ExGen - Next Generation Exams

Design documentation

Group 11

Signed by:
Brandon Skerritt
Luke Aldersley
Yales Stéfano Rios Vasconcelos
Steffan Drosinos Jones
Wen Geng Ong
Claire Mulholland

System Specification

What is ExGen?

ExGen is the future of exams and revision as we know it. The United Nations set out sustainable goals[1] to achieve by 2030 to change the world. Goal 4[2] is to create a quality education for everyone of the world. ExGen exists to fulfill this purpose.

We're giving students the ability to practice an infinite amount of times on questions, reinforcing the topics they have learnt all while professors and lecturers only have to write the exam once.

ExGen is a program to procedurally generate exams given an exam. Instead of giving your students one mock paper, give them ExGen which will automatically generate an infinite number of papers for your students to practice on.

Users of the system

There will be 4 users of the system.

1. Student
2. Course representatives (or external verifiers)
3. Professor
4. Administrators

The student will be able to generate tests and answer questions. The professor will be able to upload exams and see a statistical report on the exam. The course representatives (or external verifiers) will have all the same privileges as a normal student but will be able to see the statistics relating to the questions. The administrators are the programmers and designers of the system.

The admins will be able to create professors, professors can create modules and students will need to assign themselves to a module because some modules can have 400+ students. Professors can add other professors to a module they own (for example, one module can have demonstrators who will be classed as professors in the system).

Ideally, we would of liked to have automated this system using Liverpool Life. Our system is not designed specifically for the University of Liverpool but is instead designed as open source software anyone can implement and use. Implementing Liverpool Life may slow us down or force us to create workarounds for things that don't exist in other university systems.

[1] https://www.un.org/sustainabledevelopment/sustainable-development-goals/
[2] https://www.un.org/sustainabledevelopment/education/

Modules

Each professor will be able to create modules. Each student can 'opt-in' to the modules they have taken. The process of students and lecturers being assigned to modules can be automated in future releases through Liverpool Life or Vital.

Security and Business Rules.

Because the program runs arbitrary code written by the professor, security is one of the most important issues we face. Firstly, only certified professors can execute code on the system. But, that doesn't mean that the professor might write code which sometimes can enter into infinite loops. Enter sandboxing.

Sandboxing is a technique by which you execute code in a box, so the code can only cause problems inside that box. We will have a ~~robot slave~~ service worker which constantly monitors the resources and state of the program inside this box. If the program takes too long, demands too many resources or does something that looks suspicious the service worker will terminate the box and inform the admins that something has gone wrong.

By sandboxing the code the only things affected is the question for which that code relates to. In this instance, the question would not appear to the users. Only the professor or administrators will be able to see that a problem has occured. To the users, nothing bad will happen. No question will appear. They will not see a decrease in performance.

Network and access requirements

All administrators of the system will have SSH access to the system where ExGen is hosted. The SSH key (and subsequently all passwords, other than student / professor passwords) will need to follow the following format:

- 12 characters
- No words
- None of the top 5000 passwords
- Contains at least 1 special character
- Contains at least 1 number
- Contains at least 1 English letter

The number range is to make brute forcing the passwords harder. For non-administrators all passwords will follow this format:

- Be at least 8 characters long
- Not in the top 5000 passwords
- Not a single word

- Not a sequence
- Contains at least 1 special character
- Contains at least 1 number
- Contains at least one English letter

All passwords will be hashed client-side, so as not to send unsecured passwords through the network. If this isn't possible or out of scope, all passwords will be sent through SSL to secure the payload and will be hashed as soon as they reach the server.

To make sure we don't store the plain password, the hashing program will be a functional implementation of a caching program. In functional programming, you don't store things in variables. By not storing it in variables, we limit the amount of times the password occurs naturally in the program.

Backup and recovery

The backup procedure for the system will be split up into two ways. Firstly, Digital Ocean offers a backup solution on their own servers in Amsterdam. This will be the first backup, as it is physically and logically closet to our server it will be the fastest way to restore the system.

The second backup would be made by system administrators to another hard drive physically and logically far away from the datacenter in Amsterdam. This is because if an earthquake or some freak force of nature wipes out the datacenter in Amsterdam, it will not affect the United Kingdom. It is important to note that if you implement this system locally (IE not in Digital Ocean) or you live in Amsterdam it is important to backup away from the local hosting server.

Legal issues

The exams in our system will be treated exactly as how exams are normally treated. All content is owned either by the professor wholly or partly, or by the department wholly or partly. It changes based on the department and university.

For the University of Liverpool[3] it is complicated and differs on departmental basis. I believe, by looking at the IP laws of the University of Liverpool, they own 100% of any IP created during work time - including exams. Under exceptional circumstances can a professor request an IP to be transferred to them. If this is the case, it will no longer be a legal issue we'll deal with but instead will be a legal issue dealt with between the professor and the university.

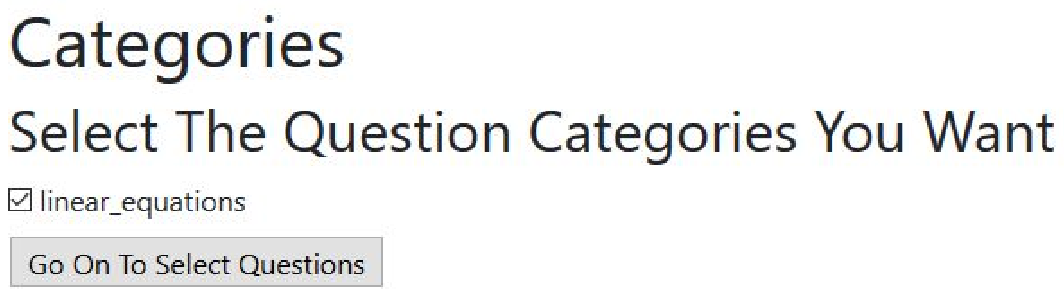
Existing Algorithms

In our requirements review, we mentioned that Khan Academy uses a similar algorithm to generate numbers for questions. Surprisingly, this isn't true. Khan Academy relies on a

[3] https://www.liverpool.ac.uk/intellectual-property/

plethora of volunteers to hand-write the numbers and answers for them, according to Khan Academy[4] themselves.

Someone else had the exact same idea as us, MathTestMaker[5]. Their program let's people choose from a range of questions:



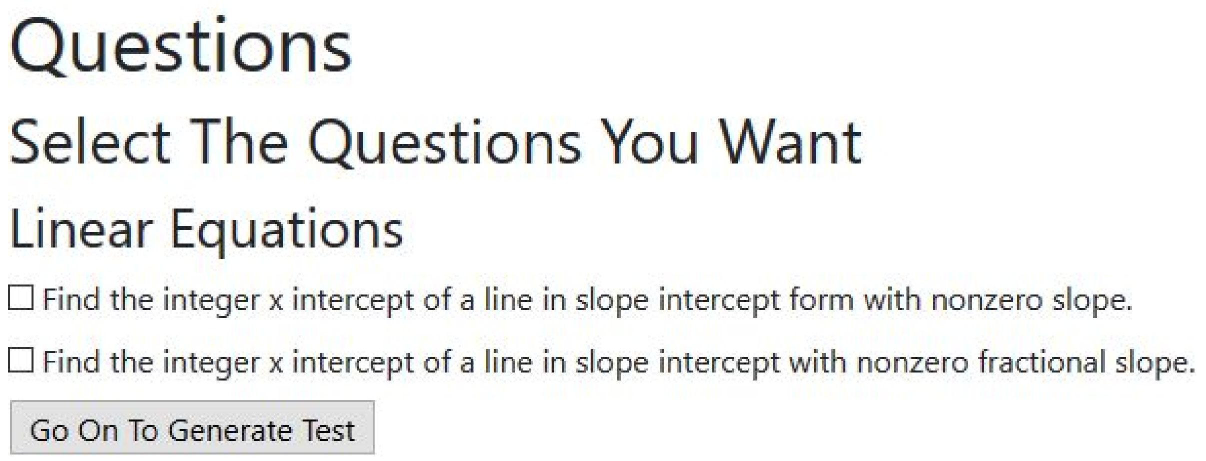
Categories

Select The Question Categories You Want

☑ linear_equations

Go On To Select Questions

And then you select the format of the question:



Questions

Select The Questions You Want

Linear Equations

☐ Find the integer x intercept of a line in slope intercept form with nonzero slope.

☐ Find the integer x intercept of a line in slope intercept with nonzero fractional slope.

Go On To Generate Test

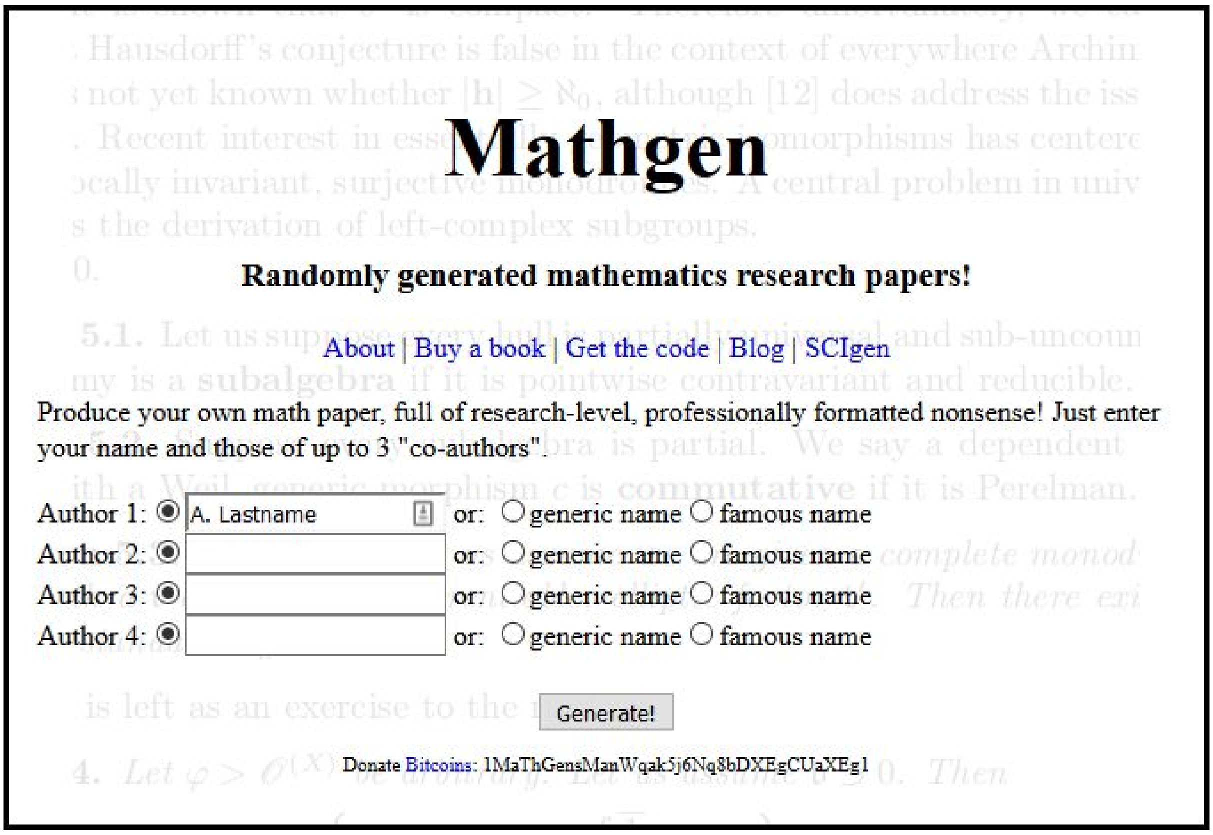
And then you get this messy output:

> *"{'problemStatement': 'Find the x intercept of $y = -5x + -25$.', 'correctAnswer': -5, 'correctAnswerIdx': 3, 'wrongAnswers': [-4, -4, -4, -4], 'points': 1, 'solution': [('y = -5x + -25', 'Find the x intercept.'), ('0 = -5x + -25', 'Set y to zero.'), ('0 - -25 = -5x + -25 - -25', 'Subtract b from both sides.'), ('25 = -5x', 'Simplify.'), ('25 / -5 = x', 'Divide both sides by m.'), ('-5', 'Simplify.')]} "*

While our ideas are similar, their code requires the programmer to have preemptively developed the questions and the format of the questions. This is very time consuming for the programmer. This project has existed for 2 years and it only has 1 type of question. It is not possible to add your own questions and add your own code from their interface. ExGen is designed to give freedom to the professor, not to take it away. While we are toying with the idea of having a similar solution where professors can choose questions which have already been made, it will not be the only thing the system will do. While the competitor MathTestMaker doesn't let professors write their own questions or code, ExGen will.

[4] https://www.khanacademy.org/about/blog/post/57701417065/100-000-practice-problems
[5] https://mathtestmaker.com

There is another competitor, MathGen, although they generate maths papers as jokes, and they're academic maths papers.


Mathgen

Randomly generated mathematics research papers!

About | Buy a book | Get the code | Blog | SCIgen

Produce your own math paper, full of research-level, professionally formatted nonsense! Just enter your name and those of up to 3 "co-authors".

Author 1: ● A. Lastname or: ○ generic name ○ famous name
Author 2: ● or: ○ generic name ○ famous name
Author 3: ● or: ○ generic name ○ famous name
Author 4: ● or: ○ generic name ○ famous name

Donate Bitcoins: 1MaThGensManWqak5j6Nq8bDXEgCUaXEg1

Risk assessment

Risks (for the system to consider)	Likelihood	Costs	Mitigation/response
Server failure	Low	Low to moderate - In the case of a server failure the users would not be able to access any of the questions stored on the system. Additionally some question, user or results data may be lost/corrupted by a server failure.	The server is maintained by the university using it so the mitigation and response/recovery strategy will ultimately be up to them. Most likely the university should make regular backup copies of important information. This would minimise data loss and should take no more than a few days to restore a backup copy minimising downtime.
Security breach	Low	High - in the case of a security breach user information may be compromised and leaked.	Since the user information should be tied to the universities own administration system the

		The information in our system may not be very in extensive but information such as usernames and passwords can be used to obtain more serious information about the user like emails and card information.	information should be relatively secure and the university should have a planned response in the event of a data breach.
Misuse/cheating by the user (students)	Low	Low to moderate - if there is a case of users misusing or cheating the system there is little damage to the university since even if students get high marks they are meant for revision and should not be used for examinations.	If misuse/cheating is found to be happening the the university/professor can easily change the questions to new ones.

Risks (for the project)	Likelihood	Cost	Mitigation/response
Running out of money for support costs of our server.	Low	Moderate - if we run out of money to pay supporting costs for our host server we will lose access to it and will be unable to continue development, or demonstrate a working prototype.	We have allocated enough money to host a server until the start of the next academic year so there should no chance of running out of money. In the unlikely event that we do run out of money we have sufficient personal funds to uses as a backup.
Server failure	Low	High - if the server we are developing on were to fail we would be unable to continue some development, there may be data loss meaning we have to remake some parts and may be unable to demonstrate a working prototype.	We have chosen Digital Ocean as the host for our server. They have 99.99% uptime on their servers so we believe that this is enough to minimise the chance of the server failure. We will also keep a backup of all the information we store on the server to minimise data loss.
Failure to complete the tasks to deadlines	Low	moderate - if we fail to meet deadlines for tasks we may have to delay dependant tasks causing a chain effect that may delay the hole project and miss the project deadline entirely, if the task is on the critical path.	We have believe we have allocated sufficient time for each task and allowed for enough spare time before each deadline to take up any delays we may have. We have also sufficiently contained the scope of the project so we don’t have too

			many tasks to complete.

