Software Design Project

FYP Chooser

CS4125 SYSTEMS ANALYSIS AND DESIGN

Authors

17244412 – Paul Kinsella
16165365 – Ibrahim Alaydi
16110366 – Norbert Cholewka
17241782 – Bartlomiej Mlynarkiewicz

Contents

Project Description

Introduction

FYP Chooser came about due to issues with students selecting FYP topics and supervisors. Due to Covid19, faculty members inboxes are overloaded with messages for students requesting supervision; therefore, we propose a prototype application to tease out the feasibility of such an application.

Business Description

The main goal of the FYP Chooser system is to have a single location where all FYP related actions and information held rather than faculty members and students having to dig through Sulis, Moodle, and their Outlook inbox. Everything will be contained at one location for easy searching. The application hopes to solve communication issues with all parties involved. Detailed requirements will be explained below in the requirements section. An issue that has been highlighted this semester is that students and faculty did not seem to know about student limits per Supervisor. This system will also address that.

Students and faculty will sign up to the site, once signed up they will be giving a specific role by the administrator. Faculty will have extra menus, and students will have basic menus such as Profile, Search, and request an FYP supervision. Faculty (aka) Supervisors can add a list of topics in which a student can search for and send a request. The Student can add a note about why she/he would like this topic. The Supervisor can see requests and a student's QCA.

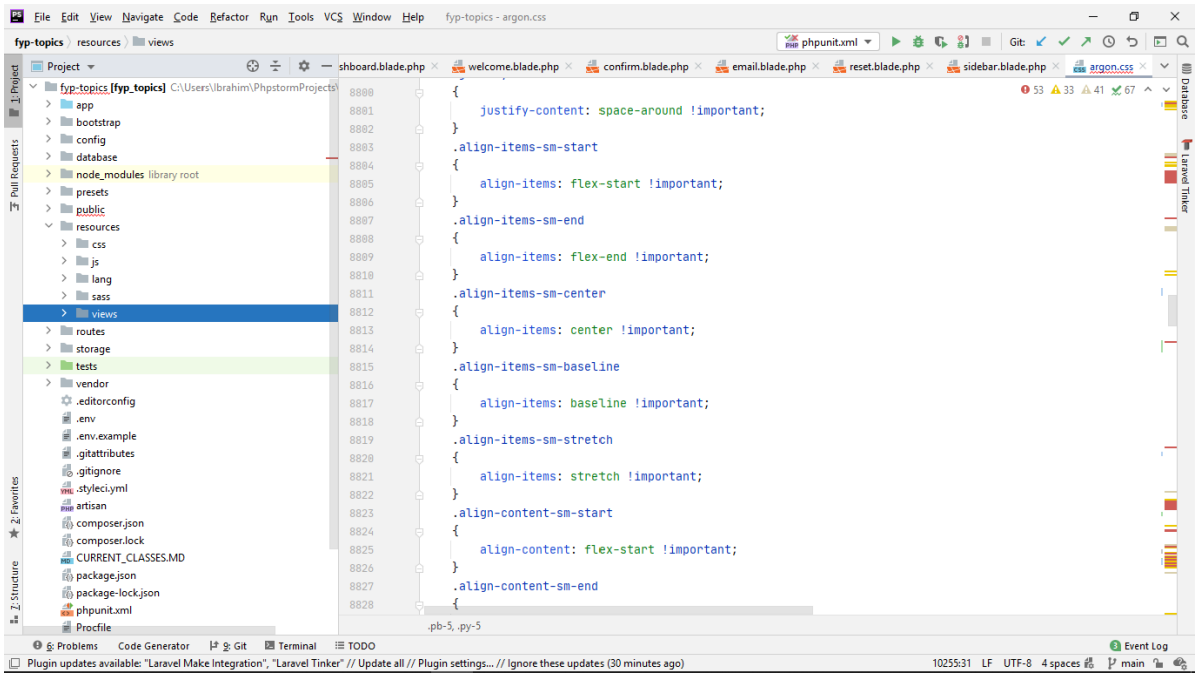
Software Tools

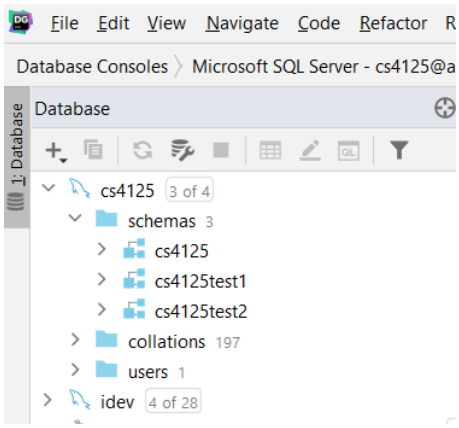
PHP-Storm

PhpStorm is a commercial, cross-platform IDE which is developed by JetBrains. PhpStorm offers a PHP, HTML and JavaScript editor with on-the-fly code analysis, error prevention and automatic PHP and JavaScript code refactoring.

We decided to go for this as we have experience in development with PHP. We also have experience in CSS and HTML from previous course modules. Furthermore, some of us know JavaScript as we learned it while on work experience.

N.B These images were added later in the project to give more context on the software setup. We are aware this information is not required in the early stages.





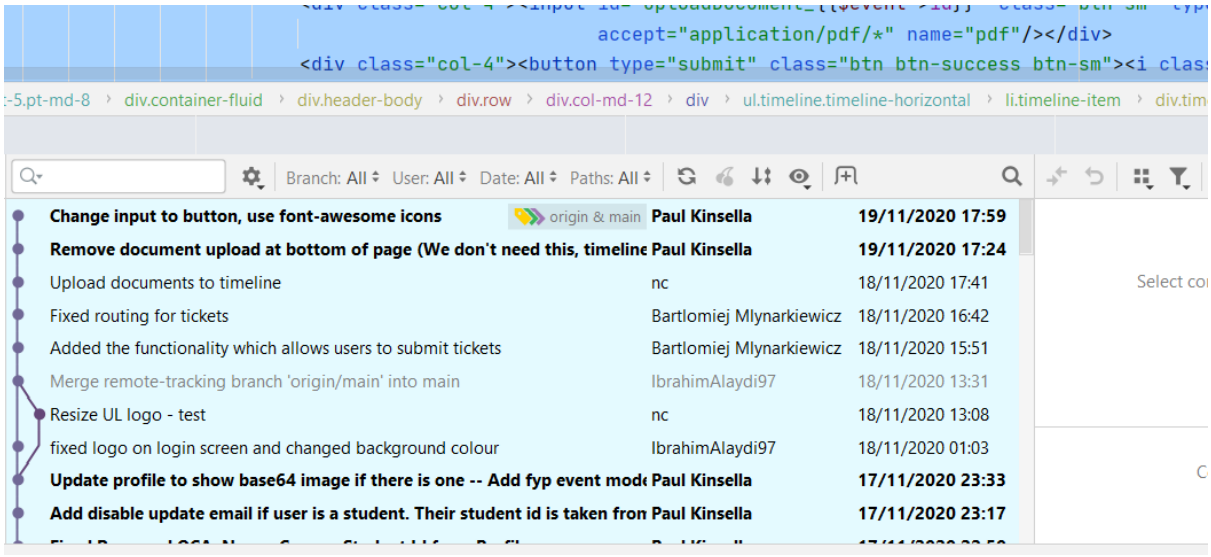
DataGrip IDE

DataGrip is a multi-purpose database management system that has enhanced functionality compared to specific database management software.

We created 2 test databases to avoid issues with migrations and logic in the solution. Once the code was working 100% on the test, we would migrate to the main Database

GitHub Plugin for PhpStorm

GitHub Plugin for Phpstorm was used to do the pushing and pulling to/from the remote repository. This plugin allowed developers to keep track of changes.



Version Control (VCS)

Github.com used to host the project remotely. The group implemented strict rules with regards to pulling and pushing commits. In a real-world scenario, there would be more branches than just main but because the group worked with a similar setup before we are confident that there shall be no issues.

- Any change to a class in the Models folder requires
    - PHP artisan make: migration SomeNameGoesHere
    - PHP artisan migrate

    These commands created updated the remote Database with the new schema based on the model classes

- Always do a pull before a commit
- Ask another to pull the new commit to compile & run.
- Any issues with a pull request or merging notify the group
- Do not push any configuration files/folders such as the idea that are unique to that user.

Draw.io

Draw.io is an excellent SaaS service that will be used to create all the diagram for the project. The software has diagrams for any requirements used in this project.

Heroku

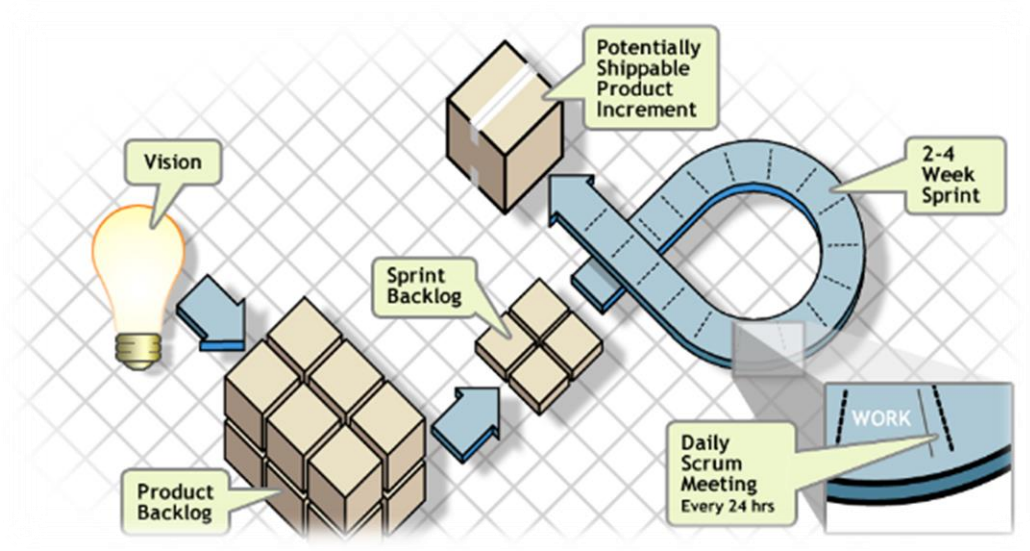
Heroku is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud. We use Heroku to host our application, our application in Heroku connected to our GitHub repository for Continuous Deployment, Continuous deployment is a strategy for software releases wherein any code commit that passes the automated testing phase is automatically released

into the production environment, making changes that are visible to the software's users (Continuous Delivery using Spinnaker on Amazon EKS | Amazon Web Services, 2020).

Software Life Cycle (SLC)

There were 3 SDLC's approached to choose from for this project, Agile Scrum, WaterFall, and the V Shape Model. End with went up going with an **agile-scrum** type approach primarily due to it being used in most Irish software companies. Every member of the team used agile to some extent in their work experience. Therefore, it was an obvious choice. The daily scrum meetings are helpful to highlight any possible delays in the project timeline.

The group decided to settle on 2-week sprints. The backlog contained all the crucial features to implemented for the project/system. Please see the Youtrack section below which is a scrum type web application that helped with the project and work



Water Fall & V Shape Cons

Due to the time restrictions, we will not be able to approach this project with a waterfall. Because this time restriction the waterfall does not allow for a working software until the end of development. Waterfall also has high risks and uncertainties as you are left wondering if what you are designing is working or not. Waterfall does not allow for changing requirements.

Another downside of designing with waterfalls is that it does not allow for much reflection or revision. Once an application is in the testing process, it is not straightforward to go back and modify something that in the design phase was not well known or thought about. Similarly, with the v-model, it is not effective if the requirements of the project are continually changing, and it does not allow for changes in the design when a fault discovered in the testing stage. Acceptance criteria could change as well as we get feedback based on what is acceptable to fulfil the requirement of this project.

Project Planning

Role Assignment

Each member of the group was assigned a specific role or roles.

ROLE ASSIGNMENT			
Role		Description	Assigned To
1	Project Manager	• Monitoring progress • Analysing and managing project plan & risk • Schedule Group Meetings	Ibrahim
2	Documentation Manager	Responsible for managing all tasks related to documentation on the project	Ibrahim
3	Requirements Engineer	Responsible for defining, documenting, and maintaining the requirements.	Paul Kinsella
4	Architect	Defines the System Architecture	Paul Kinsella
5	Systems Analysis	Creates a Conceptual Class Model	Bart
6	Designer	Develop a plan for the project based on the specifications	Norbert
7	Technical Lead	• Delegating work and tasks to members of the group • Working closely with the team to define and address technical issues	Bart
8	Software Developers	• Create software packages, components and modules related to the project • Fix bugs found in testing • Create database schema	All
9	Tester	• UAT • Unit Tests	Norbert

Project Plan Workflow

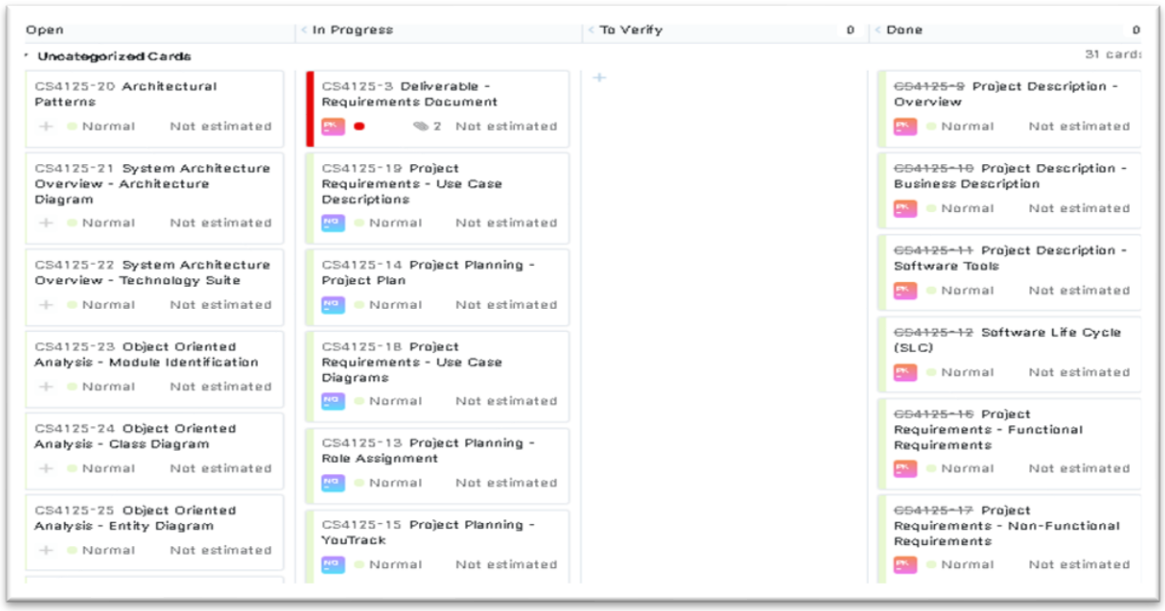
Due to the way, the course laid out, and the design covered in week ten, some of the weeks could be behind or ahead of schedule. Because of our software backgrounds, the coding phase will be a lot quicker for us compared to other groups, and this will speed up the coding iteration timeline.

Workflow

Project Plan Workflow				
Task		**Description**	**Week**	**Output**
1	Team setup	Set up a group	2-3	Team structure management
2	First group planning	Review existing projects, learn how to use GitHub, agree on a scenario	3	Planned whether the application/product is going to be. Discussion about project milestones
3	Capture and document Requirements	Agree on basic specifications/requirements for FYP booking system which determines the overall project design	4	Initial talks about defining requirements for FYP and general outline of the final product
4	Analysis	Re-analyse the proposed system based on the prerequisites	5	Get started with the use case and sequence diagrams. Models and communication as well.
5	High-Level Architecture	System design and architecture based on FYP intentions. Confirmation that the project requirements will fulfil all essential system criteria	6	Architectural Patterns Architecture Diagram Design Patterns
6	Coding Iteration 1	Essential Infrastructure and two critical use cases	7	User authorisation
7	Coding Iteration 2	Additional Use cases	8	User Management
8	Coding Iteration 3	Additional Use cases & design patterns	9	Communication/Search/FYP topics/Requests
9	Coding Iteration 4	Tidy up remaining bugs	10	Fixing bugs that overlooked in the encoding process
10	Testing	Do a full test of the system, black-box and white-box testing	11	Complete testing of the system
11	Final works	Get everything ready for hand-over to the client	12	Complete product for production

YouTrack

Youtrack used to monitor the progress of the project. The initial sprints focused on the requirements and system layout, and later sprints not seen below were used for the development phase. The scrum board consisted of a **backlog** for all mandatory requirements for this project, **in progress** which displays cards currently being worked on, **verify** which shows items to be signed off by a peer review process and lastly, **Done** - which shows completed tasks. The task we also assigned to a group member using this great system.



Stakeholders

Stakeholder Name	Organisation	Role
JJ Collins	CSIS/UL	Administrator
Anila	LERO / UL	Supervisor
Bartek	UL	Developer
Ibrahim	UL	Developer
Norbert	UL	Developer
Paul	UL	Developer
U.L. Students	U.L.	Students
U.L. Faculty	CSIS/UL/LERO	Faculty/Supervisor

Project Requirements

Functional Requirements

Requirement Id	Section	Statement	Must / Good to have
FR001	All Users	Users must log in with email + password	Must
FR002	All Users	Users must register	Must
FR003	Admin	Add User roles such as Supervisor, Student	Must
FR004	Admin	Assign Roles to User	Must
FR005	Admin	Ban, Create, Delete, Update, View Users	Must
FR006	Admin	Reset Password for User	Must
FR007	Faculty	Add, Update, Delete, View an FYP Topic	Must
FR008	Faculty	Update Profile	Must
FR009	Faculty	Upload CSV of topics	Good to have
FR010	Faculty	Add QCA range on a topic (Only students with 3+ QCA can apply)	Must
FR011	Faculty	View a student Profile	Must
FR012	Faculty	View Student Topic Requests	Must
FR013	Faculty	Decline a student Request for an FYP Topic (Can leave a note)	Must
FR014	Faculty	Accept a student Request for an FYP Topic (sends message also)	Must
FR015	Faculty	Send a message to a Student	Must
FR016	Faculty	Search all FYP topics	Must
FR017	Student	Search all FYP topics	
FR018	Student	View FYP Topics (even if QCA is not high enough)	Must
FR019	Student	Request supervision for selected FYP topic	Must
FR020	Student	Cancel a Request (Leave a message for Supervisor)	Must
FR021	Student	Update Profile (QCA is mandatory) Cannot search unless they set QCA	Must
FR022	Student	Send a message to the Supervisor	Good to have

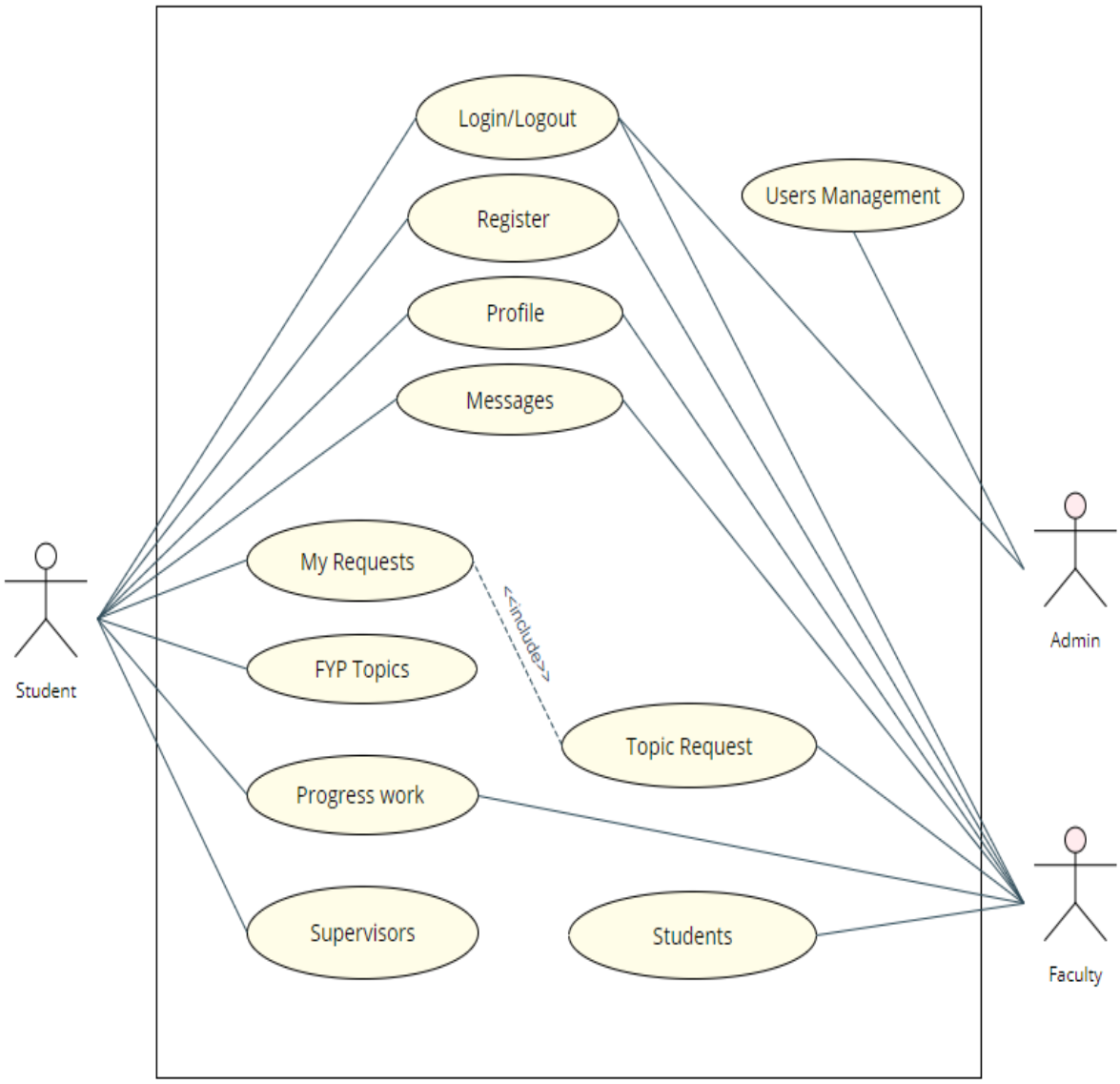
Non-Functional Requirements

Requirement Id	Section	Statement	Must / Good to have
NFR001	Security	**The website shall be encrypted with an HTTPS cert**. Having a site use HTTP is not a good idea, a malicious user on the network can sniff/view the clear text traffic send through HTTP with the help of a network analyser tool such as Wireshark. They could view passwords and usernames that were used to sign in to the website	Good to have (If budget allows)
NFR002	Security	Passwords for all users shall be encrypted in the Database	Good to have
NFR003	Security	The Database shall only be accessed by the localhost on the server when the system is in production. This will stop hackers trying to brute for the password. We have looked at the access logs, and there appear to be lots of hacking attempts coming from Chinese and Russian IP addresses	Must
NFR004	Security	No other user data shall be visible to any other user unless required	Must
NFR005	Security	The web application will implement Cross-Site Scripting mitigation. Cross-Site Scripting attacks allow attackers to inject web pages accessed by other users with client-side scripts. Attackers may use a cross-site scripting weakness to bypass access restrictions, such as the same-origin policy	Must
NFR006	Security	DDoS Protection. The cloud service we went with (Heroku) has DDoS protection as part of the package. Therefore, any type of denial-of-service attacks will be mitigated	Must
NFR007	Maintainability	The Database must be accessible remotely and securely.	Must
NFR008	Maintainability	The Database must be backed up daily, so if the system were to go down no data is lost	Good to have

NFR009	Maintainability	The web application must have CI/CD for easy update and deployment	Must
NFR010	Maintainability	The web application must log errors to help identify issues with the system	Must
NFR011	Capacity/Performance	The system should have enough CPU processing to handle a high volume of requests	Must
NFR012	Capacity/Performance	The system shall have a large amount of RAM to run the web application	Must
NFR013	Capacity/Performance	The Database should have enough space to a large amount of data	Must
NFR014	Capacity/Performance	The website should have a response time of 300-600ms for light pages and 2000-4000ms for heavier pages	Must
NFR015	Extensibility	The web application should work on multiple platforms.	Must
NFR016	Extensibility	The web application must work with a load balancing setup	Must
NFR017	Extensibility	The web application should be easy to update in case a vulnerability discovered in the framework used	Must
NFR018	Extensibility	The web application should support multiple database types such as Mysql, Postgres	Must
NFR019	Extensibility	The web application should be easy to enhance without breaking existing code	Must

Category Risk	Description
Schedule	Estimating project time can be poorly timed. Delivery of the code implementation may not be delivered on time-insufficient identification of requirements and the time needed to develop them. Unexpected changes to the design.
Budget	Incorrect budget estimate in early planning discussions. Budget overruns can be very easy due to its low-cost threshold. We are increasing the scope of the project.
Technical	They are continually changing requirements. The implementation of the product can be problematic-difficult integration of modules for design.
Operative	Failure to overcome the tasks. Failure to resolve the responsibilities. Inadequate resources. No communication in the team.
Programmatic	Change in market demand. They are changing the strategy and priority of the product stakeholders.

Use Case Diagrams



1. Availability Check Use Case Scenario

Use Case	**Description**
Login	When a User visits the FYP platform, he asked to provide a username and password which has been registered.
Register	If the FYP platform used for the first time, the user is invited to create an account.
Logout	Allows the user to end the login session.
Users Management	The manual account administration The User account sometimes is added manually. Set up a user role for the account. Ban, Create, Delete, Update, View
Profile	Update User profile.
Students	To get more information about a particular person, the Supervisor can view through students.
Supervisors	To get more information about a particular supervisor, Student can view through supervisors.
Topic Request	The Supervisor has access to view FYP requests by students.
Messages	Users can contact by sending a message.
FYP Topics	Users have the option to search for FYP.
My Requests	Students can see his/her requests about FYP topic's supervision.
Progress Work	Student can upload and delete progress work documents.

