Project No: A3070-181

A Web-based Project Management System:

NTUCollab

Submitted by: Ng Wai Kit

Matriculation No.: U1522836H

Supervisor: Prof. Chua Hock Chuan

School of Electrical & Electronic Engineering

A final year project report presented to Nanyang Technological University

in partial fulfilment of the requirements of the degree of Bachelor of Engineering

2019

Acknowledgements

I would like to extend my gratitude towards my supervisor, Professor Chua Hock Chuan, for his ongoing support and guidance throughout the course of my final year project. Thank you for believing in me and giving me an opportunity to take on your software engineering project. Your advice and experience in the subject matter had been invaluable to my progress.

I also would like to acknowledge my examiner, Professor Bi Guoan, for his time and effort in assessing my final year project.

Table of Contents

Abstract

NTUCollab is a web-based project management system built to facilitate students and supervisors in the Final Year Project (FYP). It serves as a project management tool that will help in conveying information and improve efficiency of communication between students and their supervisors.

In the school of Electrical and Electronics Engineering (EEE), every student must take on a FYP to graduate. It marks as a summary and a formal application to what they have learnt in their studies. Developing a system that aims to create a better platform and improve the process of FYP can potentially benefit every student.

NTUCollab is built on React, a frontend library in JavaScript, and Django, a python framework. It utilises SQLite as a database for data storage as well as CSS for styling. Other libraries such as date-fns is vital as a date utility library.

I have successfully developed a calendar module that displays an academic calendar and can manage tasks and reports. The calendar displayed corresponds to the assigned FYP of the student. Tasks are created by supervisors for students to complete and each task accepts a report. These reports consist of input fields such as “to-dos” and “done” which indicates the progress of the project.

Chapter 1: Introduction

This chapter contains the background, motivations, objectives and accomplishments of NTUCollab.

1.1 Background

Project-Based Learning (PBL) has been an increasingly popular method of assessment for students apart from regular examinations. In Singapore, educational institutions have supported PBL in the curriculum and students are familiar with graded assessments in the form of projects. Students are required to work individually or in groups to complete an assignment and usually involves research or problem solving. It encourages critical thinking, application of knowledge and supports individual learning. Hence, PBL is effective in tackling challenges faced by engineering undergraduates [1]. It gives students the opportunity to have a real-life adaptation to engineering problems and gain vital collaborative and communication skills. Furthermore, PBL has been distinguished as one of the most effective teaching frameworks for engineering majors [2].

The School of Electrical and Electronic Engineering (EEE) of Nanyang Technological University (NTU) adopts two project-based assessments for their full-time students, namely the Design and Innovation Project (DIP) and the Final Year Project (FYP). The DIP module requires EEE students in groups of 10 to complete a project amalgamating software, hardware as well as design. The FYP is a year-long project assigned to the students based on their field of study and/or interests. The FYP module would either be a group or an individual project. Both the DIP and FYP are compulsory to all students from EEE. In both cases, there would be a supervising professor who will guide and advice the students undertaking these projects.

1.2 Motivation

Project management systems are vital to successful and effective projects. The goals of an effective project management system are to determine objectives, list out tasks and deadlines as well as improve communication between members and contributing parties. Research has shown that “experiences that are methodically planned and assessed, from different perspectives and with a view to continuous improvement, run more smoothly and enhance students’ final results and overall learning” [3]. Hence, project management systems play an important role in enhancing the outcome of PBL.

Currently, NTU has its own platform to facilitate learning. NTULearn is a web-based system that delivers course content, announcements, assessments for each course. NTULearn has its own use cases that are effective, such as submission of assignments, publishing of course content as well as disclosing announcements. However, it lacks PBL centric tools that would be utilized by professors and students working on the same project. For instance, each group may want to set their own deadlines, publish announcements to their own project group. Some professors may even be supervising multiple projects. Hence, a centralized system that specifically caters to the requirements of project management is paramount to the effectiveness of PBL.

1.3 Objectives

NTUCollab aims to fulfil the project management needs of FYP students. Its design and use cases have been specially engineered based on the potential challenges faced by students as well as the limitations of NTULearn. Each student will have their own schedule, represented by a calendar, with their own tasks and reports to submit. These tasks can be added by their respective project supervisor. Calendar, tasks, as well as report submissions are only accessible by the assigned student and the assigned supervisor. The calendar view shows all tasks and statuses of reports in a chronological manner.

Currently, the progress of FYPs are conducted through physical meetings, emails as well as through other forms of medium or modes of communication. Each supervisor would plan out to meet his/her assigned FYP students to meet every couple of weeks. These students would also update their professor on their progress on a weekly basis via email, for instance. As such, there is no standard procedure or method as to how each supervisor and student pair tracks or updates their progress.

1.4 Accomplishment

I have successfully developed a calendar scheduling system for students and supervisors undertaking final year projects. The calendar scheduling system allow supervisors to request for weekly reports through assigning a “task”. A deadline is declared for each task and these tasks must be fulfilled by submitting a weekly report. The calendar will be display tasks at the corresponding due dates of the task and each task will display different statuses; Submitted, Late Submission, No Submission and Not Due, based on when the report is submitted. Students can view these tasks on the calendar and submit reports. Supervisors can create and view the tasks of projects that they are supervising and the corresponding reports.

1.5 Organization of Thesis

Chapter 1: Introduction

This chapter reveals the background, motivations, objectives and accomplishments of my project

Chapter 2: Review

This chapter provides the reference materials and technical considerations that went into the project

Chapter 3: Implementation

This chapter describes how the client and server are implemented

Chapter 4: Results

This chapter illustrates the achievements of the project.

Chapter 5: Conclusion and Future Work

This chapter concludes the project and recommends possible improvements that can be implemented to further enhance the results of the project.

Chapter 2: Review

This chapter comprises of the technical references and considerations that went into developing NTUCollab.

2.1 Microsoft Outlook Calendar

Microsoft Outlook is a personal information manager developed by Microsoft, and it is part of the Microsoft Office 365 Suite [4] which every student and registered personnel of NTU has access to. Outlook is more than just an email client, it has a calendar feature as well. The Microsoft Outlook Calendar allow users to create events and supports the following functions [5]:

- Schedule an event with a specified date and/or time
- Invite attendees to scheduled event
- View events in a monthly calendar

2.2 User Interface (UI)

2.2.1 Design Language: Material Design

Design language refers to the overall style or scheme that determines the outcome of a user interface design. In other words, the design language is a template or guideline that a product or software interface follows to produce a consistent look and feel. The design language act as a guiding principle for the colour, layout design, navigation, shapes of components and icons, fonts as well as interactions or animations.

Material Design is a design language developed by Google [6]. All Google products and their interfaces follow these guidelines strictly, from their mobile applications to their web-based applications and services. It is not a surprise that a user can instantly recognize a Google application just from the user interface. Google Maps, Gmail, Google Calendar and Google Drive all have the same ‘look and feel’.

Material Design creates a consistent design across various platforms and aims to improve the user experience by making it more intuitive [7]. The outcome is a clean and simple design with ease of use. Hence, the UI of my project is largely inspired by Material Design guidelines.

