-1234
juan.delpueblo@upr.edu
Software Engineering
Sign Up
Home Screen
Menu
Enroll courses for next semester
Drop a course
Currently enrolled courses
Algorithms
Object Oriented Java Programming: Data Structures and Beyond
Python Data Structures
Search for available courses on next semester

**Pop-up window for additional info (short description, co-requisites and pre-requisites, credits, semester offered, passing grade, name, course-code),

**Pop-up window for additional info (short description, co-requisites and pre-requisites, credits, semester offered, passing grade, name, course-code),

Class Name: Algorithms
Code: CIIC 3011
Credits: 3

Description: Analysis of algorithmic problems, development of solutions, and their implementation in a high level programming language using object-oriented programming techniques. Topics: Numerical systems, internal representation, constants, variables, and data types, selection, and iteration control structures, functions, and data passing mechanisms, basic data structures, pointers, and dynamic memory management, data input/output, files, and software development environments.

Pre-requisites: None

Terms Offered: First Semester, Second Semester

Years Offered: Every Year

b. *Relational Mock Up:* First the user will be presented with the login page, if they already have an account, they can put in their credentials and then be directed to the home page. If they don't have an account, from the login page they can go to the sign up page to create an account and after doing so they'll be directed to the home page. In the home page they can see the courses they're currently enrolled in, they can search for the courses that are available for next semester, and they can also be directed to the drop course page or the enroll courses page. In the home page from the current courses or next semester courses, when they click on a course, a pop up will appear that shows them that course's information.

3. Demo videos:
   a. Login: Login Screen Recording Link
      i. In this video, it can be seen the login page so far, and the request call to the backend, receiving an access token. The token is encrypted with the student info.
      ii. Note: The logs on the console were only for demonstration purposes, the decoded information will not appear on the real application when completed. For now, the token is saved on the local storage, still, a little research needs to be done in order to determine where is the safest place to save the token.

e. Phase 2- Implementation Progress (11/November/2020):

i. As a team we have been continuing with the agile development method in our project. We keep holding weekly meetings where we discuss the progress of the project and give feedback on design and implementation. We are following an approach of designing a functional aspect, implementing and testing it in order to maintain progress in all stages of the project.

ii. BACK-END

1. Backend Github Repo
2. Through the past few weeks we have been able to implement some new routes in the REST API. Here are the routes with their respective information:

   a. GET: /students/courses_firstSemester

      i. This route will respond with the student(user) current active courses, in an array of JSON objects. This request requires a valid access token to be attached to the request headers with a key of “Authorization” and a value of “Bearer <access token>”. Inside the token there is the necessary information to query the Database and respond with the users current courses. This route will be called as soon as the user logs in into the web application since the homepage shows the user current courses.

      ii. JSON Object of the course example:

```
   {
       "id": 3,
       "department": 3,
       "name": "MATE3031",
       "year": 1,
       "semestre": 1,
       "credits": 4,
       "description": "ELEMENTARY DIFFERENTIAL AND INTEGRAL
CALCULUS OF ONE REAL VARIABLE, WITH APPLICATIONS."
   }
```

   b. GET: /students/courses_secondSemester

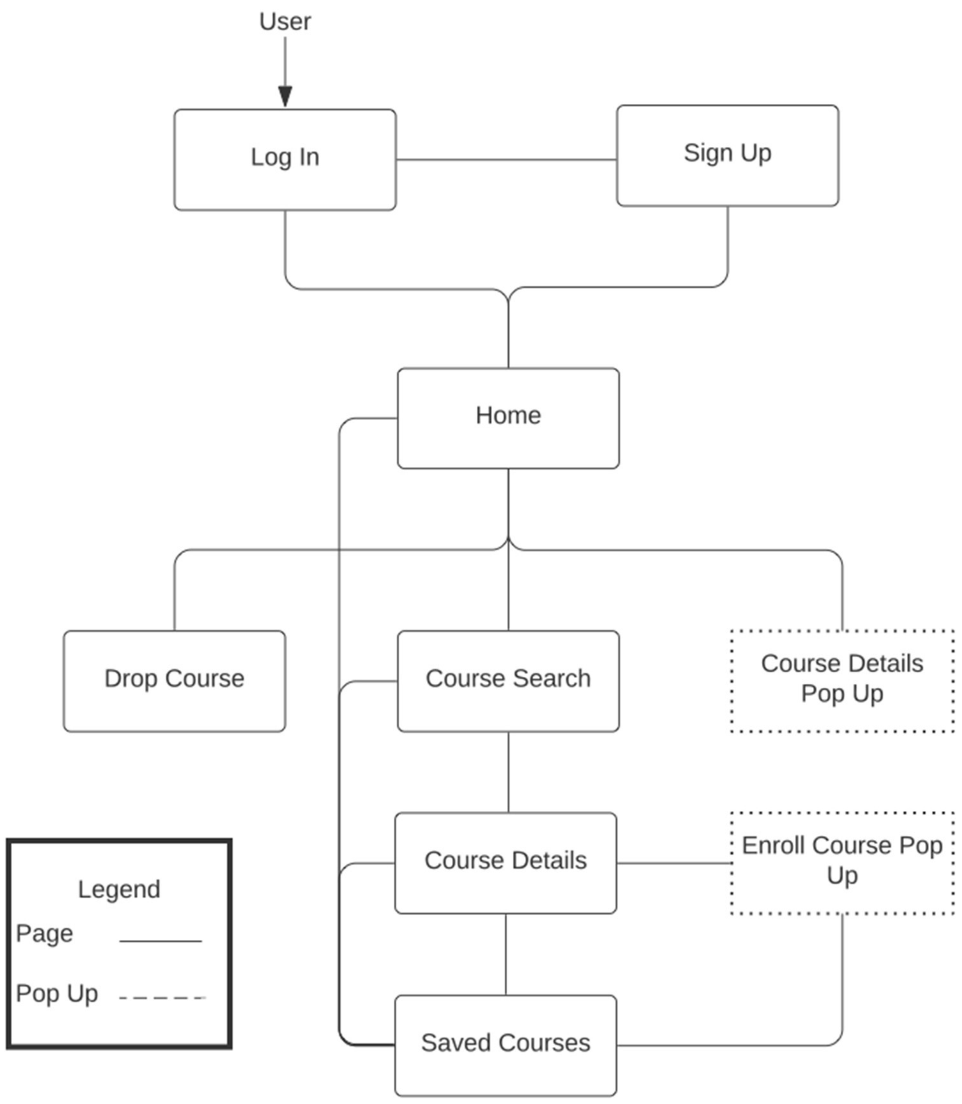
      i. This route will respond with the student(user) suggested courses for next semester, in an array of JSON Objects. This request also requires a valid access token to be attached to the request headers with a key of “Authorization” and a value of “Bearer <access token>”. Inside the access token there is the necessary information to query the Database and search for the student suggested courses for the next semester. This route will be called as soon as the users logs in since the home page of the web application will show the user suggested courses for the second semester.