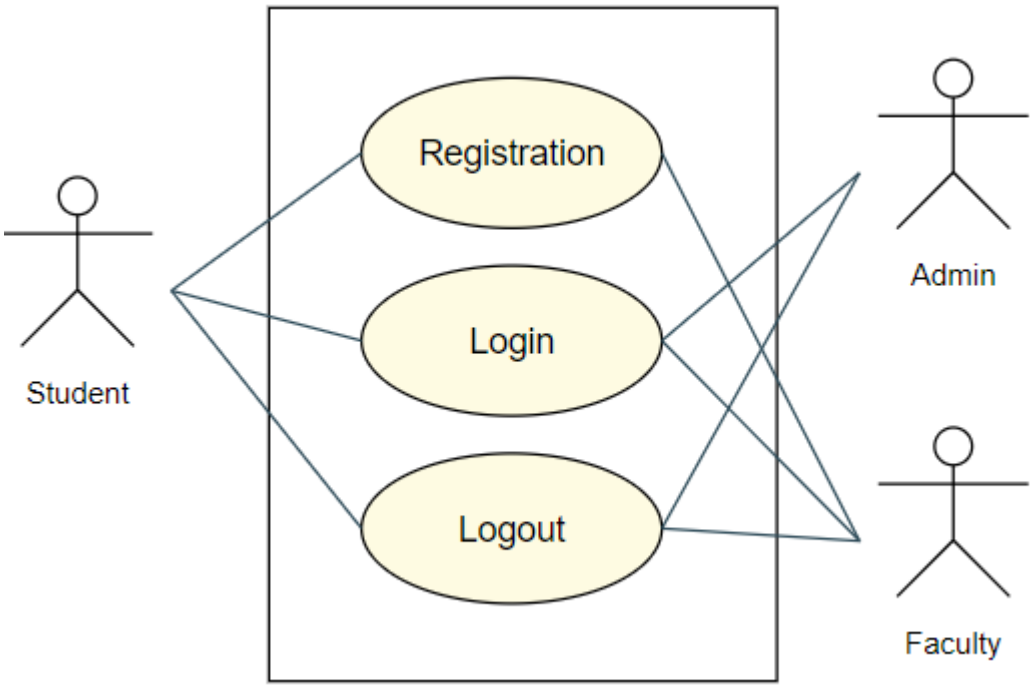
1.2. Two key Use Case Description

Use Case 1	**My Requests**	
Goal in context	The Student has proposed supervision and should expect the faculty to view it positively.	
Preconditions	The Student has sufficient QCA and project knowledge.	
Success: Postcondition	The Student has sent his FYP proposal within the specified deadline. The proposal was accepted.	
Failed: Postcondition	Within the specified deadline, the Student did not present his FYP proposal. The proposal was rejected by the faculty.	
Actors	Student/Supervisor	
Trigger	Booking FYP request is received.	
Description	Step	Action
	1	Students search for available projects based on their QCA.
	2	The faculty returns a list including descriptions of available projects.
	3	Students choose his/her proposal.

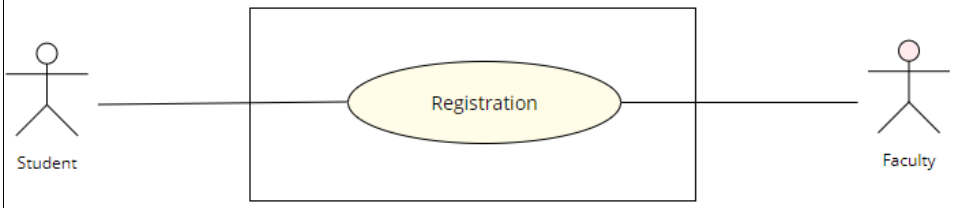
	4	Students send request FYP proposal
	5	The Supervisor received FYP proposal and accepted it.
	6	The Student received approval for the project
Extensions		
	4a	Project rejection by the faculty. The Student is searching for available projects again.

Use Case 2	**Topic Request**	
Goal in context	Faculty reserves specified FYP upon receiving of confirmation by the Supervisor.	
Preconditions	The application received within the timeframe specified.	
Success: Postcondition	The faculty Supervisor an FYP proposal. The project has been assigned, and the Student has been sent the needed acceptance.	
Failed: Postcondition	The deadline did not receive the FYP proposal. The FYP proposal was rejected.	
Actors	Student/Supervisor	
Trigger	Message from students.	
Description	Step	Action
	1	The faculty informed of the request for the FYP proposals from students.
	2	The faculty makes booking of the FYP.
	3	The faculty notifies students that their proposal has been approved.
Extensions		
	2a	The deadline provided has been passed. The Student cancelled his submission.

2. User Authentication System

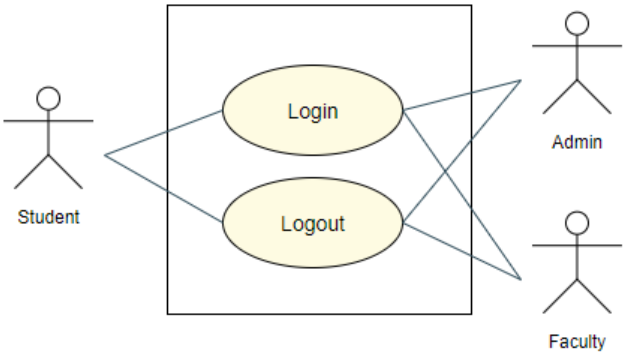


2.1. User Registration

Action Name	User Registration
Actors	Student / Faculty
Use Case Diagram	
Trigger	The user wants to register into the system.
Brief Description	The user registration process
Initial Step-By-Step Description	1. User will provide email and password. 2. Click "Registration". 3. The system will check if the email already added in the system in User D.B. 4. If the email verification successfully, the user registration is completed.
Task Dependencies	Student / Faculty Login.
Goal	To register into the system.

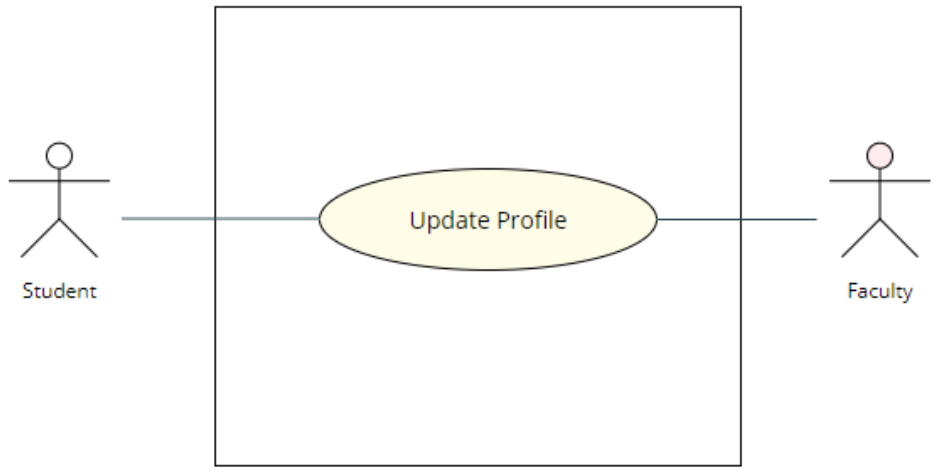
Post Conditions	The user will be registered.
Proposed Module	User Authentication System
Student/Faculty Interface	

2.2. User Login / Logout

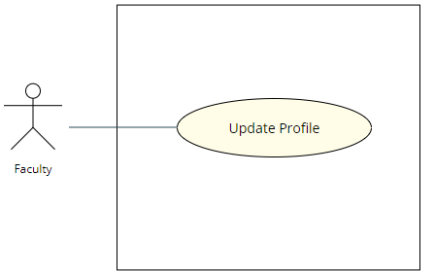
Action Name	User Login / Logout
Actors	Student / Faculty/Admin
Use Case Diagram	
Trigger	T user wants to login into or logout from the system.
Brief Description	The process of the user login/logout
Initial Step-By-Step Description	1. User will provide email and password. 2. Click "Login". 3. The system will check if the email is existing in User D.B. 4. If the email and password are correct, the system will provide to log in. 5. If the user logged the can click "Logout". 6. The user successfully logged out.
Task Dependencies	Student / Faculty / Admin Login.
Goal	The goal is to log in the user into or logout from the system.
Post Conditions	The user will be logged in or out.
Proposed Module	User Authentication System
Student/Faculty/Admin Interface	

3. User Management

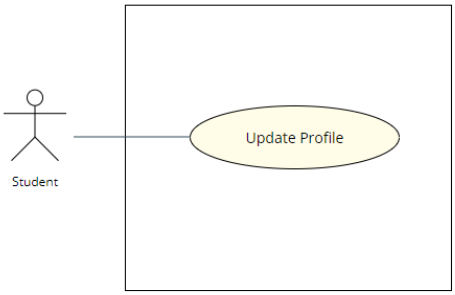
3.1. User Profile Management



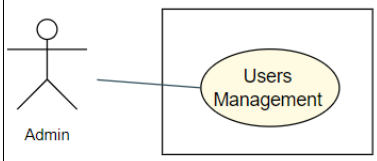
3.1.1 Update User Profile

Action Name	Update user profile
Actors	Faculty
Use Case Diagram	
Trigger	The user wants to update the profile.
Brief Description	The user profile update process
Initial Step-By-Step Description	1. Click "Update Profile". 2. The user has text boxes to choose with associated data profile. 3. Click "Save" 4. The system will update data in User D.B.
Task Dependencies	Faculty Login.
Goal	To update the user profile.
Post Conditions	The user data profile will be updated.
Proposed Module	User Management
Faculty Interface	

3.1.2 Update User Profile

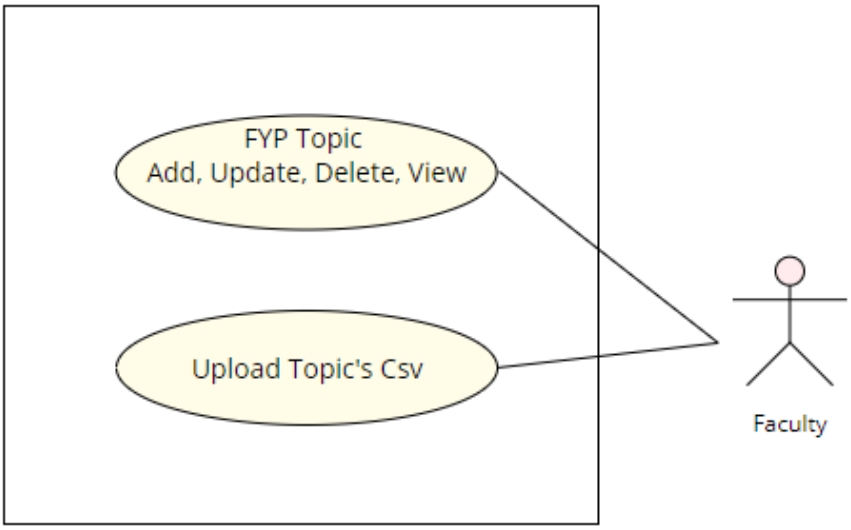
Action Name	Update user profile
Actors	Student
Use Case Diagram	
Trigger	The user wants to update the profile.
Brief Description	The user profile update process
Initial Step-By-Step Description	1. Click "Update Profile". 2. The user has text boxes to choose with associated data profile. 3. Additional QCA text box is required to fill out. 4. Click "Save" 5. The system will update data in User D.B.
Task Dependencies	Student Login.
Goal	To update the user profile.
Post Conditions	The user data profile will be updated.
Proposed Module	User Management
Student Interface	

3.2. Admin User Management

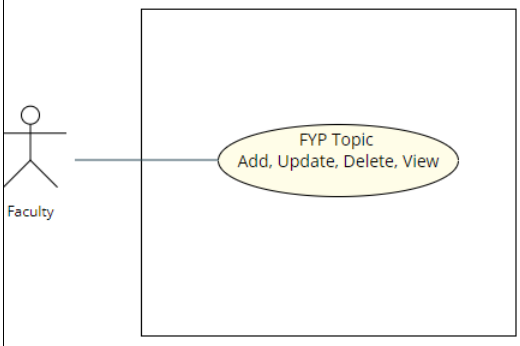
Action Name	Admin User Management
Actors	Admin
Use Case Diagram	
Trigger	The Admin wants to update, block, delete, add the user.
Brief Description	The Admin User Management process
Initial Step-By-Step Description	1. Click the associated button. 2. Admin follows the guidelines for updating data 3. The system will update data in User D.B.
Task Dependencies	Admin Login.
Goal	To update User D.B.
Post Conditions	The user data will be updated.
Proposed Module	User Management
Student Interface	

4. FYP Topic Management

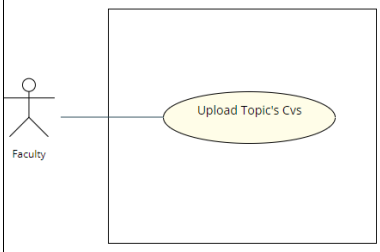
4.1. Faculty of FYP Management



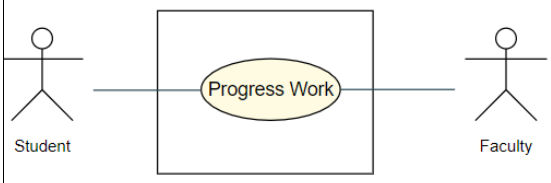
4.1.1. Faculty of FYP Management

Action Name	Faculty FYP Management
Actors	Faculty
Use Case Diagram	
Trigger	When the Supervisor wants to add, update, delete or view topic.
Brief Description	The FYP topic management process
Initial Step-By-Step Description	1. The user has buttons options to add, update, delete or view topic. 2. Click the associated button. 3. The system will update associated action in D.B.
Task Dependencies	Faculty Login.
Goal	To add, update, View or delete FYP topic.
Post Conditions	The FYP topic will be added, deleted, viewed, or updated.
Proposed Module	FYP Topic Management
Faculty Interface	

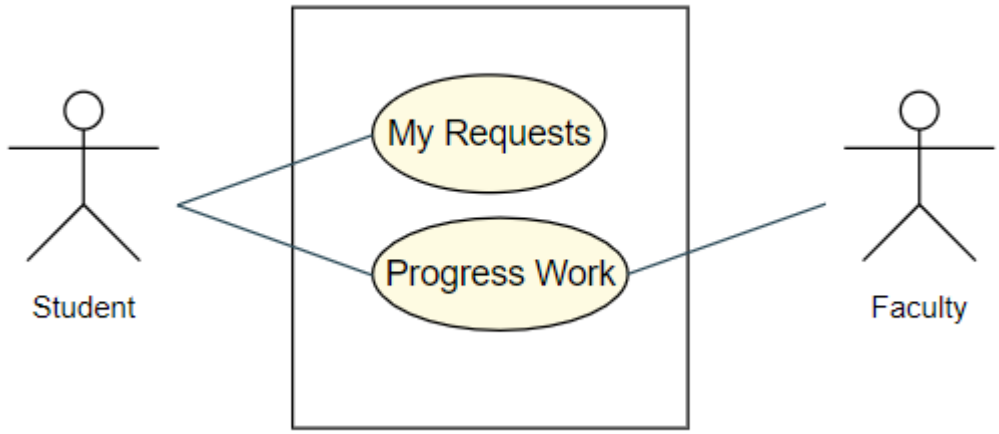
4.1.2. Upload FYP CVS file

Action Name	Upload FYP CVS file
Actors	Faculty
Use Case Diagram	
Trigger	The Supervisor wants to upload the FYP topics file.
Brief Description	The FYP topic management process
Initial Step-By-Step Description	1. The user has buttons options to upload tile. 2. Click "Upload". 3. The system will update added file in D.B.
Task Dependencies	Faculty Login.
Goal	To upload the FYP topic file.
Post Conditions	The FYP topic will be uploaded.
Proposed Module	FYP Topic Management
Faculty Interface	

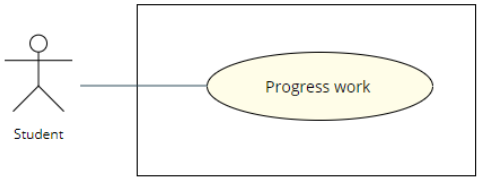
4.1.3. Student's work progress

Action Name	Student's work progress
Actors	Faculty
Use Case Diagram	
Trigger	The Supervisor wants to check the Student's progress of FYP work.
Brief Description	The FYP topic management process
Initial Step-By-Step Description	1. The Supervisor has access to view all documents uploaded by Student. 2. Click the associated button. 3. The system will allow the Supervisor to download the document.
Task Dependencies	Faculty Login.
Goal	To download the Student's FYP work process document.
Post Conditions	The FYP document will be downloaded.
Proposed Module	FYP Topic Management
Student Interface	

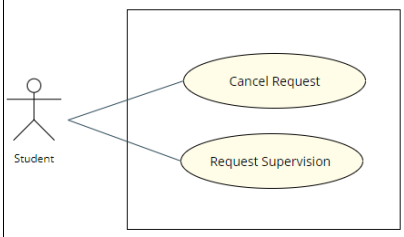
4.2. Student FYP Management



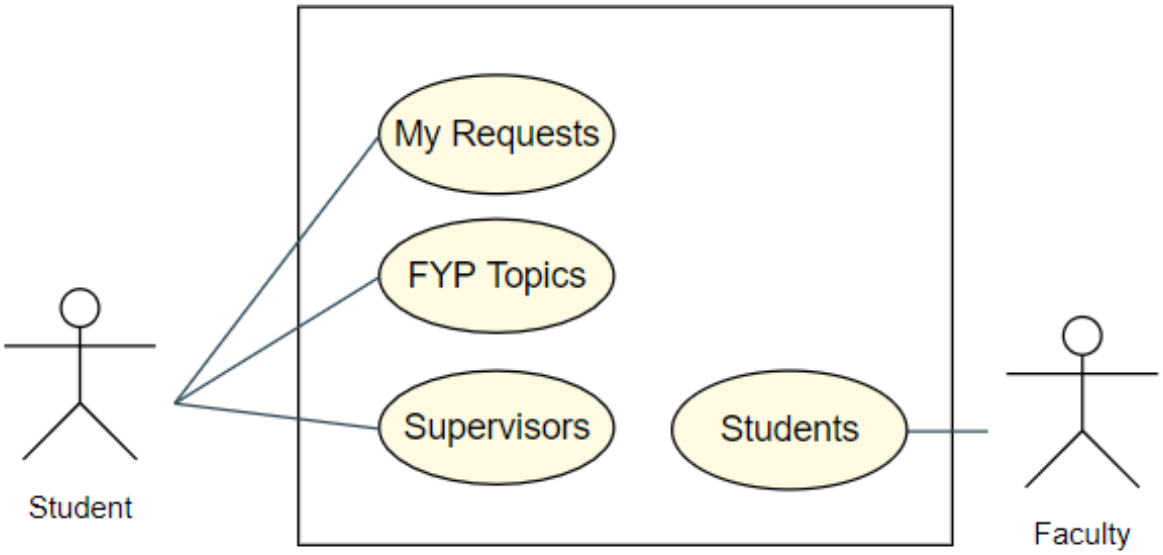
4.2.1. Progress Work

Action Name	Progress Work
Actors	Student
Use Case Diagram	
Trigger	The Student wants to update the progress of FYP work.
Brief Description	The FYP topic management process
Initial Step-By-Step Description	4. The user has buttons options to upload or delete the topic. 5. Click the associated button. 6. The system will update associated action in D.B.
Task Dependencies	Student Login. Acceptance of the Superior.
Goal	To upload or delete FYP work process doc.
Post Conditions	The FYP doc will be updated in D.B.
Proposed Module	FYP Topic Management
Student Interface	

4.2.2. Request or Cancel FYP Supervision

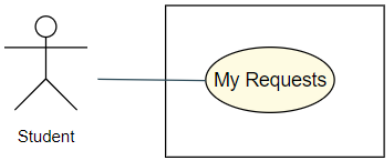
Action Name	Request or Cancel FYP Supervision
Actors	Student
Use Case Diagram	
Trigger	The Student wants to request or cancel supervision.
Brief Description	The FYP topic management process
Initial Step-By-Step Description	1. The user has buttons options to "Request" or "Cancel" supervision. 2. Click the associated button. 3. The user has an optional reason text box to fill out. 4. The system will update associated action in D.B.
Task Dependencies	Student Login.
Goal	To upload or delete FYP work process doc.
Post Conditions	The FYP doc will be updated in D.B.
Proposed Module	FYP Topic Management
Student Interface	

4.3 Search management

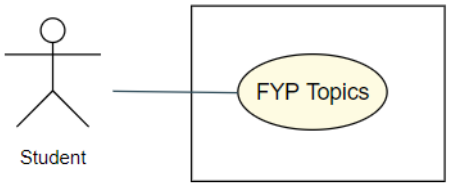


4.3.1. Search options for a Student

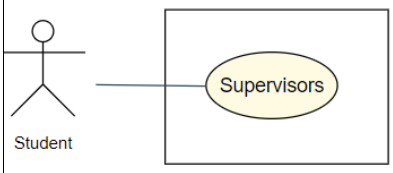
4.3.1.1 My Requests

Action Name	My Requests
Actors	Student
Use Case Diagram	
Trigger	The Student wants to search for requests about supervision.
Brief Description	The Search management process
Initial Step-By-Step Description	1. The user must type in search box 2. Click the associated button. 3. The system will search typed in the text in D.B.
Task Dependencies	Student Login.
Goal	To search requests about supervision.
Post Conditions	The Student will get founded request or nothing if D.B. will not find anything.
Proposed Module	Search management
Student Interface	

4.3.1.2 FYP Topics

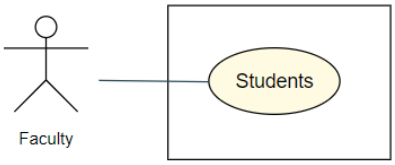
Action Name	FYP Topics
Actors	Student
Use Case Diagram	
Trigger	The Student wants to search for FYP topics.
Brief Description	The Search management process
Initial Step-By-Step Description	1. The user must type in search box 2. Click the associated button. 3. The system will search typed in the text in D.B. 4. 4. Student can see FYP topic description
Task Dependencies	Student Login.
Goal	To search available FYP topics.
Post Conditions	The Student will get founded FYP topic or nothing if D.B. will not find anything.
Proposed Module	Search management
Student Interface	

4.3.1.3. Supervisors search

Action Name	Supervisors search
Actors	Student
Use Case Diagram	
Trigger	The Student wants to search for available supervisors.
Brief Description	The Search management process
Initial Step-By-Step Description	1. The user must type in search box 2. Click the associated button. 3. The system will search typed in the text in D.B.
Task Dependencies	Student Login.
Goal	To search available Supervisors names.
Post Conditions	The Student will get founded Supervisor name or nothing if D.B. will not find anything.
Proposed Module	Search management
Student Interface	

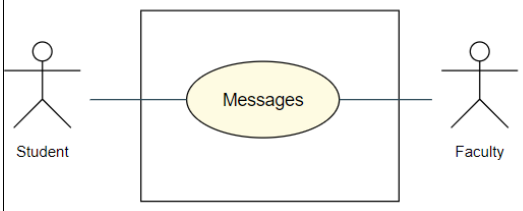
4.3.2. Search options for a Supervisor

4.3.2.1. Students search

Action Name	Student search
Actors	Supervisor
Use Case Diagram	
Trigger	The Supervisor wants to search for Student.
Brief Description	The Search management process
Initial Step-By-Step Description	1. The user must type in search box 2. Click the associated button. 3. The system will search typed in the text in D.B.
Task Dependencies	Supervisor Login.
Goal	To search available Students names.
Post Conditions	The Supervisor will get founded Student name or nothing if D.B. will not find anything.
Proposed Module	Search management
Faculty Interface	

5. Communication

5.1 Messages

Action Name	Messages
Actors	Student / Faculty
Use Case Diagram	
Trigger	The user wants to communicate with another user.
Brief Description	The user communication system process
Initial Step-By-Step Description	1. The user will provide email or name of another user. 2. Type message text in the text box. 3. Click "Send". 4. The system will send a message to the intended person.
Task Dependencies	Student / Faculty Login.
Goal	To send a message to another user in system D.B.
Post Conditions	The user will send the message.
Proposed Module	Communication
Student/Faculty Interface	

UI Mock-ups

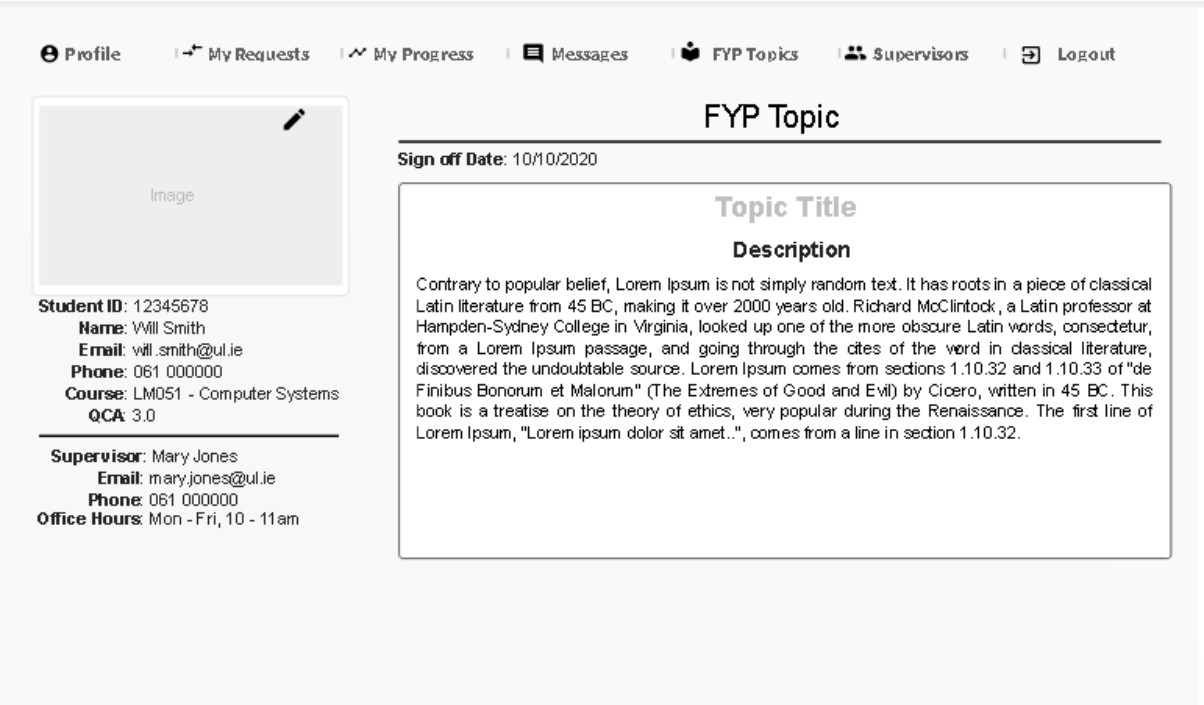
The purpose of these mock-ups was to start a discussion among the group to give the scope of the project and identify classes/entities that we would need as a base. Most likely, the design will be a lot different, but the underlying classes shall be the core of future implementation. We will not know till we start the development of this project what extra classes we will need.

