ELEC3609 Deliverable 2
Design Specifications

LECREC

Alexander Woo Hyun Jung 310250811
Khanh Cao Quoc Nguyen 311253865
Kelvin D'Amore 430515051
Martin Groen 311182291

Contents

1 System Design Document

1.1 System Architecture Diagram

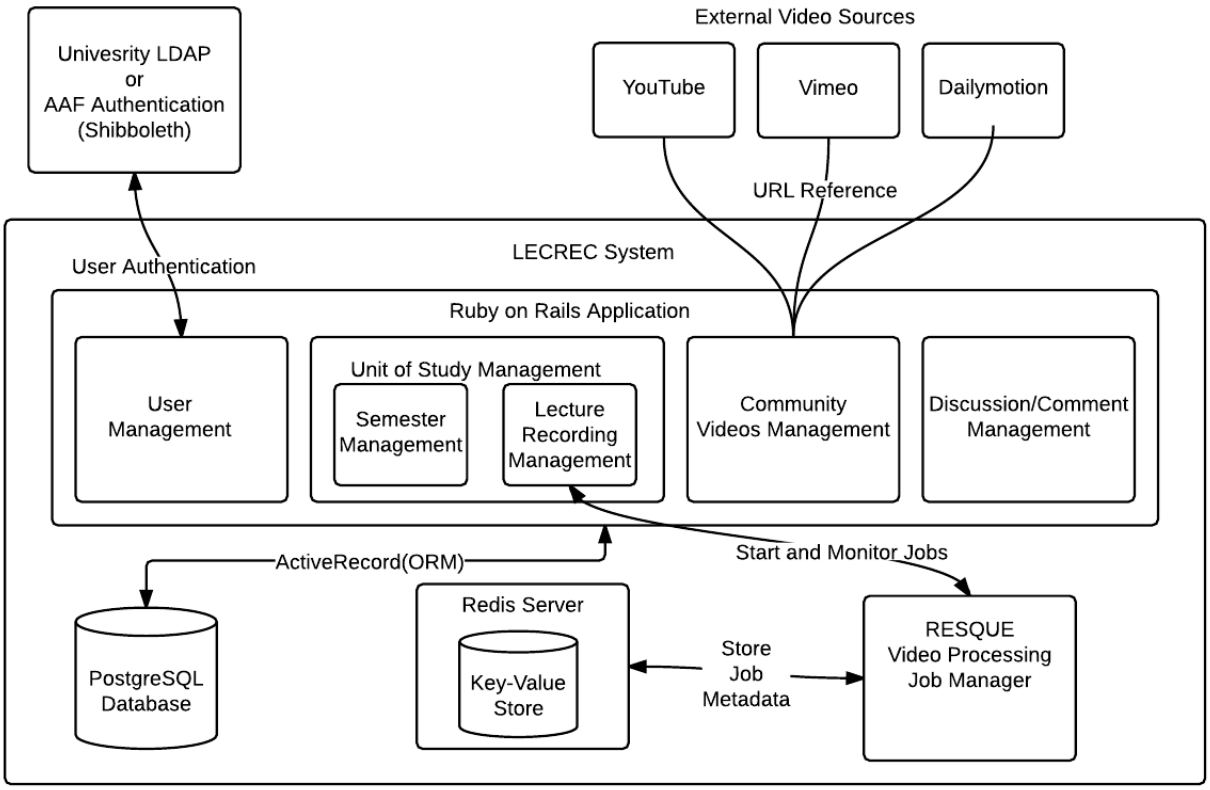

Figure 1: System architecture diagram of the LECREC system

Refer to the Package diagram (Figure 2) and the MVC diagram (Figure 15) for more information on the Ruby on Rails application and its internal workings.

1.2 Storage and Persistent Data Strategy

1.2.1 Database and Persistent

Since this application utilises an MVC framework, it is required that the models will persist with the use of relational database so we have decided to use PostgreSQL 9.3. Ruby on Rails provides tools to create and migrate the database and tables required through simple commands. It also provides us with an object relational mapping (ORM) called Active Record which does the majority of the querying. This not only speeds up the development of the application since we do not need to worry about manually writing prepared SQL queries but it will also prevent SQL injections.

1.2.2 Lecture Recording Storage

LECREC is not only required to be able to receive and store video content of lecture recordings, but also serves the videos on the application frontend. This will require the video to be stored in a location where the Apache web server can access and serve easily such as the public directory where Ruby on Rails serves the majority of it's assets such as images, styling scripts and JavaScripts.

These lecture recordings can get up to a varying range of gigabytes depending on lecture lengths and thus will need to utilise a server that can store and handle this data. The versions that is kept would be both the original video file and also the processed version which ensures that there is a backup in case a video needs to be reprocessed again or changed at a later date. This means even higher storage is needed to be able to hold both these data.

1.2.3 Temporarily Stored and Cached

As typical with most web platforms, temporarily stored and cached data will be utilised to ensure that positive user interaction can be implemented quickly and easily.

When the lecturer user uploads a lecture recording video to the application, that video will be initially cached during the actual upload as multiple fragment files in the temporary directory of the server until it's completion where it is moved to the appropriate location. The video is then processed into a compatible format if needed and that file will be stored alongside the original.

Another type of temporary data is the account cookie information which includes the session information to ensure that a user stays logged into the web platform among the different web pages. This will be stored on the user's computer itself and destroyed during sign out or when the browser is closed.

1.3 Trade-offs and choices

We chose Ruby on Rails as it is a framework that allows for rapid development. We also have some experience with working on it, which means we can spend less time setting things up and focus on the functionality of the application.

The original file will be saved even after the conversion into the specified file type (mp4), as we may require it to rerun the conversion if it fails, or an error occurs during processing. However, this means that we will need to dedicate more space to storing these video files, which may have a much larger file size than their compressed counterparts.

In our system, only certain video formats will be allowed for upload, to ensure compatibility in processing and ultimately making the video viewable on the web app. This will also help to prevent any malicious files from being sent to our server.

A concurrent job queue is used for processing videos as to be non-blocking, so that users can continue to use the system after uploading the lecture recording and not wait on the page, as it converts the file to a suitable format.

By using HTML5, we employ server sent events rather than polling. This is more efficient, as it requires less processing and calls to the server. Also, the HTML5 video player is compatible with most modern browsers and supports many video formats. Testing has shown that the player works well in Chrome and Firefox, but not very well in Internet Explorer. We are using bootstrap for styling - while it may look generic and boring compared to other applications, it is efficient and very responsive, which is an advantage when accessing the system on mobile devices.

1.4 Concurrent Processes

Concurrent processes can always be an area for concern for any type of development in terms of problems with race conditions, message output and inputs and lastly synchronisation. Due to this, the LECREC system will be designed to ensure as little use of concurrent processing and threads as possible. As such the only part that will utilise concurrency will be the upload lecture recording system of the platform.

The main reason why this has been elected for concurrency is so that a lecture recording can be uploaded to the website without affecting the main thread utilised for the user interface. This allows for appropriate usage for a user to be able to keep using the main functions of the web platform while the upload completes.

