ExGen - Next Generation Exams

Requirements documentation
Group 11

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

Date 14/02/2019

Table of Contents

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.

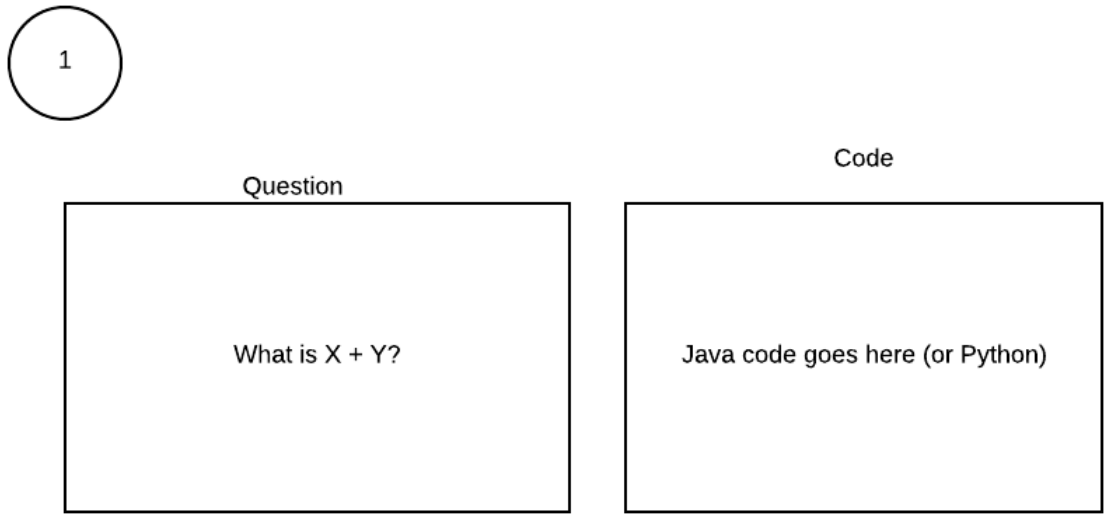
ExGen is a program to procedurally generate exams given an exam. Instead of giving your students one mock paper, give them ExGen which will automatically generate an infinite number of papers for your students to practice on.

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:

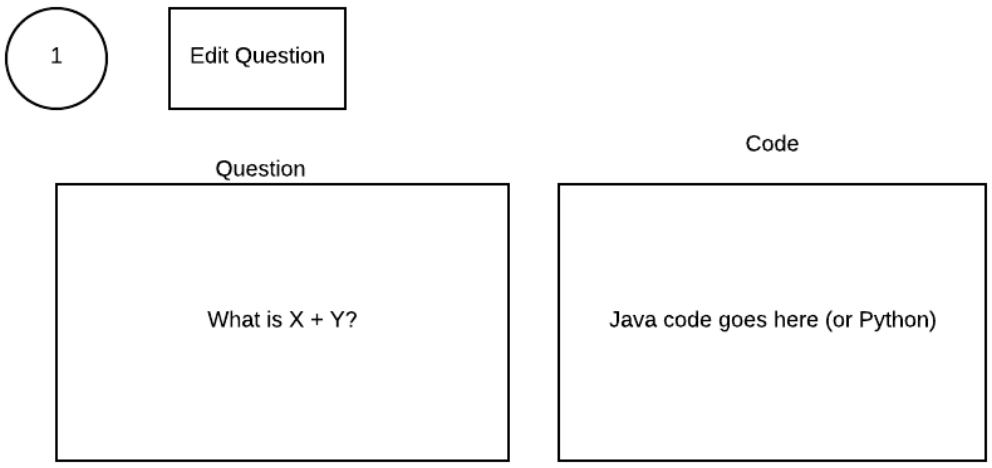


The question is LaTeX, the code is Python or Java.