Below is a handful of mock-up screens for the project.

Student Screens

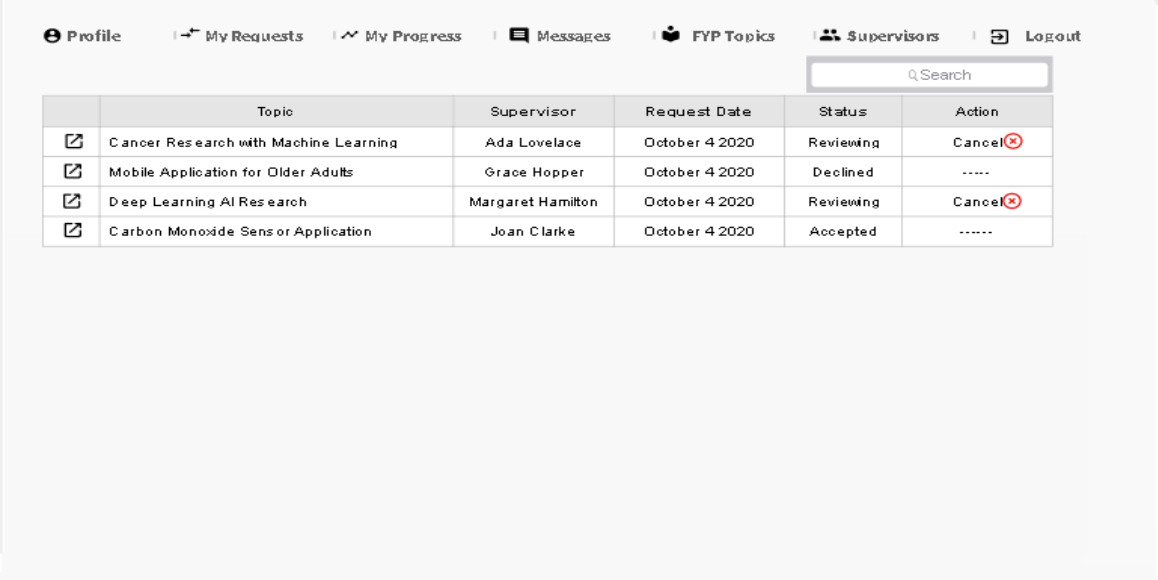
Student Profile

If a student gets selected by a supervisor for a topic, it will be displayed in the profile



My Request

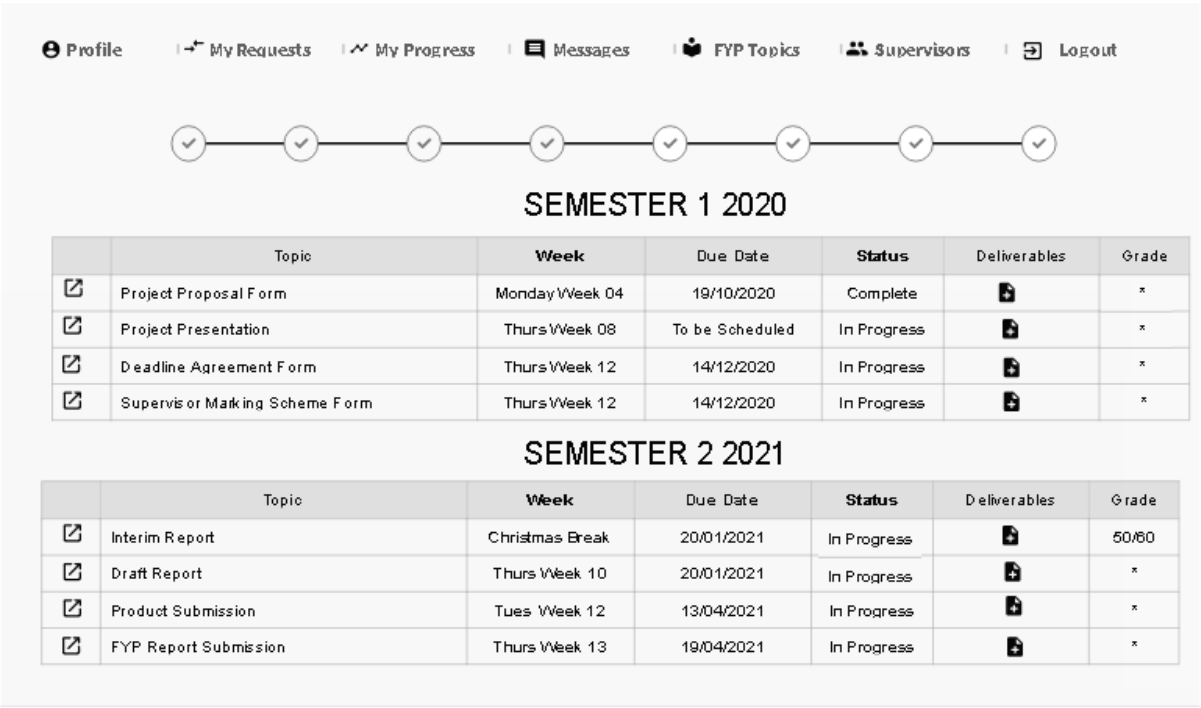
All the topics the Student has requested



	Topic	Supervisor	Request Date	Status	Action
	Cancer Research with Machine Learning	Ada Lovelace	October 4 2020	Reviewing	Cancel
	Mobile Application for Older Adults	Grace Hopper	October 4 2020	Declined	-----
	Deep Learning AI Research	Margaret Hamilton	October 4 2020	Reviewing	Cancel
	Carbon Monoxide Sensor Application	Joan Clarke	October 4 2020	Accepted	------

My Progress

Students Deliverables and Current Progress State



SEMESTER 1 2020

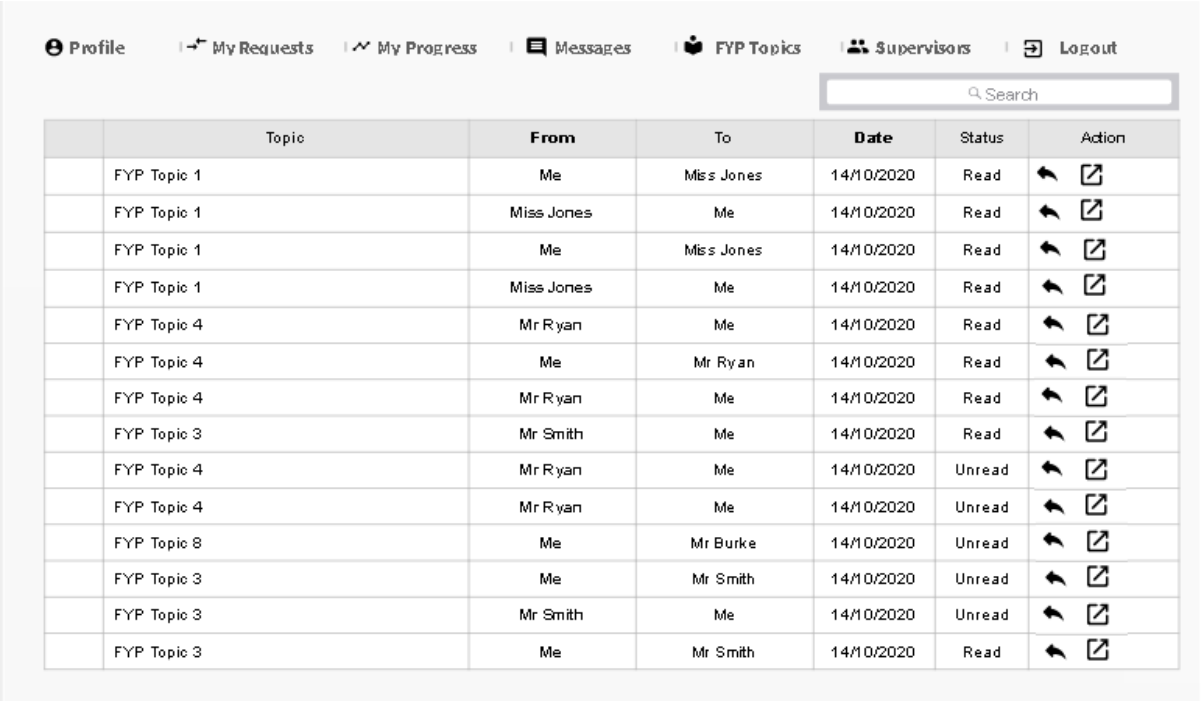
	Topic	Week	Due Date	Status	Deliverables	Grade
	Project Proposal Form	Monday Week 04	19/10/2020	Complete		*
	Project Presentation	Thurs Week 08	To be Scheduled	In Progress		*
	Deadline Agreement Form	Thurs Week 12	14/12/2020	In Progress		*
	Supervisor Marking Scheme Form	Thurs Week 12	14/12/2020	In Progress		*

SEMESTER 2 2021

	Topic	Week	Due Date	Status	Deliverables	Grade
	Interim Report	Christmas Break	20/01/2021	In Progress		50/60
	Draft Report	Thurs Week 10	20/01/2021	In Progress		*
	Product Submission	Tues Week 12	13/04/2021	In Progress		*
	FYP Report Submission	Thurs Week 13	19/04/2021	In Progress		*

Messages

All messages sent and received by the Student on a related topic request. The messages are for topics only, and general notifications should be emailed as not to clutter up the messaging.



	Topic	From	To	Date	Status	Action
	FYP Topic 1	Me	Miss Jones	14/10/2020	Read	
	FYP Topic 1	Miss Jones	Me	14/10/2020	Read	
	FYP Topic 1	Me	Miss Jones	14/10/2020	Read	
	FYP Topic 1	Miss Jones	Me	14/10/2020	Read	
	FYP Topic 4	Mr Ryan	Me	14/10/2020	Read	
	FYP Topic 4	Me	Mr Ryan	14/10/2020	Read	
	FYP Topic 4	Mr Ryan	Me	14/10/2020	Read	
	FYP Topic 3	Mr Smith	Me	14/10/2020	Read	
	FYP Topic 4	Mr Ryan	Me	14/10/2020	Unread	
	FYP Topic 4	Mr Ryan	Me	14/10/2020	Unread	
	FYP Topic 8	Me	Mr Burke	14/10/2020	Unread	
	FYP Topic 3	Me	Mr Smith	14/10/2020	Unread	
	FYP Topic 3	Mr Smith	Me	14/10/2020	Unread	
	FYP Topic 3	Me	Mr Smith	14/10/2020	Read	

FYP Topics

Students can view and request the FYP topics. They can also see the current requests for that topic

	Topic	Supervisor	QCA	Course	Requests	Action
	FYP Topic 1	Miss Jones	3.0	Computer System	3	
	FYP Topic 1	Mr Ryan	3.6	Games	1	
	FYP Topic 1	Mr Smith	2.5	Computer System	0	
	FYP Topic 1	Antone Garrud	3.0	Games	4	
	FYP Topic 4	Arel Bowle	2.5	Computer System	2	
	FYP Topic 4	Kit Skittrall	2.8	Games	5	
	FYP Topic 4	Base Summerside	2.8	Computer System	0	
	FYP Topic 3	Shawnee Stoller	2.8	Computer System	0	
	FYP Topic 4	Bronson Moohan	3.0	Games	2	
	FYP Topic 4	Genny Ranklin	3.6	Computer System	3	
	FYP Topic 8	Ado Gergely	3.0	Computer System	2	
	FYP Topic 3	Rosalia Le land	3.6	Computer System	3	
	FYP Topic 3	Caresa Dawltrey	3.6	Games	3	
	FYP Topic 3	Tony Lynch	2.5	Games	1	

Supervisors

Students can search through the supervisors

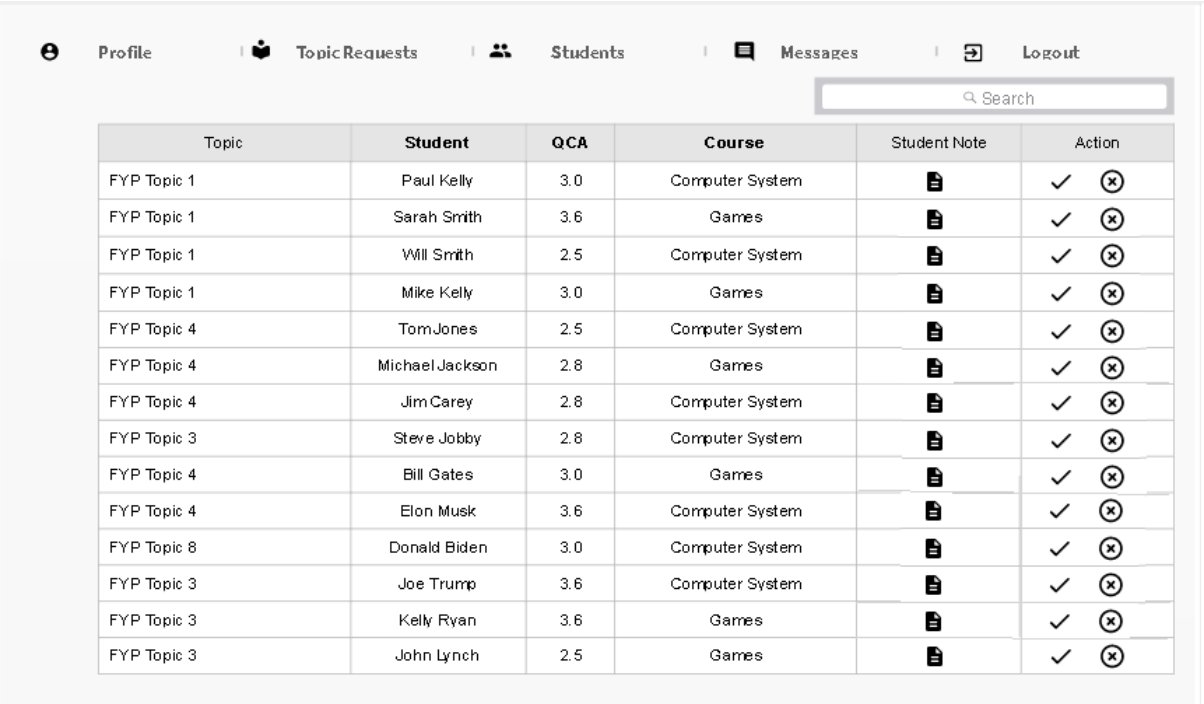
Faculty Screens

Profile

This screen will be like Student

Topic Requests

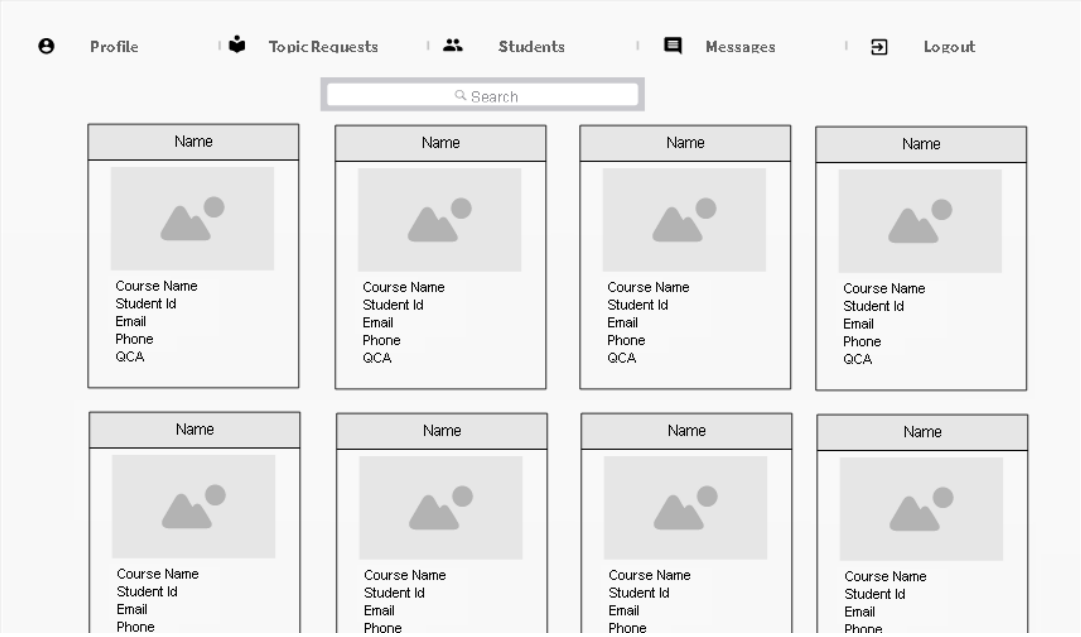
All requests for Topics linked to the Supervisor. Student note lets the Student attach a useful message to the Supervisor on why they would like this topic. Supervisor and accept, review, decline.



Topic	Student	QCA	Course	Student Note	Action
FYP Topic 1	Paul Kelly	3.0	Computer System		
FYP Topic 1	Sarah Smith	3.6	Games		
FYP Topic 1	Will Smith	2.5	Computer System		
FYP Topic 1	Mike Kelly	3.0	Games		
FYP Topic 4	Tom Jones	2.5	Computer System		
FYP Topic 4	Michael Jackson	2.8	Games		
FYP Topic 4	Jim Carey	2.8	Computer System		
FYP Topic 3	Steve Jobby	2.8	Computer System		
FYP Topic 4	Bill Gates	3.0	Games		
FYP Topic 4	Elon Musk	3.6	Computer System		
FYP Topic 8	Donald Biden	3.0	Computer System		
FYP Topic 3	Joe Trump	3.6	Computer System		
FYP Topic 3	Kelly Ryan	3.6	Games		
FYP Topic 3	John Lynch	2.5	Games		

Students

Supervisors can search through all the student profiles



Messages

This screen is the same as Students (see above)

MVC

For our project proposal, we have decided to use Laravel, which is an MVC framework written in PHP. The MVC architectural design pattern has implementations in languages such as C#, Java, Php, Django, Flask, Sinatra and many more. The separation of components leads to an easier to maintain code, and more a cohesive code structure. The MVC pattern makes it easier for the tester to create tests, and it is ubiquitous to see dependency injection with MVC type frameworks. An external library such as AutoMapper for C# would inject services into the constructor and decouple code that would usually have some dependencies if no DI were used. There is a clear separation from functionality and U.I.

Model

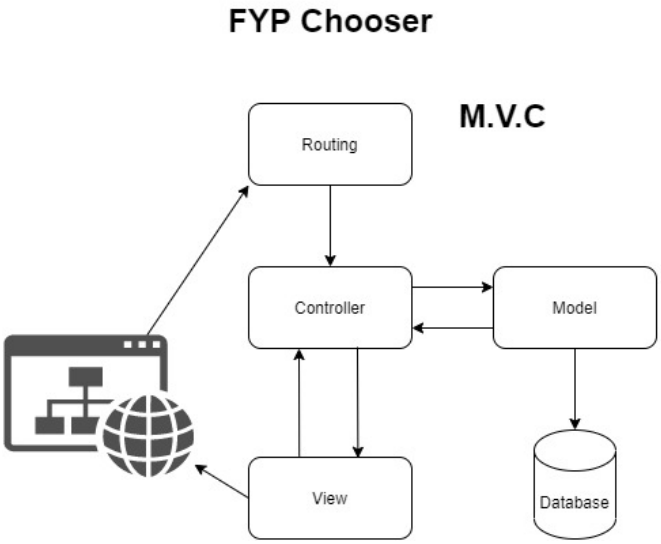
The model is responsible for getting and manipulating the data. You can think of the model as the brain of the application that usually interacts with the Database. There are times where a model could interact with other types of storage, such as a JSON file, so it is not always a database. The model communicates with the controller, and the controller can request data from the model, then the controller can be used to update the View.

View

The View is responsible for code related to the user interface such as HTML, CSS. The View would communicate with the controller and use data received to build up the pages. An example of this can be seen in the topics section of the site where the table is built from data received from the controller.

Controller

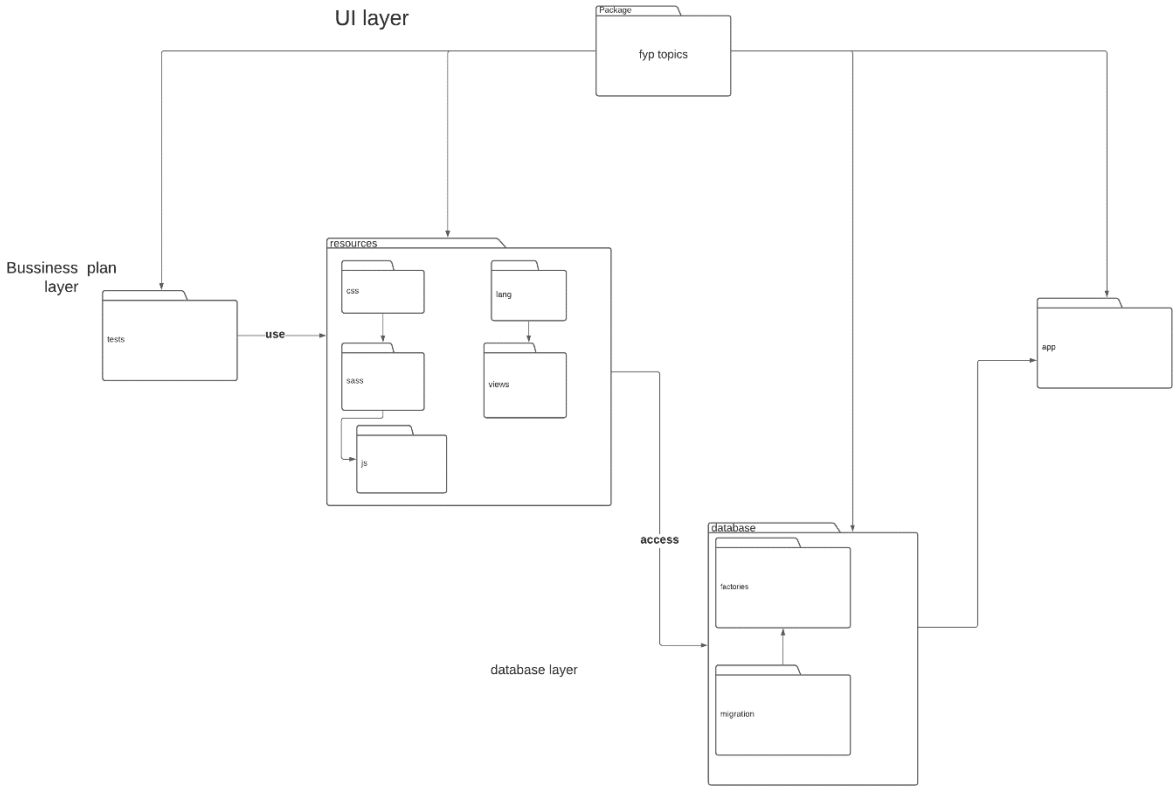
The controller is responsible for tying the model data to the views. The controller is called when a user makes a PUT/POST/PATCH request on an endpoint such as http://127.0.0.1/topics. The controller is the coordinator between the View and the model. Templating engines such as blade in Laravel can do manipulate data and display data received from the controller. The controller can also pass a plain view with plain HTML and CSS.



System Architecture Overview

Architecture Diagram

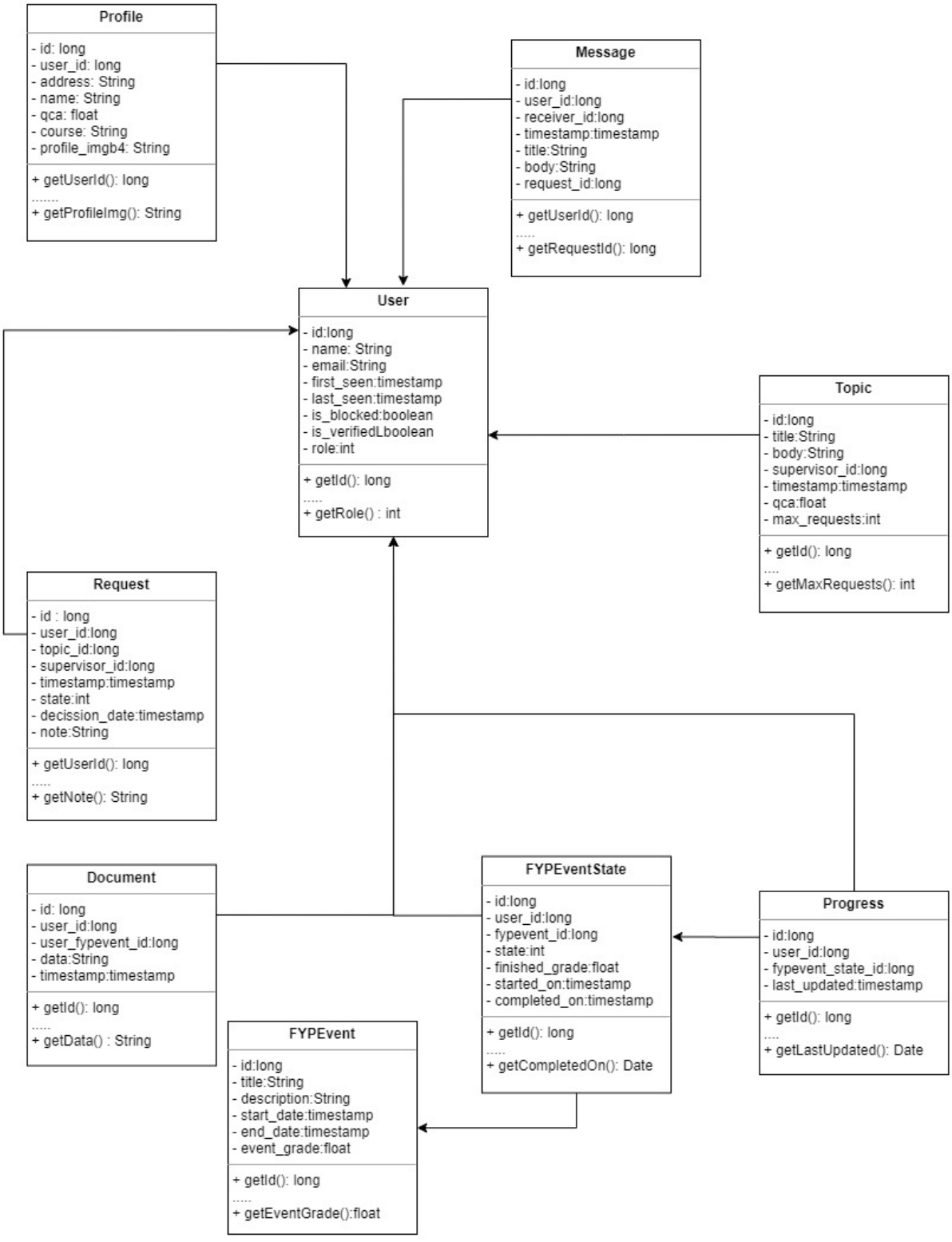
For our project, we decide to use a package diagram to understand better the system architecture overview of how our class would be laid out. This is due to the simplicity of the folder layout that we can create before implementing any code. Our diagram is made of three layers which are the U.I. layer, the business plan layer, and the database layer. PhpStorm allows us to start with a template website in the U.I. layer and will help us change that into the desired business plan layer. Once we have our business plan layer is created, we can then develop a unified database in the created folder.



