IMY 772 Project Phase 1 Growth Learning Management System (LMS)

Charles Claassen	15216498
Kenneth Mangwane	15183379

Table of Contents

Introduction

We are creating a system called Growth LMS. It is an online learning management system. The purpose of this system is to make it easy for lecturers to manage their course content and give the students easy access to the content. This system will provide lecturers with several tools to help them with the creation and management of the course content. This system allows lecturers to communicate with students via online announcements. The lecturers can create announcements that will be visible to all students to view at any time. Lecturers can upload files to the system, for example, document, images, audio and video files. Lecturers will be able to create and organize a file structure, create folders and subfolders to organize the files they upload to the system. Lecturers will also be able to create quizzes to test the knowledge of the students and upload students' marks. Students can view and download any of the course content that they are registered for. Students can register by filling in and submitting an online form and if all details are suitable an account will be created for them. The student accounts will be added to the courses they selected to register for. The lecturers for these courses will be able to view and manage the accounts of the students that registered for their courses.

We will make use of AWS web hosting services to host our system online. Also, we will use Amazon DynamoDB for a scalable and reliable database and use amazon S3 for storage of all our files. Our main language, that we will use, will be JavaScript and NodeJS, however, we will be integrating several APIs into our system.+

System Function

System Access

The system will allow access to a user who is registered on the system and provides a valid username and password. Each user will be distinguished by their roles. The system will have the following primary roles.

1. Super Admin, Which is responsible for high-level systems of the overall system and will mainly include the developers
2. Admin, This will be the person that controls all the access and all functionalities of the system.
3. HOD, This will be the person that controls the users and everything else within their assigned department.
4. Teacher, This will be the person that controls the users and everything else within their assigned module/subject

5. Student, They will have access to the assigned content and will be able to do operations that will be module/subject-specific
6. Tutor, They will have access to functionalities decided by the Teacher.

The system admin can also add roles to the system and also specify the type of access they will have. Each user that can control other users can give access to functionalities that are within their role to other users. The user can also take away the power from lessor roles that are within their control.
Users can be added by a user who has a higher access level. Tutors and Students can not add users. If a person wants to be added as a user, they can apply to become a user and select the role they choose to occupy and the person with a higher access level can accept the application.

Student management

The system will allow the teacher and anyone with a higher access level to create and maintain student information. The students' management part of the system will allow those with the appropriate levels to create student groupings where a certain number of students can be grouped. The groups will allow the teacher to assign activities to the group instead of assigning to individuals. The grouping is also there to allow the teachers to distinguish between different sets of students with their niche.
Functionalities can also be given to a particular student group to enable them to have more power within the system than other users.

Module/Subject Management

The hod and anyone else with a higher access level can create and maintain a module/subject on the system and then assign people to the module. The assigned people can then go further to perform some other activities within the module/subject.

Grading

The system will also have functionality for specific user roles to allow them to add assessments to the system. The assessments on the system can of varying types such as exams, test or more depending on what the super admin set up. Once the assessments are set the student will know what assessment is expected from them and when it is due. The grading of the assessment can also be done online which will allow the student to see their grades online.

Communication

The system will allow everyone except the student to make announcements. The announcements can only be made to the people that have lower access levels than the

person who is making the assignment. They will be stored on the system and they will also go out by email to the recipients.
The system itself will also make its own announcements/notifications. They may be triggered because a quiz has been evaluated, or an application has been accepted or for many other reasons which may be mentioned in the descriptions of other subsystems.

Content management

Anyone with an access level higher than that of a student can put content in the system directed at the student. One can add content and organise content as they choose.