Gantt charts

Requirements tasks

ID	Task	Duration (d)	Start on	End by	Dependencies	Assigned resources
1	Project Description	1	04/02/19	05/02/19	none	{Brandon}
2	Project mission statement	1	04/02/19	05/02/19	none	{Brandon}
3	User - Professor	1	05/02/19	06/02/19	1	{Brandon}
4	User - Student	1	05/02/19	06/02/19	1	{Brandon}
5	User - course rep	1	05/02/19	06/02/19	1	{Brandon}
6	User - Admin	1	05/02/19	06/02/19	1	{Brandon}
7	Questions and statistics	1	06/02/19	07/02/19	6	{Brandon}
8	Modules	1	06/02/19	07/02/19	6	{Brandon}
9	Performance	1	06/02/19	07/02/19	6	{Brandon}
10	Security	1	06/02/19	07/02/19	6	{Brandon}
11	Technology stack	1	06/02/19	07/02/19	6	{Brandon}
12	Network and access requirements	1	07/02/19	08/02/19	11	{Brandon}
13	Backup	1	07/02/19	08/02/19	11	{Brandon}
14	Legal issues	1	07/02/19	08/02/19	11	{Brandon}
15	System boundary description	2	04/02/19	06/02/19	none	{Steffan}
16	System boundary diagram	2	06/02/19	08/02/19	15	{Steffan}
17	User views and requirements	1	04/02/19	05/02/19	none	{Luke}Yales

18	Use cases	1	04/02/19	05/02/19	none	{Luke}Yales
19	Use case diagram	3	05/02/19	08/02/19	18	{Luke}Yales
20	Use case description	1	08/02/19	09/02/19	19	{Luke}Yales
21	Transaction requirements	1	04/02/19	05/02/19	none	{claire}
22	Student transactions	1	05/02/19	06/02/19	21	{claire}
23	Course rep transactions	1	05/02/19	06/02/19	21	{claire}
24	Professor transactions	1	05/02/19	06/02/19	21	{claire}
25	Admin transactions	1	05/02/19	06/02/19	21	{claire}
26	Anticipated Project tasks	1	04/02/19	05/02/19	none	{Geng}
27	Anticipated project gantt chart	1	05/02/19	06/02/19	26	{Geng}
28	Risk assessment	1	06/02/19	07/02/19	none	{Geng}
29	Document review	5	11/02/19	15/02/19	All of the above	{Brandon}Yales, Geng, Steffan, Luke, Claire
30	Document submission	0	15/02/19	15/02/19	none	{Brandon}Yales, Geng, Steffan, Luke, Claire
31	Review presentation	5	18/02/19	22/02/19	none	{Brandon}Yales, Geng, Steffan, Luke, Claire

Design tasks

ID	Task	Duration (d)	Start on	End by	Dependencies	Assigned resources
32	Design summary	1	25/02/19	26/02/19	none	{Brandon}
33	Aims and objectives	1	25/02/19	26/02/19	none	{Brandon}
34	Existing algorithms	1	25/02/19	26/02/19	none	{Brandon}

35	Anticipated Subsystems	3	26/02/19	01/03/19	34	{Brandon}
36	Anticipated Subsystem communication	3	01/03/19	05/03/19	35	{Brandon}
37	Interaction chart	3	01/03/19	05/03/19	35	{Brandon}
38	System boundary diagram	1	25/02/19	26/02/19	none	{Steffan}
39	Design evaluation criterias	3	25/02/19	28/02/19	none	{Steffan}
40	Login page design	2	25/02/19	27/02/19	none	{Luke}Steffan
41	Student page design	3	27/02/19	02/03/19	40	{Luke}Steffan
42	Profesor page design	3	27/02/19	02/03/19	40	{Luke}Steffan
43	Course rep page design	3	27/02/19	02/03/19	40	{Luke}Steffan
44	Question input page design	2	04/03/19	06/03/19	43	{Luke}Steffan
45	Question input algorithm design	5	25/02/19	02/03/19	none	{Yales}Geng
46	Question input algorithm Pseudo code	5	25/02/19	02/03/19	none	{Yales}Geng
47	Question parsing algorithm design	5	25/02/19	02/03/19	none	{Yales}Geng
48	Question parsing algorithm Pseudo code	5	25/02/19	02/03/19	none	{Yales}Geng
49	Question output algorithm design	5	25/02/19	02/03/19	none	{Yales}Geng
50	Question output algorithm Pseudo code	5	25/02/19	02/03/19	none	{Yales}Geng
51	Updated gantt chart	2	25/02/19	28/02/19	none	{Geng}
52	Entity-relationship diagrams	3	25/02/19	01/03/19	none	{Claire}
53	Logical table structures	2	01/03/19	03/03/19	53	{Claire}
54	Use case diagram	1	25/02/19	26/02/19	none	{Luke}
55	Data structure design	5	25/02/19	02/03/19	none	{Luke}Geng

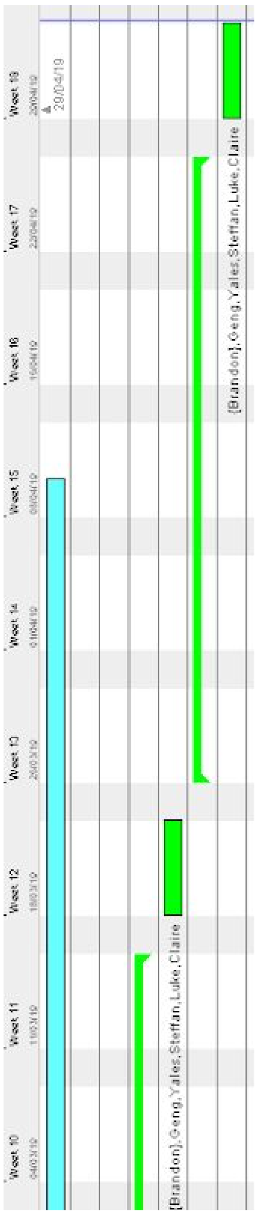
56	Document review	8	06/03/19	15/03/19	All of the above	{Brandon}Yales, Geng, Steffan, Luke, Claire
57	Presentation	15	25/02/19	15/03/19	none	{Brandon}Yales, Geng, Steffan, Luke, Claire
58	Design presentation	5	18/03/19	23/03/19	none	{Brandon}Yales, Geng, Steffan, Luke, Claire

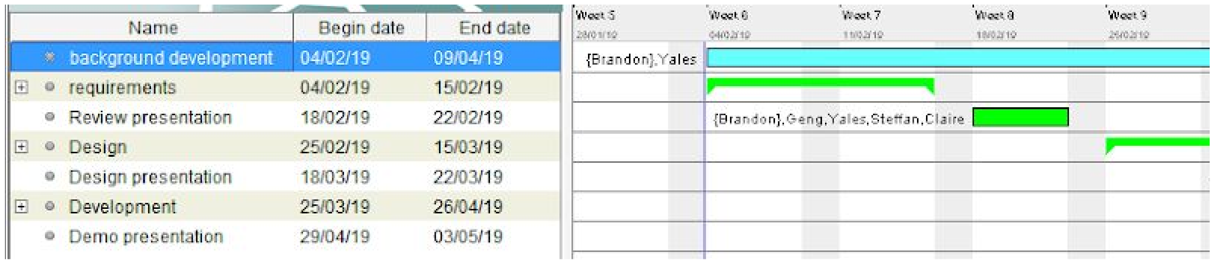
Development tasks

ID	Task	Duration (d)	Start on	End by	Dependencies	Assigned resources
59	Make user database	1	25/03/19	26/03/19	none	{Claire}
60	Make student database	1	25/03/19	26/03/19	none	{Claire}
61	Make profesor database	1	25/03/19	26/03/19	none	{Claire}
62	Make question database	1	25/03/19	26/03/19	none	{Claire}
63	Make modules database	1	25/03/19	26/03/19	none	{Claire}
64	Make exams database	1	25/03/19	26/03/19	none	{Claire}
65	Make answers database	1	25/03/19	26/03/19	none	{Claire}
66	Security	5	26/03/19	02/04/19	65	{Brandon}
67	Build login page	3	25/03/19	28/03/19	none	{Steffan} Luke
68	Build student page	3	25/03/19	28/03/19	none	{Steffan} Luke
69	Build profesor page	3	25/03/19	28/03/19	none	{Steffan} Luke
70	Build course rep page	3	25/03/19	28/03/19	none	{Steffan} Luke
71	Build question input page	3	25/03/19	28/03/19	none	{Steffan} Luke
72	Build answer/stats	3	25/03/19	28/03/19	none	{Steffan} Luke

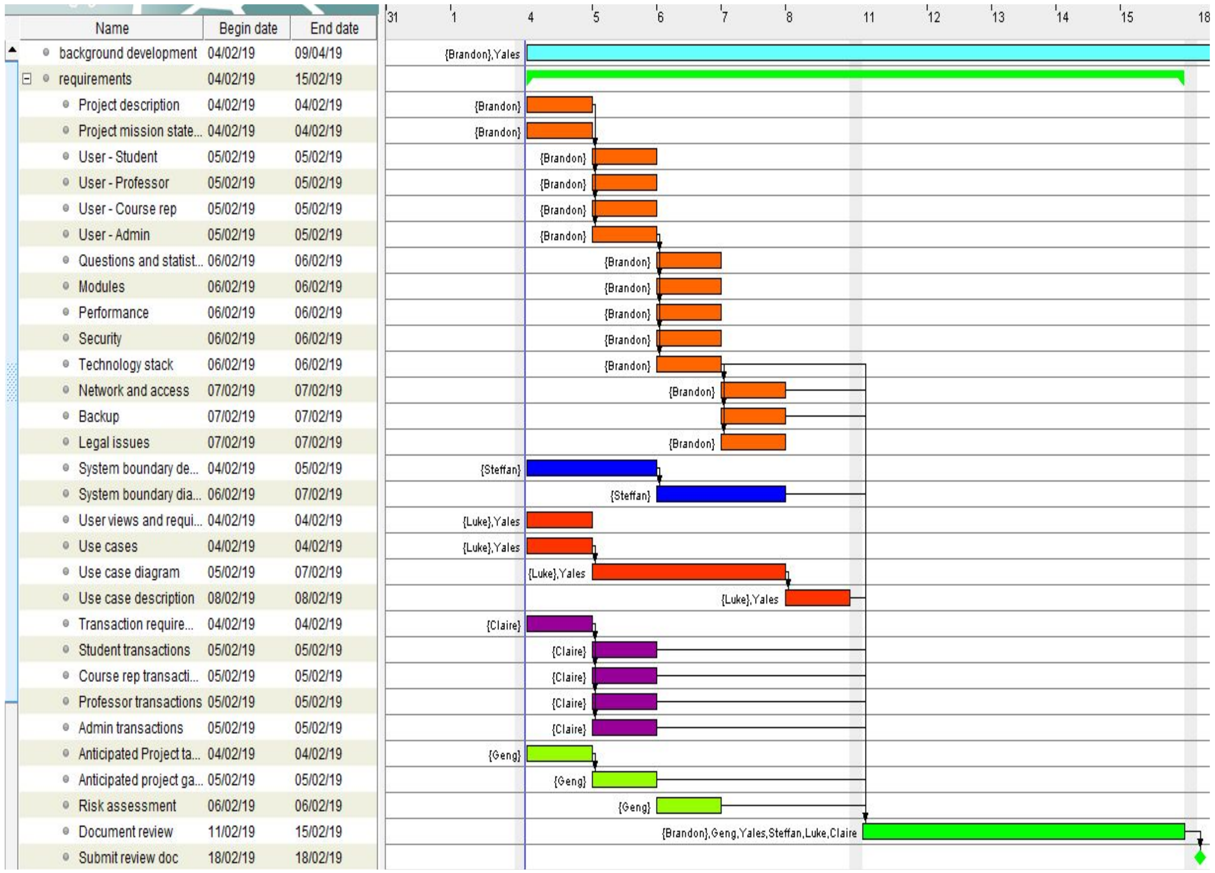
	page					
73	Question generation algorithm	12	25/03/19	10/04/19	none	{Yales}
74	Setup web server	1	25/03/19	26/03/19	none	{Steffan} Brandon
75	Set up question input	1	26/03/19	27/03/19	74	{Steffan} Geng
76	Set up question output	1	27/03/19	28/03/19	75	{Steffan} Geng
77	Set up answer recording	1	28/03/19	29/03/19	76	{Steffan} Geng
78	Set up answer processing	1	29/03/19	30/03/19	77	{Steffan} Geng
79	Set up answer output	1	01/04/19	02/04/19	78	{Steffan} Geng
80	Set up report output	1	02/04/19	03/04/19	79	{Claire}
81	Set up question generation	1	10/04/19	11/04/19	73	{Yales} Brandon
82	Set up question control	2	26/03/19	29/03/19	74	{Claire}
83	Testing	10	11/04/19	25/04/19	All of the above	{Brandon}
84	Program review	2	25/04/19	27/04/19	All of the above	{Luke} Geng
85	Demo presentation	5	29/04/19	03/04/19	none	{Luke, Geng, Claire, Yales, Brandon, Steffan}

The Gantt chart software doesn’t allow us to export to PDF or to PNG, so we’ve had to screenshot it here.

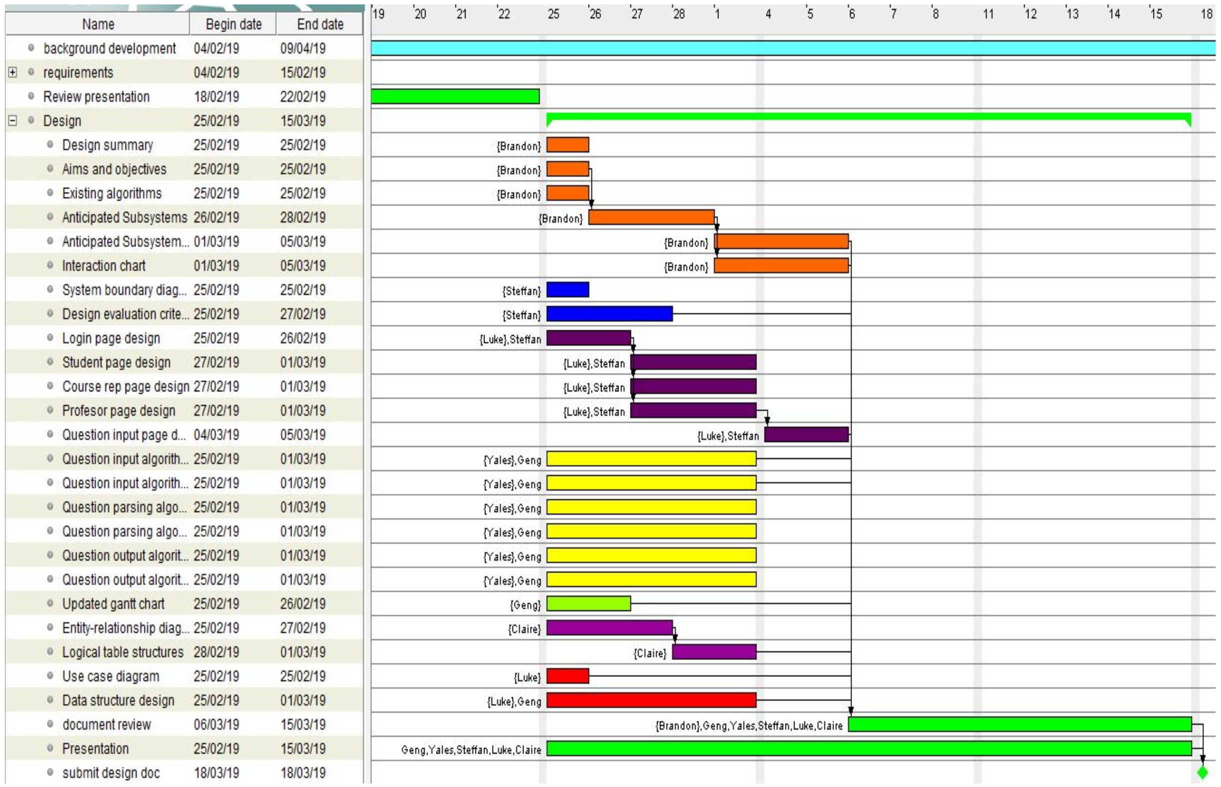

Week 10
04/03/19
Week 11
11/03/19
Week 12
18/03/19
Week 13
25/03/19
Week 14
01/04/19
Week 15
08/04/19
Week 16
15/04/19
Week 17
22/04/19
Week 18
29/04/19
29/04/19
[Brandon],Geng,Yates,Steffan,Luke,Claire
[Brandon],Geng,Yates,Steffan,Luke,Claire



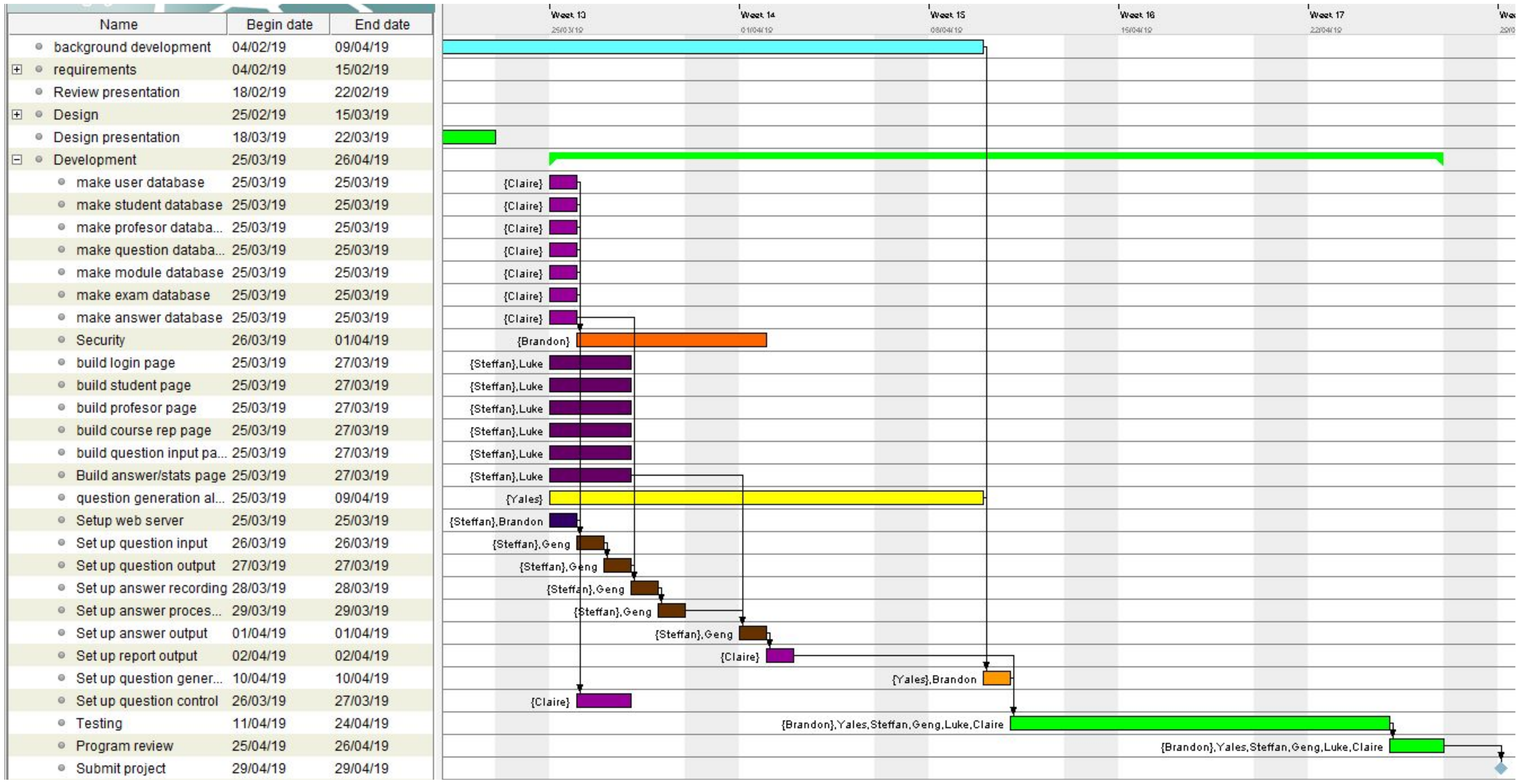
Name	Begin date	End date
background development	04/02/19	09/04/19
requirements	04/02/19	15/02/19
Review presentation	18/02/19	22/02/19
Design	25/02/19	15/03/19
Design presentation	18/03/19	22/03/19
Development	25/03/19	26/04/19
Demo presentation	29/04/19	03/05/19