To implement this feature, we have decided to use Redis, an advanced key-value cache and store along with the RESQUE gem which is a Ruby library for creating background jobs, placing those jobs on multiple queues, and processing them later.

1.5 Package Diagram

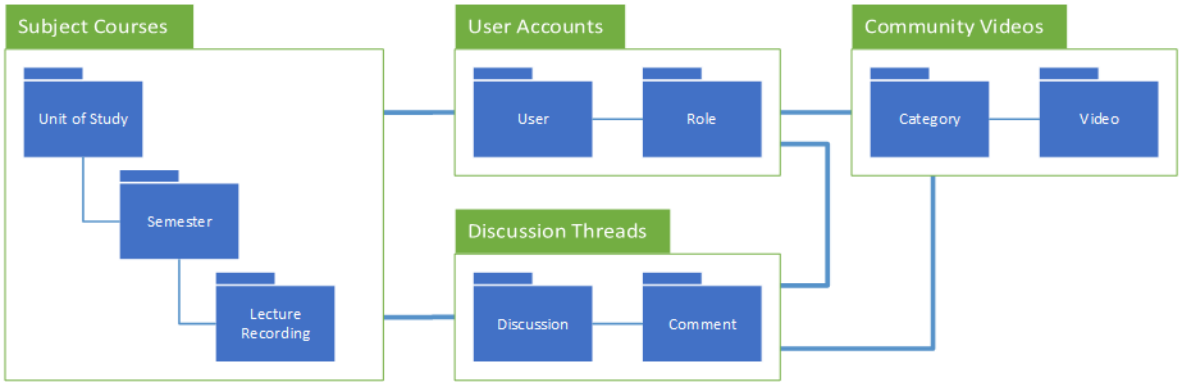

Figure 2: Package diagram of the LECREC system

The LECREC system contains four main packages which make up the complete function of the web platform project. The first package contains the main classes for the “Subject Courses” part of the system. This includes the Unit of Study of a particular course, each of which connects to a “Semester” class as all units are set with a particular semester, and then a “Lecture Recording” class so that each semester of a unit of study can contain a lecture recording.

The User Accounts package encompasses the information of classes for each user, this utilises the User databases and also roles such as whether a user is the administrator, lecturer or a student. The Discussion Threads package then outlines the discussion classes where a user can create a video timed based discussion of a lecture recording or also a comment through the comment class. Lastly the Community Videos package integrates the category class where a user can create a new category that they would like to discuss, and a video class so that embedded videos can be added for each category.

Overall, the system has a specific user account and roles for a particular user through the user accounts package. This package is then utilised by the subject courses package for displaying or editing units of studies and watching lecture recordings as well as by the Community Videos package where users are able to use their account information to create category based discussions. The Discussion thread is then utilised by the Subject courses package to create threads for lecture recordings and the community videos package to create category based discussion threads, posting as a user based on the information from the User Accounts package.

2 User Interface Layout

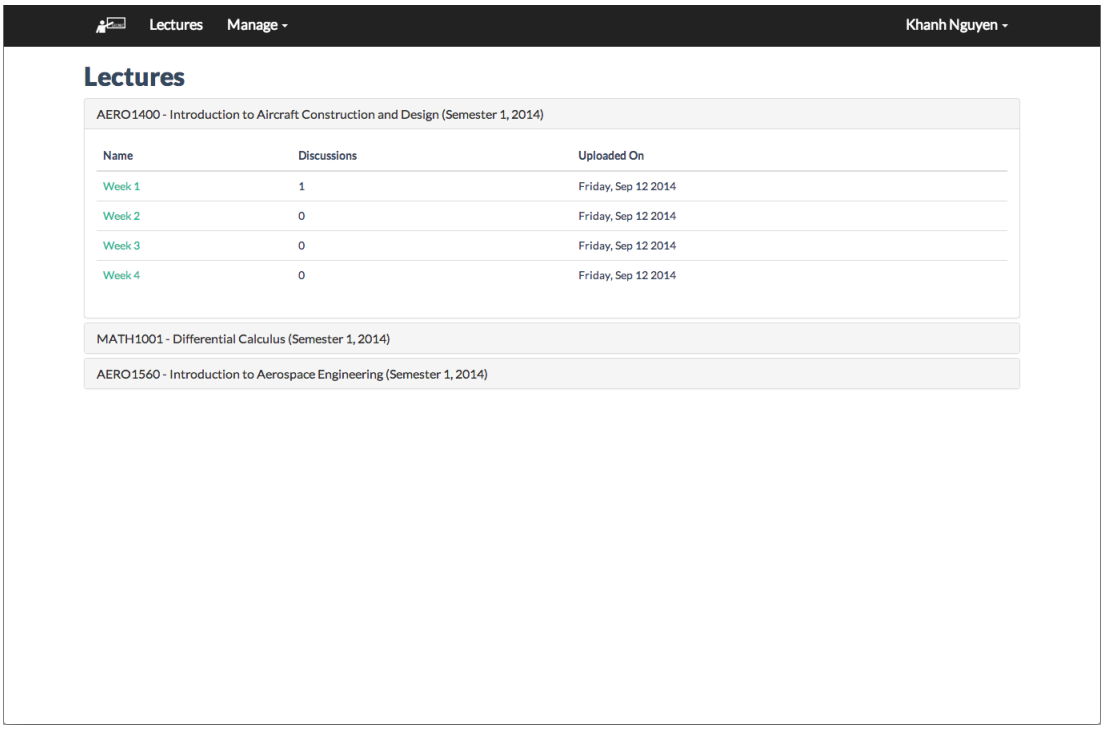

Figure 3: Page for lecture recordings under an enrolled subject

2.1 User Familiarity

The interface that we have designed is based around user-oriented terms and concepts, in this case, university oriented. We use concepts such as 'unit of study', 'lectures', 'semesters' which are university related, but also commonly used terms in modern web applications such as 'discussion', 'comments' and 'likes'.

2.2 Consistency and Minimal Surprise

We have chosen to use the Twitter Bootstrap frontend framework to style our application. This is a widely used framework used in many popular web applications today. Using this framework encourages a consistent and clean UI elements through the entire web application.

Another benefit of the Bootstrap framework, is that it also provides many built in functionality such as modals and alert JavaScripts, which will be used often in the application.

With the use of the Ruby on Rails application layout rendering, which renders a template for the headers and navigation links on every page and the use of DRY (don't repeat yourself) CSS, fonts, colours, button, icons and forms throughout the application this makes the web platform always consistent. The user should not be surprised by any styling that is not consistent with the rest of the application.