c. POST: /students/search_courses
    i. This route will respond with the results of a search done in the Database based on a user input. The user should send their input in the body of the request with a key value of “name” and a value of their input. For example, say that we want to make a search for “CIIC” courses. We make a POST request with our access token attached to the headers of the request and the request body will look like this { “name”: “CIIC” }. The REST API will respond with an array of JSON objects of the courses found based on the input received.
    ii. Example of the JSON objects:

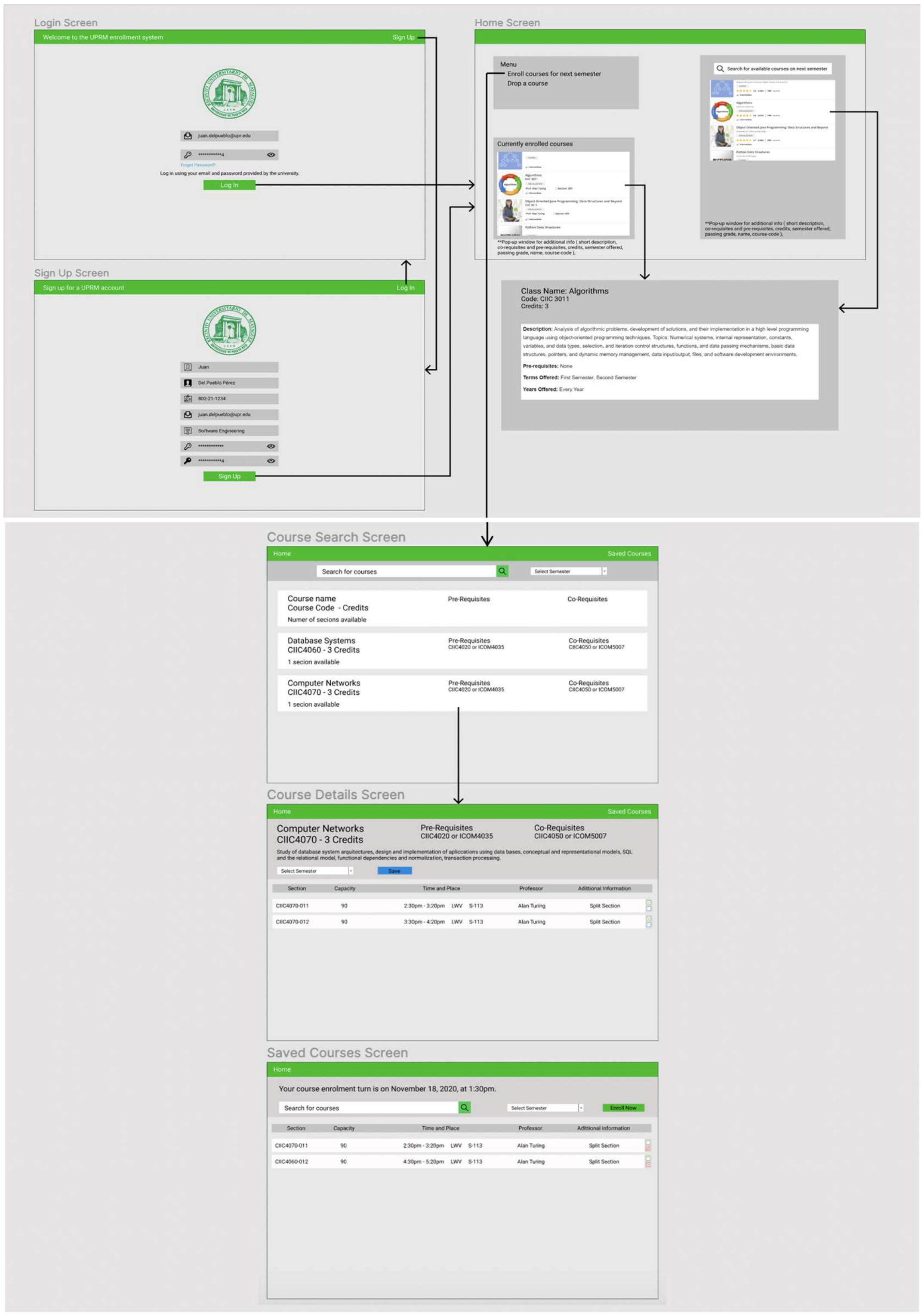
```
{
    "id": 49,
    "department": 1,
    "name": "CIIC4070",
    "year": 4,
    "semestre": 2,
    "credits": 3,
    "description": "STUDY AND DEVELOPMENT OF SKILLS REQUIRED FOR THE DESIGN OF NETWORK PROTOCOLS AND NETWORK-CENTRIC APPLICATIONS"
}
```

iii. FRONT-END

1. Frontend Github Repo
2. The site map and a mock up design of the pages on Figma have both been updated. The following images display the design plan made so far. The arrows represent the flow the page will have for the user.