Name	Begin date	End date
background development	04/02/19	09/04/19
requirements	04/02/19	15/02/19
Project description	04/02/19	04/02/19
Project mission state...	04/02/19	04/02/19
User - Student	05/02/19	05/02/19
User - Professor	05/02/19	05/02/19
User - Course rep	05/02/19	05/02/19
User - Admin	05/02/19	05/02/19
Questions and statist...	06/02/19	06/02/19
Modules	06/02/19	06/02/19
Performance	06/02/19	06/02/19
Security	06/02/19	06/02/19
Technology stack	06/02/19	06/02/19
Network and access	07/02/19	07/02/19
Backup	07/02/19	07/02/19
Legal issues	07/02/19	07/02/19
System boundary de...	04/02/19	05/02/19
System boundary dia...	06/02/19	07/02/19
User views and requi...	04/02/19	04/02/19
Use cases	04/02/19	04/02/19
Use case diagram	05/02/19	07/02/19
Use case description	08/02/19	08/02/19
Transaction require...	04/02/19	04/02/19
Student transactions	05/02/19	05/02/19
Course rep transacti...	05/02/19	05/02/19
Professor transactions	05/02/19	05/02/19
Admin transactions	05/02/19	05/02/19
Anticipated Project ta...	04/02/19	04/02/19
Anticipated project ga...	05/02/19	05/02/19
Risk assessment	06/02/19	06/02/19
Document review	11/02/19	15/02/19
Submit review doc	18/02/19	18/02/19

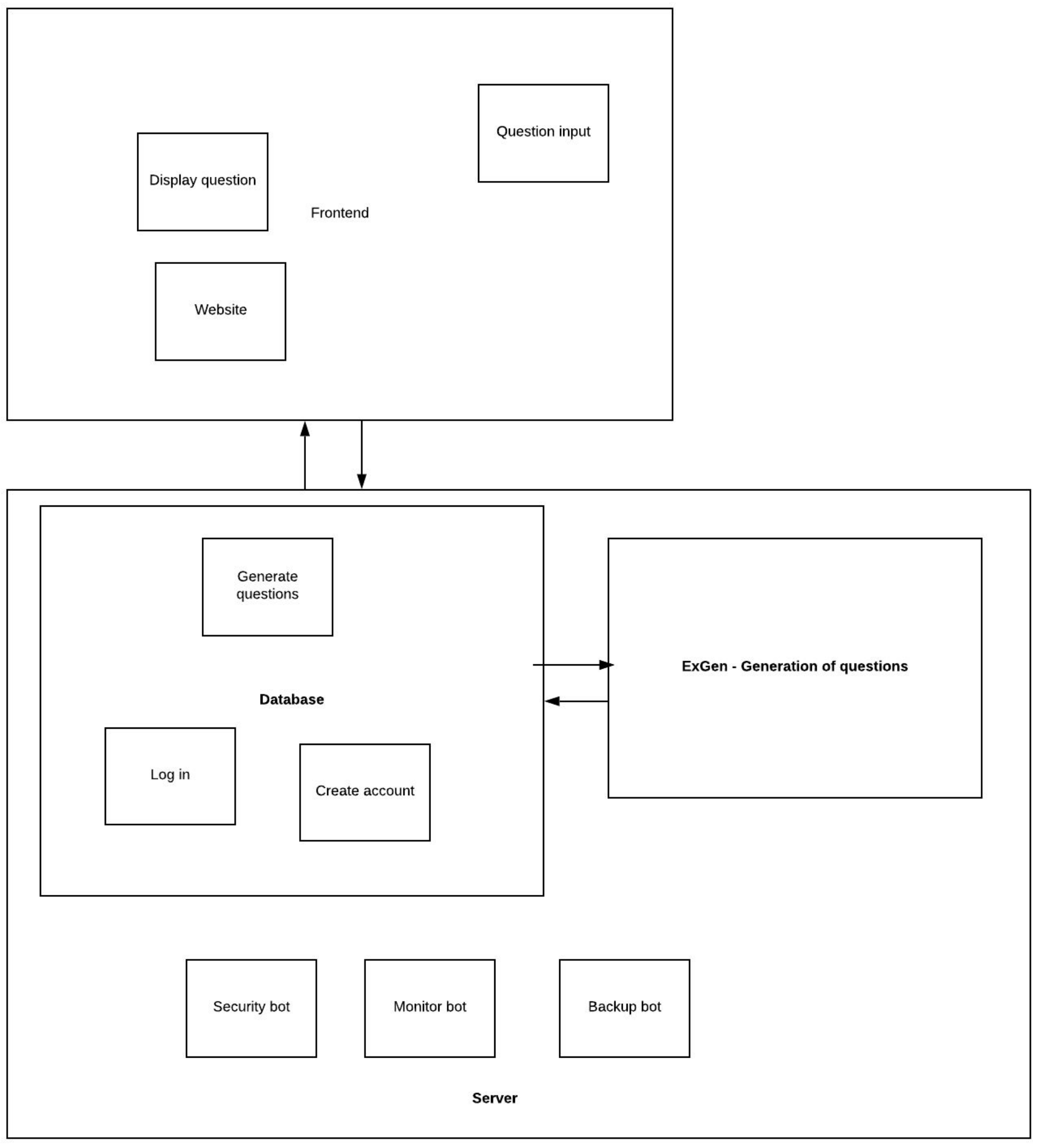


Name	Begin date	End date
background development	04/02/19	09/04/19
requirements	04/02/19	15/02/19
Review presentation	18/02/19	22/02/19
Design	25/02/19	15/03/19
Design summary	25/02/19	25/02/19
Aims and objectives	25/02/19	25/02/19
Existing algorithms	25/02/19	25/02/19
Anticipated Subsystems	26/02/19	28/02/19
Anticipated Subsystem...	01/03/19	05/03/19
Interaction chart	01/03/19	05/03/19
System boundary diag...	25/02/19	25/02/19
Design evaluation crite...	25/02/19	27/02/19
Login page design	25/02/19	26/02/19
Student page design	27/02/19	01/03/19
Course rep page design	27/02/19	01/03/19
Profesor page design	27/02/19	01/03/19
Question input page d...	04/03/19	05/03/19
Question input algorith...	25/02/19	01/03/19
Question input algorith...	25/02/19	01/03/19
Question parsing algo...	25/02/19	01/03/19
Question parsing algo...	25/02/19	01/03/19
Question output algorit...	25/02/19	01/03/19
Question output algorit...	25/02/19	01/03/19
Updated gantt chart	25/02/19	26/02/19
Entity-relationship diag...	25/02/19	27/02/19
Logical table structures	28/02/19	01/03/19
Use case diagram	25/02/19	25/02/19
Data structure design	25/02/19	01/03/19
document review	06/03/19	15/03/19
Presentation	25/02/19	15/03/19
submit design doc	18/03/19	18/03/19



Name	Begin date	End date
background development	04/02/19	09/04/19
⊞ requirements	04/02/19	15/02/19
Review presentation	18/02/19	22/02/19
⊞ Design	25/02/19	15/03/19
Design presentation	18/03/19	22/03/19
⊟ Development	25/03/19	26/04/19
make user database	25/03/19	25/03/19
make student database	25/03/19	25/03/19
make profesor databa...	25/03/19	25/03/19
make question databa...	25/03/19	25/03/19
make module database	25/03/19	25/03/19
make exam database	25/03/19	25/03/19
make answer database	25/03/19	25/03/19
Security	26/03/19	01/04/19
build login page	25/03/19	27/03/19
build student page	25/03/19	27/03/19
build profesor page	25/03/19	27/03/19
build course rep page	25/03/19	27/03/19
build question input pa...	25/03/19	27/03/19
Build answer/stats page	25/03/19	27/03/19
question generation al...	25/03/19	09/04/19
Setup web server	25/03/19	25/03/19
Set up question input	26/03/19	26/03/19
Set up question output	27/03/19	27/03/19
Set up answer recording	28/03/19	28/03/19
Set up answer proces...	29/03/19	29/03/19
Set up answer output	01/04/19	01/04/19
Set up report output	02/04/19	02/04/19
Set up question gener...	10/04/19	10/04/19
Set up question control	26/03/19	27/03/19
Testing	11/04/19	24/04/19
Program review	25/04/19	26/04/19
Submit project	29/04/19	29/04/19

Subsystem Design



The frontend will have 3 main subsystems. Display question, the website itself and an input form so the professor can input questions.

When a professor writes a question, it gets sent to the database through a pre-configured SQL Transaction. When a student loads an exam, the questions to that exam will be quired to the database using a pre-configured SQL transaction. If the database does not have the question for that exam, it asks ExGen to generate the question.

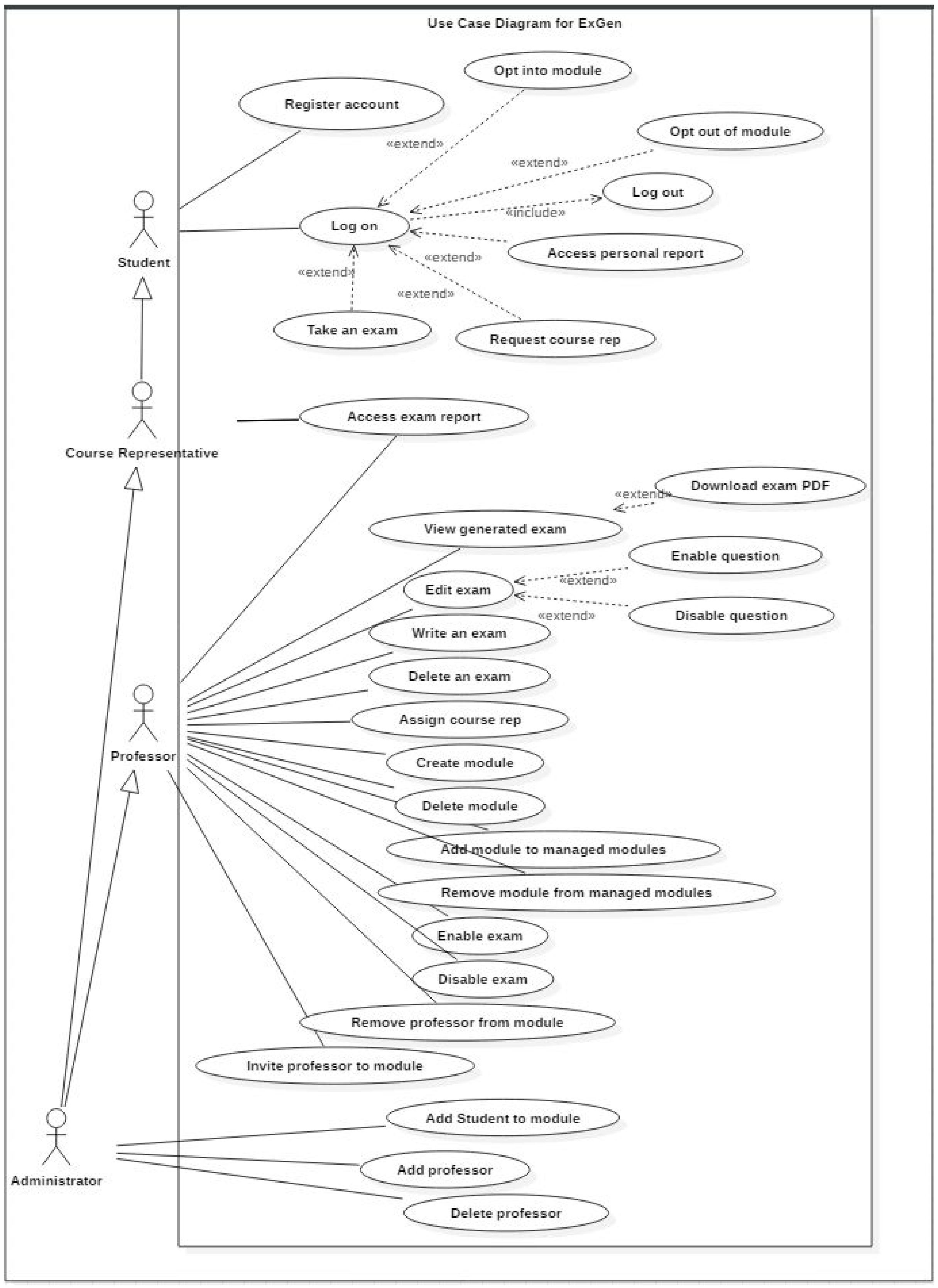
ExGen will then send back the generated question to the database. The database then sends it to the frontend for the user. No user in our system has direct access to the database apart from the administrators, who maintain the system.

The server has 3 robots to help maintain it. The security bot was talked about in our requirements document. It maintains the security of the system. If the professors code takes too long to execute, it ends that thread and informs the professor of an error.

The monitor bot is only partly hosted on our system. It has a sister monitor bot on another server. If these two bots do not talk to each other, then the server is presumed to be down. It checks every minute to make sure the system is operational.

The backup bot backs up the database locally, and then again to an overseas server.

Use Case Diagram



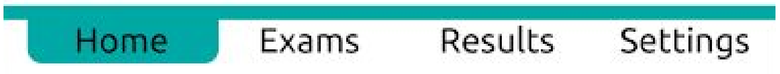
Note: There are certain use cases for the Admin that have been stated in order to avoid any unsolvable situations that could occur in our system, so the admin can have total power and control of everything.

Frontend

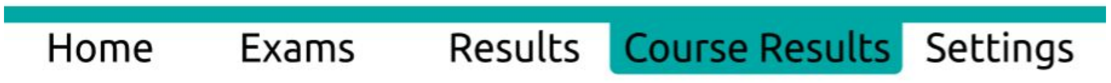
Interface design

The interface is designed to be very simplistic and minimalistic. The basic idea of the interface is that all accessible features are displayed in the navigation bar on the top of the page. This bar will also contain the "ExGen" logo and will appear on all pages. The bar will contain a logout button for all users.

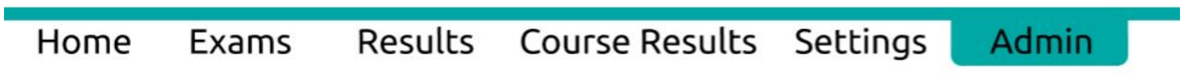
The navigation bar for students will contain the following: Home, Exams, Results and Settings and will look like this.



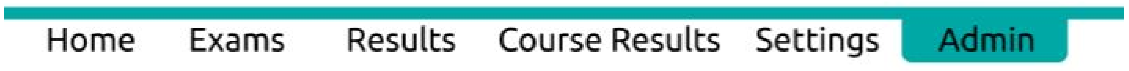
The navigation bar for course representatives will contain the following: Home, Exams, Results, Course Results, Settings and will look like this.