Object-Oriented Analysis (OOA) + Database

Class Analysis Diagram

These are our initial Classes and open to change upon implementation. We created these during the analysis phase

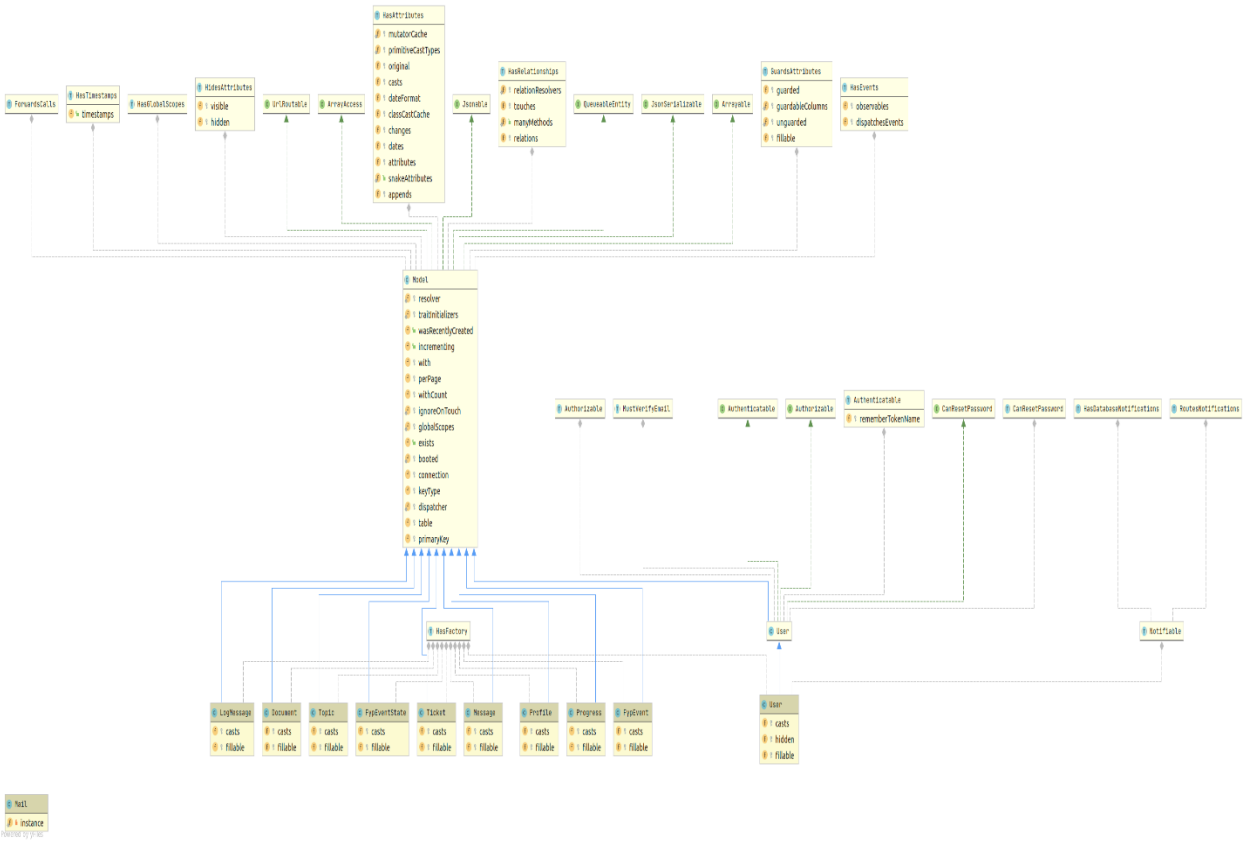


Module Identification

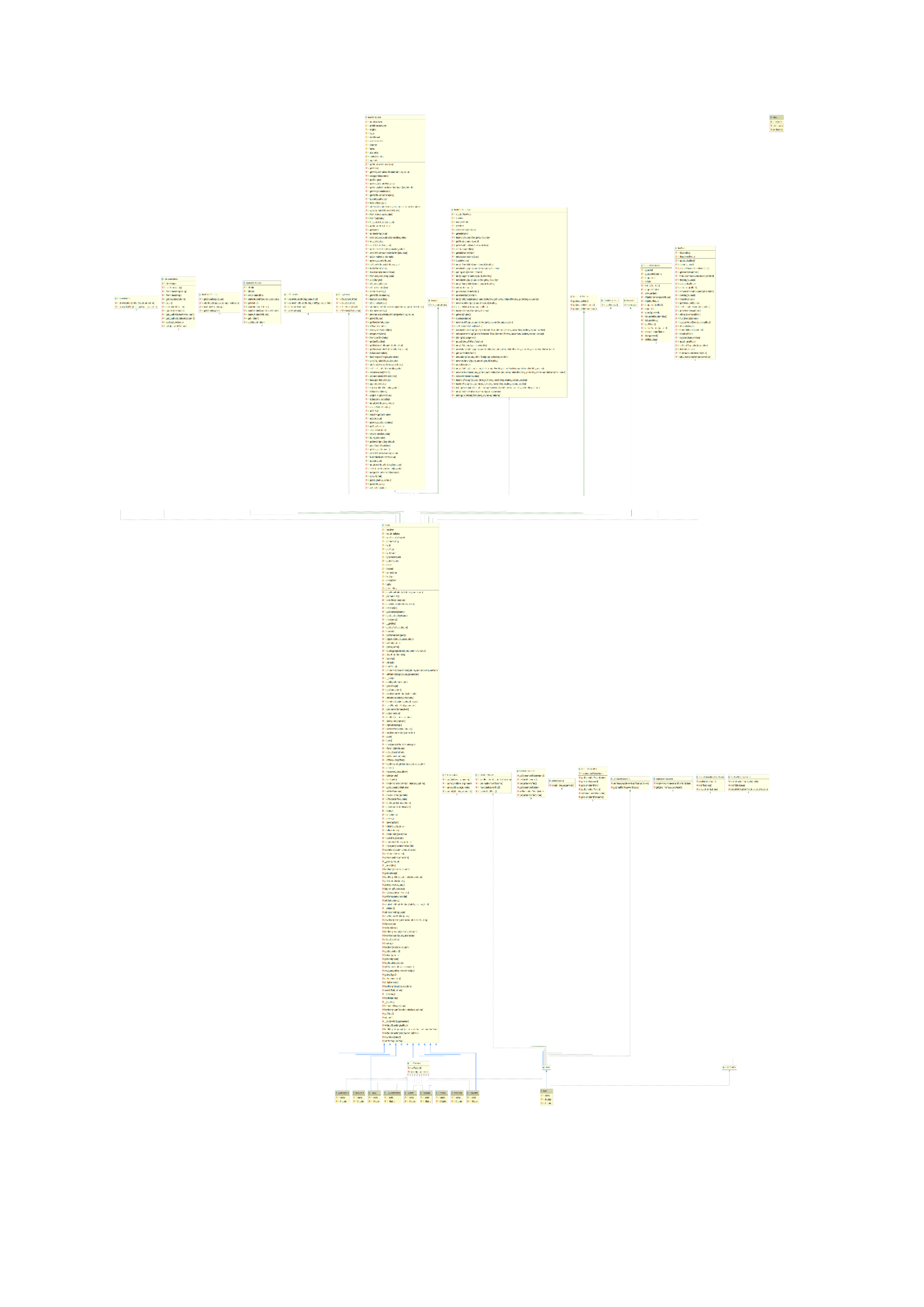
The methods we used to identify the required class diagram were based on group discussions carried out after identifying the requirements. We began by coming up with the necessary class names that we were going to use after analysing the use case descriptions above. We then identified the classes and objects that we were going to use and how they link with one another. As a group, we discussed how these classes and Objects could be implanted using the PhpStorm. This was important because it is necessary to understand how that IDE would allow us to implement these classes to create a class diagram.

Class Diagram

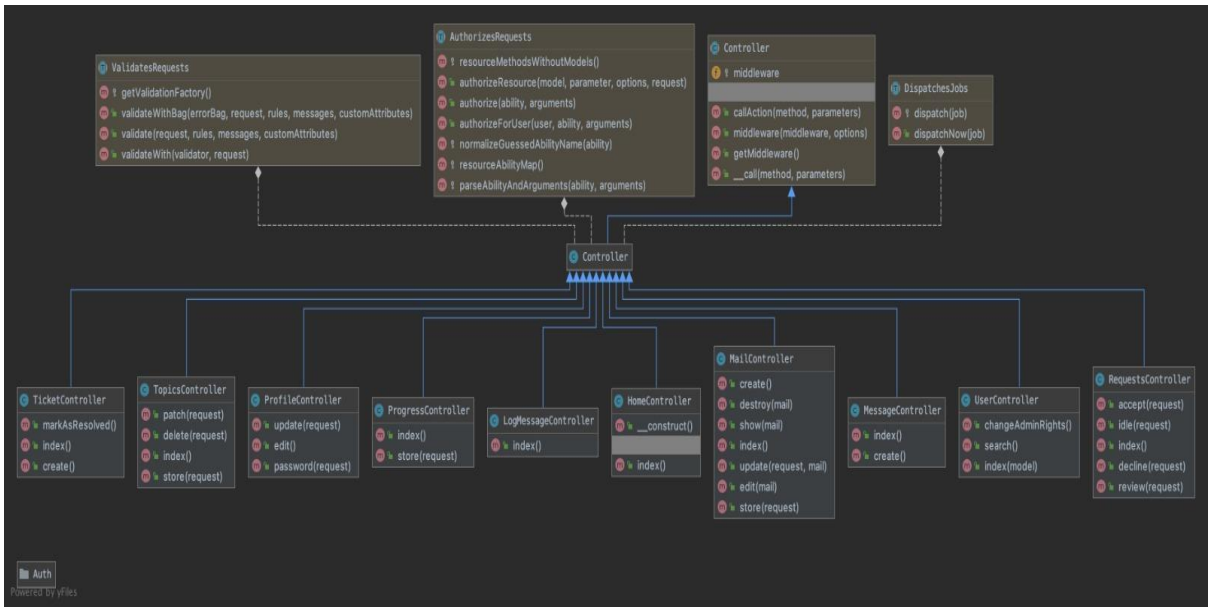
Simplified version



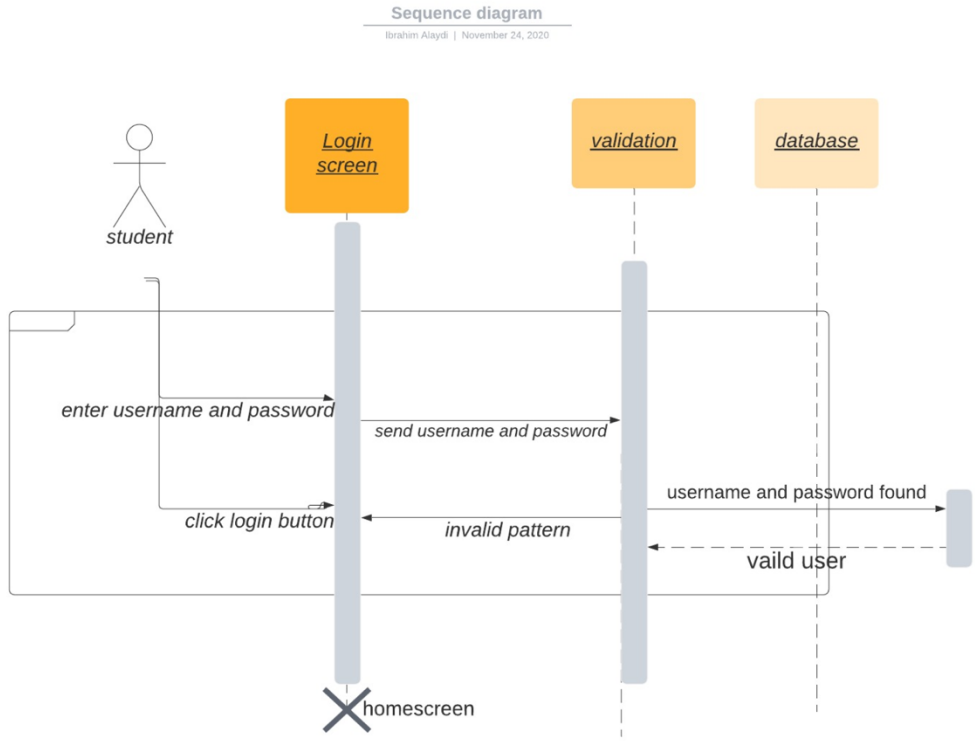
Full version



Controller Diagram



Sequence diagram



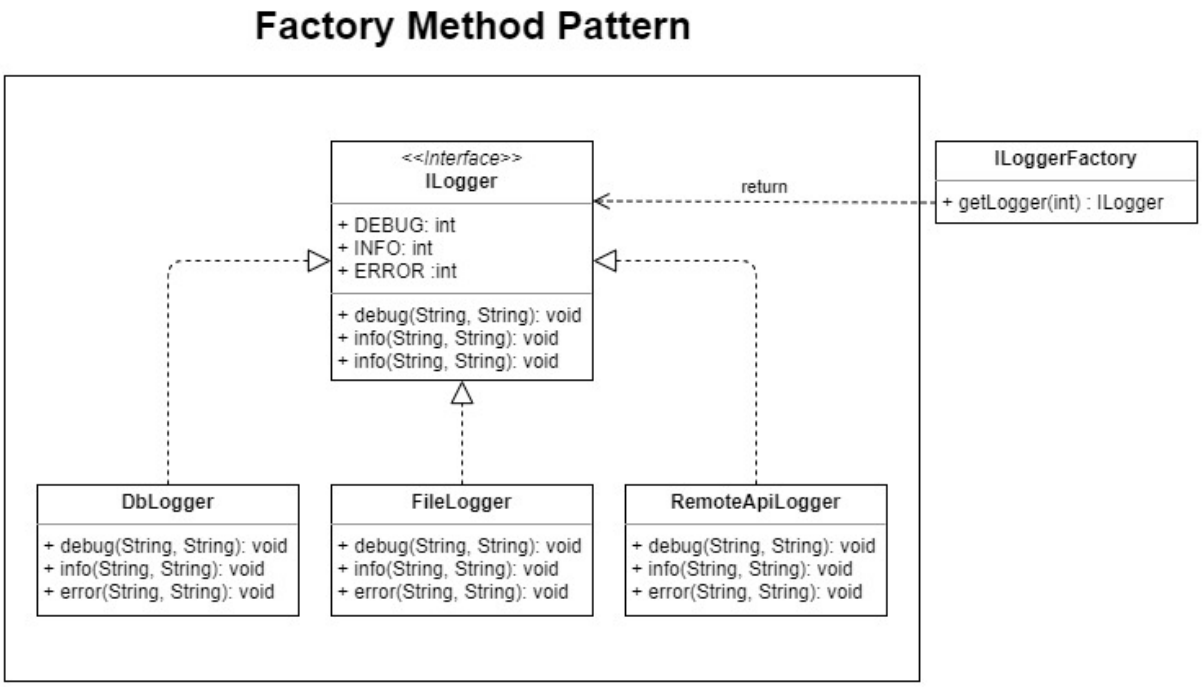
Entity Diagram

Please zoom in to see relationships. * = Many, 1 = One to One relationship

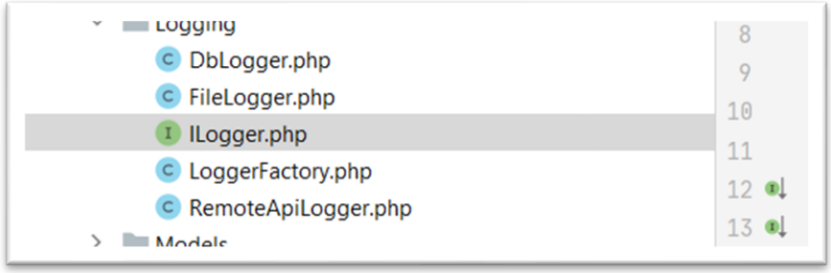
Design Patterns (OOD)

Factory Method Pattern (Creational)

We decided to implement the Factory Method Pattern in a logging feature in the application. There are cases where we need to log to a file and log to a database or both. Logging to the Database allows the administrator to quickly view logs of a failed lot in attempts among other things. We create a common ILogger interface with logging methods such as debug, info, and error. The beauty of this pattern is no change is required in the code if a new Logger implementation as added (new is glue). The caller knows that the contract will adhere to methods defined in the interface. This logger has come in very useful for debugger issue throughout out or development. See page 107 of GOF book



To keep in-line with good cohesion we created all the related files in a logging package



```php
<?php


namespace App\Logging;


interface ILogger
{
    const DEBUG = 0;
    const INFO = 1;
    const ERROR = 2;
    public function debug($tag, $message);
    public function info($tag, $message);
    public function error($tag, $message);
}
```

To add a new logging source, the class needs to implement the ILogger interface. We have found it debug, info, and error sufficient for the logging hence why we have added them as const fields in the interface.

```php
{
    const LOGGER_CODE = 0;

    private function log($tag, $message, $type) {
        $log = new LogMessage();
        $log->tag = $tag;
        $log->message = $message;
        $log->type = $type;
        $log->save();
    }
    public function debug($tag, $message)
    {
        $this->log($tag, $message, type: self::DEBUG);
    }

    public function info($tag, $message)
    {
        $this->log($tag, $message, type: self::INFO);
    }

    public function error($tag, $message)
    {
        $this->log($tag, $message, type: self::ERROR);
    }
}
```

DbLogger

Implementation of the ILogger interface. LOGGER_CODE is used in the factory method to distinguish which logger to return.

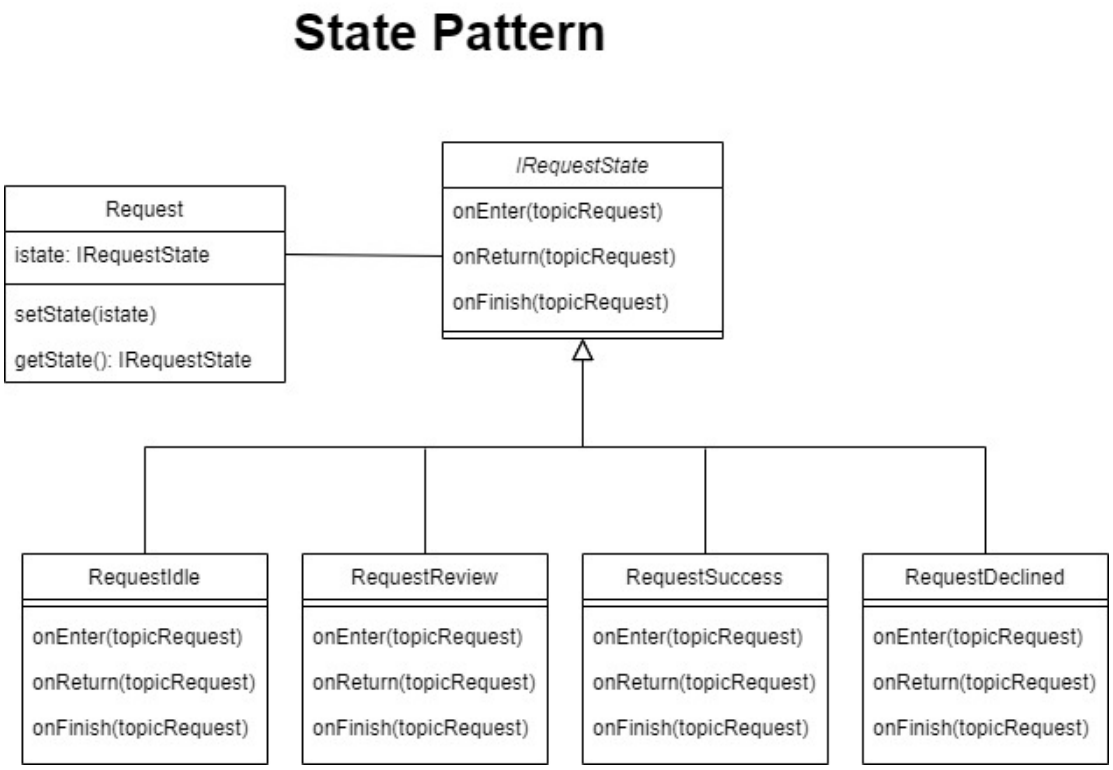
This class does not need to be instantiated, so we added a private constructor. Php is not strongly typed like Java; hence there is no return type like ILogger defined in the method. If this were Java, this method would be public static ILogger getLogger(….). Type checking is completed at runtime.

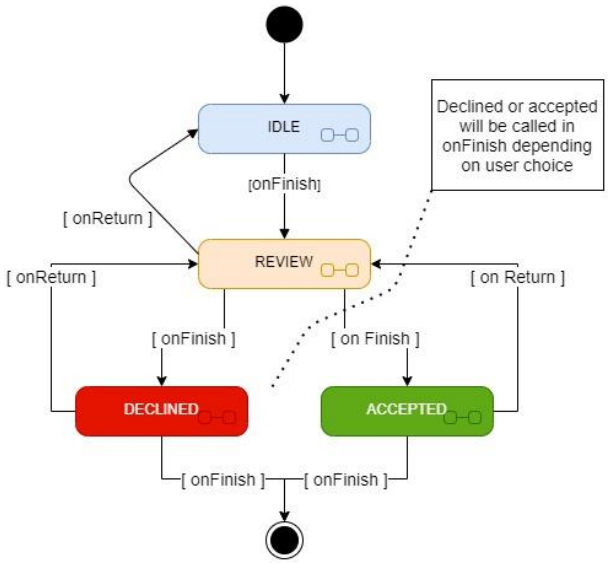
Have the newing up of classes in one location makes the code less breakable and more comfortable to maintain and test. If we use new Logger implementations throughout the code and decided to change a constructor, we will break the code in many places. With this pattern, all this is avoided

```php
class LoggerFactory
{
    private function __construct() {
    }

    /**
     * @param $loggerCode
     * @return DbLogger|FileLogger|RemoteApiLogger
     * Default is DbLogger
     */
    public static  function getLogger($loggerCode) {
        switch ($loggerCode) {
            case FileLogger::LOGGER_CODE: return new FileLogger();
            case RemoteApiLogger::LOGGER_CODE: return new RemoteApiLogger();
        }

        return new DbLogger();
    }
}
```

State Pattern (Behavioral)

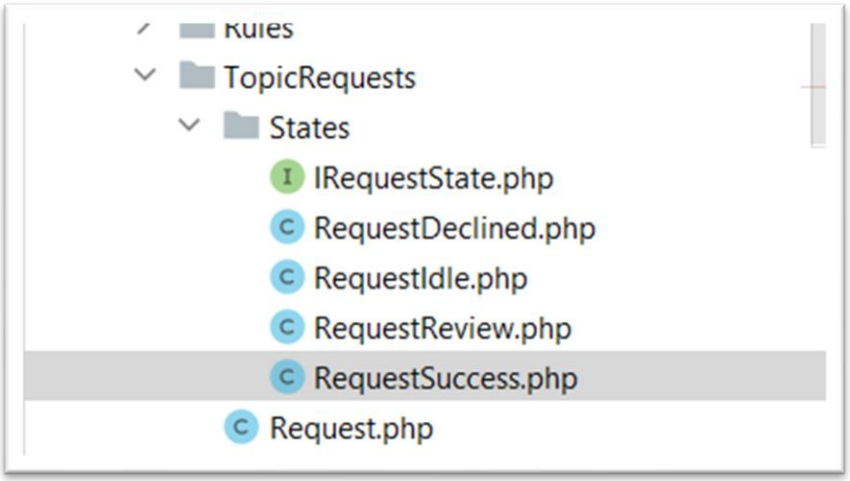
We have implemented the state pattern to perform different actions depending on the state the request was in. For example: if the request for a student was successful, the onFinish method would be called in the RequestSuccess state. In this method, there is code to remove all the requests that Student has for other topics. The method call also starts the student progress timeline logic and sets the first deliverable to IN_PROGRESS.





Here is a sample flow of how a topic request might go. By default, any request is put into the IDLE state. If a supervisor is interested in the Student but needs to review the state is moved into the review, onEnter will be called, and a message will be sent to the user in the method that his request is being reviewed. The Supervisor can put back into the IDLE state if they please or move to Accepted or Declined.

In this image with have the state interface, the four states and the Request context.