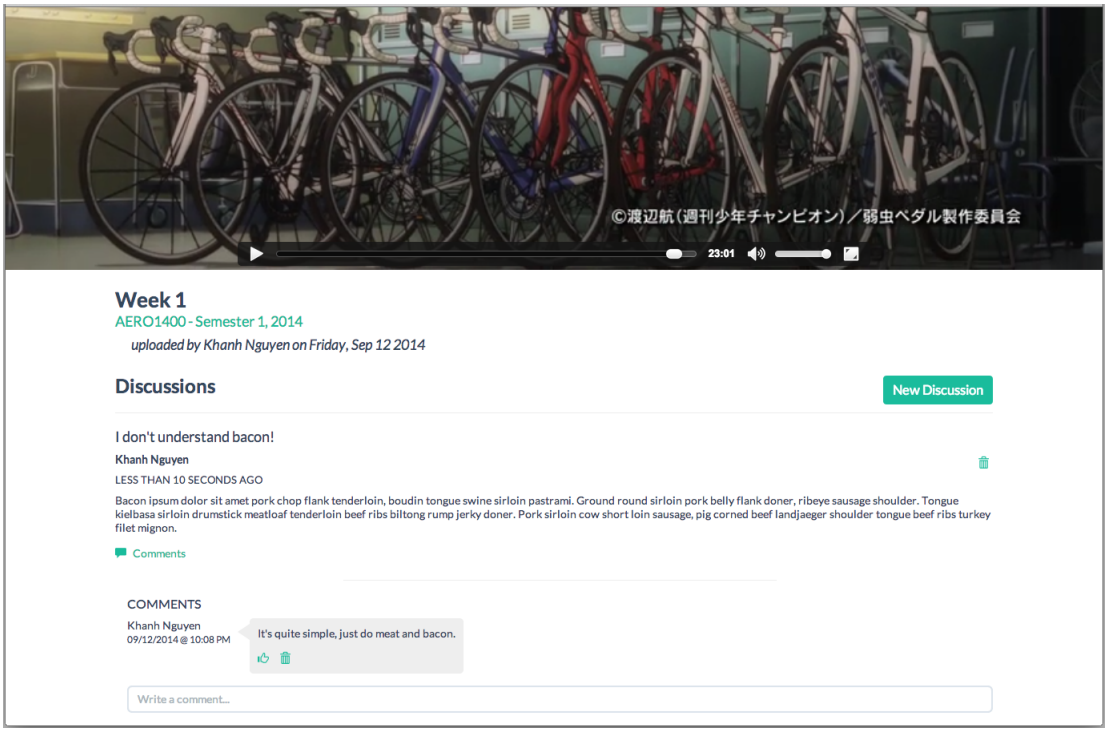

Figure 4: Page for lecture recording displaying discussions and comments

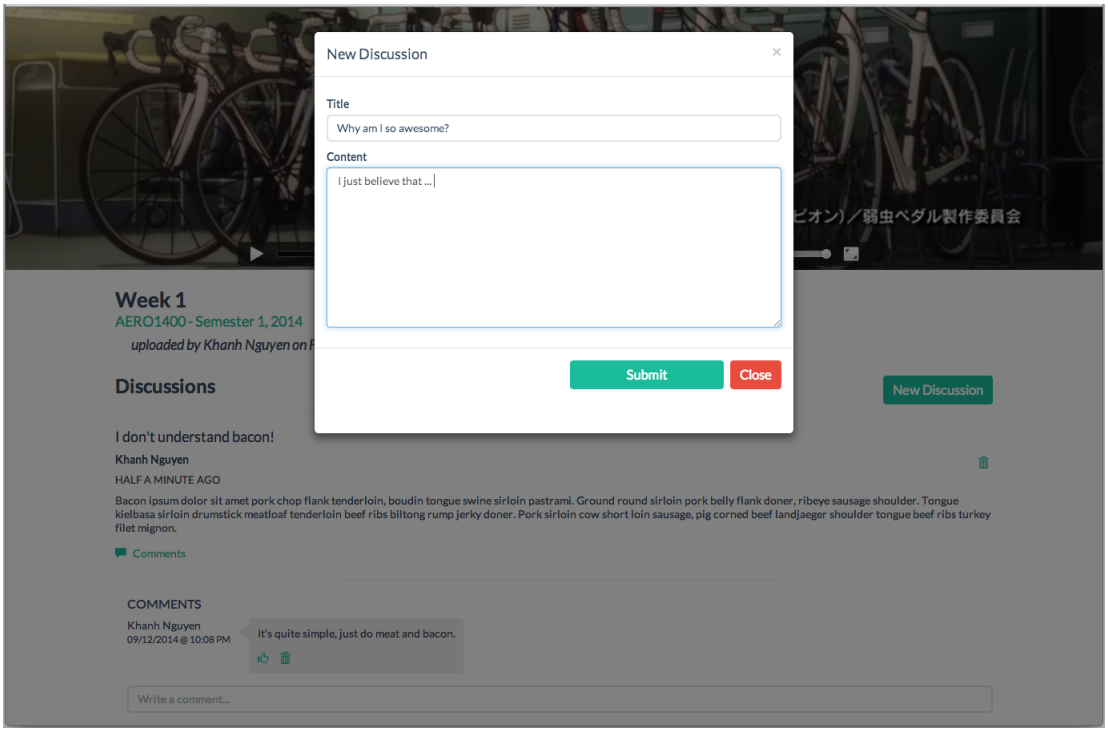

Figure 5: New discussion dialog on lecture recording

2.3 User Guidance and Recoverability

Ruby on Rails comes default with user guidance and error alerts. Additionally Bootstrap provides the alert styling with appropriate colours to match the messages (green for success and red for errors or warnings).

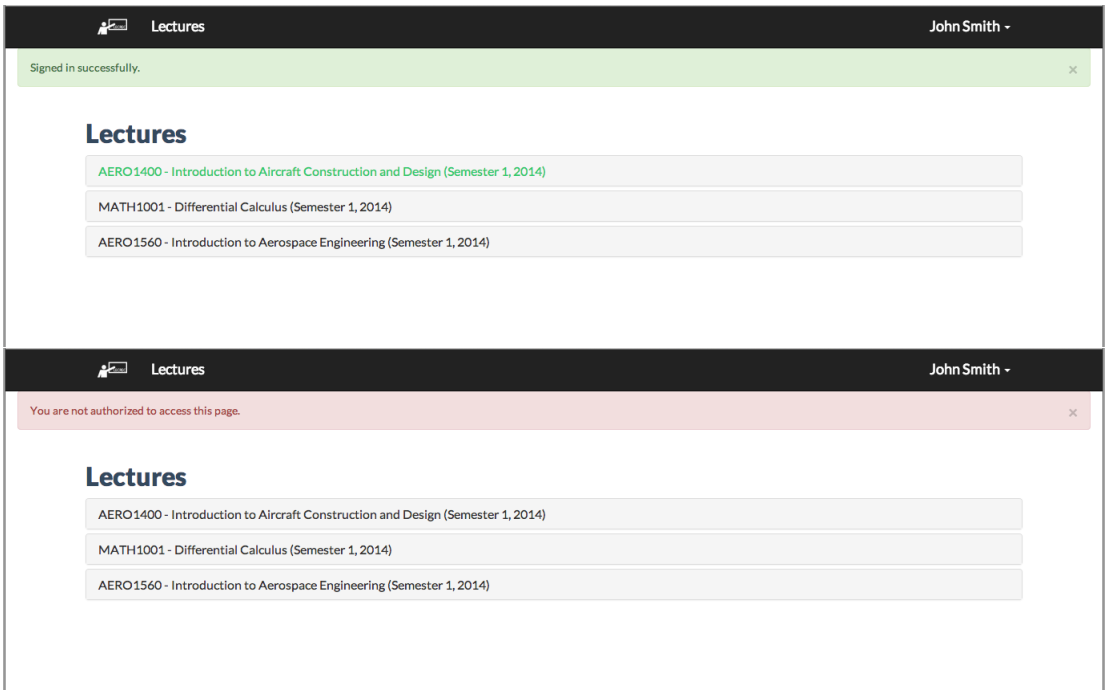

Figure 6: Page for enrolled subject outline demonstrating positive and negative feedback