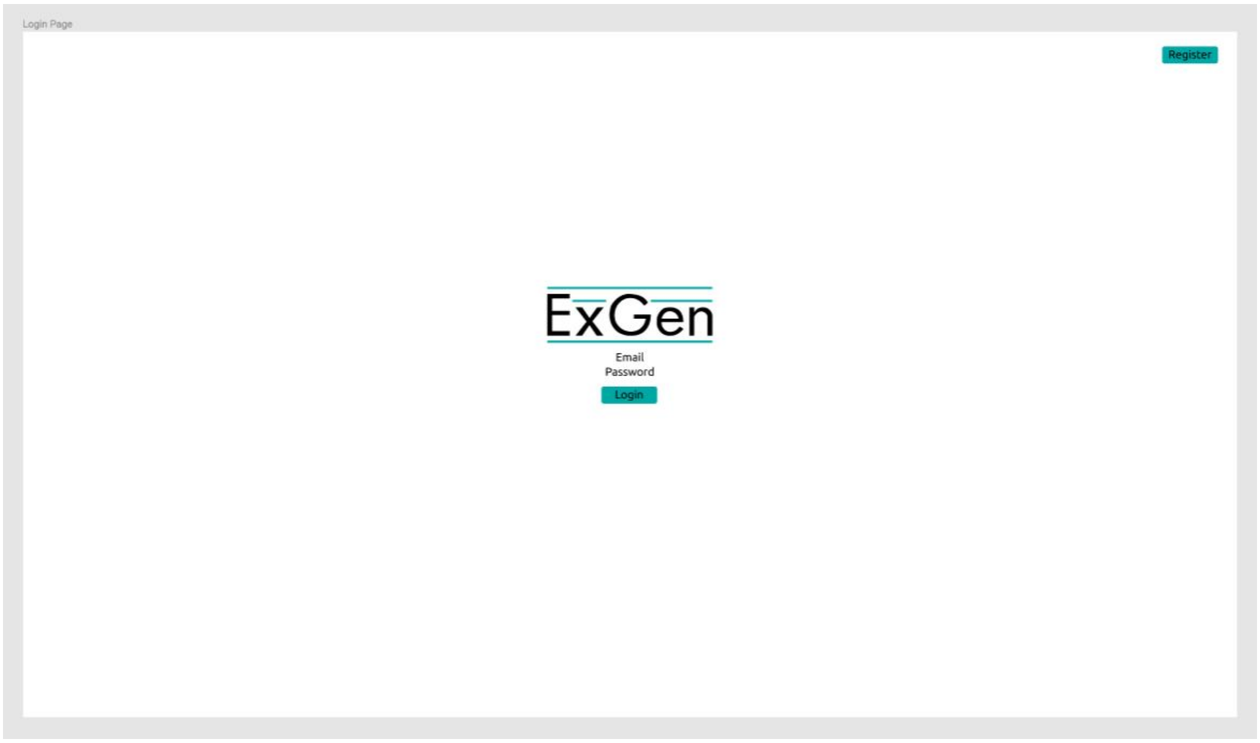
The navigation bar for professors will contain the following: Home, Modules, Exams, Course Results and Settings.



The navigation bar for admins will contain the following: Home, Exams, Results, Modules, Course Results and Settings.



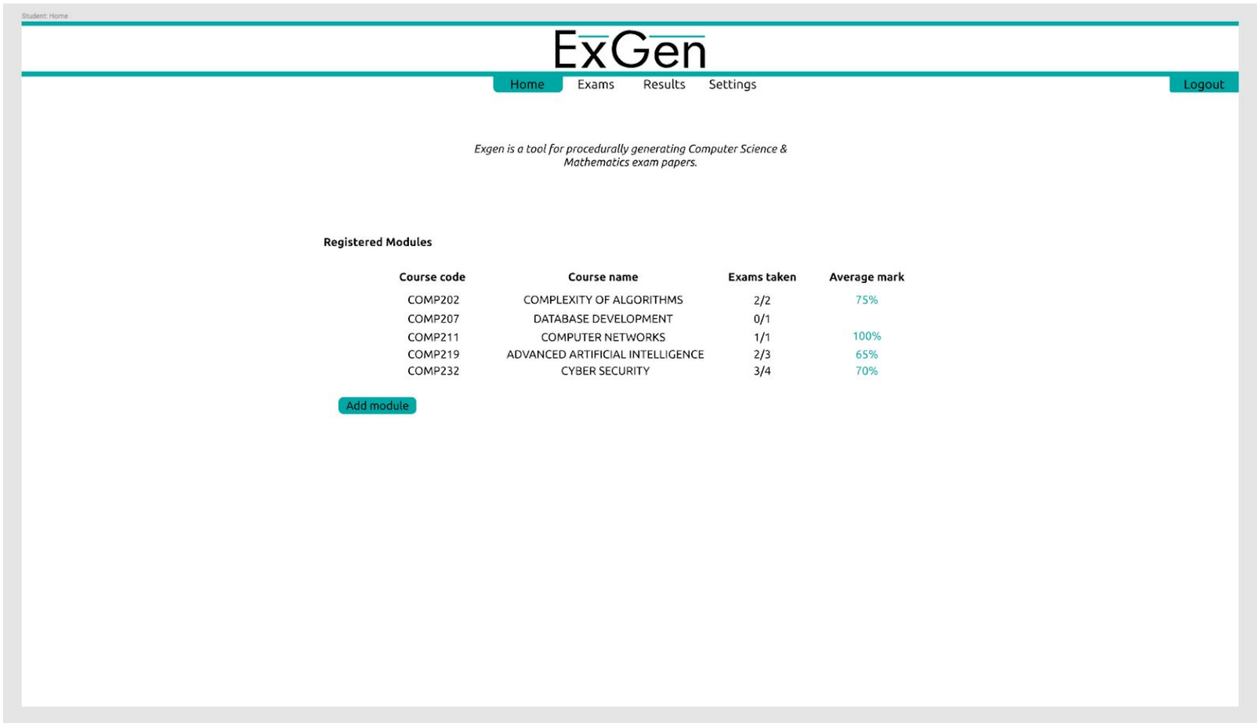
The theme that we are going for is a simple white background, blue features and black text with our font being the free to use "Ubuntu" font. The default page for our interface is the login page unless the user has already logged in.
The login page will be very simple. It will contain our logo, an email input field, a password input field, a “login” button and a register button. If the user is logged in then the default page is the home page. The login page will look like this.



All 4 users will have the home page where they can see basic information about their relevant permissions.The Students and course representatives(as they are also students) "home" page will contain a table with the following information:

- Course code
- Course name
- Exams taken
- Average mark

There will also be an “Add module” button here where the user can opt-in to modules.

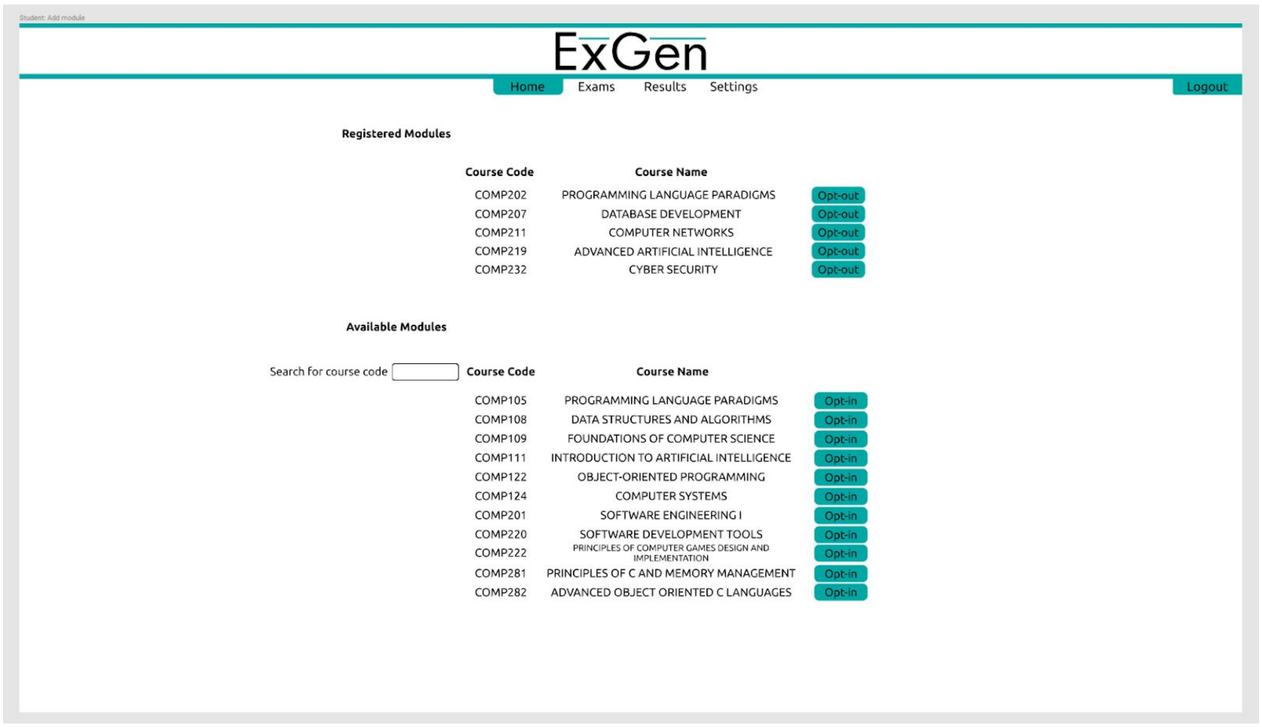


Exgen is a tool for procedurally generating Computer Science & Mathematics exam papers.

Registered Modules

Course code	Course name	Exams taken	Average mark
COMP202	COMPLEXITY OF ALGORITHMS	2/2	75%
COMP207	DATABASE DEVELOPMENT	0/1	
COMP211	COMPUTER NETWORKS	1/1	100%
COMP219	ADVANCED ARTIFICIAL INTELLIGENCE	2/3	65%
COMP232	CYBER SECURITY	3/4	70%

When the “Add module” button is pressed the following page will replace the content on the home page. This page contains a list of the modules that the student has already opted into and a list of available modules that the student can opt into. The user can press “opt-out” or “opt-in” on the desired module to leave or join that module. Upon clicking these the two tables will update with the new information. The user can also search the available modules table by the course code.



Registered Modules

Course Code	Course Name	
COMP202	PROGRAMMING LANGUAGE PARADIGMS	Opt-out
COMP207	DATABASE DEVELOPMENT	Opt-out
COMP211	COMPUTER NETWORKS	Opt-out
COMP219	ADVANCED ARTIFICIAL INTELLIGENCE	Opt-out
COMP232	CYBER SECURITY	Opt-out

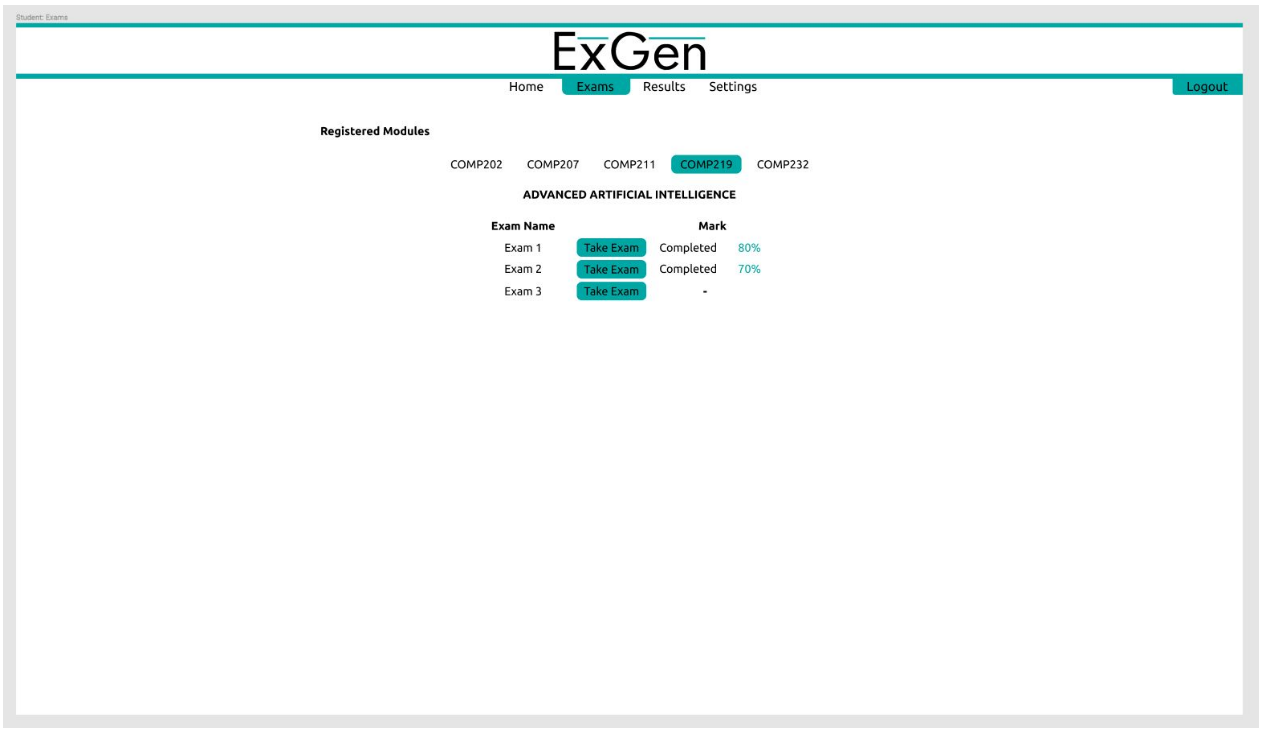
Available Modules

Search for course code

Course Code	Course Name	
COMP105	PROGRAMMING LANGUAGE PARADIGMS	Opt-in
COMP108	DATA STRUCTURES AND ALGORITHMS	Opt-in
COMP109	FOUNDATIONS OF COMPUTER SCIENCE	Opt-in
COMP111	INTRODUCTION TO ARTIFICIAL INTELLIGENCE	Opt-in
COMP122	OBJECT-ORIENTED PROGRAMMING	Opt-in
COMP124	COMPUTER SYSTEMS	Opt-in
COMP201	SOFTWARE ENGINEERING I	Opt-in
COMP220	SOFTWARE DEVELOPMENT TOOLS	Opt-in
COMP222	PRINCIPLES OF COMPUTER GAMES DESIGN AND IMPLEMENTATION	Opt-in
COMP281	PRINCIPLES OF C AND MEMORY MANAGEMENT	Opt-in
COMP282	ADVANCED OBJECT ORIENTED C LANGUAGES	Opt-in

The home page for Admin and course representatives will be the same as students as they are also students.

The exam page for students and course representatives will look like this.



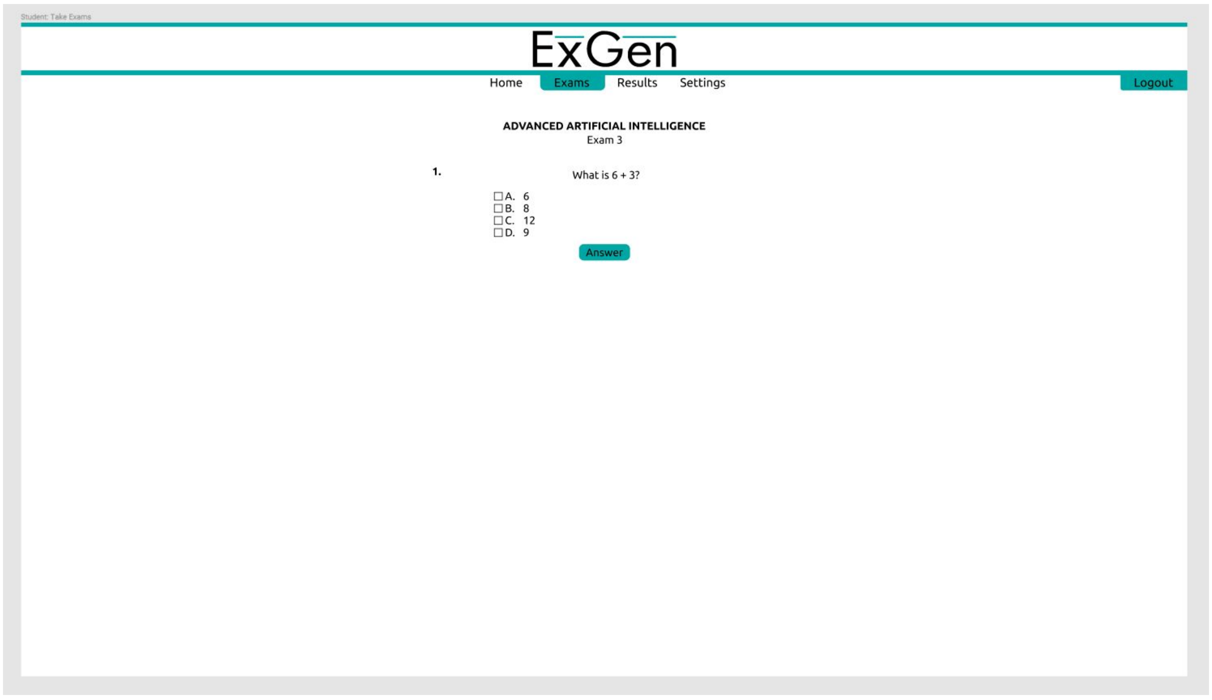
Registered Modules

COMP202 COMP207 COMP211 COMP219 COMP232

ADVANCED ARTIFICIAL INTELLIGENCE

Exam Name		Mark	
Exam 1	Take Exam	Completed	80%
Exam 2	Take Exam	Completed	70%
Exam 3	Take Exam	-	

The user can change the list of available exams by clicking on another course code. The interface will default to the first registered course code of the user. If there are no exams for a given module then the list will be empty and should display “No exams available”. If a user has completed an exam from the list, the mark the user got for that exam should be displayed. The “take exam” button will generate an exam where questions will be loaded and answered one by one on this page. The questions will overlay/replace what is in the exam page and will look like this.