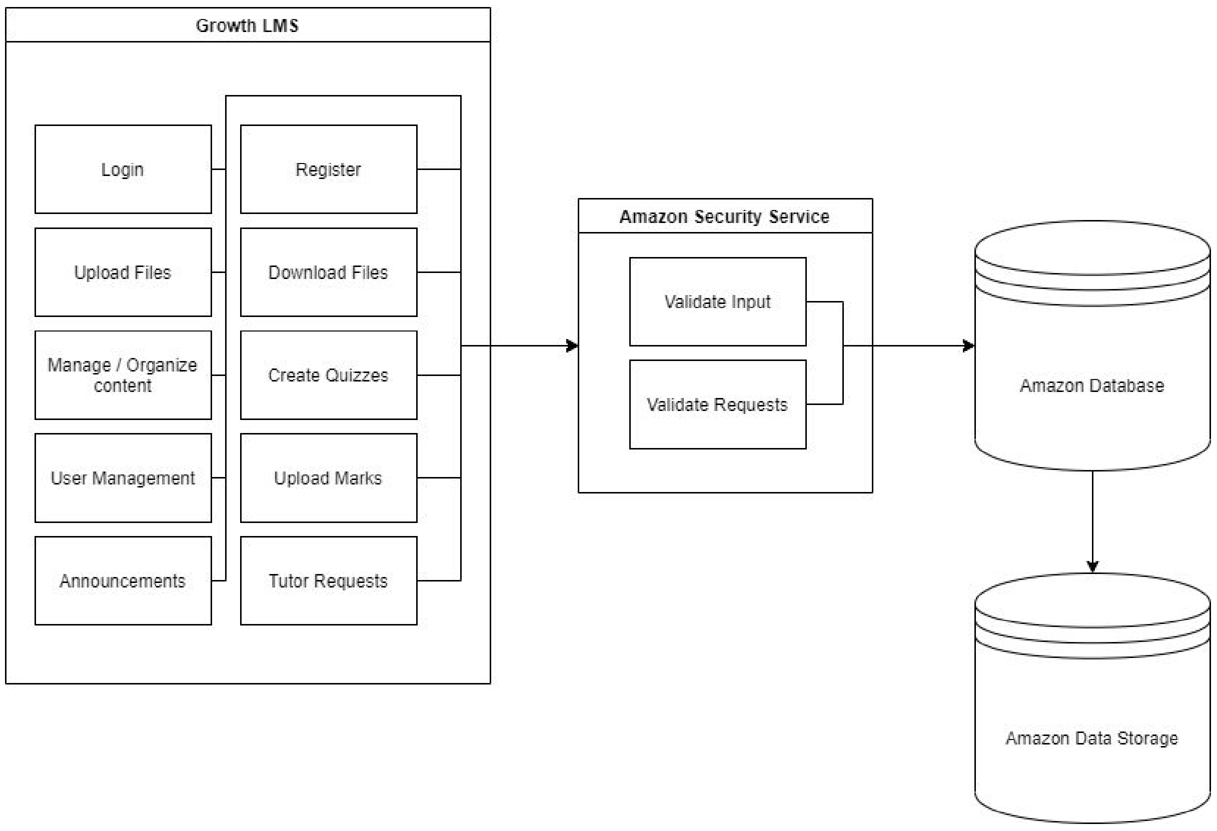
Quizzing

The system will allow teachers and anyone else who has a higher access level to create, edit and update a quiz. The quiz can then be assigned to students or a student group. The quiz can then be manually evaluated by the teacher or marked automatically depending on the settings on the system. The mark will be communicated to the student after the evaluation is complete. The mark will also be added to the grades as a quiz is another type of assessment.