2.3 Software Architecture

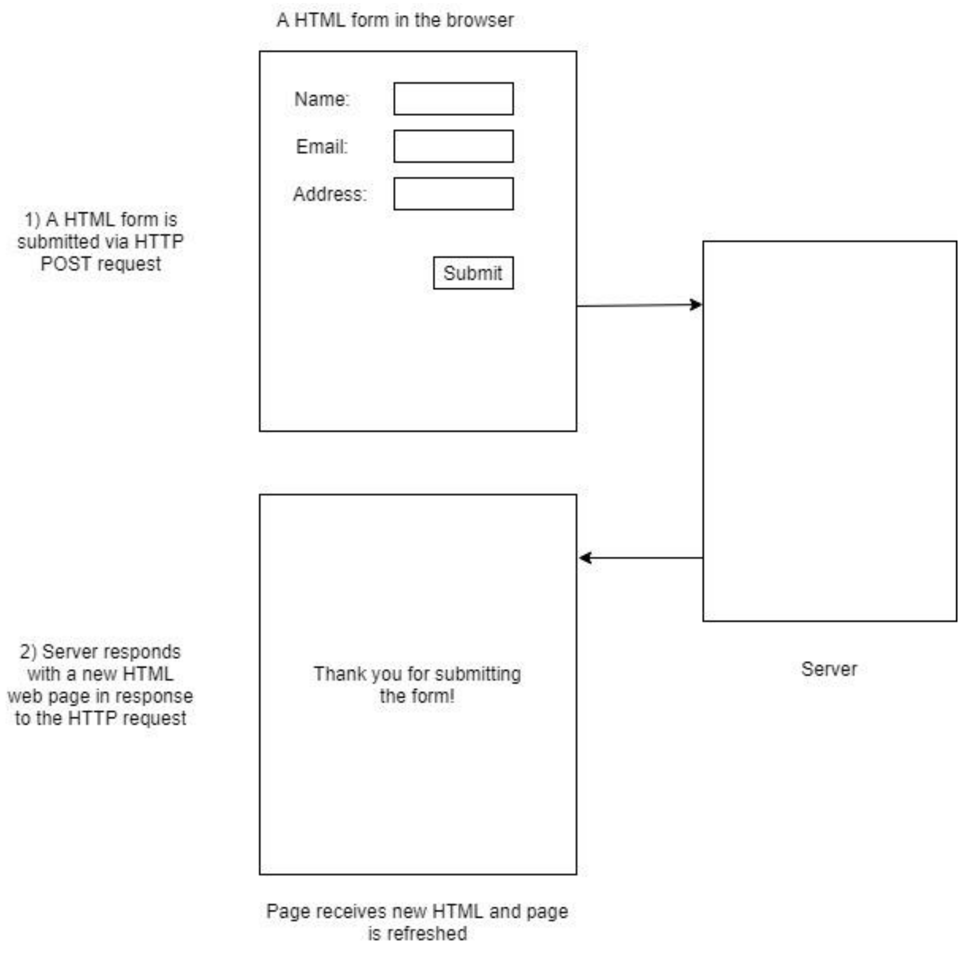
The traditional N-tier architecture of software engineering separates applications into various physical layers based on their functions [8]. These layers consist of the presentation layer that contains the views, the service/application layer that controls the business logic, processes data and services and finally, the data access tier that consist of the database.

My project deviates from this norm and explores new technological trends in web development. NTUCollab utilises Single Page Application (SPA) concepts to build the client and adopts a ‘RESTful API’ in the server side which will be further elaborated below.

2.3.1 Client: ReactJS

The Single Page Application (SPA) concept is aimed at creating web application with better performance, availability and scalability.

In a traditional web app, every time a client makes a request to the server, the server renders a new HTML page and sends it back to the client, triggering a page reload in the browser. This means that with every interaction a user makes (e.g. clicking a button, clicking a link to another page or submitting a form) it leads to a page reload and receives a new HTML web page with the updated information due to the interaction. This is illustrated by the diagram below.


A HTML form in the browser
Name:
Email:
Address:
Submit
1) A HTML form is submitted via HTTP POST request
2) Server responds with a new HTML web page in response to the HTTP request
Thank you for submitting the form!
Server
Page receives new HTML and page is refreshed

On the other hand, when a SPA loads in the browser it makes an initial request for a HTML webpage. Subsequent interactions triggers AJAX or HTTP request calls to the server and results in a response in the form of JavaScript Object Notation (JSON) objects that updates the webpage with the new data/information [9]. Only components of the webpage that received new data will be updated and does not result in an entire page reload.

React is a JavaScript library created by Facebook that allows us to build SPAs in modern web development. React's repository in GitHub has over 125,000 stars and is the 4th most starred repository [10]. Big technology names such as Facebook, Netflix, Airbnb, Twitter, Atlassian and Instagram utilise React to build their web interfaces [11]. Netflix explains their decision to adopt this library due to the start up speed, runtime performance as well as modularity in their blog [12].

One of the biggest strengths of React is that it is declarative. Being declarative means that React allow users to build web components without touching the Document Object Model (DOM) [13] directly. A comparison of a declarative and an imperative approach is illustrated with the following example.

In this example we imagine a use case where a simple button component should be rendered in red colour by default. By clicking the button, the button turns green. Clicking the button toggles the colour between red and green. The different approaches are illustrated below in pseudocode:

An imperative method:

```
if ( button.clicked() ) {

        if ( isRed() ) {

                removeRed();

                addGreen();

        }

        else {

                removeGreen();

                addRed();

        }

}
```

A declarative method:

```
if ( this.state.clicked ) {

        return <GreenButton />;

} else {

        return <RedButton />;

}
```

In the imperative approach, which JQuery adopts, the developer has to code and determine exactly what to do at every single step. A developer must code explicitly and "command" the computer on what to do. First it checks whether the button is clicked, then removes one colour, and adds another colour.

On the other hand, the declarative approach is simpler and more efficient. A developer is only required to describe and express the computational logic and the desired outcome in their code [14]. When the button is clicked, a green button will be rendered and if not, a red button will be rendered. This means that there is no interaction required to manipulate actual DOM events.

Another strength of React is that it is component based. A React application is comprised of building blocks of nested components where encapsulated component manages its own state and properties. In an example of comment box UI interface, each comment component is only needed to be built once and can be reused to display multiple comments.

Lastly, React is not a full-fledged framework but a library. Frameworks can be bloated and includes many features that may not always be needed in every application. React being a library allows developers to have more control and freedom of the frontend technological stack. Moreover, a web application should be as lightweight as possible for a faster client experience.

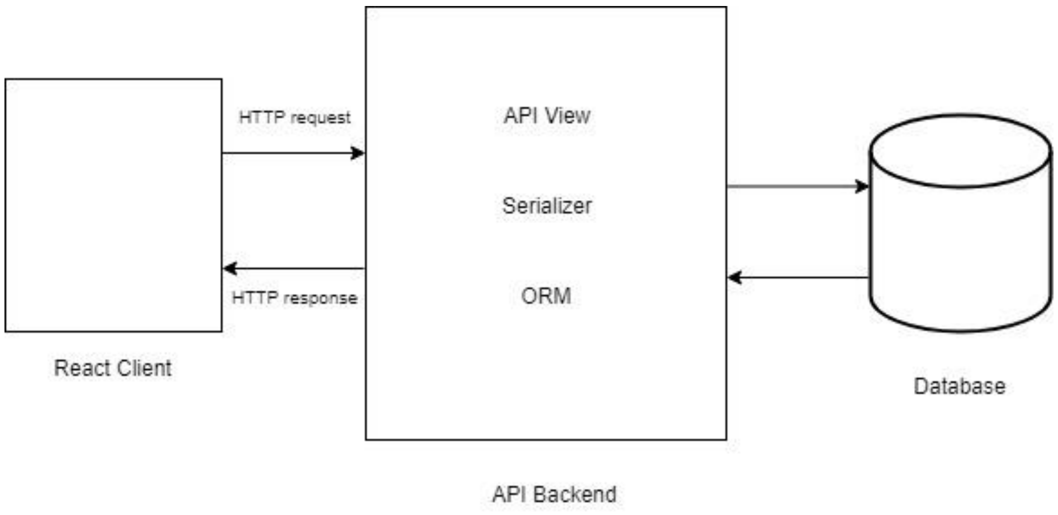
2.3.2 Server: Web Application Programming Interface (API)

An API is a documented communication protocol for interaction between various web services. It is an intermediary between software to communicate with each other [15].

Representational State Transfer (REST) is an architectural principle where the client and servers can interact without the client knowing anything about the server and the resources it hosts. Key principles of RESTful APIs involve separating the API into logical resources. These resources are manipulated using HTTP verbs such as GET, PUT, POST and DELETE and these requests interact with the resources. The list and table below display the most commonly used HTTP verbs/requests and their respective actions.

- POST – Creates a new resource
- GET – Reads a set of resources
- PUT – Updates a resource
- DELETE – Deletes a resource

HTTP Verb	URI	Description
GET	/api/users	Get a list of all users
GET	/api/users/{id}	Get the user with an id={id}
PUT	/api/users/{id}	Updates the user with id={id}
POST	/api/users	Add a new user
DELETE	/api/users/{id}	Delete a user with id={id}



2.3.3 Server: Django Rest Framework

Django Framework is a high-level python framework with a lot of built-in functionalities that a enables quick development without the need to reinvent the wheel. This framework enables developers to quickly extend and scale their projects. Django provides utilities such as user authentication, content administration, and many security features [16].

