Integration and assessment of streaming video content and API development into a spaced repetition service

YOUBEI JIN

KTH Information and Communication Technology

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

Integration and assessment of streaming video content and API development into a spaced repetition service

Youbei Jin

2014-03-05

Master's thesis

Examinor and academic adviser
Professor Gerald Q. Maguire Jr.

School of Information and Communication Technology (ICT)

KTH Royal Institute of Technology

Stockholm, Sweden

Abstract

There are a lot of spaced repetition system based online learning services available nowadays, but none of them are popular or widely accepted as a good means for studying. The problem of these services is, instead of exploiting and utilizing the possibility of modern technology, they simply brought this old school learning method to the web as an application with exactly the same experience as one could have with textbooks and flash cards. This master's thesis project concerns Sharplet, a spaced repetition system based web service, who is trying to stand out by offering some features that none of the other comparable services have. One of these features is the integration of YouTube video clips, so that in addition to text and pictures, study materials may now include both audio and video material.

This thesis begins by reviews spaced repetition systems in general, and then focuses on some of the existing services and how to provide a better service. Next the thesis describes the design, implementation, and evaluation of a new service that includes both audio and video content. The main practical result of this master's thesis project is a working prototype audio and video enabled spaced repetition based service. Unfortunately, the actual performance of this prototype is unacceptable, hence there is a need to improve its performance before it can be integrated with the production spaced repetition system.

Keywords: *spaced repetition, YouTube API, user experience, responsive design, bootstrap*

Sammanfattning

Det finns en hel del repetitionssystem baserat online- lärande tjänster som finns idag, men ingen av dem är populära och allmänt accepterat som ett bra sätt för att studera. Problemet med dessa tjänster är, i stället för att utnyttja och utnyttja möjligheten till modern teknik, de helt enkelt fört din gamla inlärningsmetod för webben som ett program med exakt samma upplevelse som man kunde ha med läroböcker och flash-kort. Denna magisteruppsats projektet gäller Sharplet, ett repetitionssystem baserat webbtjänst, som försöker sticka ut genom att erbjuda vissa funktioner som ingen av de andra jämförbara tjänster har. En av dessa funktioner är att integrera YouTube- videoklipp, så att förutom text och bilder, samt ljud och video både är nu tillgänglig för läromedel.

Detta examensarbete recensioner repetitionssystemi allmänhet, fokuserar sedan på några befintliga tjänster och hur man kan ge bättre service. Den största utmaningen kommer att undersöka YouTube API och studera användarbeteendeför att ge en bättre användarupplevelse. Det viktigaste resultatet av detta examensarbete är en fungerande prototyp av en ljud -och video aktiverat repetitionsbaserad tjänst. Tyvärr är det faktiska utförandet av denna prototyp oacceptabelt, därför finns det ett behov av att förbättra sina resultat innan den kan integreras med tillverkningen fördelade upprepning systemet.

Nyckelord: *spaced repetition, YouTube API, user experience, responsive design, bootstrap*

Table of Contents

List of figures

List of tables

List of Acronyms and Abbreviations

apps	applications
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IE	Microsoft Internet Explorer
px	pixels
RWD	Responsive web design
UI	user interface
UX	better user experience

1 Introduction

This chapter gives an introduction to the area of concern in this thesis, the problem statement, goals, and an overview of this master's thesis.

1.1 General introduction to the area

Sharplet[1] is an online learning content management system available through most popular internet-enabled devices. Sharplet features a unique scheduling algorithm based upon the work of Cecil Alec Mace and Piotr Woźniak[2] that enables users to maximize their learning and study time. Information spacing of this kind has been demonstrated through numerous peer-reviewed studies to dramatically increase memory retention and reduce study time[3–5].

There are several E-learning platforms on the market that are based on the spaced repetition method, but one proposed feature that could make Sharplet unique from its competition is the ability to allow users to edit educational videos hosted on YouTube. Users will be able to chop a video into time stamped segments that need to be studied and then synchronize them from a cloud server to web and mobile applications (apps), thus users can watch these video segments anywhere whilst transitioning the material from short to long term memory based on the scheduling algorithm.

1.2 Problem statement

The main problem to be addressed during the course of this thesis project is to create a working prototype of a video study and editing mode, optimized for web and mobile web use. In order to achieve that, YouTube API and Bootstrap[6] will be used.

Most example apps on the Internet are using YouTube API Version 2. Instead of Version 2, Version 3 will be explored. Although Version 3 is an experimental version, it provides many enhanced functions, some of which are key functions to realizing the goals of this thesis project. YouTube API v3.0 makes it possible to add YouTube functionality to an application. The API can be used to fetch search results and to retrieve, insert, update, and delete resources such as videos or playlists. In combination with the YouTube Player APIs and the YouTube Analytics API, YouTube API v3.0 provides an application with a full-fledged YouTube experience that includes search and discovery, content creation, video playback, account management, and viewer statistics.

Unintuitive user experience (UX) [7] often causes users confusion. The result will be either the users will not use the application as often or in the worst case will never return to using the application based on their poor initial experience with it. In order to provide a better user experience Twitter's Bootstrap will be used in this project. The final working prototype will have:

- A responsive design, which adapts to most screen sizes on the market (such as laptops, tablets, smart phones, etc.). Thus the layout will be automatically adjusted to suit the size and height to width ratio of the screen.
- A design that maintain consistency with other parts of the Sharplet system.
- An interface that users can understand with minimal new concepts to learn.

1.3 Goals

During the course of this master's thesis project, the following objectives will be met:

- Study and understand spaced repetition system so the following development makes sense both for myself and for future users.
- Investigate the use of the YouTube API, quantifying the merits of different API versions. Section 2.21 explains how OAuth which is an authorization standard works to authorize and enable users to access Google and YouTube resources. In Section 2.22, two versions of the YouTube API are compared and an explanation is given of why version 3 was ultimately used in throughout this master's project.
- Section 2.2.3 discusses how best to ensure compatibility between version 3 and future versions of the Youtube API in a seamless manner.
- A basic functional design will be done of the system (see Chapter 4). A block diagram will be developed to explain how the feature works in general and how it interacts with the Sharplet Database and spaced repetition algorithm.
- Develop the video editing and video study features. A video editing feature will be that allows users to easily curate lecture video resources from YouTube (see section 4.2.1). Following this a video study mode will be developed that allows users to watch entire video lectures and select pre-curated video segments for long term memorization using the spaced repetition algorithm (see section 4.2.2).
- Study users' behavior in order to provide a better user experience (UX) and better interaction via the user interface (UI). Responsive web design will be discussed and studied as this is necessary due to the various devices users will be using to access the service. Details of this user study are descried in section 4.4. Some potential browser compatibility issues will be discussed as well (see section2.3.2.4).
- A performance analysis will be made of system via load testing before deploying it in order to find ways to maximize its performance. This performance analysis will be presented in section 4.6.
- Once the new feature has been refined both in terms of performance and design, it will be integrated it into the live Sharplet web application. The main problems in this step will be the database connection and design consistency. This integration will be described in section 0
- An analysis will be made based upon quantitative and qualitative data collected during actual site usage in order to better understand the system dynamics for a number of scenarios. Data will be collected by using Google analytics, load testing, etc. This analysis will be presented in section 4.8.
- As this feature will eventually be made available via a mobile application, it is important to understand what technical and functional requirements should be addressed before introducing this new feature into the mobile application. These requirements will be described in section 4.9.

1.4 Research Methodology

Design research methodology was selected as the research methodology for this thesis as one of the goals was to have a working prototype of a video study and editing mode, optimized for web and mobile web use. By having this prototype it would be possible to collect data about how the prototype functions as well as how both user and developers like its functionality. Alternatives research methodologies that were considered included quantitative research methodology and qualitative research methodology, but these did not suite the goals of this project because for quantitative research method, a questionnaire is not capable for getting the feedback necessary to improve the user experience and for qualitative research methodology, the interview for individuals will bring too much of the individual's personal opinions and often ignore the technical difficulty to achieve them. Readers who are interested in further details of design research methodology should read *Getting Real*[*].

1.5 Structure of this thesis

The first chapter of this thesis introduced the area, the problem to be addressed, and the goals of the project. Chapter 2 provides the background needed to read the rest of the thesis. Chapter 3 describes the method to be used to solve the problem stated in section 1.2 and along with the reasons why this method has been chosen. Chapter 4 describes the implementation and analysis of a prototype developed in this thesis project. Chapter 5 concludes the thesis with conclusions, suggestions for future work, and reflections on social, economic, ethical, and other aspects of this thesis project.

[*] Getting Real by 37signals is a great book introduces a smarter, faster, easier way to build a successful web application.[8]

2 Background

This chapter explains background of the thesis, including Spaced Repetition System, YouTube API and a few aspect of web design.

2.1 Spaced Repetition System

This section gives an introduction to how spaced repetition system works and gives a comparison between Sharplet and several other similar services.

2.1.1 Spaced Repetition System

The spacing effect was reported by a German psychologist Hermann Ebbinghaus in his book "Memory: A Contribution to Experimental Psychology" in 1885[9]. He discovered that we remember things more effectively if we study and review things repeatedly over a long period of time instead of studying them many times within a short period of time. There have been papers about studying and utilizing the spacing effect to improve learning such as Herbert F. Spitzer's "Studies in Retention" in 1939[10] and C. A. Mace's "Psychology of Study" in 1932[11, 12], in these paper the spaced repetition method was proposed.

