



Knight Foundation School of Computing and Information Sciences

Fall 2022 Senior Design Project Course Evaluation System

Team member(s): Gerardo Parra, Jose Milanes, Artur Vorojeykin, Daniel Nunez, Amani Hunt, Mariela Torres Zuniga, Shaoting Wu

Product Owner: Eric Johnson, Nagarajan Prabakar

Instructor: *Masoud Sadjadi*



Problem Description

The Course Evaluation System (CES) is an online computer software system designed to conduct course evaluations at the end of every semester. The system provides students with a list of courses provided by the School of Computing and Information Sciences that are receiving evaluations and for which they are registered; after the evaluation period is over the system generates reports maintaining the participant's anonymity, and then it allows administrators to look at the generated reports.

The Course Evaluation System (CES) was first developed in 2006 by the KFSCIS web development team and was used for more than fifteen years at the end of each term. This classic package was built on a Postgres database with several Perl modules to collect student course evaluation data and to generate reports. This package is not modular enough to classify the collected data according to student majors and to generate reports specific for each KFSCIS degree program.





Purpose of New System

The purpose of the new system is to Design a new CES system with a modular design approach that would consist of:

- Secure web interface with FIU credentials for students (provide course outcome evaluation), faculty and administrators (access reports generated from the CES data).
- A database to host the student assessment data for every term- course-major without any inconsistencies and to support the report generation & ad hoc queries with GUI.
- Report generators to generate reports at various granular levels for a given group (term/year/course) with an extension feature to add new plugins.
- Documentation and help instructions for each group of users (student, faculty, administrator).



Problem Management

This project was developed in a semester with the following timeline:

7 Sprints

1st Sprint:

- ☐ Meeting and coordinating with project owner.
- ☐ Worked on getting familiar with the technologies being used for the new version of CES and formalizing project goals
- ☐ Get familiar with current CES applications and documentations.
- ☐ Create GitHub repo and backend in Django

2nd Sprint:

- ☐ Work on Database design
- ☐ Work on Serializers, Views, and Models

3rd Sprint:

- ☐ Work on Database actions (Methods available to users based on role and view)
- ☐ Initiate work on Authentication using Django
- ☐ Start creating scripts to add data to database

4th Sprint:

- ☐ Improve current Database Models
- ☐ Continue working on Database actions
- ☐ Continue working on Authentication
- ☐ Begin working on front-end with React

5th Sprint:

- ☐ Create an evaluation page, for students to make evaluations
- ☐ Work on frontend Student page and Evaluation page
- ☐ Work on creation of new database actions requested by frontend work
- ☐ Create Login page and Faculty page for faculty members

6th Sprint:

- ☐ Finalized Database model Improvements
- ☐ Finalized front-end work
- ☐ Began planning for final deliverables

7th Sprint:

- ☐ Work on Final Deliverable: Documentation
- ☐ Work on Final Deliverable: Poster
- ☐ Work on Final Deliverable: Presentation
- ☐ Work on Final Deliverable: Videos



User Stories

User Story #1

As a product owner, I want to have everybody access to the resources needed so that we are ready to start to work on the project

Acceptance Criteria:

- Set up a meeting with the product owner and go over project
- Produce a product backlog

User Story #2

As a developer, I want to familiarize myself with the tools needed so that we can effectively implement what is asked of us.

Acceptance Criteria:

- Set up an environment (GitHub repository) for development.

User Story #3

As a team player, I want to understand the scrum process so that I can collaborate with my team and produce the best product possible.

Acceptance Criteria:

- Attend meetings and submit documents at their appropriate times.

User Story #4

As a team player, I want to know the best time to congregate with the rest of my team so that we can all speak to each other and avoid rescheduling often.

Acceptance Criteria:

- Set up a when2meet and have everyone input their available times.

User Story #5

As a developer, I want to know my position in developing the product so that no redundant work is done.

Acceptance Criteria:

- Assign roles and responsibilities

User Story #6

As a developer, I want to review the current system.

Acceptance Criteria:

- About 2 weeks to review the current system



User Stories

User Story #7

As a developer, I want to start designing the database.

Acceptance Criteria:

- Implement ER diagram for the design of the database

User Story #8

As a developer, I want to create a basic project in Django to start developing.

Acceptance Criteria:

- Create a Django project.

User Story #9

As a developer, I want to create the models.

Acceptance Criteria:

- Create models for each table from the database design.

User Story #10

As a developer, I want to create the respective serializers.

Acceptance Criteria:

- Create serializers for each model.

User Story #11

As a developer, I want to create the respective views.

Acceptance Criteria:

- Create the views for each model.

User Story #12

As a developer, I want to start creating the scripts.

Acceptance Criteria:

- Create a modular CLI in Python
- Create a modular CLI in Bash

User Story #13

As a developer, I want to start creating authentication.

Acceptance criteria:



User Stories

User Story #14

As a developer, I want to create some rest actions.

Acceptance criteria:

- Create possible actions needed in future development.
- Create actions for report gathering..

User Story #15

As a developer, I want to continue creating the scripts with the new database model.

Acceptance Criteria:

- Modify and create scripts for new database design.

User Story #16

As a developer, I want to finish creating authentication.