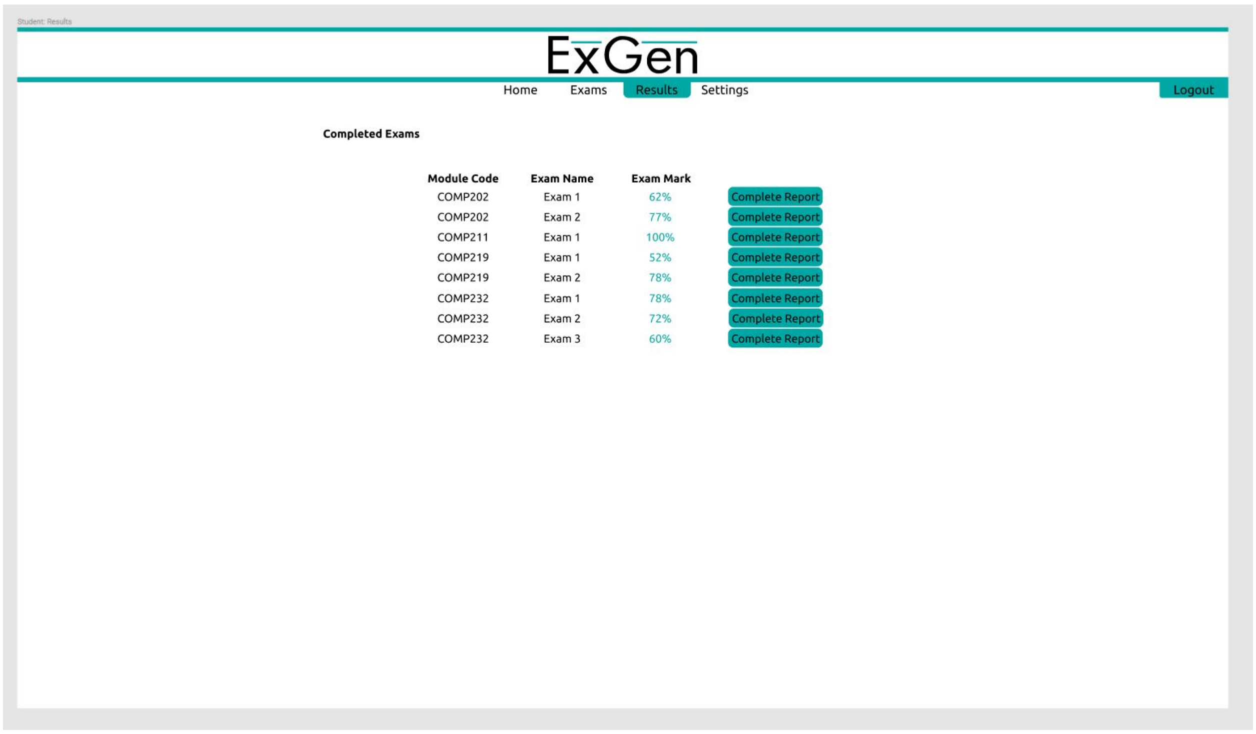
ADVANCED ARTIFICIAL INTELLIGENCE
Exam 3

1. What is 6 + 3?

☐A. 6
☐B. 8
☐C. 12
☐D. 9

The user can then tick one of the answers and click the “answer” button. The system will then check if the question is correct and if it is correct then the system will return a new question to the user. If the user answers the question incorrectly then the system will show another example of the same question.

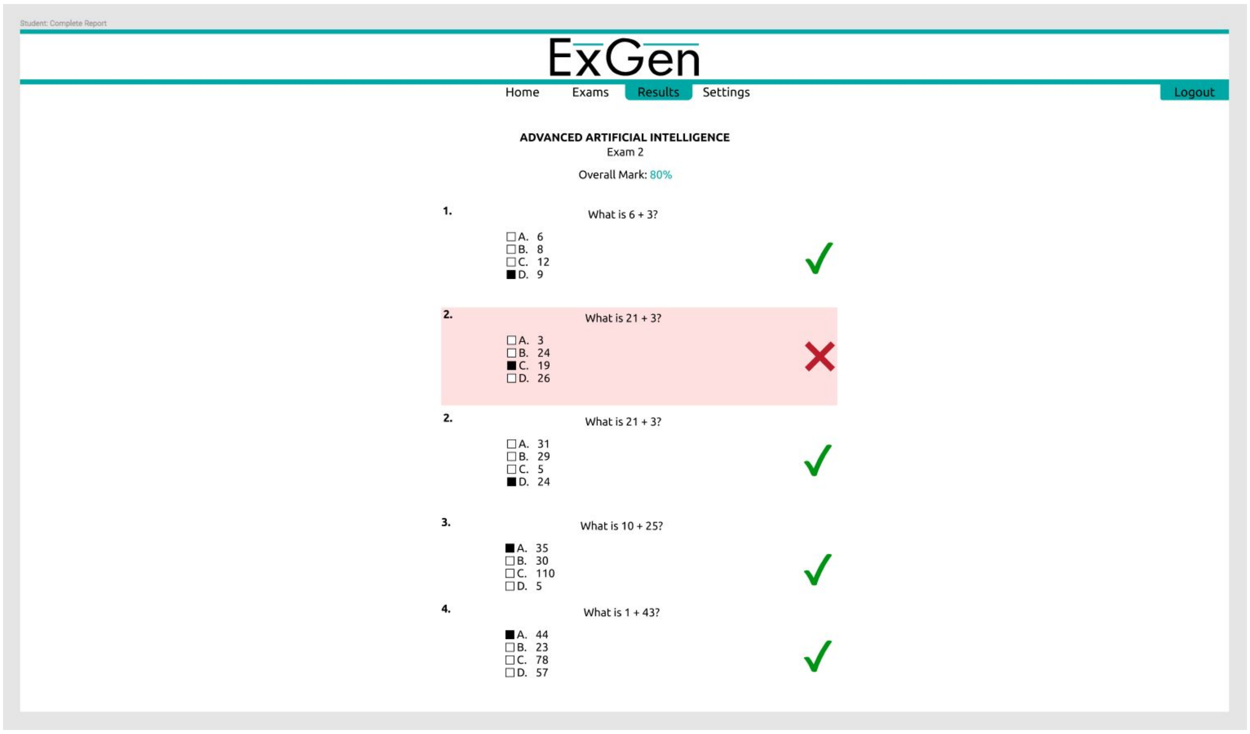
The next page is the students, course representatives and admins results page. This page contains a list of all completed exams with their mark for each exam. They can then get a complete report per exam.



Completed Exams

Module Code	Exam Name	Exam Mark	
COMP202	Exam 1	62%	Complete Report
COMP202	Exam 2	77%	Complete Report
COMP211	Exam 1	100%	Complete Report
COMP219	Exam 1	52%	Complete Report
COMP219	Exam 2	78%	Complete Report
COMP232	Exam 1	78%	Complete Report
COMP232	Exam 2	72%	Complete Report
COMP232	Exam 3	60%	Complete Report

The complete report button opens the answered exam that the user completed. This will also show the users failed attempts at a question. If an exam had 3 questions and questions 1 and 3 were answered correctly but question 2 was incorrect the first time. The report would contain 4 questions. This screen will look like this.



ADVANCED ARTIFICIAL INTELLIGENCE
Exam 2

Overall Mark: 80%

1. What is 6 + 3?

☐A. 6
☐B. 8
☐C. 12
■D. 9

✓

2. What is 21 + 3?

☐A. 3
☐B. 24
■C. 19
☐D. 26

✗

2. What is 21 + 3?

☐A. 31
☐B. 29
☐C. 5
■D. 24

✓

3. What is 10 + 25?

■A. 35
☐B. 30
☐C. 110
☐D. 5

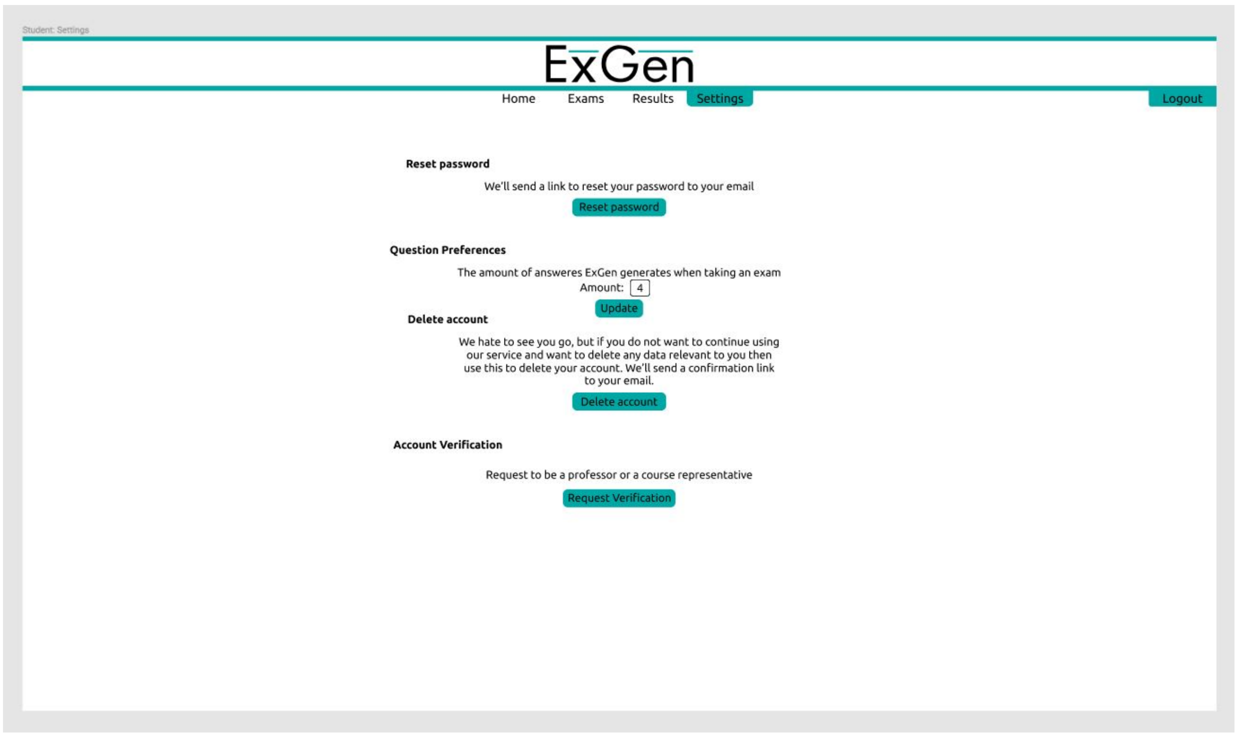
✓

4. What is 1 + 43?

■A. 44
☐B. 23
☐C. 78
☐D. 57

✓

The settings page will be the same for every user. The users will be able to reset their password, change their question preferences, delete their account and ask for different user privileges. The settings page will look like this.



Reset password

We'll send a link to reset your password to your email

Question Preferences

The amount of answeres ExGen generates when taking an exam

Amount: 4

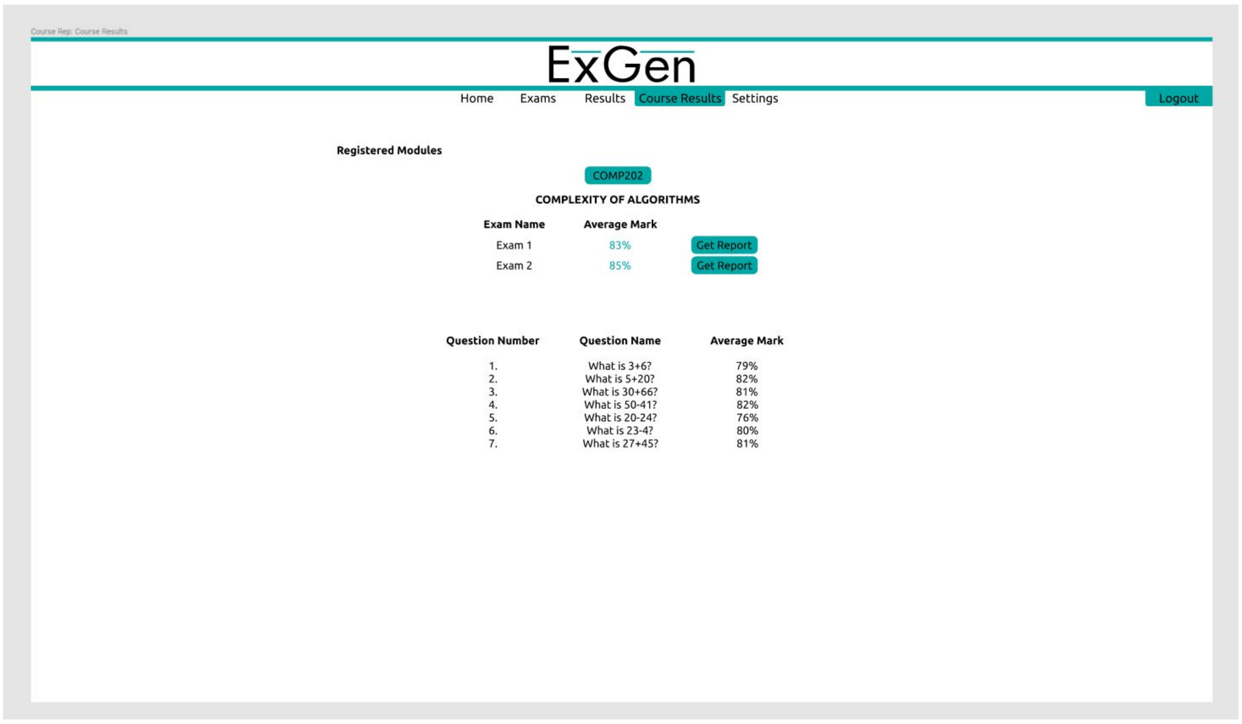
Delete account

We hate to see you go, but if you do not want to continue using our service and want to delete any data relevant to you then use this to delete your account. We'll send a confirmation link to your email.

Account Verification

Request to be a professor or a course representative

Course reps and the professor can view results for exams from all users who completed that exam. This page will list all exams that the course reps are a rep of. The course rep can click on "Get Report" which will bring up average marks per question for all of the exam.



Registered Modules

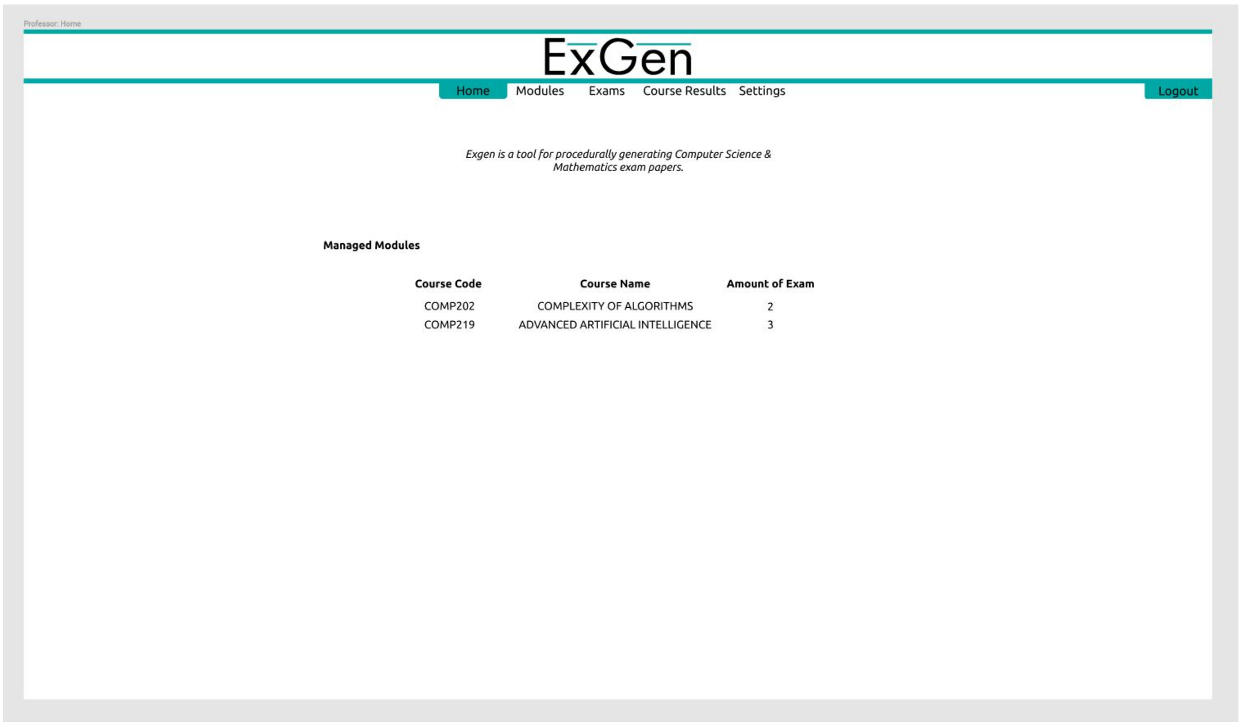
COMP202

COMPLEXITY OF ALGORITHMS

Exam Name	Average Mark	
Exam 1	83%	Get Report
Exam 2	85%	Get Report

Question Number	Question Name	Average Mark
1.	What is 3+6?	79%
2.	What is 5+20?	82%
3.	What is 30+66?	81%
4.	What is 50-41?	82%
5.	What is 20-24?	76%
6.	What is 23-4?	80%
7.	What is 27+45?	81%

The professor's home page is similar to the other users home pages but the table shows the professors modules and the modules that the professor handles.