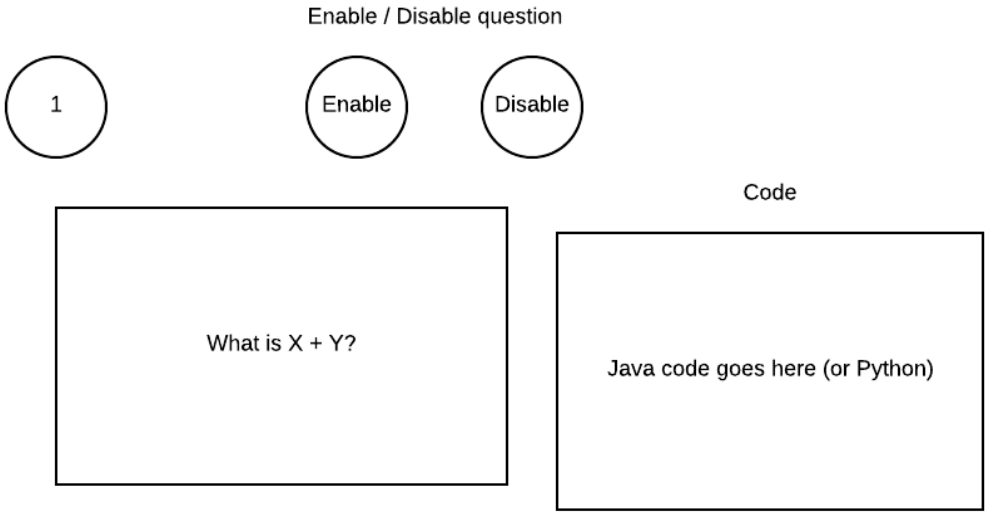
[1] https://www.un.org/sustainabledevelopment/sustainable-development-goals/
[2] https://www.un.org/sustainabledevelopment/education/

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 queston.



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

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

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

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. This is where sandboxing comes into play.

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

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

Stack

The technology stack is:

Frontend

Backend

Running on

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.

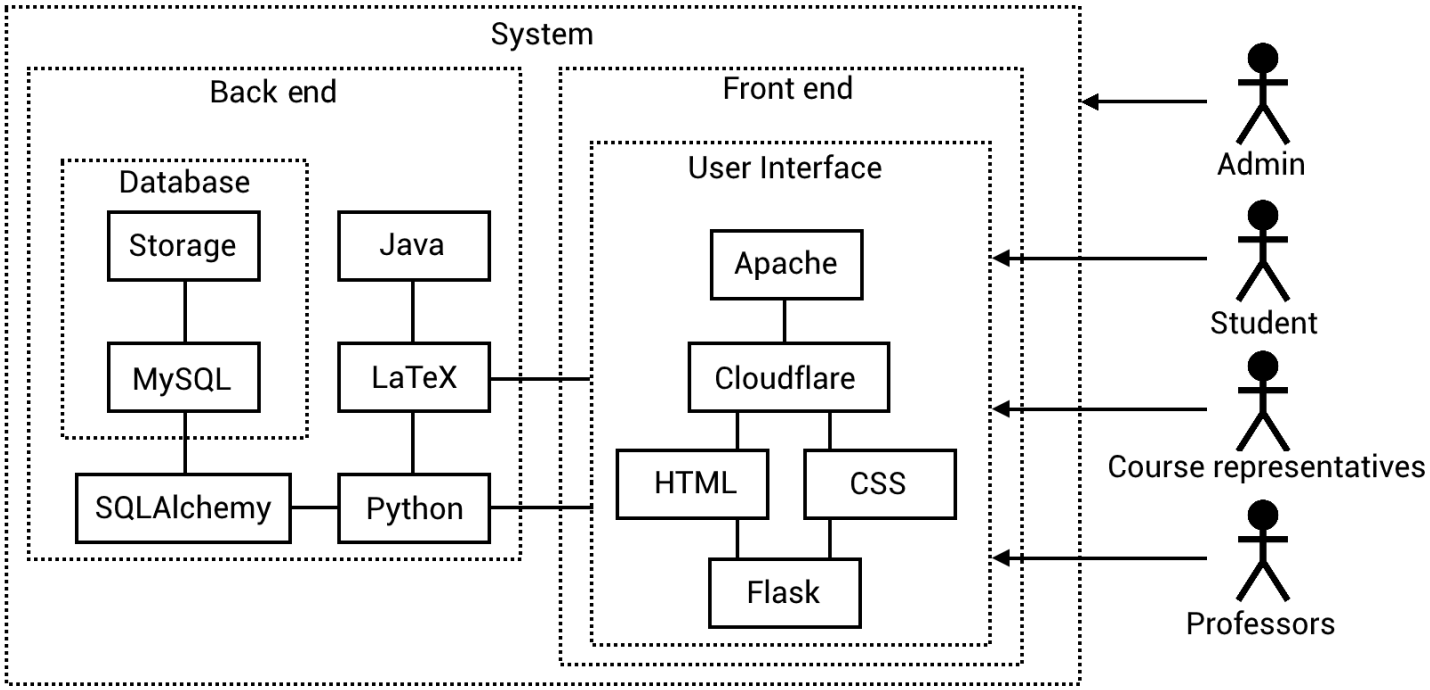
[4] https://www.liverpool.ac.uk/intellectual-property/

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:

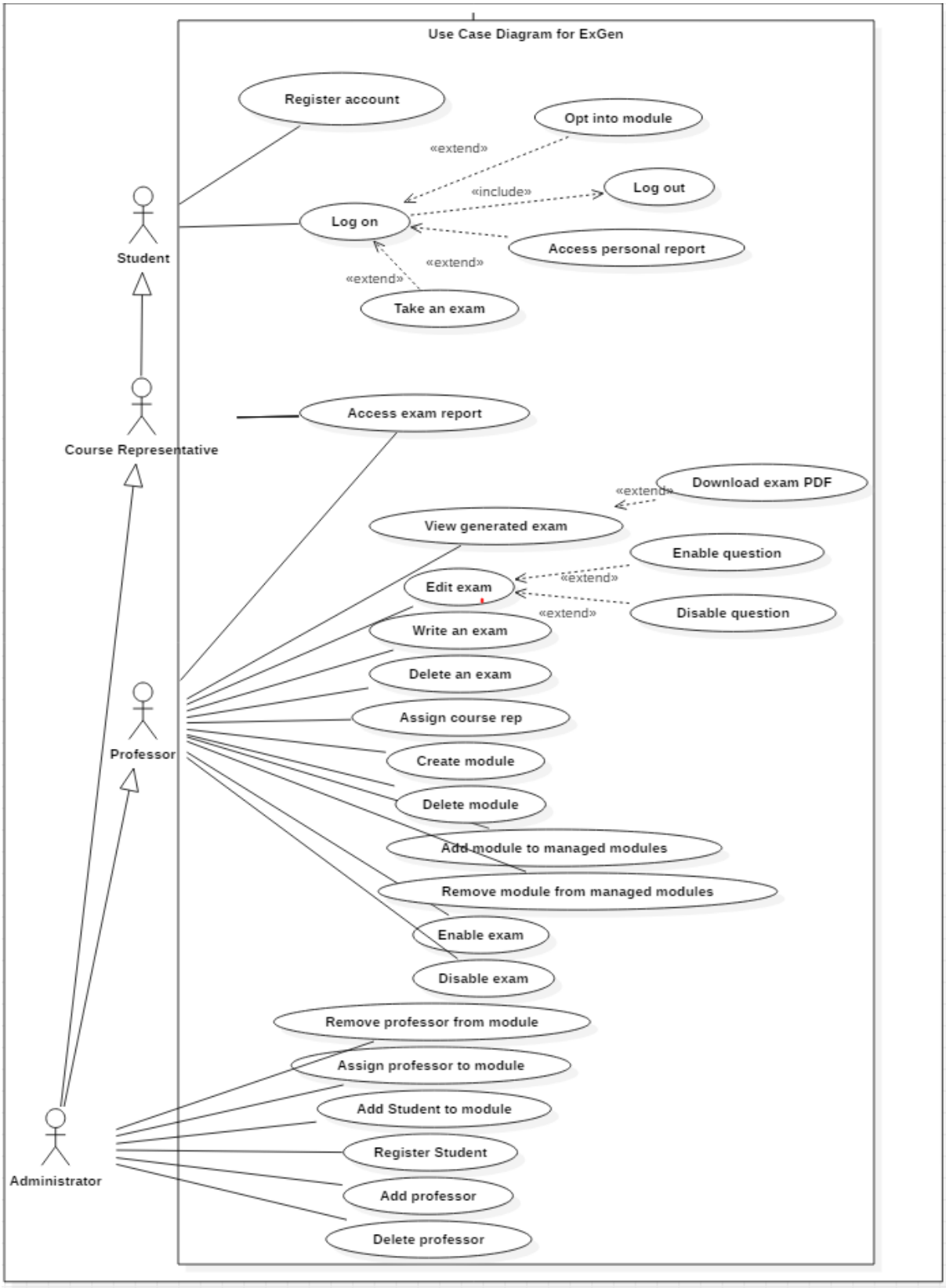
1. Student
2. Course representatives (or external verifiers)
3. Professors
4. Administrators