Acceptance criteria:

- Terminate with token-based authentication.

User Story #17

As a developer, I want to continue creating more actions.

Acceptance criteria:

- Create more actions for report gathering.

User Story #18

As a developer, I want to create a frontend project.

Acceptance criteria:

- Create a frontend project in React

User Story #19

As a user (student), I want to be able to login.

Acceptance criteria:

- Be able to enter credentials and being authenticated

- **User Story #20**

As a user (student), I want to be able to see available surveys.

Acceptance criteria:

- Be able to see available surveys after the student is authenticated.



User Stories

User Story #21

As a user (student), I want to be able to take an evaluation.

Acceptance criteria:

- Be able to click on an available survey and do the evaluation.
- Be able to know if I missed some question with an error message in red color.
- Be able to see some confirmation if the evaluation is submitted successfully.

User Story #22

As a user (student), I want to be able to use available pages with no errors and with a good design.

Acceptance criteria:

- Improve design in available pages.
- Improve logic in available pages.

User Story #23

As a user (faculty), I want to be able to login, and be authenticated and redirected to a faculty page.

Acceptance criteria:

- Create a login page for the faculty site with authentication.

User Story #24

As a user (faculty), I want to be able to gather reports by different ranges like semester, course, and year.

Acceptance criteria:

- Create a faculty page to gather reports.
- Gather reports by course, semester, and year.

User Story #25

As a developer, I want to continue creating more actions for report gathering requested by frontend developers.

Acceptance criteria:

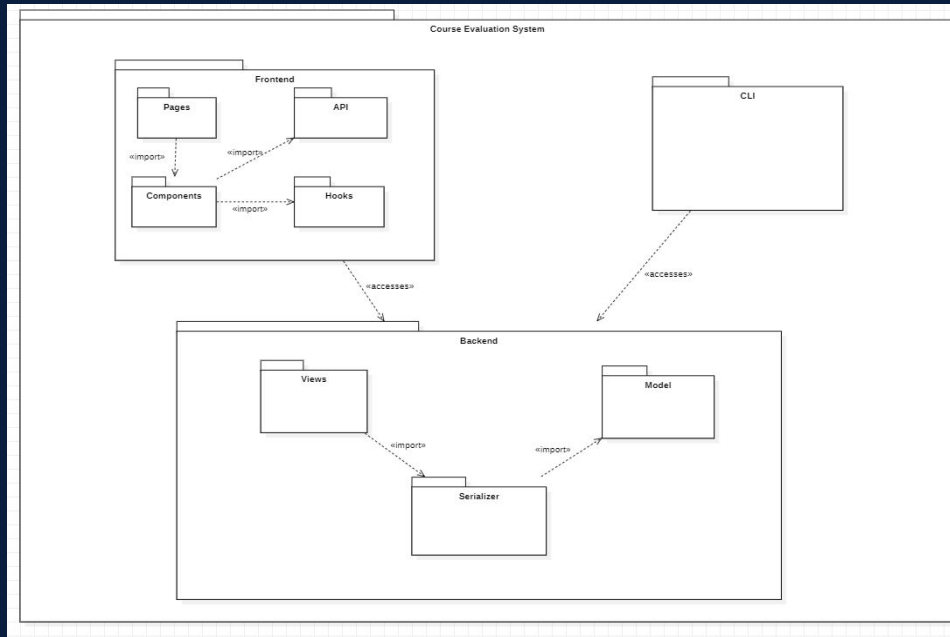
- Create actions to gather reports by course, semester, and year.