In 1972, a German scientist called Sebastian Leitner started using paper flashcards with the method of spaced repetition and developed Leitner system[13]. The Leitner system works like this:

- Use a box contained several compartments to store flashcards with questions and study materials.
- At beginning, all flashcards are stored in the first compartment.
- Correctly answered or remembered flashcards move to the next compartment. From the first one to the second one, the second to the third one etc.
- The higher the compartment number the longer the repetition interval (in days) that the learner should study them again.
- Incorrectly answered or remembered flashcard move to the first compartment where the cycle starts.
- So the less the learner remembers the content of the flashcards the more frequently he (or she) needs to repeat them.

This is a huge improvement over the previous method and it established the general idea of later computer flashcard applications. However, compared to the applications today, this mechanical system cannot tell the learner the exact date on which something should be reviewed again, thus reducing the effect of the method.

The first commercial computer flashcard application that implemented spaced repetition was SuperMemo[14]. SuperMemo solved the main problem with Leitner system as it could accurately track when a specific item should be reviewed based on the performance of the learner. Every time the learner sees a question, he (or she) tells the application how well he (or she) is able to remember or answer the question, with an answer such as: "Unknown", "Familiar", "Known", etc. This feedback will be used by the application to optimize the time before this flashcard is shown again. The time between reviews increases, from 3 days to 15 days to longer, every time learner gives "Known" as their feedback as the memory of the material gets stronger each time it is recalled. This approach is driving a revolution in studying and formed the starting point for other flashcard applications in the market today.

2.1.2 SuperMemo Algorithm

Piotr A Woźniak's SuperMemo algorithm[15] is the algorithm being used in Sharplet.com. This section explains how this algorithm works.

2.1.2.1 SM-0 algorithm

We begin with a formula to calculate inter-repetition intervals:

```
I(1):=1
I(2):=6
for n>2 I(n):=I(n-1)*EF
```

Where I(n) is inter-repetition interval after the nth repetition (in units of days) and EF (E-Factor) is a factor reflecting the easiness of memorizing a given item. E-Factors range from 1.1 to 2.5 where 1.1 is the most difficult and 2.5 is the easiest. E-Factor equals 2.5 by default when a new item is introduced into a SuperMemo database. The more recall problems the learner has more the value of the E-Factor will be *decreased* during the studying process (as this will cause the item to be presented *more* frequently).

The formula to calculate the E-Factor is:

```
EF'=EF-0.8+0.28*q-0.02*q*q
```

Where EF' is the new value of the E-Factor, EF is the old value of the E-Factor, and q is the quality of the response (i.e., the feedback from the learner).

This is not the final algorithm used in SuperMemo programs, as some minor changes were made to produce the SM-2 algorithm.

2.1.2.2 SM-2 algorithm

Before computing when to present study items, SM-2 begins with a preparation of the material to produce a set of pieces (smaller study items). The algorithm is modified to consider threshold effects (by avoiding items getting too low a E-Factor value and repeating items more frequently until the feedback is consistently very good). The resulting algorithm is[16]:

1. Prepare study materials and split them into pieces.
2. E-Factor equals 2.5 by default.
3. Using the formula in SM-0 to calculate inter-repetition intervals.

```
I(1):=1
I(2):=6
for n>2 I(n):=I(n-1)*EF
```

4. The quality of response is in 0-5 grade scale (with 0 being unknown and 5 being well known).
5. Using the formula to calculate E-Factor:

```
EF':=EF + (0.1-(5-q)*(0.08+(5-q)*0.02))
```

6. The value of EF will never be lower than 1.3, if it does make it be 1.3.
7. If the quality of response is lower than 3 then start repetitions for the item all over again *without* changing the E-Factor.

8. Repeat all items whose quality of response is below 4, until all of the time the score is at least four.

2.1.3 Existing services

Today Anki[17] is the most popular application that uses spaced repetition system to help learners remember things easily. Anki is a Japanese word means “learning by heart”. It is free and open sourced and is similar to SuperMemo (a commercial application for the same purpose). Anki is a multi-platform application. It is supported on Microsoft’s Windows, Apple’s Mac OSX, and Linux/FreeBSD. Anki’s spaced repetition system is based on the SuperMemo SM2 algorithm. Since web apps have become very popular, the lack of a well-designed web app service and a relatively more complicated installation and configuration of the application typically immediately scares a lot of potential users away. Anki is open source, this means that it is maintained by a community of people using their free time. Unfortunately, the result of this is an ugly and out of date looking interface that requires a not so friendly configuration before it can be used.

Cram[18] is a service originally founded in January 2001. Cram includes a web app, mobile apps for different mobile OSs, and is based on the Leitner System. Cram differentiates itself from other websites because it makes use of its own “Cram Mode” version of the Leitner System[19]. This mode is basically the same as the SM-2 algorithm described above, but with a community feature. This community feature allows user to create learning materials from any of the hundreds of thousands of flashcards previously created by other users in Cram system. Some features such as making flashcards into multiple-choice questions and including images in flashcards are also very cool.

Quizlet[20] was founded in 2005 and is a similar service using the same (SM-2) algorithm. Quizlet offers users the ability to study flashcards in a normal way as well as playing a couple of games with their flashcards. When creating and editing flashcards users can enter information for each card individually or import information from a text file. Users can also browse and study other publicly shared learning materials.

Table 2-1 compares these various flashcard based services.

Table 2-1: Existing services comparison

	Advantages	Disadvantages	Unique Features
Anki	• Free and open source • Multi-platform	• There is no web app. • Interface is outdated. • Configurations are too complicated and not user friendly compared to other services.	Since Aniki is open source, user can modify or add whatever features he wants, make it his personal study tool and share these new features with the community.
Cram	• A modern and clean web app. • Android/iOS app available. • A huge flashcard database for creating new courses.	Annoying advertisements.	• Multiple choices in flashcards. • Image in flashcard.
Quizlet	• A modern and clean web app. • Android/iOS app available.	Annoying advertisements.	Games in flashcards to help memorize.
Sharplet	• A modern and clean web app.	• Android/iOS App not available yet. • Newly founded company with limited resources.	YouTube integration makes video flashcards possible.

2.2 Youtube API

This section explains what is needed to make a web application that can use YouTube resources and how to use these resources. An overview of the Google YouTube API is given in [21]. This section begins with some background about OAuth 2.0, as this authorization mechanism is essential to using YouTube resources. This is followed by a description of the YouTube API, the Data and Player APIs, and finally DOM storage.

2.2.1 OAuth 2.0

OAuth [22–25] is an authorization standard used by a lot of services providers, such as Google, Facebook, and others. OAuth enables users to access specific resources and services as the resource owner from other applications, meanwhile users do not have to give out their username-password pairs to third-party applications to authorize this access. OAuth 2.0[26] is the next generation of OAuth, but is not backward compatible with OAuth 1.0.

When a user first attempts to use the video learning feature in Sharplet, he or she will be required to login using a Google or YouTube account. To perform this login the application initiates the OAuth2 authorization process. First the user will be directed to Google's authorization server via a link. This link specifies the exact access scope that the application is requesting for the user's account. The scope specifies the resources that the application can retrieve, insert, update, or delete when acting as the authenticated user.

If the user agrees to this authorization of the application to access the specified resources, the server will return a token to the application. The application will either validate this token or exchange this token for a different type of token. An access token and a refresh token will be exchanged by a server-side web application. The access token is a token that allows the application to authorize requests on the user's behalf. When a token expires, a refresh token allows the application to get a new access token.

2.2.2 API Version 2 and API Version 3

Some of the key differences between API Version 2 and API Version 3 are listed in Table 2-2.

Table 2-2: Comparison between YouTube API versions 2.4 and 3.0

	API version 2.4	**API version 3.0**
Compatibility	Backwards compatible with previous API versions.	Not backwards compatible with previous API versions.
Responses output	XML	JSON
Google Data JavaScript client library	Not supported	Supported
Authorization methods supported	OAuth 1.0/2.0	OAuth 1.0/2.0
Registration and developers tokens required	Yes	Yes
Future update and development from Google	No	Yes

Version 3 was chosen for use in this master's thesis project for the following reasons:

1. Version 3 will have a longer life than version 2, so re-development because a change in the API version will not be needed in the near future. Version 2 is no longer being developed, this means that Google will abandon it soon and replace it with Version 3.
2. The documentation for version 3 is cleaner and easier to navigation than for version 2.
3. Version 3 uses JSON for responses; this reduces the size of the response by nearly a factor of 10 from the previous XML output used in Version 2.
4. Usage of version 3 is encouraged in the YouTube API documentation.

2.2.3 Data API and Player APIs with a chromeless player

A chromeless player is a YouTube player that has no controlling interface displayed above the video content, unlike the standard YouTube videos that has play/pause controls at the bottom. Such a player is preferable for this project because many editing features will be added and their controls will be customized and different from the default YouTube controls.

The Javascript Player API provides some basic control functions such as queuing, play, pause, stop, adjusting volume, and jumping to a specific time. By combining all of these functions a sophisticated learning mode can be built.

The Data API provides many of the operations available for the YouTube website to an application. It allows an application to search for videos by key words and retrieve related information, such as thumbnail pictures, video durations, and a description of the videos.

2.2.4 DOM Storage

DOM storage[27] is a set of storage-related features introduced in the Web Applications 1.0[28] specification, and now is a part of W3C Web Storage specification[29]. DOM storage can provide an easier and more secure way to store data than using cookies[30].

The DOM Storage mechanism can store JavaScript string key/value pairs in a web application securely and retrieve them later for use. DOM Storage is very useful since it is the only browser-only method to store data with a reasonable capacity and it can store data for any period of time. Browser cookies on the other hand, have limited capacity and do not have the ability to organize data; while other methods such as Flash Local Storage require external plugins, making them not so easy to use and causing unnecessary compatibility issues.

