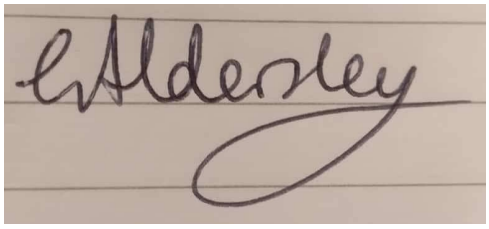
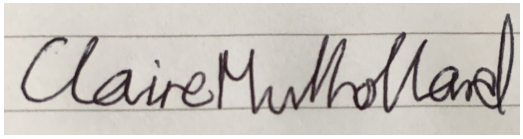
ExGen – Next Generation Exams

Group 11

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

UNIVERSITY OF LIVERPOOL

GROUP PROJECT DECLARATION OF ACADEMIC INTEGRITY

NAME (Print)	LUKE ALDERSLEY
STUDENT NUMBER	201242265
GROUP NAME	Team 11
MODULE TITLE/CODE	COMP208
TITLE OF WORK	ExGen

This form should be completed by one student on behalf of the group and appended to any piece of work that is submitted for summative assessment. Submission of the form by electronic means by a student constitutes their confirmation of the terms of the declaration.

Students should familiarise themselves with Section 9 of the Code of Practice on Assessment and Appendix L of the University's Code of Practice on Assessment which provide the definitions of academic malpractice and the policies and procedures that apply to the investigation of alleged incidents.

Students found to have committed academic malpractice are liable to receive a mark of zero for the assessment or the module concerned. Unfair and dishonest academic practice will attract more severe penalties, including possible suspension or termination of studies.

STUDENT DECLARATION

I confirm that the group has read and understood the University's Academic Integrity Policy.

I confirm that the group has acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.

I confirm that no member of the group has copied material from another source nor committed plagiarism nor fabricated data when completing the attached piece of work.

I confirm that no member of the group has previously presented the work or part thereof for assessment for another University of Liverpool module.

I confirm that no member of the group has copied material from another source, nor colluded with any other student in the preparation and production of this work.

I confirm that no member of the group has incorporated into this assignment material that has been submitted by any other person in support of a successful application for a degree of this or any other University or degree awarding body.

SIGNATURE.....LAldersley.....

DATE.....08.05.19.....

Portfolio

Team Members

Brandon:

- Cloudflare - I set up our DNS and CDN.
- Backend - I worked on ExGen Core (the backend).
- Accessibility - I made sure the frontend was accessible.
- Web Developer - I worked on the frontend, but I mainly worked on connecting the frontend to the backend.
- Security - Making sure the app is as secure as can be. Penetration testing our app and testing it for any exploits.
- GitHub management - Making sure all issues are okay and followed up on. Reviewing code and merge requests.
- Some design stuff, such as the initial logo

Luke:

- Interface design - Done through figma concepts.
- Web developer - Development of www.exgen.xyz alongside Steffan.
- Use cases - Creation of use case diagrams and descriptions with Yales.
- Leading demonstrations - Responsible for demonstrating the final project as well as visual concepts.
- Document formatting - Checked that any documents were formatted correctly with consistent fonts and spacing.
- Data structure design with Geng

Yales:

- Backend Algorithm Programming
- Backend Algorithm Design
- Backend Algorithm Demonstrator
- Version Control (GitHub) management
- Use case diagrams
- General Documentation
- Document editing

Steffan:

- Interface design - Done through Figma concepts.
- Web developer - Development of www.exgen.xyz alongside Luke.
- Created any assets needed such as the final logo.
- Backend/Server setup and maintenance.
- Flask setup and maintenance.
- Implementing the database with our code.

Geng:

- Project tasks - identified tasks that needed to be done for the project to be completed.
- Risk assessment - explored possible risks and possible solutions/mitigations for those risks.
- Gantt chart - produced a outline of what tacks needed to be done, in what order and who should do then.
- Data structure design - designed the outline of the data structures used alongside Luke.

Claire:

- Database Design - completed through mySQL workbench and lucidCharts
- Database Model building - with SQLAlchemy
- Database Query and Transaction building - with SQLAlchemy

What is ExGen?

ExGen is a tool for procedurally generating exams. Given a question such as:

> *"What is $X + $Y?"*

ExGen generates the questions:

> *"What is 6 + 3?"*
> *"What is 7 + 9?"*

As well as generating the answers for these questions.

Who would use ExGen?

ExGen is designed to be used by universities. It has 2 use cases.

- Professors

Professors (or demonstrators) will use ExGen to automatically generate class tests for their students (or mock papers). By creating one ExGen paper they will be able to generate an infinite amount of papers.

Professors will use ExGen because in the long run, it saves them time.

- Students

Students will use ExGen to learn. They have an opportunity to practice an infinite amount of times.

Students will use ExGen because it provides an infinite source to practice.

To what extent were the requirements met?

Complete features:

- Registering an account with ExGen (partial validation on password).
- Login system fully work - Different account types(student/professor etc) with different emails and hashed passwords. Logging out also worked.
- Database was fully implemented for all the features we had.
- The UI of the site was fully implemented.
- Opting in and out of modules as a student and viewing visible exams for that module.
- Creating/deleting modules as a professor.
- Taking an exam (only addition questions working).
- (Backend) Template question parsing (#2)
- (Backend) Question generation (#2)
- (Backend) Answer generation (#18)
- (Backend) Adding hidden constants as well as variables to template questions (#4)

Incomplete features:

- Full validation on registration (email verification).
- More advanced exam questions (more than addition).
- Settings pages - Resetting password and requesting verification

- Exam results - Although we did store user answers, the results pages we're not implemented
- Admin page - The admin page was not implemented. All actions performed by admins were performed on the actual server itself and not through the website.
- Exam limit - There was no limit to the exam questions so the user could repeat the same question.
- Creating a basic exam as a professor for a module - This was almost complete, the code works but there is a permission issue with writing files that I did not manage to fix.
- Adding course reps to modules as a professor.
- Adding professors to modules as a professor.
- Course rep results page
- Professor results page
- (Backend) Visible constants (#8)
- (Backend) Let user escape special characters in their question templates (#14)

Strengths and Weaknesses of the project

Strengths

- Easy to use
- Generates infinite amount of exam resources
- Intuitive design
- Potential for further development and applications
- Saves professor's time in the long term.

Weaknesses

- The final program only had addition problems as an user-written code preset.
- There wasn't a way to save questions for later use.
- Creates more work for the professor in the short term.
- Limited potential for more literature based questions.

Future developments

In the future, we would like these features to be added:

Saving questions

The ability to save questions so they can be reused by the same professor - or another professor. We had this idea in the requirements document, but we didn't have time to complete this feature.

More than addition

ExGen current only handles addition problems as a code preset. It would be good to create more presets to show the potential of the underlying algorithm and save time for professors

More polished design

Although the overall visual aesthetic of our website was good, we received feedback that our website had a lot of wasted space, this could be filled with either more content or a better visual representation of what is already on the page.

Clear instructions for installation

Throughout the project we have been using github to manage our code and documentation. There needs to be clear instructions available on the github explaining how to install and implement what we have created as there is nothing of the like available at the moment.

API

It would of been more useful to the public if we created an API, so that anyone can query our API and generate answers to questions.

Mobile version of our website or even an app version

Our website www.exgen.xyz does work on both desktop and mobile but it isn't optimised well on mobile platforms with a smaller screen. It is very difficult to see the content on the web page and the user is required to zoom in. Further developments need to be made on improving how the website is viewed on a mobile device. Although some of the current issues regarding this would be addressed and solved when polishing the design such as filling the blank space, as this would remove the need for zooming in so much on mobile.

Codes of practice and conduct

Here are the ways we've made ExGen aligned with the BCS code of practice and BCS code of conduct.

- Maintain Your Technical Competence

Throughout this process we have all learnt new skills and worked together to build on past skills. Some of the skills we have learnt are:

1. Flask
2. Server management
3. Python
4. Version control software (Git, GitHub)
5. String parsing to extract information

- Adhere to Regulations

We adhered to GDPR & Data Protection Act Regulations. If a user wanted to delete their account, it is possible. All data is encrypted using the industry standard AES. All passwords are hashed with a salt.

- Act Professionally as a Specialist

Every member of the team was expected to complete the work assigned to them and regular meetings were held to get members on the same page

- Use Appropriate Methods and Tools

We used github to share our code effectively with each other as we worked simultaneously. We used varying IDEs and version control to help us with this. We used Python as our programming language and Flask (a popular web development framework built on Python) for our website.

- Manage Your Workload Efficiently

We used project planning tools such as KANBAN or Agile to properly manage the workload amongst us, as a team.

- Participate Maturely

We worked productively and professionally, keeping our gaze set on what we wanted to achieve. Communication between team members at work times was professional, clear and effective.

- Respect the Interests of your Customers

Our product is made with the customers in mind, and to make students and professors' lives easier. We never did anything they wouldn't like, such as compromising their privacy.

- Promote Good Practices within the Organisation

As an organisation we worked collaboratively and tried to balance the workload between different individuals to avoid overworking of any one party. We also used PEP8 - the Python style guide for all of our code in order to make it readable for anyone in the team.

- Represent the Profession to the Public

We worked in the open. Allowing anyone to see and study what we do and what we've made. We hope that we've represented the profession to not just other programmers but also people interested in learning to program.

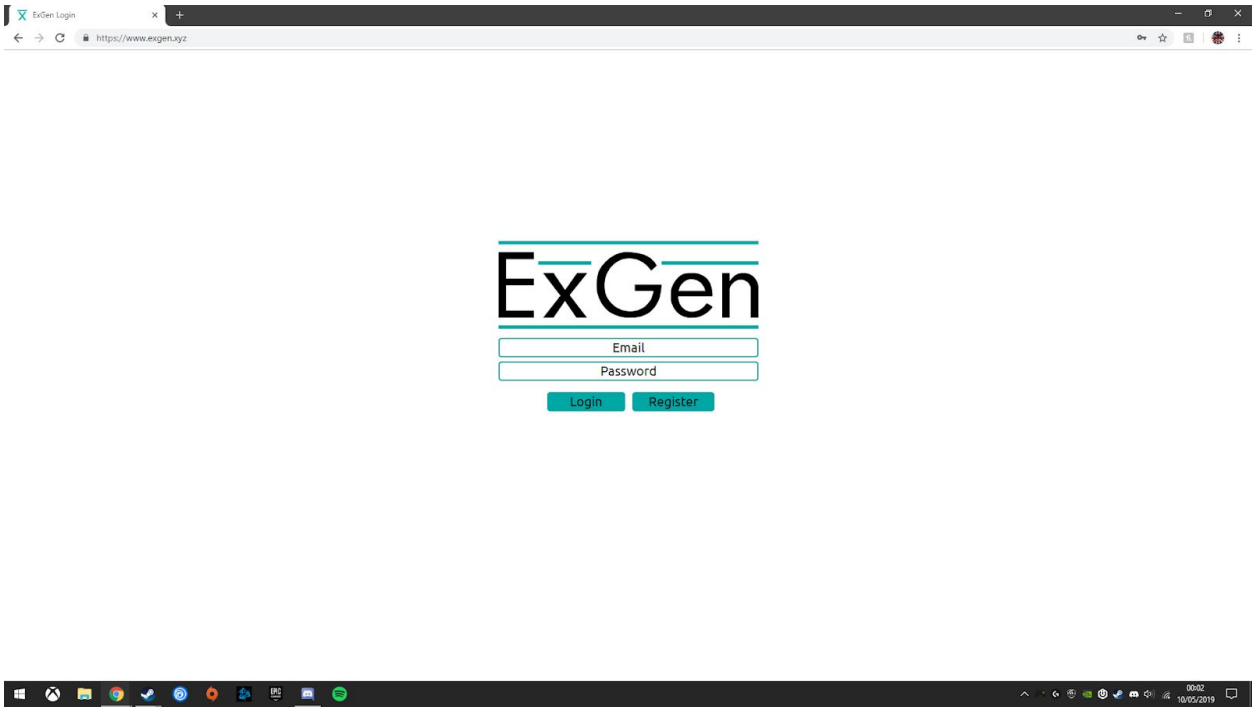
There are over 36 pages of the BCS Code of Practice (Bcs.org, 2019). We're not going to list them all here. We've been using the Code of Practice as a guidebook. For instance, the following phrase:

"Where risk is created by virtue of the scale or novelty of a solution for which there is no reliable benchmark for estimation, consider a modular or incremental approach to reduce risk."

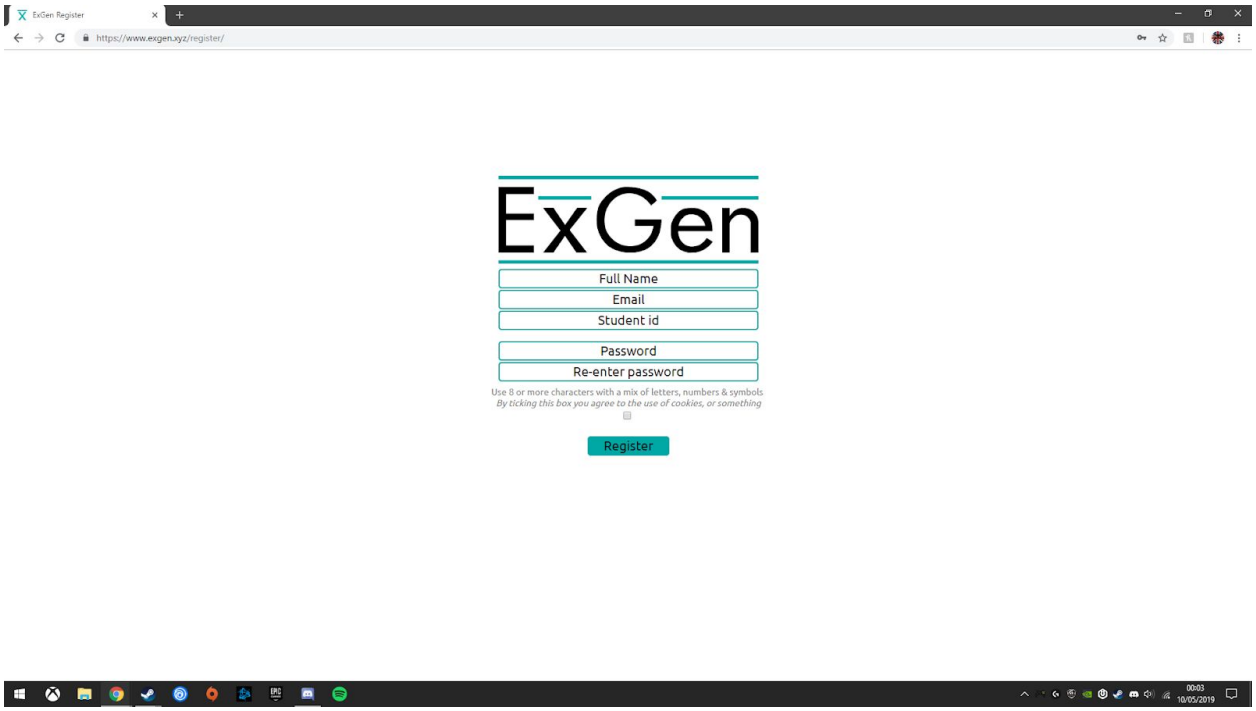
Served as a good guide for us when dealing with risk.

The finished product www.exgen.xyz:

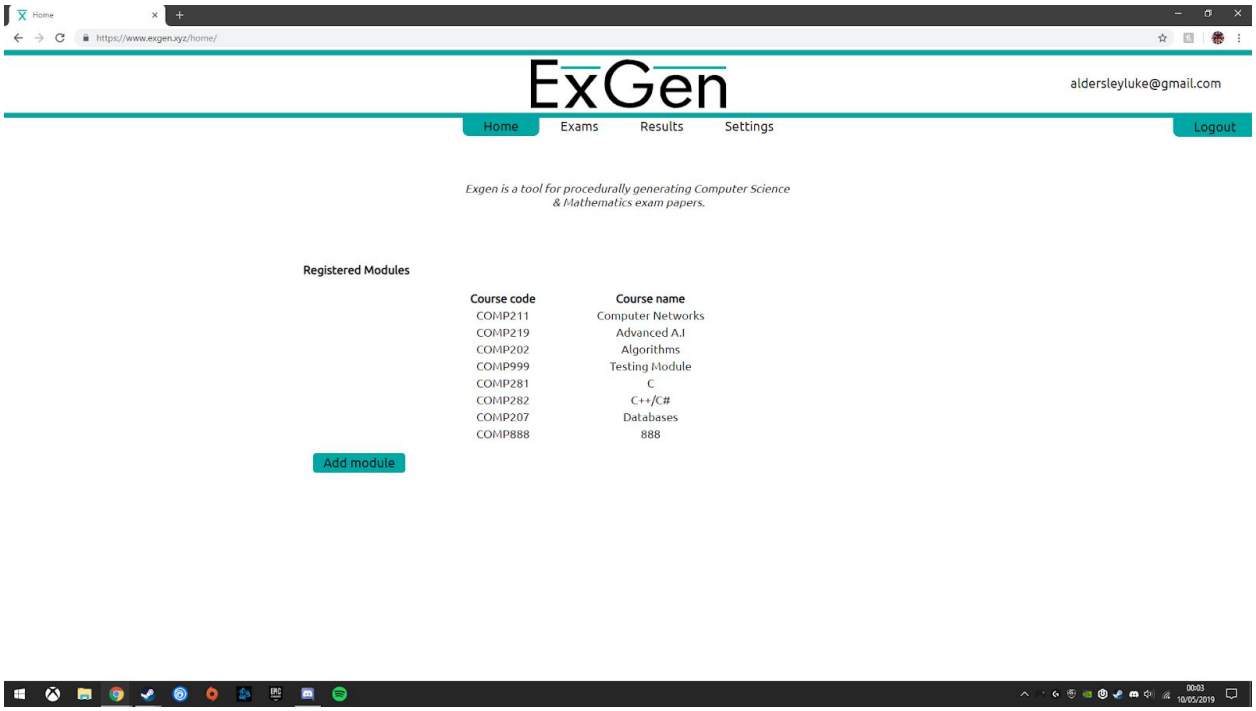
Login page



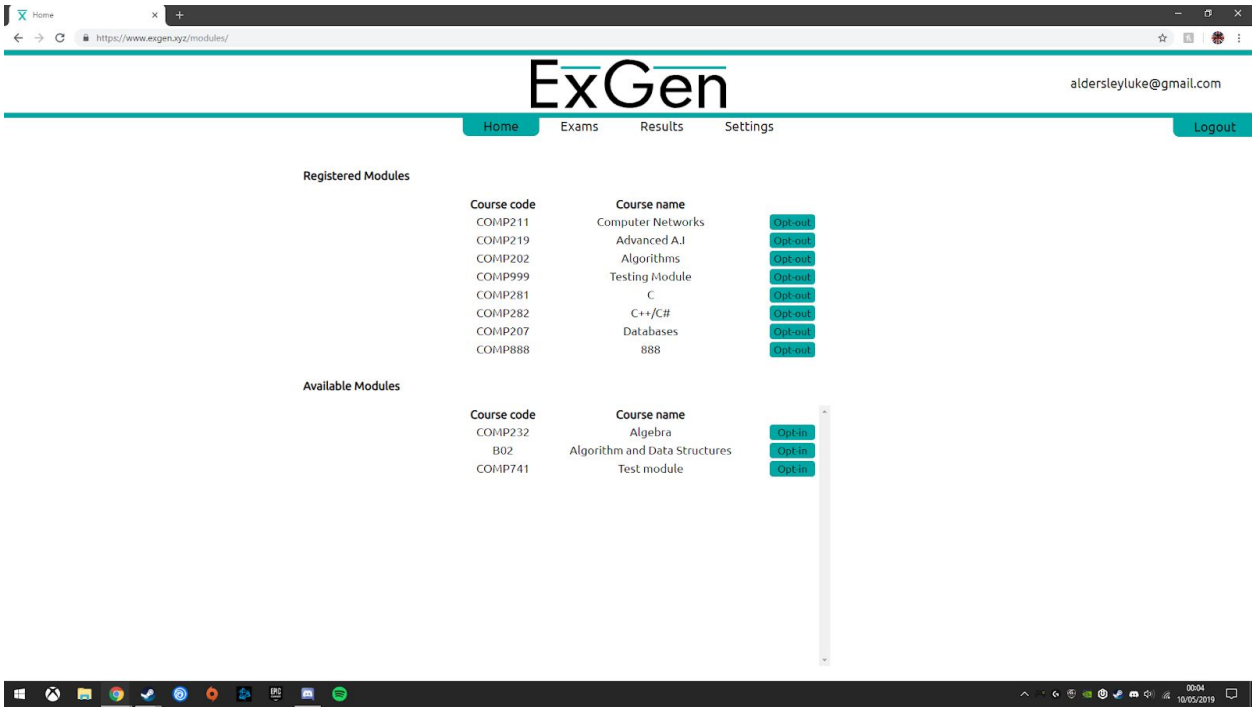
Registering



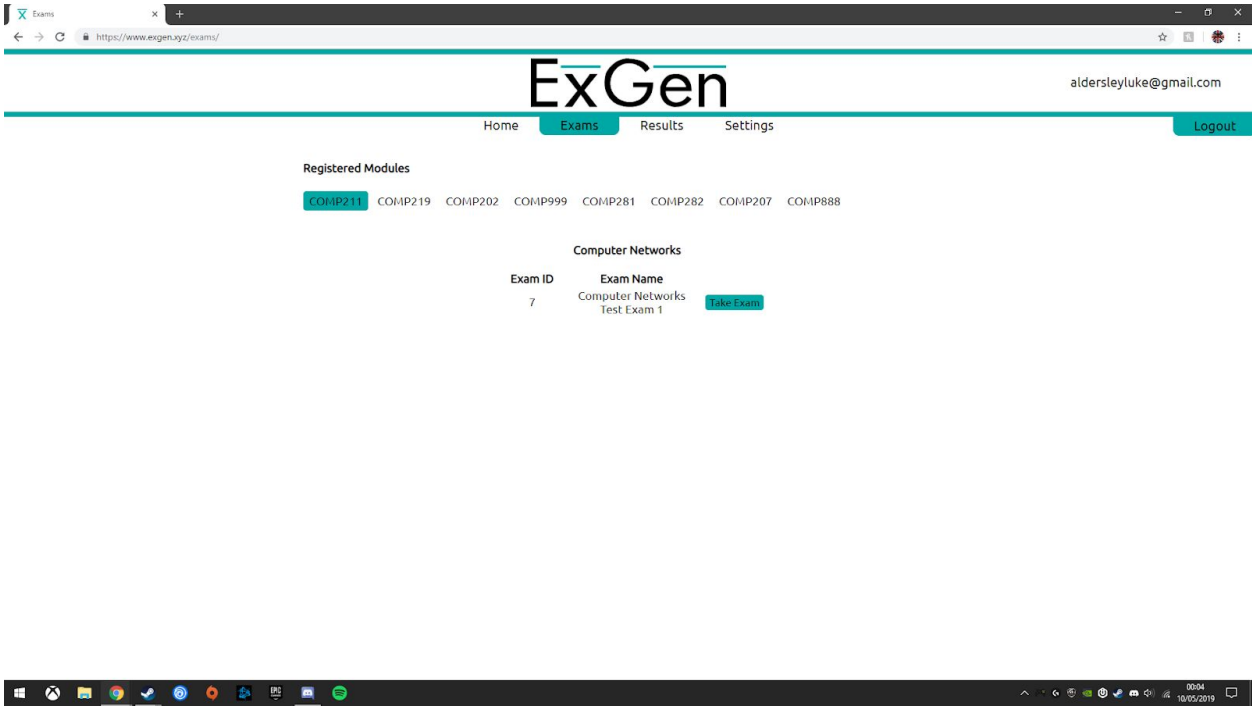
Student home page



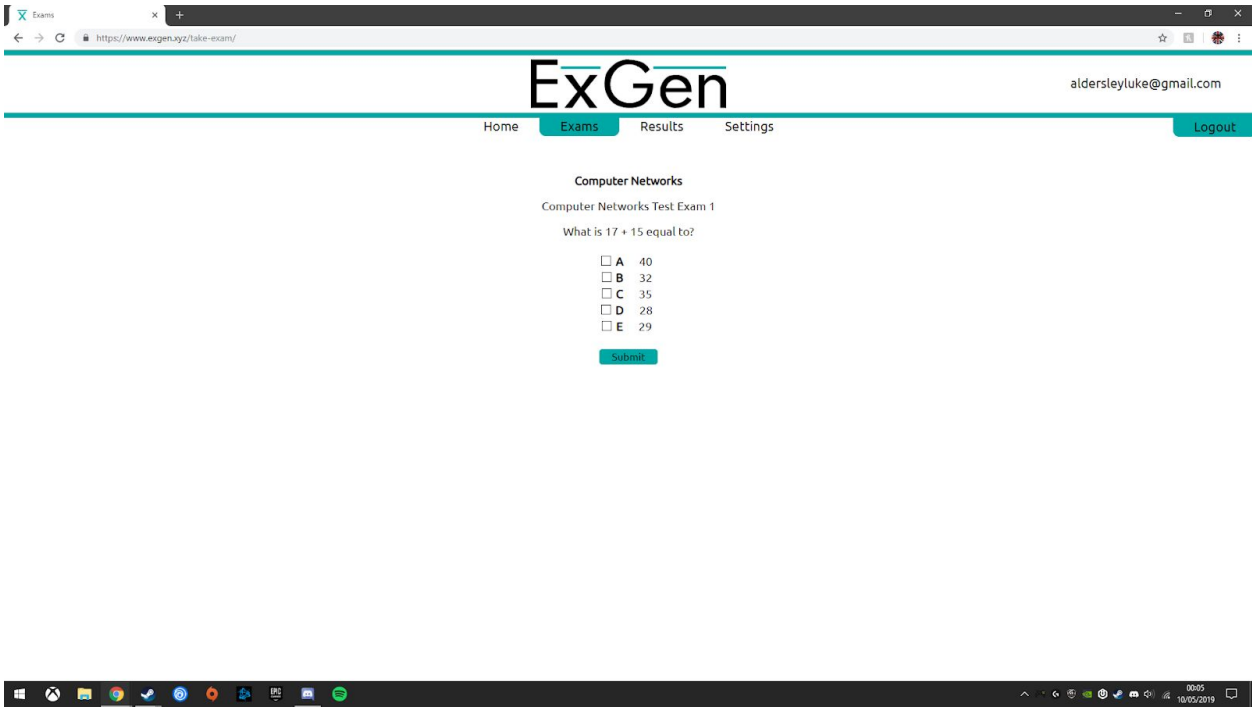
Opting in and out of modules



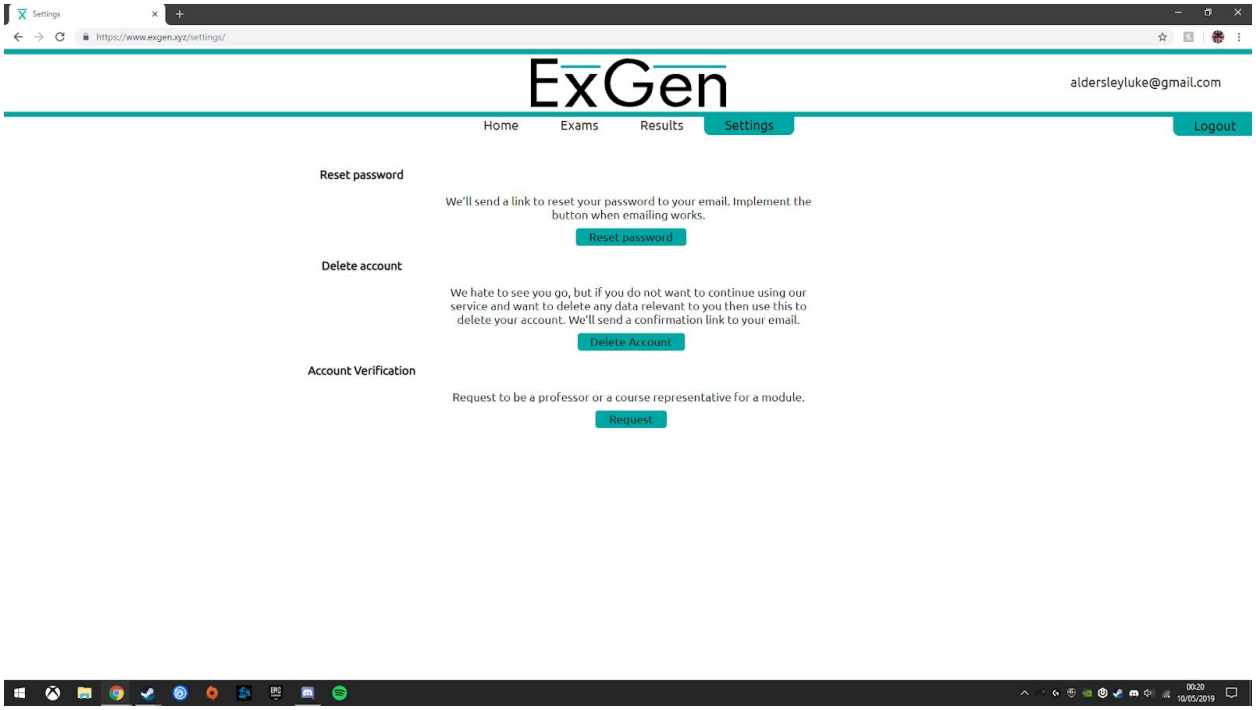
Exam page



Taking an exam

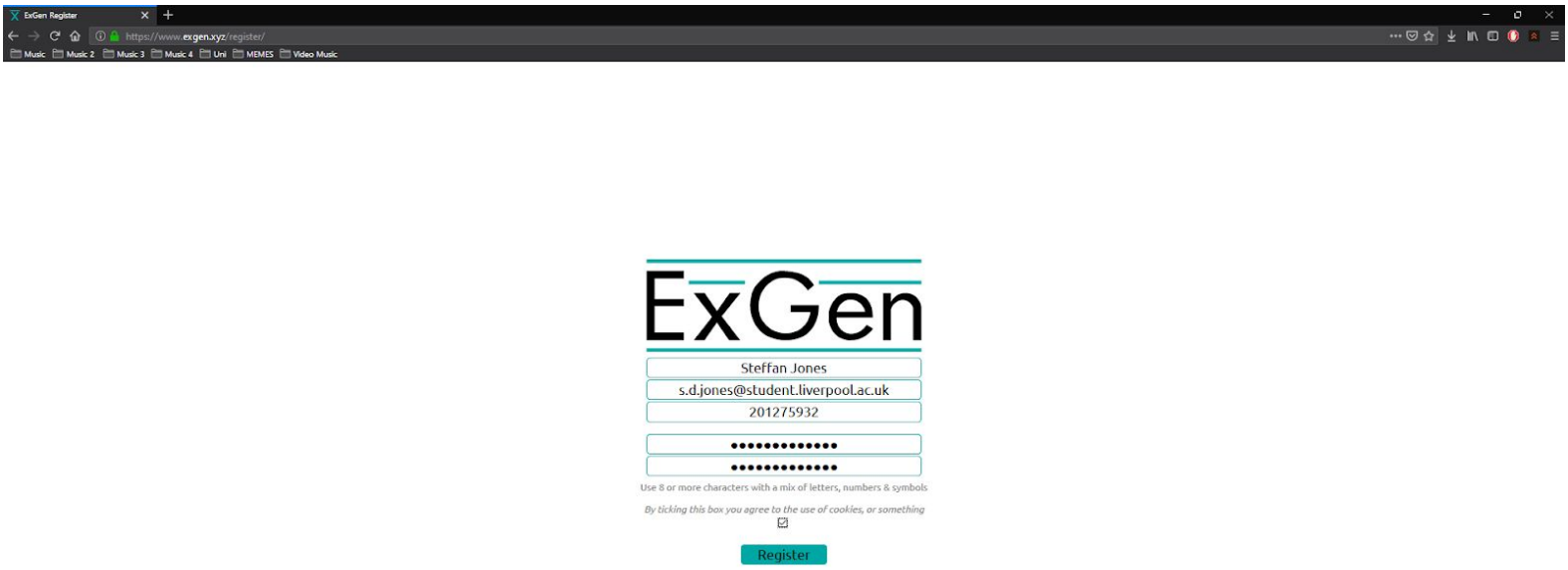


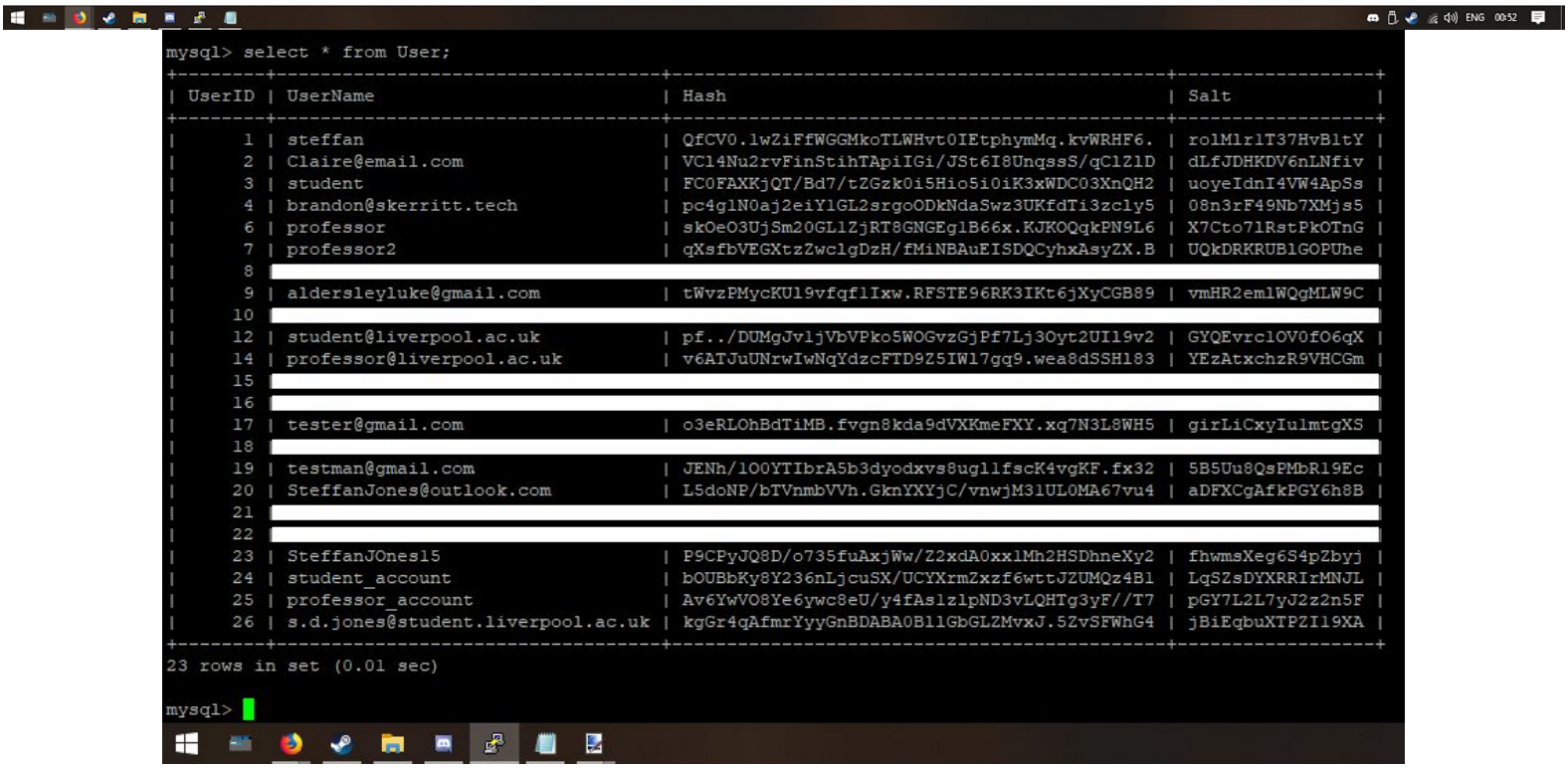
Settings


Settings
https://www.exgen.xyz/settings/
ExGen
aldersleyluke@gmail.com
Home
Exams
Results
Settings
Logout
Reset password
We'll send a link to reset your password to your email. Implement the button when emailing works.
Reset password
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.
Delete Account
Account Verification
Request to be a professor or a course representative for a module.
Request