System design

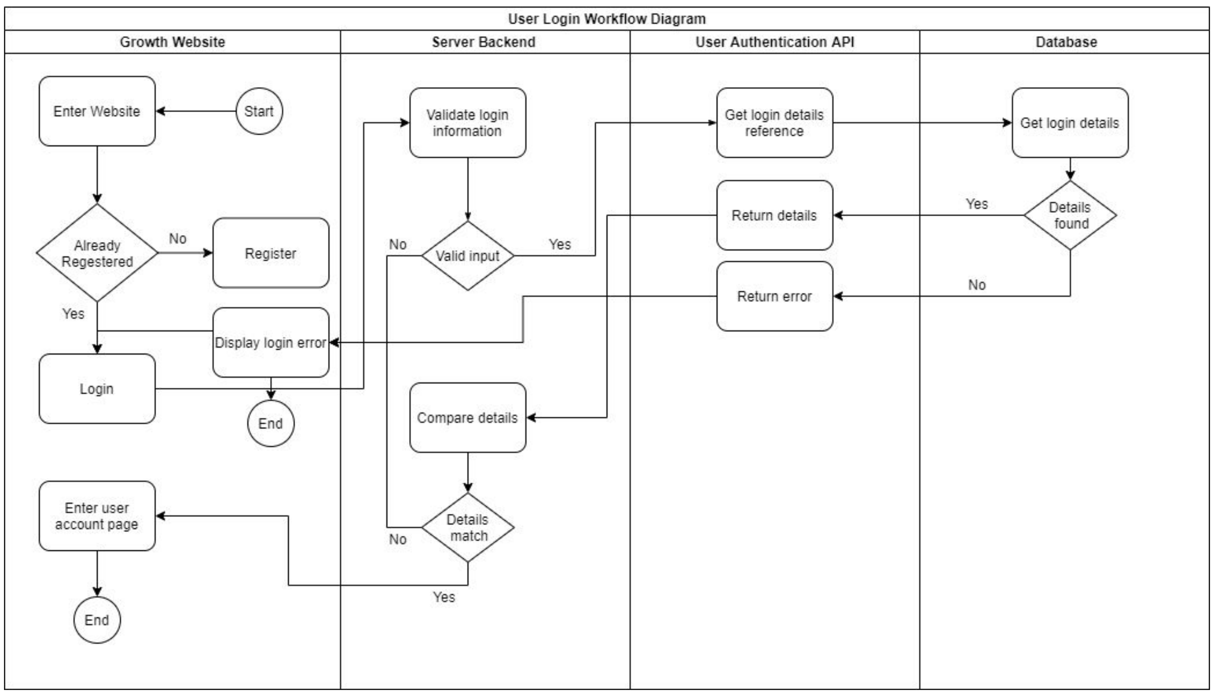
Our system will integrate several Amazon Web Services into our system. Mainly, the web will use AWS security services to validate input and data access requests. We will use a database to store references for files and other data for quick access and queries. We will use AWS storage service to store our system files, other large files and data that is too big for our database. Everything will still have a reference to it inside the database. Therefore, the storage can only be accessed through the database and the database can only be accessed through the security service. This is to protect our system against unauthorized data access.

System Overview Diagram



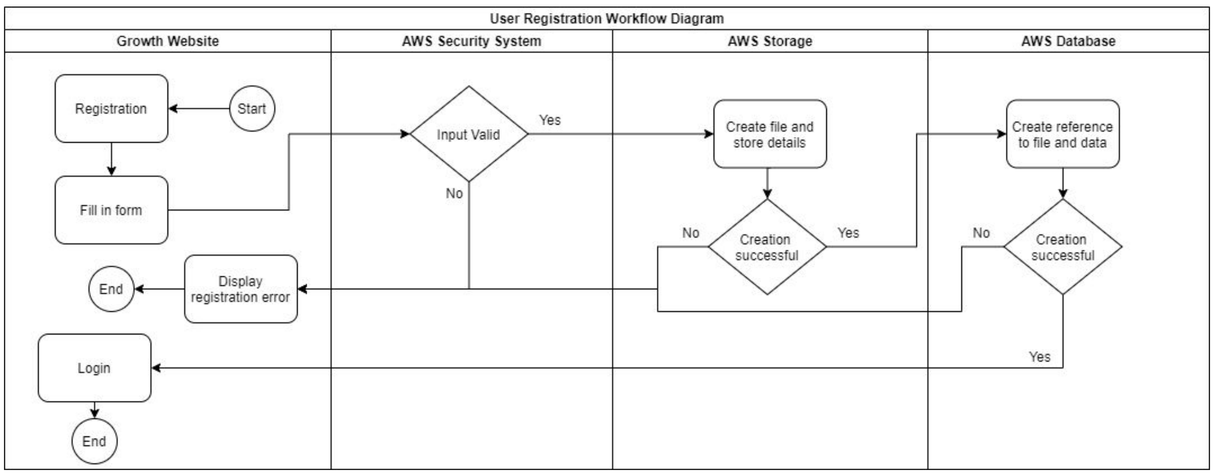
There are a few flowchart diagrams to demonstrate a couple of simple operations in the system and how they will interact with these interacted services.