It supports the Model-View-Controller (MVC) paradigm which allows developers to keep the UI and business logic layers separated. The MVC paradigm is like the N-tier architecture, where applications are separated into various layers based on their functionalities. The Model layer determines the data logic of the applications. Database models are managed in this layer. The View layer (or more sometimes called the Template layer amongst Django users and communities) controls what is displayed in the user interface. Lastly, the Controller layer is the input mechanism of the user interface that sits between the Model and View layers.

Django Rest Framework is a flexible toolkit built on top of Django Framework to build APIs. It includes features such as serialization and authentication policies such as session and token authentication.

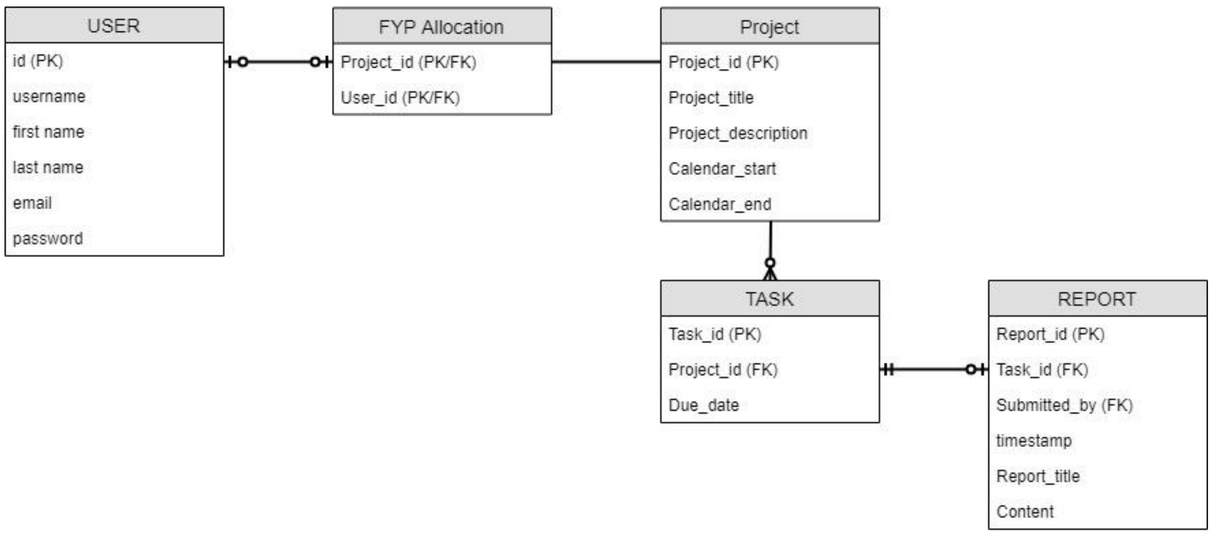
Chapter 3: Implementation

3.1 Database Design

Each student is allocated a project. Each project can only be assigned to one student. Each project can have many tasks and each task belongs to one project. For every task, there can be one or no report. Only one report can be submitted for a task.

3.1.1 Entity-Relationship Diagram

The E-R diagram is determined the scenario that is required for NTUCollab. The database models are designed, adhering to the restrictions, and follows this E-R diagram.



3.2 Server

3.2.1 Project Structure

```
- app/
   - __pycache__
   - __init__.py
   - settings.py
   - urls.py
   - wsgi.py
-pms/
   - __pycache__
   - __init__.py
   - settings.py
   - urls.py
   - wsgi.py
-README.md
-db.sqlite3
-manage.py
-requirements.txt
```

3.2.2 Authentication

The log in endpoint permission is set to “AllowAny” which means no credentials are required to make a POST request. For other resources, the endpoint permissions are only available if authenticated. When a POST request is made to the log in API, the server returns a JSON response with the token if the log in credentials are authorized. Else, an invalid response will be returned.

Pseudocode:

```
def login(request):
        if username is None or password is None:
                return "Please provide username/password", status=HTTP_400_BAD_REQUEST
        user = authenticate()
        if not user:
                return "Invalid Credentials", status=HTTP_400_NOT_FOUND
        token = token.objects.get_or_create(user=user)
        return token, status=HTTP_200_OK
```

The token is stored on the client after a successful login. Once a user is authenticated, further requests for other resources are sent with the token appended to the headers of the GET or POST requests which enables authorized access to the protected API routes.

3.2.3 Django Models

The Django model defines the essential fields and behaviours of the data to be stored in the database. Each model represents a database table. Below are snippets of code on how the database tables are represented in Django models:

```
class Project(models.Model):
        project_id = models.AutoField(primary_key=True)
        project_title = models.CharField(max_length=255)
        project_description = models.TextField(max_length=255)

        start = models.DateField(default=None, blank=True, null=True)

        end = models.DateField(default=None, blank=True, null=True)



        def __str__(self):

                return self.project_title



class ProjectAllocation(models.Model):

        project = models.ForeignKey(Project, on_delete=models.CASCADE)

        user = models.ForeignKey(User, on_delete=models.CASCADE)



        def __str__(self):

                return ‘%s %s’ % (self.project, self.user)

        class Meta:

                unique_together = ((‘project’, ‘user’))



class Task(models.Model):

        task_id = models.AutoField(primary_key=True)

        project = models.ForeignKey(Project, related_name=‘tasks’, on_delete=models.CASCADE)

        due_date = models.DateField(default=None)



        def __str__(self):
                return ‘%s’ % (self.due_date)


class Report(models.Model):
        task = models.OneToOneField(Task, related_name=‘reports’, on_delete=models.CASCADE,
primary_key=True)
        report_title = models.CharField(max_length=100)
        text = models.TextField()
        author = models.ForeignKey(User, on_delete=models.CASCADE)
        timestamp = models.DateTimeField(auto_now_add=True)


        def __str__(self):
                return ‘%s’ % (self.report_title)
        def __unicode__(self):
                return ‘%s’ % (self.report_title, self.text, self.timestamp)
```

3.2.4 Django Serializers

Django Serializers converts model instances into native Python datatypes which are then rendered into JSON format. The serializers give control over the output of the JSON response.

```
class UserSerializer(serializers.ModelSerializer):
        class Meta:
                model = User
                fields= ‘__all__’
```

```
class ReportSerializer(serializers.ModelSerializer):
    author = serializers.StringRelatedField(default=serializers.CurrentUserDefault())
    class Meta:
        model = Report
        fields = '__all__'


class TaskSerializer(serializers.ModelSerializer):
    reports = ReportSerializer()
    class Meta:
        depth = 1
        model = Task
        fields = '__all__'


class TaskOnlySerializer(serializers.ModelSerializer):
    class Meta:
        model = Task
        fields = '__all__'


class ProjectSerializer(serializers.ModelSerializer):
    #tasks = serializers.StringRelatedField(many=True)
    tasks = TaskSerializer(many=True)
    class Meta:

        model = Project

        fields = ( 'project_id', 'tasks', 'project_title', 'project_description', 'start', 'end')



class ProjectAllocationSerializer(serializers.ModelSerializer):

    class Meta:

        model = ProjectAllocation

        fields = '__all__'



class UserProjectSerializer(serializers.ModelSerializer):

    project = ProjectSerializer()

    user = UserSerializer()



    class Meta:

        depth = 0

        model = ProjectAllocation

        fields = ('project', 'user')
```

3.2.5 Django Views

```
class UserProject(generics.ListAPIView):

    serializer_class = UserProjectSerializer

    queryset = ProjectAllocation.objects.all()
    def get_queryset(self):

        user = self.request.user

        if user.is_superuser:

            return ProjectAllocation.objects.all()

        else:

            return ProjectAllocation.objects.filter(user=user)



class AddNewReport(generics.ListCreateAPIView):

    permission_classes = (AllowAny,)

    queryset = Report.objects.all()

    serializer_class = ReportSerializer



    def perform_create(self, serializer):

        req = serializer.context['request']

        serializer.save(author=req.user)



class TaskListCreate(generics.ListCreateAPIView):

    queryset = Task.objects.all()

    serializer_class = TaskSerializer



class NewTask(generics.CreateAPIView):

    permission_classes = (AllowAny,)
    queryset = Task.objects.all()

    serializer_class = TaskOnlySerializer
```

Django views accept two attributes which are the “serializer class” and the “queryset”. The queryset must be set to determine the objects returned in this view. The serializer class is responsible for validating and serializing/deserializing the output. In the code snippet above, calling the UserProject view will query and return all ProjectAllocation data.