There are 3 types of storage methods in DOM Storage:

sessionStorage	**sessionStorage** provides a global object that maintains a storage area that is available for the duration of the page session. This means that the data is stored as long as the browser is open and survives over page reloads and restores. Opening a new page in a new tab or window in the browser will initiate a new session.
localStorage	a global object similar to sessionStorage, but the data stored can be used in a new page opened in a new tab or window as long as the browser is open
globalStorage	a non-standard and obsolete method that has been removed from the HTML5 specification in favor of localStorage

DOM storage works in most of modern browsers, including Mozilla-based browsers, Internet Explorer 8+, Safari 4+, and Chrome. This browser compatibility is summarized in Table 2-3.

Table 2-3: Browser compatibility for DOM Storage

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
localStorage	4	3.5	8	10.50	4
sessionStorage	5	2	8	10.50	4
globalStorage	Not supported	2-13	Not supported	Not supported	Not supported

2.3 UX

This section looks at how we can provide a high quality user experience (UX). It begins by explaining how to work with Twitter's Bootstrap in order to create a responsive web design. This is followed by a discussion of responsive web design.

2.3.1 Bootstrap

Bootstrap is a front-end toolkit that provides an easy way for a developer with limited design training to rapidly make an adequate design for a web application interface. Bootstrap is basically a collection of Cascading Style Sheets (CSS) and HTML conventions[31] with some of the latest browser techniques.

Bootstrap is just CSS built with Less at its core. Less is a dynamic stylesheet language, it offers much more power and flexibility than normal CSS and extends CSS with dynamic behavior, such as variables, mixins, operations, and functions[32]. Due to the use of pure CSS compiled via Less, Bootstrap is very easy to implement.

2.3.2 Responsive Web Design

Responsive web design (RWD)[33, 34] is a web design technique aimed at providing the best viewing and user experience on most of the devices that are available nowadays. Reichenstein describes RWD as: “this is the idea that your layout automatically adapts to the screen definition”[35]. The goal of RWD is not simply prettiness or to show technical trickery, but rather the goal is to facilitate user interaction the user changes their platform from desktops and laptops to smartphones and tablets. This is especially important today when IDC has said: “Shipments of PCs fell 14% worldwide last quarter, according to IDC. It was the worst yearly decline since IDC began tracking the data in 1994.”[36]. Thus today rather than buy a PC, consumers are buying tablets and smartphones. This means that application developer have to be able to have applications that run on both traditional platforms and smart phones, as the later are in increasingly large part of the market.

It is impossible to buy every device with every screen size on the market and write a specific CSS file for it, thus web designers have adopted some techniques to make their life easier. There are two approaches to responsive web design[37]: adaptive layouts and liquid layouts.

Adaptive layouts change the layout in several stages; by defining several different CSS files each optimized for a range of widths. For example, there are basically 4 kinds of screens to consider: desktop monitors, laptops, tablets, and smart phones. We do not know the exact pixel dimensions for each, but we can define a set of width ranges in pixels (px) to define a desktop, laptop, tablet, and smart phone – as follows:

```
1600px < desktop

1024px < laptop <= 1600px,

768px <= tablet <= 1024px, and

Minimum acceptable pixel dimensions <= smart phone < 768px
```

We can design different layouts for each of these above ranges, in order to take best advantage of the available pixels on the screen.

Liquid layouts adjust the layout continuously to every possible width. In this case there is only one layout, so it might look very strange on huge monitor or tiny smart phone screen (since their height to width ratio is different), with the first one frequently being a wide screen, while the other is frequently a narrow screen. Consider the website: http://d.alistapart.com/responsive-web-design/ex/ex-site-flexible.html, while this web page magically fits the window if you change the browser’s window width; however, I do not consider it a good design. If the screen is too wide, each line will be too long for the eyes to easily move from one line to the next, giving users a terrible reading experience. On the other hand, if the screen is too narrow, the layout is no longer suitable since the elements are squeezed together. In this case, actually changing the layout is a wiser choice, for example moving the left list to the top of the page.

While both approaches have advantages and disadvantages, I believe that the first method (i.e., the adaptive layout) with as few as break points possible is the way to go, because readability is more important than having a layout that is always as wide as the viewport.

2.3.2.1 Media Queries

To make a webpage responsive with some possible break points, Media Queries in CSS3 are used. A **media query** [38] consists of a media type and at least one expression that limits the style sheets' scope by using media features, such as width, height, or color. Media queries, added in CSS3, allow the presentation of content to be tailored to a specific range of output devices without having to change the content itself. An example of such a media query is:

```
  <link rel="stylesheet" type="text/css" media="only screen and (max-
width: 1023px) and (min-width: 800px)" href="/assets/small.css" />
```

The above example of a media query means that when the browser’s width is greater than 800 pixels and smaller than 1023 pixels, the browser should use the “small.css” which defines the layout, text size, line height, padding, margin, etc. for this width range.

2.3.2.2 Responsive videos

Sometimes it is a problem that a video iframe[*] cannot be easily adjusted, unlike the case for an image; hence to make a responsive video iframe some tricks are needed. For example, a key to make this trick work is to use 56.25% (9/16=0.5625) for padding, which makes the height of the iframe equal to 56.25% of its width. In other words, this trick causes the iframe ratio to always be 16:9 ,no matter how the browser’s width changes. Padding at the bottom is used instead of padding at the top because of some Microsoft Internet Explorer (IE) issues. The CSS below does this:

```
.video {
position: relative;
padding-bottom: 56.25%;
height: 0;
margin: 0.625em 0 0 0 0;
}

.video iframe,
.video object,
.video embed
{
position: absolute;
top: 0;
left: 0;
width: 100%;
height: 100%;
}
```

With the following code, the iframe can be put beyond the boundary of its parent, beyond the 56.25% padding area.

[*] The <iframe> tag specifies an inline frame that is used to embed another document within the current HTML document, the document can be a flash video (as in this case)[39].

```
  padding-bottom: 56.25%;
  position: absolute;
```

The code below causes the iframe stretch to fill the width of its container:

```
  width: 100%
```

The code below causes the iframe stretch to fill the height of its container:

```
  height: 100%
```

2.3.2.3 Responsive Typography

There always will be a choice between serif and sans serif fonts. Reichenstein has said that "A serifed typeface has a more authoritarian touch, whereas a sans serif feels more democratic"[35]. However, I believe that neither one is necessarily better than the other.

The size of the body text does not depend on a designer's personal preference. It depends on a user's reading distance. Since in general computers are further away than books, the metric size of a desktop typeface needs to be bigger than the sizes used for printed matter.

2.3.2.4 Browser compatibility

Due to using Bootstrap as the front-end toolkit browser compatibility browser compatibility is not a big issue in this project. One of Bootstrap's advantages is that it handles different browsers and handles them very well. Bootstrap is built to work best in the latest desktop and mobile browsers[6], it supports the latest versions of the following browsers:

- Chrome (Mac, Windows, iOS, and Android)
- Safari (Mac and iOS only, as the Windows version is being abandoned),
- Firefox (Mac, Windows),
- Microsoft's Internet Explorer, and
- Opera (Mac, Windows).

Unfortunately, for Microsoft's Internet Explorer 7 and Internet Explorer 8, some CSS3 properties and HTML5 elements are not fully supported. However, it is not particularly important to support these two largely obsolete browsers. According to W3Counter[40], in October 2013 the total market share for IE7 and IE8 is 12.24%, hence for our purposes they are not worth spending much time on.

3 Method

This thesis project has been divided into several stages in order to make it manageable. These states are:

Stage 1 Understand the basic requirements and the overall development environment of the existing Sharplet website.

Stage 2 Employ the exploratory programming paradigm. By coding small functions, such as searching YouTube videos, Chromeless YouTube player, a dragging bar to control the play time for the player, and so forth. Finally put these functions together.

Stage 3 Testing the prototype and to find out what feels wrong. After bugs are found and new requirements emerge, return to stage 2 after fixing the bugs and adding new functions to improve the prototype.

Stage 4 After everyone in the corporate team is satisfied with the prototype, sent it to friends and volunteers to test in order to identify additional bugs to fix and to identify additional requirements. Return to stage 2 to start the next iteration.

Stage 5 After no more complaints from most of people in the small group of volunteers with the prototype, integrated it into the live site and ready for be tested by a wider range of users.

This method is slow and wastes a lot of efforts because some small features will be removed if they turn out to be unnecessary or undesired, but this is the best method for this project. In order to provide as good user experience as possible, advice and opinions from users must be considered, but in most cases because of the difference in expertise between developers and other users the original idea from the developers is a better idea. Developing first and making changes later can give the user a clear concept of the project, so that feedback and advice will be more specific and relevant. Additionally, having an implementation can help define the technical boundary of the project as a lot of ideas from users are difficult or impossible to achieve at the current stage of the project.

4 Implementation and Analysis of a prototype

As a result of this master's thesis project a web service was developed that can search for YouTube videos using key words, allow a user to chop any YouTube Video into several clips that include one or more concepts or facts that need to be memorized, fine tune these video clips to the most suitable time range, add questions and other related information to each clip, and finally put these clips into a study page, so that learners will first see the question and can try to answer it, and the answer will be shown as a (edited) video clip.

This chapter describes the functionality and architecture of the prototype service, some of the difficulties that were solved, and some of the difficulties that were not solved during the project. Section 4.8 describes an analysis of some of the data collected from actual use of the resulting application. The chapter concludes in section 4.9 with a description of technical requirements

4.1 General prototype overview

The overall architecture of the prototype is shown in Figure 4-1. Users can access the service through all kinds of devices, including web browsers on a laptop or a tablet, or a smartphone application (such an application will be developed in the future). Because of limited hardware and bandwidth resources of the prototype server, YouTube videos will *not* be stored on it and the server will *not* provide the video service itself, instead the prototype server will call the YouTube API in order to utilize the resources of the YouTube server. Otherwise the hardware and bandwidth cost becomes very large as the number of users grow and these costs would be unrealistic for a startup company[*].