User Story #26

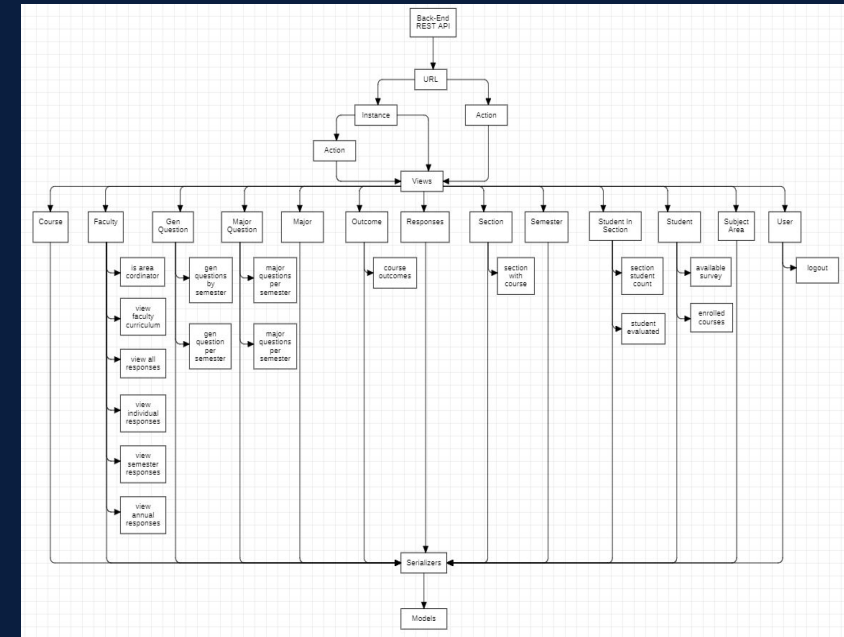


System Architecture

System Package

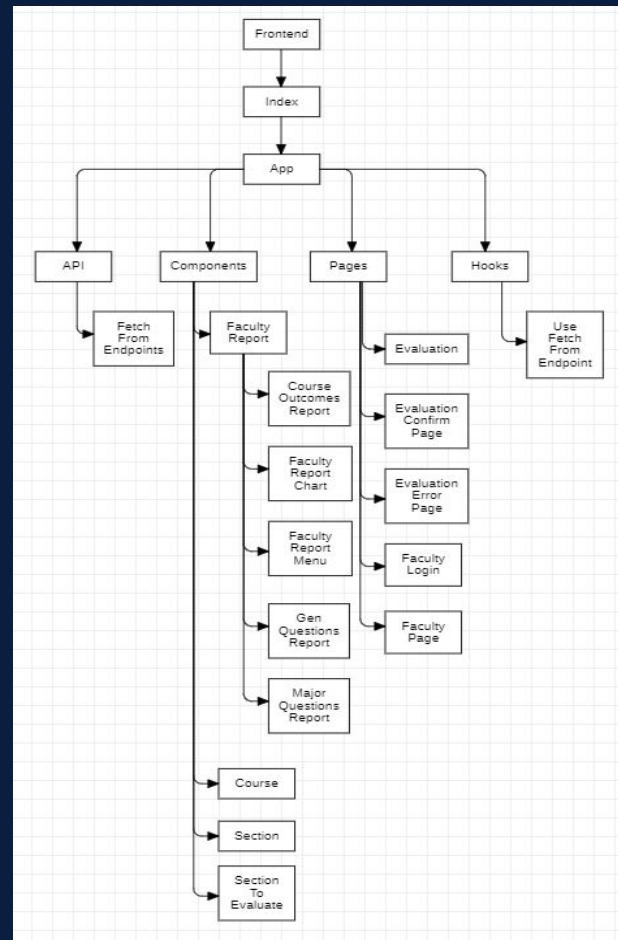


Backend system decomposition

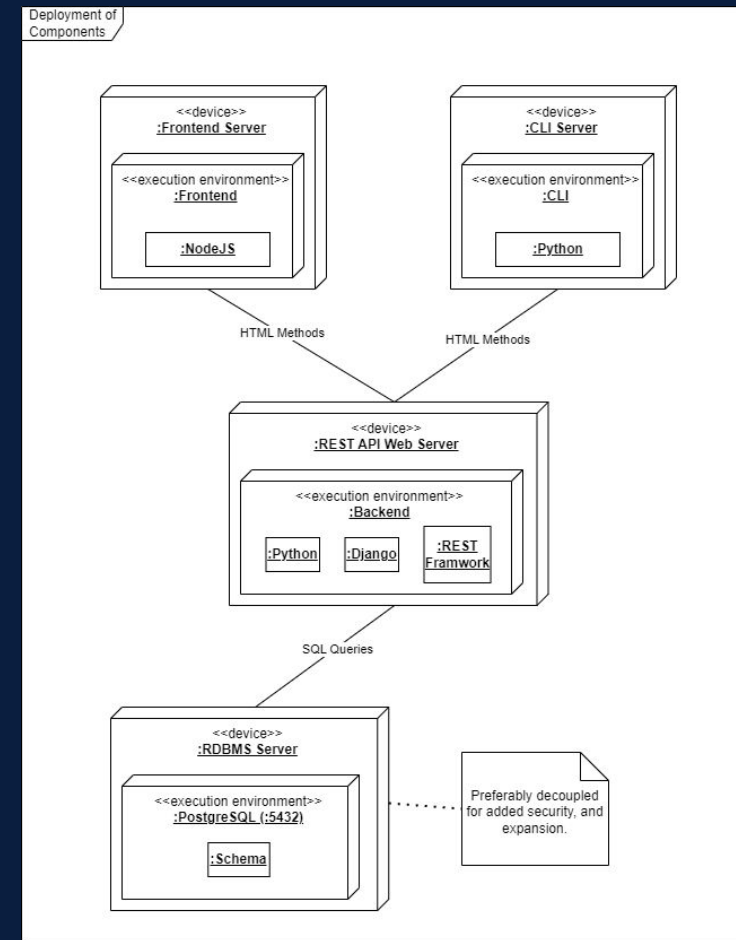


System Architecture contd...