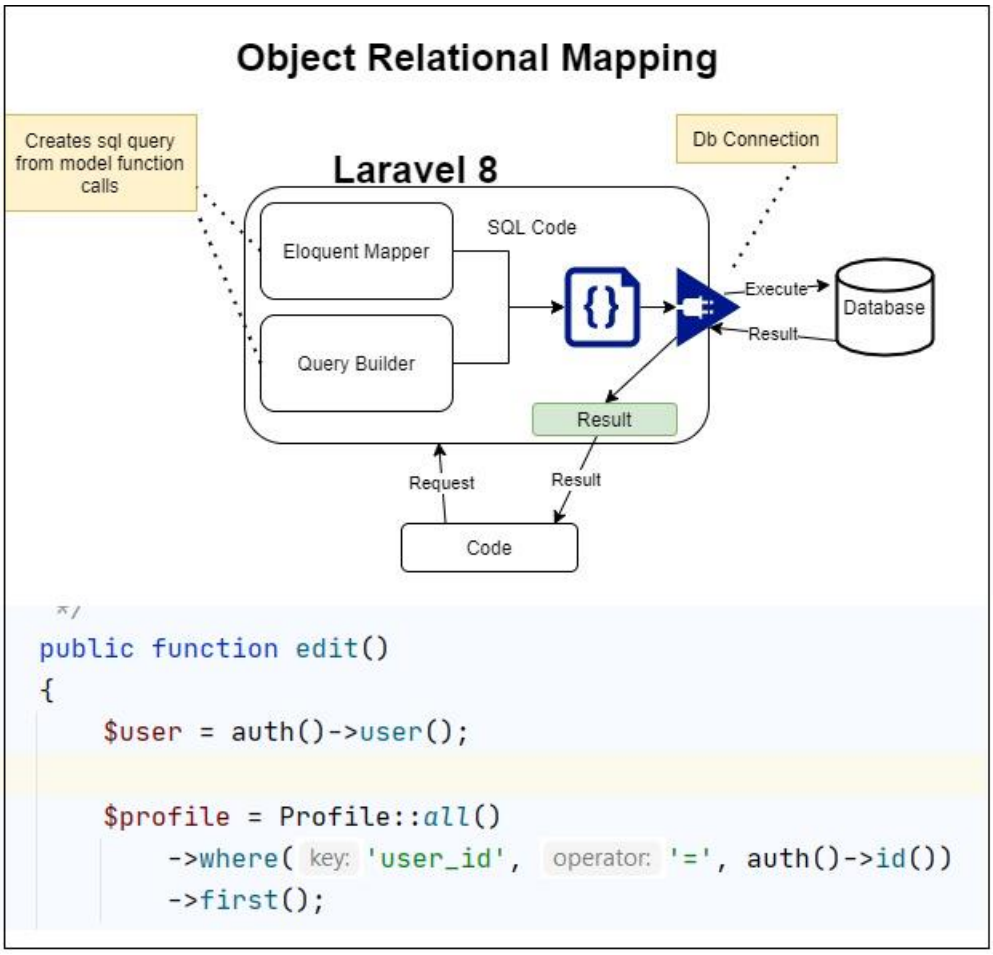


onFinish RequestSuccess State

```php
public function onFinish($topicRequest)
{
    $topicRequest->setState($topicRequest::SUCCESS);
    $topicRequest->save();

    $userRequests = \App\TopicRequests\Request::all()
        ->where( key: 'user_id', $topicRequest->user_id)
        ->where( key: 'state',  operator: '!=', $topicRequest::SUCCESS);

    $this->dropUserRequests($userRequests);
    $this->dropTopicRequests($topicRequest);
    $fypevent = $this->setFirstEventActive($topicRequest);
    $this->setProgress($fypevent);

    $this->logger->debug('RequestSuccess:onFinish', 'State finished');
}
```

We have decided to highlight the ORM design pattern, which is present in Laravel because of this type of pattern used in most MVC frameworks. The code first approach helps developers focus on code, and they do not have to worry about writing complex raw SQL queries. The data source in the ORM model can be changed without affecting the calling code, which leads to easy to maintain code and decouples dependencies for a particular database. The same concept can be and is used in Laravel to build the database entities. With a code-first approach, the Database is entities, and SQL code is created based on the model. You can see this code first implementation if you look in the migrations folder of our project

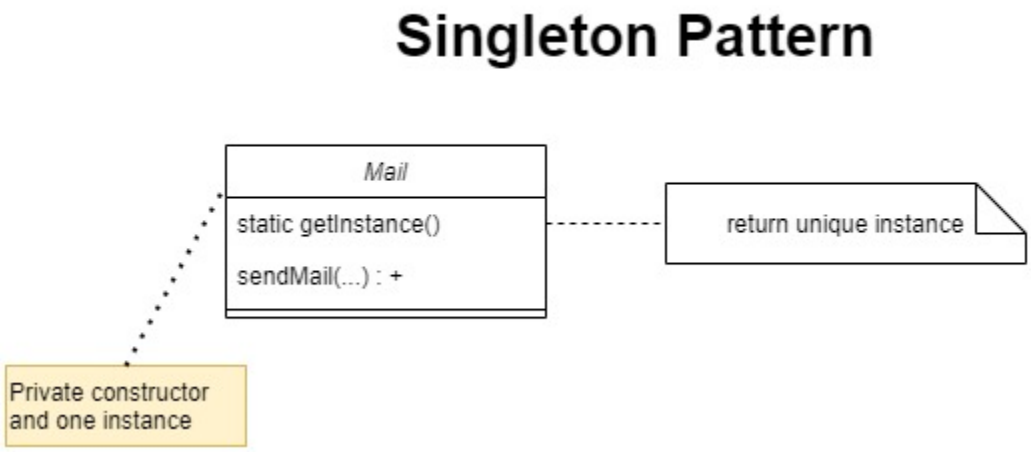
The diagram below shows a sequence of method calls. The query builder will convert these calls into SQL code with is then run on the Database. The result is converted back into models or a model.




```
*/
public function edit()
{
    $user = auth()->user();

    $profile = Profile::all()
        ->where( key: 'user_id', operator: '=', auth()->id())
        ->first();
```

Singleton Pattern (Creational)

We have used the singleton pattern to ensure that there was only one instance of the Mail client. This class is used from sending mail. However, we have not implemented a queuing service in this project for in-memory mail due to insufficient time. The instantiation of the Mail client is also heavy; therefore, it made sense to use the singleton pattern. A singleton pattern is a viable choice, as only one live connection is required to our mailing service, which takes to, from, topic and message parameter.



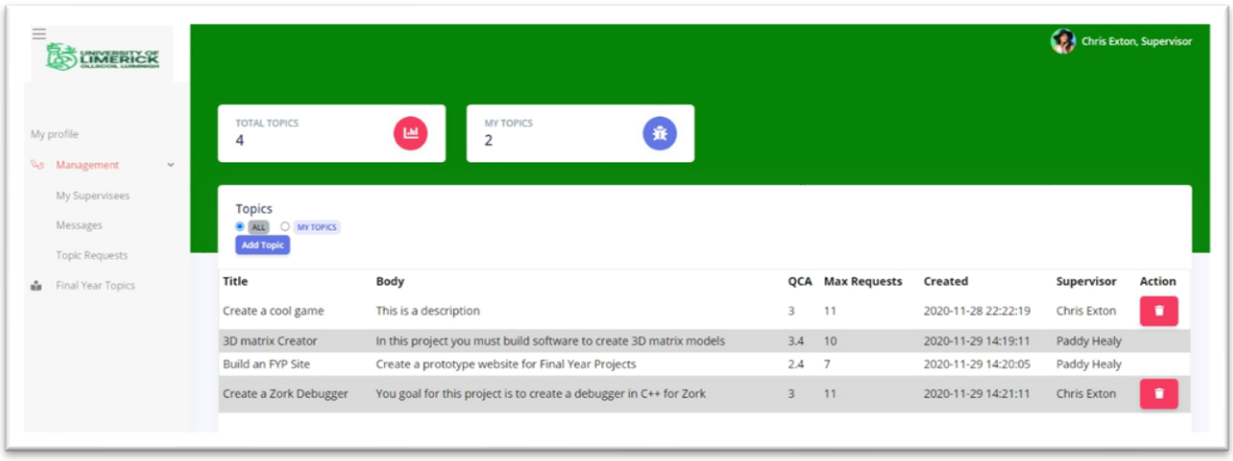
System Design & Implementation

User Interface

This section will showcase the implementation of the mock-ups created early in the design phase

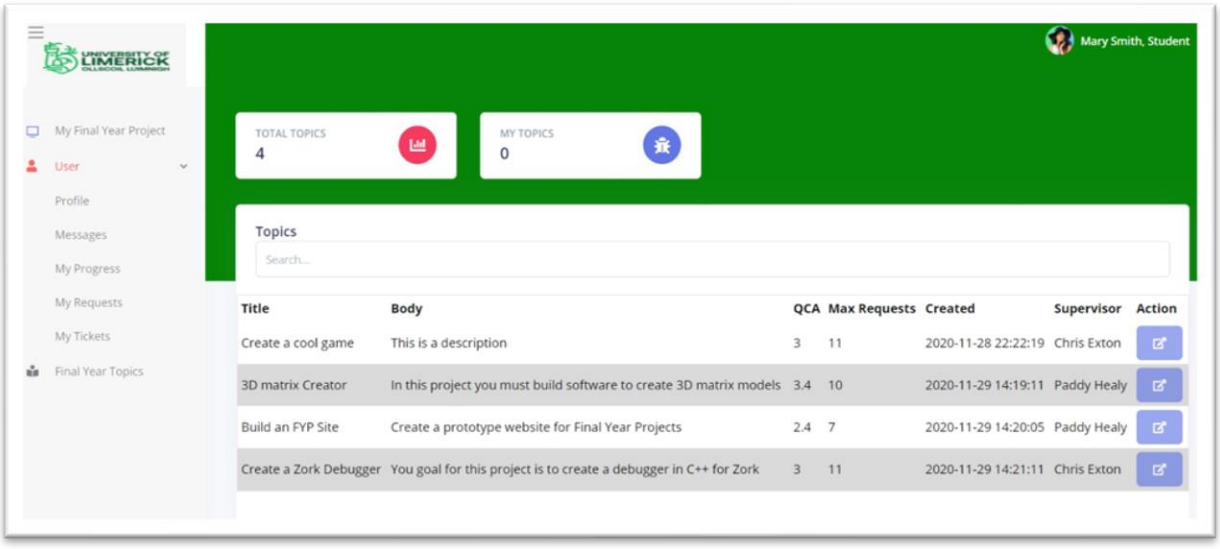
Supervisor Topic CRUD

Supervisors can create, delete, and view his/her topics in this section. They can also view other supervisor topics but cannot delete them. They can only delete their own.



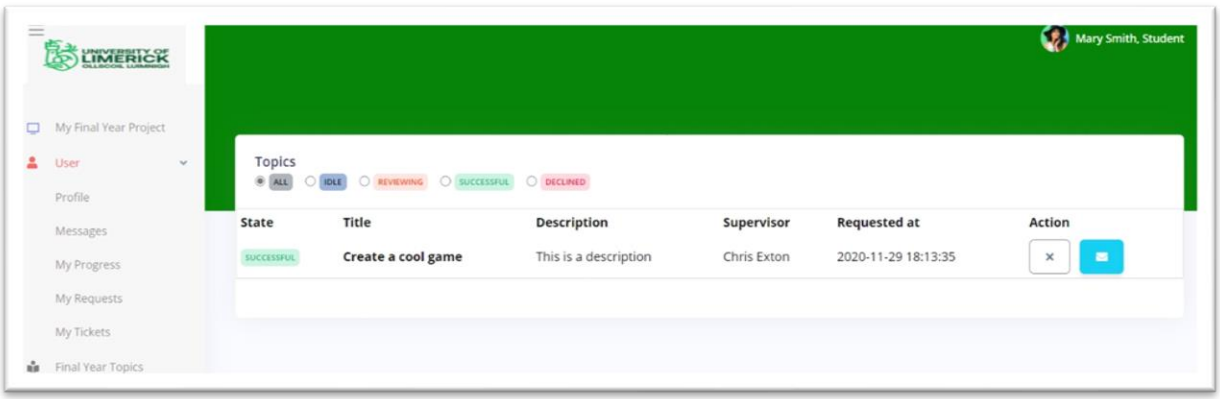
Student Topic Request

A student can request as many topics as they like. The screenshot below shows the request button has been disabled because the Student was successful with a request, therefore, cannot request a new one



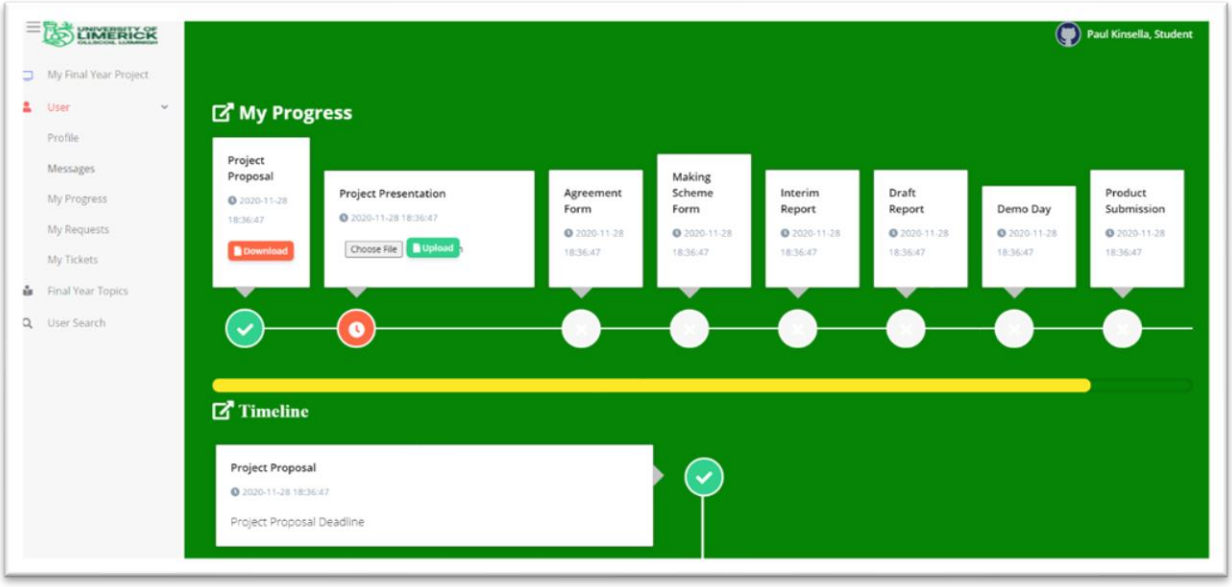
Student requests overview

When a student has been successful, all their other pending requested will be dropped. They are no longer able to request new topics once a supervisor has selected them. The white cancel button indicates that Student cannot cancel this request at this time.



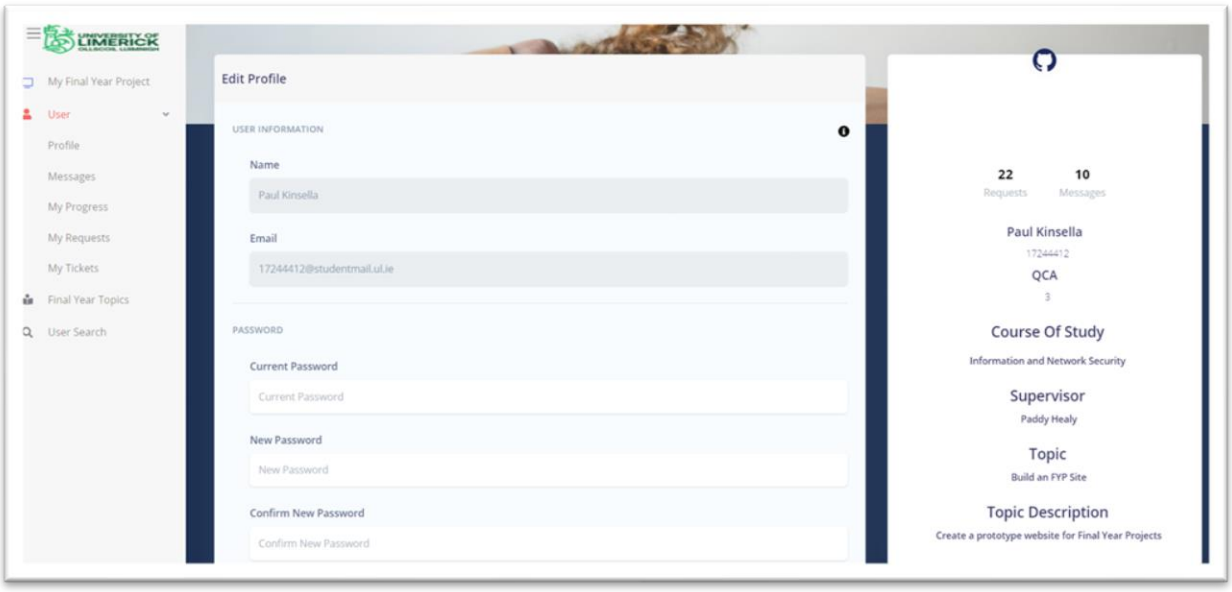
Student Progress

When a student first registers, they will be giving a set of deliverables. Only when a supervisor selects that Student for an FYP Topic will the timeline be active. A short horizontal timeline will help the Student visualise what is to be done by date. The vertical timeline has more detailed information on each deadline



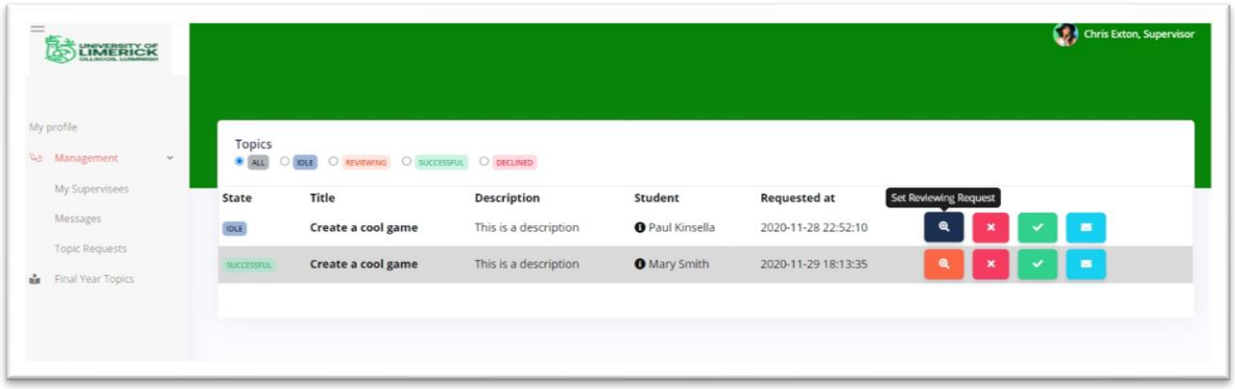
User Profile

When a user has been selected for an FYP Topic, the Supervisor and the topic will appear in their profile. They can change their password here if needs be



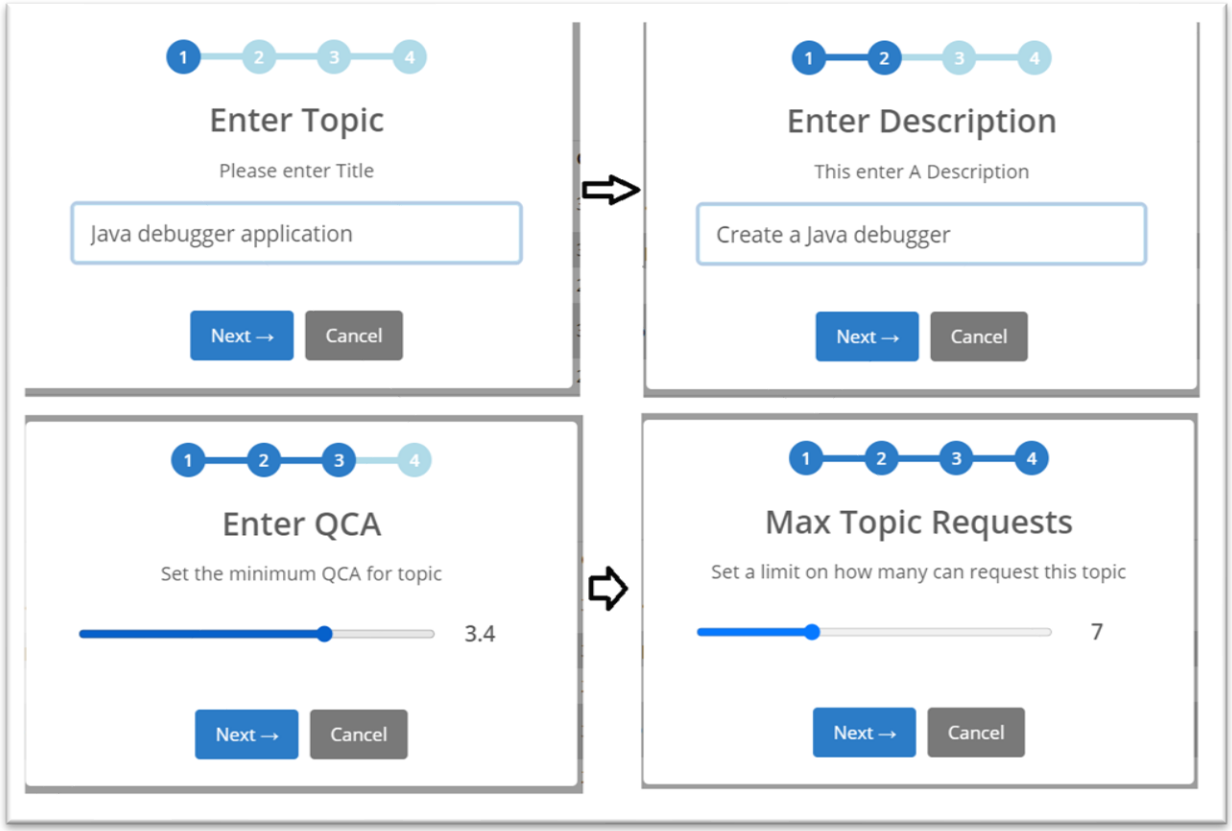
Supervisor Request Page

This section allows supervisors to see a request from students for a topic. There are four states in a request. Idle, Reviewing, Success, Decline. When the Supervisor is interested in the Student's request, they will click the set reviewing button.



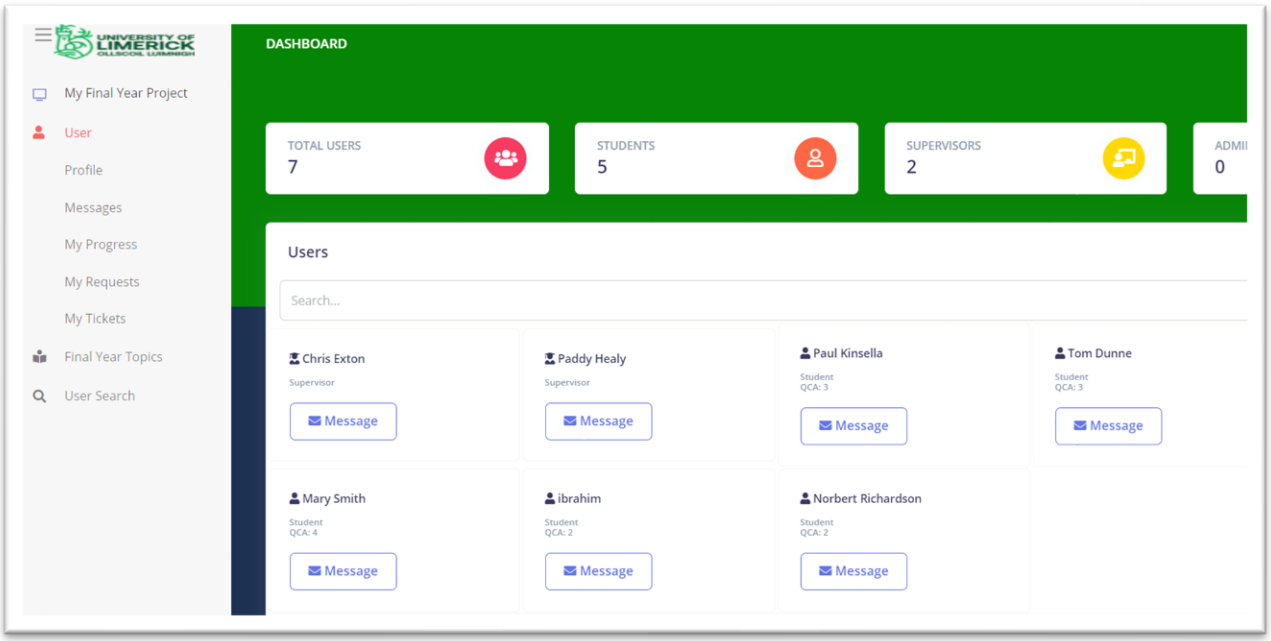
Add Topic

Screenshots of a supervisor creating a new topic



User Search

This section allows all members to search each for other members. The user is a student the QCA will be shown. This can help pro-active supervisors look for students with a high QCA and ask them if they would be interested in doing a topic. It also serves as a place for students to find Supervisors.



Frameworks

Bootstrap

Bootstrap is a free and open-source CSS front end framework tailored for faster and easier web development of web applications.

Laravel

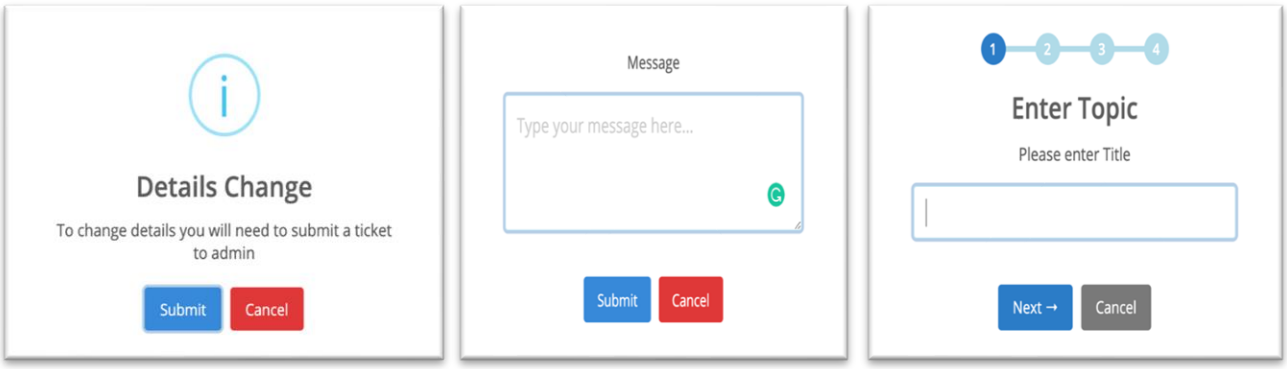
Laravel is a powerful MVC PHP framework, designed for developers who need a simple and elegant toolkit to create full-featured web applications. We decided to settle with Laravel due to our rich experience with PHP in previous projects. Laravel also makes it simple for those who have no prior experience using an MVC framework and is too easy for those who have no previous experience with PHP. We had taken Django consideration, but due to not everyone having prior experience with Python, we have settled on Laravel.

