Technical Specification

for

Student Project Management Systems

Prepared by Shabb03

Student Number: ******

22/02/2023

Table of Contents

Introduction

Overview:

The Web-Based Student Project Management System allows DCU students and DCU project supervisors to organize their proposals, files, projects, meetings, reports and deadlines within the system as opposed to working on each individual aspect separately and forgetting some important information in the midst of all the work spread out.

The system allows for an administrator to view the data on all students, professors and projects as well as set the deadlines, create a demonstration timetable instantly and authorize the entry of professors into the system for security reasons.

Students get proposal form which can be saved as a draft for further editing then submitted to a project supervisor where they will wait for the professor's approval and be notified via email if their proposal was accepted or rejected

Both Students and Professors can request a meeting, accept/reject a meeting proposal and view their meeting history

Professors can view a project status report for the students they are supervising and view each project in more detail then mark the project with grades for each aspect of the project. Professors can also add available time slots for project demonstration

The admin can authurize the entry of professors into the system, view all of the registered students and professors, assign students without a project supervisor to one and create a demonstration timetable with one click.

Glossary:

Modal = Confirmation box
Object(s) = Database object data

System Architecture

Architecture Overview Diagram

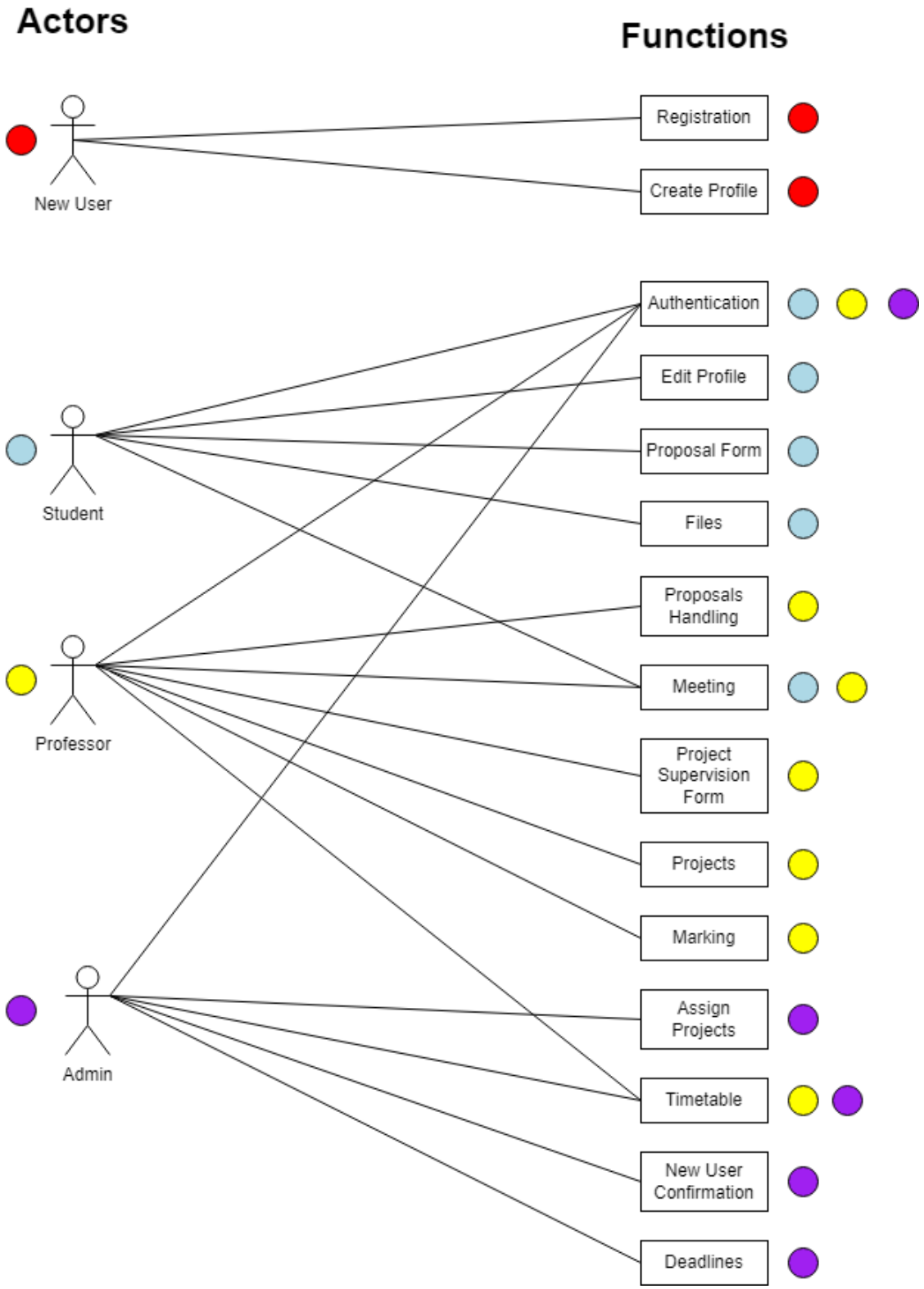

Figure 1, Architecture Overview Diagram

Actors on the left side are color coded for readability

Functions on the right are color coded to display which actors can access which functions for improved readability

New Users can choose to register and create a profile to become a student actor or professor actor

All actors have access to authentication to be able to use the system

Student actors have access to use their profile component, proposal component, files component and meeting component

Professor actors have access to use the proposal handling component, meeting component, project supervision form component, projects component, marking component and timetable component

Administrator actors have access to use the assign projects component, timetable component, new user confirmation component and deadlines component

This is an overview of the actors and the components they can use

Re-Used Components

CA298 Project: [1]

- Sign Up Function
- Login Function
- Access Token
- Serializer Blueprints
- Viewset Blueprints
- API url paths
- Frontend Index path blueprints

W3Schools:

- Password visibility function[2]
- Pie charts[3]

Programming Languages

- Django - Python web based framework, Backend
- NodeJS - Javascript cross-platform, runtime environment, Frontend
- Express - NodeJS web application framework, Frontend
- Bulma CSS - CSS open source framework with ready-to-use frontend components