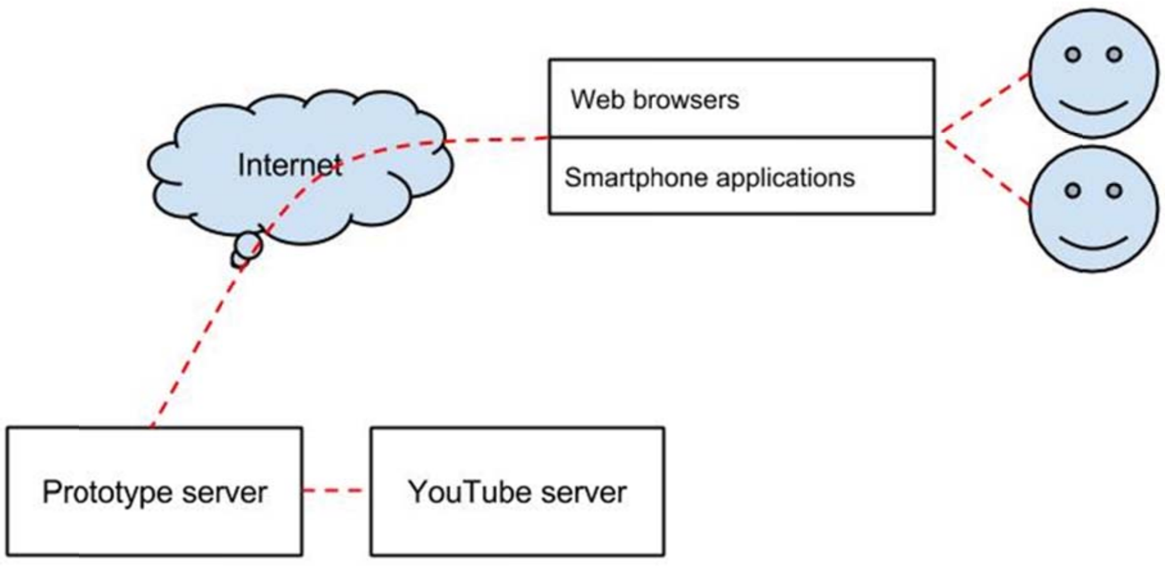


Figure 4-1: Prototype overview

[*] Methods to address the scalability of web services has been addressed by a recent thesis by Md. Iqbal Hossain and Md. Iqbal Hossain "Dynamic scaling of a web-based application in a cloud architecture"[41].

The programming language used for the inner functions is JavaScript. However, the YouTube API provides different Data APIs for different languages, such as Go, Java, and .NET. However, to keep the project simple I chose to use fewer languages. Additionally, because this project focuses on the front-end, JavaScript was an easy and obvious choice. Another reason for choosing JavaScript was that it is one of the most popular programming languages today, so there are lots of discussion groups, open-source plugins, and tools on the Internet. All of this material made using JavaScript a very smooth development and study experience.

4.2 Web interface design

The web interface includes a video editor mode and a video study mode, this section explains how each of these work.

4.2.1 Video editor mode

There are two steps before a user can invoke the video study mode. Step 1 is shown in Figure 4-2. In step 1 the user can:

- Search for videos by key words via the YouTube Data API.
- Choose a specific video in the search results and play it in a video player on a web page.
- Go to anywhere in the video by dragging the progress bar under the video player.
- "Chop" a video clip at any time point that the user sees fit (for example, as an answer to his or her question), then a part of the video will be "chopped" (i.e., the start and end points of the clip will be stored). The time range for a clip is programmed as follows:

```
   Time when chop button is clicked - 10 seconds ~ the time when chop
button is clicked + 10 seconds.
```

- If there is not enough time (on either side of a "chop" operation), e.g. a user clicks the chop button at the beginning or the end of the video, then the range will become:

```
   0 ~ time when chop button is clicked + 10 seconds.

   Time when chop button is clicked – 10 seconds ~ the end time of the
video.
```

- After the user goes through the whole video and chops out the clips he or she wants, then the user can go to step 2 to fine tune and edit these video clips (of questions/ answers). In Step 2:
  - Play and review each clip.
  - Adjust the time range for each clip by dragging the progress bar under the player.
  - Add question and text answer for each clip, while the clip itself is the answer to the question.

Now that the user has created the studying materials he or she can study using the video study mode (described in the next section).

The layout for the video study mode will change according to the screen size, as shown in Figure 4-2 and Figure 4-3.