FontAwesome

FontAwesome is a font and icon toolkit based on CSS and Less. It is quick and easy to use with the Bootstrap mentioned above, the free tier contains a lot of icons which can be browsed on fontawesome.com

SweetAlert2

SweetAlert2 is an open-source library of responsive, customisable popup boxes that is hosted on GitHub and actively updated. We used it in our project to replace the standard popup boxes that are available by default. Source - https://sweetalert2.github.io/



Testing

For the PHP programming language, Sebastian Bergmann's PHPUnit is an advanced PHP unit testing framework. Example of the xUnit architecture that started with SUnit for unit testing frameworks and became popular with JUnit. Unit testing aims to isolate each part of the software and demonstrate the correctness of the individual pieces. A unit test provides a strict, written contract that must be fulfilled by the selection of code. As a result, early in the development cycle, unit tests identify issues. PHPUnit is based on the idea that developers should be able to find mistakes in their newly committed code. This way quickly asserts that no code regression has occurred in other parts of the codebase.

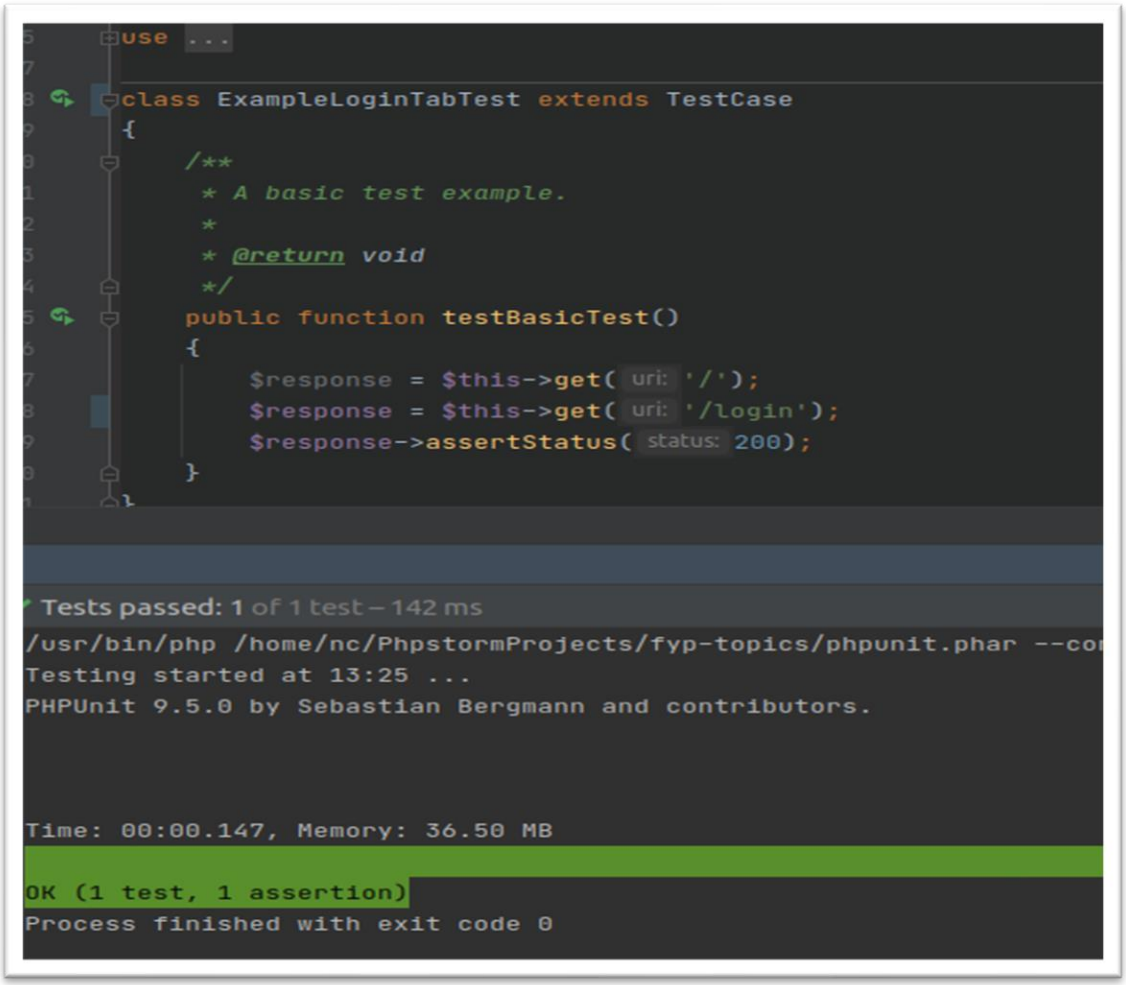
Much like other unit testing frameworks, PHPUnit uses assertions to verify that the behaviour of the specific component - or "unit" - being tested behaves as expected (PHPUnit, 2020).

The tests are shown here intended to simulate the requests to access the FYP website by a user's browser. The idea behind automatic testing is to simplify this process to the shortest possible time, which must spend every software developer on test the product for any changes made during the code update. Manual testing requires a lot more resources related to people and the money allocated to it. As mentioned above, the tests presented below are only based on user authorisation which is one of the foundations of any software product.

Test 1

Login Page Test

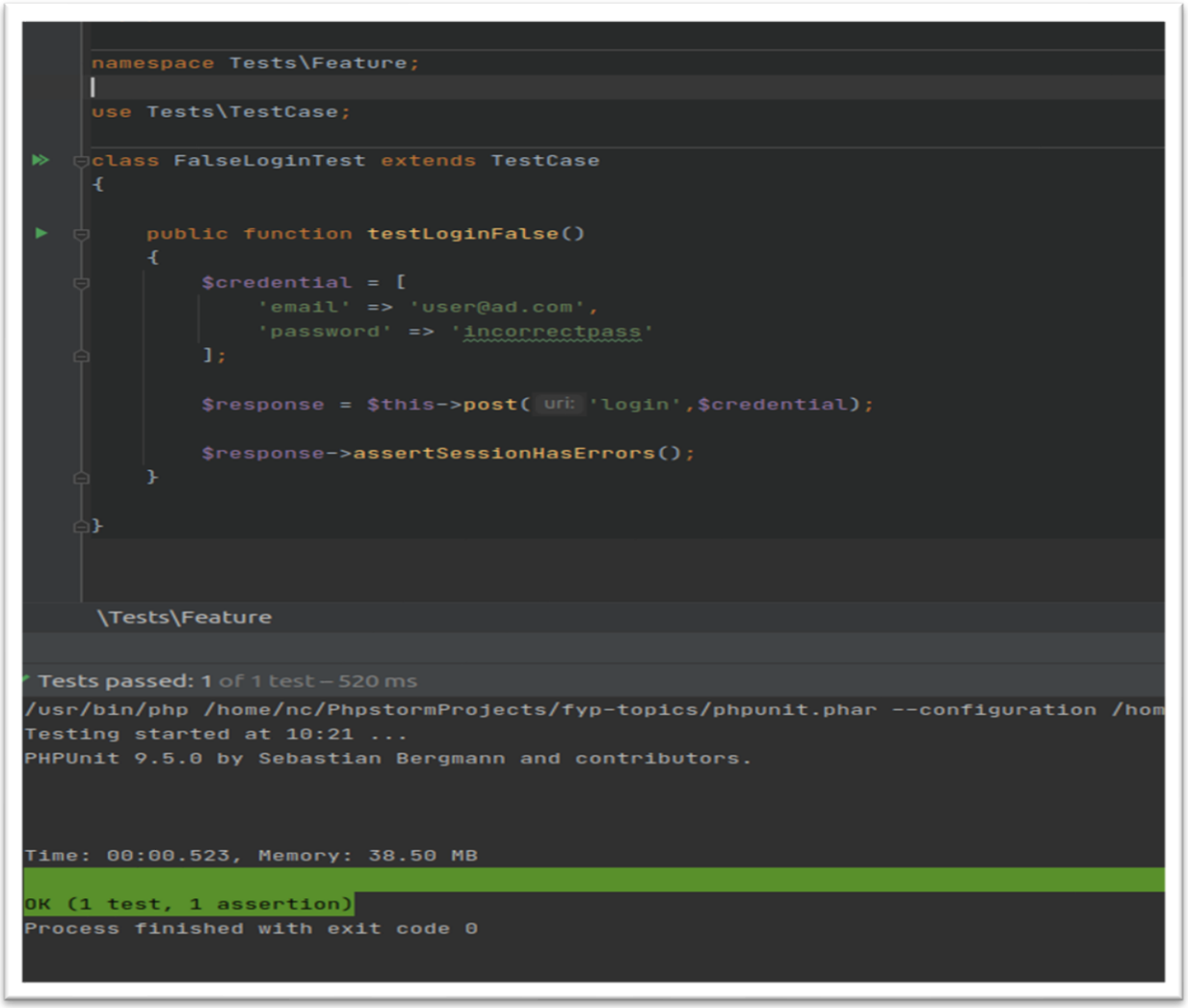
The picture below shows a basic test in which the correctness of going to the login page tested if the user wants to log in. assertStatus() function returns "True" if the value is "200".



Test 2

False Login Test

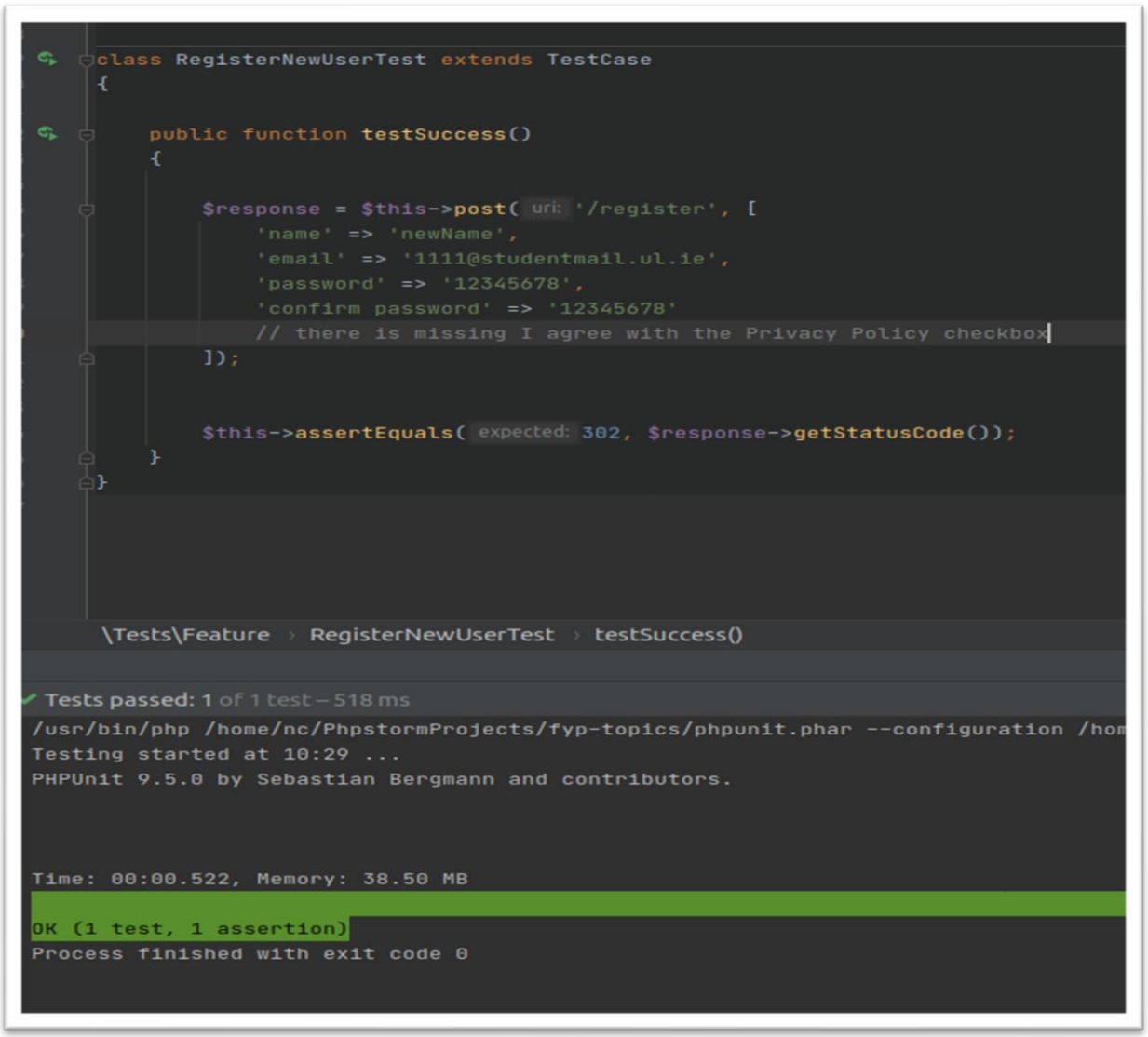
The second test checks the correctness of the data entered for user authorisation. assertSessionHasError () function, checks entered user data with data stored in database. Any incorrect input data returns the value "True".



Test 3

Register New User Test

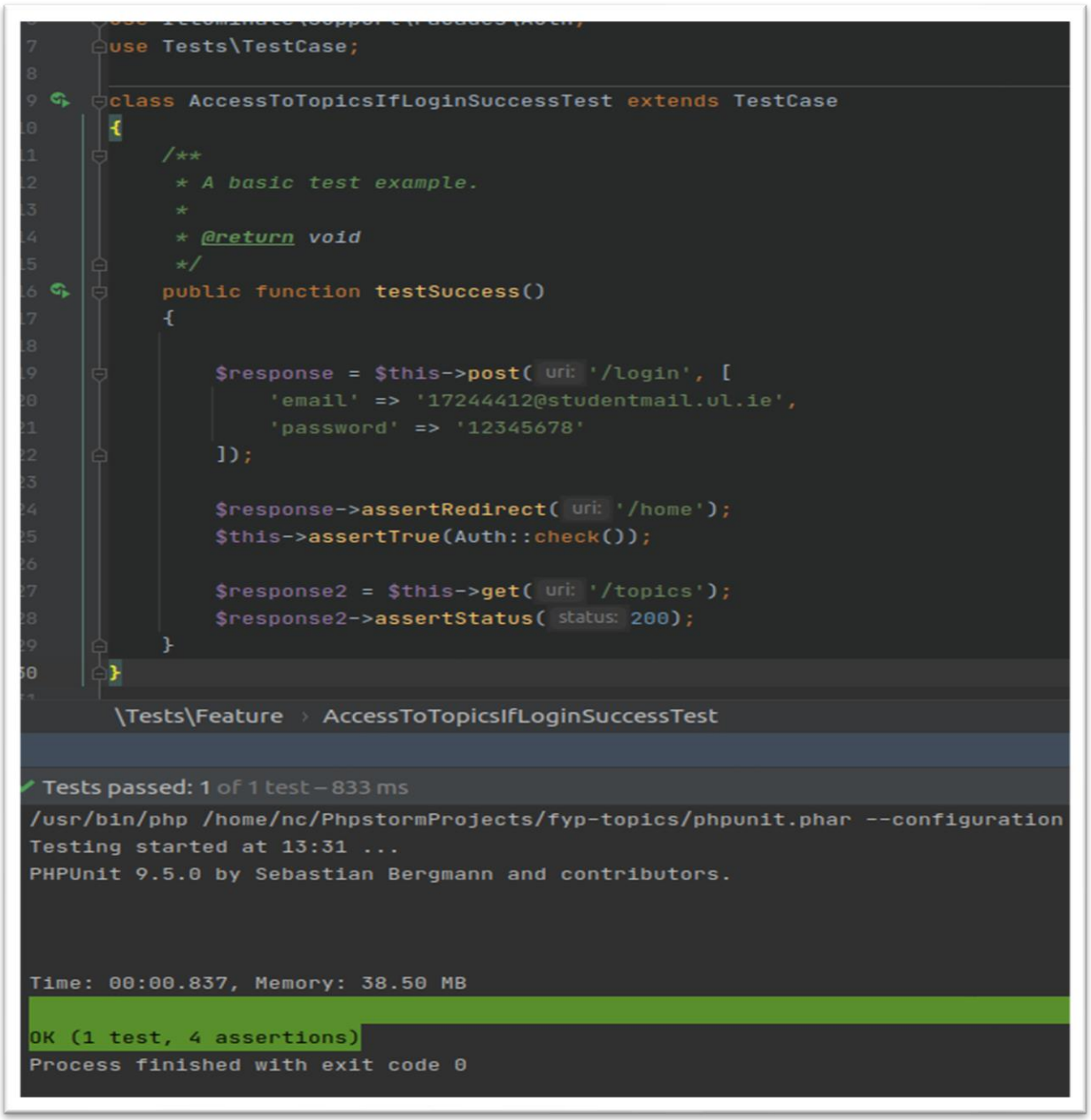
The third test checks the correctness of data entry when registering a new user. From time to time, the user may bypass the website terms of the user acceptance process, and the purpose of this test is to demonstrate this. The registration process requires entering data and agreeing to the Privacy Policy, if the user does not uncheck this box, it is impossible to complete the registration. assertEquals () function returns "True", in this case, it is "302" if the user does not tick Privacy Policy checkbox.



Test 4

Access to Topics Tab by Logged User

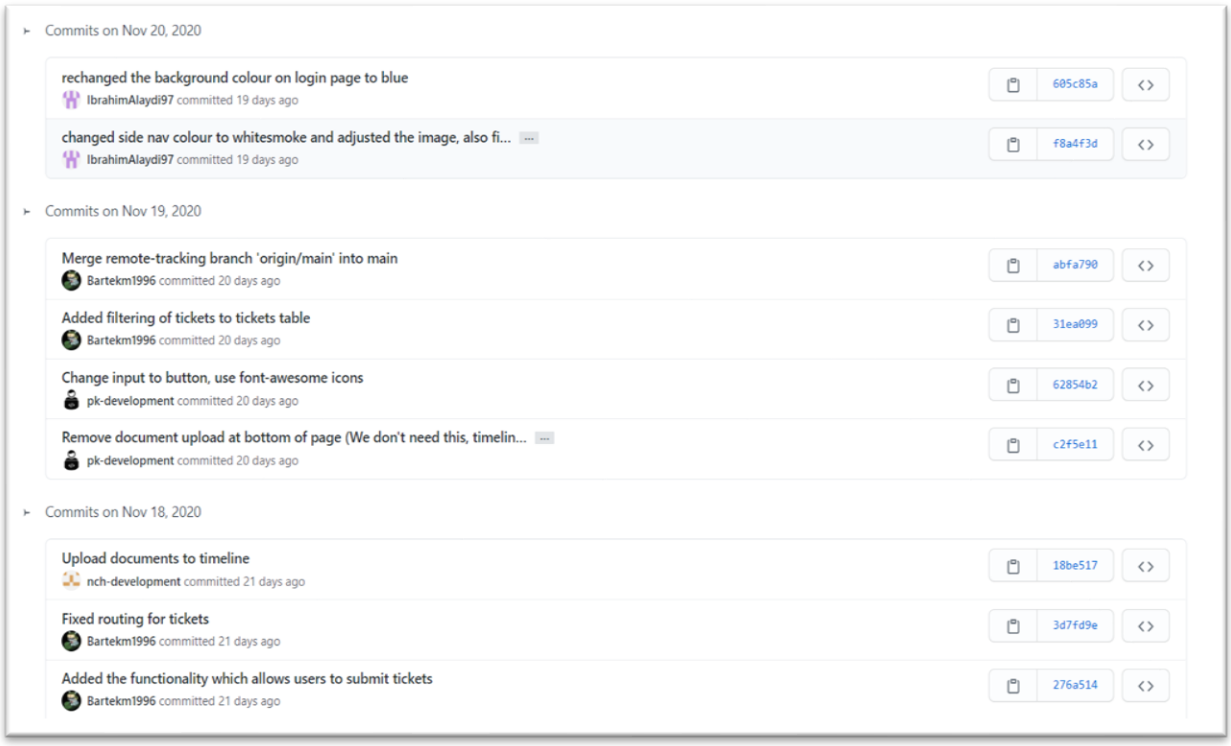
The last test is to check the correct navigation to the Topics tab after the user has logged correctly. The post () function enters the test data, contains whether the user data authorised with Auth::check () function. If the data is correct, the test navigates to the Topics tab, then if the user moved there, returning "True" in a valid test. In case of invalid navigation to the Topics tab, the test will return the value "False".



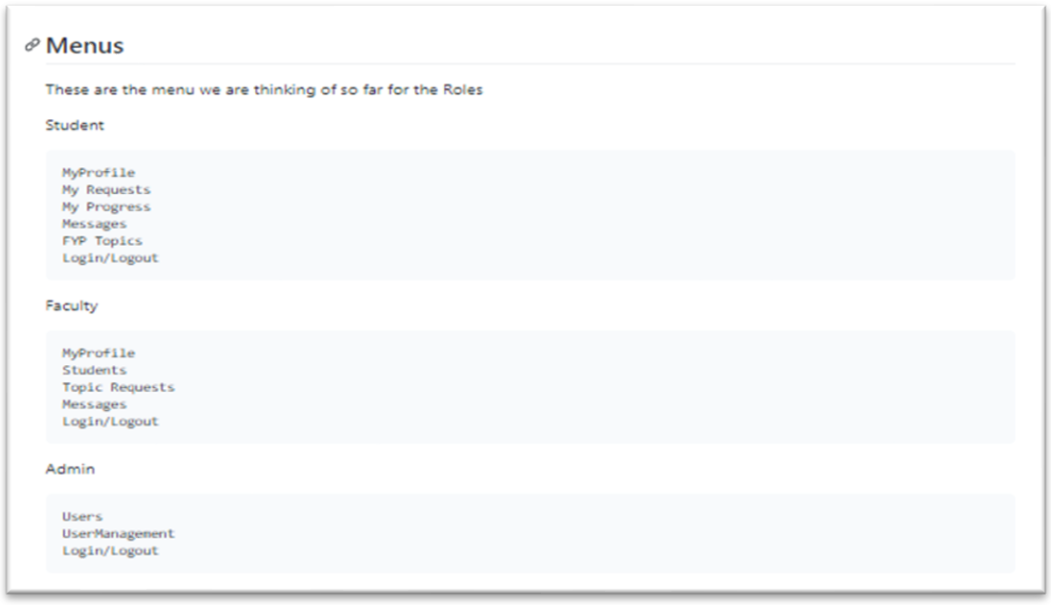
Use of GitHub

URL: https://github.com/Bartekm1996/fyp-topics

The repository is private for the time of the semester however it'll be made public once the semester finished, if it's still private during the review of this document, please email 17241782@studentmail.ul.ie or bmlynarkiewicz1996@gmail.com with the request for it to be made public.



Besides using GitHub for version control, we also used it for scoping our classes and U.I. components



URL to wiki: https://github.com/Bartekm1996/fyp-topics/wiki/Classes-from-prototype

Classes

Profile (Links to a User)

```
-user_id int FK
-address varchar(250)
-name varchar(100)
-qca float
-course varchar(100)
-profile_imgb4 (base64 string)
```

User

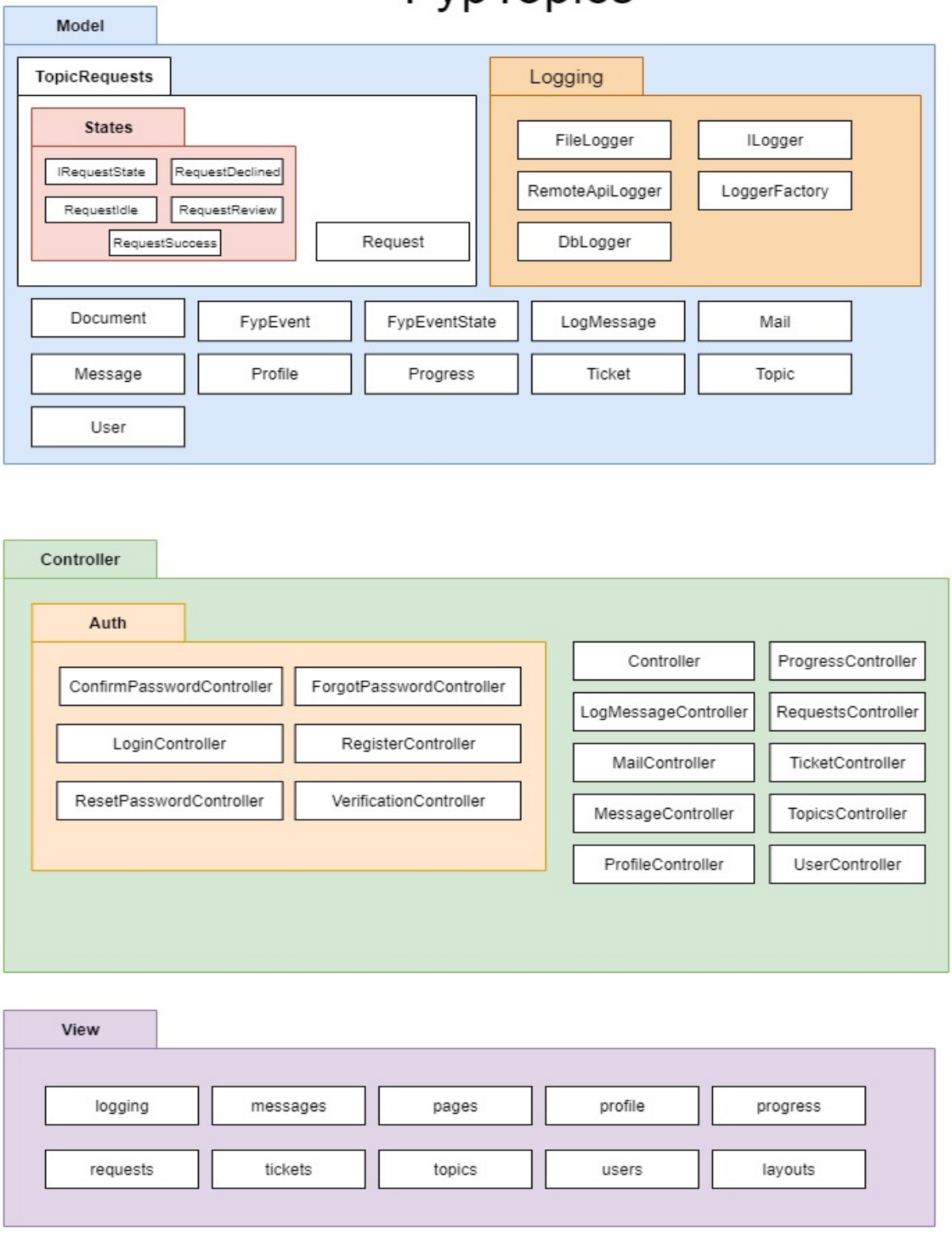
```
-Student

-student_id int (unique)
-email varchar(250)
-first_seen timestamp
-last_seen timestamp
-is_blocked boolean
-is_verified boolean
-role_id int (0:student, 1:faculty,2:admin)
```