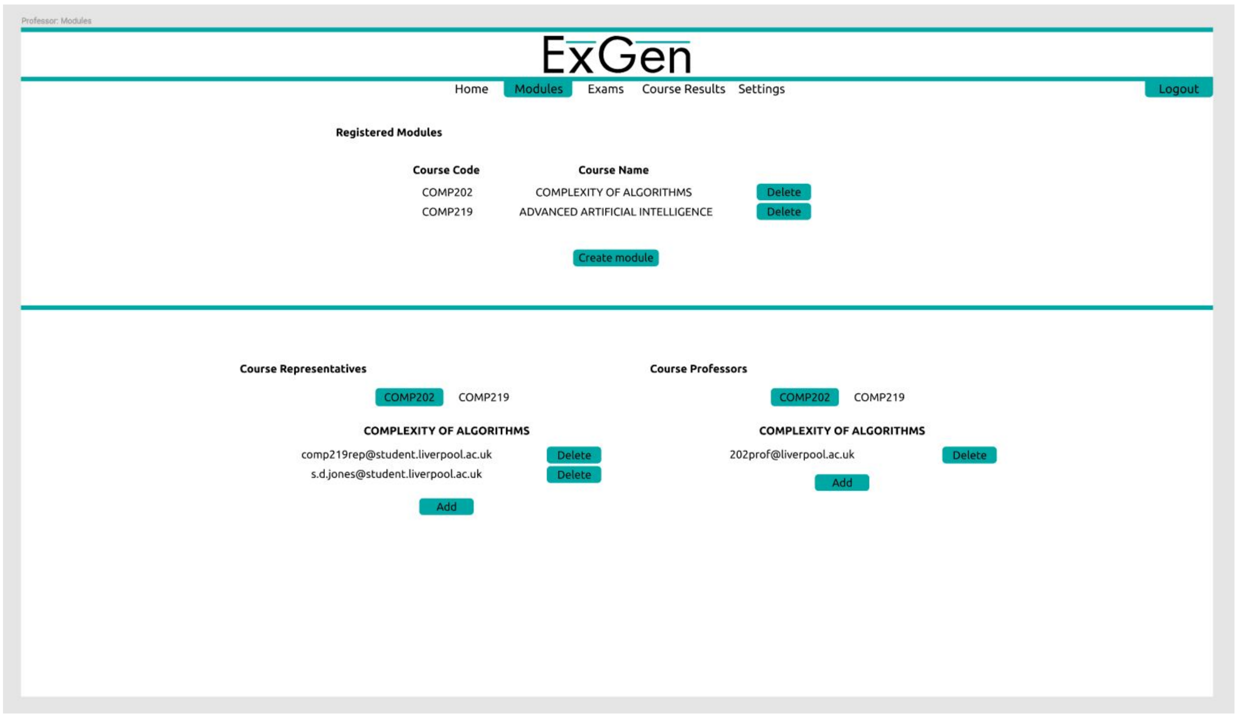


Exgen is a tool for procedurally generating Computer Science & Mathematics exam papers.

Managed Modules

Course Code	Course Name	Amount of Exam
COMP202	COMPLEXITY OF ALGORITHMS	2
COMP219	ADVANCED ARTIFICIAL INTELLIGENCE	3

The Modules page is where the professor is able to create and delete modules, add course reps and other professors to manage their module.



Registered Modules

Course Code	Course Name	
COMP202	COMPLEXITY OF ALGORITHMS	Delete
COMP219	ADVANCED ARTIFICIAL INTELLIGENCE	Delete

Create module

Course Representatives

COMP202 COMP219

COMPLEXITY OF ALGORITHMS

comp219rep@student.liverpool.ac.uk	Delete
s.d.jones@student.liverpool.ac.uk	Delete

Add

Course Professors

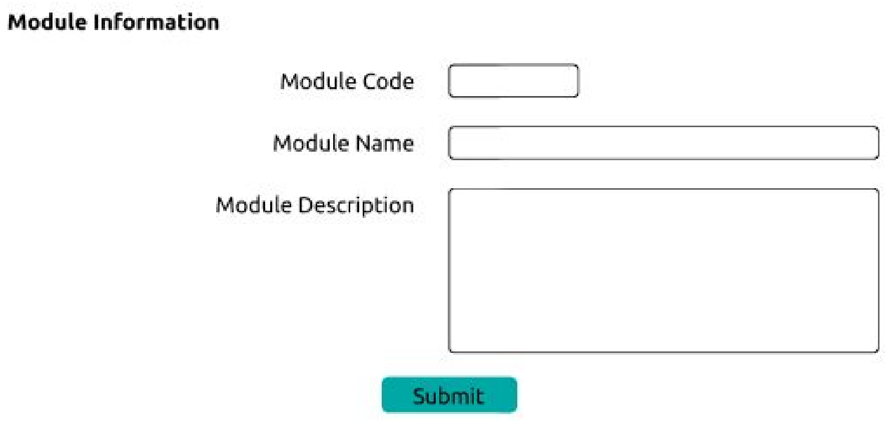
COMP202 COMP219

COMPLEXITY OF ALGORITHMS

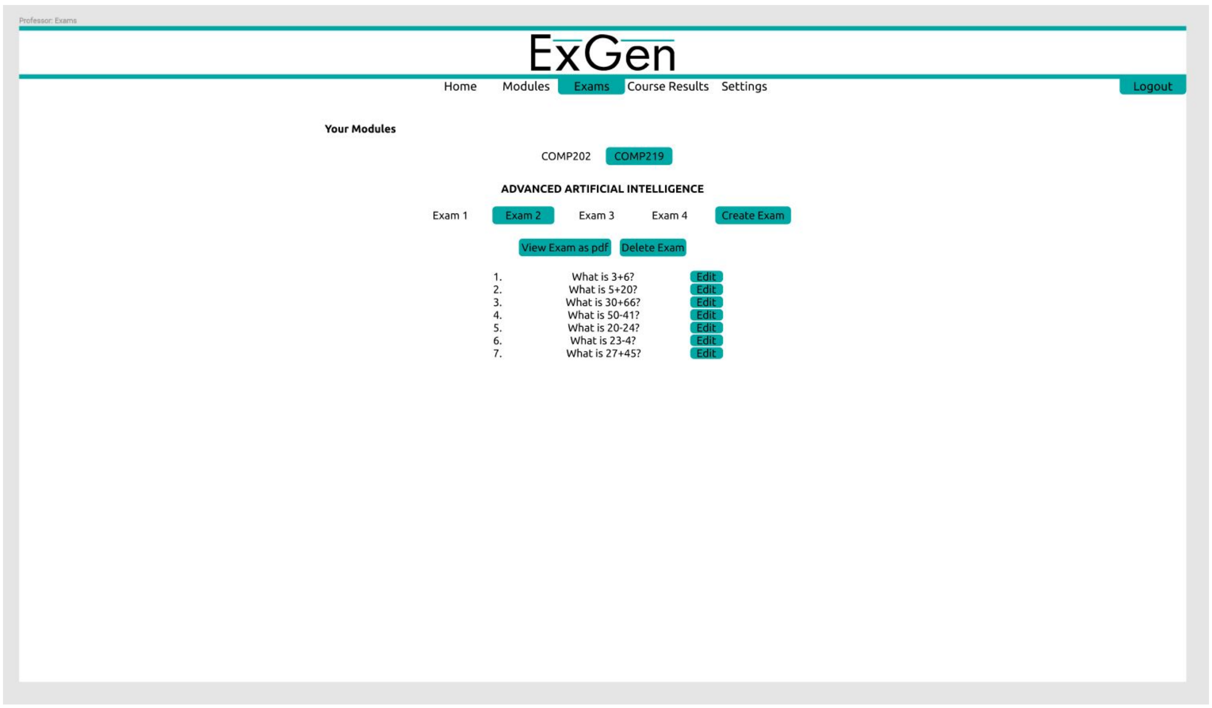
202prof@liverpool.ac.uk	Delete

Add

When a professor presses the “Create module” button a new panel will appear replacing the course representatives/course professors section. This panel is where the professor enters information about the module.

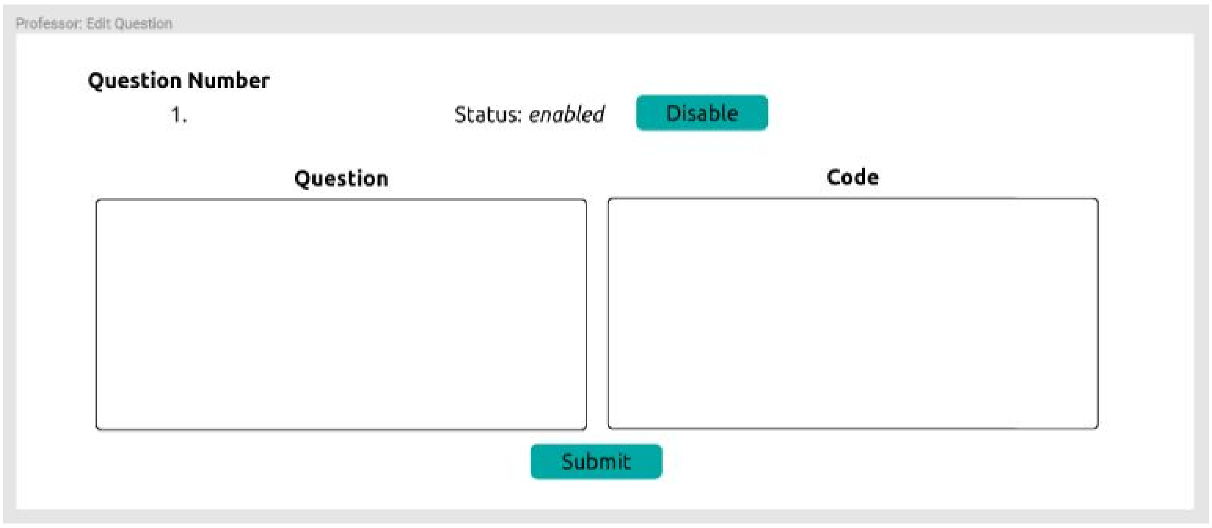


The professor will have an exam page where they can edit existing exams(enable/disable questions, rewrite questions etc), create a new exam, completely delete existing exams and view the exam questions. This page will look like this.

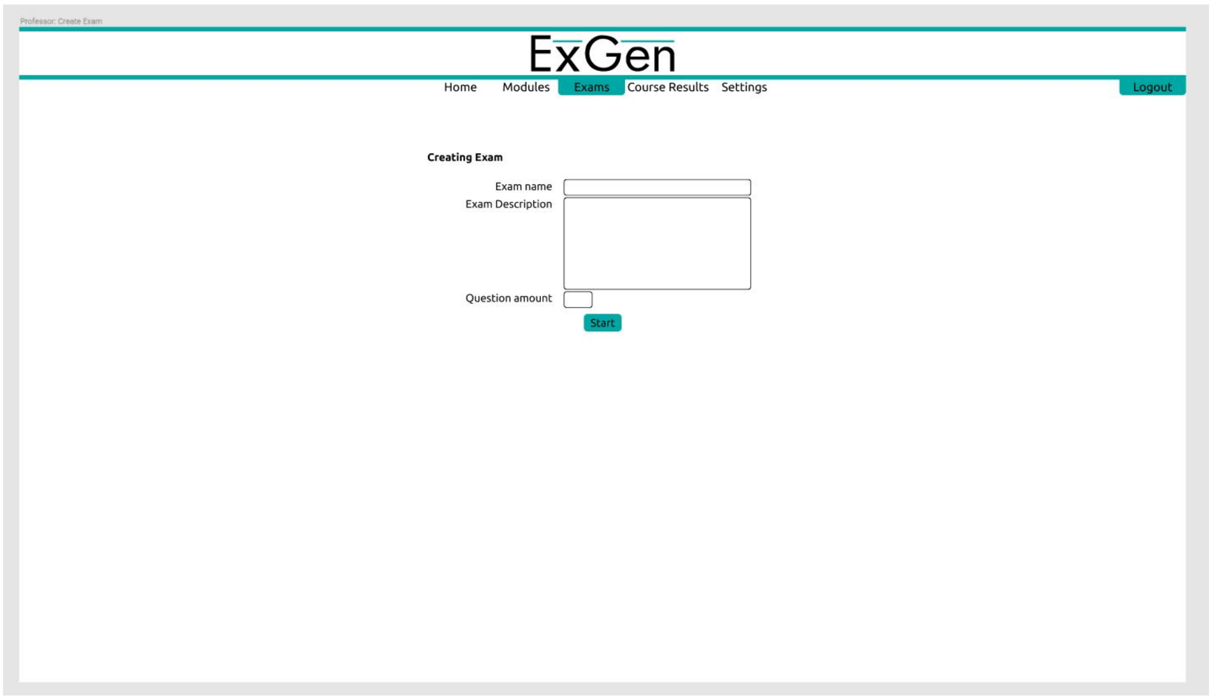


1.	What is 3+6?	Edit
2.	What is 5+20?	Edit
3.	What is 30+66?	Edit
4.	What is 50-41?	Edit
5.	What is 20-24?	Edit
6.	What is 23-4?	Edit
7.	What is 27+45?	Edit

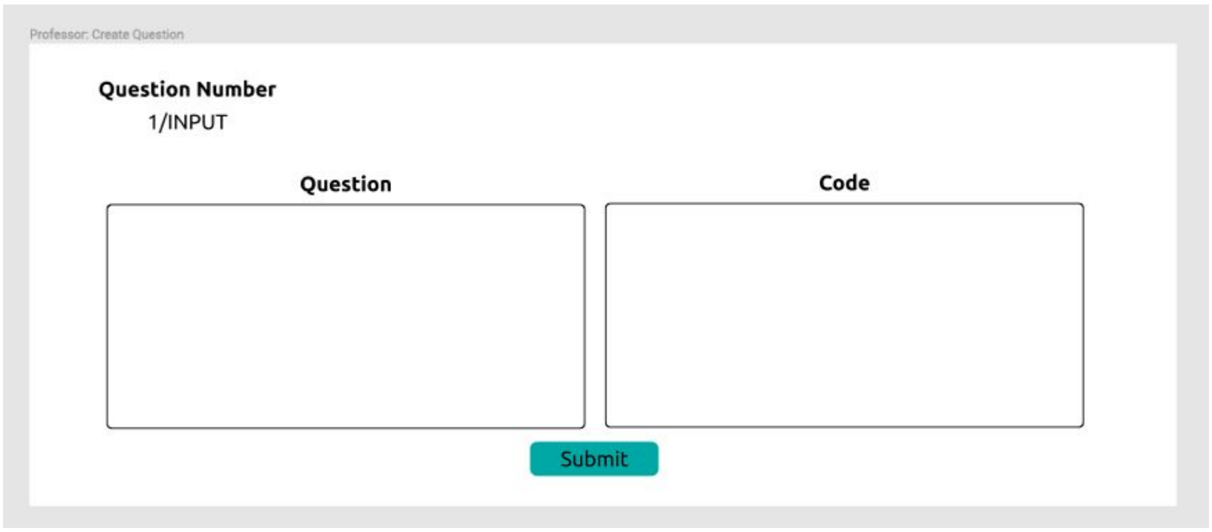
When the professor clicks the “Edit” button a new panel will appear over the questions. This panel is the editing question panel. It will be the panel where the professor enable/disable and edit questions. This panel looks like this. The question would appear in the “question” box and the code for that question would appear in the “code” box.



When the professor clicks “Create exam” a new page will appear. This page will allow the professor to create an exam. This page will have 3 input fields where they will enter the exam name, exam description and the question amount. This page looks like this, it will replace/overlap the default exam page.