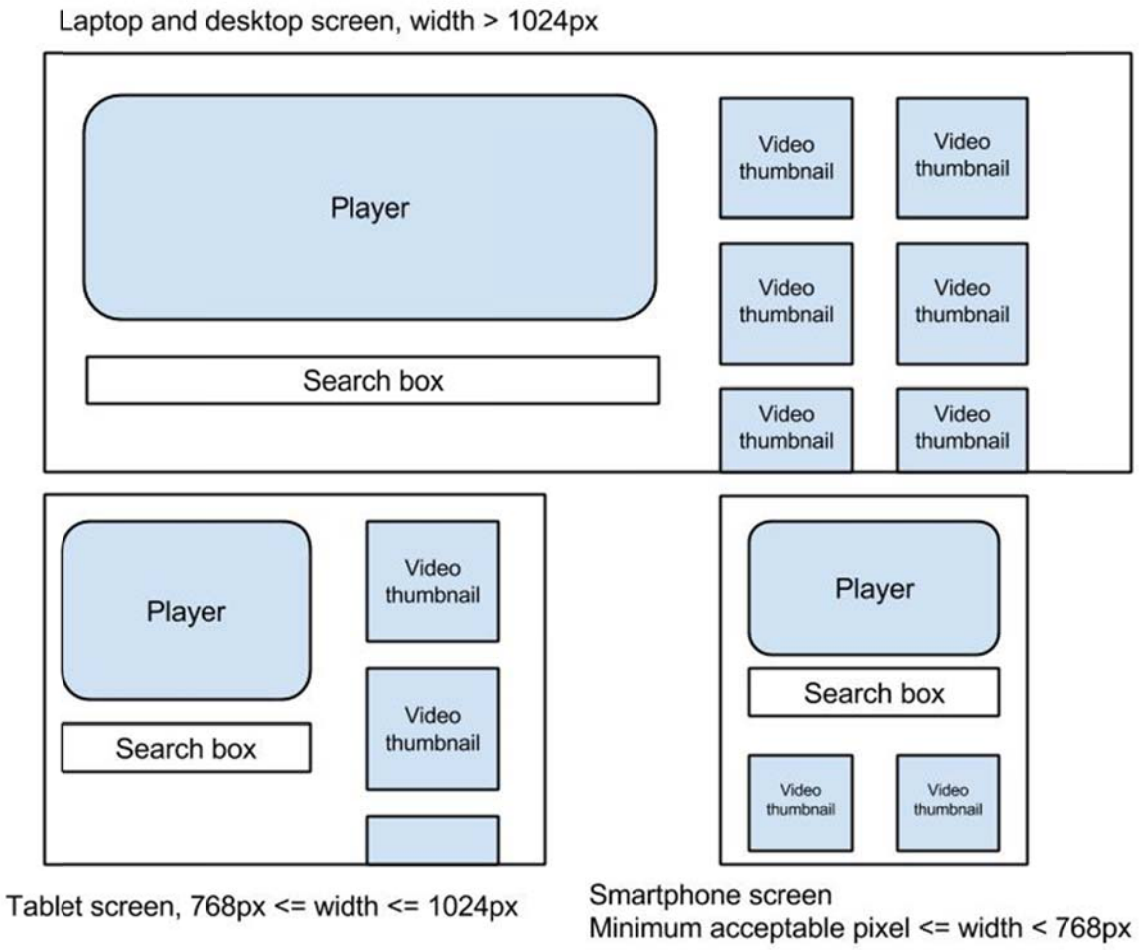


Figure 4-2: Editor mode: step 1 layout with different screen sizes

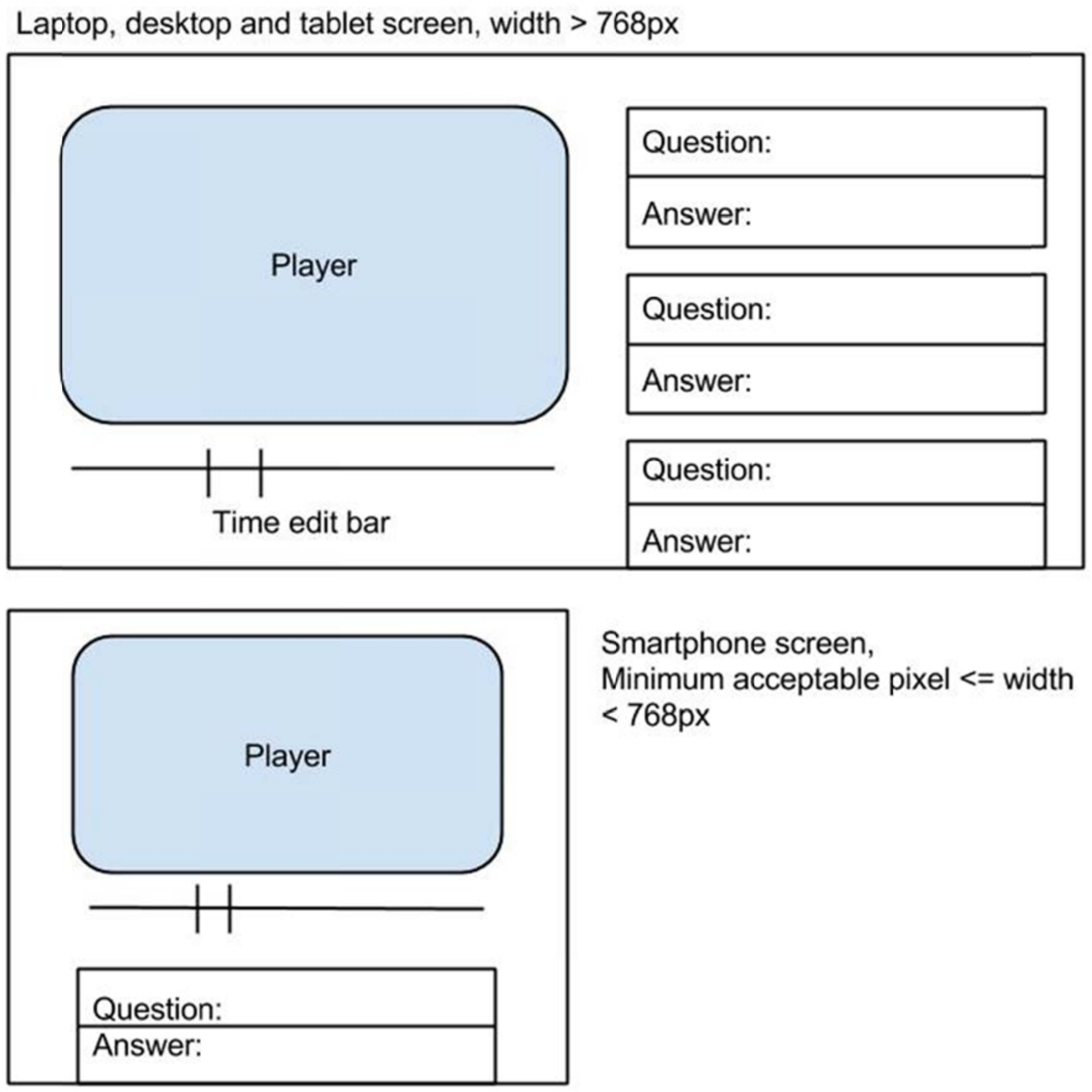


Figure 4-3: Editor mode: step 2 layout with different screen sizes

4.2.2 Video study mode

The interface for the video study mode is shown in Figure 4-4. In this mode the user can try to answer the questions he or she created earlier using the video editor mode and then checks those video clips he or she edited as answers to each question.

The layout for this page is similar on different screen sizes, as the basic layout looks good on all kinds of screens.

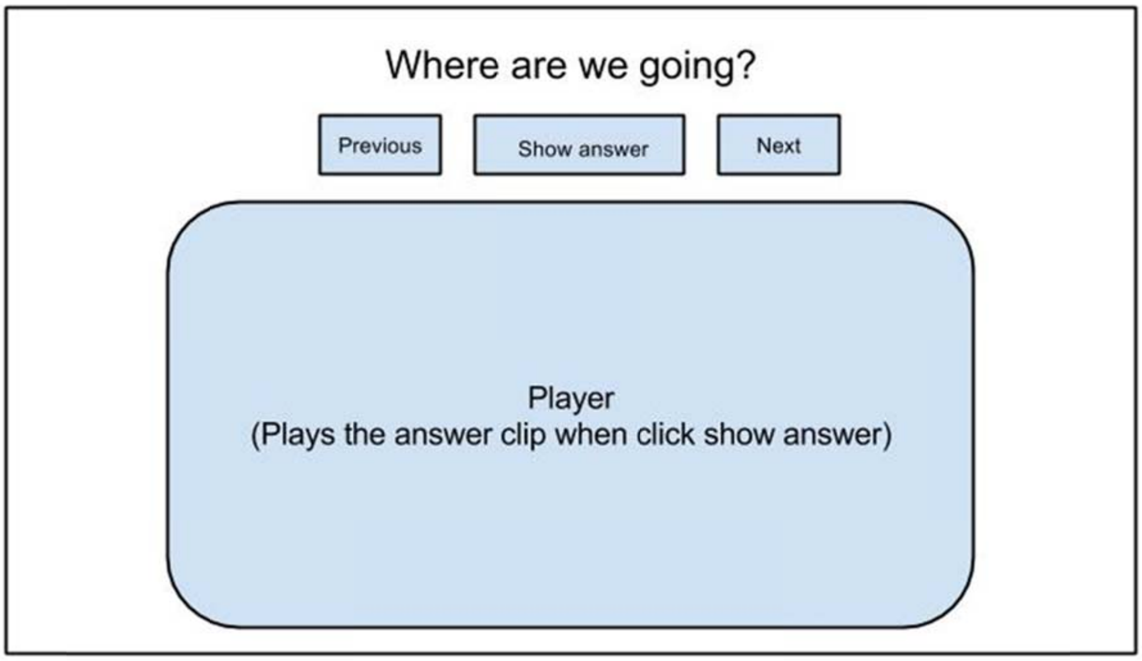


Figure 4-4: Study mode layout

4.3 Web service's inner workings

This section explains web service's inner workings. The first subsection describes the data that is used. The second subsection described how to use the YouTube Player. Finally, the third section describes how to use the YouTube Search API.

4.3.1 Data

Only one object with several properties[42] is used in the project. This data object later will be stored in the database. The properties of this object and the associated values are shown in Table 4-1.

Table 4-1: Data object properties and values used in this project

Property name	Value contains
videoId	videoId contains a unique 11-character YouTube video ID that identifies the video. This value will be used in calls to the player to play a specific video. Storing different videoIds make it possible for user to chop clips from different videos.
videoStart	videoStart contains the start time of a clip.
videoEnd	videoEnd contain the end time of a clip. Combining the start time and the end time defines a time range of a video to play (this is understood by the user as "clip").
videoQuestion	videoQuestion contains the question user entered for this clip.
videoAnswer	videoAnswer contains the textual answer the user entered for the question. Even if there is a video clip as the answer, it is nice to have a textual answer as an addition[*].
videoLongAnswer	videoLongAnswer contains the textual answer (it is similar to the videoAnswer but is longer in text).

DOM storage was used in the project to avoid the need for database connections and integration, as it seemed easier to develop under a pure HTML/CSS/JavaScript environment then do database integration. Here localStorage in the DOM storage is used since the data that is stored can be used in a new page opened in a new tab or window. This access suits the purposes of the development environment as information needs to be transferred from video editing mode to video studying mode. Using localStorage the object generated in the editing mode page is transmitted to the studying mode as the learning material.

JavaScript Object Notation (JSON)[43] is used as the format to transmit data objects because it is powerful and very easy to read, thus it is easy to debug. In the video editing mode the JSON.stringify()[44] method is used to convert a generated object into JSON format. In the video studying mode JSON.parse()[45] is used to parsed a JSON String sent by the video editing mode page.

Later in integration when database connections is needed localStorage will not be used any more and the some of localStorage properties will be converted and adding to the database. The property videoQuestion, videoAnswer, videoLongAnswer will be using the existing column to store data in the database as shown in table 4—2. The SQL query to create new columns is listed in appendix B.

[*] It is important to have a textual answer even if there is a video clip as some users may not be able to see the video clip, but they might use a screen reader to hear the textual answer.

Table 4-2: New added column in database converted from localStorage

Property name	Database Column name	New or existing column
videoId	video_id	New
videoStart	video_start	New
videoEnd	video_end	New
videoQuestion	q_title	Existing
videoAnswer	a_text	Existing
videoLongAnswer	la_text	Existing

4.3.2 YouTube Player

The YouTube Player provides the core functionality of this thesis project, nothing will work properly without it. When calling a player to show a video clip on the page, a function with following value for the variable "params" needs to be called.

```
    var params = { allowScriptAccess: "always"};
```

Setting allowScriptAccess (as shown above) enables an instance of Flash loaded from another domain to call JavaScript[46]. This value for the allowScriptAccess parameter must be specified in the HTML code that loads a SWF file (in this case a YouTube video) and it gives this SWF file the ability to perform outbound URL access. When there is no value is set for allowScriptAccess, then the value is set to "sameDomain" by default, thus SWF can only communicate with a HTML page from the same domain. When the allowScriptAccess parameter is set to "always", then the SWF file can communicate with a HTML page from a different domain than itself.

Setting the element id of the embedded Flash is used to identify the player. If more than 1 player is needed, then the programmer must define more than one element id (each with different id values). The code below sets the id to be "ytPlayer":

```
    var atts 1 = { id: "ytPlayer"};
```

The SWFObject 2 dynamic publishing method is a standards-friendly method to embed Flash content. It offers a JavaScript API that provides a tool set for embedding SWF files and retrieving Flash Player related information[47]. Table 4-3 explains the arguments used in swfobject.embedSWF() call below:

```
    swfobject.embedSWF("http://www.youtube.com/apiplayer?" +
  "version=3&enablejsapi=1&playerapiid=player1", "videoDiv",
  "480", "295", "9", null, null, params, atts 1);
```

Table 4-3: Swfobject.embedSWF arguments.

Arguments	Explanation
"http://www.youtube.com/apiplayer?" + "version=3&enablejsapi=1&playerapiid=player1"	The URL of the SWF content.
videoDiv	The id of the HTML element.
480	The width of the SWF content in px.
295	The height of the SWF content in px.
9	The Flash player version.
params	Params defined above.
atts_1	Attribute defined above.