Frontend system decomposition



Deployment Diagram



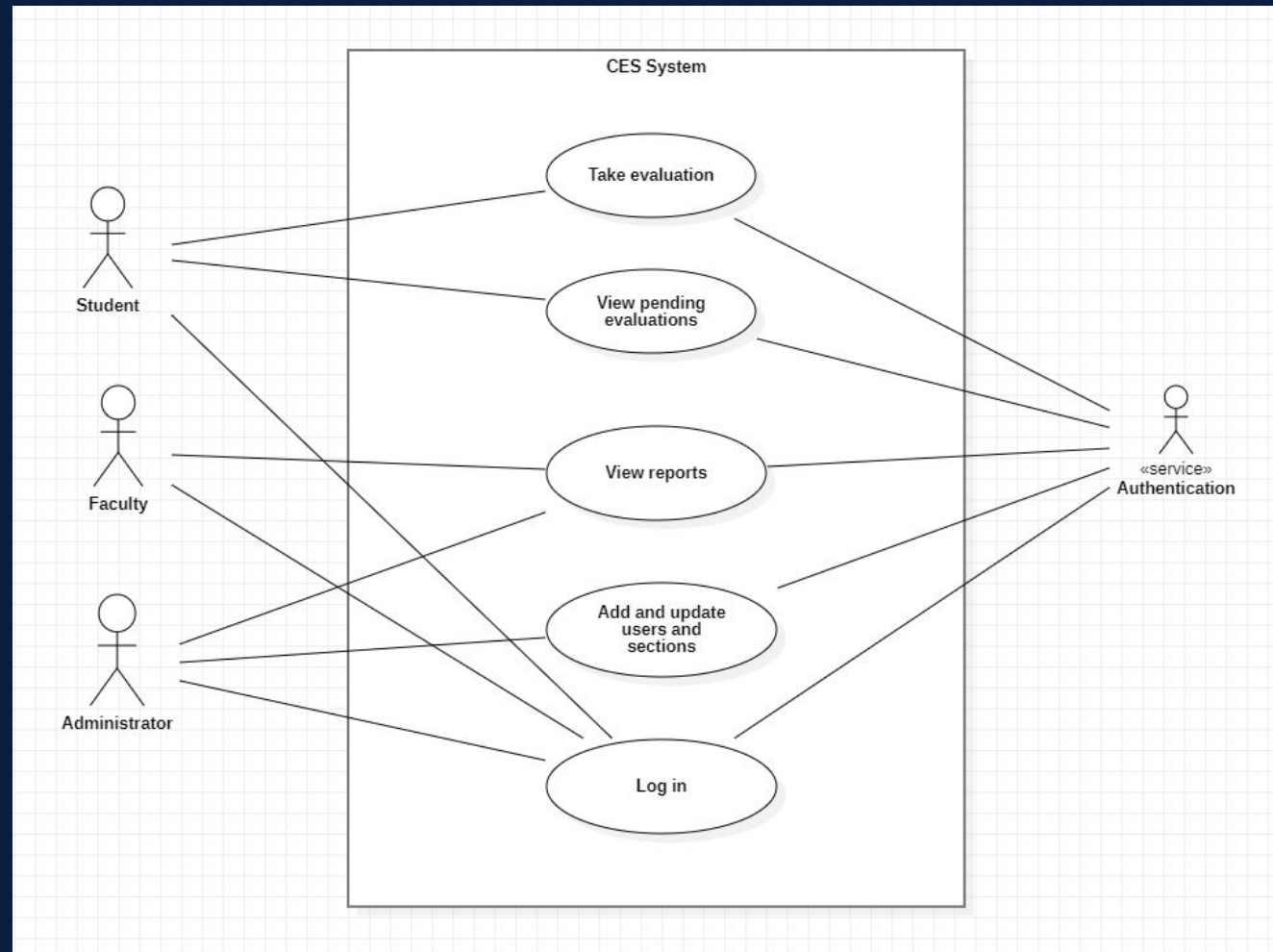
System Design Development

The architecture of the Course Evaluation System consists of three main components, the frontend, CLI, and the backend:

- The backend component is a REST API which is hosted by a web server. Because of this feature, a frontend can be developed in any framework/language that can do web requests. The backend reacts to different requests (GET, PUT, PATCH, OPTIONS) and depending on the type of the request, it generates different SQL queries that can create, get, or modify entries in the PostgreSQL database.
- The frontend component uses React to allow for rapid development of the UX. It communicates with the backend using the fetch API, which hits the specific endpoints depending on what needs to be rendered.
- The CLI component is simply a Python script that uses the requests package to communicate with the backend. The input to the CLI will depend on the desired command from the admin; creating faculty, courses, sections, majors and students, will require a CSV file with the proper format as input; modifying faculty, courses, and sections, require the correct string as input.