   a. *Site Map:* The diagram above shows all the separate pages and the user flow of the website.



Log in using your email and password provided by the university.

**Pop-up window for additional info (short description, co-requisites and pre-requisites, credits, semester offered, passing grade, name, course-code),

Class Name: Algorithms
Code: CIIC 3011
Credits: 3

Description: Analysis of algorithmic problems, development of solutions, and their implementation in a high level programming language using object-oriented programming techniques. Topics: Numerical systems, internal representation, constants, variables, and data types, selection, and iteration control structures, functions, and data passing mechanisms, basic data structures, pointers, and dynamic memory management, data input/output, files, and software development environments.

Pre-requisites: None

Terms Offered: First Semester, Second Semester

Years Offered: Every Year

Course name Course Code - Credits Numer of secions available	Pre-Requisites	Co-Requisites
Database Systems CIIC4060 - 3 Credits 1 secion available	Pre-Requisites CIIC4020 or ICOM4035	Co-Requisites CIIC4050 or ICOM5007
Computer Networks CIIC4070 - 3 Credits 1 secion available	Pre-Requisites CIIC4020 or ICOM4035	Co-Requisites CIIC4050 or ICOM5007

Computer Networks
CIIC4070 - 3 Credits
Pre-Requisites
CIIC4020 or ICOM4035
Co-Requisites
CIIC4050 or ICOM5007

Study of database system arquitectures, design and implementation of apliccations using data bases, conceptual and representational models, SQL and the relational model, functional dependencies and normalization, transaction processing.

Section	Capacity	Time and Place	Professor	Adittional Information
CIIC4070-011	90	2:30pm - 3:20pm LWV S-113	Alan Turing	Split Section
CIIC4070-012	90	3:30pm - 4:20pm LWV S-113	Alan Turing	Split Section

Your course enrolment turn is on November 18, 2020, at 1:30pm.

Section	Capacity	Time and Place	Professor	Adittional Information
CIIC4070-011	90	2:30pm - 3:20pm LWV S-113	Alan Turing	Split Section
CIIC4060-012	90	4:30pm - 5:20pm LWV S-113	Alan Turing	Split Section

b. *Relational Mock Up:* First the user will be presented with the login page, if they already have an account, they can put in their credentials and then be directed to the home page. If they don't have an account, from the login page they can go to the sign up page to create an account and after doing so they'll be directed to the home page. In the home page they can see the courses they're currently enrolled in, they can search for the courses that are available for next semester, and they can also be directed to the drop course page or the enroll courses page. In the home page from the current courses or next semester courses, when they click on a course, a pop up will appear that shows them that course's information. From the home page they can go to the course search, where they can type in the course name or code and select the semester to search. Once the search results show up, they can select a course from the list and be directed to the course details page, where they can see all the sections available for that course in the selected semester. From the course details, they can enroll the course section, or they can save the course section to enroll it once it's their enrollment turn. They can then visit their saved courses page and when it's their enrollment turn, they can enroll the course sections they had previously saved.

c. Demo:

   i. During these past weeks there have been a vast amount of progress on the implementation part in terms of coding. The Signup page is complete and connected to the API as well as the Login page. Both pages take care of invalid inputs situations by displaying error messages to the screen. The Home page was added and it displays the flow presented in the Figma model with the students current courses and next semester course. Also the user can search for available courses for the next semester. In order to maintain security we implemented 'route guards' which basically takes care that if an user is not authorized(i.e. not logged in) then he or she would not be able to access the internal parts of the web page as Home page or enrollment of courses etc. and it will redirect to the Login page.

      - All this will be shown during the in-class demo.

f. **Final Phase -** Implementation Progress (24/November/2020)

i. We've been continuing with the agile development method in our project. We've held at least one full team meeting each week and at times we've held separate back-end front-end meetings to discuss more in depth design and implementation details. We've been focusing on making sure that all features are implemented and fully functional, while also making the page easy to use and visually pleasing. We practiced some meetings of peer programming in both teams in order to finish the final methods faster and more efficiently. *A demo video of the final product will be handed in to demonstrate the page fully functional.*

