Test Plan

for

CAR: Course Analysis and Registration System

Version 1.0

Prepared by:

Akshat Totla
Juan Piñeros
Liyan Ibrahim

NYUAD

May 8, 2022

1. Introduction

The test plan is for CAR: Course Analysis and Registration system. The system is aimed at providing a one-stop place for students from universities to efficiently manage their academics - to track their academic progress, create four-year plans, and find new courses for upcoming semesters, while administrators can manage and edit the course information and requirements.

As our Software Requirement Specification document indicated, the CAR is a flexible system that is built for application by all universities making it a general-purpose tool that can be utilized by multiple customers and has significantly low development time for each new customer. This would make it a scalable software and serve as a solution to the need for such a system at most universities worldwide.

The demo software is our current implementation and we have personalized it for our first customer: New York University Abu Dhabi.

2. Acceptance/Validation Tests for All Use Cases

I. Four Year Plans

This use case allows the users to produce and download customizable four year plans. The four year plans will depend on the major, location and year but will be customizable for every student and keeps the student on track to graduate on time.

This use case test will consist of a two separate stage testing with authentication of user login as a pre-condition: 1. Checking if the user has entered any valid past courses taken according to the bulletin 2. Checking if the user enters a valid major according to the bulletin. The pass criteria chosen for these testing stages will be based upon if the function operates effectively and generates the correct four-year plan using the database information extracted from Schedge API or displays no result depending on validation of major or courses. Both would involve visibility of the four-year plan.

II. Course Search

This use case allows the users to browse and search for courses offered based on major, location and other information.

This use case test will consist of a four separate stage testing with authentication of

user login as a pre-condition: 1. Checking if all necessary tabs – year, semester, department, keyword – for search query are visible on the website 2. The drop-down button displays relevant information and users should be able to select only one option 3.The filters should display correct results based on the search query 4. All information should be displayed course description, professor rating and prerequisites for every class offered, and the current registration status for a course. The pass criteria chosen for these testing stages will be based upon if the function operates effectively and the search results are accurate to the query.

III. Degree Progress

This user case allows the users to see a degree progress report calculating how much of their degree (based on major/minor) is completed and how much needs to be done.

This use case test will consist of a four separate stage testing with authentication of user login as a pre-condition: 1. Retrieval of data about courses and major previously entered by the user is correct and selectively visible 2. The download button saves the report locally 3. As courses are added, the progress should be updated 4. The report information is expanded if the expand button is clicked and relevant information is displayed. The pass criteria chosen for these testing stages will be based upon if the function operates effectively and generates the correct degree progress report depending on the variables provided. This would involve visibility of the degree progress report.

IV. Registration Planning

This user case allows the users to select their major, and courses already taken, and see recommendations for classes to take in the coming semester and see how those fit into their 4-year plans depending upon different requirements.

The test for this use case is done concurrently with Four Year Plan use case, where the same tests will be done for displaying the recommendations of courses to take.

V. Admin Console

This user case allows the users to add or modify different information for all the users including course information, course requirements, and add/remove users while also controlling year progression.

This use case test will consist of a four separate stage testing with authentication of

user login as a pre-condition: 1. Retrieval of data about course information sorted by the majors and visible 2. The information should be expanded when expand button is clicked 3. The user should be able to create edits when the edit button is clicked 4. The edits made should be updated on the database and reflected on the webpage. The pass criteria chosen for these testing stages will be based upon if the function operates effectively and update data in the database depending on the information edits provided. This would involve visibility of all course information.

3. Testing Approach

The testing of the system was done by the software development team and was simultaneous testing was done throughout the development process. We decided to use this approach after learning about the Agile SCRUM technique in the class to be utilized in case when there are time restrictions to development process, and requires the implementations work effectively as they go into usage. Using this approach has helped us save time by not dedicating significant time to only testing or the updates required from a failed test case as we were able to perform that parallel to the software development.

The thinking process behind choosing to conduct testing by the entire software development team was to ensure that the core use cases are tested thoroughly, and utilizing the entire team would be on board and wouldn't miss on testing outcomes so they can be immediately modified in the development of different, dependent functions of the system in case of bugs. Additionally, all the team members were able to get a thorough understanding of how the program works internally and save time in fixing bugs, highlighting the Agile SCRUM Model.

4. Resources

The testing of the system was carried out by running the software files on a local machine. No cloud servers were used for testing purposes.