2.4 User Diversity

Lecturers and System Administrators will have additional pages and functionality. The lecturer will be able to access the unit of study management of units they have access to and be able to access the lecture recording upload forms and discussion management panel. The system administrator will be able to manage every aspect of the application and will be provided with a variety of different forms and services.

We have also decided to use HTML validation on our form which will validate the forms dynamically on submit and will not reload the page if the form is not valid for submission. Error handling and recovery is handled by the Rails controllers or authentication and authorization modules such as Devise and CanCan.

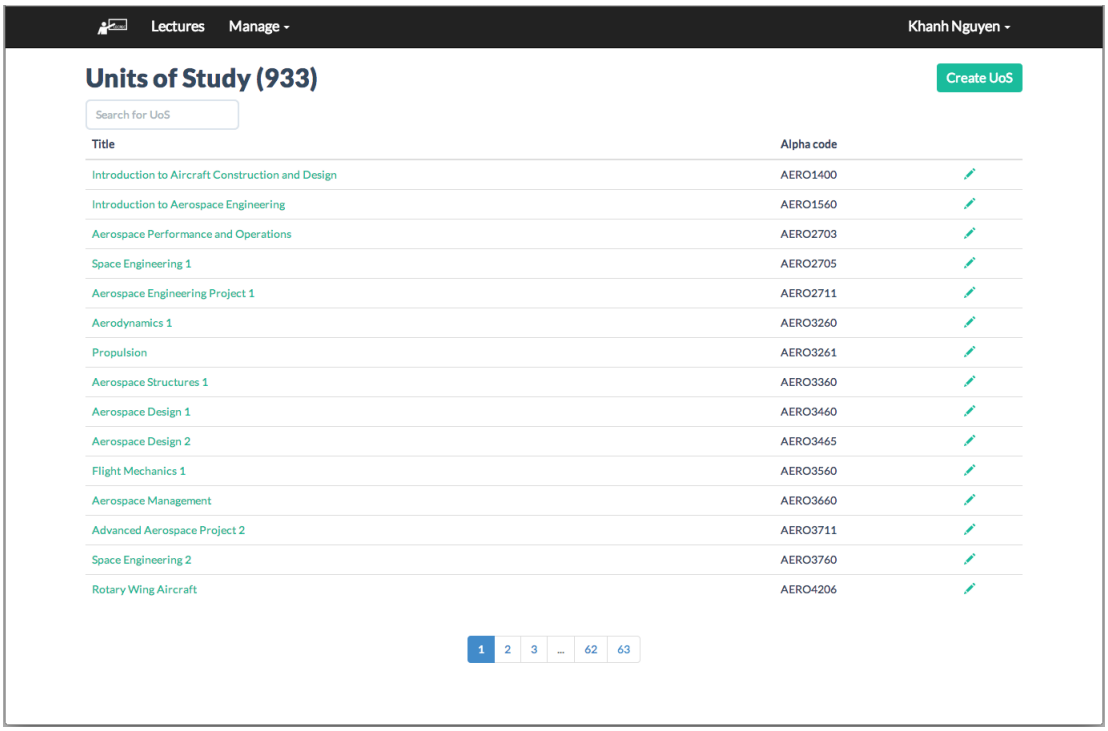

Figure 7: Tabulated list of units of study

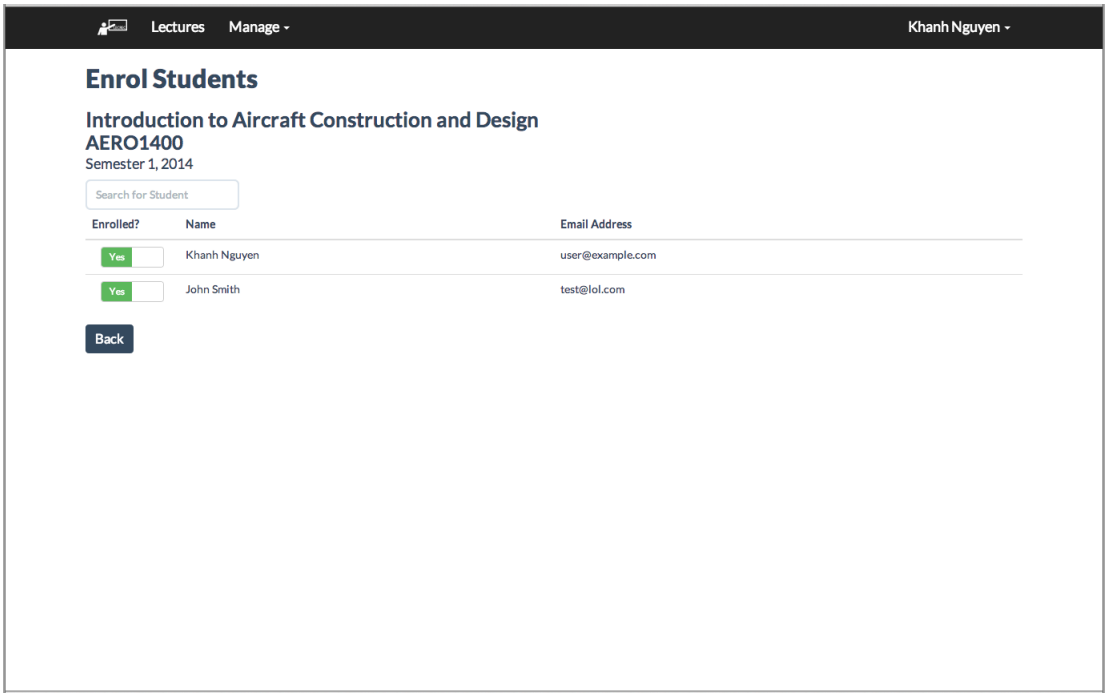

Figure 8: Page for lecturer to enrol students in a particular course

3 Program Navigation Diagrams

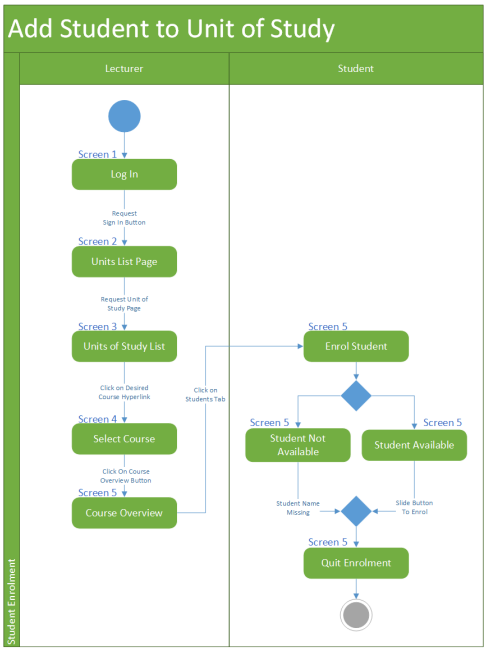

Figure 9: Enrolment of student to unit of study

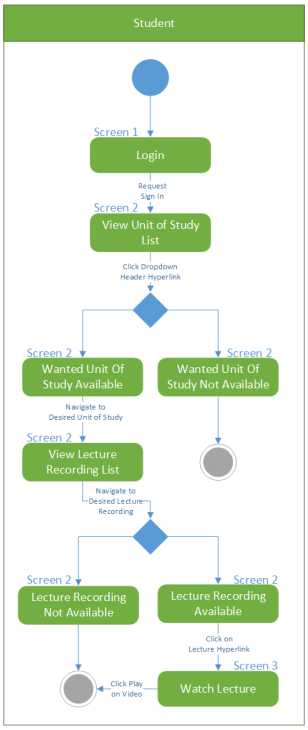

Figure 10: Student navigation to watch lecture

4 Data Definitions

Where	Variable	Type	Constraints	Example	Definition
User	id	Integer	Primary Key	1448	id of user
	name	String		John Mayer	name of the user
	email	String	not NULL	john@mayer.com	email to send messages to
	encrypted_password	String		10hn@m4y37	password for login
	sign_in_counter	Integer		22	counter variable to track how many times a user has logged into the system
	reset_pass_sent_at	Datetime		2014-09-12 06:54:58.466537	date of last password reset requested by user
	remember_created_at	Datetime		2014-09-12 06:54:58.466537	
	current_sign_in_at	Datetime		2014-12-12 06:54:58.466537	current user sign in date and time
	last_sign_in_at	Datetime		2014-09-12 06:54:58.466537	last user sign in date and time
	current_sign_in_ip	Inet		60.242.159.117	IP address of the current logged in user
	last_sign_in_ip	Inet		60.242.123.11	last IP sign in address of the current user account
	created_at	Datetime		2014-09-12 06:54:58.466537	user created date and time
	updated_at	Datetime		2014-12-12 09:45:23.483745	user information update date and time
Role	id	Integer	Primary Key	35	id of role used for reference in the LECREC system
	name	String	not NULL	Student	role title that will be associated with a particular user account
	created_at	Datetime		2014-12-12 09:45:23.483745	created date and time of new role
	updated_at	Datetime		2014-12-12 09:45:23.483745	updated date and time of current role
Semester	id	Integer	Primary Key	3668	id as a number reference for use in the system
	session	Integer	not NULL	2	session is used to reference which semester a lecture recording is a part of
	year	Integer	not NULL	2014	year of the semester
	unit_of_study_id	Integer	Foreign Key	1111	id value related to the "Unit of Study" below
	created_at	Datetime		2014-01-01 09:55:34.543875	created time and date of the semester
	updated_at	Datetime		2014-12-12 09:45:23.483745	last update date and time of semester

Unit of Study	id	Integer	Primary Key	1111	id of the unit of study for used in references in the system