Testing

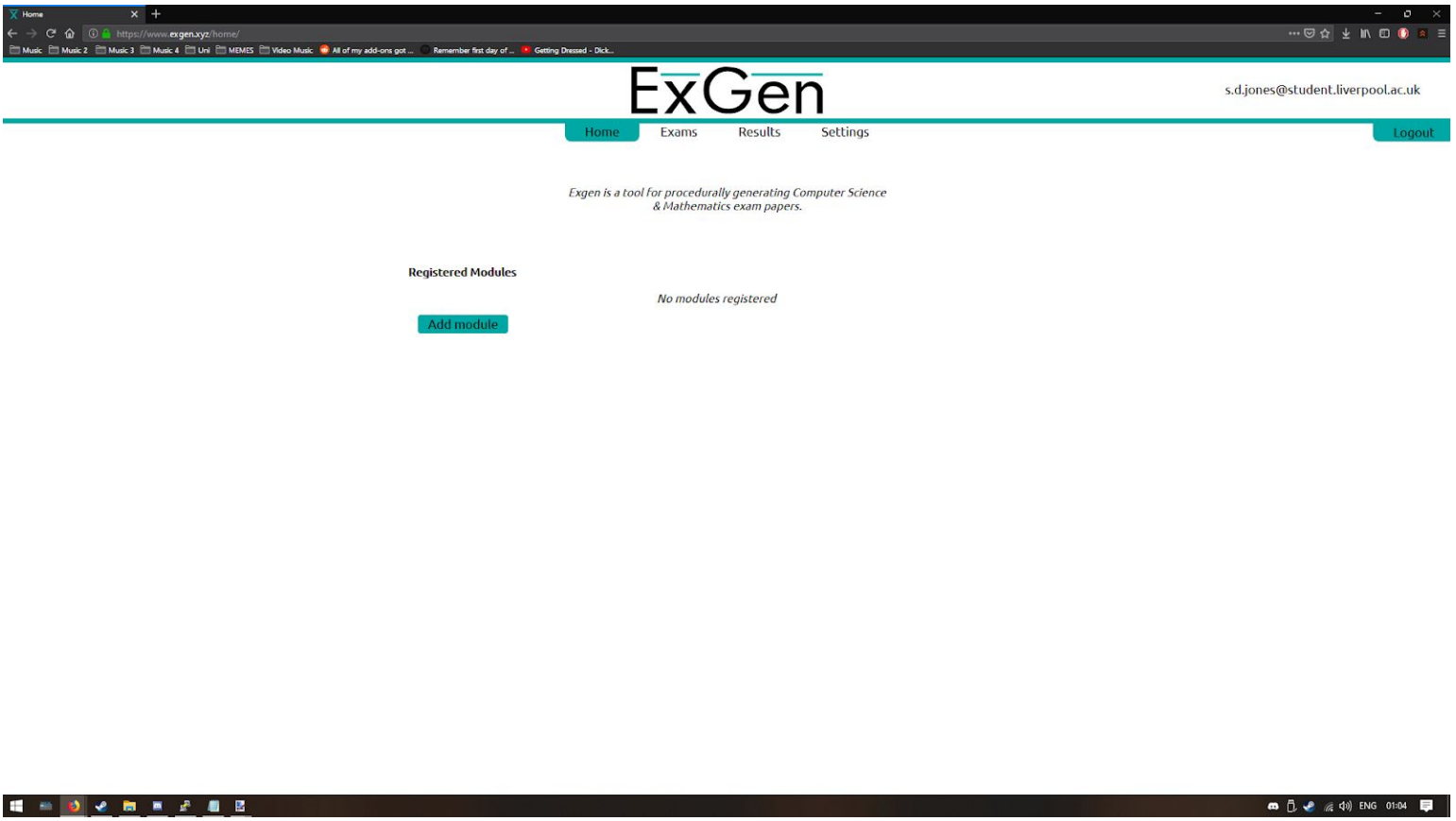
Registering an account





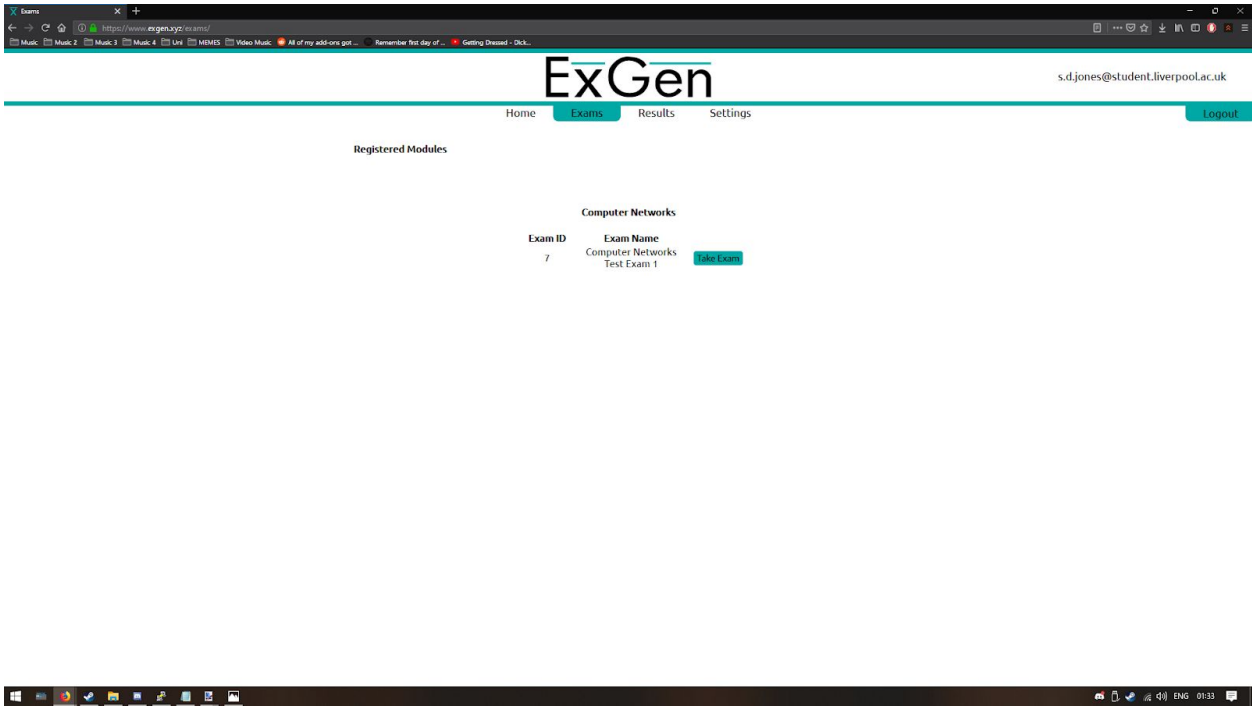
```
mysql> select * from User;
+--------+----------------------------------+--------------------------------------------+------------------+
| UserID | UserName                         | Hash                                       | Salt             |
+--------+----------------------------------+--------------------------------------------+------------------+
|      1 | steffan                          | QfCV0.lwZiFfWGGMkoTLWHvt0IEtphymMq.kvWRHF6. | rolMlrlT37HvBltY |
|      2 | Claire@email.com                 | VC14Nu2rvFinStihTApiIGi/JSt6I8UnqssS/qClZlD | dLfJDHKDV6nLNfiv |
|      3 | student                          | FC0FAXKjQT/Bd7/tZGzk0i5Hio5i0iK3xWDC03XnQH2 | uoyeIdnI4VW4ApSs |
|      4 | brandon@skerritt.tech            | pc4glN0aj2eiYlGL2srgoODkNdaSwz3UKfdTi3zcly5 | 08n3rF49Nb7XMjs5 |
|      6 | professor                        | skOeO3UjSm20GLlZjRT8GNGEg1B66x.KJKOQqkPN9L6 | X7Cto7lRstPkOTnG |
|      7 | professor2                       | qXsfbVEGXtzZwclgDzH/fMiNBAuEISDQCyhxAsyZX.B | UQkDRKRUBlGOPUhe |
|      8 |                                  |                                            |                  |
|      9 | aldersleyluke@gmail.com          | tWvzPMycKUl9vfqflIxw.RFSTE96RK3IKt6jXyCGB89 | vmHR2emlWQgMLW9C |
|     10 |                                  |                                            |                  |
|     12 | student@liverpool.ac.uk          | pf../DUMgJvljVbVPko5WOGvzGjPf7Lj3Oyt2UIl9v2 | GYQEvrclOV0fO6qX |
|     14 | professor@liverpool.ac.uk        | v6ATJuUNrwIwNqYdzcFTD9Z5IW17gq9.wea8dSSHl83 | YEzAtxchzR9VHCGm |
|     15 |                                  |                                            |                  |
|     16 |                                  |                                            |                  |
|     17 | tester@gmail.com                 | o3eRLOhBdTiMB.fvgn8kda9dVXKmeFXY.xq7N3L8WH5 | girLiCxyIulmtgXS |
|     18 |                                  |                                            |                  |
|     19 | testman@gmail.com                | JENh/lOOYTIbrA5b3dyodxvs8ugllfscK4vgKF.fx32 | 5B5Uu8QsPMbR19Ec |
|     20 | SteffanJones@outlook.com         | L5doNP/bTVnmbVVh.GknYXYjC/vnwjM31UL0MA67vu4 | aDFXCgAfkPGY6h8B |
|     21 |                                  |                                            |                  |
|     22 |                                  |                                            |                  |
|     23 | SteffanJOnes15                   | P9CPyJQ8D/o735fuAxjWw/Z2xdA0xxlMh2HSDhneXy2 | fhwmsXeg6S4pZbyj |
|     24 | student_account                  | bOUBbKy8Y236nLjcuSX/UCYXrmZxzf6wttJZUMQz4Bl | LqSZsDYXRRIrMNJL |
|     25 | professor_account                | Av6YwVO8Ye6ywc8eU/y4fAslzlpND3vLQHTg3yF//T7 | pGY7L2L7yJ2z2n5F |
|     26 | s.d.jones@student.liverpool.ac.uk | kgGr4qAfmrYyyGnBDABA0BllGbGLZMvxJ.5ZvSFWhG4 | jBiEqbuXTPZI19XA |
+--------+----------------------------------+--------------------------------------------+------------------+
23 rows in set (0.01 sec)

mysql>
```

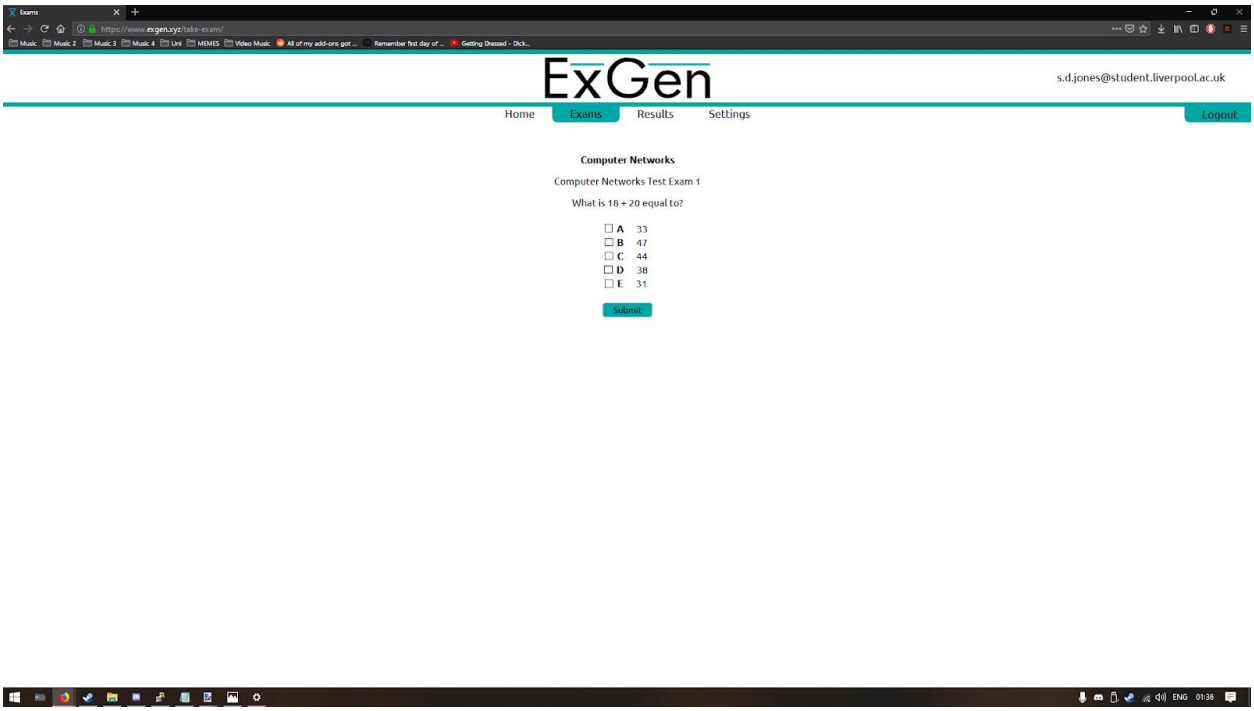


So after registering an account we can see that it is added into the database. We've had to blur some of these entries as they are real users. After registering, I am now able to login onto the system as a student.

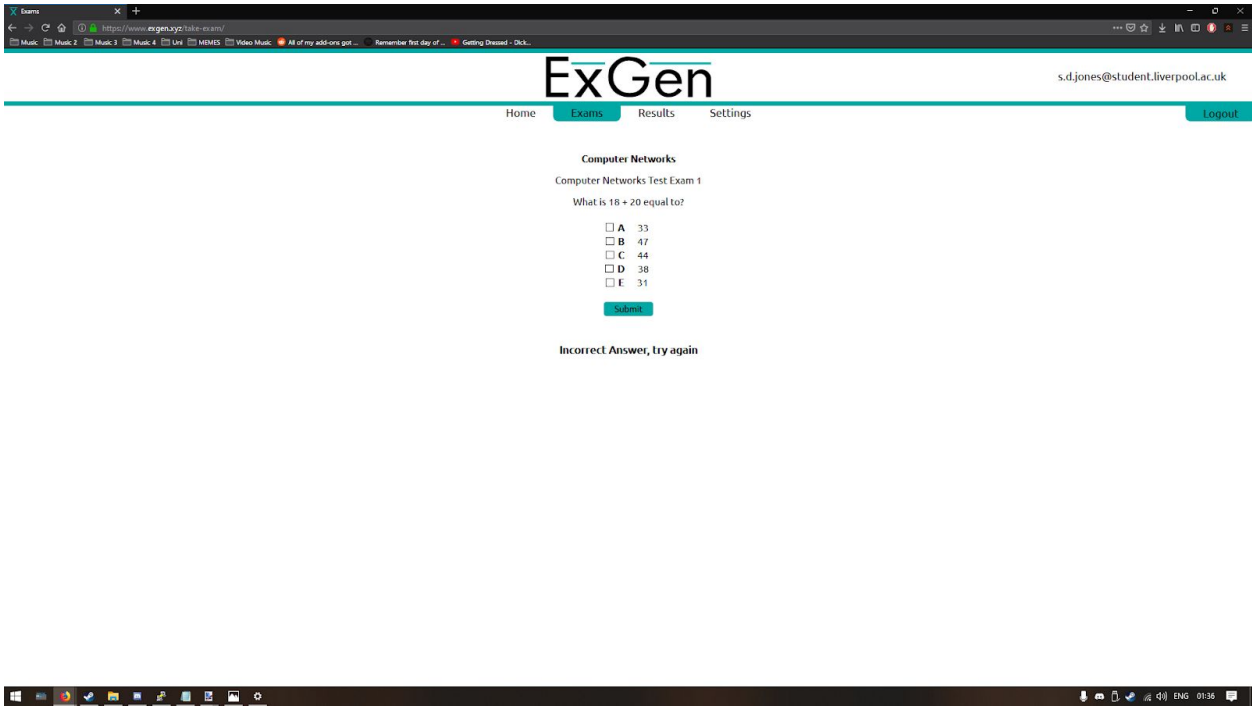
Taking an exam



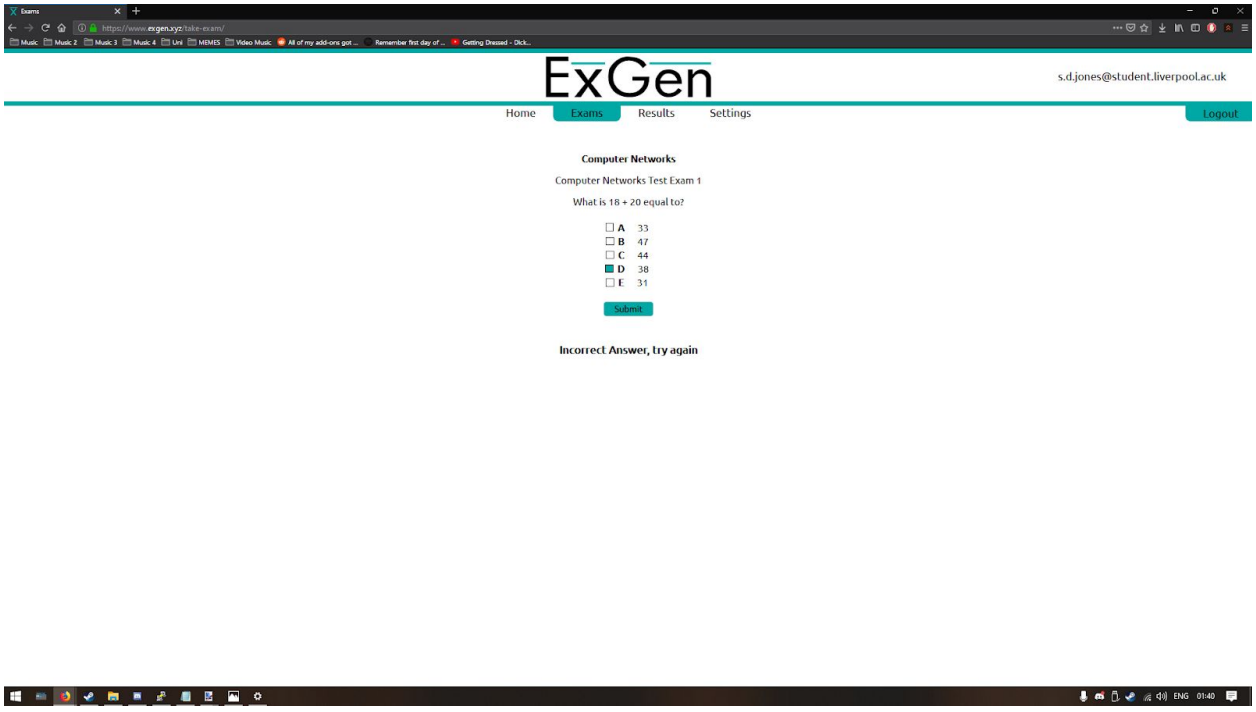
When clicking “Take exam” I’m presented with a new page.



This brings up the exam. The user can now answer as many questions as he wants. When choosing an incorrect answer the following shows up.



And the same question shows up again. When the user answers correctly a new question shows up. This then repeats until the user has had enough. The finished product would've had a question limit on the exam.



After entering the correct answer:

Requirements

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.

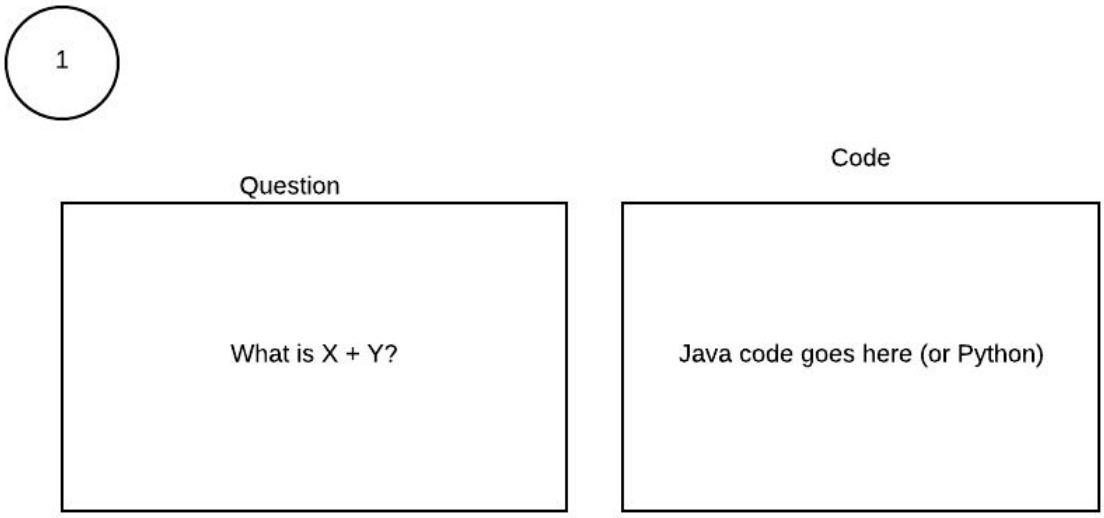
ExGen is a program to procedurally generate exams given an exam. Instead of giving your students one mock paper, give them ExGen which will automatically generate an infinite number of papers for your students to practice on.

The tech

How the technology works depends on who is using the system. There are 2 core users (3, but course reps are a type of student)

Professor

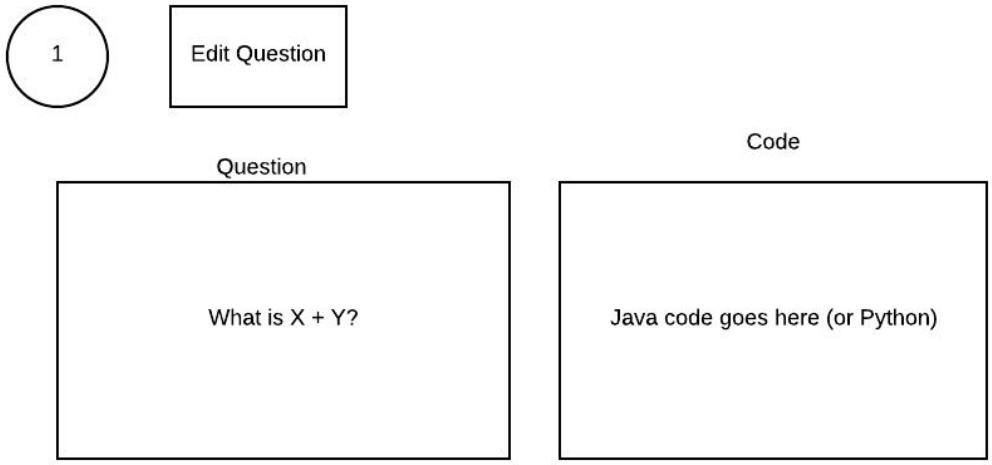
The professor uploads the exam question by question, roughly speaking this is how it'll look:



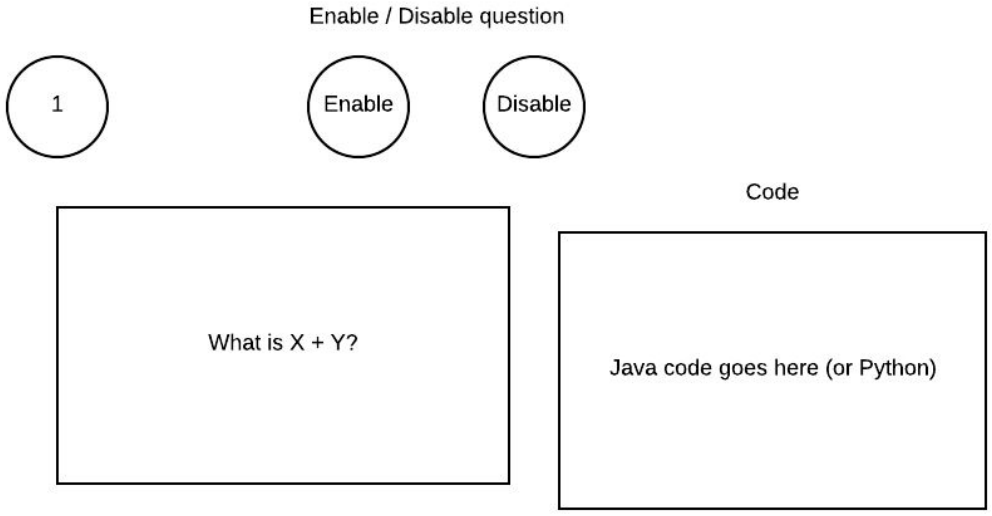
[1] https://www.un.org/sustainabledevelopment/sustainable-development-goals/
[2] https://www.un.org/sustainabledevelopment/education/

The question is LaTeX, the code is Python or Java.
The professor will create questions in the system manually, preferably in LaTeX. The professor will tell the system “I want 20 questions” and the system will loop 20 times asking for the LaTeX for that question and the code required to solve it.

When the professor views the question, the question will have an “edit” button to let the professor edit the question.



Once the professor clicks “edit”, they can edit the question and enable / disable it from the edit screen.



Students

The student will click “take exam”. Upon clicking this, our code runs the Python and generates the answers / the numbers in the question. It then compiles the LaTeX in the browser. The idea is to give students near limitless amounts of questions, while giving the professor as little work as possible. By doing this, the professor spends perhaps a few minutes to an hour longer on the exam, but the students benefit greatly from this.

Course Representatives

All course representatives are students but will also have the ability to view the reports generated for exams for modules they’re on.

Administrators

Have infinite power and can do anything, as they are the maintainers and controllers of the system.

We’re supporting students and professors

ExGen isn’t just to improve the education of students. It’s also to improve the productivity of professors. All the professor has to do is create the paper (just like they normally would) and write code to solve the question. Once they’ve done this, they can be sure that every paper the students download is unique with correct answers and correct questions.

Once students have completed a few papers, the lecturer will see a report filled with statistics on what the students found hard, what they found easy and more. With this information in hand, the lecturer can laser-focus into what the students need to improve at to get better grades.

Questions & Statistics

The questions will be stored in a database, along with what users got them right or wrong. The worth of the database is the square of the number of users in the system (Metcalfe's law[3]). Because the worth of the database grows exponentially, the size will also grow exponentially. Originally, the system will have little to no size or entries into the database. But, as the worth of the system increases so will the data it stores.9

[3] https://www.wikiwand.com/en/Metcalfe%27s_law

The system will make repeated searches to the database. One search for the question, one search for the user logging in, one search for whether the user got the question right or wrong.

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.

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.

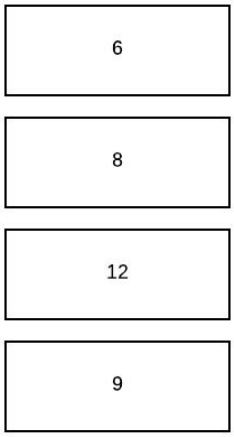
Performance

Performance is an important aspect of our project. We want to be able to generate exams as fast as possible without sacrificing the accuracy of the generated exam. The first step in maximising performance is to use a content delivery network such as CloudFlare. The second step would be to increase the performance of the system as the system grows. The worth of the

system grows exponentially with its use, but increasing the specs of the server exponentially is costly.

We're only going to generate questions when the user needs it, lazy-generating the questions. When the student clicks on the module they want, and then the exam they will be presented with the first question like so:

1) What is 6 + 3?



This question is automatically generated when the user presses “take exam”. The question generated by the user will be stored in a database to be used as a caching system. Each exam will have a table like so:

Question 1	[ID1][ID2][ID3]
Question 2	[ID4]
Question 3	[ID5]

Each ID corresponds to a unique question generated by the program. The program will show the user the question & answers that they have not seen before.

When the user answers the question, the program will then either retrieve the next question from the cache or generate a new question for the user. This speeds up the program considerably. Some students will likely answer 2 or 3 questions before giving up for the day, so generating every question in an exam for every student is pointless.