3.2.6 Django URLs

Django’s built in URL dispatcher maps the URL path expressions to the Python Views. The snippet below illustrates the respective views being mapped to the corresponding URLs.

```
urlpatterns = [

        path(‘userproject/’, views.UserProject.as_view() ),

        path(‘addnewreport/’, views.AddNewReport.as_view() ),

        path(‘admincalendar/’, views.TaskListCreate.as_view() ),

        path(‘addtask/’, views.NewTask.as_view() ),

]
```

3.2.7 Database Migration

NTUCollab uses SQLite, the default and embedded database engine for Django. SQLite is a compact library and doesn’t not require a separate server process.

Django’s built-in migrations are responsible for making changes to the database schema. Once the models are developed in Django, migration commands are given to propagate the changes in the models to the database.

A typical workflow for migrating changes in the models to the database is as follows:

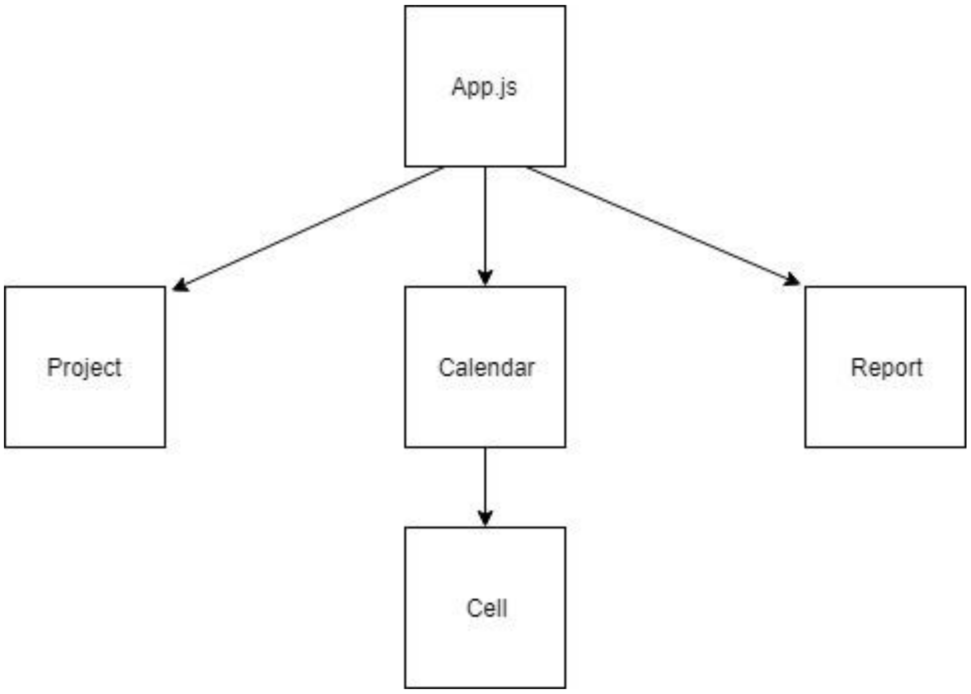
```
$ python manage.py makemigrations

$ python manage.py migrate
```

- makemigrations is responsible in creating new migrations based on the changes in the models
- migrate is responsible for applying these migrations to the database

3.3 Client

The component tree diagram below illustrates the component hierarchy in the NTUCollab web application.



When the app is loaded, there are three routes for the three main components of the app; a project component to view the assigned projects, a report component to view available reports, and a calendar component which consists of nested cell components for each date in the calendar.

3.3.1 Project Structure

- dist/
  - index.html
  - main.js
- src/
  - components/
    - supervisorview/
      - SupervisorCalendar.js
      - SupervisorCell.js
      - SupervisorProjectView.js
    - Calendar.js
    - Cell.js
    - Loader.js
    - LoadingCircle.js
    - LoginForm.js
    - Navbar.js
    - Projects.js
    - ReportContainer.js
    - ReportForm.js
    - ReportModel.js
    - listItems.js
  - styles/
    - styles.css
  - App.js
  - index.html
  - index.js

3.3.2 Authentication

Use cases:

a) Student log in
    1. Student enters credentials into form on log in page
    2. Student reaches home page if authenticated
    3. Student logs out
    4. Student is directed to the log in page

b) Supervisor log in
    1. Supervisor enters credentials into form on log in page
    2. Supervisor reaches home page if authenticated
    3. Supervisor logs out
    4. Supervisor is directed to the log in page

c) Invalid log in
    1. Invalid credentials entered at log in page
    2. User remains at log in page

Pseudocode:

```
if token exists {
        user can access protected routes;
} else {
        redirect to log in page
}
```

3.3.3 Project View

The Project component is created to display information on the assigned FYP. The component retrieves the data from the server and receives the data as shown below:

```
[
        {
                “id”: 1,
                “project”: {
                        “project_id”: 3,
                        “project_title”: “Web-based project management system”,
                        “project_description”: “some description”,
                        “start”: “2018-08-13”,
                        “end”: “2019-11-16”
                }
        }
]
```

The component loops through the array of objects (in this case there is only one object because each student is only assigned one FYP) and displays the project information such as the project title and project description for the user to view.

3.3.4 Calendar View

The Calendar view is a little more complex. The calendar is generated based on the start and end date of the project. Each date is represented by a cell and these cells are styled in rows of 7 cells for every day of the week.

In each cell, the component checks if there are any tasks due on that date. If there is, there will be an indicator to display the status of the task and the status will depend on when the tasks are due and when the report is submitted.

The four statuses are as follows:

- Pending Report – No report submitted, task if not due yet
- No Submission – No report submitted, task is due
- Late Submission – Report is submitted late
- On Time – Report is submitted on time

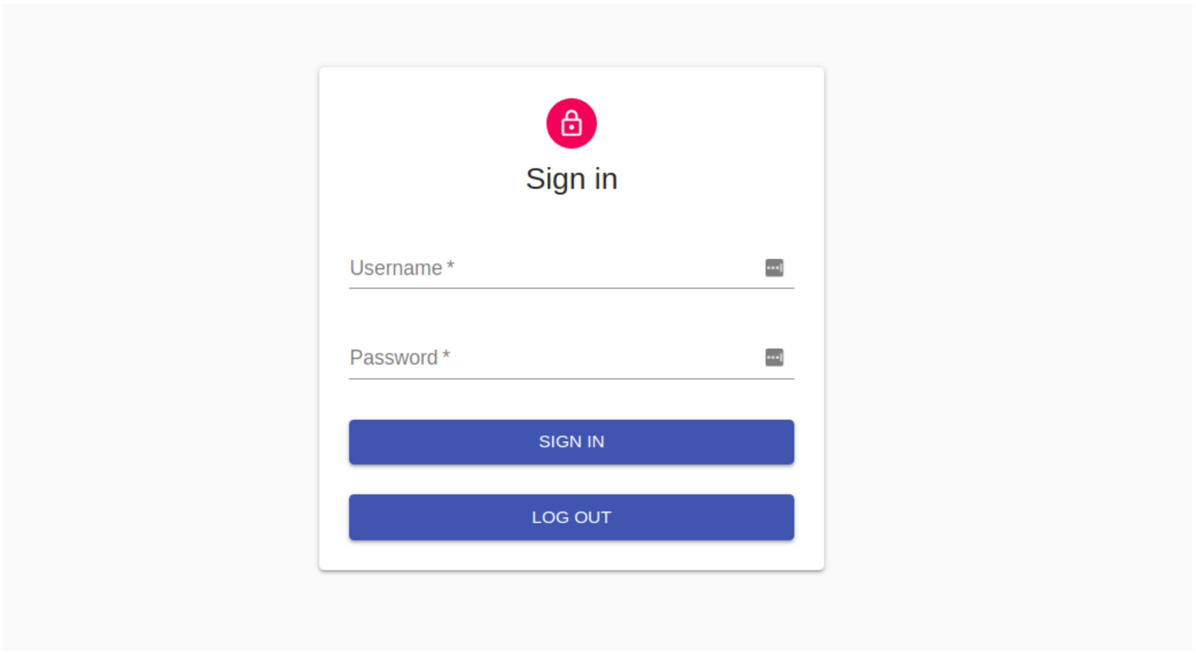
When a status is clicked it will display a pop up. For statuses such as “Pending Report”, and “No Submission” where no report has been submitted, a pop-up form will appear to submit a report. For “Late Submission” and “On Time” where a report has been submitted, a pop-up will display the submitted report.