Message

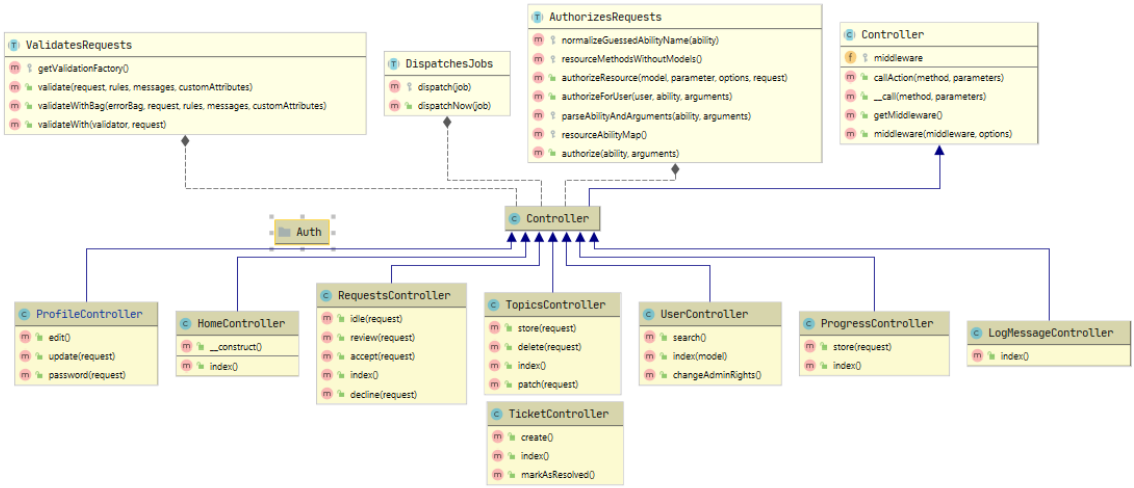
```
-user_id int FK (User who sent message)
-reciever_id int FK (Reciever who the message was sent to)
-timestamp timestamp
-title varchar(200)
-body TEXT
-request_id FK(all the messages should be linked to a Request)
```

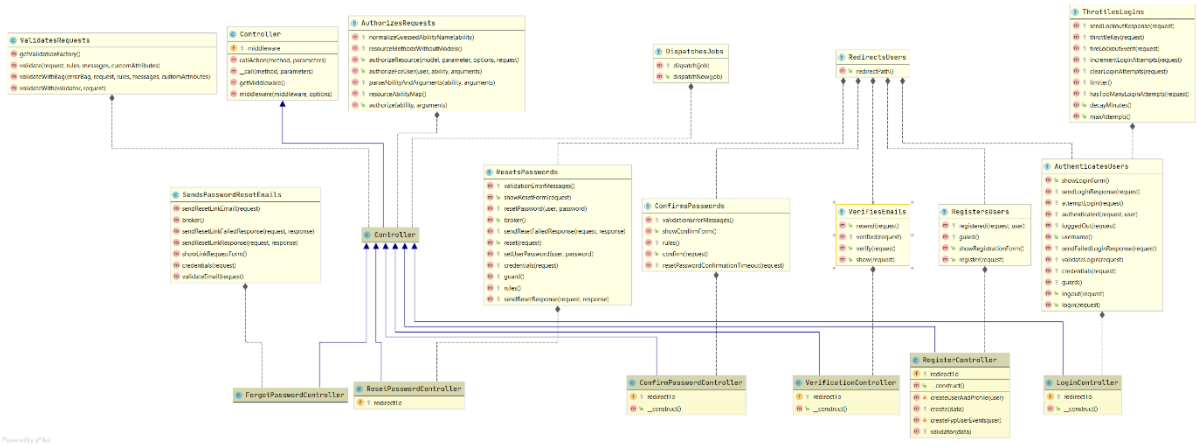
Model View Controller

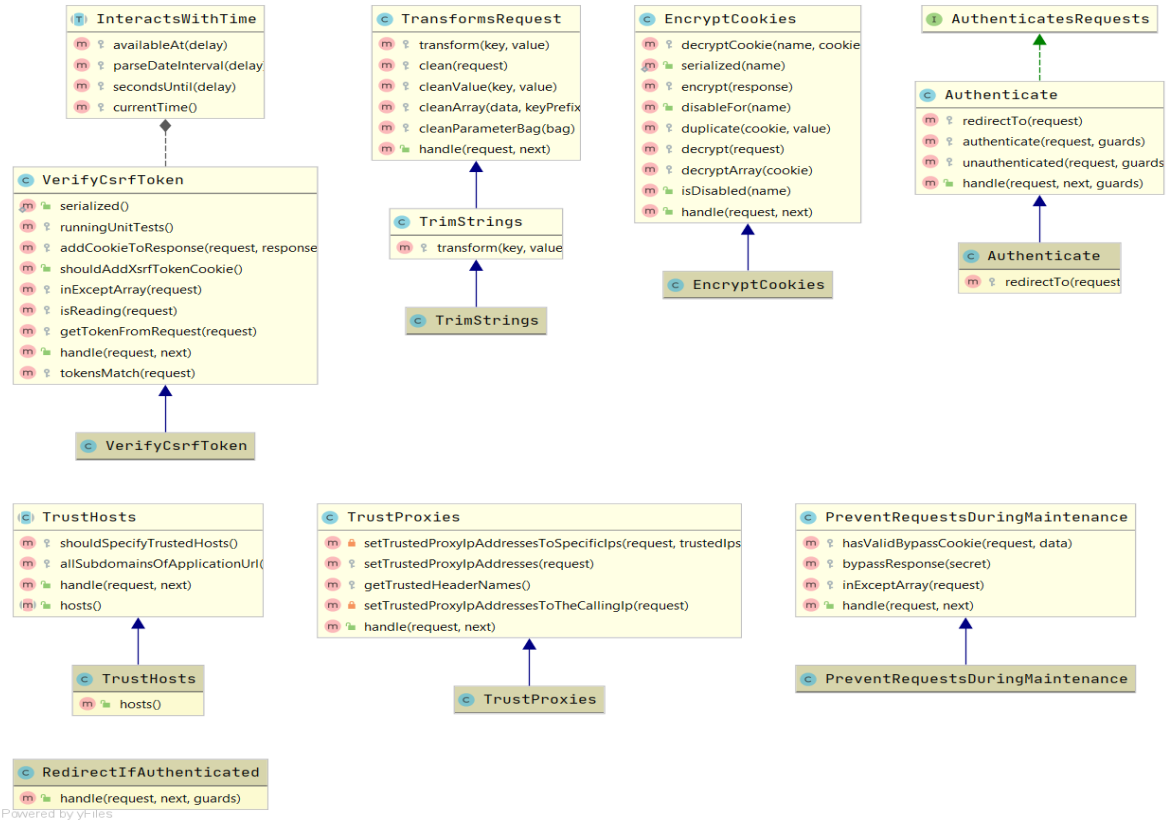
FypTopics

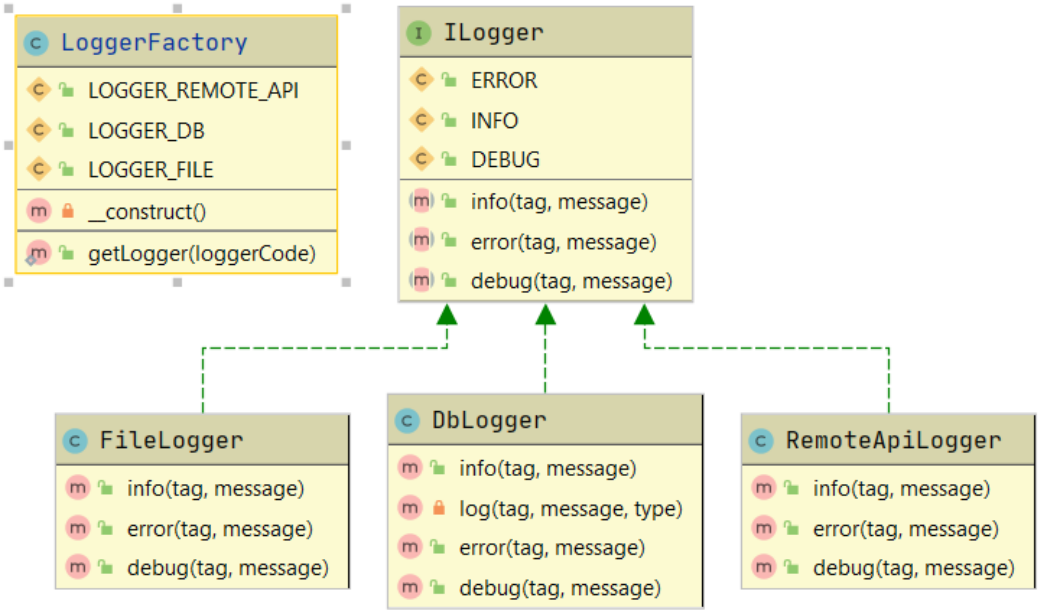


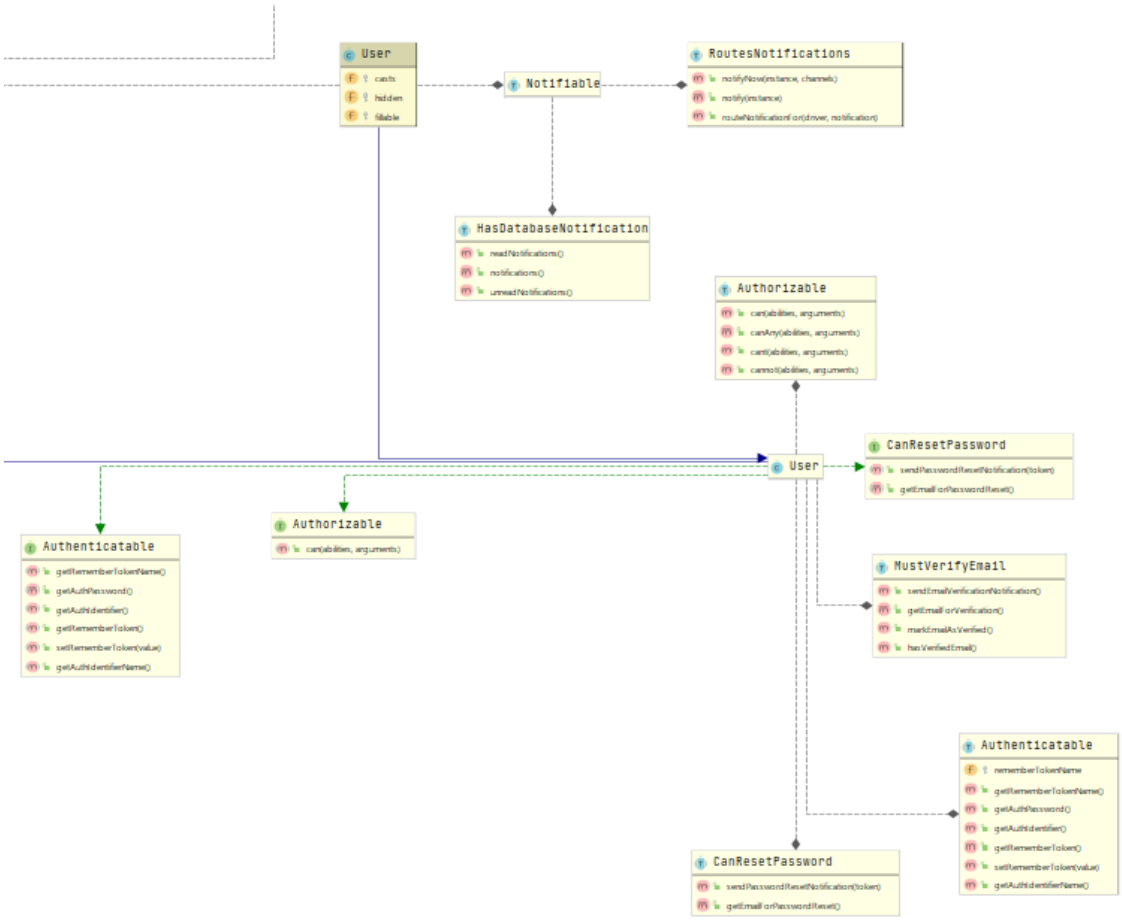
Recovered Class Diagrams

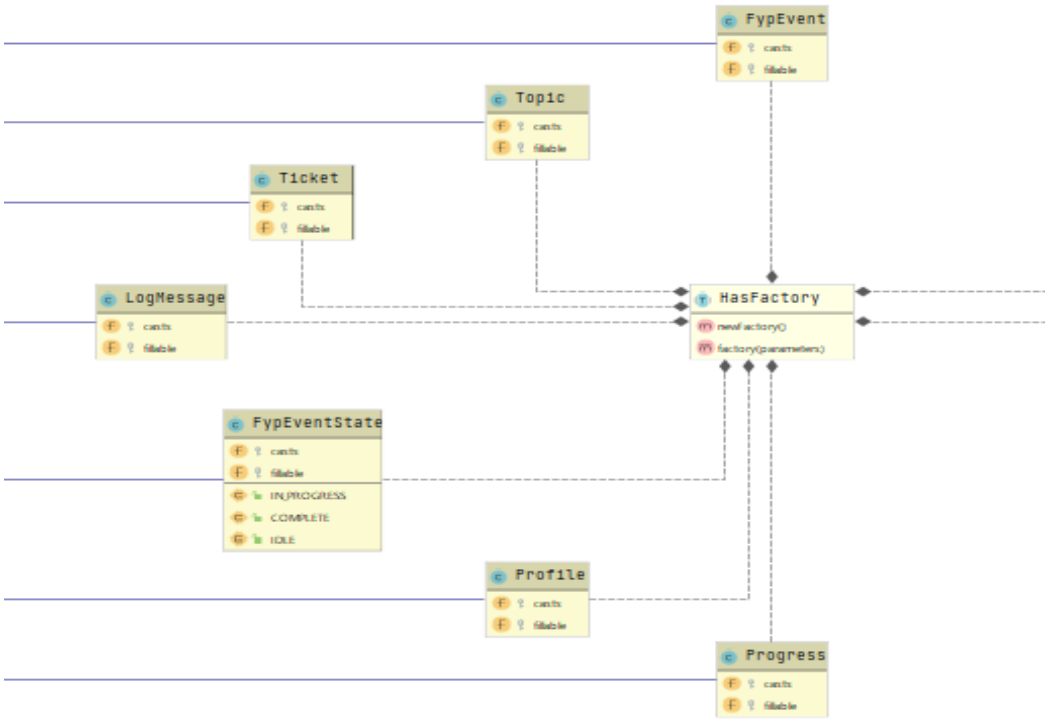


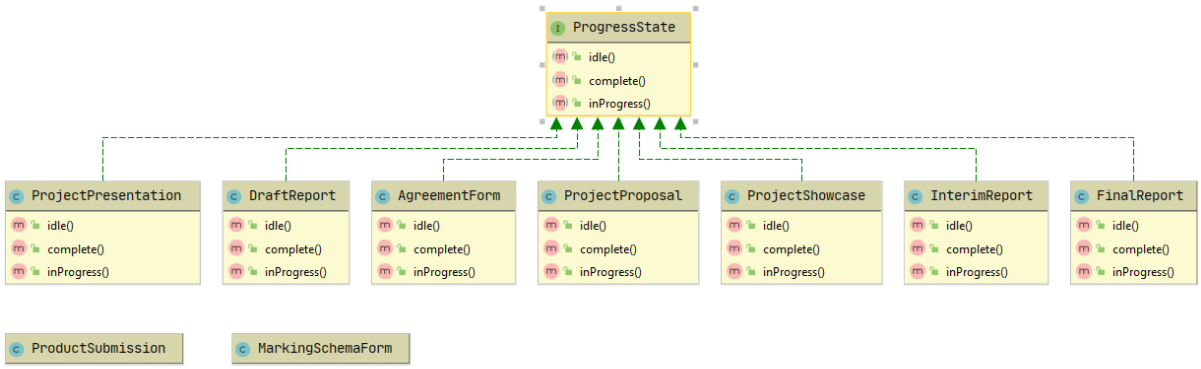


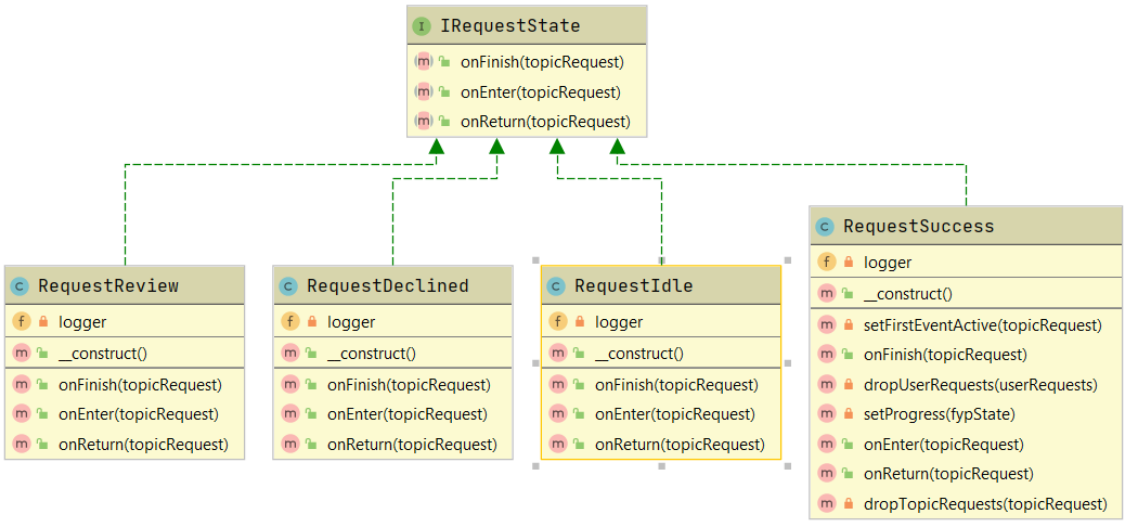

InteractsWithTime
availableAt(delay)
parseDateInterval(delay
secondsUntil(delay)
currentTime()
VerifyCsrfToken
serialized()
runningUnitTests()
addCookieToResponse(request, response
shouldAddXsrfTokenCookie()
inExceptArray(request)
isReading(request)
getTokenFromRequest(request)
handle(request, next)
tokensMatch(request)
VerifyCsrfToken
TransformsRequest
transform(key, value)
clean(request)
cleanValue(key, value)
cleanArray(data, keyPrefix
cleanParameterBag(bag)
handle(request, next)
TrimStrings
transform(key, value
TrimStrings
EncryptCookies
decryptCookie(name, cookie
serialized(name)
encrypt(response)
disableFor(name)
duplicate(cookie, value)
decrypt(request)
decryptArray(cookie)
isDisabled(name)
handle(request, next)
EncryptCookies
AuthenticatesRequests
Authenticate
redirectTo(request)
authenticate(request, guards)
unauthenticated(request, guards
handle(request, next, guards)
Authenticate
redirectTo(request
TrustHosts
shouldSpecifyTrustedHosts()
allSubdomainsOfApplicationUrl(
handle(request, next)
hosts()
TrustHosts
hosts()
TrustProxies
setTrustedProxyIpAddressesToSpecificIps(request, trustedIps
setTrustedProxyIpAddresses(request)
getTrustedHeaderNames()
setTrustedProxyIpAddressesToTheCallingIp(request)
handle(request, next)
TrustProxies
PreventRequestsDuringMaintenance
hasValidBypassCookie(request, data)
bypassResponse(secret)
inExceptArray(request)
handle(request, next)
PreventRequestsDuringMaintenance
RedirectIfAuthenticated
handle(request, next, guards)
Powered by yFiles


LoggerFactory
LOGGER_REMOTE_API
LOGGER_DB
LOGGER_FILE
__construct()
getLogger(loggerCode)
ILogger
ERROR
INFO
DEBUG
info(tag, message)
error(tag, message)
debug(tag, message)
FileLogger
info(tag, message)
error(tag, message)
debug(tag, message)
DbLogger
info(tag, message)
log(tag, message, type)
error(tag, message)
debug(tag, message)
RemoteApiLogger
info(tag, message)
error(tag, message)
debug(tag, message)


User
Notifiable
RoutesNotifications
HasDatabaseNotification
Authorizable
User
CanResetPassword
Authenticatable
Authorizable
MustVerifyEmail
Authenticatable
CanResetPassword


FypEvent
Topic
Ticket
LogMessage
HasFactory
FypEventState
IN_PROGRESS
COMPLETE
IDLE
Profile
Progress



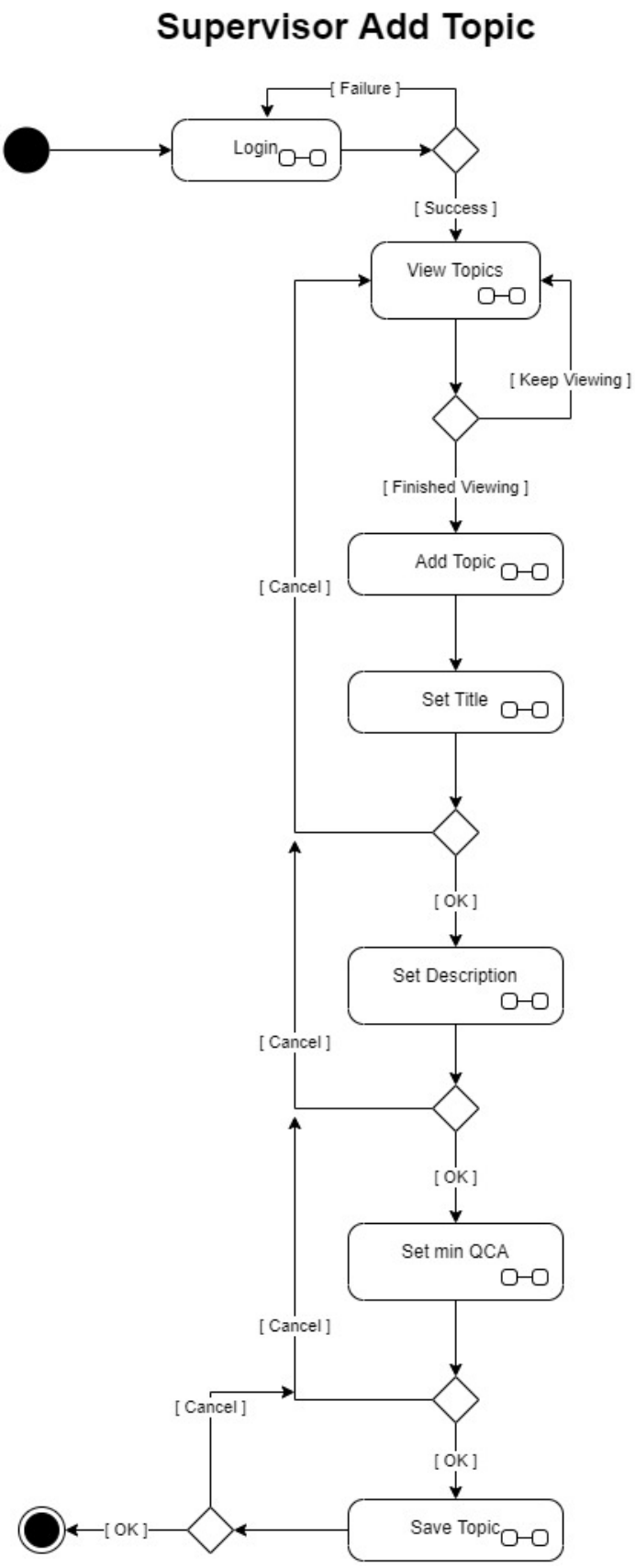


File Structure

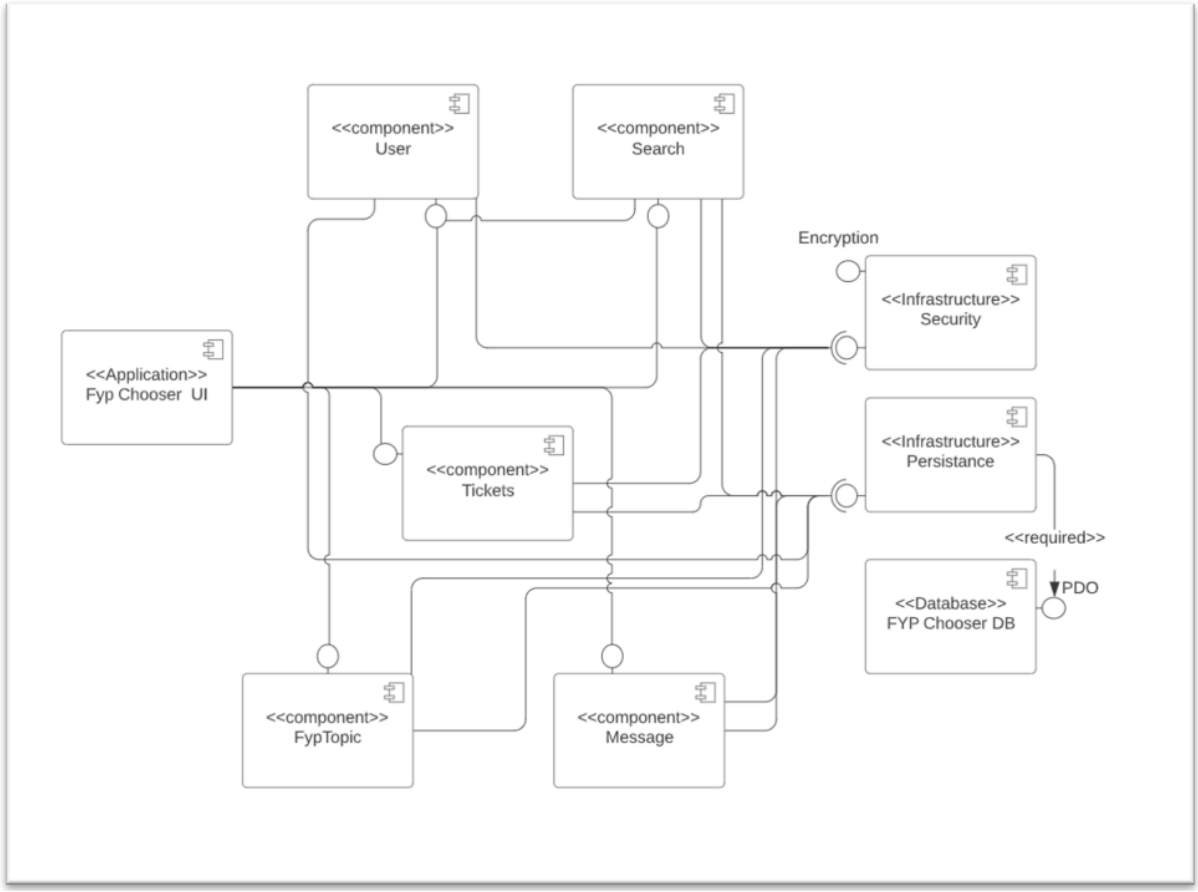
```
Console
    Kernel.php

Exceptions
    Handler.php

Http
    Kernel.php

    Controllers
        Controller.php
        HomeController.php
        LogMessageController.php
        ProfileController.php
        ProgressController.php
        RequestsController.php
        TicketController.php
        TopicsController.php
        UserController.php

        Auth
            ConfirmPasswordController.php
            ForgotPasswordController.php
            LoginController.php
            RegisterController.php
            ResetPasswordController.php
            VerificationController.php

    Middleware
        Authenticate.php
        EncryptCookies.php
        PreventRequestsDuringMaintenance.php
        RedirectIfAuthenticated.php
        TrimStrings.php
        TrustHosts.php
        TrustProxies.php
        VerifyCsrfToken.php

    Requests
        PasswordRequest.php
        ProfileRequest.php
        UserRequest.php

Logging
    DbLogger.php
    FileLogger.php
    ILogger.php
    LoggerFactory.php
    RemoteApiLogger.php

Models
    Document.php
    FypEvent.php
    FypEventState.php
    LogMessage.php
    Profile.php
    Progress.php
    Ticket.php
    Topic.php
    User.php

Progress
    ProgressState.php

    States
        AgreementForm.php
        DraftReport.php
        FinalReport.php
        InterimReport.php
        MarkingSchemaForm.php
        ProductSubmission.php
        ProjectPresentation.php
        ProjectProposal.php
        ProjectShowcase.php

Providers
    AppServiceProvider.php
    AuthServiceProvider.php
    BroadcastServiceProvider.php
    EventServiceProvider.php
    RouteServiceProvider.php

Rules
    CurrentPasswordCheckRule.php
```

```
TopicRequests
    Request.php

    States
        IRequestState.php
        RequestDeclined.php
        RequestIdle.php
        RequestReview.php
        RequestSuccess.php
```