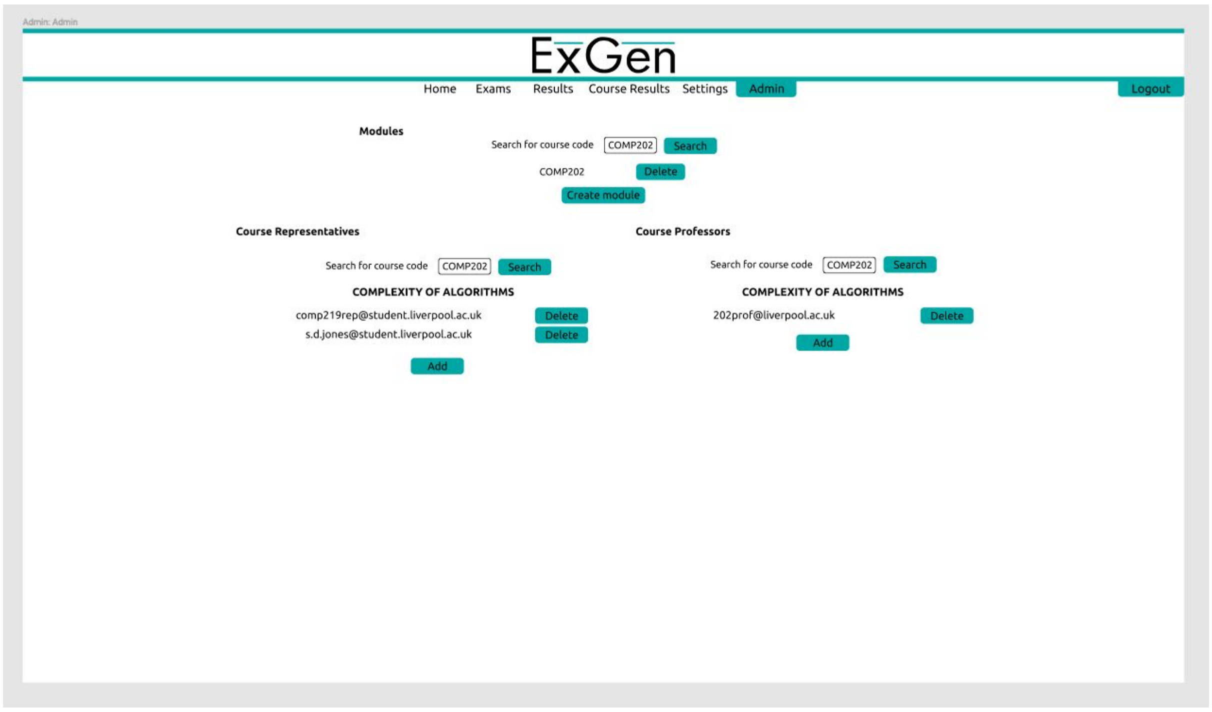


When the user presses the “Start” button a new panel will appear. This panel represents one question out of the question amount. This panel has two fields, a question field and a code field. Once these fields have been filled the user will press submit and the system will clear these fields and ask for the next question. This panel looks like this.



Once the amount of questions has been reached the page will return to its default and the exam will show up in the exam list.

The admin user can see and do everything the other users see. The admins have a specialized page called “Admin” that is very similar to the modules page that professors have but admins down have to own the module to add to them.



Modules

Search for course code COMP202

COMP202

Course Representatives

Search for course code COMP202

COMPLEXITY OF ALGORITHMS

comp219rep@student.liverpool.ac.uk

s.d.jones@student.liverpool.ac.uk

Course Professors

Search for course code COMP202

COMPLEXITY OF ALGORITHMS

202prof@liverpool.ac.uk

Overall, the interface we’ve decided to go with will be easy to implement but will also give the minimalist look that we wanted.

Evaluation design

We will evaluate the interface design by its effectiveness, efficiency and satisfaction. We will divide these into subfactors and use these points as guidelines:

- Clarity: *The information content is conveyed quickly and accurately*
- Conciseness: *Users are not overloaded with extraneous information*
- Consistency: *A unique and good design*
- Detectability: *The user's attention is directed towards information required*
- Legibility: *The information is easy to read*
- Comprehensibility: *The intent and meaning of the interface is clearly understandable and unambiguous*

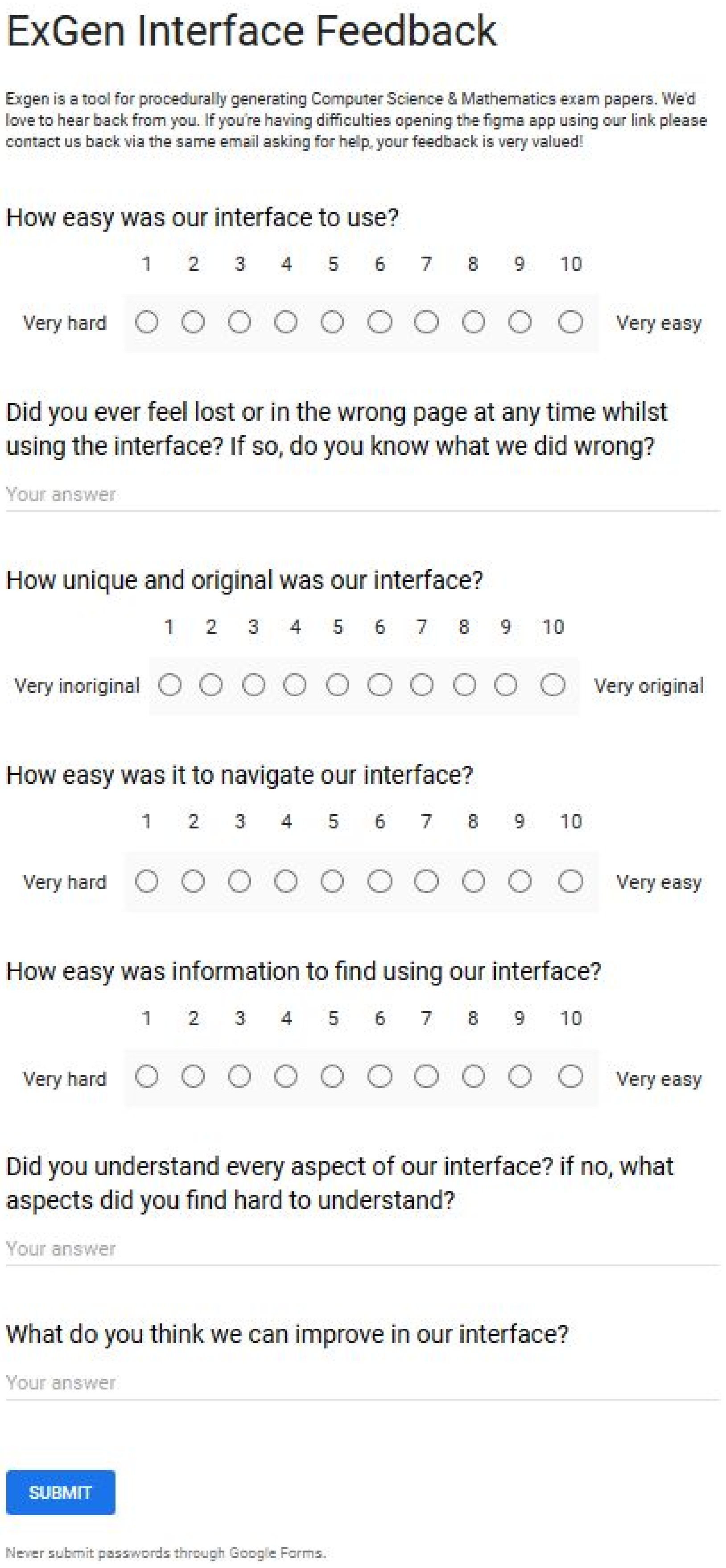
We gave our figma build of the principles of our interface, figma is an online design tool, to the following people:

- Friends from other courses (we decided not to share with other computer science groups)
- Roommates
- Family
- Online friends

This group of people cover a good age group and a good range of technology skills. Balancing the average complaint from this feedback will give us an optimum solution for all age groups and technology skills. We should also be biases in our decision on whether feedback is positive as our system will be used primarily by students who’s age group averages between 17-27. Of course others will use our system i.e. professors and admins but they do not make up the majority. We made a questionnaire along with our figma interface. This questionnaire ties heavily with our guidelines that we made earlier.

- Clarity - How easy was our interface to use?
- Conciseness: - Did you ever feel lost or in the wrong page at any time whilst using the interface?
- Consistency: How unique and original was our interface?
- Detectability: How easy was it to navigate our interface?
- Legibility: How easy was information to find using our interface?
- Comprehensibility: Did you understand every aspect of our interface?
- Feedback: What do you think we can improve in our interface?

The survey we used can be seen below:



ExGen Interface Feedback

Exgen is a tool for procedurally generating Computer Science & Mathematics exam papers. We'd love to hear back from you. If you're having difficulties opening the figma app using our link please contact us back via the same email asking for help, your feedback is very valued!

How easy was our interface to use?

	1	2	3	4	5	6	7	8	9	10	
Very hard	○	○	○	○	○	○	○	○	○	○	Very easy

Did you ever feel lost or in the wrong page at any time whilst using the interface? If so, do you know what we did wrong?

Your answer

How unique and original was our interface?

	1	2	3	4	5	6	7	8	9	10	
Very inoriginal	○	○	○	○	○	○	○	○	○	○	Very original

How easy was it to navigate our interface?

	1	2	3	4	5	6	7	8	9	10	
Very hard	○	○	○	○	○	○	○	○	○	○	Very easy

How easy was information to find using our interface?

	1	2	3	4	5	6	7	8	9	10	
Very hard	○	○	○	○	○	○	○	○	○	○	Very easy

Did you understand every aspect of our interface? if no, what aspects did you find hard to understand?

Your answer

What do you think we can improve in our interface?

Your answer

SUBMIT

Never submit passwords through Google Forms.

From our responses we could see that the majority of people found our system to be accessible and clear to use. This is especially good as some of those answering the survey didn't have much experience with technology. The same applies for consciousness as no one responded stating they felt lost within our system. One thing brought up by our survey was that the design of our interface wasn't particularly unique. Despite this, we feel that it was more important to maintain ease of use than appearance or originality, especially with the time constraints given to us.

Backend

Data Structures

Data structures used in addition.py are chosen and provided by the professor when writing their own question code. The addition.py file produced by our team serves as a **demonstration** of how question generation code should be input into our system by a professor.

The following data structures can be found within our fileReader.py file, this file handles the parsing of questions to obtain terms and bounds required for the generation.

fileReader.py	**Name**	**Description**	**Purpose**
Array	contentFile	A container that holds a fixed number of strings.	Temporarily stores the question input before it is processed.
Dictionary (also known as a hash)	dictOfVars	An associate array that holds any type of value mapped to a key. Dictionaries contain unordered key-value pairs.	Stores the terms within the question and any min/max bounds for the terms used within a generated question.

Algorithm design for key backend functions

Pseudocode for fileReader.py:

identifyVariables:

- takes: question (a string containing the question typed by the user)

- scans the question for both the visible variables and hidden constants
- returns: a dictionary containing the names of variables and constants as keys, the values of the constants and no values for the variables

```
define function identifyVariables(takes question)
        create empty list called variables
        create empty list called dictOfVars

        for each word separated by a space in question:
                if the word begins with "$":
                        set variableName as the string after the "$"
before a space
                        append variableName to variables
                        create empty entry in dictOfVars with key
variableName

        if there is a "{" in question
                assign the string between the "{" and the "}" to a
variable called hiddenConstantsString
                for each string separated by a "," in
hiddenConstantsString with whitespaces deleted
                        add whatever is on the left of the "=" as a
key to dictOfVars and whatever is on the right of the "=" as the
value in dictOfVars

        return dictOfVars
```

---

writeQuestion:

- takes: question (a string containing the question typed by the user) and dictVariables(a dictionary of the filled variables and constants filled by the user's code)
- Removes hidden constant definitions from the question string and substitutes the variable definitions for their values in the dictionary
- returns: Procedurally generated question string with the variables filled

```
define function writeQuestion(takes question, dictOfVariables)
        create empty list called returnString

        if there is a "{" in question
                remove the "{", the "}" and whatever is in between
from the question
```

```
        for each word separated by a space in question:
                if the word does not begin with "$":
                        append the word to the return string
                else:
                        for every variable in dictVariables
                                if the word after the "$" matches
the key of a variable:
                                        append the value in the
dictionary to returnString
        return returnString with a whitespace between every element
as a string
```

---

getQuestion:

- takes: question string
- serves as a wrapper and invokes the other functions in the file to generate the final question
- returns: The question with values instead of variables

```
define function getQuestion(takes questionInput)
        assign the output of the user's code for generating the
variables as varDict

assign the output of the user's code for generating the answers as
answers

        create variable question_complete that holds the output of
writeQuestion when passing questionInput and varDict

        return question_complete
```

---

exportToPDF:

- takes: text (string or text file) and outpath (the path of the output PDF file that will be saved)
- converts the text from latex into a pdf file and saves it at the defined path
- returns: nothing

```
define function exportToPDF(takes text and outpath)
        from latex library import buil_pdf function
        assign the return of build_pdf when passing the text
received to a variable called pdf

        save the pdf to the file path given in outpath
```

Pseudocode for example user code (addition.py)

It is important to note that this file is merely an example of how the user would write their code to make it compatible with the system in its current design and is not part of the core implementation.

add:

- takes: variable dictionary
- adds the values in the dictionary called term1 and term2
- returns: the sum

```
define function add (takes varDict)
        return element in key term1 + element in key term2 from
varDict
```

---

getAnswers:

- takes: integer
- Generates wrong answers based on the correct one
- returns: the list of answers in the test

```
getAnswers(takes n)
        returns (a tuple containing n and a series of possible wrong
answers based on n)
```

---

generateNumbers:

- takes: variable dictionary
- Generates terms for the empty values in the dictionary
- returns: filled variable dictionary

```
define function generateNumbers(takes varDict)
        for each key in varDict
                if there is no element in the key
```

```
                        if there is a minbound and maxbound in
varDict
                                create a random integer between
minbound and maxboundand put it in the dictionary with the key
                        if there is a minbound but no maxbound in
varDict
                                create a random integer between
minbound and 50 and put it in the dictionary with the key
                        if there is no minbound but there is a
maxbound in varDict
                                create a random integer between 1
and maxbound and put it in the dictionary with the key
                        if there is no minbound and no maxbound in
varDict
                                create a random integer between 1
and maxbound and put it in the dictionary with the key

        return varDict
```

---

generate:

- takes: variable dictionary

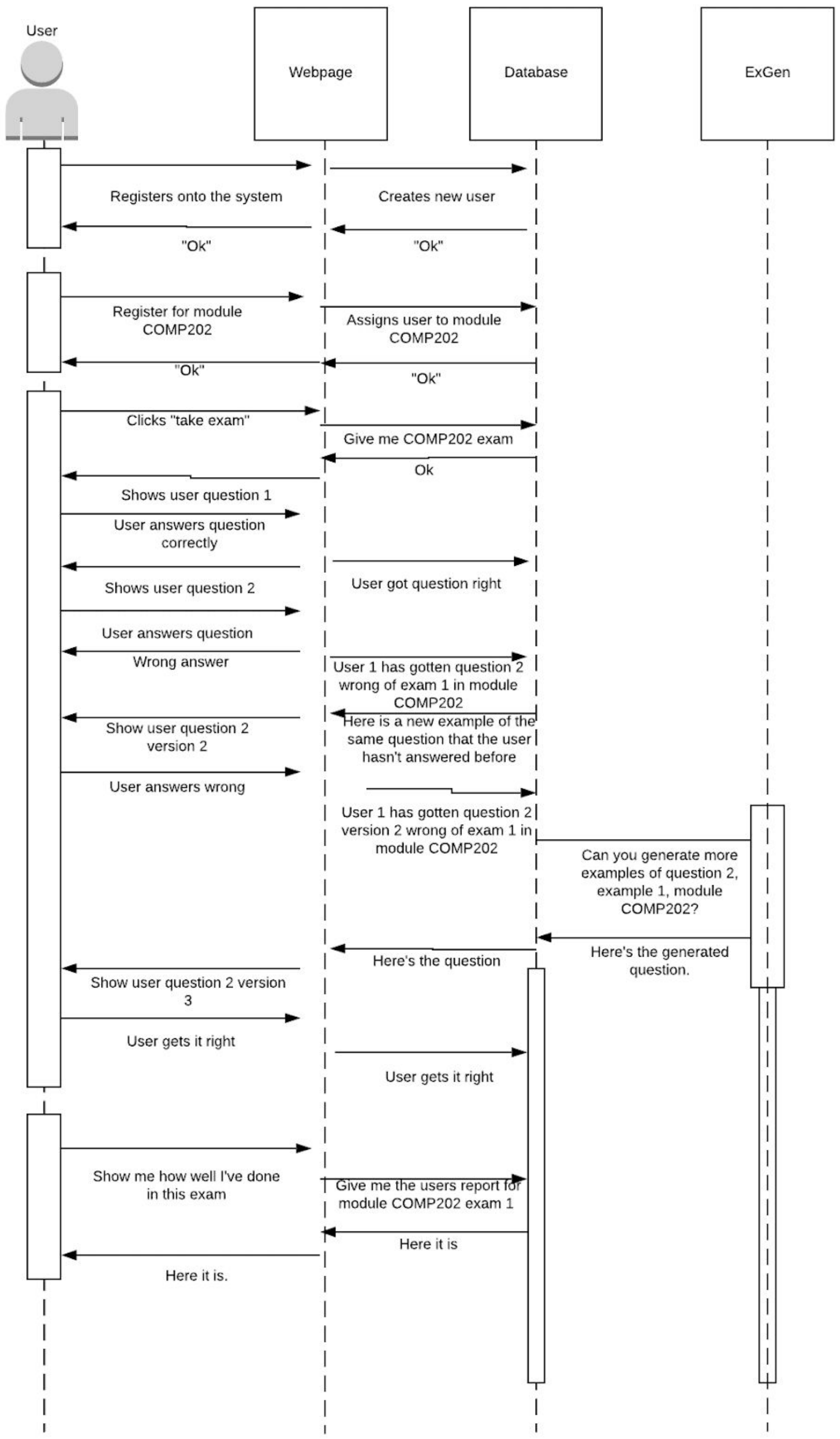
- A wrapper function that uses previously defined functions to generate the terms and the correct answer