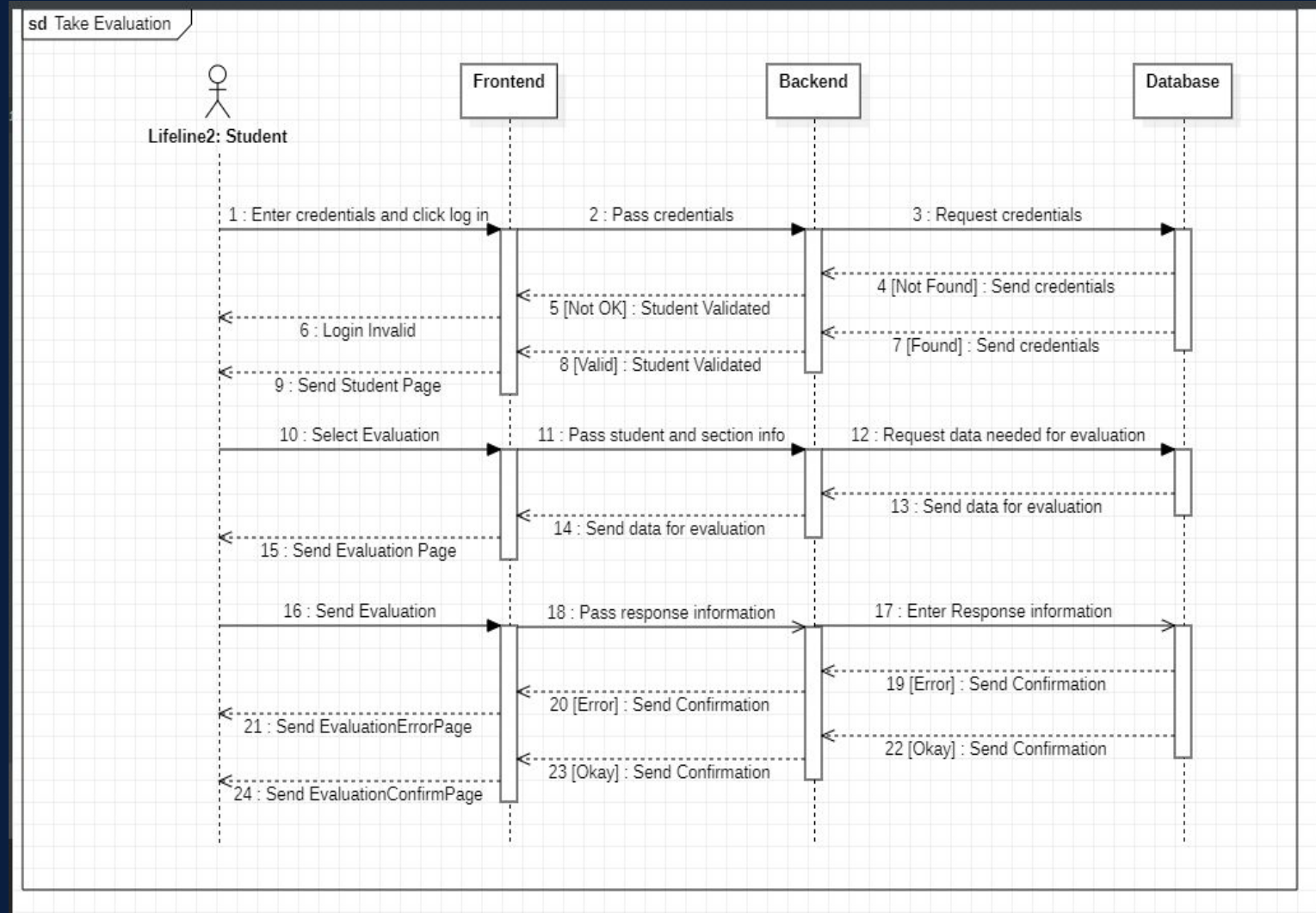


Use Case Diagram

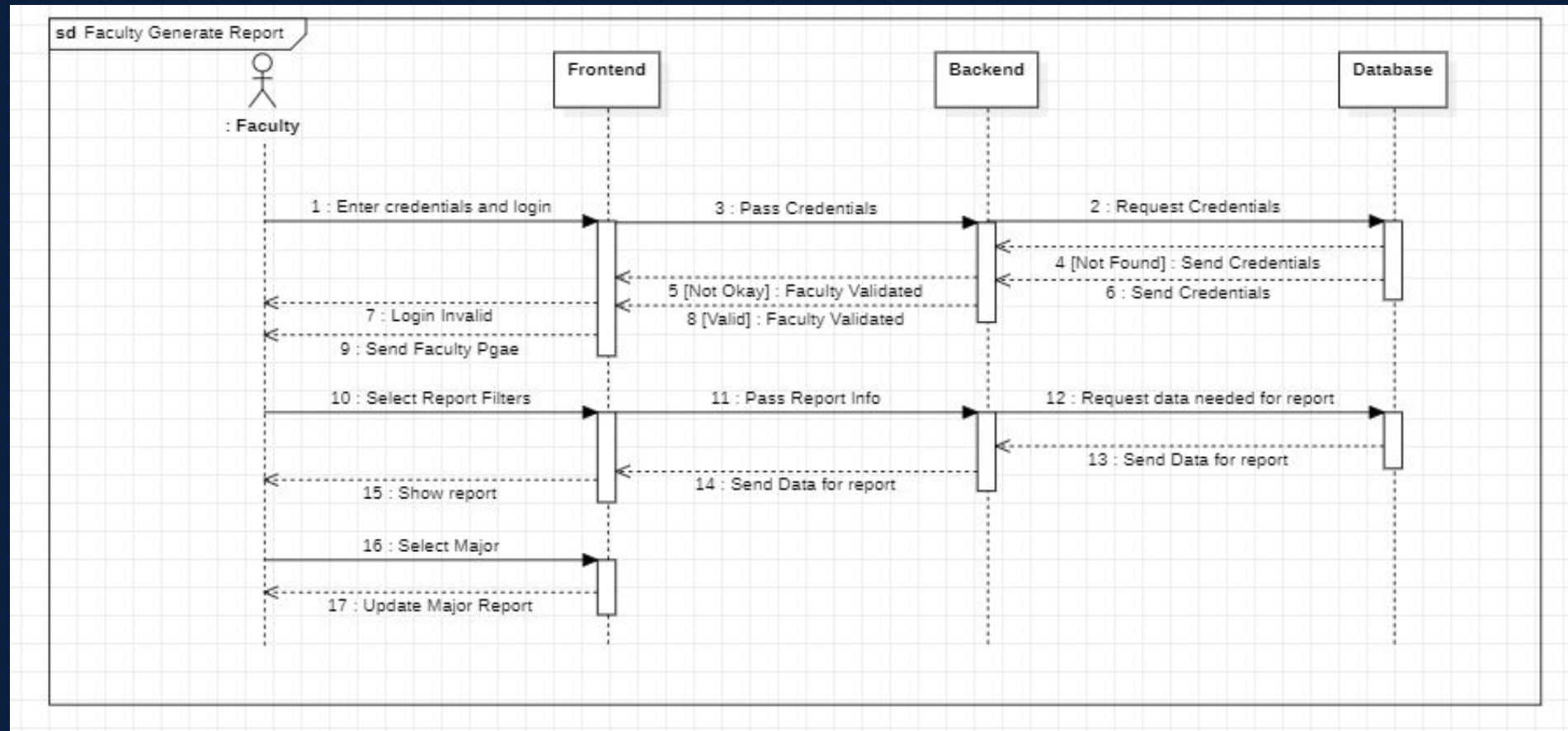