3.3.5 Report View

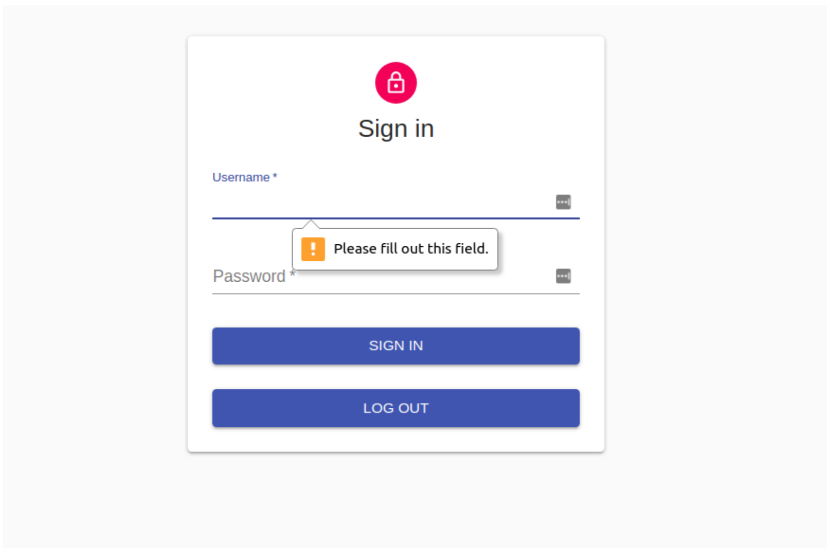
The report view is like the project view. The report component receives an array of report objects. The component loops through the array and displays the reports and its attributes.

Chapter 4: Results and Discussion

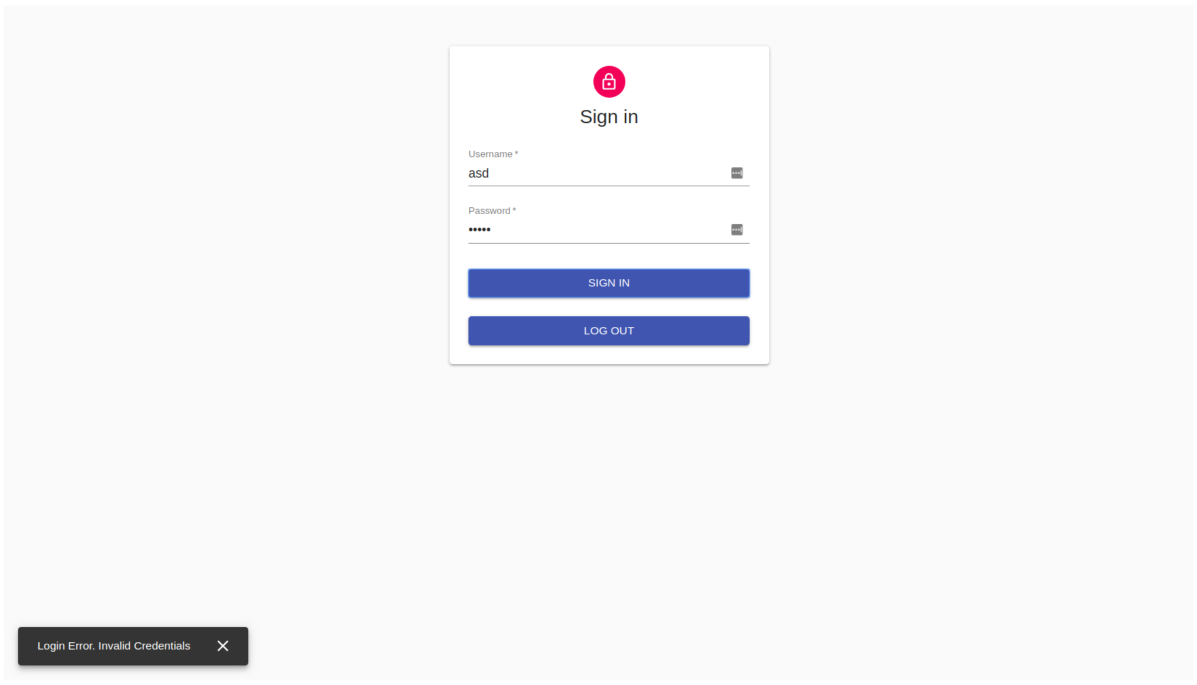
4.1 Authentication



The log in view is as shown above when an unauthorized user tries to access the URL of the web application. The other routes are disabled for any unauthorized user. When an unauthorized user attempts to access other routes, they will be redirected to this log in page.



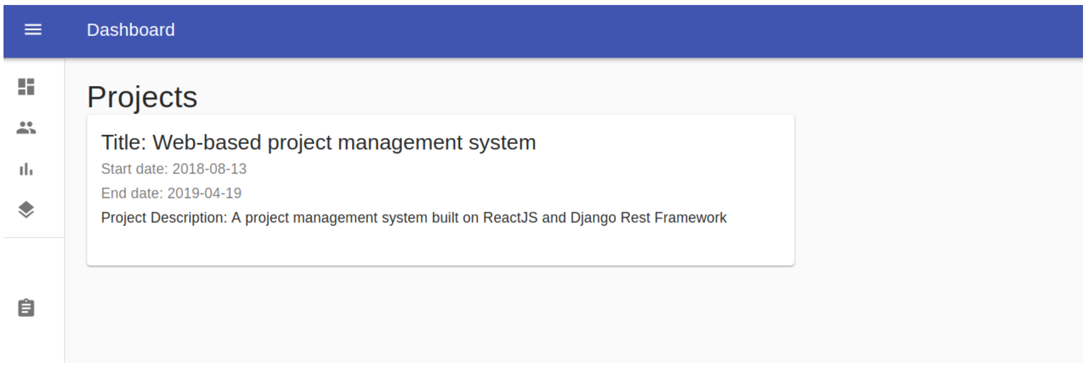
When a required field is not entered, an error message is relayed. A user is required to input an email and password. Either fields can not be left empty, else an error as shown above is displayed.



Login Error. Invalid Credentials

A user is denied access when the log in form is submitted with invalid credentials. The error message will appear as a “snackbar” at the bottom of the page. Any log in error message should not indicate if the wrong credential is in the username or password for security reasons.