User Login Diagram



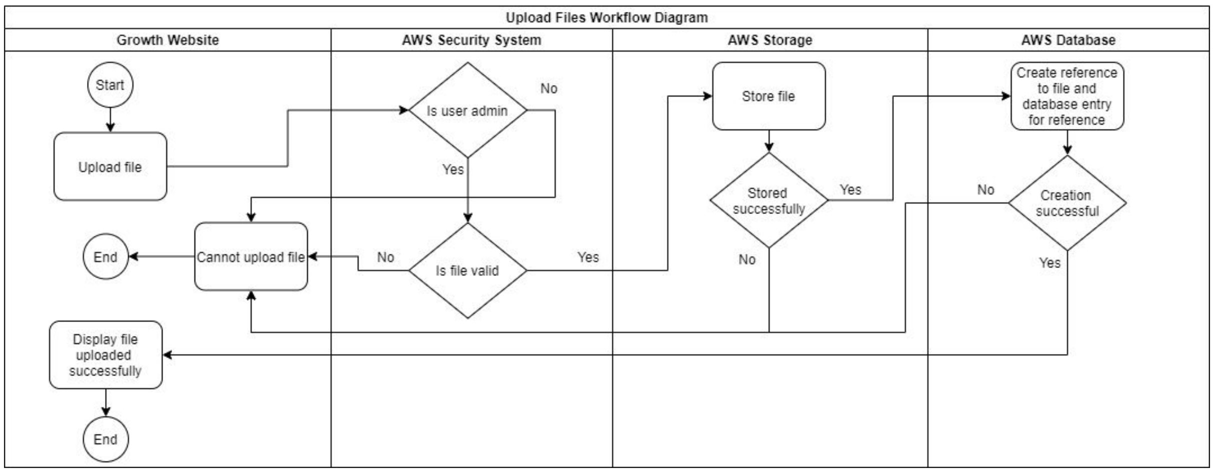
The login system is where the users have to enter a username and password. If the users do not already have an account they must first register. Once the login details have been submitted, the data is sent to the security services. Firstly, it will check whether the input is valid, if not, the security service will send a request to the database to get the login details of the given username. The data will be stored in a data file in the storage for security and the database will check if it can find a reference to the requested data. If the data was found it will be returned to the security service where it will compare the passwords. If it is successful the user will be logged in and sent to their user account page. If at any point something fails an error will be displayed to the user stating they cannot log in.

User Registration Diagram



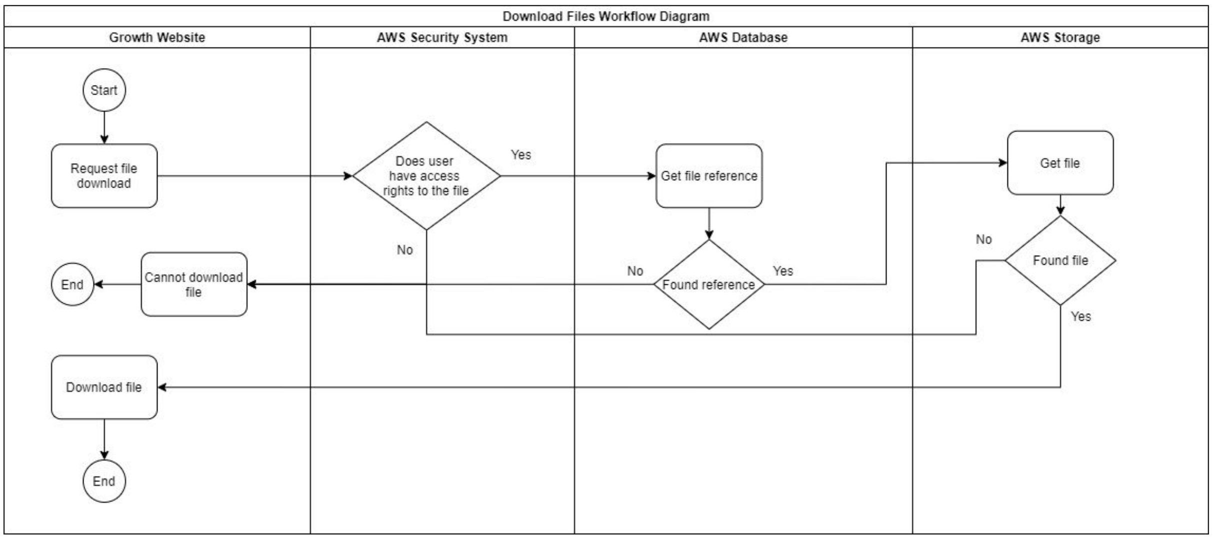
The registration works similarly to the login, however this time data is created and stored instead of being retrieved. The user fills in a form to register and once the form is submitted the data will be sent to the security services to validate the input. If the data is valid it will be stored in a file. Thereafter, an entry, to reference the file, will be created in the database and the file will be stored in the storage service. If everything is successful the user will be redirected to the login screen. If at any point something fails, an error message will be displayed stating that the user can not register.