When a professor enters their paper the program will preemptively generate a set of questions so the professor can check the exam before releasing it. The above example is showing how it works in an instance where a question isn’t stored in a cache.

Although the exact layout of the database isn’t decided yet, this is generally how it’ll look.

Ideally, the program, for the students, should load in as little time as possible and most of the content should be cached.

The generation process will be executed on another thread. If that thread exists for more than 10 seconds, an error has occurred and the thread will be terminated with the lecturer and administrators knowing that something has gone wrong.

Security

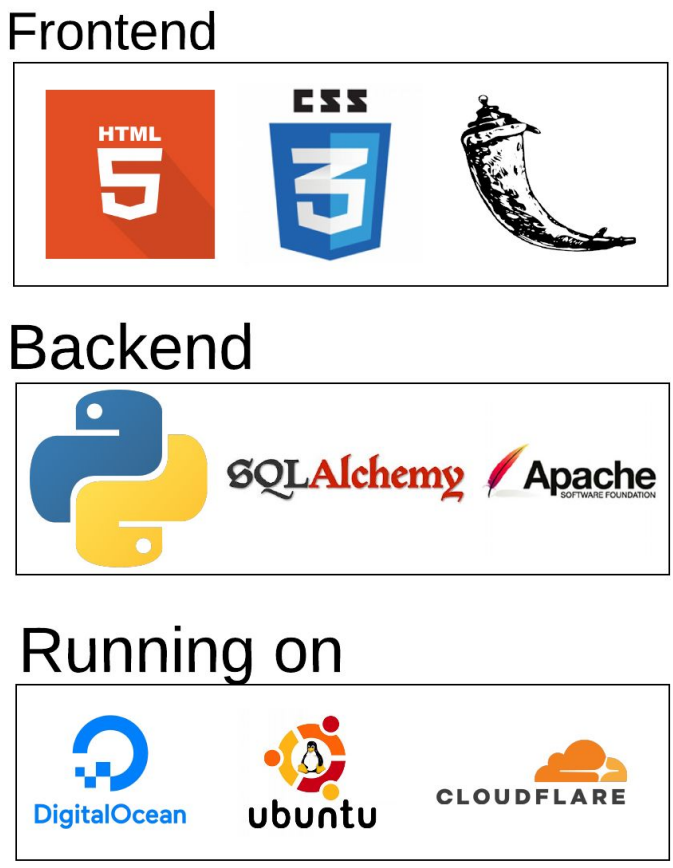
Because the program runs arbitrary code written by the professor, security is one of the most important issues we face. Firstly, only certified professors can execute code on the system. But, that doesn't mean that the professor might write code which sometimes can enter into infinite loops. Enter sandboxing.

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 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.

Stack

The technology stack is:



The frontend is HTML, CSS and Flask. The backend is an SQL database (accessed via SQLAlchemy in Python) and Python. It's all running on an Ubuntu server hosted on Digital Ocean, with the content delivery network being CloudFlare.

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[4] 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.

Business Rules

Constraints

- When a course rep is assigned, it is for the whole course. Not the module.
- Only professors can create exams.
- All demonstrators and PhD students are considered professors in this system.
- All question have 4 answers.
- Admins have full and direct control of the database.
- All users start out as students, and they can later be upgraded to course reps or professors.
- A normal user can't have the same privileges as an admin.
- Professors or course reps can't have the same privileges as an admin.
- Professors and course reps both see the same reports.

Derivations

- The time of development of this project is the sum of the people developing it multiplied by 2 each day. Each member should put in at least 2 hours a day.
- As we are students, our budget is £0. We only have free tools from the GitHub Student Pack and other free tools.Operation rules

Operations

- When registering both passwords need to be the same.
- Users must verify their email addresses.
- After verification, the user can be automatically logged in.
- Opting into a module will add that module to a students account.
- Opting out of a module will remove it from a students account.
- You cannot opt into a module if there are no modules created by a professor.
- Only course reps and professors can access the reports of an exam.

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

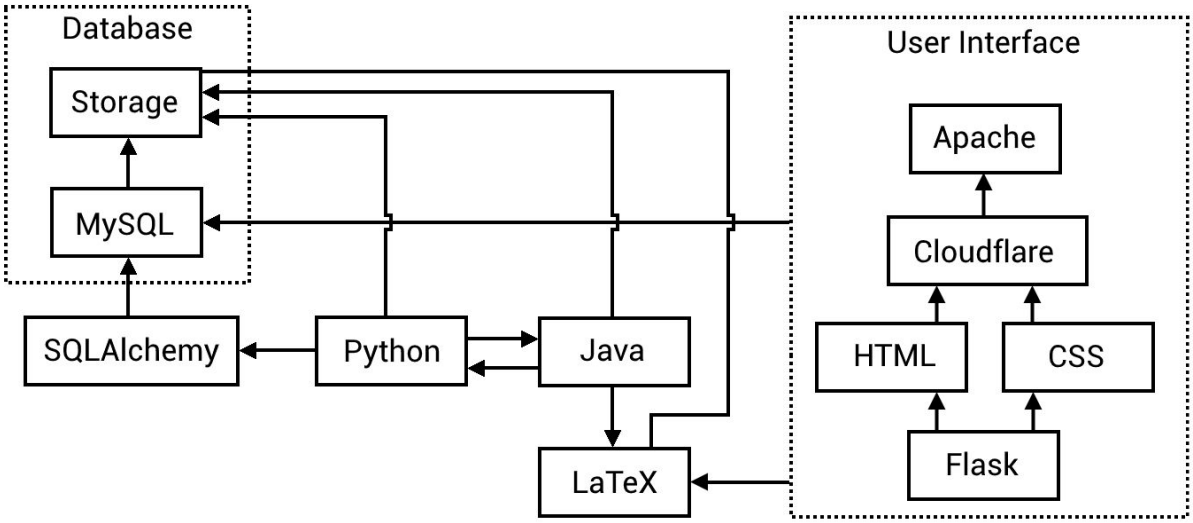
- When deleting an exam questions will be saved for use in another exam
- Questions can be used in multiple exams
- You cannot take an exam if no exam has been created for that module

System Boundary Design

The system itself as a whole will be hosted on a server(on Digital Ocean). This server will be running Apache(as a web server) and MySQL(to handle databases). The server will also have Python, Java and LaTeX installed. These are needed as our backend code is made in Python and Java and LaTeX.

The backend code of the system will be manageable by the admins and not visible to any other user. The professor will be able to create/upload his own LaTeX code but will not see any backend code, they will see our frontend website.

The databases will be managed by MySQL and will be queried via our frontend website through SQLAlchemy which is a python library.



User Views and Requirements

Actors:

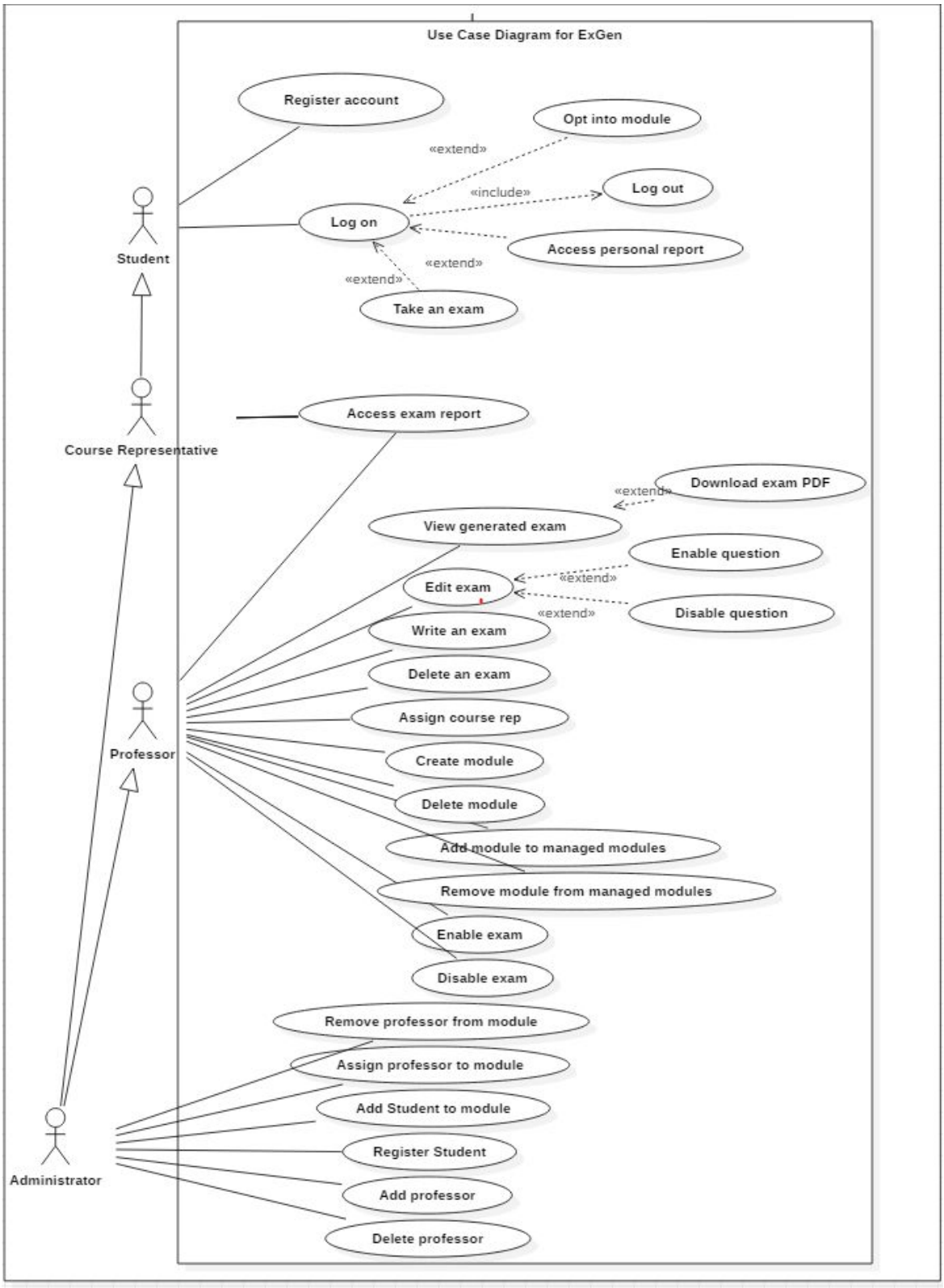
1. Student
2. Course representatives (or external verifiers)
3. Professors
4. Administrators

Use Cases:

Total: 27 use cases.

- Register account.
- Log on.
- Opt into module
- Log out.
- Access personal report
- Take an exam.
- Access exam report.
- View generated exam.
- Download exam PDF.
- Edit exam.
- Enable question.
- Disable question.
- Write an exam.
- Delete exam.
- Assign course rep.
- Add student to module.
- Register student.
- Add professor.
- Delete Professor.
- Remove professor from module.
- Assign professor to module.
- Enable exam.
- Disable exam.
- Remove module from managed modules.
- Add module to managed modules.
- Create module.
- Delete module.

Use Case Diagram



Use Case Descriptions

ID	S1
Name	Register Account
Description	Student registers an account with ExGen by using their university email account and creating a password.
Pre-conditions	● ExGen service live.
Event flow	1. Student goes to the ExGen website and clicks register account.
Post-condition	● Student account registered.
Includes	N/A
Extensions	N/A
Triggers	Student clicks register account.

ID	S2
Name	Log on
Description	Student logs onto the ExGen service by using his/her uni email address and the password created when registering.
Pre-conditions	● ExGen service live.
Event flow	1. Student registers an account with the ExGen service. 2. Student logs onto the ExGen service with the details used to register.
Post-condition	● Student account registered.
Includes	Log out (S6)

Extensions	Opt into module (S3), Access personal report (S4), Take an exam (S5).
Triggers	Student logs onto the ExGen service.

ID	S3
Name	Opt into module
Description	Student can choose to opt into their own modules.
Pre-conditions	● ExGen service live. ● Logged in.
Event flow	1. Student registers with the ExGen service. 2. Student logs onto the ExGen service. 3. Student finds the module they wish to be opted into and opts in.
Post-condition	● Student has access to exam questions created for the module.
Includes	N/A
Extensions	N/A
Triggers	Student clicks opt in on the specific modules.

ID	S4
Name	Access personal report
Description	Student can access a personal report of all exam questions taken.
Pre-conditions	● ExGen service live. ● Logged in. ● Exam taken prior.
Event flow	1. Student registers with the ExGen service. 2. Student logs onto the ExGen service. 3. Student opts into a module.

	4. Student takes an exam. 5. Student accesses their personal report.
Post-condition	● Student views personal report.
Includes	N/A
Extensions	N/A
Triggers	Student clicks access personal report.

ID	S5
Name	Take an exam
Description	Student takes an exam of questions related to the module they're opted into.
Pre-conditions	● ExGen service live. ● Logged in. ● Opted into a module. ● Professor uploaded/created an exam.
Event flow	1. Student registers with the ExGen service. 2. Student logs onto the ExGen service. 3. Opts into a module. 4. Student takes an exam.
Post-condition	● Exam report generated.
Includes	N/A
Extensions	N/A
Triggers	Student clicks take an exam.

ID	S6
Name	Log out
Description	Student logs out of the ExGen service
Pre-conditions	● ExGen service live (S1). ● Logged in (S2).

Event flow	1. Student registers with the ExGen service. 2. Student logs onto the ExGen service. 3. Student logs out of the ExGen service.
Post-condition	● Student logged out.
Includes	N/A
Extensions	N/A
Triggers	Student clicks log out.

ID	CR1
Name	Access exam report
Description	A course rep or professor can access an exam report, giving an overview of the students performances on a given exam.
Pre-conditions	● ExGen service live (S1). ● Logged in (S2). ● Exam taken prior. ● Assigned course rep by a professor.
Event flow	1. Students register with the ExGen service. 2. Students log onto the ExGen service. 3. Students take an exam. 4. Assigned course rep by professor. 5. Course rep or professor accesses the exam report.
Post-condition	● Course rep or professor views exam report.
Includes	N/A
Extensions	N/A
Triggers	Course rep or professor clicks view exam report.

ID	P1
Name	View generated exam
Description	Professor can view a exam that has been generated by ExGen.

Pre-conditions	● ExGen service live. ● Logged in. ● Exam generated.
Event flow	1. Professor uploads/enters an exam 2. Professor views the exam.
Post-condition	
Includes	N/A
Extensions	● Download exam PDF.
Triggers	Professor views exam.

ID	P2
Name	Download exam PDF
Description	Professor can download a PDF of the exam being viewed.
Pre-conditions	● ExGen service live (S1). ● Logged in (S2). ● Exam generated. ● Viewing exam.
Event flow	1. Professor uploads/enters an exam 2. Professor views the exam. 3. Professor downloads.
Post-condition	● PDF downloaded.
Includes	N/A
Extensions	N/A
Triggers	Professor presses download exam.

ID	P3
Name	Edit exam
Description	Professor can edit the exam from within the ExGen web field.

Pre-conditions	● ExGen service live (S1). ● Logged in (S2). ● Exam generated.
Event flow	1. Professor uploads/enters an exam 2. Professor views the exam. 3. Professor edits the exam from within the ExGen web field.
Post-condition	● Updated version of the exam available.
Includes	N/A
Extensions	Enable question (P4), Disable question (P4).
Triggers	Professor edits exam.

ID	P4
Name	Enable question
Description	As questions are automatically enabled when added, this case involves re enabling a question that has previously been disabled.
Pre-conditions	● ExGen service live. ● Logged in. ● Exam generated. ● Question disabled.
Event flow	1. Professor uploads/enters an exam. 2. Professor views exam. 3. Professor disables a question. 4. Professor enables the question.
Post-condition	● Question enabled and within the exam question pool.
Includes	N/A
Extensions	N/A
Triggers	Professor clicks enable question.

ID	P5
Name	Disable question
Description	Professor disables a question within a generated exam.
Pre-conditions	● ExGen service live. ● Logged in. ● Exam generated.
Event flow	1. Professor uploads/enters an exam. 2. Professor views exam. 3. Professor disables a question.
Post-condition	● Question disabled and not within the exam question pool.
Includes	N/A
Extensions	N/A
Triggers	Professor clicks disable question.

ID	P6
Name	Write an exam.
Description	Professor enters exam questions into the ExGen web field, and can either allow it to generate answers or manually enter answers.
Pre-conditions	● ExGen service live. ● Logged in.
Event flow	1. Professor logs in. 2. Professor writes an exam.
Post-condition	● Finished exam generated and available to students.
Includes	N/A
Extensions	N/A
Triggers	Professor enters an exam.

ID	P7
Name	Delete an exam.
Description	Professor deletes an exam that has been previously written into the ExGen service.
Pre-conditions	● ExGen service live. ● Logged in. ● Exam previously written.
Event flow	1. Professor logs in. 2. Professor writes an exam. 3. Professor deletes written exam.
Post-condition	● Exam deleted and unavailable for students.
Includes	N/A
Extensions	N/A
Triggers	Delete exam button clicked.

ID	P8
Name	Assign Course Rep
Description	Professor assigns a student registered to the ExGen service to act as a course representative, this enables a student to see an overview of the exam performance of all students who have taken an exam.
Pre-conditions	● ExGen service live. ● Student and professor registered.
Event flow	1. Professor logs in. 2. Professor then assigns a registered student to be a course rep.

Post-condition	● Student now able to access the exam report for all students for a given exam.
Includes	N/A
Extensions	N/A
Triggers	Professor clicked assign course rep for a given student.

ID	P9
Name	Create module
Description	Create a module mirroring a module in the institution that have a set of students and resources on the platform.
Pre-conditions	● ExGen service live. ● Logged in.
Event flow	1. Professor types in name of module 2. Professor creates module
Post-condition	New module is created
Includes	N/A
Extensions	N/A
Triggers	User clicks “Create module” button

ID	P10
Name	Delete module
Description	Delete module, removing user’s access to its resources and removing it from the professor’s list of managed modules
Pre-conditions	● ExGen service live. ● Logged in.
Event flow	1. Professor chooses which module to delete 2. Module is deleted
Post-condition	Module is deleted from database

Includes	N/A
Extensions	N/A
Triggers	User clicks “Delete module” button

ID	P11
Name	Add module to managed modules
Description	Adds an already created module to a list of modules the professor can manage (write exams, see statistics, etc.)
Pre-conditions	● ExGen service live. ● Logged in.
Event flow	1. Professor searches for module in database 2. Professor adds said module to their modules
Post-condition	Professor is now able to manage module (write exams, see statistics, etc.)
Includes	N/A
Extensions	N/A
Triggers	User clicks “Add module” button on the module page

ID	P12
Name	Remove module from managed modules
Description	Removes a module to a list of modules the professor can manage (write exams, see statistics, etc.)
Pre-conditions	● ExGen service live. ● Logged in.
Event flow	1. Professor finds module in their list of managed modules 2. Professor deletes it from their list of managed modules

Post-condition	Professor is no longer able to manage module (write exams, see statistics, etc.)
Includes	N/A
Extensions	N/A
Triggers	User clicks “Remove module” button on the module page

ID	P13
Name	Enable exam
Description	Make exam visible and interactive to students
Pre-conditions	● ExGen service live. ● Logged in.
Event flow	1. Professor finds written exam to be enabled 2. Professor enables said exam
Post-condition	Students have access to exam
Includes	N/A
Extensions	N/A
Triggers	User clicks “Enable exam” button on the exam page

ID	P14
Name	Disable exam
Description	Make exam not visible and not interactive to students
Pre-conditions	● ExGen service live. ● Logged in.
Event flow	1. Professor finds enabled exam to be disabled 2. Professor disables exam
Post-condition	Students no longer have access to exam

Includes	N/A
Extensions	N/A
Triggers	User clicks “Disable exam” button on the exam page

ID	A1
Name	Remove professor from module
Description	Remove professor from list of professors who manage a certain module and removes that module from the professor’s list of managed modules
Pre-conditions	● ExGen service live. ● Logged in.
Event flow	1. Admin finds module they want to modify 2. Admin finds professor they want to remove 3. Admin removes professor from module
Post-condition	Professor is no longer able to manage module (write exams, see statistics, etc.)
Includes	N/A
Extensions	N/A
Triggers	User clicks “Remove professor” button on the professors section of the module page

ID	A2
Name	Assign Professor to module
Description	Puts professor in list of professors who manage a certain module and puts that module in the professor’s list of managed modules
Pre-conditions	● ExGen service live. ● Logged in.

Event flow	1. Admin finds module they want to modify 2. Admin finds professor they want to add 3. Admin adds professor to module
Post-condition	Professor is now able to manage module (write exams, see statistics, etc.)
Includes	N/A
Extensions	N/A
Triggers	User clicks “Add professor” button on the professors section of the module page

ID	A3
Name	Add student to module
Description	Puts student in list of students who take a certain module and puts that module in the student’s list of modules
Pre-conditions	● ExGen service live. ● Logged in.
Event flow	1. Admin finds module they want to modify 2. Admin finds student they want to add 3. Admin adds student to module
Post-condition	Student has access to exam questions created for the module.
Includes	N/A
Extensions	N/A
Triggers	User clicks “Add student” button on the students section of the module page

ID	A4
Name	Register student

Description	Student is registered with an account with ExGen by using their university email account and creating a password.
Pre-conditions	● ExGen service live. ● Logged in.
Event flow	1. Admin enters the information of the student they want to register 2. Student is registered in the database
Post-condition	Student account registered.
Includes	N/A
Extensions	N/A
Triggers	User clicks “Register new student” button

ID	A5
Name	Add professor
Description	Professor is registered with an account with ExGen by using their university email account and creating a password.
Pre-conditions	● ExGen service live. ● Logged in.
Event flow	1. Admin enters the information of the professor they want to register 2. Professor is registered in the database
Post-condition	Professor account registered.
Includes	N/A
Extensions	N/A
Triggers	User clicks “Register new professor” button

ID	A6

Name	Delete professors
Description	Professor's account is deleted
Pre-conditions	● ExGen service live. ● Logged in.
Event flow	1. Admin finds professor in the database 2. Admin deletes professor from database
Post-condition	Professor account deleted
Includes	N/A
Extensions	N/A
Triggers	User clicks "Remove professor" button in professor page

Transaction Requirements

There are three fundamental types of transactions we need to plan for from each user:

- Data Entry
    - The creation of new data
- Data Update and Deletion
    - The changing and eliminating of old data
- Data Queries
    - The questioning of current data

Let's take a look at this per user, the first of which being the Student.