ii. BACK-END

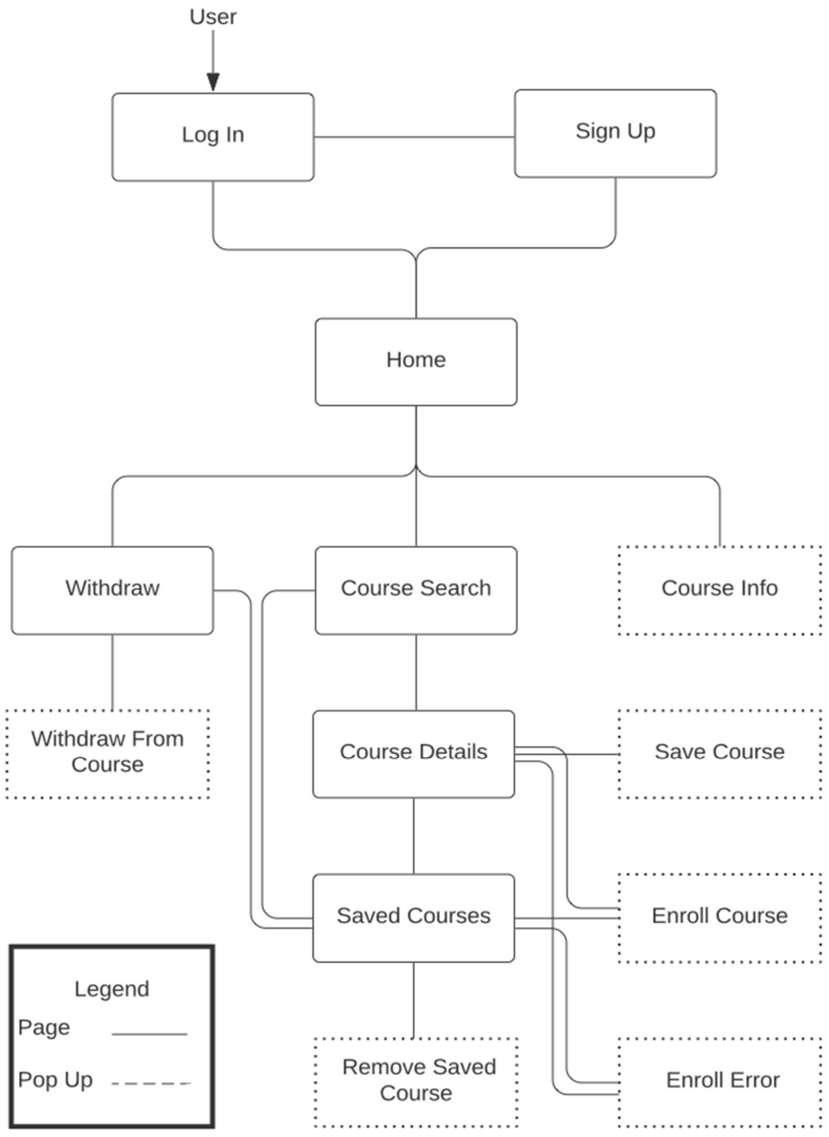
1. Backend Github Repo
2. Through the past few weeks we keep up working and implementing new routes to our REST API. All of the new implemented routes need an access token in order to be accessed. This access token is provided to the user once he signs in into the application. Then it is stored in the local storage of the browser. Each time a request is being made to the API, this token is attached in the headers of the request in order to access protected routes. Below are the new routes added with their respective information:
   i. Method and Route: GET /student_enrollment
   This route gets the courses that the user(student) already has enrolled for his next semester. If the user hasn't enrolled in any courses, this will bring an empty array.
   ii. Method and Route: GET /search_sections
   Optional Query Params: <search> and <semester>
   This will get **all** course sections available for next semester if no query parameters are provided. The "*search*" query parameter is for filtering sections by their respective name, description or course code(ej: INSO4020). The "*semester*" query parameter is for filtering sections by their semester, the query parameter expects and number, 1 or 2.
   iii. Method and Route: POST /enroll_course
   Body: { sectionId: <unique identifier of the section> }
   This method expects a JSON Object with a key value of "*sectionId*" and a value of the unique identifier of the section in the database. This will enroll the received section of the course for the next semester.
   iv. Method and Route: POST /withdraw_course
   Body: { sectionIds: [<unique identifier of the section>] }
   This method expects a JSON Object with a key value of "*sectionIds*" and a value of an array of unique identifiers of the sections that you want to remove from your current enrollment.
3. Save Sections feature; This feature acts like a shopping cart for students to save posible sections that they want to enroll in. For this feature we implemented three routes that they all required an access token in order to be accessed. These are the routes;
   i. Method and Route: POST /save_section
   Body: { sectionIds: [<unique identifier of the section>] }
   This method expects a JSON Object with a key value of "*sectionIds*" and a value of an array of unique identifiers of the sections that you want to save.
   ii. Method and Route: DELETE /remove_section
   Body: { sectionIds: [<unique identifier of the section>] }

This method expects a JSON Object with a key value of "*sectionIds*" and a value of an array of unique identifiers of the sections that you want to remove from your save sections.

iii. Method and Route: GET /saved_sections
This method brings an JSON Object array of the sections that the user has saved.

iii. FRONT-END

1. [Frontend Github Repo]



a. *Site Map:* The diagram above shows all the separate pages and the user flow of the website. Pop ups were added in the course details, saved courses and withdraw; they serve the purpose of giving the user a visual acknowledgement that an action was successfully completed or that an error occurred .