4.2 Project View



Projects

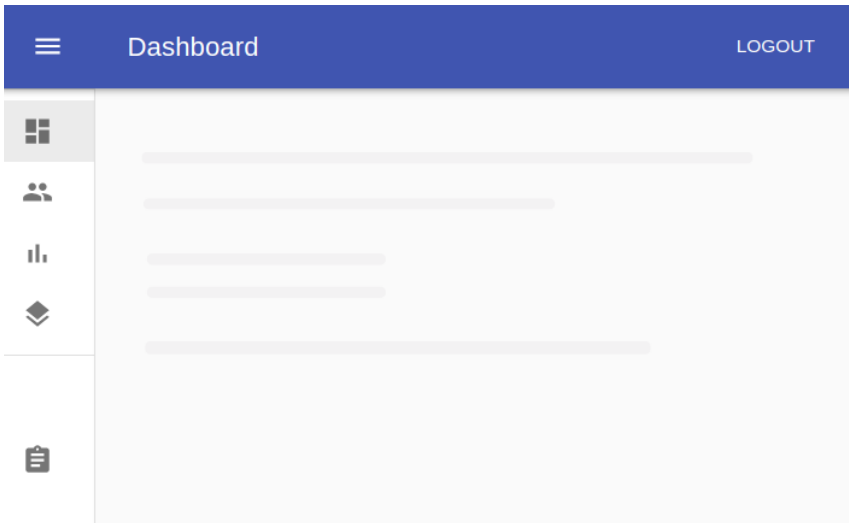
Title: Web-based project management system

Start date: 2018-08-13

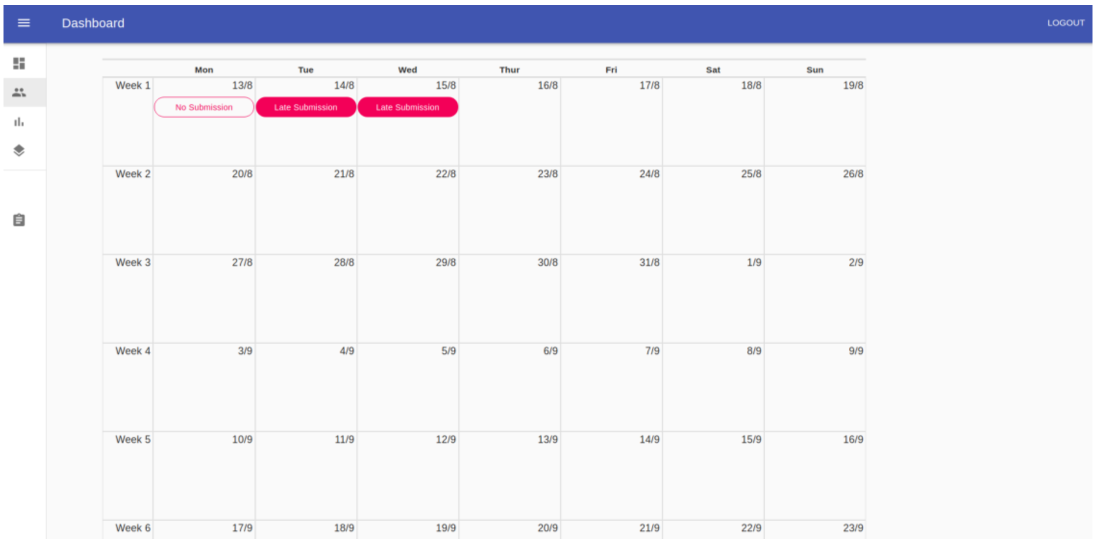
End date: 2019-04-19

Project Description: A project management system built on ReactJS and Django Rest Framework

This is the project view, where a user can view the assigned FYP and the project details. I have implemented a loader component that will be displayed before a response from the server is received and mounted onto the component and populated with data. In a non-development environment, the server response will not be as quick. The loading animation is to improve user experience, giving users an idea that a component on the page is loading and will be receiving data.

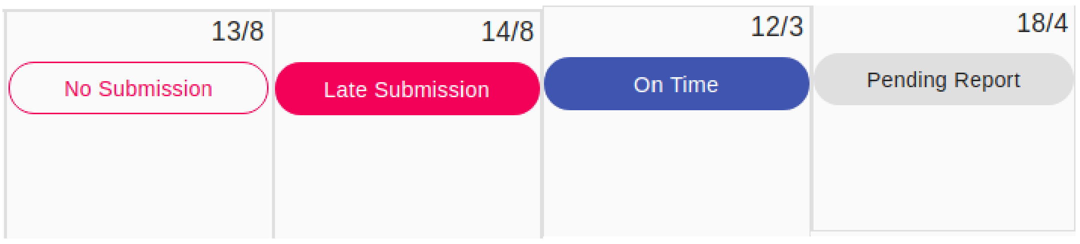


4.3 Calendar View

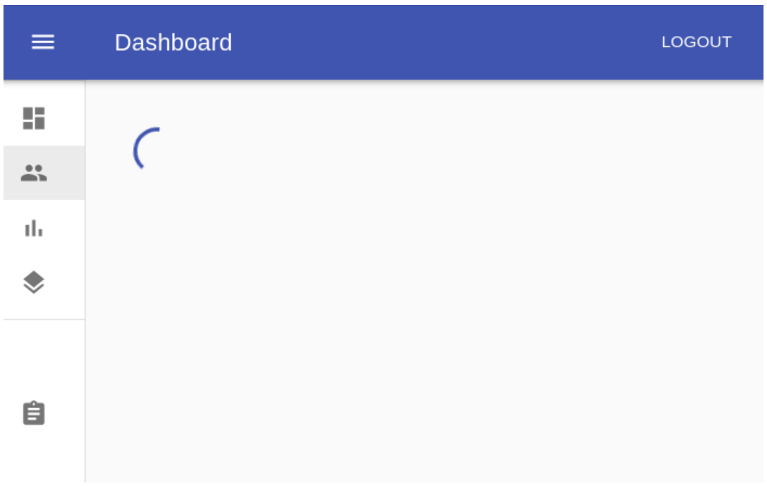


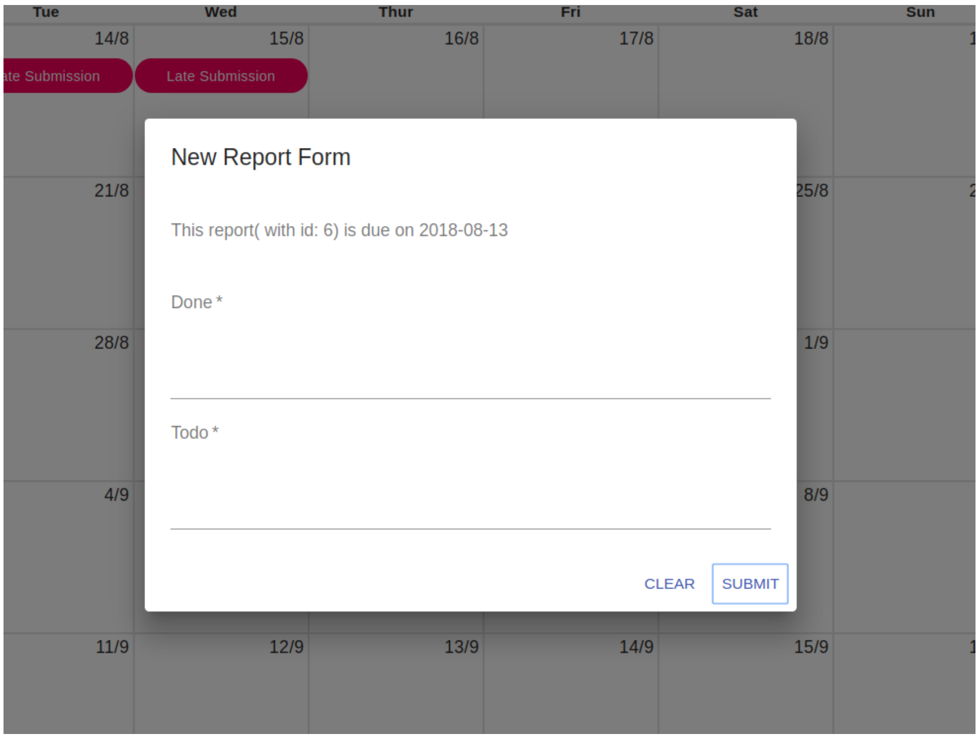
	Mon	Tue	Wed	Thur	Fri	Sat	Sun
Week 1	13/8 No Submission	14/8 Late Submission	15/8 Late Submission	16/8	17/8	18/8	19/8
Week 2	20/8	21/8	22/8	23/8	24/8	25/8	26/8
Week 3	27/8	28/8	29/8	30/8	31/8	1/9	2/9
Week 4	3/9	4/9	5/9	6/9	7/9	8/9	9/9
Week 5	10/9	11/9	12/9	13/9	14/9	15/9	16/9
Week 6	17/9	18/9	19/9	20/9	21/9	22/9	23/9

The calendar view, which is the main component, is a vertical scrolling calendar. The months are not separated for a more sequential and unobstructed view. In this view we can see that there are four tasks that appeared on their respective dates. The four available statuses for the tasks are shown in the next image.