File Upload Diagram



Uploading a file is also a simple process. The user requests to upload a file and then chooses a file to upload. The request is sent to the security service. Firstly, it will check that the user has the right privilege to upload a file, as not all users are allowed to do so. Secondly, it will check whether it is a valid file type to upload, only certain file types may be uploaded. If the file type is valid it is sent to the database where an entry and reference to the file is made. Thirdly, the file will be stored in the storage service. If everything was successful the user will get confirmation that the file was uploaded. If at any point something fails, an error message will be displayed stating that the file cannot be uploaded.

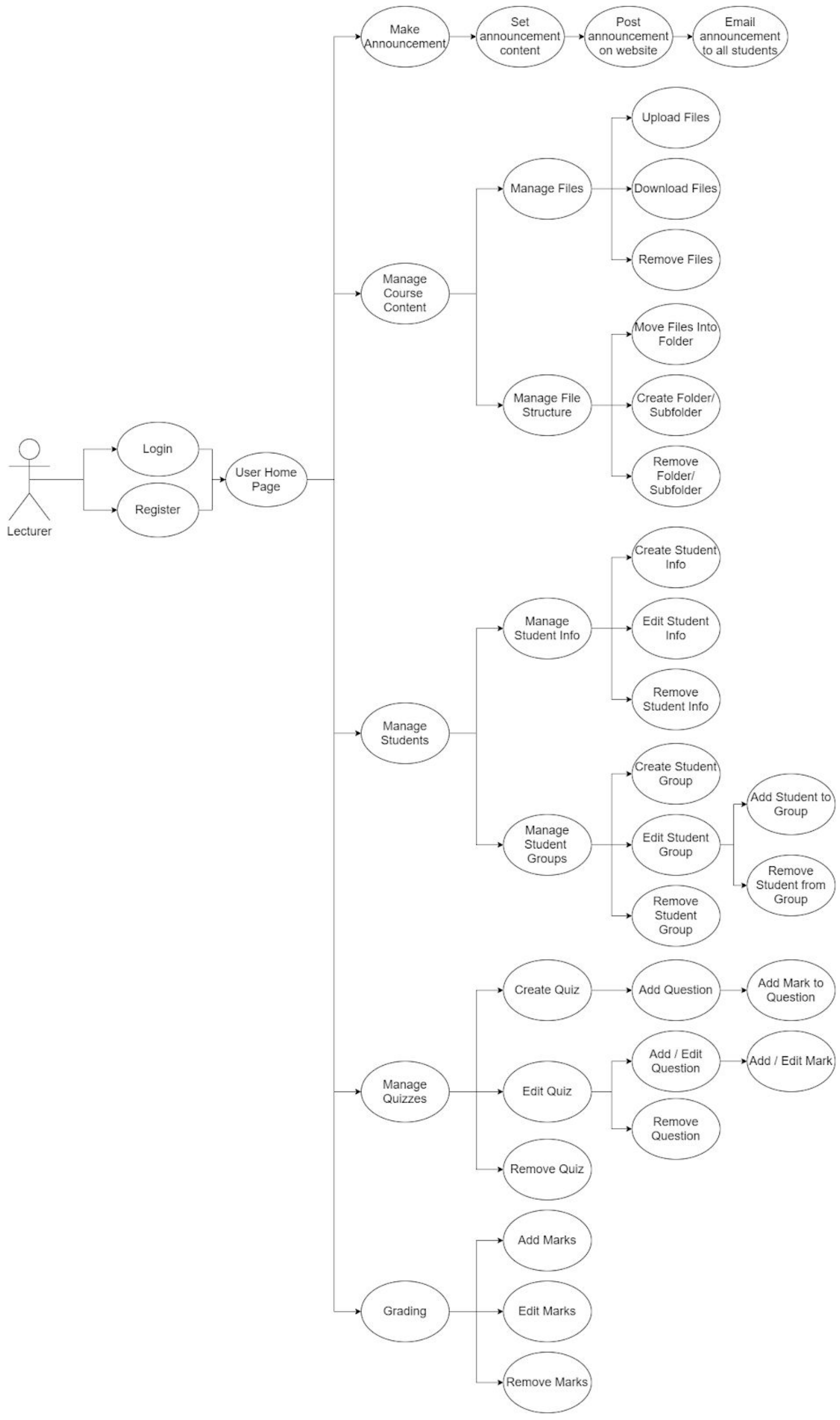
File Download Diagram



Downloading a file follows a lot of the same logic as uploading a file, however nothing is stored. The user requests to download a file. The request is sent to the security services that checks to see whether the user has access to the requested file. If the user does have the right, the database will be accessed to find the reference to the file ID. If the file was found the reference will be accessed to retrieve it from storage and once it is found it will be sent to the user as a downloadable file. If everything is successful the file will start to download on the user’s browser. If something fails the file will not be downloaded and an error message will appear.

These are just a few simple operations that can be performed on the system. The following diagram is a use case diagram that outlines how a lecturer can interact with the system. This is to give an example of how it can be used.

Lecturer Use Case Diagram:



Lecturers must log in to the system or register if they have not already done so. They will then be directed to their user homepages, from where they can perform various tasks. The tasks include making an announcement, managing their course content, managing their students, managing quizzes and grading.