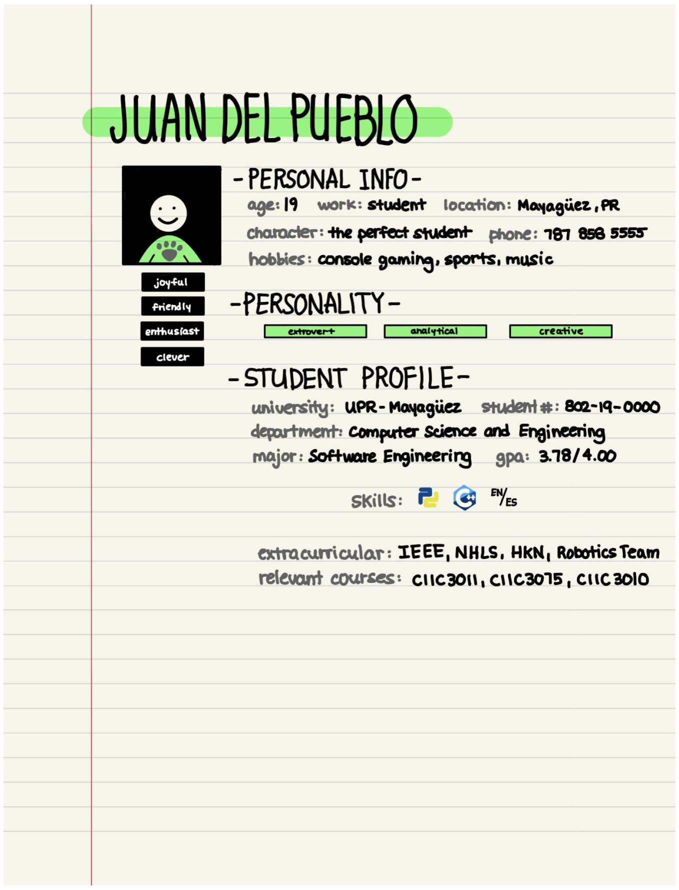
b. *Demo:* The page is now fully functional; all the requirements and user stories have been completed. A regular usage scenario would be: the user logs in, they go to the course search and find the courses they want for next semester, from there they can directly enroll in the course or add it to their saved courses. From their saved courses they can remove the courses they decided they didn't want and enroll all their courses at once. If they change their mind and want to withdraw from a course, they can go to the withdraw page where they'll see all the courses they're enrolled in, and from there they can withdraw from the courses they previously enrolled in.

Q. Terminology

a. Course Enrollment System: the tool students use to enroll in classes each semester.
b. Course: these are the classes that form the curriculum and program; it is composed of a course name/code, a number of credits (which is the "value" of a course) and a professor that will teach that course.
c. University: the institution that a student attends to.
d. Credentials: these are pieces of information that details the authentication of a student.
e. Curriculum: this is the map for a student to follow for completing the program.
f. Easy-to-use/user-friendly: this term refers to an interface that a first time user could understand without having to look for support to complete the desired task.
g. Semester: a period of time (in months usually) that a student will take the courses he/she enrolled in.
h. Student: The entity that interacts with the courses and the enrollment system.
i. Program/Degree: this is the degree program that the student is pursuing that belongs to a department.
j. Department: The institution is divided into several departments which groups several programs of related fields. (Institution contains departments which contains different programs available)
k. Department director: the person with greater authority in a department, usually gives special permission to courses.

R. Stakeholders

a. Regarding our domain we developed two personas which will be mainly affected by the project, which are the students and the program/department directors. We decided to create these personas first since they will be the one working directly with the interface so they have a strong point of view regarding the system to be. Students since they are the ones who will enroll the courses and program directors since they are the ones in charge of granting special permission during the enrollment process.



JUAN DEL PUEBLO

-PERSONAL INFO-

age: 19 work: student location: Mayagüez, PR

character: the perfect student phone: 787 858 5555

hobbies: console gaming, sports, music

joyful

friendly

enthusiast

clever

-PERSONALITY-

extrovert analytical creative

-STUDENT PROFILE-

university: UPR-Mayagüez student #: 802-19-0000

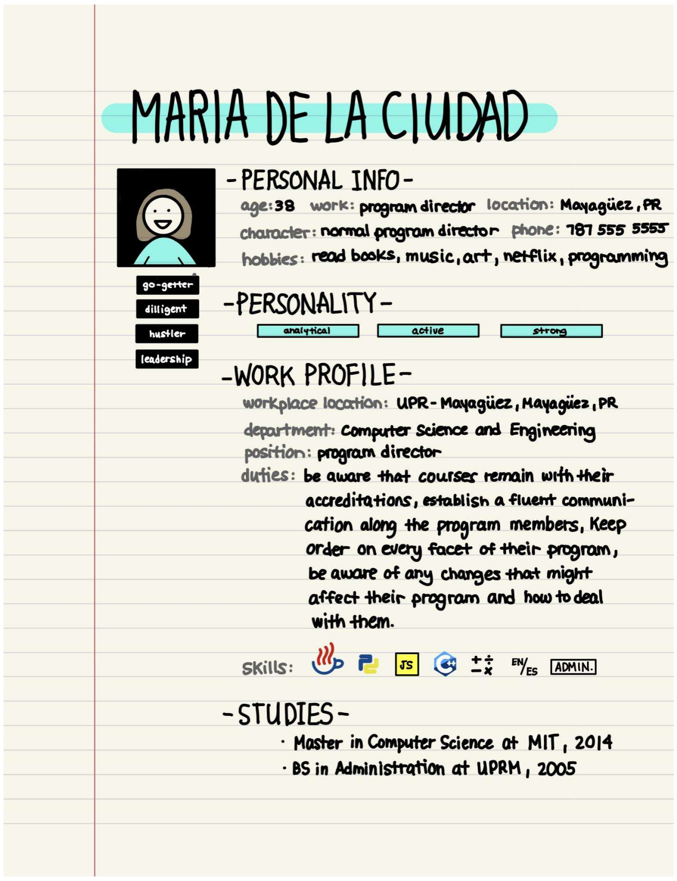
department: Computer Science and Engineering

major: Software Engineering gpa: 3.78/4.00

Skills: EN/ES

extracurricular: IEEE, NHLS, HKN, Robotics Team

relevant courses: CIIC3011, CIIC3075, CIIC3010



MARIA DE LA CIUDAD

-PERSONAL INFO-

age: 38 work: program director location: Mayagüez, PR

character: normal program director phone: 787 555 5555

hobbies: read books, music, art, netflix, programming

go-getter

dilligent

hustler

leadership

-PERSONALITY-

analytical active strong

-WORK PROFILE-

workplace location: UPR-Mayagüez, Mayagüez, PR

department: Computer Science and Engineering

position: program director

duties: be aware that courses remain with their accreditations, establish a fluent communi-cation along the program members, Keep order on every facet of their program, be aware of any changes that might affect their program and how to deal with them.