When the calendar component has yet to receive data, a loading component is displayed just like the Project view. A loading indicator is useful to the user can distinguish if there is no data to be displayed or that there is incoming data to be loaded and displayed.





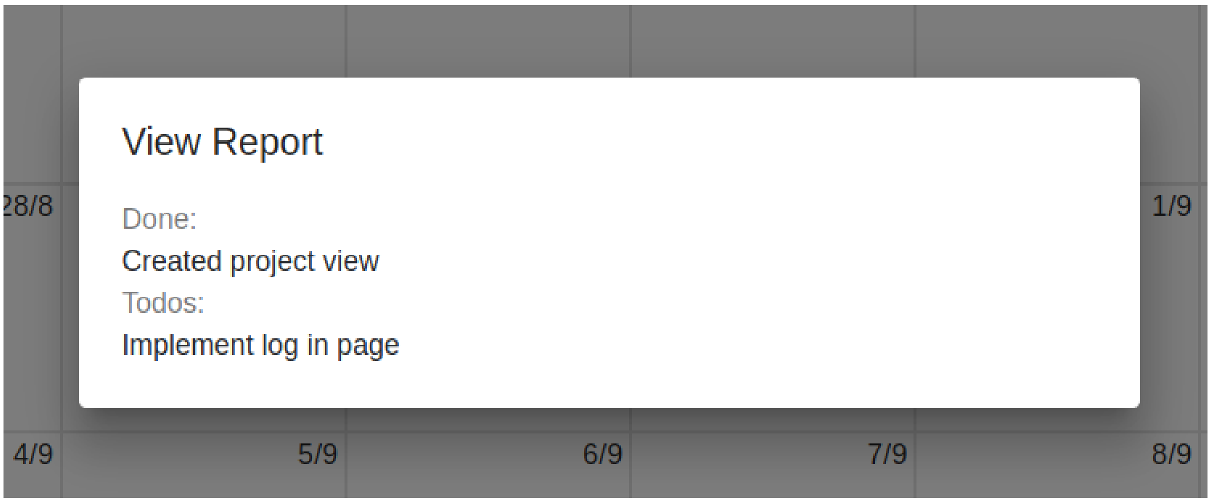
New Report Form

This report(with id: 6) is due on 2018-08-13

Done *

Todo *

For tasks with no reports submitted (i.e. No Submission, Pending Report), clicking on the status will bring up a form to submit a form. This is implemented to facilitate the submission of incomplete tasks. The student can simply click on the date of a task that is due and submit the corresponding report for the task.


View Report

Done:

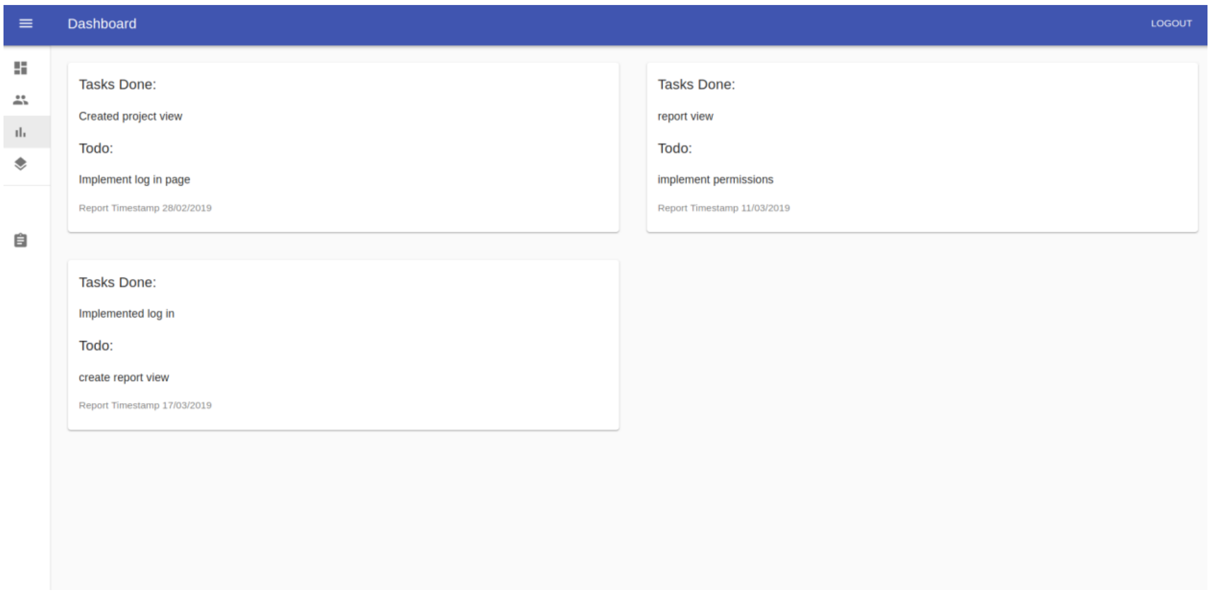
Created project view

Todos:

Implement log in page

For tasks fulfilled with a submission (i.e. Late Submission, On Time), clicking on the status will bring up the corresponding report for that task/date.

4.4 Report View



Tasks Done:

Created project view

Todo:

Implement log in page

Report Timestamp 28/02/2019

Tasks Done:

report view

Todo:

implement permissions

Report Timestamp 11/03/2019

Tasks Done:

Implemented log in

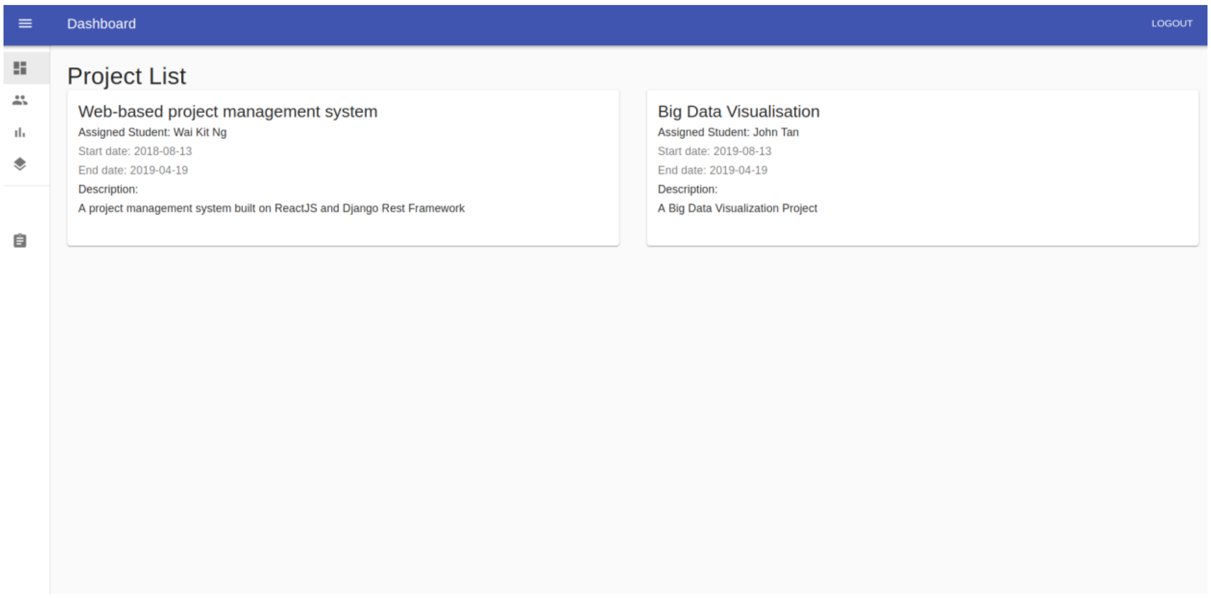
Todo:

create report view

Report Timestamp 17/03/2019

From this view, the student/supervisor can view all previously submitted reports in one page. An overview of past reports that were submitted can be viewed here. Students and supervisors alike can track and have a good idea of the progress over the past few months.