High Level Design

Component Diagram

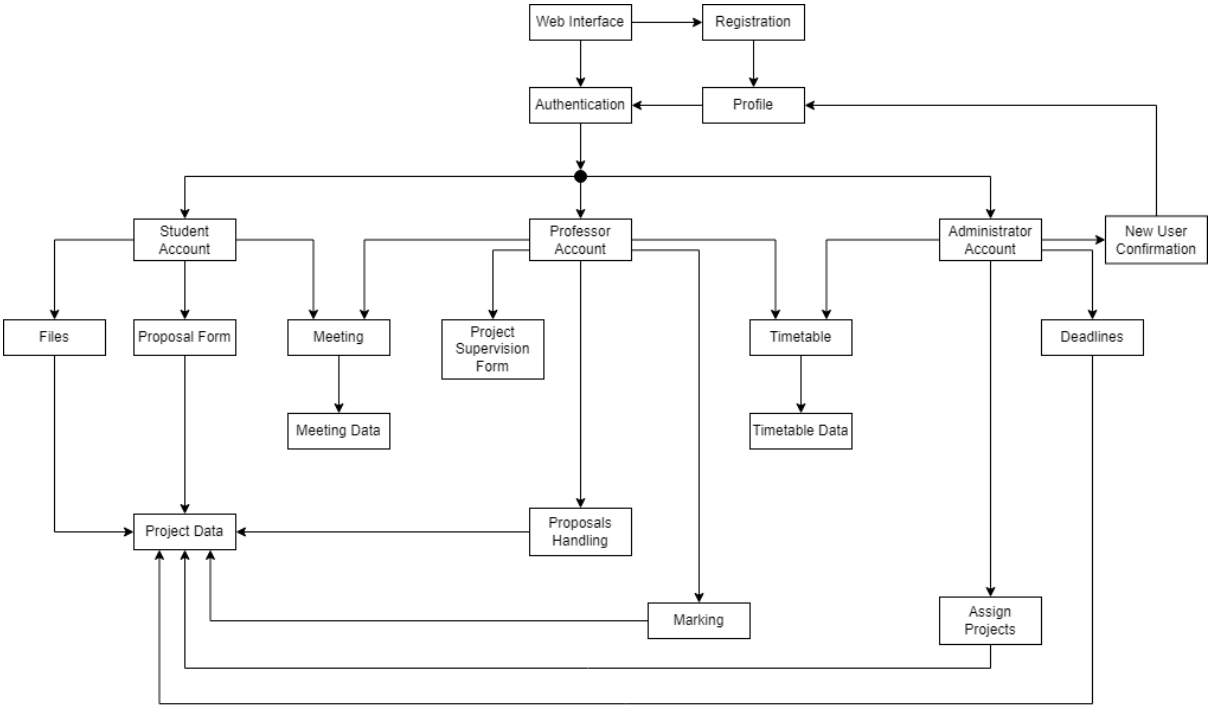

Figure 2, Component Diagram

Figure 2 shows the different components of the system and how they relate to each other

The web interface can either register the user and create a profile or proceed to authorisation, authorisation then relates to three components, student account, professor account and administrator account, each account has its own unique components that others cannot access

The student account can connect to the files component, proposal form component which then connect to the project data and meeting component which connects to the meeting data