- returns: a tuple containing a filled dictionary of variables and the answers

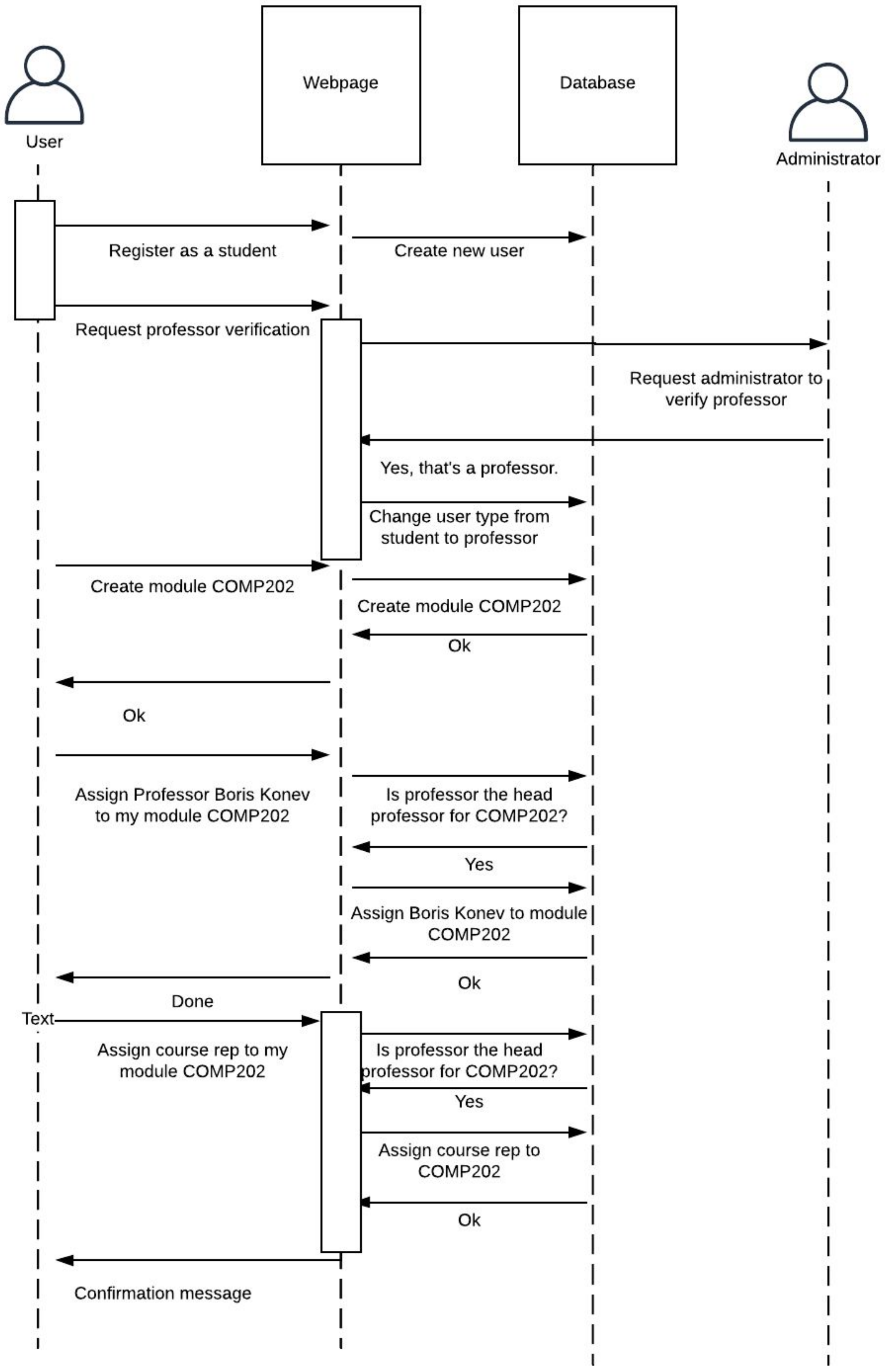
```
define function generate(takes varDict)
        update varDict using generateNumbers
        return a tuple containing (the updated varDict, the output
of getAnswers when passing the output of add when passing varDict)
```

Sequence Diagrams

This is the state charts for a user registering onto the system, registering for a module and taking an exam. Note, these charts are large so they start on the next page.


User
Webpage
Database
ExGen
Registers onto the system
Creates new user
"Ok"
"Ok"
Register for module COMP202
Assigns user to module COMP202
"Ok"
"Ok"
Clicks "take exam"
Give me COMP202 exam
Ok
Shows user question 1
User answers question correctly
Shows user question 2
User got question right
User answers question
Wrong answer
User 1 has gotten question 2 wrong of exam 1 in module COMP202
Here is a new example of the same question that the user hasn't answered before
Show user question 2 version 2
User answers wrong
User 1 has gotten question 2 version 2 wrong of exam 1 in module COMP202
Can you generate more examples of question 2, example 1, module COMP202?
Here's the generated question.
Here's the question
Show user question 2 version 3
User gets it right
User gets it right
Show me how well I've done in this exam
Give me the users report for module COMP202 exam 1
Here it is
Here it is.

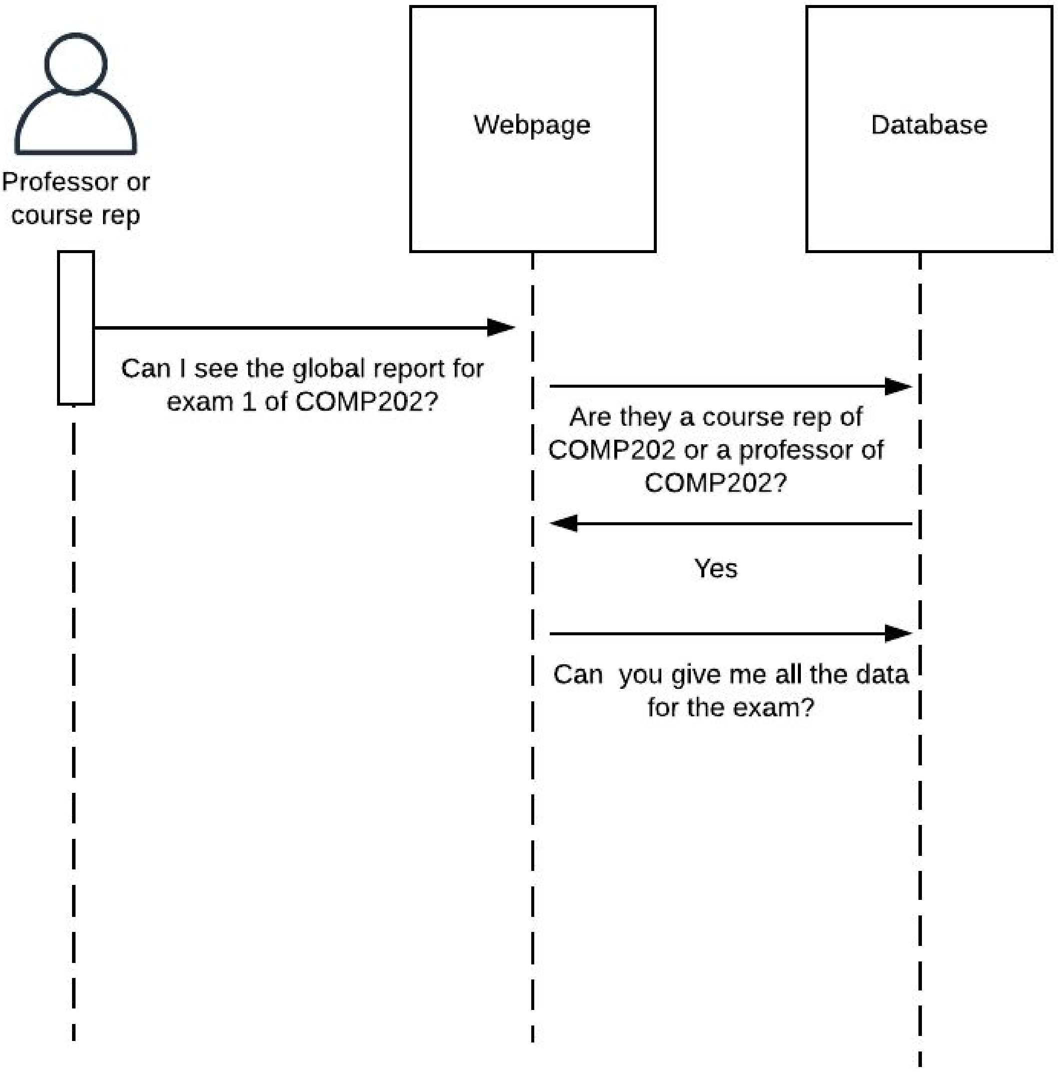
In reality one might use Liverpool life to verify users and assign them to modules. We are not using it as we want to make this software as easy as possible for anyone to implement.


User
Webpage
Database
Administrator
Register as a student
Create new user
Request professor verification
Request administrator to verify professor
Yes, that's a professor.
Change user type from student to professor
Create module COMP202
Create module COMP202
Ok
Ok
Assign Professor Boris Konev to my module COMP202
Is professor the head professor for COMP202?
Yes
Assign Boris Konev to module COMP202
Ok
Done
Text
Assign course rep to my module COMP202
Is professor the head professor for COMP202?
Yes
Assign course rep to COMP202
Ok
Confirmation message

Above is a professor:

- Registering for the system
- Confirming they are a professor
- Making a module
- Assigning a professor to that module
- Assigning a course rep to that module

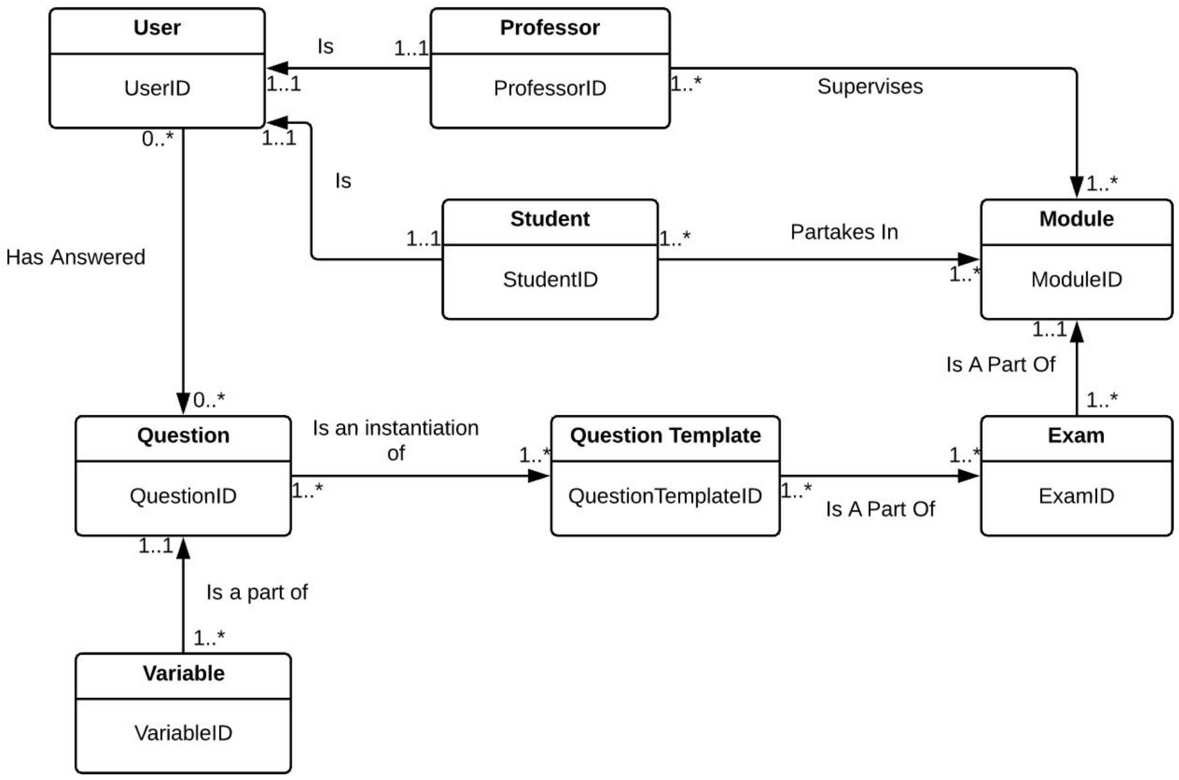
If the student has been verified as course rep (by a professor) then any modules they are assigned to they will automatically become course reps.


Professor or course rep
Webpage
Database
Can I see the global report for exam 1 of COMP202?
Are they a course rep of COMP202 or a professor of COMP202?
Yes
Can you give me all the data for the exam?

Here are course reps requesting to see the global report of a module. A professor can do the same. Course reps are normally assigned by professors, as talked about in the requirements document.

Entity-Relationship Diagram

Below is the design of out database, without any connector tables. These tables are used in many to many contexts but do not contain a primary key, only connects foreign keys together. Therefore they do not belong in the global ERD



Logical Table Structure

These tables include all the attributes of each table and the connector tables for the many to many relations

User(UserID, UserName, Hash, Salt) **Primary Key** UserID	**Student**(StudentID, UserID, isCourseRep) **Primary Key** StudentID **Foreign Key** UserID
Professor(ProfessorID, ProfessorName, ProfessorDescription, UserID) **Primary Key** ProfessorID **Foreign Key** UserID **References** User(UserID)	**Exam**(ExamID, ModuleID, Title, Description, Enabled) **Primary Key** ExamID **Foreign Key** ModuleID **References** Module(ModuleID)
QuestionTemplate(QuestionTemplateID, LaTex, SolutionCode, Enabled) **Primary Key** QuestionTemplateID	**CourseModule**(ModuleID, ModuleName, ModuleDescription, ModuleCode) **Primary Key** ModuleID

Question(QuestionID, QuestionTemplateID) **Primary Key** QuestionID **Foreign Key** QuestionTemplateID **References** QuestionTemplate(QuestionTemplateID)	**Variable**(VariableID, VariableName, VariableValue, QuestionID) **Primary Key** VariableID **Foreign Key** QuestionID **References** Question(QuestionID)

Additional connecting tables for many to many relations

StudentModule(StudentID, ModuleID) **Foreign Key** StudentID **References** Student(StudentID) **Foreign Key** ModuleID **References** Module(ModuleID)	**ProfessorModule**(ProfessorID, ModuleID, HeadProfessor) **Foreign Key** ProfessorID **References** Professor(ProfessorID) **Foreign Key** ModuleID **References** Module(ModuleID)
ExamQuestion(ExamID, QuestionID) **Foreign Key** ExamID **References** Exam(ExamID) **Foreign Key** QuestionID **References** Question(QuestionID)	**AnsweredQuestion**(QuestionID, UserID, Correct) **Foreign Key** UserID **References** User(UserID) **Foreign Key** QuestionID **References** Question(QuestionID)

Transaction Matrix

Below is a list of transactions that the system will perform and a transaction matrix to satisfy them.

a) Creation of a student account
b) Creation of new questions, once the pre rendered ones have been exhausted by a student
c) A student deletes their account
d) A student completes a question
e) A student requests their own statistics for a module
f) A student queries for a new exam question to complete
g) A Course Representative requests an overview of the statistics of an exam
h) A professor adds new questions to an exam
i) A professor appoints a Course Representative
j) A professor updates the question code and latec of a question
k) A professor disables or enables a question
l) A professor disables or enables an entire exam
m) A professor deletes a student from their course
n) A professor rejects a Course Representative
o) A professor queries their own questions
p) A professor requests the overall statistics of a module/exam
q) An administrator adds a new professor
r) An administrator deletes a professor

s) An administrator deletes a module

Transaction/ Table	(a)				(b)				(c)				(d)				(e)				(f)				(g)				(h)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
User	X					X						X		X																		
AnsweredQuestion					X							X	X								X											
Professor																														X		
ProfessorModule																														X		
Student	X											X						X				X				X						
StudentModule																		X				X				X						
Module																		X				X				X				X		
Exam																		X				X				X				X		
ExamQuestion																		X				X				X				X		
QuestionTemplate						X												X				X				X			X			
Question					X													X				X				X						
Variable					X																	X										

Transaction/ Table	(i)				(j)				(k)				(l)				(m)				(n)				(o)				(p)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
User																																
AnsweredQuestion																														X		
Professor		X				X				X				X				X				X				X				X		
ProfessorModule		X				X				X				X				X				X				X				X		
Student			X																													
StudentModule		X																		X				X								
Module		X				X				X				X				X				X				X				X		
Exam						X				X					X											X				X		
ExamQuestion						X				X																X				X		
QuestionTemplate							X				X															X				X		
Question																										X				X		
Variable																										X						

Transaction/ Table	(q)				(r)				(s)			
	I	R	U	D	I	R	U	D	I	R	U	D
User	X							X				
AnsweredQuestion								X				X
Professor	X							X				
ProfessorModule								X			X	
Student												
StudentModule											X	
Module												X
Exam												X
ExamQuestion												X
QuestionTemplate												X
Question												X
Variable												X