To make an announcement the user will be provided with a text block wherein the announcement can be typed in and be submitted. Thereafter, it is posted on the course page under “announcements”. The announcements will also be emailed to all students registered for the particular course.

Lecturers can manage their course content by managing files and the file structure of the course content. They can upload new files, edit existing one or remove files. Furthermore, they can create and alter folders and move the files from one folder to another to better organize their work.

Lecturers can also manage all the students that are registered for their course. They can create or change students' information or create and alter groups for students. They can add and remove students to and from a group for assessment purposes.

Lecturers can also create quizzes for students to test their knowledge. The lecturers choose the questions for the quiz and how many marks each question will count.
Lecturers can provide marks for their students’ work that will be displayed to the students under the course’s grades section.

Technology

List of technologies

- Javascript
- NodeJs
- HTML, CSS, Jquery, Bootstrap (and possibly other common web development technologies)
- AWS amplify
- AWS lambda
- AWS E3
- AWS S3
- GraphQL

How will the technologies be used

The main technology that will be used to make the majority of the system is **Javascript**. We are going to incorporate some frameworks and tools to aid in making the backend and front-end of the application. The application will be built using **NodeJs** for the backend and **Graphql** for the database.

Nodejs will be used for coding the business logic and some of the API functions. We will also be using **AWS's amplify** API management to extend the API along with external API's. Other backend sub-systems of the system will be created separately and then applied through **AWS's Lambda service**.

The front end will be built using basic **HTML** and accompanying tools and frameworks to make it look nice. The whole system which includes the frontend and the backend components will be put inside an **AWS E3** virtual machine. The system will also include a content management subsystem which will require storage. The storage that will be used will be **AWS S3**.