4.5 Supervisor Views

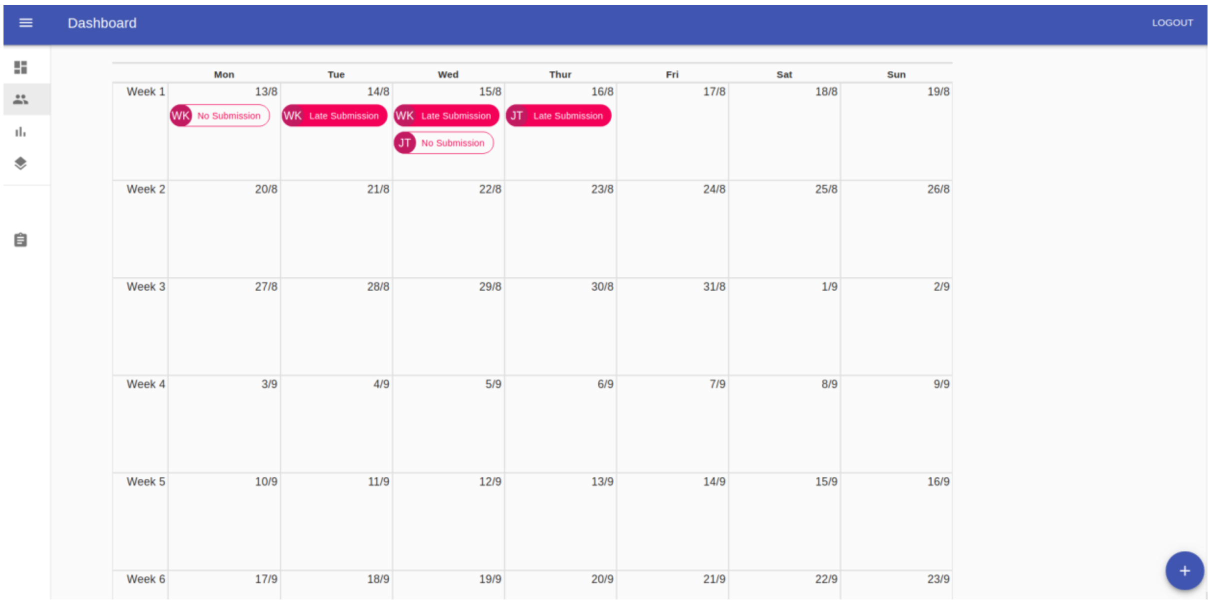


Project List

Web-based project management system
Assigned Student: Wai Kit Ng
Start date: 2018-08-13
End date: 2019-04-19
Description:
A project management system built on ReactJS and Django Rest Framework

Big Data Visualisation
Assigned Student: John Tan
Start date: 2019-08-13
End date: 2019-04-19
Description:
A Big Data Visualization Project

The supervisor has access to view more than one project because each supervisor may supervise more than one student for their FYP.



	Mon	Tue	Wed	Thur	Fri	Sat	Sun
Week 1	13/8 WK No Submission	14/8 WK Late Submission	15/8 WK Late Submission JT No Submission	16/8 JT Late Submission	17/8	18/8	19/8
Week 2	20/8	21/8	22/8	23/8	24/8	25/8	26/8
Week 3	27/8	28/8	29/8	30/8	31/8	1/9	2/9
Week 4	3/9	4/9	5/9	6/9	7/9	8/9	9/9
Week 5	10/9	11/9	12/9	13/9	14/9	15/9	16/9
Week 6	17/9	18/9	19/9	20/9	21/9	22/9	23/9

The supervisor's calendar is populated with data from multiple projects. In the screenshot above, this supervisor's calendar can view the statuses of tasks belonging to two students with the initials “WK” and “JT”. This is to help the supervisor differentiate the tasks and the status of each student.

Chapter 5: Conclusion and Future Work

5.1 Comments Module

Apart from viewing tasks and reports on NTUCollab, a possible module that can be added to improve the project management system is through a comments module. For every report, students and supervisors alike can post threaded comments to allow discussion of the report. The discussion can be in the form of a clarification or an exchange of new ideas.

5.2 Notifications Module

Another possible improvement would be through the implementation of notifications. With efficiency in mind, any form of update such as a new task or a new report being submitted should create a form of notification to inform parties involved of the new information.

For instance, once a new report is submitted by a student, a notification would inform the respective supervisor and there will not be delays or misinformation because they are ignorant of any new changes. It will be unwise if students need to send an email to the professor informing them of their submitted report and requesting for review or feedback.

5.3 Announcements Module

An announcement module can be implemented for further improvement of NTUCollab. Apart from tasks and reports, announcements can be helpful in project management. An announcement module would be a method to relay important information that can be referenced in the future. For example, a supervisor can make an announcement and give instructions regarding upcoming submissions such as the final report submission. After the announcement has been made, the student can refer to it in the future.

5.4 Mobile Application Development

Today, more people are accessing the internet on the go on their mobile phones. Mobile applications replaced many web services that we use today. For example, people can access their email using mobile applications such as Gmail and Outlook. Developing a mobile application for NTUCollab will improve its usability.

One of the reasons why the server is developed as a RESTful API is to be able to cater to mobile application development in the future. Mobile applications built using React Native, a framework to build native mobile apps using JavaScript and React, can piggyback on the same API backend. Furthermore, the syntax of React Native is similar to React and hence it is a possible consideration for future development.

5.5 Conclusion

The main solution of NTUCollab is to keep students and professors up to date with the progress of a final year project. Having a more organized and structured platform for communication between the students and their supervisors will go a long way. In one look, the student and supervisor can have a view of the past reports and have a good idea on the progress as well as the status of the project.

NTUCollab was developed with the intention of aiding future students with their FYP. From personal experience, I realised that apart from a student's abilities, effective project management and fruitful planning can influence greatly on the outcome of the project. A consistent and reliable system of managing tasks improves the productivity any project.

I have learnt a lot in this full-stack software development project. I was able to experience the process of planning, prototyping as well and implementing them into the development process.

References

[1] E. d. S. Zancul, “Project-based learning approach: improvements of an undergraduate course in new product development,” 2017.

[2] M. S. Y. C. M. L. E. &. L. A. Carpenter, 2016. [Online]. Available: https://www.asee.org/public/conferences/64/papers/14876/view. [Accessed 24 September 2018].

[3] A. D. Lantada, 17 May 2015. [Online]. Available: https://www.researchgate.net/publication/261180509_Towards_Successful_Project-Based_Teaching-Learning_Experiences_in_Engineering_Education. [Accessed 20 September 2018].

[4] Microsoft. [Online]. Available: https://products.office.com/en-us/compare-all-microsoft-office-products?&activetab=tab%3aprimaryr2.

[5] Microsoft. [Online]. Available: https://support.office.com/en-us/article/introduction-to-the-outlook-calendar-d94c5203-77c7-48ec-90a5-2e2bc10bd6f8.

[6] Google LLC, [Online]. Available: https://material.io/design/material-studies/#. [Accessed 2019].

[7] G. LLC. [Online]. Available: https://developers.googleblog.com/2014/06/this-is-material-design.html.

[8] Microsoft, [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier.

[9] Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/magazine/dn463786.aspx.

[10] Facebook, “Github,” [Online]. Available: (https://github.com/facebook/react. [Accessed 2019].

[11] “Stackshare,” [Online]. Available: https://stackshare.io/react. [Accessed 2019].

[12] “Medium,” [Online]. Available: https://medium.com/netflix-techblog/netflix-likes-react-509675426db. [Accessed 2019].

[13] “Mozilla,” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. [Accessed 2019].

[14] O. Torgersson, “A Note on Declarative Programming Paradigms”.

[15] “mulesoft,” [Online]. Available: https://www.mulesoft.com/resources/api/what-is-an-api. [Accessed 2019].

[16] “djangoproject,” [Online]. Available: https://docs.djangoproject.com/en/2.1/topics/security/. [Accessed 2019].

Python Environment

```
Django==2.1.5
django-cors-headers==2.4.0
django-extensions==2.1.4
djangorestframework==3.9.0
pkg-resources==0.0.0
pytz==2018.7
six==1.12.0
```