Skills: JS EN/ES ADMIN.

-STUDIES-

· Master in Computer Science at MIT, 2014

· BS in Administration at UPRM, 2005

S. Business Processes

a. One of the business processes concerning our domain is the one a student goes through the enrollment process. First a student needs to evaluate the curriculum of the program he/she is seeking in the institution to have an idea of which courses he or she should look for to take on the next semester. After having an idea of those courses, the student goes to the "portal" to check the availability of the courses for next semester. Here the student proceeds to plan several "pre-matriculas" which are possible scenarios of his/her schedule next semester. After this planning, the student waits for his/her enrollment shift while constantly checking how the availability of the courses is behaving in order to know if he/she needs to modify the plan. When the student enrollment shift arrives he/she enters the system and proceeds to make the enrollment of the desired courses. When the semester arrives he/she confirms the "matricula" by paying or confirming it on the "portal" and there is when the enrollment process concludes until next semester is close that is where the cycle begins again.

b. Another business process relevant to our domain is from the institution point of view, who is in charge of updating the enrollment system with the courses available for the next semester. In order to do this, each department of the university needs to gather information from the professors to see availability to give courses for next semester. Department directors need to verify that the courses meet the demand and needs of the students and the program of said department and after this they have to provide the sections with assigned professors and schedules to the ones in charge of updating the enrollment system in order for it to be up to date for the students during the process of enrollment.

T. Domain Facets

a. Intrinsics:

   i. In order for our domain to exist there are several indispensable aspects which are needed to be able for the domain to be sustainable.

      - From the point of view of a potential student of the UPR there has to exist an institution where he/she can enroll in a program.
      - From the point of view of a current student of a program in the institution, in order to enroll courses for next semester there has to exist a system where he/she can login and see the available courses. In this system, when the time comes, he/she should be able to enroll in courses from the program he/she was admitted to.
      - From the point of view of the institution, aside from students and programs, there have to exist courses related to each program which can be enrolled by students on an enrollment system available to them with all the information (professors, schedule,etc.) provided by directors of each program in the institution.
      - From the point of view of program directors which are in charge of organizing courses, then there have to be professors available to provide these courses for the students in the institution.

b. Support Technologies

   i. Before having advantages in technology, the enrollment record of students was tacked on paper. Students had to go to their departments and wait to be enrolled in the courses by a secretary or administrative assistant.

   ii. With advantages of technology, the enrollment system was moved to a virtual environment, which is the one currently being used by the university consisting of a SSH protocol with the students credentials and browsing through a terminal to enroll courses. The process was more certain and fast, but the continous growth in technology is becoming outdated.

c. Management and organization

   i. When a student wants to enroll in a course and the system is not letting it be enrolled, then the student proceeds to go to his/her department. In the department he/she will contact the administrative assistant which has a higher level of access to the system and see if the problem can be solved. If it is a course that needs director permission then the assistant will contact the director in order to receive permission to enroll the course for the student.

ii. When the enrollment system crashes the personnel in charge of maintaining the system analyse the problem and report the situation to the university chancellor which takes the decision of pausing or postponing the enrollment process if the problem is of higher concern.

d. Rules, Regulations

i. Some rules and regulations concerning the domain are:

Rules	**Regulation**
In order for a student to enroll in a course he/she needs to meet all the prerequisites for this course.	If the student does not fulfill the prerequisites of the course the system will not let them enroll the course or the student must go and ask for director permission.
If a student is a debtor, they will not be able to enroll for the next semester.	If the student has a debt with the institution the enrollment system access will remain closed until he/she pays the debt.
Each program in the institution has a maximum amount of credits per semester per student.	If a student wants to exceed the total amount of credits per semester he/she needs to ask permission from the director of the department so he or she enrolls the exceeding credits for the student.
A student needs special permission in order to take a class that is part of another degree.	The student needs to contact the department of the course he/she is interested in and ask for special permission. A person with higher permissions on the system (i.e. department director) will proceed to enroll the class if the permission is granted.
A student cannot enroll in more than one section of the same class in a semester.	If the student tries to enroll in two sections, the system will not let him/her and will ask to choose only one in order to proceed with the enrollment.

e. Human Behaviour

i. Software Programmer behaviour:

We would describe the programmer as *diligent* as one who follows the standard protocols to protect the student information and verifies and tests that the system follows the rules established for the course enrollments restrictions. We would characterize the programmer as being *sloppy* if the individual does not complete all the tests needed to fulfill the security and requirements that the system implies. We would consider the programmer as *delinquent* if the person does not check and tests the system and leads to problems during the enrollment process and possible security flaws concerning the protection of student information. Finally we would consider the programmer being a *criminal* if intentionally the individual does not follow the rules of the enrollment system in such a way that may be beneficial for certain students. Example maybe someone paid the programmer to let them take a course with a certain professor or to pass them a class that they never took, etc.