The professor account can also connect to the meeting component which continues to the meeting data as well as the project supervision form component, proposals handling component and marking component which connect to the project data aswell

The administrator account can connect to the new user confirmation component which connects to the profile component, the deadlines component and the assign projects component which again also connect to the project data

Operational Diagram

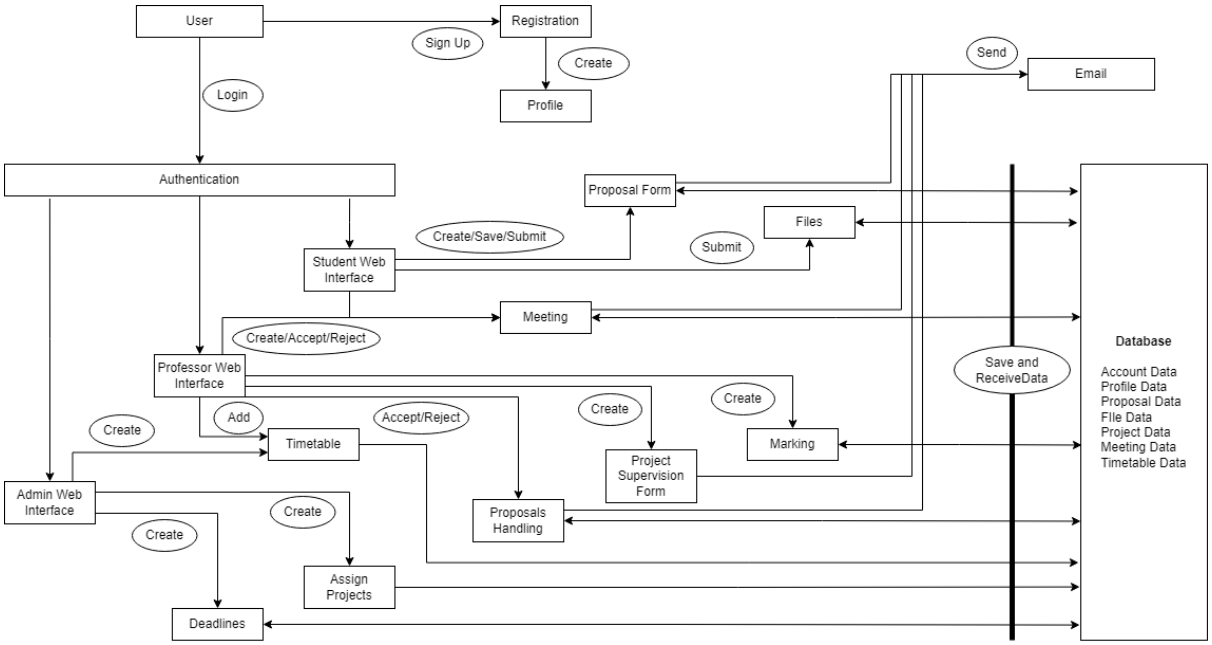

Figure 3, Component Diagram

Figure 3 shows the different components of the system and how they interact with each other and their environment