The Student

The Student is able to create an account and set it up with all the necessary information to verify it. For example a password and university email. The student may also cause new questions to be generated and new question IDs added to the database if they exhaust the cached ones.

The Student may delete their account at any time but all the statistics and information will be lost to them. They also cause the database to update when they complete a question, as the question ID is added to the list of questions they have completed and whether or not they correctly answered the question is recorded.

The student may also see their statistics for a module and statistics for individual exams. as well as querying for new exam questions to complete.

The Course Representative

Course Representatives are subsets of Students. This group can do everything a normal student can but may also see an overview of the current exam statistics for the module they are assigned to.

The Professor

The Professor may add new questions, as demonstrated above in the system specification. They have the ability to appoint Course Representatives from the students currently taking their course and they may create new modules as needed.

The Professor may update any already made question through changing the question in the interface. If a question is found to be incorrect or unnecessary they may also disable the questions. The professor has the ability to enable or disable entire exams. As the overseer of a course they also will have the ability to delete a Student entirely from that course and reject a Course Representative.

The Professor may query questions to check their correctness and access the overall statistics, along with statistics on a student by student basis, if given access by the student.

The Administrator

The Administrator has full control over the database and therefore can do all of the above but they are the only users to be able to directly access the database and may change any field. They are also the only users able to add a professor, delete a professor and delete a module.

Gantt Chart

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}

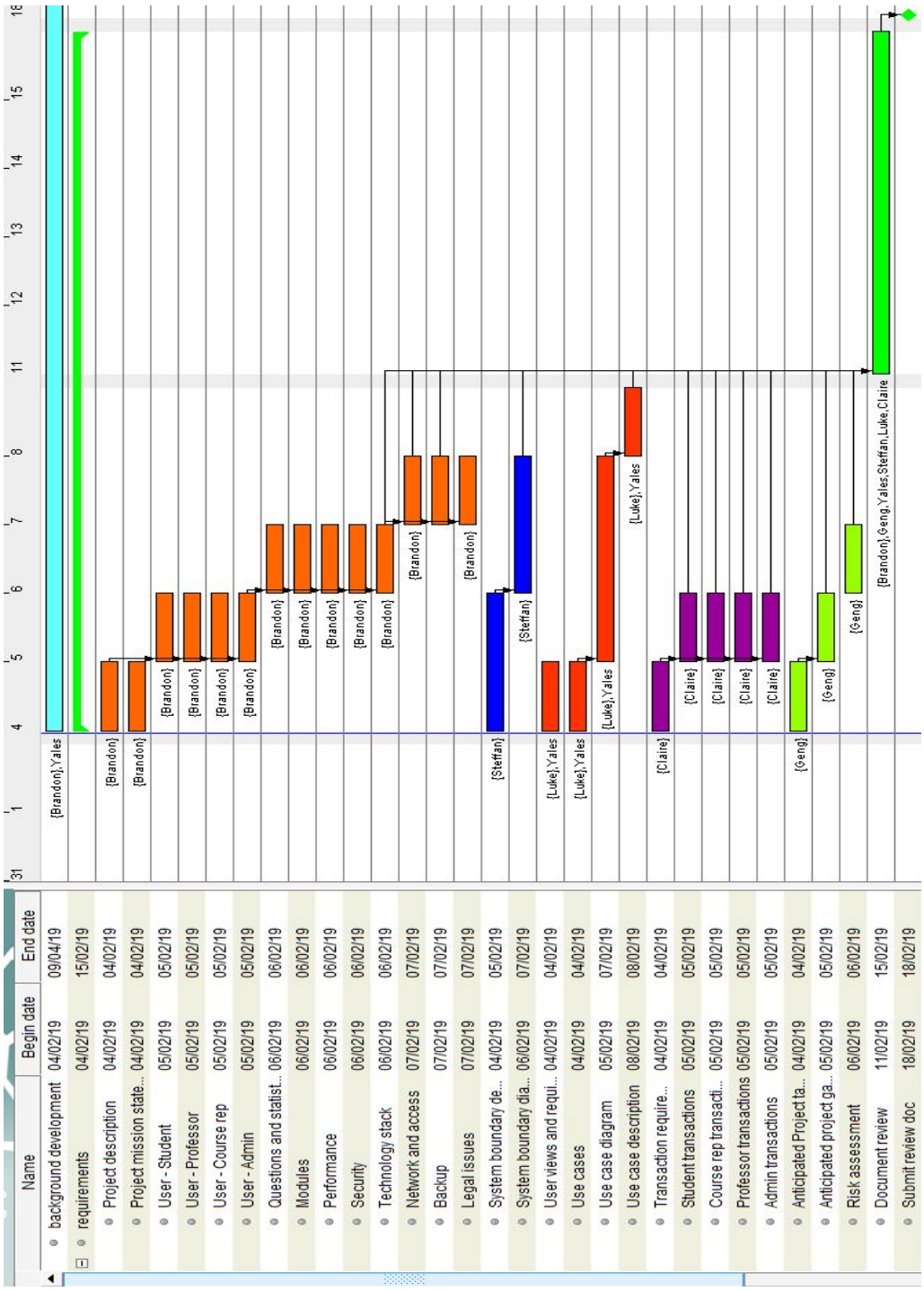
53	Logical table structures	2	01/03/19	03/03/19	53	{Claire}
54	Use case diagram	1	25/02/19	26/02/19	none	{Luke}
55	Data structure design	5	25/02/19	02/03/19	none	{Luke}Geng
56	Document review	8	06/03/19	15/03/19	All of the above	{Brandon}Yales, Geng, Steffan, Luke, Claire
57	Presentation	15	25/02/19	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

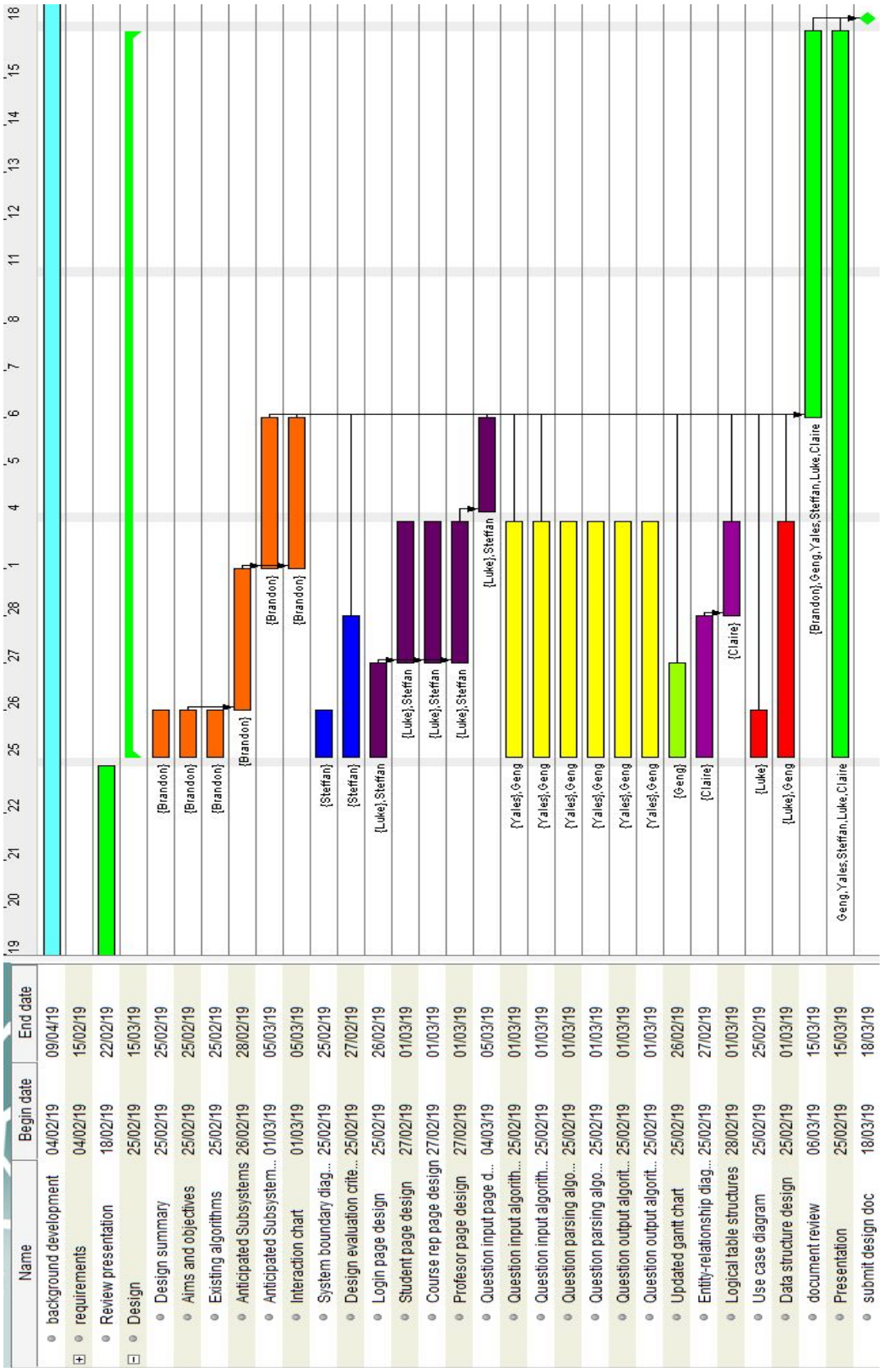
ID	Task	Duration (d)	Start on	End by	Dependencies	Assigned resources
59	Make user database	1	25/03/19	26/03/19	none	{Claire}
60	Make student database	1	25/03/19	26/03/19	none	{Claire}
61	Make profesor database	1	25/03/19	26/03/19	none	{Claire}
62	Make question database	1	25/03/19	26/03/19	none	{Claire}
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 page	3	25/03/19	28/03/19	none	{Steffan} Luke
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}

Requirements tasks Gantt chart

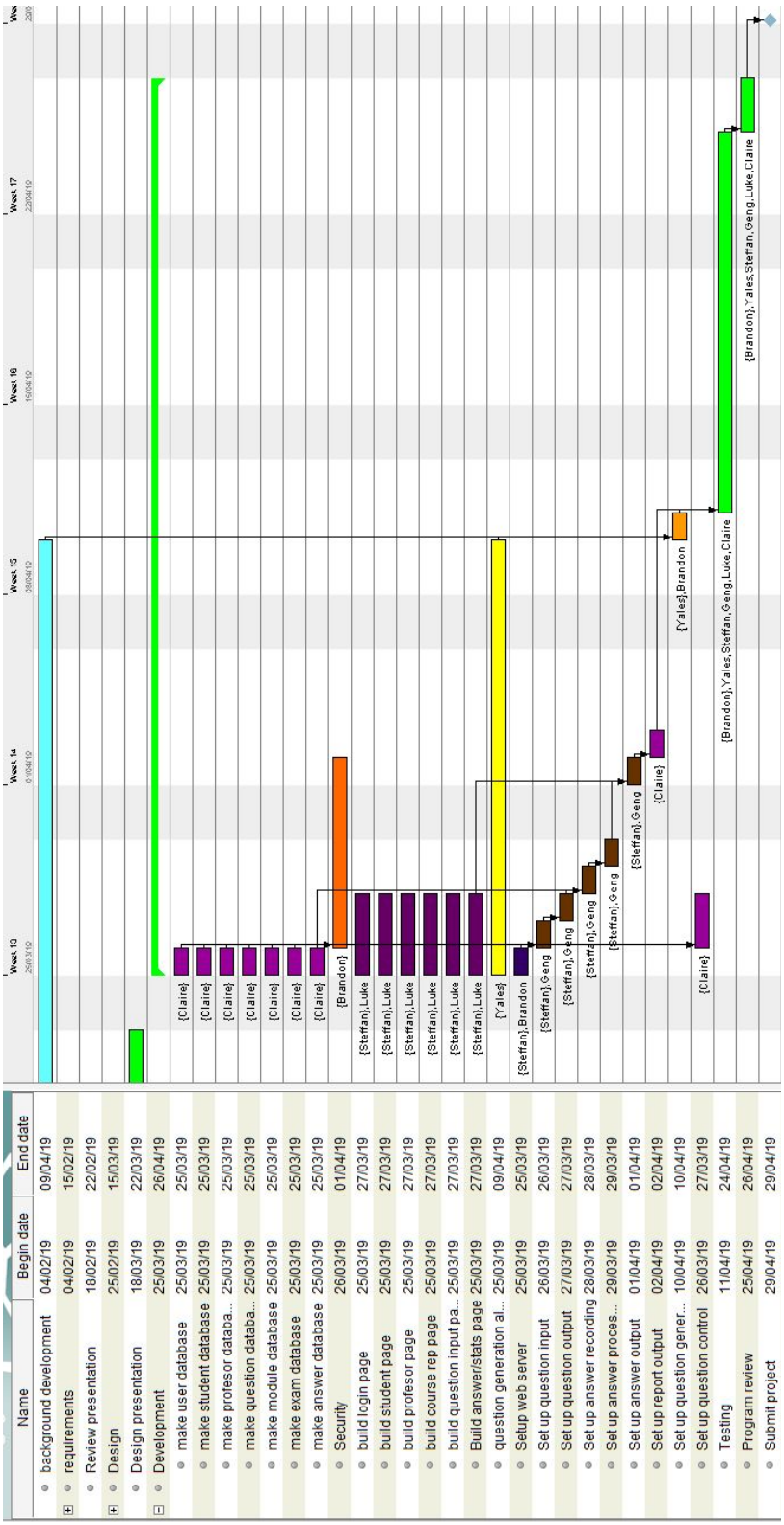


Design tasks Gantt chart



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

Development tasks



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

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. 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.	Since the user information should be tied to the universities own administration system the 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	Low	Moderate - if we run out of money to pay supporting costs	We have allocated enough money to host a server until the

our server.		for our host server we will lose access to it and will be unable to continue development, or demonstrate a working prototype.	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.

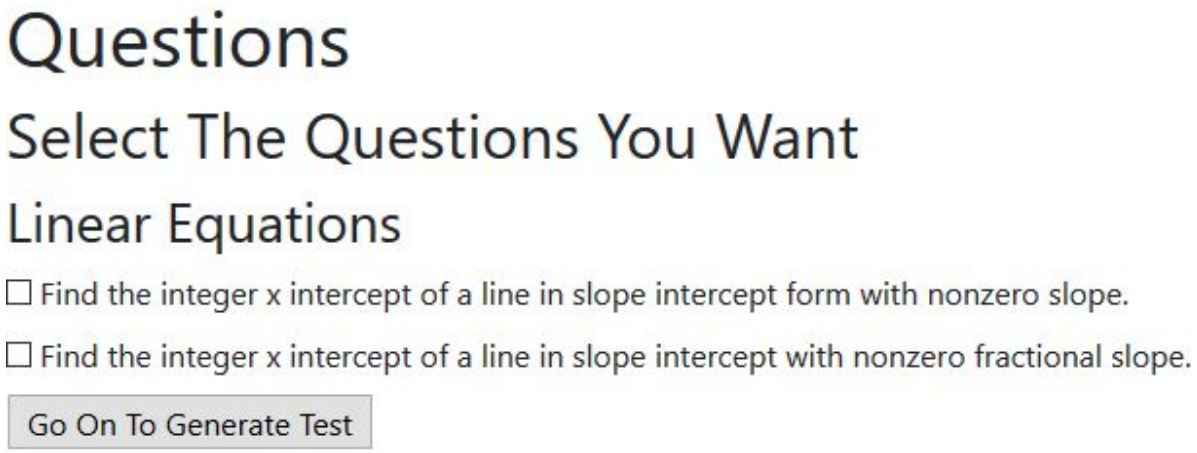
Design

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 plethora of volunteers to hand-write the numbers and answers for them, according to Khan Academy[5] themselves.

Someone else had the exact same idea as us, MathTestMaker[6]. Their program let's people choose from a range of questions:

And then you select the format of the question:



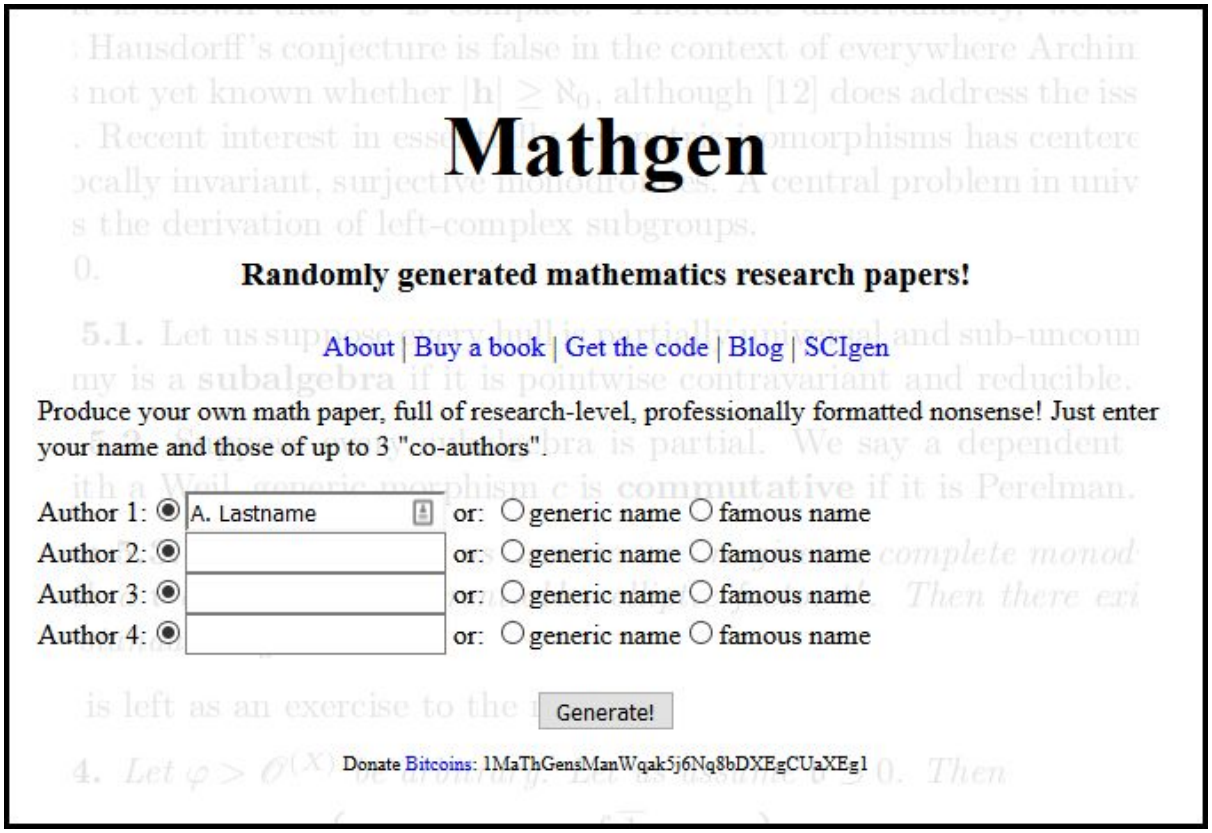
And then you get this messy output:

> *"{'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.')]} "*

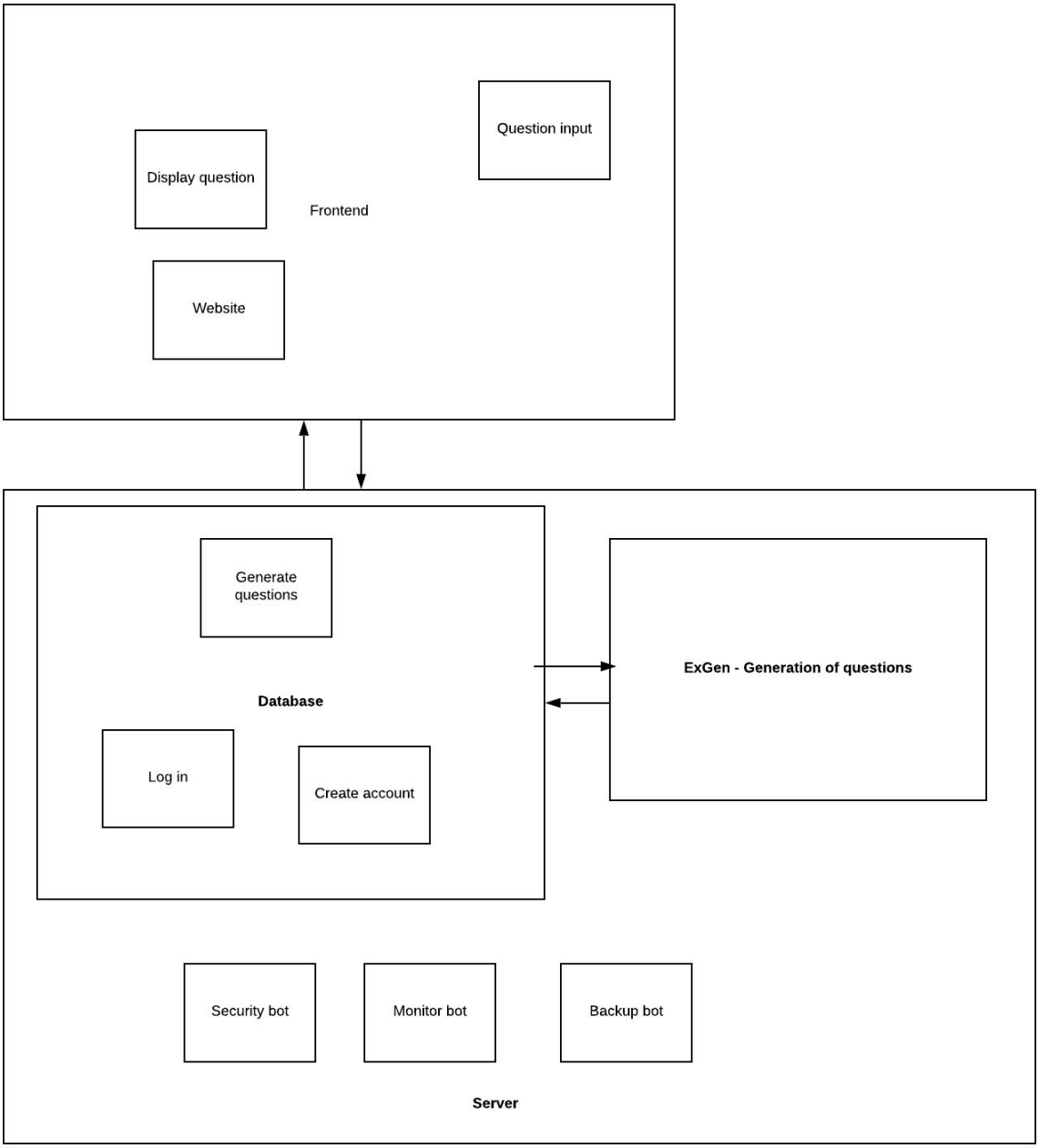
[5] https://www.khanacademy.org/about/blog/post/57701417065/100-000-practice-problems
[6] https://mathtestmaker.com

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.