ii. Student behaviour:

We would describe a student as *diligent* as one who follows the enrollment process rules, i.e., waits for his/her enrollment turn and then proceeds to enroll in the courses. If a course capacity is over, he/she will proceed to join a waiting list for the course. In addition to that, a diligent student enrolls the classes that he can take, fulfilling the prerequisites of these.We would characterize the student as being *sloppy* if, due to ignorance, she tries to enroll in courses that do not have the prerequisites or tries to enroll, and her turn has not arrived.We would consider the student as a *delinquent* if he/she tries to buy a turn or class that cannot enroll through a student or someone who has access to the system.Finally, we would consider the student being a *criminal* who does not follow the enrollment process's rules and if he tries to infiltrate the system for his benefit or that of someone else. Either by enrolling classes that cannot enroll or enroll a course which is capacity is over, advance his enrollment turn or try to access the database in any way.

iii. Department director behaviour:

We can describe a Department Director as *diligent* as one who follows the rules stated by the university in order to use the platform and is constantly checking how the professors, courses and students are doing in his/her department to keep improving the academic offer of the department. We can describe a *sloppy* Department Director as one who uses the platform without any knowledge, just by assumptions and due to his actions, we as a university, can face some minor consequences. We can describe a *delinquent* Department Director as one who knows that his or her actions using the platform can provoke some significant damage to the department, but he/she does not get interested in improving the offer of the department or changing something that he/she knows needs to change but nobody has noticed. We can describe a *criminal* Department Director as one who uses the platform to his or her favor, abusing his or her power in the platform. For example; he/she keeps assuming this type of behavior for personal use or for someone who is paying them or has a personal relationship with them.

iv. Professor behaviour:

We can describe a Professor as *diligent* as one who follows the rules stated by the university to use the platform responsibly, i.e. hands in documents and grades on time and provides the availability for courses in an equal way to every student, etc. We can describe a Professor as *sloppy* as one who uses the platform without double checking availability of courses. For example: If a student complains that he needs a course to graduate, he doesn't bother to double check if that's true and enroll the student. A *delinquent* Professor is the one who uses the platform irresponsibly because he didn't read the procedures provided by the university on how to utilize the platform. A *criminal* Professor can be described as one who abuses his power on the platform. For example, he gets paid and he modified a student grade through the platform.

III. Analytic Part

A. Concept analysis of rough sketch

A University is made up of Departments, and it is for the Students. The Departments are made up of Degrees, Courses and Faculty members. Students belong to Departments and pursue Degrees that are from the same Department. Degrees are made up of Courses in which Students take according to previous Courses they have taken (if any). These Courses have a value called Credits and the value is assigned by Faculty members of the corresponding Department, the value usually ranges from 1 to 6.

An enrollment system is characterized by a set of students with specific actions such as: enroll classes, remove classes and a navigation system through the courses to see their availability. Each student has an established period to enroll in classes called sets of duration. After the student logs in, the process of choosing the desired classes according to their requirements, concentration, and convenience, we call classes set. At the same time that process is carried out, it verifies if the classes you want to be enrolled meet some requirements such as: passing the previous class, availability of time and availability of rooms, this is called a set of requirements. To carry out everything desired by the student, the changes will be saved, and a receipt will be sent validating the changes made and the amount to be paid for this we will call a set of confirmation.

B. Verification

The implementation of our system ensures that the current situation will be eased and needs would be satisfied, that most issues students confront with the current system would be of no concern in our system. For example, server slowing down or crashing won't be an issue because of our serverless implementation, no third party app is needed, navigation through commands is eliminated and instead a cursor-based navigation is available, reliability and stability are going to be the new common.

Our system provides an elegant but simple user interface, consisting of a login page, a sign up page, a home page, an enrollment page, and a dropdown page, all of them accessible with just on click. In the login page, users are able to access the system only by using their institution credentials. This log in system is implemented so that any user that hasn't signed up yet, will have the access denied. In order to gain access, a sign up page is available, this page is built so that only the most important information about the user is required. Information that will only be used to guarantee a better overall user experience, a more personalized environment, and successful enrollment changes. Our system will also ensure user confidentiality, meaning this that no information would be exposed to anyone. From the home page the user will be able to have an overview of the current semester courses and following semester courses, also a quick search will be available, but no action concerning the enrollment/dropdown processes would be made from this page. The enrollment page is concerned about next semester's courses, and users will be able to add/remove courses that would be part of the following semester. On the other hand the dropdown page is concerned about the courses that the user is currently taking but wants to remove, so users will be able to eliminate classes they don't want to take.

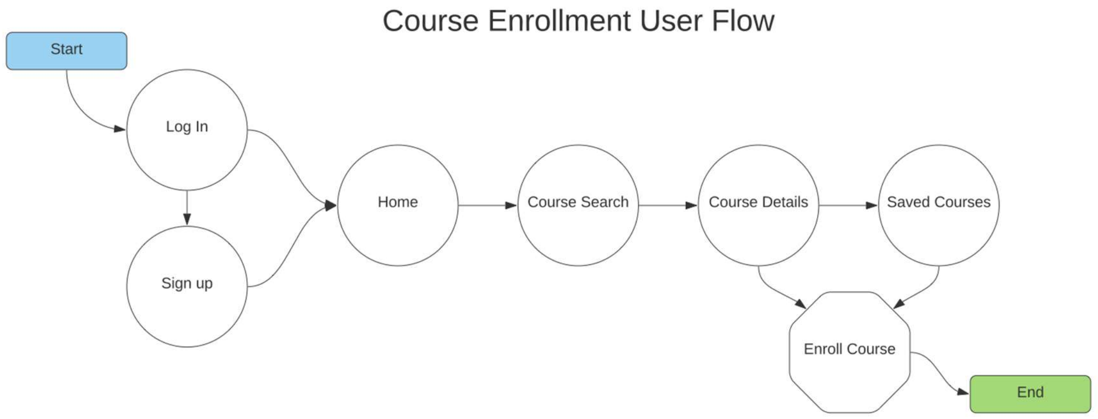
Terminology concerning the system will be as extensive as needed in order to make the system understandable to every concerning partner/stakeholder of our project. Concepts in general are defined in relevance to the current situation, and described in the simplest way possible, ensuring the understanding of most or all members of the project.