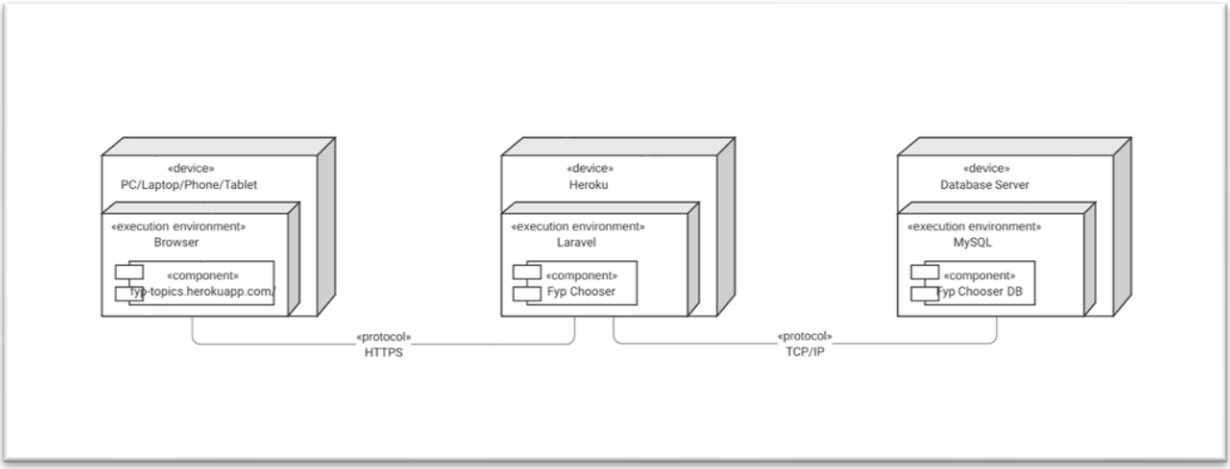
State Chart


Supervisor Add Topic
[Failure]
Login
[Success]
View Topics
[Keep Viewing]
[Finished Viewing]
Add Topic
[Cancel]
Set Title
[OK]
Set Description
[Cancel]
[OK]
Set min QCA
[Cancel]
[Cancel]
[OK]
Save Topic
[OK]

Component diagram

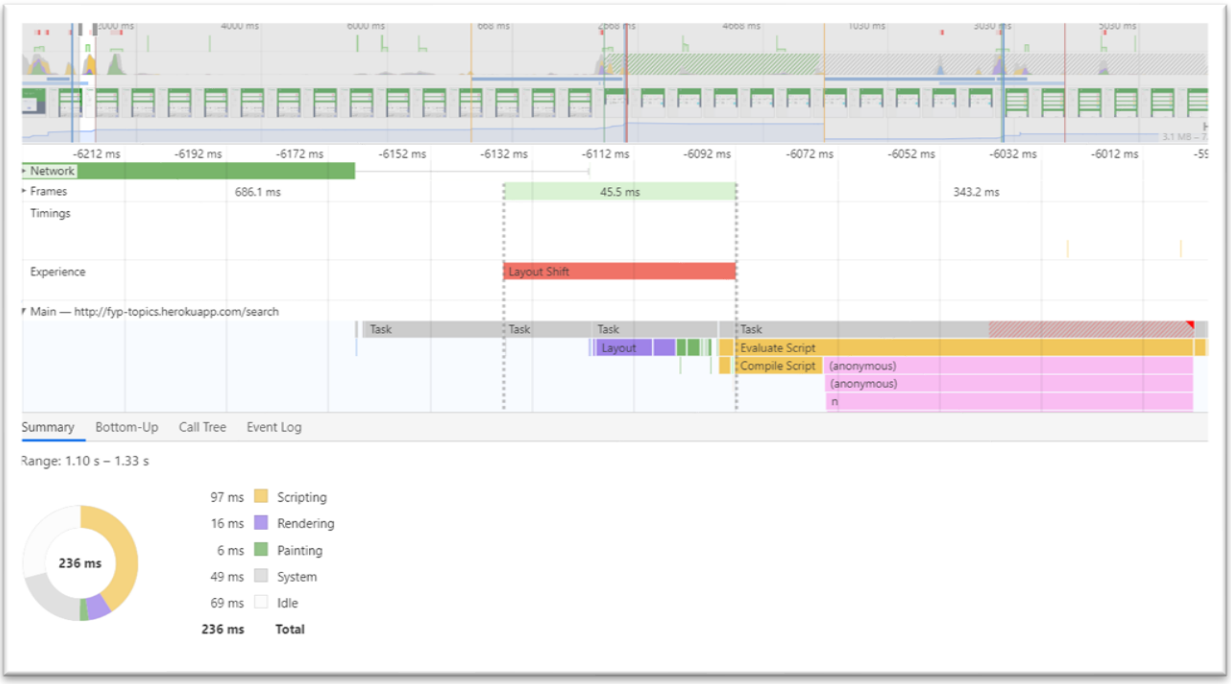


Deployment diagram



Performance

We used Google's chrome dev tools to do site performance. We recorded the user moving around the website and flicking through sections, and this created a timeline of those actions. Over-all the Ms was relatively low which would comply with the non-functional requirements requested by the client



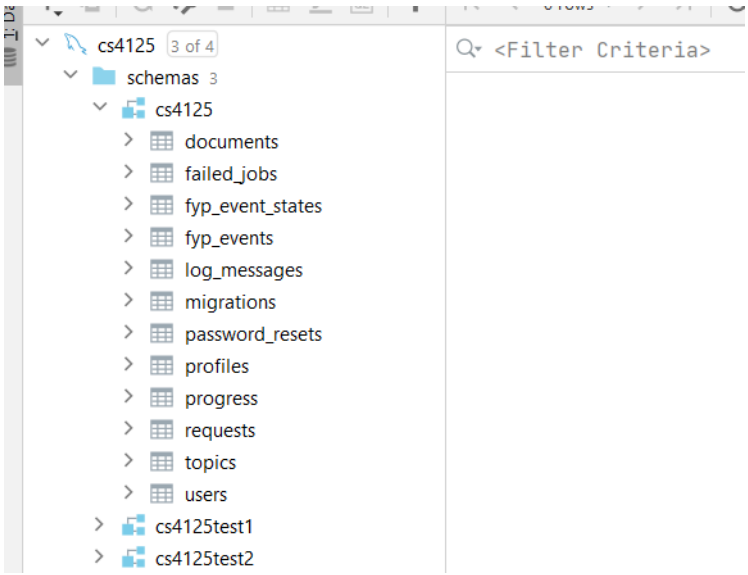
Added Value

Draw.io / Lucidchart.com

A lot of the diagrams were created with draw.io or lucidchart.com rather than using software to extract them. Although we have used software extracted diagrams, we found the manual diagram gave a better feel for the architecture.

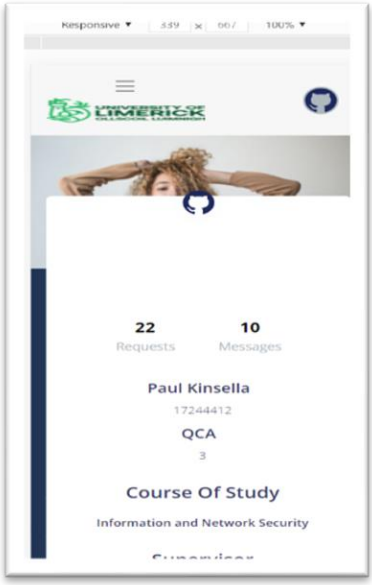
Remote Database

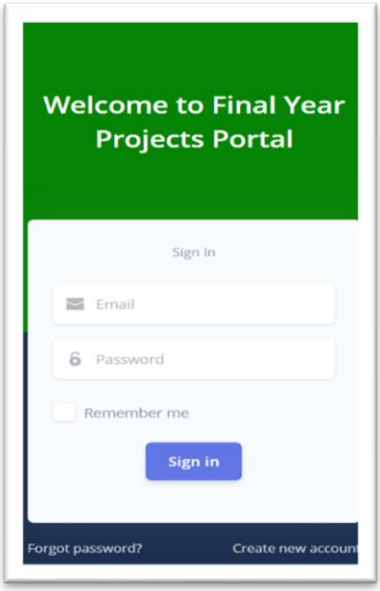
Having test databases is a common practice in the industry. For our project, we have used one primary and two dev databases that were hosted on a remote VPS. We limited the connections to I.P. address we trusted to avoid brute force logging attempts

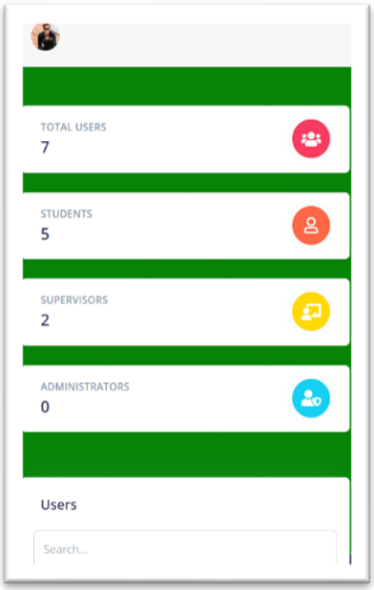


Mobile-Friendly Site

Our website has been designed with mobility in mind to maximise our reach. Having a mobile is crucial nowadays to maximise coverage of the user database.

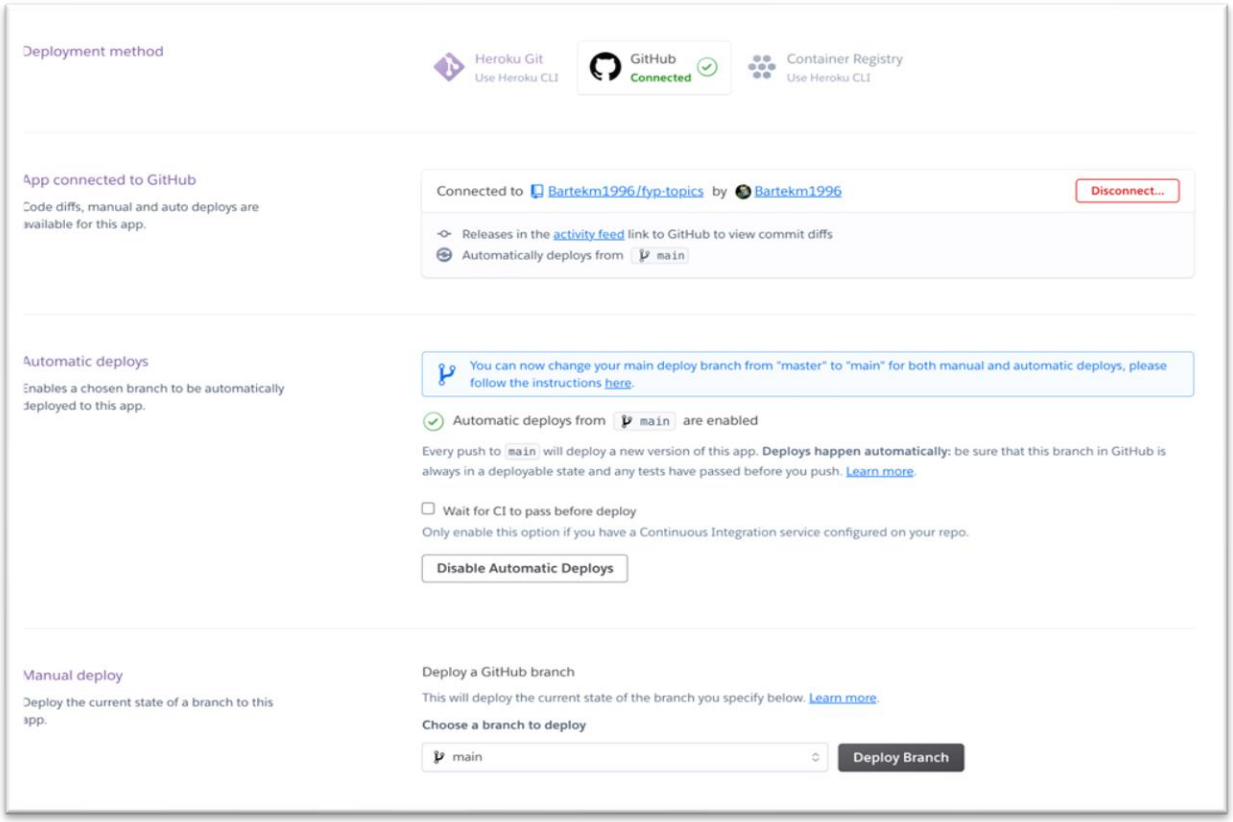




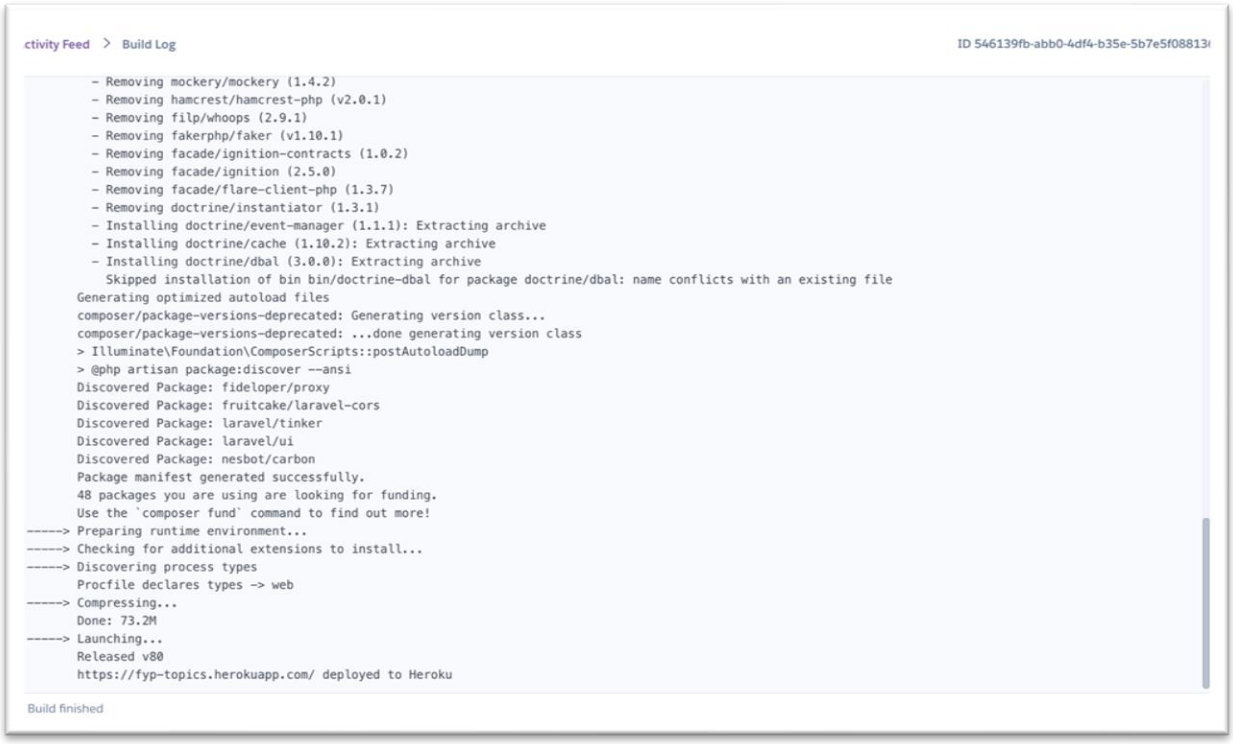


Continuous Integration (CI) / Continuous Deployment (CD)

Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project as Continuous Deployment (CD) is a software release process that uses automated testing to validate if changes to a codebase are correct and stable for immediate autonomous deployment to a production environment (What is Continuous Integration | Atlassian, 2020).



This added value to our project as each commit wash pushed to GitHub, Heroku automatically detected it and ran each commit against several tests before deployment. We were also notified of failed deployments by email, which removed the need of checking if each deployment was successful manually by logging into Heroku.



Using Heroku also gives the advantages of a free SSL certificate unless own domain is used then that benefit is lost with a free tier.

SSL Certificate

The benefit of having a secure website is it adds a layer of security and overall increases the trust rating for the website. The main advantages of using SSL are as follows:

1. Encryption

   One of the greatest benefits of HTTPS is data encryption. The data that travels through an HTTPS always gets encrypted and therefore has an increased layer of protection.

2. Protection

   Data is not stored on the client-side unlike with HTTP, and therefore it's not exposed to the risk of theft.

3. Verification

   Always a certificate makes sure that their policies and the website's policies are the same. If not, the users will get a notification that it is an unsecured connection (Roomi and Roomi, 2020).

4. Data Validation

   HTTPS does the process of data validation through handshaking. All the data transfers that are taking place, and components such as sender and receiver are validated (Roomi and Roomi, 2020).

5. Reliability

   The green padlock that appears on the webpage URL always gives a sense of trust to the visitors that the site is security conscious (Roomi and Roomi, 2020).

6. Search Engine Optimisation

   Having an HTTPS connection is one of the ranking signals to Google. A site that contains an HTTPS certificate will rank higher than the site that doesn't (Roomi and Roomi, 2020).

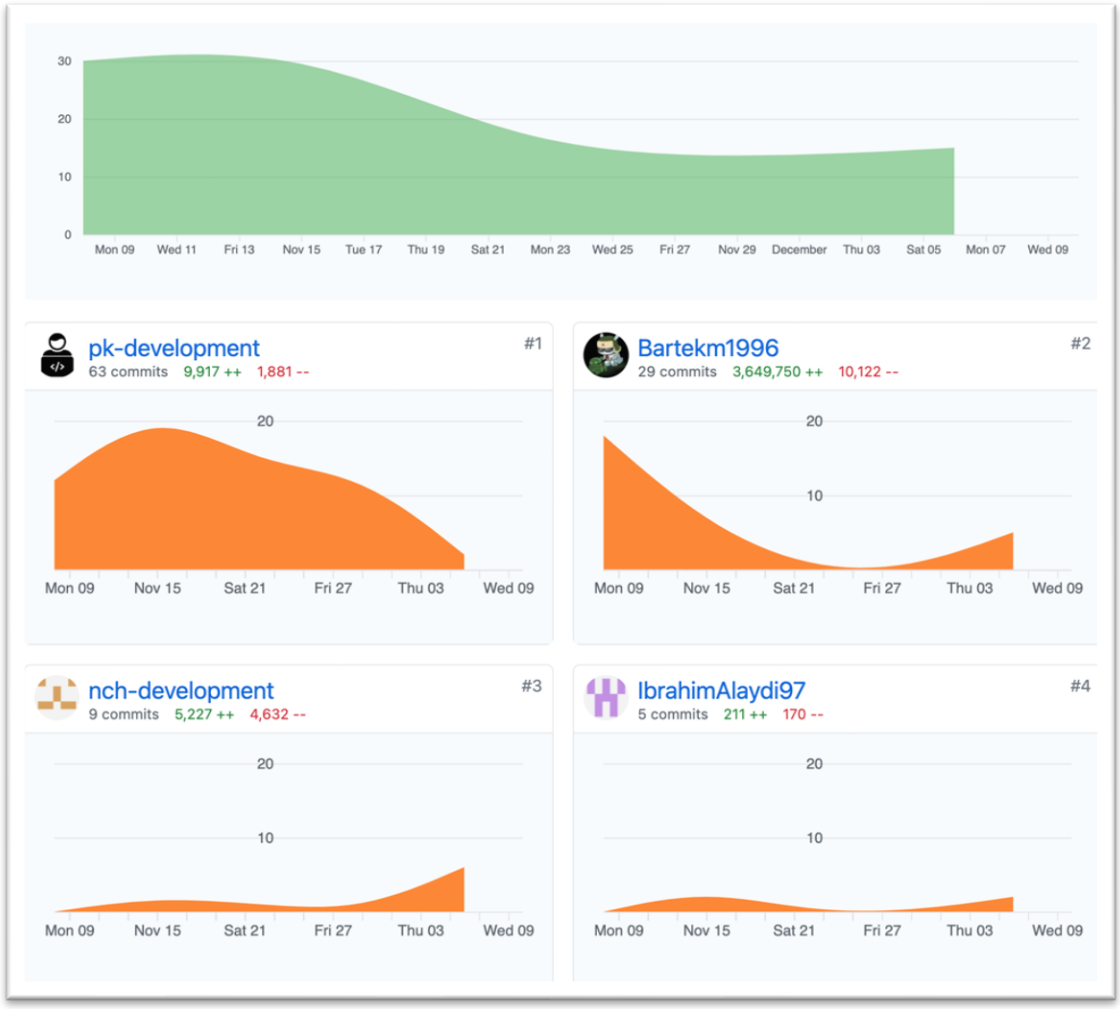
Critiques

Initially, we intended to develop an Android App due to our strong backgrounds in both Java and Kotlin. However, after our initial meeting with our lab assistant, it was decided to settle with an MVC framework. Finally, it chose Laravel due to everyone's previous experience with PHP, which was also covered in the 2nd semester of 3rd year. It quickly came to our attention that implementing design patterns in a framework may not be as optimal and comfortable as it would have been in Android or without using a framework, therefore if we were to redo this project, we would either do it without a framework or else in Android. Some of these design patterns fit other types of systems better; for example, the decorator pattern would be more suited to an application like a pizza store. We did want to use the decorator pattern for generating a certificate when the student finished the course, but this was out of scope.

Although we started on this project early, we have switched project ideas quite a lot rather than settling on one. This delayed us in the final stages of this project. We had to decide to stop development and leave out some features rather than run the risk of an incomplete document. We are happy enough with the prototype and implemented a lot of what was requested in the requirements.

We took a gamble on this using PHP rather than a language we knew, and we looked at it as a learning experience but in hindsight that was a silly choice.

Code Contributions



Between the four collaborators, the work was equally shared out between the coding and the document, those with weaker skills in development did a greater amount in the document. Bartekm1996's input may seem like extraneous however please note that he set up the GitHub repo with the initial commit and a significant proportion of those additions amount to Laravel's source code. Everyone worked extremely hard on this project, and there was a large amount of pressure on all of us due to external factors such as COVID and having to get tested on quite a few occasions.

Sample Classes

User Class

```
<rpnp

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
}
```

Progress Class

```
<rpnp

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

/**
 * Class Progess
 * @package App\Models
 * This holds the state the current user is at
 */
class Progress extends Model
{
    use HasFactory;

    protected $fillable = [
        'user_id',
        'fypevent_state_id'
    ];

    protected $casts = [
        'created_at' => 'datetime',
        'lastupdated' => 'datetime',
    ];
}
```

Document Class

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Document extends Model
{
    use HasFactory;

    protected $fillable = [
        'filename',
        'data',
        'fyp_es_id',
        'user_id',
    ];

    protected $casts = [
        'updated_at' => 'datetime',
    ];
    /**
     * @var int|mixed
     */
}
```

Profile Class

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Profile extends Model
{
    use HasFactory;

    protected $fillable = [
        'student_id',
        'course',
        'image',
        'qca',
    ];

    protected $casts = [
        'updated_at' => 'datetime',
    ];
}
```

References

Amazon Web Services. 2020. Continuous Delivery Using Spinnaker On Amazon EKS | Amazon Web Services. [online] Available at: <https://aws.amazon.com/blogs/opensource/continuous-delivery-spinnaker-amazon-eks/> [Accessed 11 December 2020].

Atlassian. 2020. What Is Continuous Integration | Atlassian. [online] Available at: <https://www.atlassian.com/continuous-delivery/continuous-integration> [Accessed 11 December 2020].

[1] Belatrixsf , Benefits and Disadvantages of Scrum Methodology in Software Development (2013), available: https://www.belatrixsf.com/blog/benefits-scrum-software-development [Accessed 31 October 2020]

En.wikipedia.org. 2020. Phpunit. [online] Available at: <https://en.wikipedia.org/wiki/PHPUnit#:~:text=PHPUnit%20is%20a%20unit%20testing,development%20is%20hosted%20on%20GitHub.> [Accessed 14 November 2020]

Roomi, M. and Roomi, M., 2020. 6 Advantages And Disadvantages Of HTTPS | Drawbacks & Benefits Of HTTPS. [online] HitechWhizz - The Ultimate Tech Experience. Available at: <https://www.hitechwhizz.com/2020/08/6-advantages-and-disadvantages-drawbacks-benefits-of-https.html> [Accessed 11 December 2020].