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



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.

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.

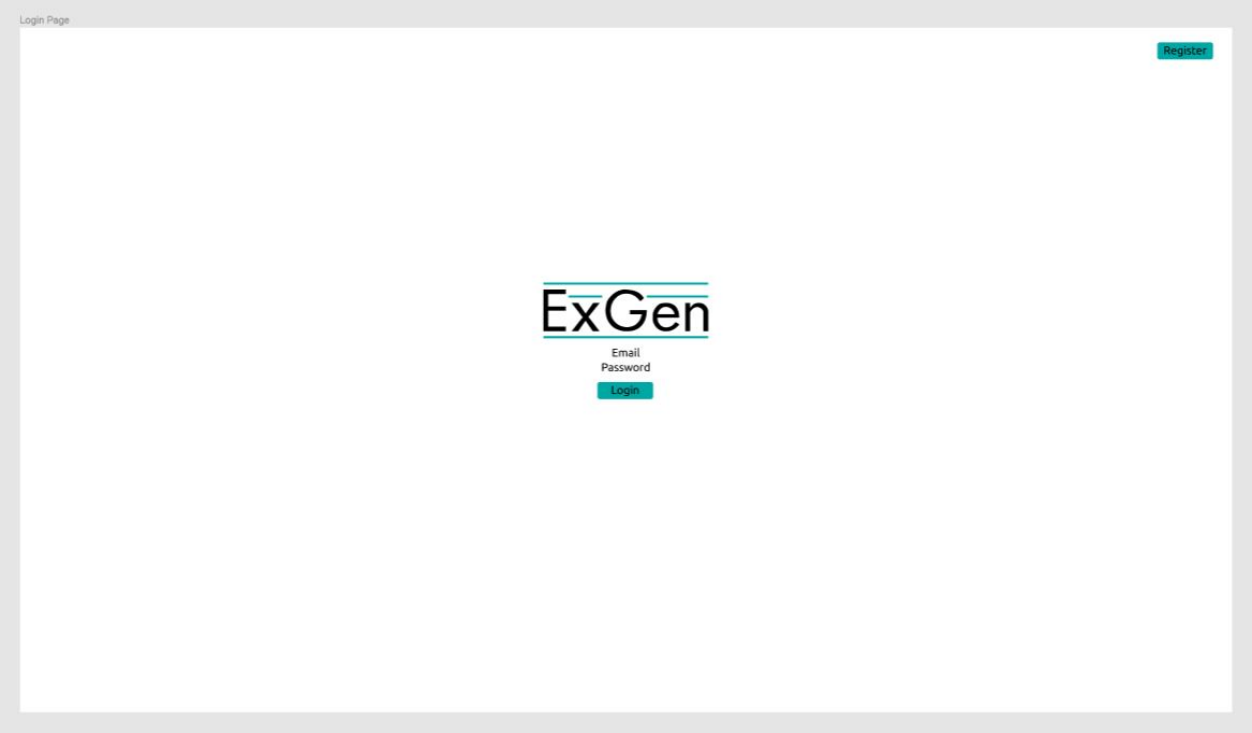
The navigation bar for students will contain the following: Home, Exams, Results and Settings and will look like this.

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.

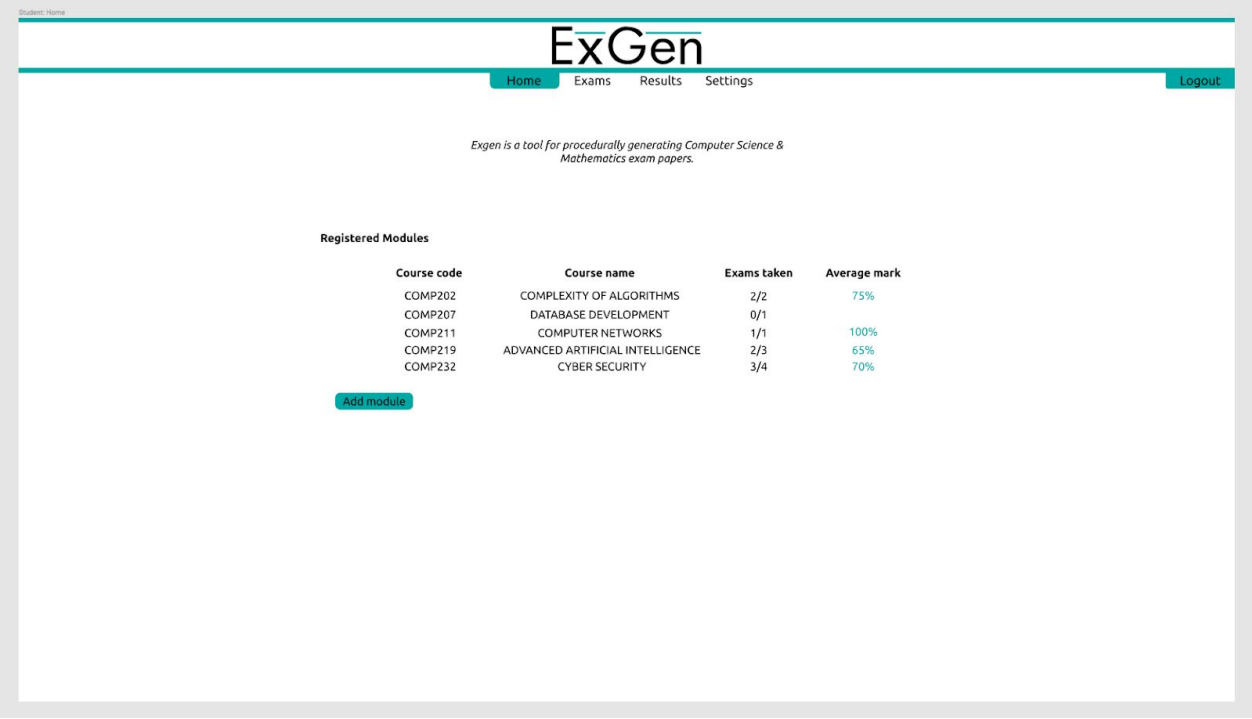
The theme that we are going for is a simple white background, blue features and black text with our font being the free to use "Ubuntu" font. The default page for our interface is the login page unless the user has already logged in.
The login page will be very simple. It will contain our logo, an email 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.



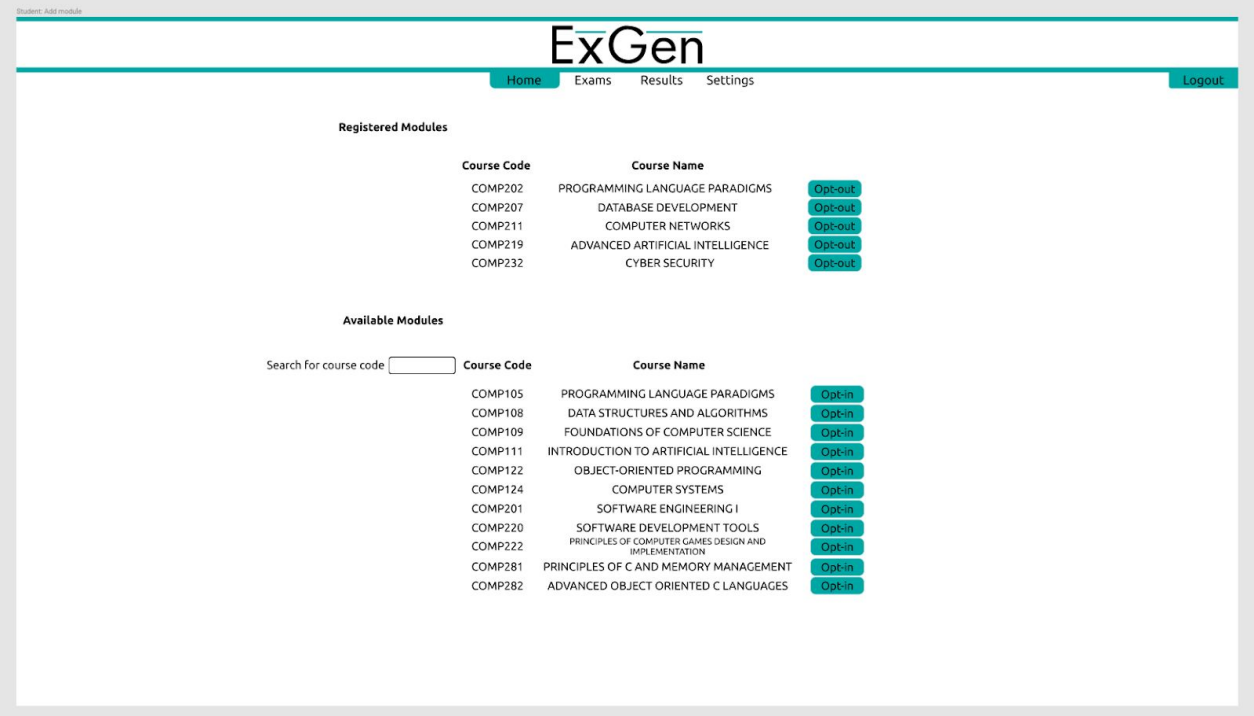
All 4 users will have the home page where they can see basic information about their relevant permissions.The Students and course representatives(as they are also students) "home" page will contain a table with the following information:

- 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.

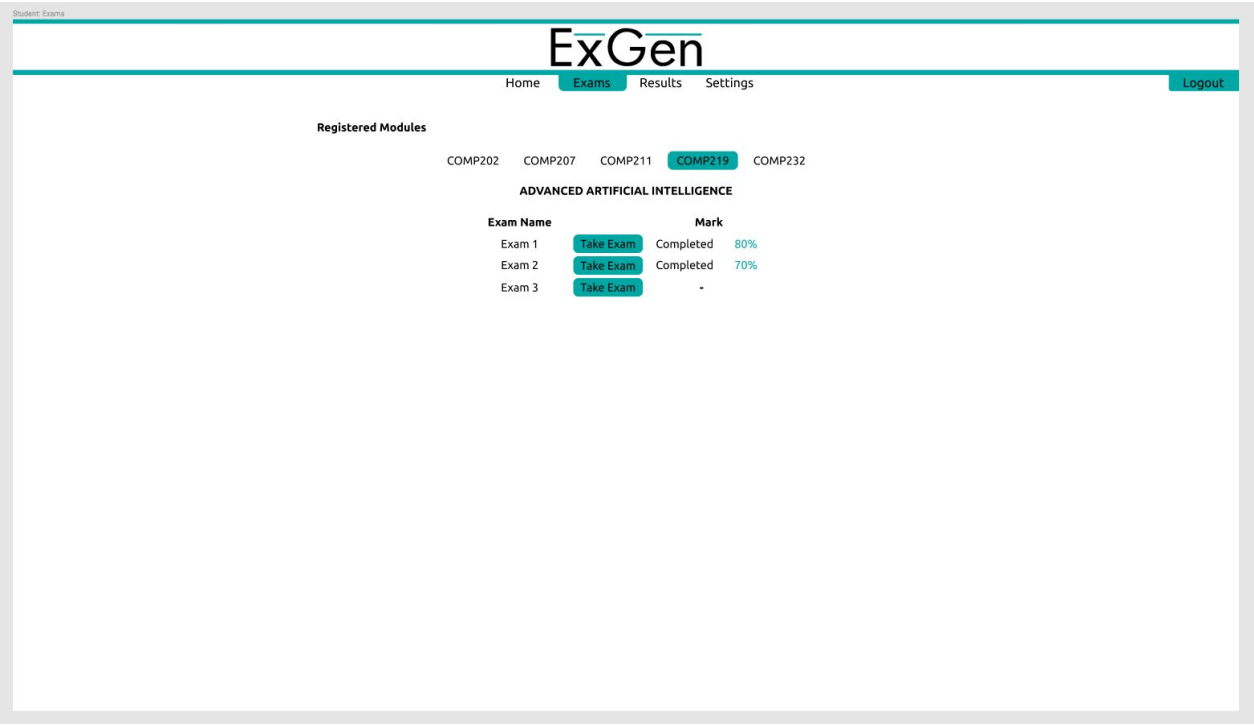


When the “Add module” button is pressed the following page will replace the content on the home page. This page contains a list of the modules that the student has already opted into and a list of available modules that the student can opt into. The user can press “opt-out” or “opt-in” on the desired 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.

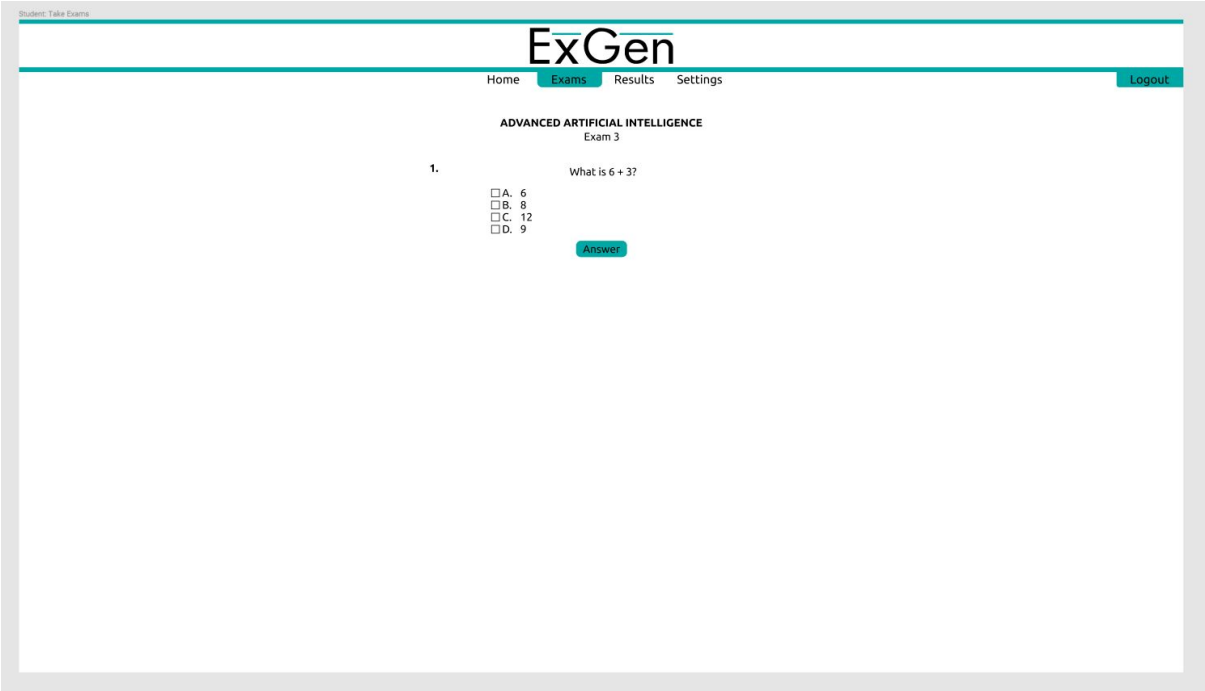


The home page for Admin and course representatives will be the same as students as they are also students.

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

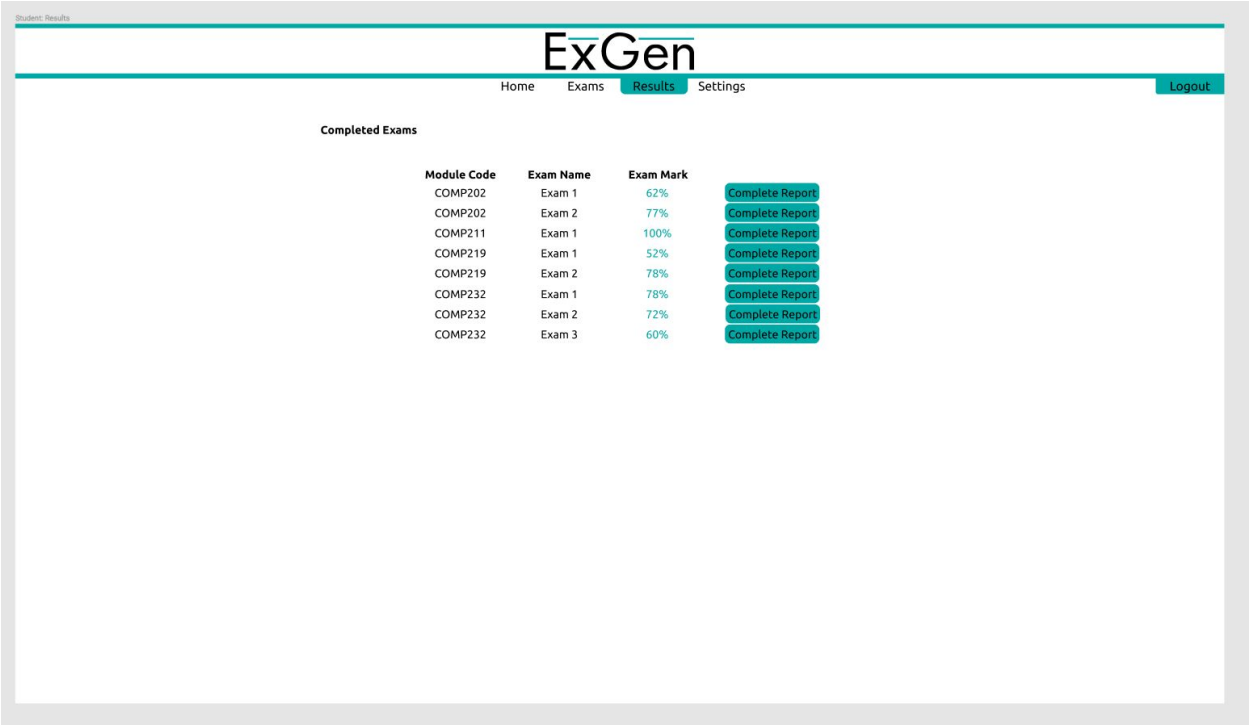


The user can change the list of available exams by clicking on another course code. The interface will default to the first registered course code of the user. If there are no exams for a given module then the list will be empty and should display “No exams available”. If a user has completed an exam 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.