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

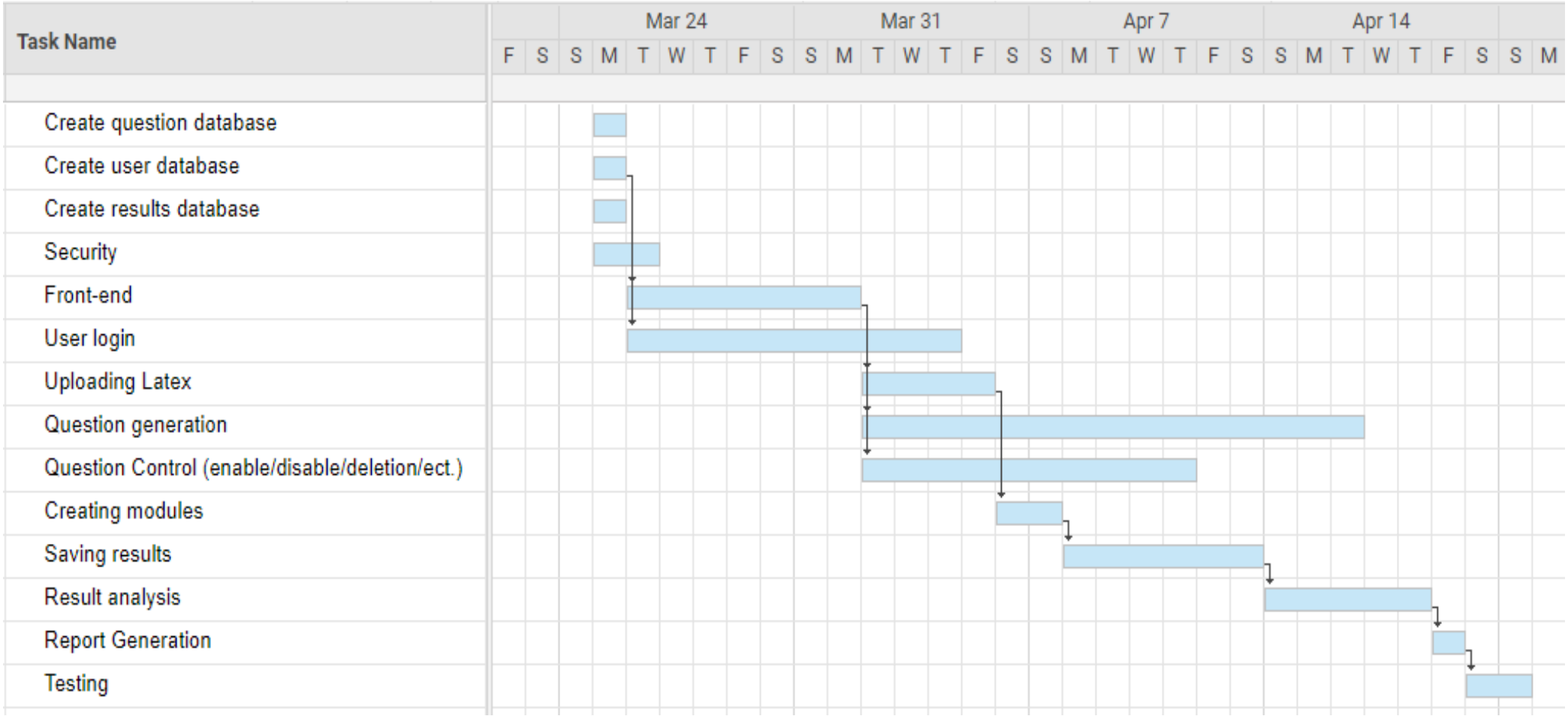
The Administrator has full control over the database and therefore can do all of the above but they are the only users to be able to directly access the database and may change any field. They are also the only users able to add a professor, delete a professor and delete a module.

Gantt Chart

Tasks

ID	Task Name	Duration	Start	Finish	Notes/Comments
1	Create question database	1d	03/25/19	03/25/19	
2	Create user database	1d	03/25/19	03/25/19	
3	Create results database	1d	03/25/19	03/25/19	
4	Security	2d	03/25/19	03/26/19	Should be done at same time as everything else, as security is a continual thing
5	Front-end	7d	03/26/19	04/01/19	Has to be done after database and before uploading LaTeX. Independent of generating questions
6	User login	10d	03/26/19	04/04/19	Must be done after database
7	Uploading Latex	4d	04/02/19	04/05/19	Is independent of database
8	Question generation	15d	04/02/19	04/16/19	
9	Question Control (enable/disable/deletion/ ect.)	10d	04/02/19	04/11/19	Can be done indp of everything but preferably at same time as latex
10	Creating modules	2d	04/06/19	04/07/19	Has to be done after database + latex + user login
11	Saving results	6d	04/08/19	04/13/19	Has to be done after database + latex + user login
12	Result analysis	5d	04/14/19	04/18/19	Has to be done after saving results (and all of its children)
13	Report Generation	1d	04/19/19	04/19/19	Has to be done after result analytic (and all of its children)
14	Testing	2d	04/20/19	04/21/19	Should be done at same time as everything, as

Gantt Chart


Task Name
Mar 24
Mar 31
Apr 7
Apr 14
F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M
Create question database
Create user database
Create results database
Security
Front-end
User login
Uploading Latex
Question generation
Question Control (enable/disable/deletion/ect.)
Creating modules
Saving results
Result analysis
Report Generation
Testing

Conclusion

This document has been signed by all members of the team. These members are:

- Brandon Skerritt
- Luke Aldersley
- Yales Rios
- Steffan Drosinos Jones
- Wen Geng Ong
- Claire Mulholland

On the date 14/02/2019.