Sequence Diagram for Course Evaluation System

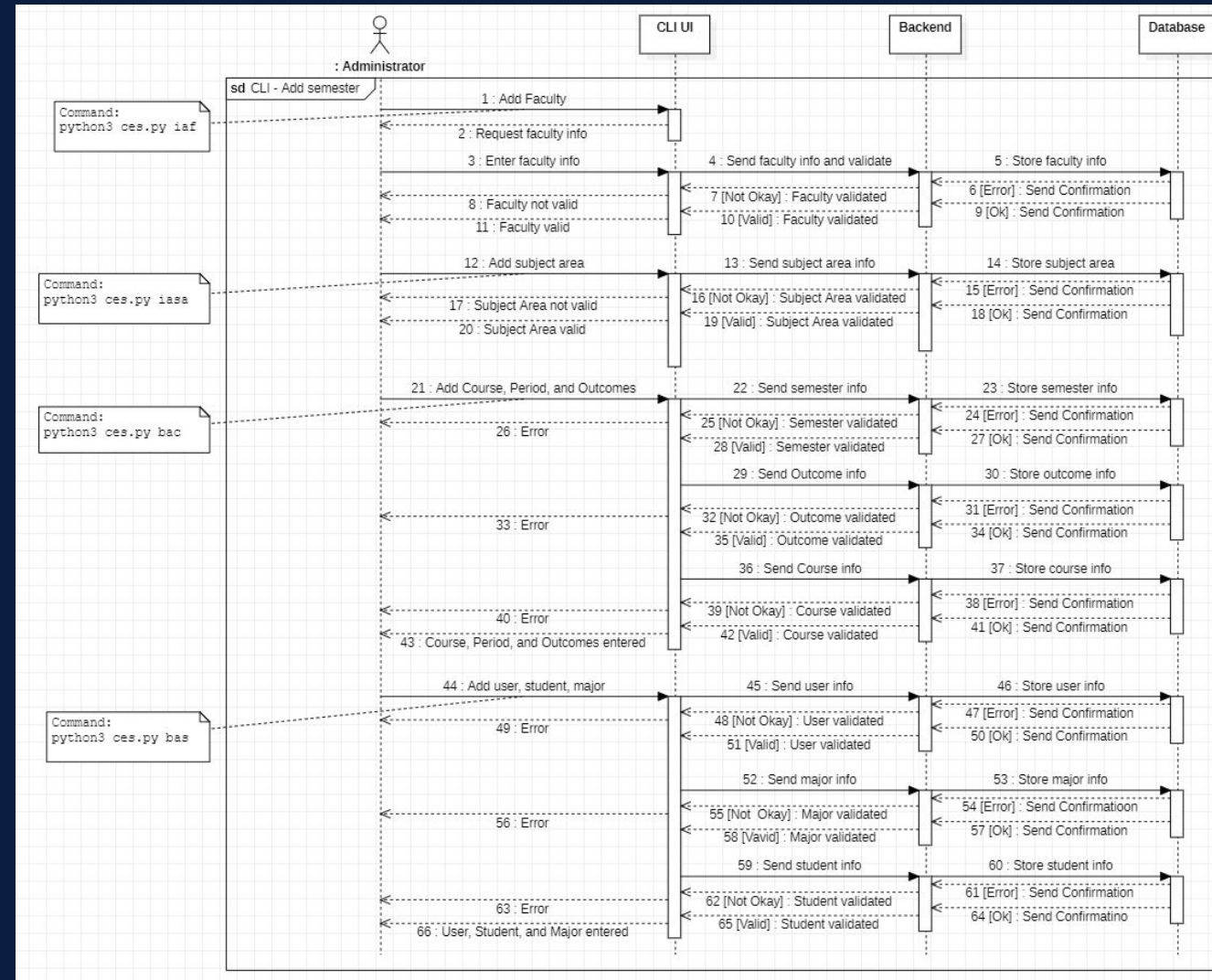
Sequence Diagram – Take Evaluation [Student]