	title	String	not NULL	Web Applications	string value of the course name
	alpha_code	String	not NULL, UNIQUE	ELEC3609	alpha numeric code of the course name based on typical university standard
	created_at	Datetime		2014-01-01 09:45:23.483745	the date this unit of study was created
	updated_at	Datetime		2014-06-01 09:45:23.483745	last update, date and time of Unit of study
Lecture Recording	id	Integer	Primary Key	1223	id of recording for use in reference in system
	name	String	not NULL	Wk1_dotnetframe	name of the lecture recording to display to users
	created_at	Datetime		2014-03-03 12:23:24.133745	created date and time of the lecture recording
	user_id	Integer	Foreign Key	112	id of the user who uploaded the recording
	raw_video	String		myVideo.avi	string of location of the original video
	processed_path	String		/p/myVideo.mp4	string of the path that the processed video is stored into
	semester_id	Integer	Foreign Key	3668	semester id value that the recording relates to
	updated_at	Datetime		2014-12-12 09:45:23.483745	last update date and time of the Lecture Recording
Discussion	id	Integer	Primary Key	162	id value of the newly created discussion
	title	String	not NULL	Help with .net	string title of the discussion
	content	Text	not NULL	I don't understand	content text used within the discussion
	lecture_recording_id	Integer	Foreign Key	1223	id referring to which lecture recording that the discussion is a part of
	user_id	Integer	Foreign Key	1448	id of user who created the discussion
	created_at	Datetime		2014-03-04 13:23:21.484235	date and time of a new discussion
	updated_at	Datetime		2014-12-12 09:45:23.483745	date and time of an updated discussion.

Comment	id	Integer	Primary Key	3	id for reference of a newly made comment
	content	Text	not NULL	Me neither	text of comment from user
	user_id	Integer	Foreign Key	1449	id of user who created the comment
	discussion_id	Integer	Foreign Key	162	id of the discussion the user comments in
	created_at	Datetime		2014-03-06 13:23:21.484235	date and time of a new comment for display
	updated_at	Datetime		2014-03-12 13:23:21.484235	last updated date and time of a user's comment
Video	id	Integer	Primary Key	69	id value of the video
	title	String	not NULL	Mathstley	title string of video
	content_url	String	not NULL	`https://www.youtube.com/watch?v=dQw4w9WgXcQ`	URL as a string to the video to embed
	category_id	Integer	Foreign Key	234	id value of the category the video pertains too
	user_id	Integer	Foreign Key	1	id value of user who posted the video
	created_at	Datetime		2014-12-12 09:45:23.483745	created date and time of the video class
	updated_at	Datetime		2014-12-14 09:45:23.483745	updated date and time of the video class
Community Category	id	Integer	Primary Key	234	id value of each community category
	title	String	not NULL	Maths Help	string title of a particular category discussion
	created_at	Datetime		2014-12-12 09:45:23.483745	created time and date of a category discussion
	updated_at	Datetime		2014-12-14 09:45:23.483745	last updated time and date of a category discussion