The SWFObject 2 offers two methods to embed Flash Player content:

- **Static publishing method** uses standards compliant markup most of the time and uses JavaScript to do what markup could not do. This method has the best embed performance and runs on devices and browsers with poor JavaScript support.
- **Dynamic publishing method** uses JavaScript heavily and it works very well with scripted applications, hence this method is used in this project.

To use dynamic publishing method, the programmer must first define an id for the flash content in HTML and the player will be shown. For example:

```
<div id="videoDiv"></div>
```

Then include the SWFObject JavaScript library in the head or the button of HTML page with:

```
<script type="text/javascript">
google.load("swfobject", "2.2");
</script>
```

If everything is correct and player is loaded on the page, then the function onYouTubePlayerReady() will be automatically called and some initializations can be done.

The following code allows the player execute the function updatePlayerInfo()every 0.25 seconds. In the updatePlayerInfo() function the real time information about the video (such as current time, video duration, volume, etc.) are displayed. Later this function will be used to solve problem due to a malfunctioning YouTube API, see 4.5.1.

```
setInterval(updatePlayerInfo, 250);
updatePlayerInfo();
```

4.3.3 YouTube Search API

In order to allow a user to search for YouTube videos, the YouTube Search API is used. First adding the JavaScript client library as follows:

```
  <script src="https://apis.google.com/js/
  client.js?onload=onClientLoad"
type="text/javascript"></script>
```

Then load the client interfaces for the YouTube Analytics and Data APIs:

```
gapi.client.load('youtube', 'v3', onYouTubeApiLoad);
```

To use the search API, requires that an API key be set for the application. This API key can be found in the Developer Console when registration is needed. The key is specified as follows:

```
gapi.client.setApiKey('API key here');
```

Finally to perform a search, the code below will return a collection of search results that match the specified parameters:

```
gapi.client.youtube.search.list({
part: 'snippet',
q: q,
type: 'video',
maxResults: 10
});
```

part: 'snippet' means it will return information contains other properties that identify the result's title, description, and so forth[48].

q: q, the later q is the query string to search for.

maxResults: 10, to only show 10 search results.

To make the search results more readable thumbnails of the videos are necessary in the JSON string of the search results. The thumbnail image URL is item.snippet.thumbnails.medium.url and can be used directly in HTML.

4.4 UX and UI study

User's behavior was studied in order to provide a better user experience (UX) and better interaction via the user interface (UI). These studies were performed with several volunteers selected from among my friends and colleagues. During these studies which took place while developing and changing approaches, several major problems were encountered and solutions were found that improved the UX.

Although some major changes have been made to improve the overall UX, the result of the project is still far from perfect. The number of volunteers, in total 4 (of which 3 were men and 1 were women) thus far is too small and the results are biased because most of these volunteers have the IT background hence we can only draw some weak conclusions. However, this testing during the development process results in a number of changes to the design. There are the areas that have benefited from this testing are described in the following subsections. The design will continue to evolve as additional feedback is received. Some early results from the website are given in section 4.8.

4.4.1 "Go back" feature

There are two steps in video edit mode, as section 4.2.1 described:

1. Step 1 is for searching videos and chop clips from the selected video. (Later the chop button was moved to step 2)

2. Step 2 is for fine tuning the time range of the clips and adding questions and text answers.

One of the major debates was should a user be able to explicitly switch between steps 1 and 2, specifically can a user go back to step 1 from step 2 after he choosing the video he (or she) wants. The initial studies with volunteers really gave no answer. Most volunteers preferred to have a "go back" feature when they hear about it for first time. A typical comment for why this function was desired is because "I will make mistakes choosing the wrong video, and I will want to go back to correct it." However, this functionality turns out to cause trouble for first time users because it complicates the user interface and because it can lead to unexpected behaviors. The interface that allows the users to selected the first or second step is shown in Figure 4-5.

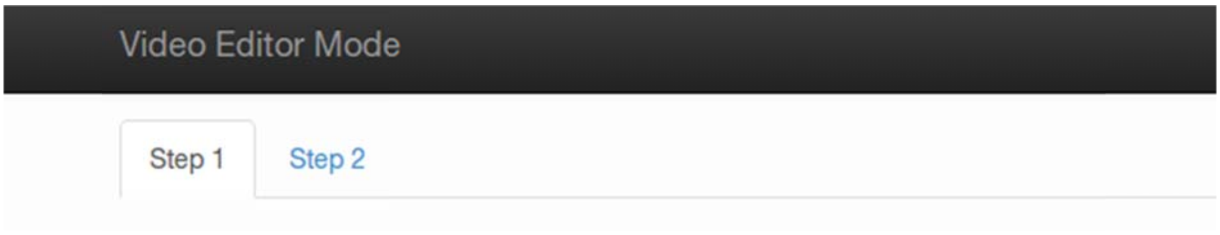


Figure 4-5: Design with switch steps

To simplify the interface another approach is to remove the step 1 and step 2 buttons and make the process of choosing video and editing it irreversible. In this approach the user will see step 1 when he first goes to the editing page, then there will be a "go to next step" button on the page that is only clickable after a video has been selected. After user the user is automatically presented with the step 2 page and there is no "go back" button that would enable a user return to page 1 to select a new video. The only way to go back and select a new video is to refresh the page. Table 4-4 summarizes the advantages and disadvantages of these two approaches.

Table 4-4: Advantages and Disadvantages of the two approaches

	Advantages	Disadvantages
Reversible	The user can go back and select a different video if desired.	Though the buttons have the label "step 1" and "step 2" on them, user will first notice those 2 buttons and start clicking them. If user goes to step 2 without a selected video, there will be nothing to show and become very confusing.
Irreversible	The interaction process is straight-forward and clear for most users.	The only way to change a video is to refresh the page.

According to Reichenstein, "Web design is engineering: it's not about finding perfection, it's finding the best compromise."[35] A compromise was made and the original design that is reversible design was selected. However, new problems arise from this approach. Some of the questions that arise are: Since users can go back and choose another video, does this mean that users can chop clips from different videos? How can we decide a user wants to choose another video to chop or is simply wandering around through the pages?

After many debates and meetings, a few changes were made to provide a better interactive user experience. The code to implement the first alternative is included in Appendix A. As a result:

1. The user can chop clips from different videos, thus the user can switch between step 1 and step 2 and choose as many as videos he (or she) wants and chop from them clips in order to create a course.

2. When there is no video selected, then the step 2 button is grey and unclickable.

4.4.2 Position of the "Chop" button

In the original design, the chop button was in step 1, when the user clicks the chop button the chopped video clips will be displayed in step 2 along with input boxes for the questions and text answers. Since there is no room left on the page for them the input boxes are shown only in step 2. Therefore, the user has to go to step 2 to see the clips he just chopped.

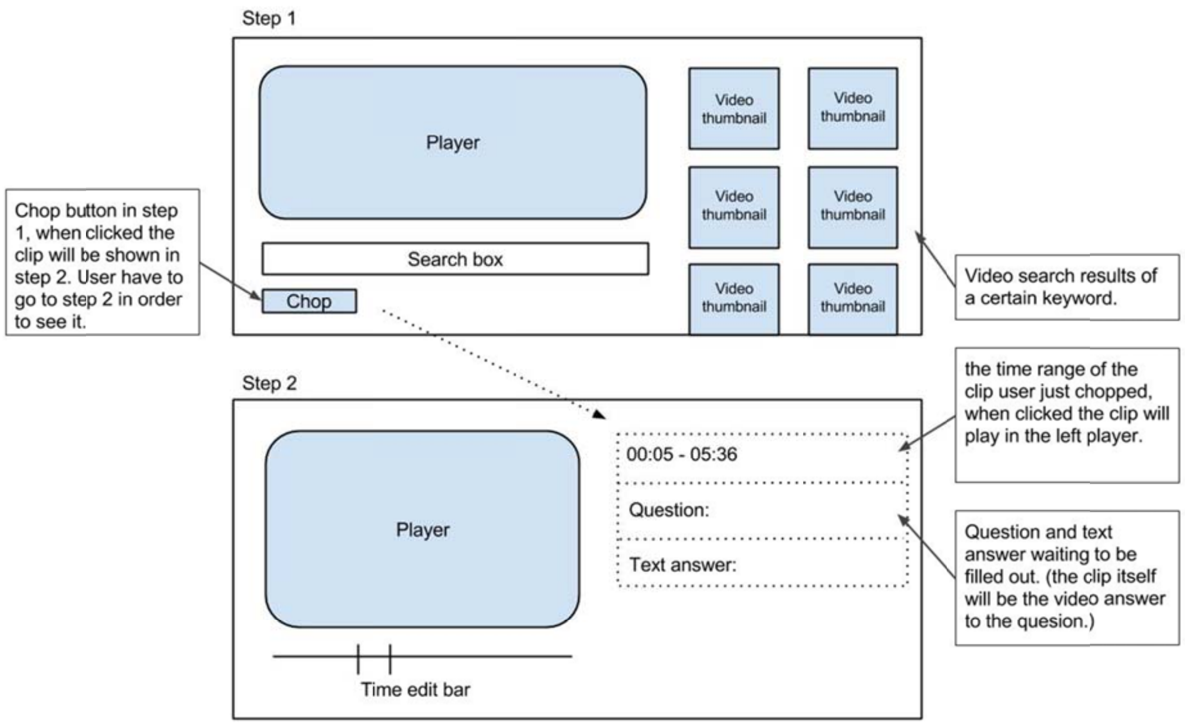

Step 1
Player
Video thumbnail
Video thumbnail
Video thumbnail
Video thumbnail
Video thumbnail
Video thumbnail
Chop button in step 1, when clicked the clip will be shown in step 2. User have to go to step 2 in order to see it.
Search box
Chop
Video search results of a certain keyword.
Step 2
the time range of the clip user just chopped, when clicked the clip will play in the left player.
Player
00:05 - 05:36
Question:
Text answer:
Question and text answer waiting to be filled out. (the clip itself will be the video answer to the quesion.)
Time edit bar

Figure 4-6: Layout of the chop button in step 1.