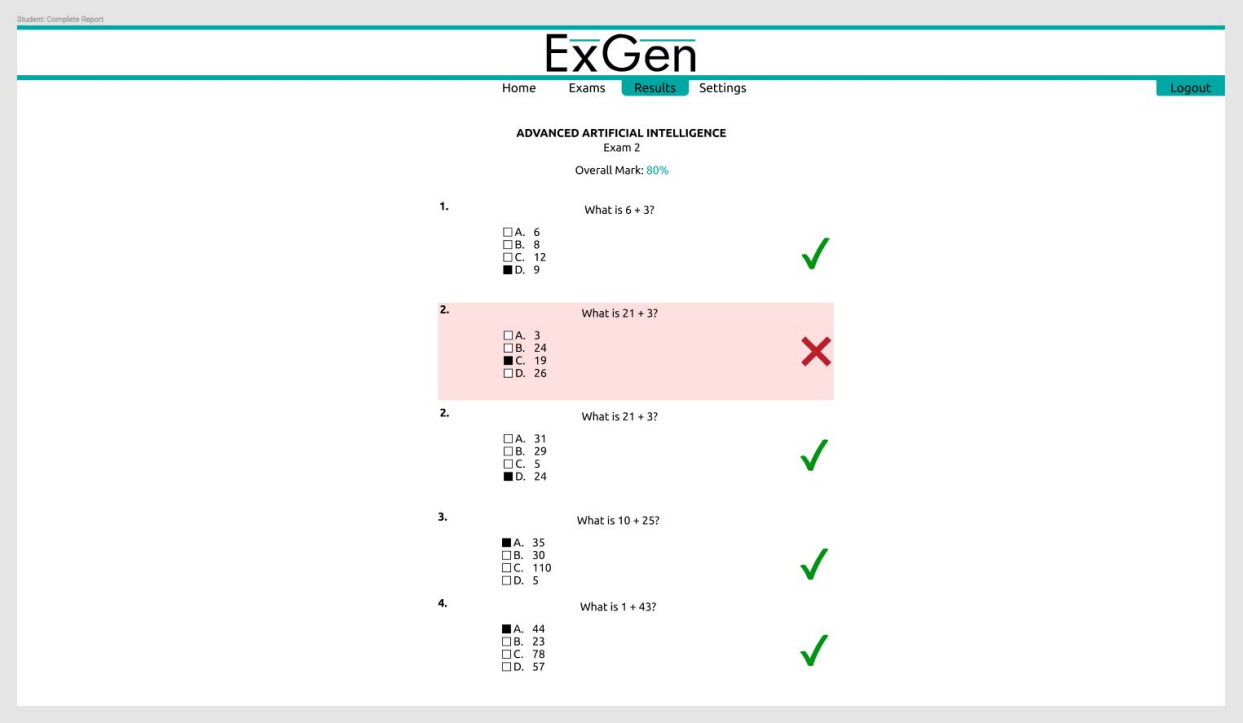


The user can then tick one of the answers and click the “answer” button. The system will then check if the question is correct and if it is correct then the system will return a new question to the user. If the user answers the question incorrectly then the system will show another example of the same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

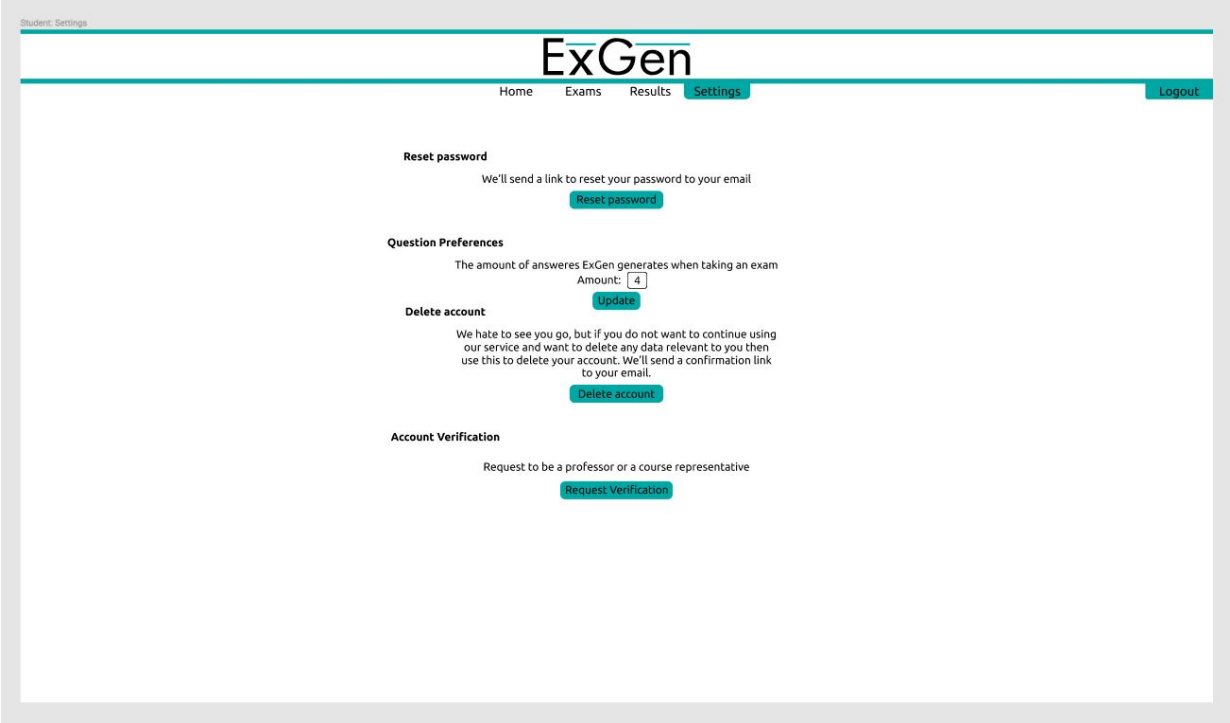
The next page is the students, course representatives and admins results page. This page contains a list of all completed exams with their mark for each exam. They can then get a complete report per exam.



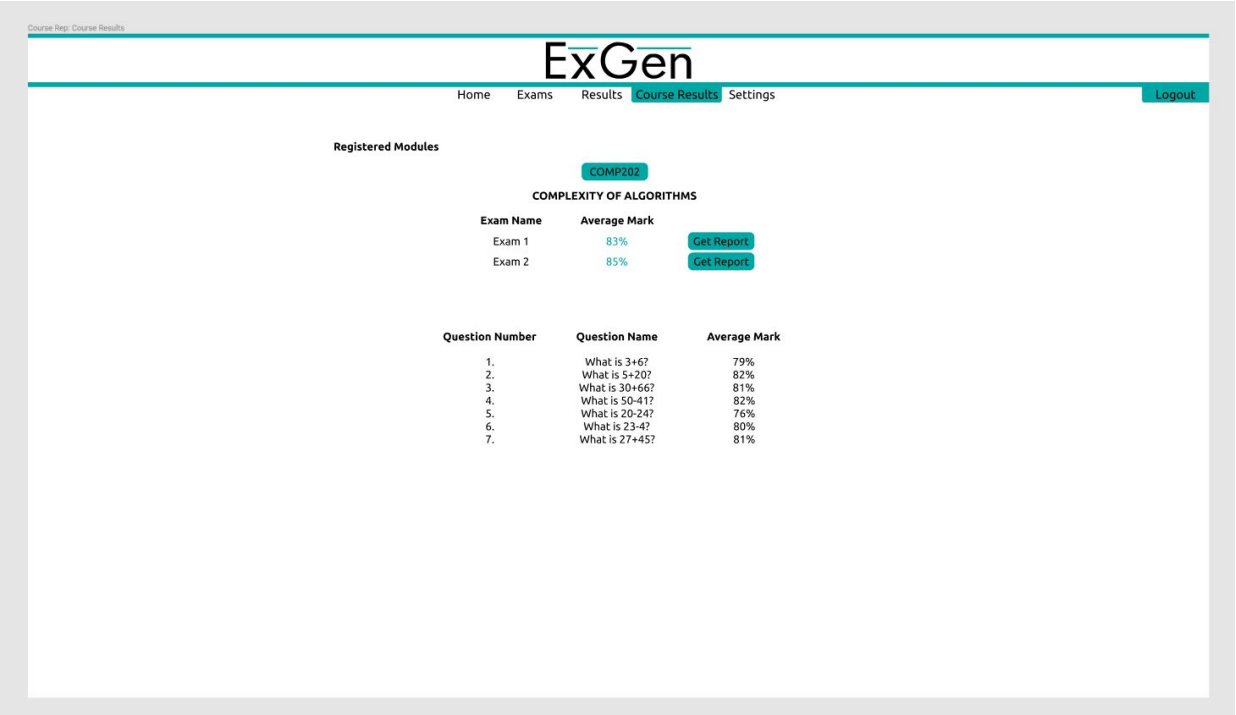
The complete report button opens the answered exam that the user completed. This will also show the users failed attempts at a question. If an exam had 3 questions and questions 1 and 3 were answered correctly but question 2 was incorrect the first time. The report would contain 4 questions. This screen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



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.

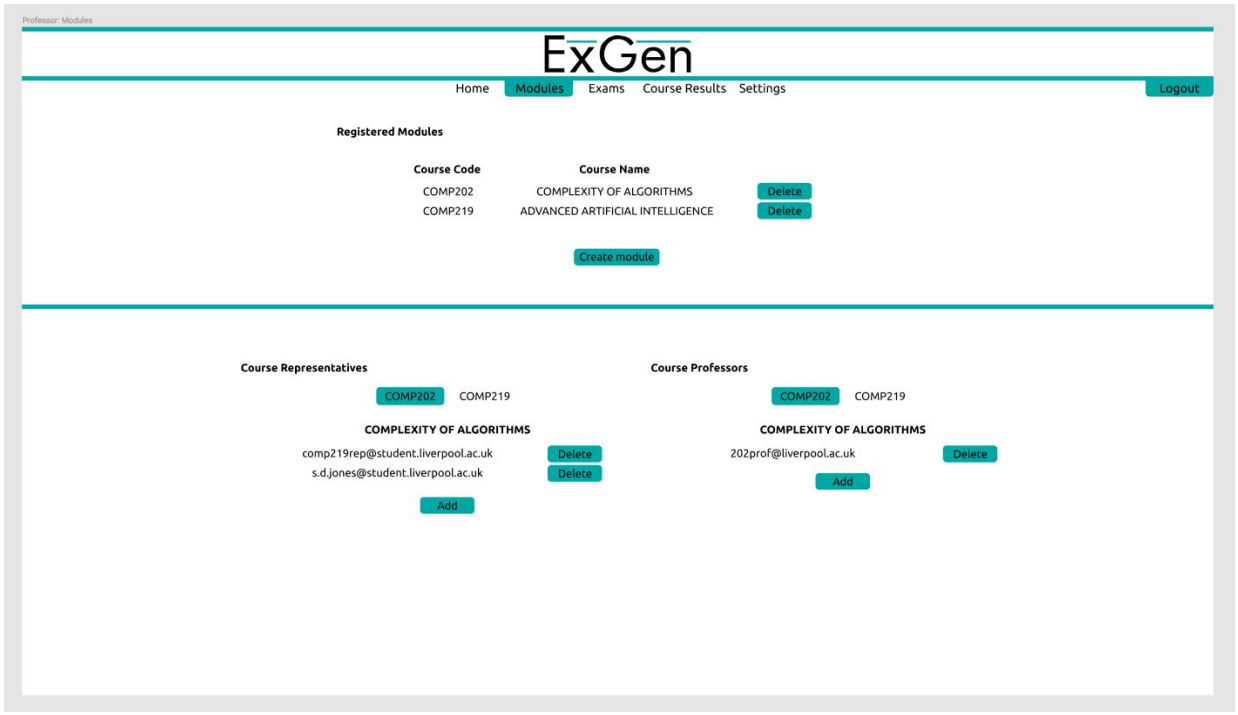


Course reps and the professor can view results for exams from all users who completed that exam. This page will list all exams that the course reps are a rep of. The course rep can click on “Get Report” which will bring up average marks per question for all of the exam.

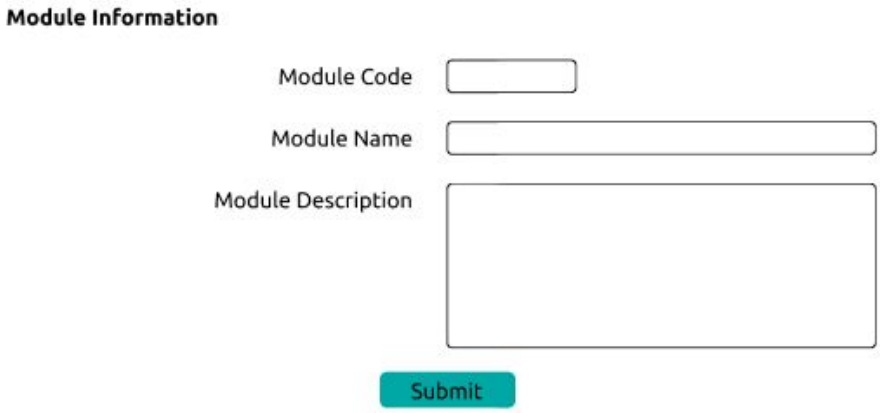


The professor's home page is similar to the other users home pages but the table shows the professors modules and the modules that the professor handles.

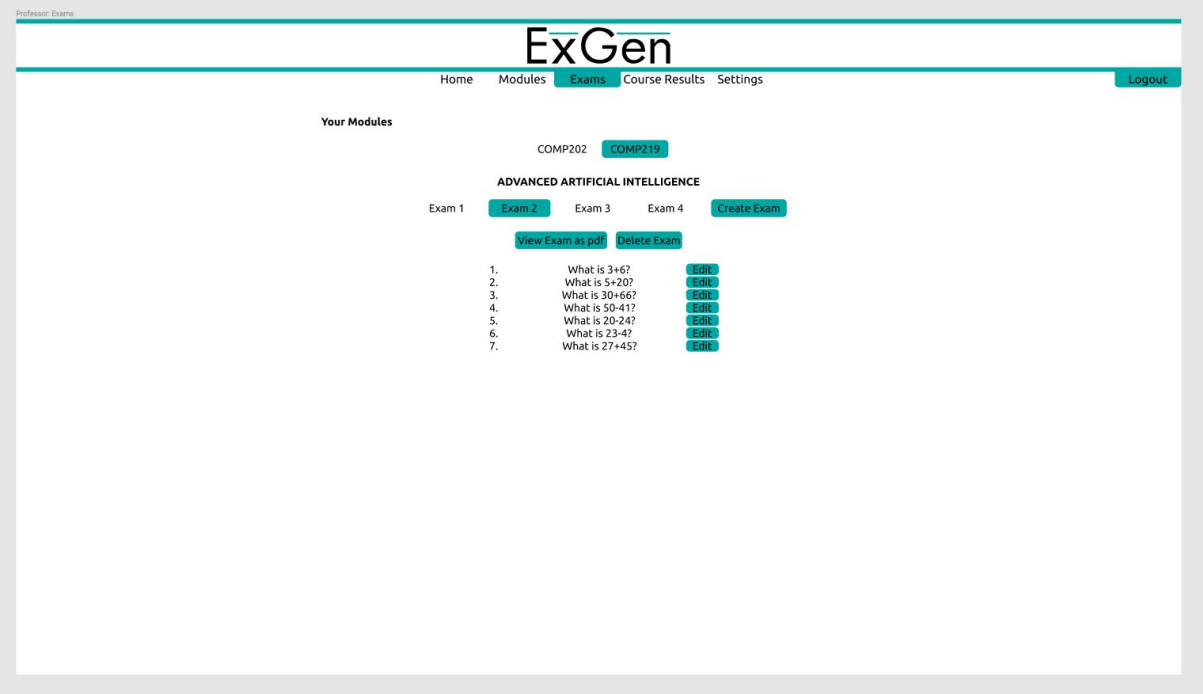
The Modules page is where the professor is able to create and delete modules, add course reps and other professors to manage their module.



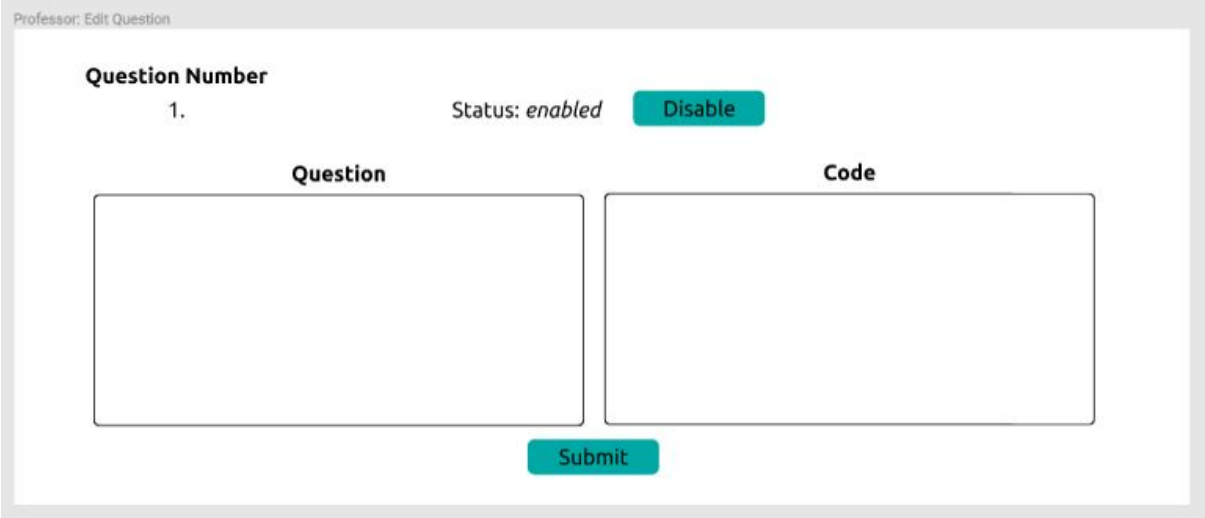
When a professor presses the “Create module” button a new panel will appear replacing the course representatives/course professors section. This panel is where the professor enters information about the module.



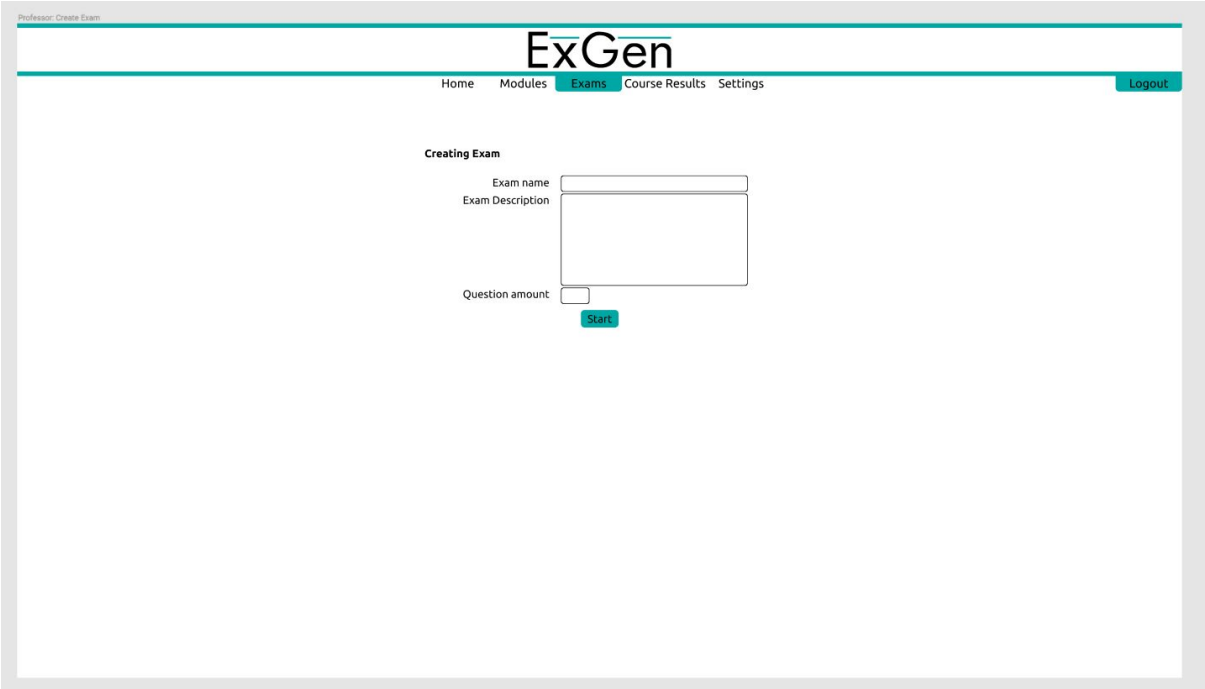
The professor will have an exam page where they can edit existing exams(enable/disable questions, rewrite questions etc), create a new exam, completely delete existing exams and view the exam questions. This page will look like this.