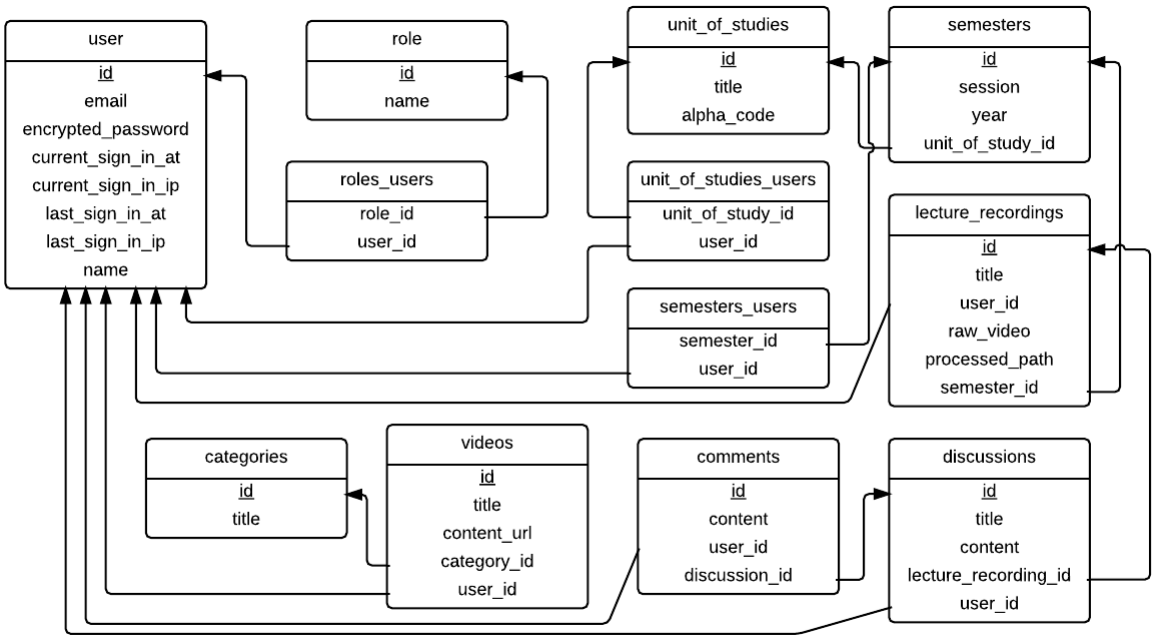

Figure 11: Entity Relational Diagram

5 Design Class Diagram

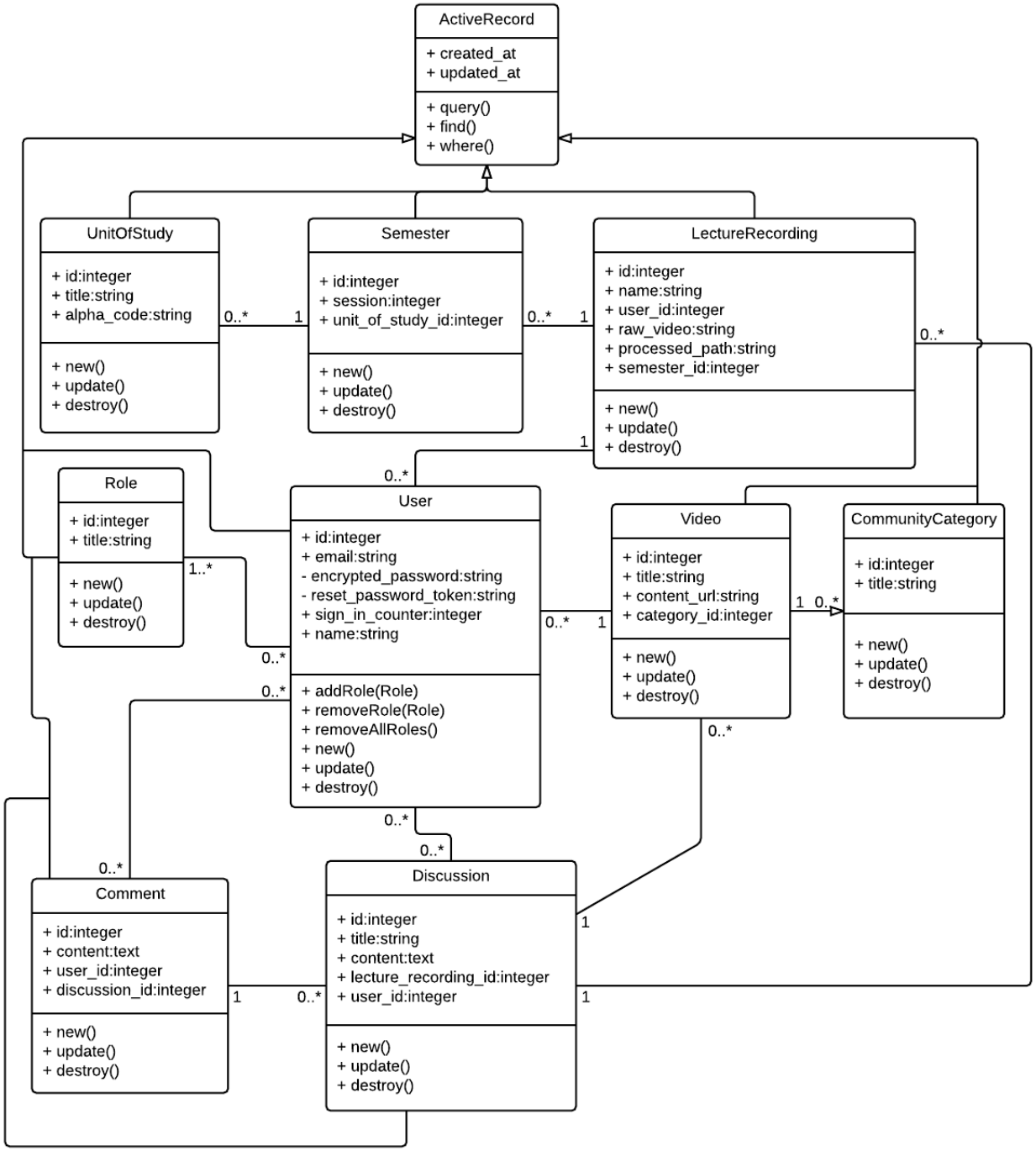

Figure 12: Class diagram of the LECREC System

6 State Diagrams

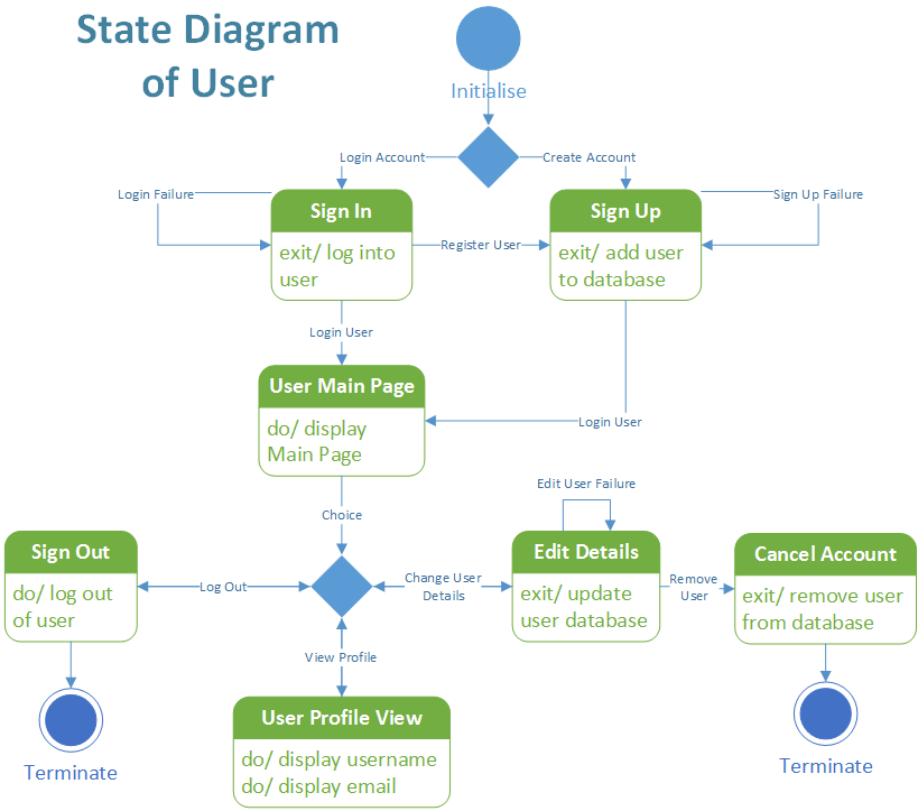

Figure 13: State diagram of user Login

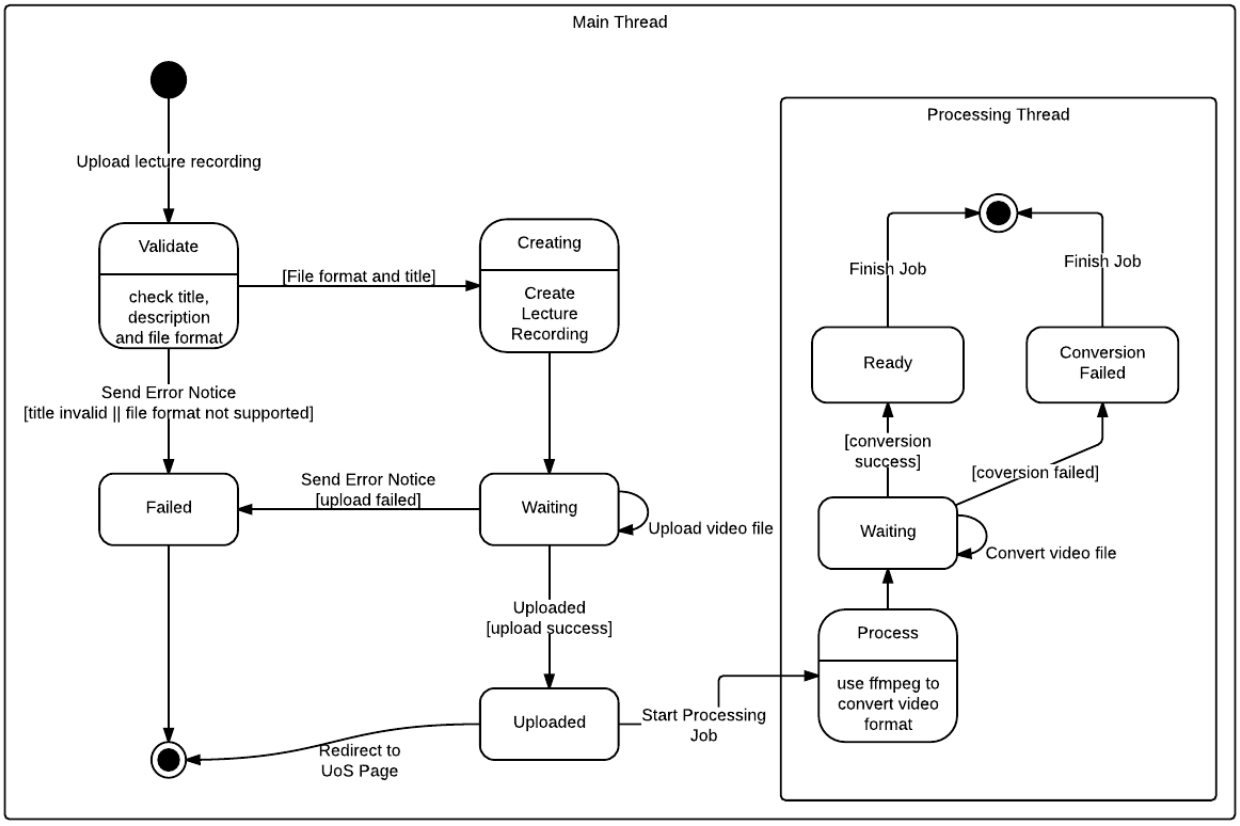

Figure 14: State diagram of the upload process

7 MVC Diagrams

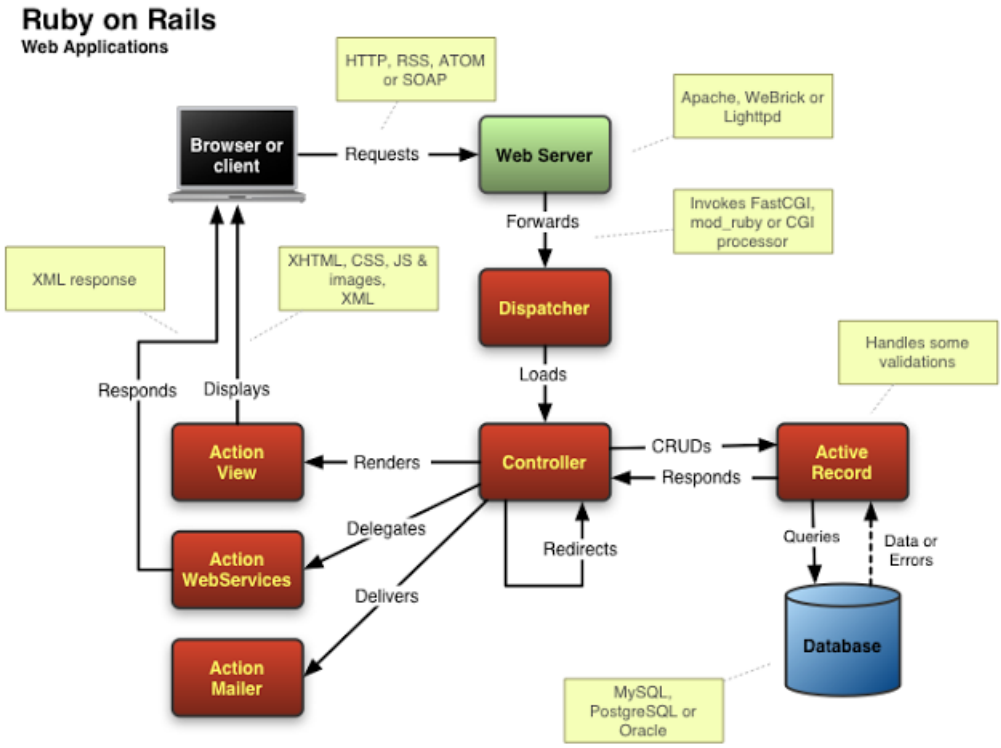

Figure 15: MVC diagram of a Ruby on Rails application

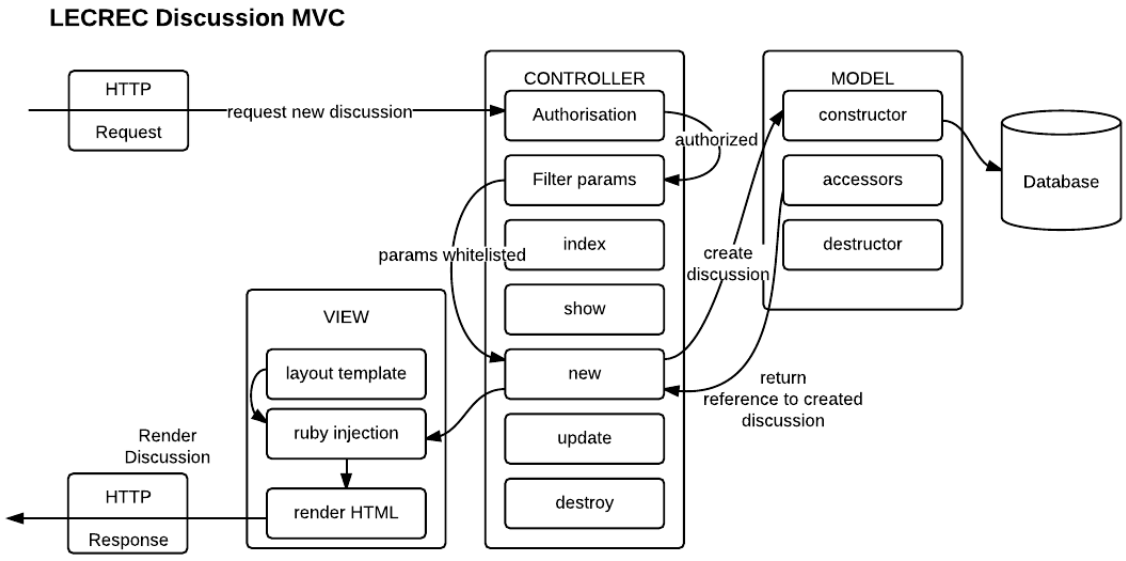

Figure 16: MVC Diagram for creating a new discussion in LECREC

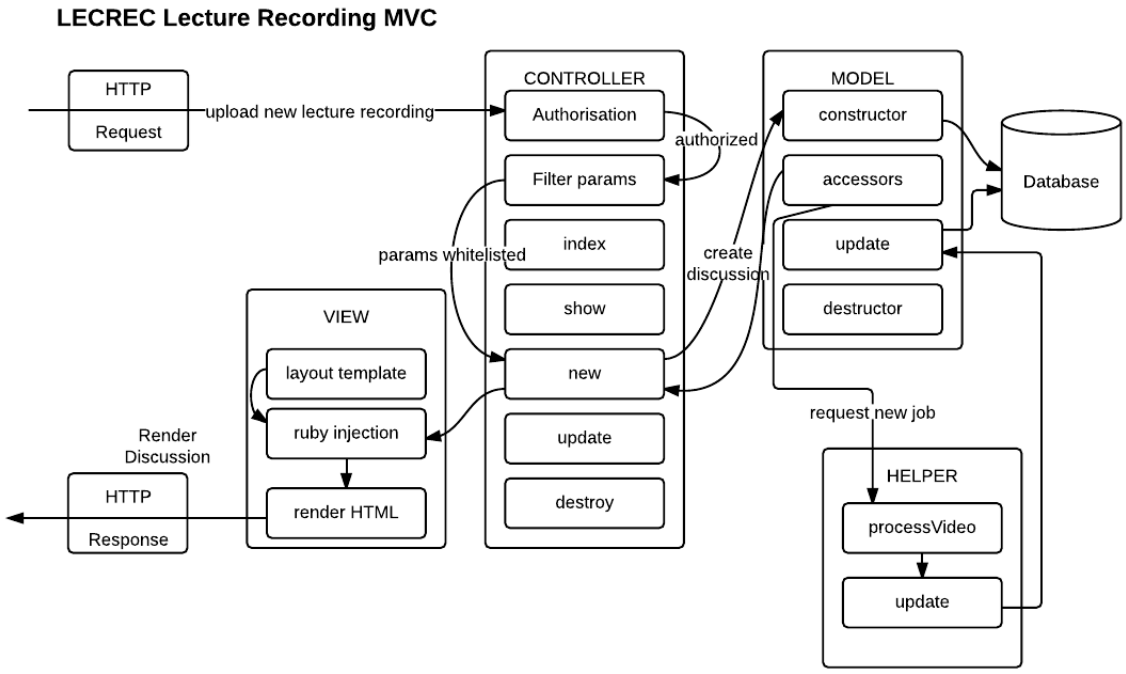

Figure 17: MVC diagram for uploading new lecture recording

8 List of Assumptions

8.1 Application

- LECREC will be built on Ruby on Rails 4.1
- This application will use a relational database, PostgreSQL 9.3
- A Redis server will setup to store key-value pairs