Most disliked this behavior because they did not know if they managed to chop or could not see what they had chopped. If there is a long list of chopped clips, then the user will forget what clips he has previous chopped, since there is nothing to show these clips in step. The user's only means to see what clips have been selected is to switch from step 1 to step 2.

A decision was made to move the chop button to step 2 and to make another slight change. The results are:

- Chop button moved from step 1 to step 2. Now the user can see exactly what he has chopped on the right side of the screen and can remove clips if desired.
- The purpose of step 1 is now only for searching and selecting a desired video.
- Adding the feature "play from start" (via a new button) and a progress bar, the player is step 2 can be used to play clips - as well as the whole video. When clicking on the time range on the right side to play the clip, the user will need to play the whole video and a progress bar lets the user go to any time offset in the video where he wants to chop more clips.
- In step 2 the player will automatically play the video selected from step 1, so user does not need to click the "play" button. If a user switches frequently between step 1 and step 2, this new behavior will save the user a lot of time and effort.

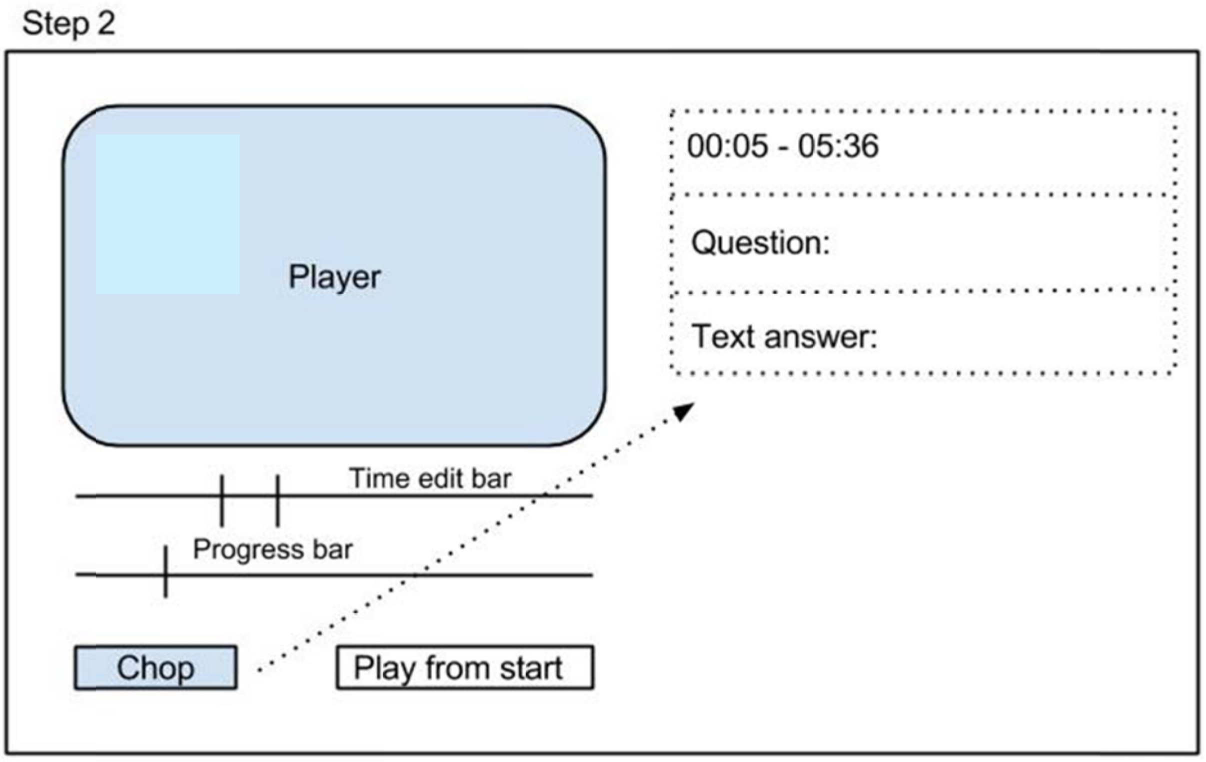


Figure 4-7: Layout of the chop button in step 2.

4.4.3 Editable clips

The editable clips feature has been constantly improving. The description that follows is based upon the final result of the current stage of the development.

When a user plays a clip, there will be a time edit bar shown beneath the player for fine tuning the time range for the clip. The minimum value and maximum value for the edit bar are:

```
start time of the clip – 30 seconds
end time of the clip + 30 seconds
```

The margin of 30 seconds was selected because none of the volunteers needed more and because if a clip is more than 80 seconds long it becomes too long to be an answer for the spaced repetition method. Also it is better to keep the values small as this makes the editing more accurate. Some of the improvements that were made during the development of this feature are:

- The time range of the clip will change while editing time range using the edit bar.
- The interface will automatically play the changed clip. For example, when the user changes the start time from 00:50 to 00:30, once the mouse has release the bar, the new clip will be played from 00:30. If the new clip does not play user will not be able to know if the new time point is what was desired or not because the user will cannot directly see the outcome of the new clip starting point.

4.5 Difficulties

Two major difficulties were encountered during the project: one of the YouTube API functions that is crucial to make the video clip play properly does not work and the waiting time before playing the next clip is sometimes too long. The subsections below give further details of these two difficulties.

4.5.1 Malfunction of the YouTube API

According to the YouTube JavaScript Player API Reference[49], the `loadVideoById` fuction can be called to load and play a video with the selected time range. This documentation states:

```
    Argument syntax
  player.loadVideoById(videoId:String, startSeconds:Number,
suggestedQuality:String):Void

    Object syntax
  player.loadVideoById({videoId:String, startSeconds:Number,
endSeconds:Number, suggestedQuality:String}):Void
```

Additionally, the YouTube JavaScript Player API Reference states: "The optional `startSeconds` parameter accepts a float/integer. If it is specified, then the video will start from the closest keyframe to the specified time. And the optional endSeconds parameter accepts a float/integer. If it is specified, then the video will stop playing at the specified time."[49]

This is the most suitable function for our purpose. For each clip, its videoId, startSeconds, and endSeconds are stored. When a clip is to be played, `loadVideoById` is called with the correct parameters and the clip will play from the start second to end second.

The function worked for a while until one day it failed to stop at the specific end second, instead it played all the way to the end. Although YouTube has prepared an API and documentation, no one else exactly knows how their API works. As a result of this problem a substitution was needed for this function. This project used the `seekTo()` function as a replacement. This function is defined as:

```
  player.seekTo(seconds:Number, allowSeekAhead:Boolean):Void
```

This function seeks to a specified time in a video. The `seconds` parameter identifies the start time where the player should start playing the video. The player will actually start playing at the closest keyframe before that time. However, if the player has already downloaded part of the video within which the user is seeking, the player will start playing at the closest keyframe before or after the specified time[49]. So the combination of `loadVideoById` and `seekTo` allows the player to play video starting from a specific time.

To enable the player to stop playing at a specified time, a more complicated solution is used. In the project's code, the function `onYouTubePlayerReady()` is automatically called by the player once it loads the video, then the function `updatePlayerInfo` is called every 0.25 seconds to update the player information (such as the current playing time) so it can be displayed on the web page, in this case the solution was to call the `stopVideo()` function when the value of current playing time is greater than the value of the specified stop time.

4.5.2 Efforts to reduce waiting time when switching clips

The method described above successfully overcomes the YouTube API malfunction. Now for every video, no matter how many clips has been chopped from it, when the player plays these clips the video is only loaded once. After this the switching between clips is done simply by seeking to different start time offsets. This reduces the waiting time between each clips, because player does not need to load the video every time the user changes between clips within the same video, but the user will still feel a lag between clips, just as one will have to wait a little while when dragging the progress bar of any video to a part of the video that has not yet been downloaded by browser from YouTube.

To reduce the waiting time between clips and to provide a better UX, a preloader was designed. This preloader utilize the resources of the YouTube server and the user's device(s) yet does not add any burden to the Sharplet server – because the extra work is being placed on the *YouTube server*.

The preloader consists of two players rather than one. One player is always on top of the other one, so it looks as if there is only one player. The idea is that when player1 is playing clip 1, player2 is playing clip 2 at the same time beneath player1. So when clip 1 finishes playing and the "next clip" button is clicked, player2 will be moved on top of the player1 and will cover it while playing clip 2 from the start. Since clip 2 has already been played for a while beneath player1 (i.e., during the playout of clip 1 on player1), the browser has already downloaded some part (if not all) of clip2, the waiting time for switching from clip 1 to clip 2 become nearly zero. When player2 is playing clip 2, player1 is now is playing clip3 simultaneously in the background, and so on. The two players alternate, one is on top playing the active clip while the other one is playing the next clip in order to cache some portion of the next clip in the background. This use of two players and the clip list is shown in Figure 4-8.