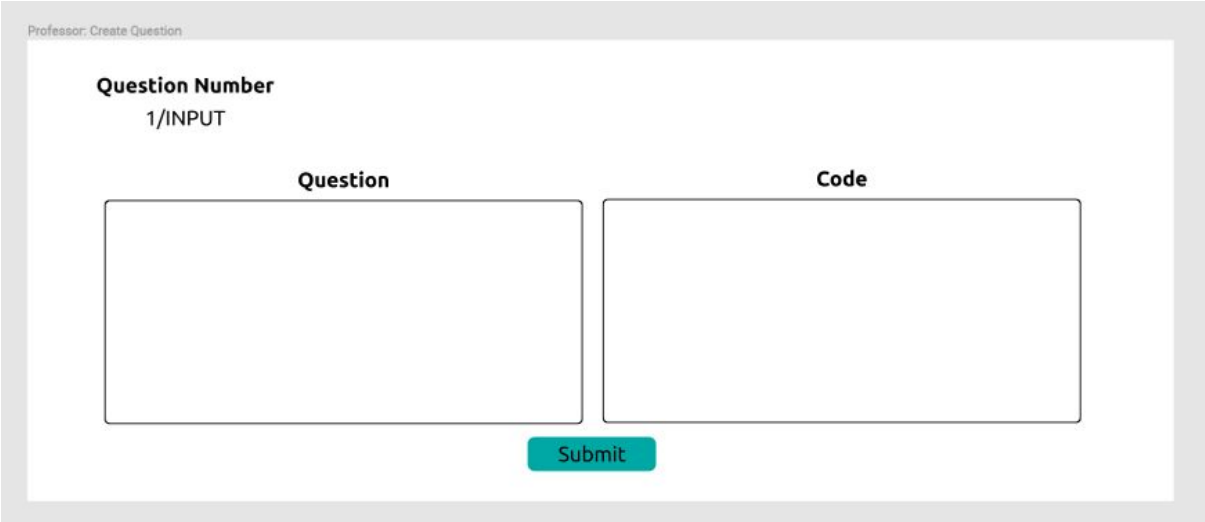
When the professor clicks the “Edit” button a new panel will appear over the questions. This panel is the editing question panel. It will be the panel where the professor enable/disable and edit questions. This panel looks like this. The question would appear in the “question” box and the code for that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

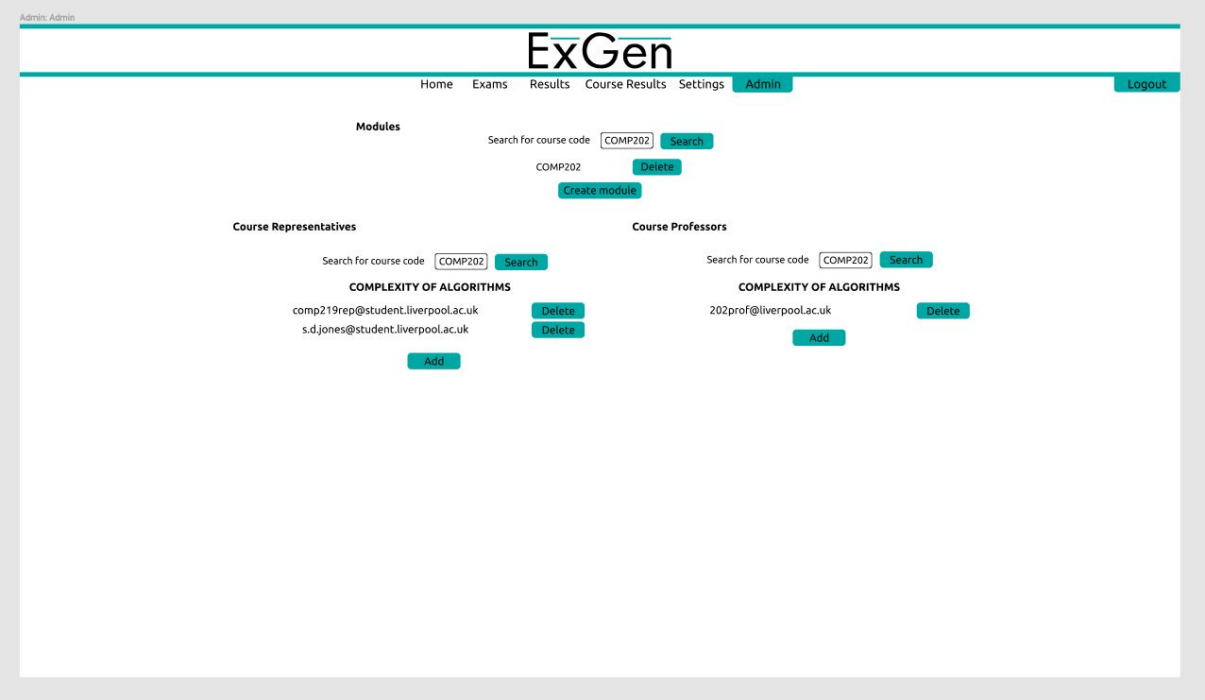


When the user presses the “Start” button a new panel will appear. This panel represents one question out of the question amount. This panel has two fields, a question field and a code field. Once these fields have been filled the user will press submit and the system will clear these fields and ask for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



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.
Dictionary (also known as a hash)	answers	A dictionary in which each generated answer is a key, and the value associated with the key is a boolean representing whether or not the answer in the key is correct.	To hold the generated answers to the questions and to check if the answer clicked by the user is correct.

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 text file path

- serves as a wrapper and invokes the other functions in the file to generate the final question

- returns: A tuple with the question with values instead of variables and the dictionary of answers for that question

```
define function getQuestion(takes path)
        define questionInput as the output of readFile when passing
path
        assign the output of the user's code for generating the
variables and asnwers as the tuple varDict, answer

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

```
        create variable answers that holds the output of
shuffleAnswers when passing answer

        return question,answers
```

---

shuffleAnswers:

- takes: dictionary of answers
- even though dictionaries have no inherent index or order, in our tests the correct answer consistently appeared at the same spot in the dictionary, so this method turns it into a list, shuffles it and then turns it back into a dictionary
- returns: shuffled dictionary of answers

```
define function shuffleAnswers(takes answers)
        turn answers into a list and assign it to answerList

        shuffle answerList

        turn answerList into dictionary and return it
```

---

writeAnswers:

- takes: dictionary of answers
- returns a string with the answers in better formatting for the terminal
- returns: shuffled dictionary of answers

```
define function writeAnswers(takes answers)
        create list returnString

        shuffle answers using shuffleAnswers

        print the dictionary as is

        for each answer:
             append answer to returnString
        return a string concatenating every item on the list with a
‘\n’ between them
```

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: correct answer
- Generates wrong answers based on the correct one
- returns: a dictionary of answers

```
getAnswers(takes correctAnswer)
        creates dictionary answers
        while length of dictionary is less than 4:
              define variable deviation as a random integer between
positive and negative 25% of the correct answer
              if deviation is not 0:
                   add the correct answer plus deviation to the
dictionary as an incorrect answer
        add the correct answer to the dictionary as the correct answer
return dictionary answers
```

---

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
```

---

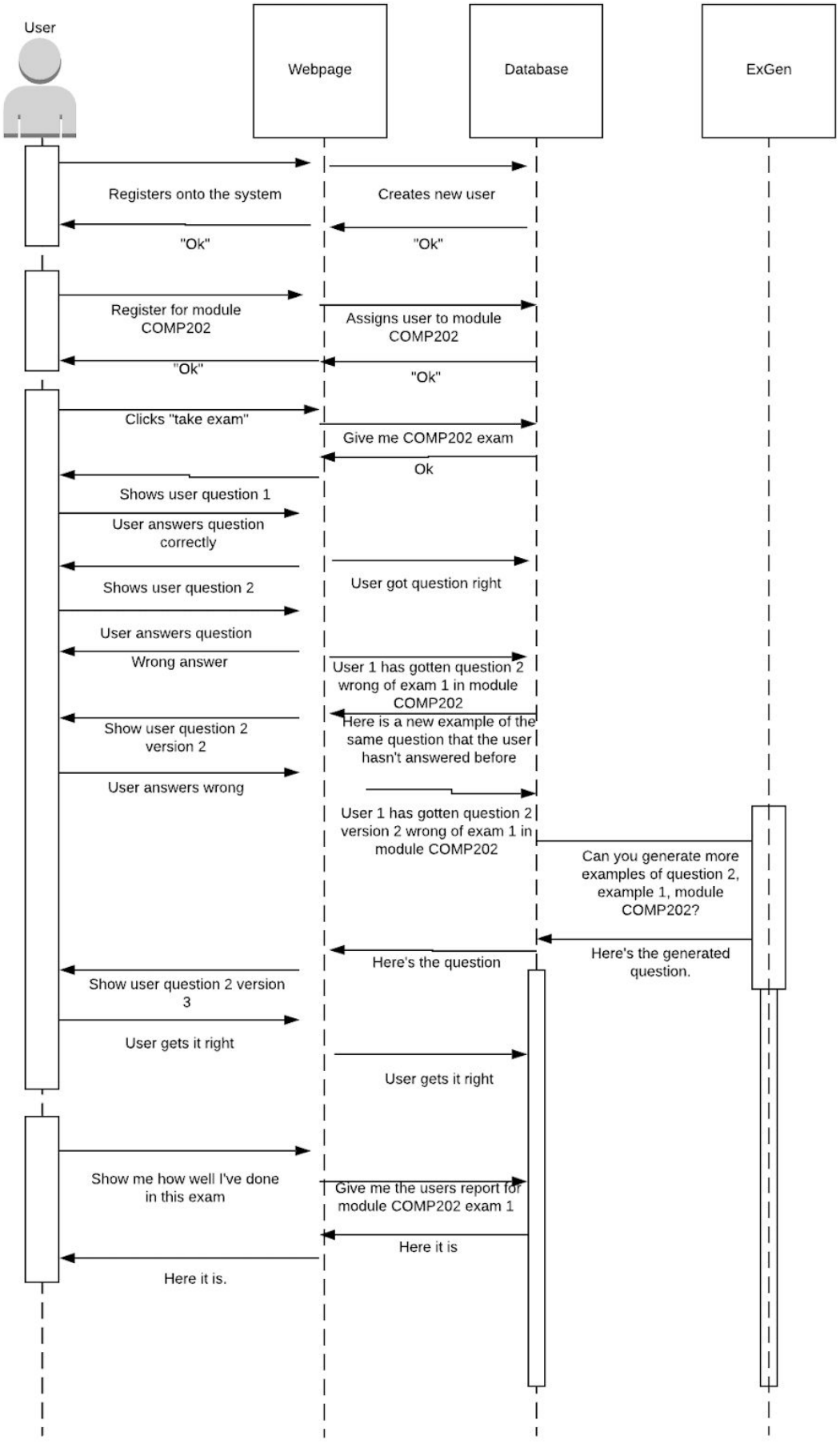
generate:

- 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

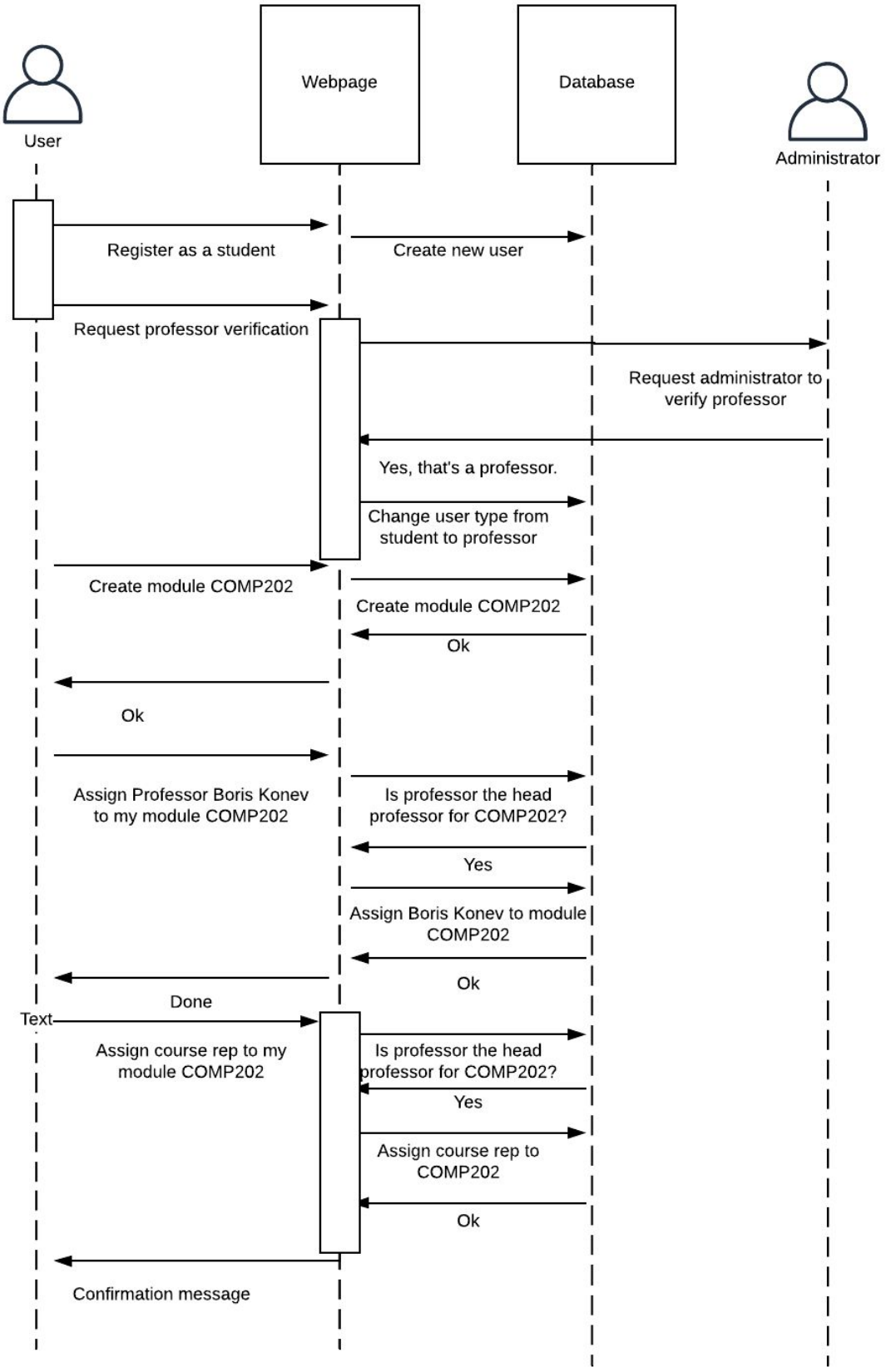
```
define function generate(takes varDict)
        update varDict using generateNumbers
        return a tuple containing (the updated varDict, the output of
getAnswers when passing the output of add when passing varDict)
```

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
User got question right
Shows user question 2
User answers question
Wrong answer
User 1 has gotten question 2 wrong of exam 1 in module COMP202
Show user question 2 version 2
Here is a new example of the same question that the user hasn't answered before
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 question
Here's the generated 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.

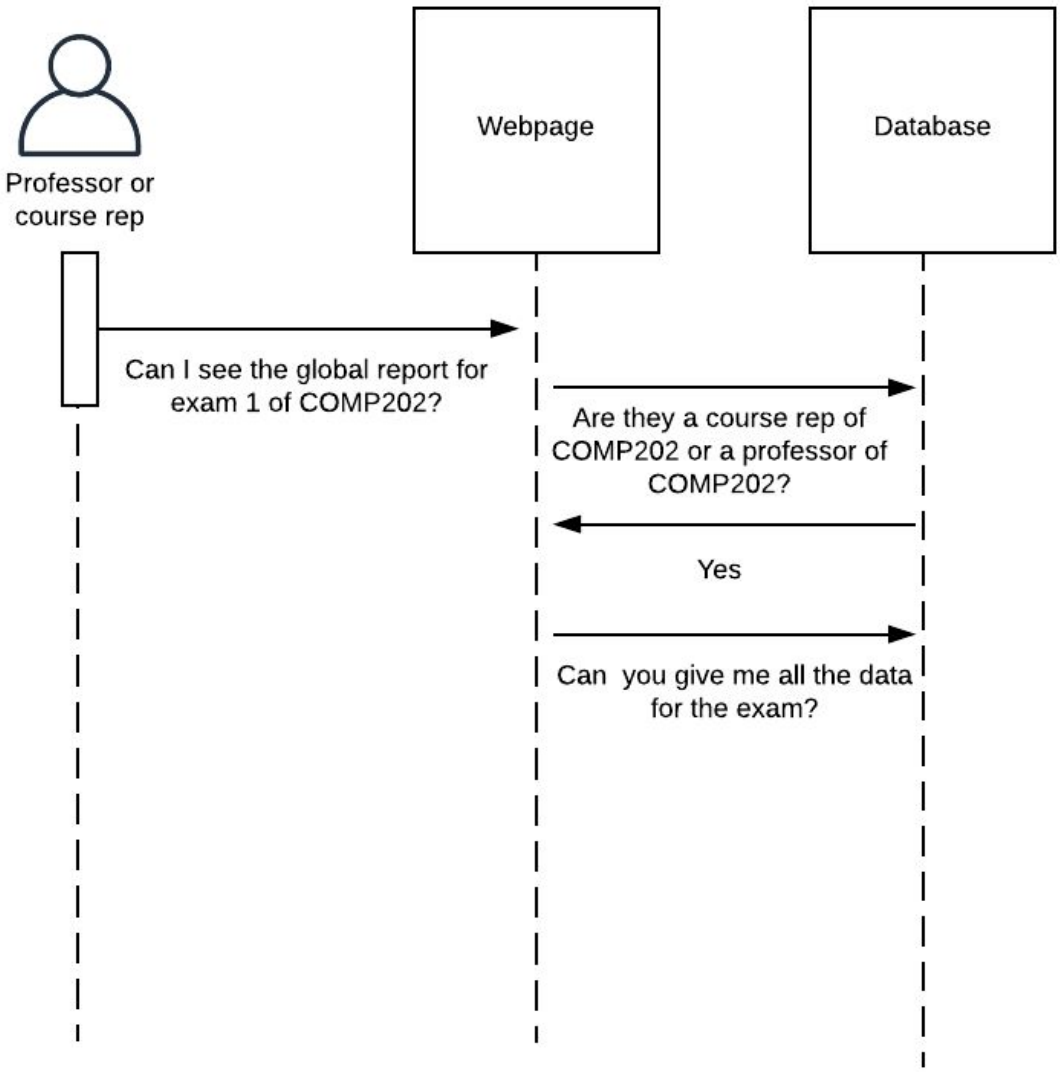
In reality one might use Liverpool life to verify users and assign them to modules. We are not using it as we want to make this software as easy as possible for anyone to implement.


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

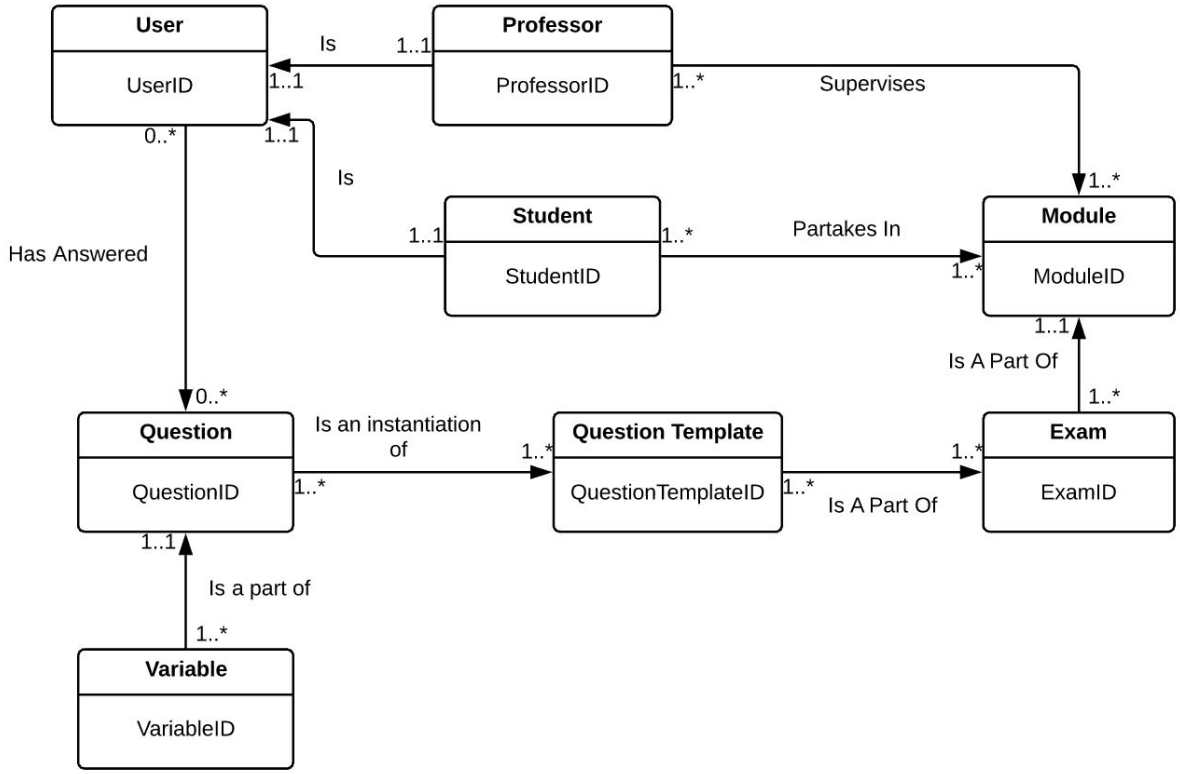
If the student has been verified as course rep (by a professor) then any modules they are assigned to they will automatically become course reps.



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

Bibliography

Exploreflask.com. (2019). *Patterns for handling users — Explore Flask 1.0 documentation*. [online] Available at: https://exploreflask.com/en/latest/users.html [Accessed 9 May 2019].

Flask.pocoo.org. (2019). *URLs with Payload | Flask (A Python Microframework)*. [online] Available at: http://flask.pocoo.org/snippets/50/ [Accessed 2 May 2019].

Bcs.org. (2017). [online] Available at: https://www.bcs.org/upload/pdf/cop.pdf [Accessed 14 Jan. 2019].

Grinberg, M. (2019). *The Flask Mega-Tutorial Part I: Hello, World! - miguelgrinberg.com*. [online] Blog.miguelgrinberg.com. Available at: https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world [Accessed 9 May 2019].

Pegarella, S. (2019). *Sample Cookies Policy Template*. [online] TermsFeed. Available at: https://termsfeed.com/blog/sample-cookies-policy-template/#Download_Cookies_Policy_Template [Accessed 9 May 2019].

Pythonhosted.org. (2019). *Features — Flask-Security 3.0.0 documentation*. [online] Available at: https://pythonhosted.org/Flask-Security/features.html [Accessed 4 May 2019].

uncommon?, W. and Henderson, A. (2019). *Why is client-side hashing of a password so uncommon?*. [online] Information Security Stack Exchange. Available at: https://security.stackexchange.com/questions/53594/why-is-client-side-hashing-of-a-password-so-uncommon [Accessed 1 May 2019].