  - http://redis.io/
- RESQUE will be used to manage the video processing job queue
  - https://github.com/resque/resque
- ffmpeg will be used to convert videos
  - https://www.ffmpeg.org/
  - Videos will converted into HTML5 friendly formats such as mp4, ogg or webm.

8.2 User roles and target audience

- LECREC is aimed for university students and lecturers
- Students will be able to watch lecture recordings, community videos and create discussion on each video.
- Students and Lecturers can comment on discussions
- Lecturers will be able to manage their units of study
  - create a new session (semester 1 or 2 of a certain year)
  - enrol student to that session
  - upload lecture recordings

8.3 Browser support

LECREC will be developed to support the following browsers:

- Internet Explorer 11
- Google Chrome
- Mozilla Firefox 30+

8.4 HTML5 Limitations

HTML5 is the latest standard for HTML and it is still not fully support by some browsers. It is assumed that some functionalities will not working as expected on Internet Explorer, especially Server Sent Events and some video formats.

8.5 Hardware and hosting

- LECREC will be hosted on a Centos 6.5 virtual machine
- The virtual machine will have at least 4GB of RAM and 100GB of disk space
- Server will be hosted within Australia for potential copyright and privacy reasons involving lecture recording material

9 Team contributions

Alexander Woo Hyun Jung: SDS discussion. Documentation planning, write up and diagram construction.
Kelvin D'Amore: SDS discussion. Documentation planning, write up and diagram construction.
Khanh Cao Quoc Nguyen: SDS discussion. Documentation planning, write up and diagram construction.
Martin Groen: SDS discussion. Documentation planning, write up and diagram construction.