The user can go to registration and sign up then create a profile to become either a student or professor in the system or proceed straight to authentication which displays either the student web interface, professor web interface or admin web interface depending on the user's role

Each component receives the data from the database has an action that leads to it which then saves the data into the database, some components also send an email through the provided Gmail account

Dependancies

- Gmail, a password is generated from the chosen gmail account which is used to send emails
- FontAwesome, all icons are imported from https://fontawesome.com/

Problems and Resolution

Problem: Creating different user types and roles
Solution: Upon research, a solution was found to set a user type field with multiple choices to assign different user's with a user type of "New", "Student", "Professor" and "New_Professor"
Link:
https://simpleisbetterthancomplex.com/tutorial/2018/01/18/how-to-implement-multiple-user-types-with-django.html

Problem: Deleting a model migration
Solution: Upon research, a solution that successfully worked was to delete the latest migration in backend/migrations to fix the latest migration error
Link:
https://stackoverflow.com/questions/37729042/altered-django-model-i-provided-a-one-off-default-datetime-date-for-datefield

Problem: Sending Automated Emails:
Solution: Upon research, a helpful video was found on how to send emails through Django, however, as of May 30 2022, Google had disabled their option to allow less secure apps to use a gmail account, another solution was found to generate an app password to allow Django to use for sending emails
Links:
https://www.youtube.com/watch?v=xNqnHmXIuzU,
https://www.youtube.com/watch?v=uVDq4VOBMNM

Problem: Filtering for objects using foreign key values
Solution: Some objects required to be filtered based on a foreign key's value which took some research to learn how to do but it simply required two underscores, e.g
`Book.objects.filter(foreignkey__foreignkeyvalue="horror")`
Link:
https://stackoverflow.com/questions/1981524/django-filtering-on-foreign-key-properties

Problem: Creating serializer api requests
Solution: Unfamiliar with API requests and how they worked, this section on get requests and post requests took some days of research and mostly borrowing components from the CA298 full-stack project
Links:
https://www.django-rest-framework.org/
https://gitlab.computing.dcu.ie/sidhur2/2022-ca298-template
https://www.youtube.com/watch?v=cJveiktaOSQ&t=121s

Problem: Submitting files through an API post request
Solution: Files cannot be submitted through a JSON request so the body required a different input through the form-data post request
Link: https://www.youtube.com/watch?v=e13T3O0Iyvc

Problem: Sending a primary key as an API post request field
Solution: A primary key could not be sent as a post request as it would return an error so it required some special handling to allow a primary to be passed through as a field by simply adding one line

```
id=serializers.PrimaryKeyRelatedField(queryset=Model.objects.all()
)
```

Link:
https://stackoverflow.com/questions/37857600/how-to-use-primary-key-in-a-serializer-in-django

Problem: Retrieving a Student or Professor object primary key as a field
Solution: A function was added to each model to return a primary key as a field

```
def student_url(self):
        return self.pk
```

Link: https://openai.com/blog/chatgpt/

Problem: Allowing for the addition of an error404 page in the case of an incorrect url entered
Solution: A simple request was added at the end for any url's that did not match the urls above to be redirected to a custom page
Link:
https://stackoverflow.com/questions/6528876/how-to-redirect-404-errors-to-a-page-in-expressjs

Problem: Creating a search bar
Solution: A simple solution was found to get the table and hide all names that did not match the query
Link: https://www.w3schools.com/howto/howto_js_filter_lists.asp