Sequence Diagram – Retrieve Report [Faculty]



Sequence Diagram – Enter User & Section [Admin]



Database Models, Views & Serializers

```
from django.db import models
from .outcome import Outcome
from .subject_area import SubjectArea
from .semester import Semester

class Course(models.Model):
    number = models.CharField(max_length=10)
    name = models.CharField(max_length=35)
    outcomes = models.ManyToManyField(Outcome, blank=True)
    subject_area = models.ForeignKey(
        SubjectArea, on_delete=models.CASCADE, blank=True, null=True)

    semester = models.ForeignKey(Semester, on_delete=models.DO_NOTHING)

    def __str__(self) -> str:
        return f'{self.number} {self.name}'

    class Meta:
        ordering = ['number']
```

```
from rest_framework import serializers
from ces_fall_rest_api.models.course import Course

class CourseSerializer(serializers.ModelSerializer):
    class Meta:
        model = Course
        fields = '__all__'
```

This code example shows how the course Models Views and Serializers were created using Django

```
from ces_fall_rest_api.models.course import Course
from ces_fall_rest_api.serializers.course_serializer import CourseSerializer
from rest_framework import viewsets
from rest_framework.permissions import IsAuthenticated

class CourseViewSet(viewsets.ModelViewSet):
    queryset = Course.objects.all()
    serializer_class = CourseSerializer
    #permission_classes = [IsAuthenticated]

    def get_object(self):
        lookup_args = ['number', 'year', 'month']

        if all(arg in self.kwargs for arg in lookup_args):
            return Course.objects.get(number=self.kwargs['number'], semester__year=self.kwargs['year'], semester__month=self.kwargs['month'])
        else:
            return super().get_object()
```



Screenshots of Frontend

Student Page

The first screenshot shows the 'Course Evaluation System' login page for students. It features the FIU logo, a login form with fields for 'Username' and 'Password', and a 'LOGIN' button. Below the login form, there is an 'About' section and an 'Instructions' section.

The second screenshot shows the 'Sections to Evaluate' page. It displays a table with columns for '4567891', 'FirstName LastName', and 'BS Information Technology'. The table contains one row with the value '45677 COP 1101 RVV Information Tech'.

The third screenshot shows the 'Part A: Survey of Course Delivery' page. It contains five questions, each with five radio button options: 'I agree strongly', 'I agree moderately', 'I am not sure', 'I disagree moderately', and 'I disagree strongly'. The first three questions have a green checkmark next to the 'I disagree strongly' option.

Faculty Page

The screenshot shows the 'Course Evaluation System' Faculty Login page. It features the FIU logo, a login form with fields for 'Username' and 'Password', and a 'LOGIN' button. The background of the page shows a blurred image of a building and palm trees.

The screenshot shows the 'Survey of Course Delivery (17 evaluations received)' page. It displays a table with columns for '3333333', 'Test Faculty', and 'Instructor'. The table contains one row with the value '2023 - COP 1101 INFORMATION TECH - SPRING - 45677 RVV - GENERATE REPORT'.

Survey of Course Delivery (17 evaluations received)				
This is question 1				
I strongly agree 18%	I agree moderately 12%	I am not sure 12%	I disagree moderately 18%	I disagree strongly 41%
Average responses 2.47				
This is question 2				
I strongly agree 18%	I agree moderately 18%	I am not sure 29%	I disagree moderately 18%	I disagree strongly 18%
Average responses 3				
This is question 3				
I strongly agree 18%	I agree moderately 0%	I am not sure 41%	I disagree moderately 24%	I disagree strongly 18%
Average responses 2.76				
This is question 4				
I strongly agree 24%	I agree moderately 18%	I am not sure 18%	I disagree moderately 24%	I disagree strongly 18%
Average responses 3.06				



Screenshots of backend

Api Root

Api Root

The default basic root view for DefaultRouter

GET /

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "faculty": "http://127.0.0.1:8000/faculty/",
  "outcomes": "http://127.0.0.1:8000/outcomes/",
  "responses": "http://127.0.0.1:8000/responses/",
  "sections": "http://127.0.0.1:8000/sections/",
  "courses": "http://127.0.0.1:8000/courses/",
  "gen_questions": "http://127.0.0.1:8000/gen_questions/",
  "users": "http://127.0.0.1:8000/users/",
  "students_in_sections": "http://127.0.0.1:8000/students_in_sections/",
  "students": "http://127.0.0.1:8000/students/",
  "subject_areas": "http://127.0.0.1:8000/subject_areas/",
  "major_questions": "http://127.0.0.1:8000/major_questions/",
  "semesters": "http://127.0.0.1:8000/semesters/",
  "majors": "http://127.0.0.1:8000/majors/"
}
```

Api Root / Faculty List

Faculty List

A viewset to get the all the Faculty objects
This class has a token authentication.
I also has several actions to retrieve reports from database:
1- is_area_coordinator
2- view_faculty_curriculum
3- view_all_responses
4- view_individual_responses
5- view_semester_responses
6- view_annual_responses
7- get_sections
8- current_courses

Args:
viewsets (_type_): viewsets class

Returns:
type: Response in JSON format

GET /faculty/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "count": 4,
  "next": null,
  "previous": null,
  "results": [
    {
      "user": {
        "username": "hooter@25",
        "pid": 123987,
        "first_name": "harry",
        "last_name": "h123987p",
        "password": "pbkdF2_sha256$390000$Kxc8tb7EQugpr1ac1Q1Zf$THwqTYPLaws1Ju18dZ877867AUK1PRIZHT0VhVank-",
        "email": null
      }
    }
  ]
}
```

Api Root / Section List

Section List

A viewset to get the all the Section objects
This class has a token authentication.
It also has an action that retrieve a specific section with the correct course name.