a. Verification Front End

i. Course Enrollment User Flow

The user will first be greeted with the login page, where if they have an account already they can log in, if not they can go to the sign up page where they can create an account. After that, they'll be directed to the home page, where they can then go to the course search, where they'll write the course name or code and select the semester they're enrolling in. From the search results they can select a course and be directed to the course details page, where they'll see the available course sections for that semester; from there they can directly enroll in the section if it's their enrollment turn, or they can save the course and enroll it later. Finally, they can go to their saved courses and when it's their enrollment turn, they can enroll their courses from there.

By following this wireframe and user flow we can conclude that de implementation satisfies the user stories and requirements.



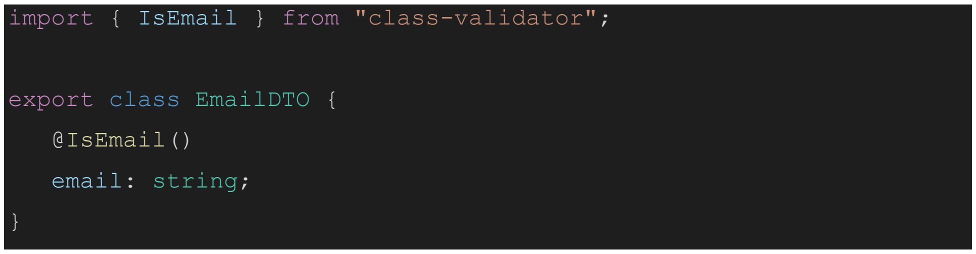
b. Verification Backend

i. CORS

Since in a API anyone can make requests we need to limit them for only those coming from the domain *.upr.edu(where * means any subdomain). We implement this using CORS, this will only accept the HTTP Methods(Ex: POST, OPTIONS, GET, PUT, DELETE, PATCH) that we choose and come from the domain of our choice. In this case *.upr.edu.

ii. Pipes and Class Validators

We need to have control of the values that we are using inside the code of our API. That's why we need to validate the values that are being sent to us first before using them in our code to query the Database. For that we use the "class-validator" library. This library will allow us to validate the values that we specify. For example; if we want to validate in the /login route that the value of "email" inside the body is actually an email we create a DTO(Data Transfer Object) by creating a typescript file that export a class, inside the class with create a email variable and assign a decorator to it. This decorator is imported from the "class-validator" library and is the one who will be in charge of validating the values inside that variable. This is how it will look:



```
import { IsEmail } from "class-validator";


export class EmailDTO {
   @IsEmail()
   email: string;
}
```

Once we create our DTO, inside our route controller we tell our route that the body of our post request will be of type “EmailDTO”. By doing this if we get sent a random value inside the email key like “123” instead of an actual email the API will respond with a HTTP 400 Bad Request error and will not execute anything unless the user sends a valid email. That is why using class validators and pipe is crucial for creating an API.

iii. API Gateway throttle

From AWS we have the API Gateway throttle. This tool will allow us to prevent the API from being overwhelmed by too many requests, Amazon API Gateway throttles requests to our API using the token bucket algorithm, where a token counts for a request. Specifically, API Gateway sets a limit on a steady-state rate and a burst of request submissions against all APIs in your account, per Region. In the token bucket algorithm, the burst is the maximum bucket size.

When request submissions exceed the steady-state request rate and burst limits, API Gateway fails the limit-exceeding requests and returns 429 Too Many Requests error responses to the client. Upon catching such exceptions, the client can resubmit the failed requests in a way that is rate limiting, while complying with the API Gateway throttling limits.

C. Validation

The current enrollment system is trouble for every person that forms the university: from the students to the professors to the administration. Students are indeed having a bad experience with the current enrollment process. This is why a betterment of the system is required, starting with eliminating the old terminal-based view and incorporating a nice-looking, easy-to-use UI. This means a better and simpler navigation system that allows students to get quicker and more intuitively (and easier) their corresponding courses. It is not stable (usually confirmation dates are postponed) because we are relying on outdated applications and servers to do a job which requires much newer, user-friendlier and stabler software. It is not adequate to "fix" the system the day before or, inclusive, the same day as the enrollment process. Eliminating current infrastructure for a newer and better one is definitely on demand. There is a lot of proof and issues that confirm it, but the most recent one happened at the start of this semester, when no one was able to enroll a course because of low maintenance of the system. Then, after that issue was "fixed", which caused the enrollment dates to be delayed, a system failure happened, leading to another postponement, students were still without any course enrolled, and another "fix". This on occasions can cause panic and anxiety among students and faculty members. Meaning this that our solution to this problem is right on track. An updated and more user-friendly system could improve the lives and experiences of every partner/stakeholder concerning this project.