Problem: Displaying a custom confirmation box to replace the default confirmation panel
Solution: The default confirmation panel did not go well with the Design of the Webpage but it's design could not be changed either so the correct method would be to create a custom confirmation box and luckily the Bulma CSS framework provided just that as a modal
Links:
https://bulma.io/documentation/components/modal/
https://stackoverflow.com/questions/22512536/js-confirm-refreshes-page-if-cancelled

Problem: Displaying a custom confirmation box using forms
Solution: Any button inside a form acts as a submit button even without the type field, to add a modal confirmation box was quite a challenge as that placed 3 buttons inside a form whereas even just one button would activate the form request, after a lot of testing, one solution worked where the modal was placed inside the form, the modal cancel button was changed to a textfield and the original form button was placed outside the form as to not trigger any events for the form

Problem: Creating pie charts
Solution: Upon research, an easy solution was found to display pie charts for showing data to the admin
Link: https://www.w3schools.com/js/js_graphics_chartjs.asp

Problem: Fetch requests not operating in order
Solution: Javascript unlike any other language did not work from top to bottom as most other languages would operate, this was shown using console.log() statements that showed some numbers would get printed but not in the right order, this proved to be challenging for fetching multiple fetch requests so fetch requests were not put inside functions to call but nested inside other fetch requests as to operate in the correct order

Installation Guide

Required Hardware and Software: Desktop OS
The system will work on any of the following Operating Systems:

- Windows
- Linux
- Mac
- Python
- Django
- Node + Express

Required Components

- SQLite3 Database
- Gmail

Step 1. Install Visual Studio Code
https://code.visualstudio.com/download
Open up VSCode and create a new terminal

Step 2. Install Python
Download https://www.python.org/downloads/

Linux: https://www.tutorialspoint.com/python/python_environment.htm
Windows: https://www.educative.io/answers/how-to-add-python-to-path-variable-in-windows
Mac: https://www.educative.io/answers/how-to-add-python-to-the-path-variable-in-mac

Step 3. Install Django
In your terminal type the following:

```
pip3 install django
```

Step 4. Install Django REST Framework
In your terminal type the following:

```
pip3 install djangorestframework
pip3 install markdown
pip3 install django-filter
pip3 install djangorestframework-simplejwt
pip3 install django-cors-headers
```

Step 5. Install Node
Linux: https://github.com/nodesource/distributions/blob/master/README.md
Windows: https://nodejs.org/en/download/
Mac:
In your terminal type the following:

```
xcode-select --install
```

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)
"

brew install node
```

Step 6. Install Express

Linux:

In your terminal type the following:

```
sudo npm install express -g npm install express-generator -g npm
install nodemon -g
```

Windows:

Use the link provided

https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/jj717276(v=ws.11)

Open an administrator command prompt and type the following:

```
npm install express -g npm install express-generator -g npm
install nodemon -g
```

Mac:

In your terminal type the following:

```
sudo npm install express -g npm install express-generator -g npm
install nodemon -g
```

Step 7. Create an Admin

In your terminal type the following:

`cd backend` (Make sure you are in the same directory with a file named manage.py)

```
python3 manage.py createsuperuser
```

Step 8. Add Changes

In your backend/backend/settings.py file

Go to the following and added your preferred urls or port numbers

```
CORS_ALLOWED_ORIGINS = []
```

Go to the following change to your preferred email and get the password here

```
EMAIL_HOST_USER = 'youremail@gmail.com'
EMAIL_HOST_PASSWORD = 'password'
```

Step 9. Deploy the system

In your terminal type the following:

`cd backend` (Make sure you are in the same directory with a file named manage.py)

```
python3 manage.py runserver {{PORT NUMBER}}
```

Change `{{PORT NUMBER}}` to your actual port number e.g 0.0.0.0:8000

```
cd ../frontend

npm start
```

References

[1] [https://gitlab.computing.dcu.ie/sidhur2/2022-ca298-template]
[2] [https://www.w3schools.com/howto/howto_js_toggle_password.asp]
[3] [https://www.w3schools.com/js/js_graphics_chartjs.asp]