Args:
viewsets (_type_): _description_

Returns:
type: _description_

GET /sections/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 6,
      "reference": 45678,
      "name": "RW",
      "course": 24,
      "instructor": "testfaculty1",
      "semester": 3
    }
  ]
}
```

Api Root / Faculty List / Faculty Instance

Faculty Instance

A viewset to get the all the Faculty objects
This class has a token authentication.
I also has several actions to retrieve reports from database:
1- is_area_coordinator
2- view_faculty_curriculum
3- view_all_responses
4- view_individual_responses
5- view_semester_responses
6- view_annual_responses
7- get_sections
8- current_courses

Args:
viewsets (_type_): viewsets class

Returns:
type: Response in JSON format

GET /faculty/testfaculty1/

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "user": {
    "username": "testfaculty1",
    "pid": 1243121,
    "first_name": "Test",
    "last_name": "Faculty",
    "password": "pbkdF2_sha256$390000$ASb8vYvWd8gBmgKC2YQPtJ$Sudw0mZDXB1loFCPKErQwGly3puEQzobVQbX0IMYV+",
    "email": null
  },
  "role": "instructor"
}
```



Test Cases Examples

Test CES GUI for Students

Test Case ID: Display_Login_Student

Purpose: To display CES GUI login page.

Setup: Activate virtual environment, run server and frontend, and access localhost:3000 on a web browser.

Input: Home.js file containing the basic GUI.

Expected Output: Login GUI on the localhost:3000 server.

Status: Pass

Test Case ID: Authenticate_Login_Student

Purpose: To authenticate a student through CES GUI login page.

Setup: Activate virtual environment, run server and frontend, and access localhost:3000 on a web browser.

Input:

username: *teststudent1*

password: *password1*

Expected Output: Http status response 200.

Status: Pass

Test Case ID: Display_StudentPage_Student

Purpose: To display CES GUI student page.

Setup: Activate virtual environment, run server and frontend, access localhost:3000 on a web browser, enter valid credentials for a student, and click login button.

Input:

username: *teststudent1*

password: *password1*

Expected Output: StudentPage GUI on the address localhost:3000/user with all evaluations rendered.

Status: Pass



Test Cases Examples

Test Case ID: StudentPage_Select_Evaluation

Purpose: To select an evaluation in StudentPage and redirect to EvaluationPage successfully.

Setup: Activate virtual environment, run server and frontend, access localhost:3000 on a web browser, enter valid credentials for a student, and click login button to be redirected to StudentPage.

Input: StudentPage.js file containing the GUI and logic.

Expected Output: EvaluationPage GUI on the address localhost:3000/evaluation.

Status: Pass

Test Case ID: EvaluationPage_Submit_CorrectForm

Purpose: To fill evaluation form in EvaluationPage and redirect to ConfirmationPage successfully.

Setup: Activate virtual environment, run server and frontend, access localhost:3000 on a web browser, enter valid credentials for a student, click login button to be redirected to StudentPage, select an evaluation to be redirected to EvaluationPage, and answer all questions.

Input: Evaluation.js file containing the GUI and logic.

Expected Output: ConfirmationPage GUI on the address localhost:3000/evaluation/confirm and response entered in database.

Status: Pass

Test Case ID: EvaluationPage_Submit_MissingFields

Purpose: To fill evaluation form in EvaluationPage and submit it with missed fields.

Setup: Activate virtual environment, run server and frontend, access localhost:3000 on a web browser, enter valid credentials for a student, click login button to be redirected to StudentPage, select an evaluation to be redirected to EvaluationPage, answer some questions, and leave at least one question (radio button) unanswered.

Input: Evaluation.js file containing the GUI and logic.

Expected Output: Error message indicating all missing questions in the same page.

Status: Pass



Test Cases Examples

Test CES API

Test Case ID: Test_All_CES_API

- **Purpose:** To test all CES API calls, since some API calls are required before testing GUI features.
- **Setup:** Activate virtual environment, run server, and access localhost:8000 on a web browser. Choose all GET API endpoints and call them. Choose all POST APIs endpoints and call them. Choose all PATCH APIs endpoints and call them. Choose all DELETE APIs endpoints and call them.
- **Input:** JSON files for GET, POST, PATCH, DELETE calls.
- **Expected Output:** Respective successful code response on each API call
- **Status:** Pass

Test CES GUI for Faculty

Test Case ID: PENDING

- **Purpose:** PENDING
- **Setup:** PENDING
- **Input:** PENDING
- **Expected Output:** PENDING
- **Status:** Pass