Testing Environment Resources:

- Local machine with website and application files
- Web browser loading HTML & CSS files, pointing to file paths
- Python IDE for running the application files and implementing the backend database

The testing was done by the entire software development team which is consists of:

- Akshat Totla
- Juan Piñeros

- Liyan Ibrahim

Multiple other resources were used by software development team members including software tools and physical hardware:

- GitHub
- Google Drive
- Laptop Computers

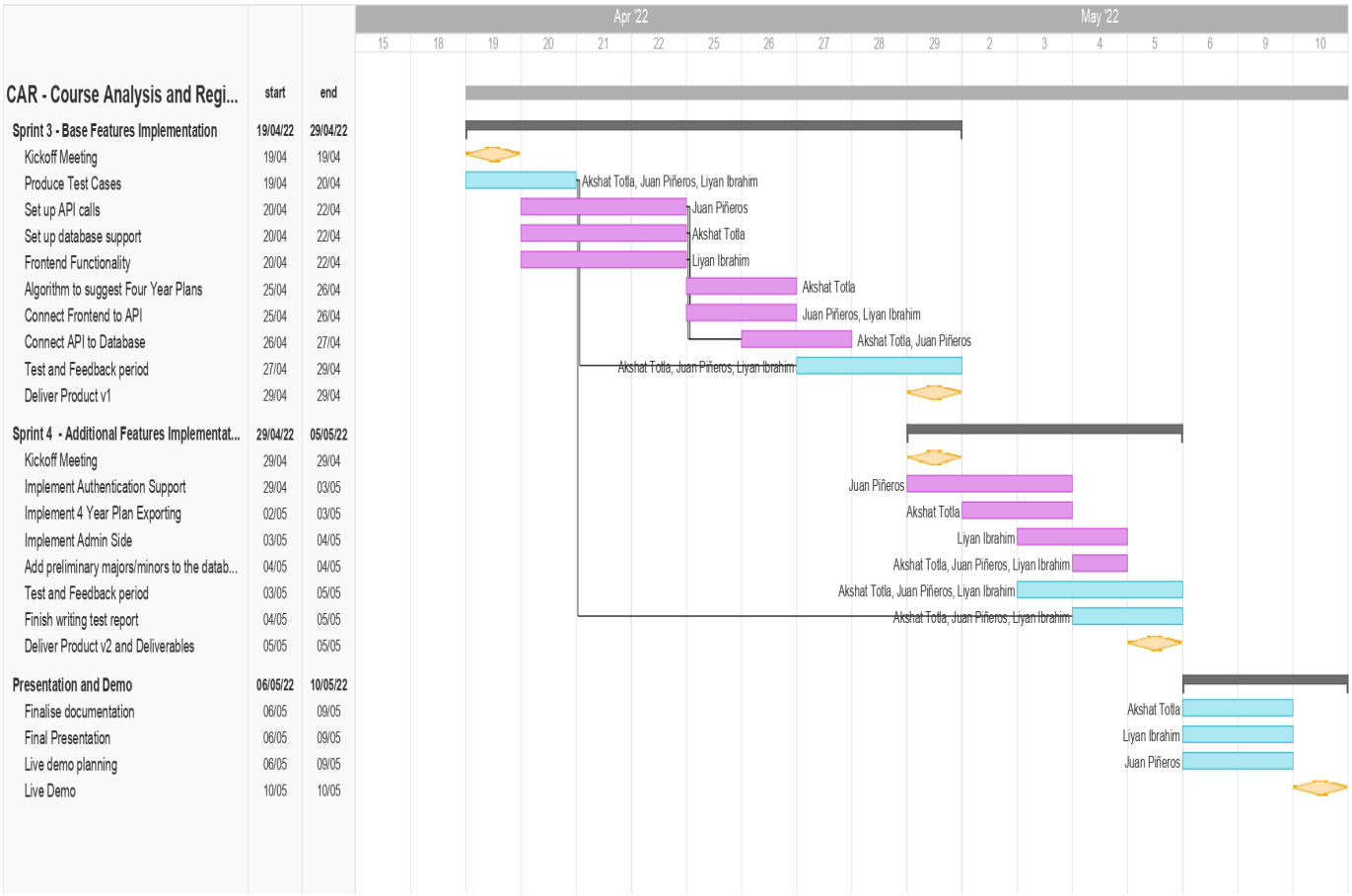
The time required to finish the testing process would be 5 days in total:

- Firstly, 3 days to finish the initial testing documentation (including the Test Plan and Test Cases)
- Secondly, 2 days to carry out and finish the actual testing on the system as included in the testing documentation

Refer to the Section 5 for detailed information on the timeline.

5. Schedule

We've attached the Gantt Chart used for timeline and scheduling of Implementation and Testing stage of the system.



6. Additional Notes

For failed test results and bugs we noted, refer to the Test Result document.