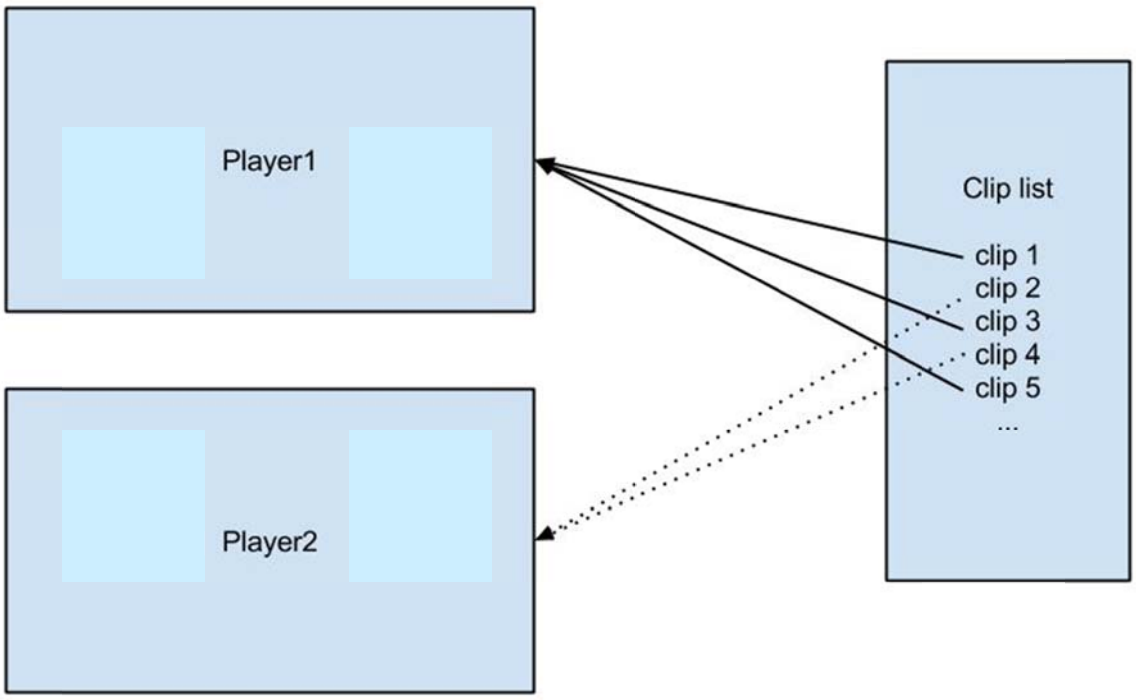


Figure 4-8: How preloader works

This adds twice the workload to the YouTube server and the user's devices, because YouTube always needs to play two parts of video to two players and the user's devices will open two YouTube player instances. In theory this is not a problem because YouTube server has plenty of resources and user's devices has even more. This tradeoff is necessary to make the switching between clips smoother.

One way to reduce this workload would be to add a local caching web proxy in the user's device. With such a caching proxy the clip would only need to be fetched from the YouTube server once. The browser would still fetch the two copies (once for each player) – but since the second copy is coming from the local cache there is no extra load on the YouTube server nor does a second copy have to be transferred to the device.

Unfortunately, the preloader does not work all the time. In the test environment there was roughly a 1 in 10 chance that the other player will become stuck in the background. When it was its turn to play it was still stuck at the beginning of the video clip. This behavior made this function somewhat unusable, hence there is a need for future research to solve either the problem with the YouTube API, to solve the problem of this preloader solution, or the parallel processing to do preloading and caching could be built into the web browser.

4.6 System performance analysis and tuning

The system performance analysis in this chapter is done using Chrome for Linux 64-bit on localhost[*] with pure static video editor mode HTML/CSS/JavaScript files.

One of the methods to improve total loading time on browser is to use minified[†] CDN files[‡] for Bootstrap, jQuery, jQuery UI. Figure 4—9 compares the loading time between before and after using minified CDN files in millisecond by loading video editor page for 5 times each, the loading time is slightly improved using minified files. On average the minified CDN file takes 5.96 % less time to load.

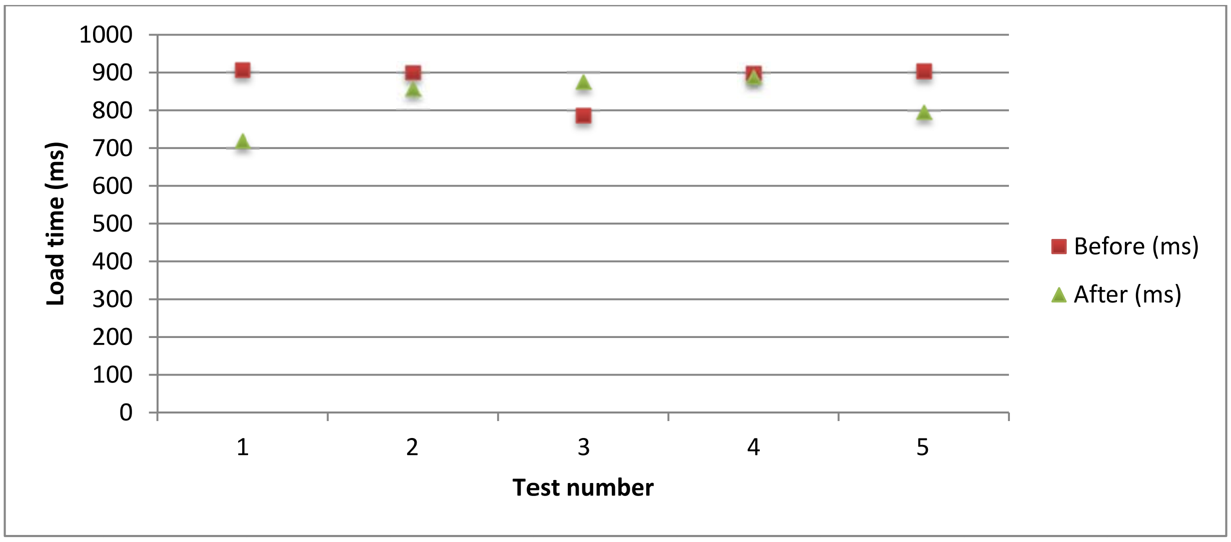
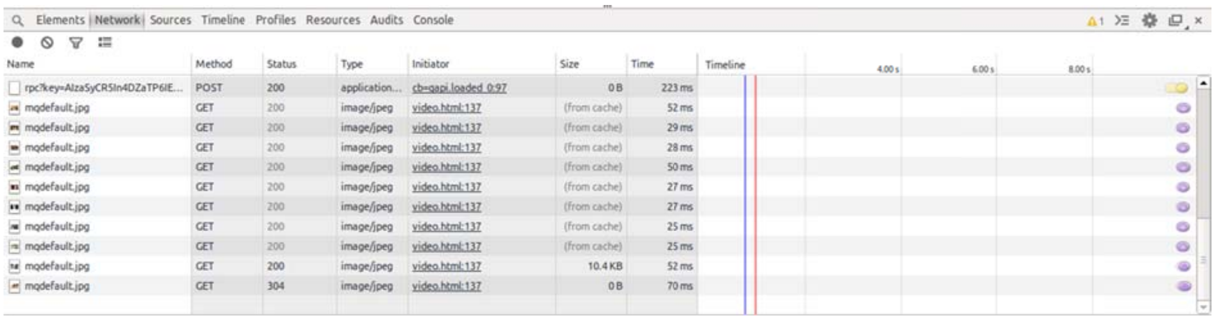


Figure 4-9: Comparison of loading time between before and after using minified CDN files

Figure 4—10 shows the time for loading the Google API and the time for downloading each thumbnail of the video. Due to the fact that most of modern browser can perform multiple GET requests in parallel, the total time to load the page is less than the sum of the time show in the figure. The testing browser Chrome as well as other popular has 6 connections by default[50].



Name	Method	Status	Type	Initiator	Size	Time
rpc?key=AIzaSyCR5In4DZaTP6IE...	POST	200	application...	cb=gapi.loaded_0:97	0 B	223 ms
mqdefault.jpg	GET	200	image/jpeg	video.html:137	(from cache)	52 ms
mqdefault.jpg	GET	200	image/jpeg	video.html:137	(from cache)	29 ms
mqdefault.jpg	GET	200	image/jpeg	video.html:137	(from cache)	28 ms
mqdefault.jpg	GET	200	image/jpeg	video.html:137	(from cache)	50 ms
mqdefault.jpg	GET	200	image/jpeg	video.html:137	(from cache)	27 ms
mqdefault.jpg	GET	200	image/jpeg	video.html:137	(from cache)	27 ms
mqdefault.jpg	GET	200	image/jpeg	video.html:137	(from cache)	25 ms
mqdefault.jpg	GET	200	image/jpeg	video.html:137	(from cache)	25 ms
mqdefault.jpg	GET	200	image/jpeg	video.html:137	10.4 KB	52 ms
mqdefault.jpg	GET	304	image/jpeg	video.html:137	0 B	70 ms

Figure 4-10: Searching time on Chrome

[*] This means that the web server is running on the same computer as the web browser, hence there is not delay caused by having to send the traffic between the two across a network – where the packets might have to compete for network bandwidth with other traffic.

[†] Minification in JavaScript is the process of removing all unnecessary characters from source code.

[‡] Those files need to be downloaded and loaded on user's browser before using Bootstrap, jQuery, and jQuery UI.

4.7 Integration of the new feature into the live Sharplet web application

This chapter explains the integration of the feature into the live Sharplet web application. The section begins with a description of the development environment. Next the configuration and data base connections are described. Finally the design consistency is examined.

4.7.1 Development environment

The proposed new feature was developed in pure HTML/CSS/JavaScript first and then integrated into the existing live Sharplet environment. To create a development environment similar to a live website locally, the following software shown in Table 4-5 is required.

Table 4-5: Backend softwares

Softwares and OS	Description	Installation note
Ubuntu 13.10 64-bit[51]	One of the most popular Linux distributions	
Java SE Development Kit 7[52]	A development environment for applications using Java programming language.	The development computer is Ubuntu 13.10 64-bit so choose the Linux x64 version to download and install.
Glassfish[53]	Glassfish is an open-source application server project for Java EE platform.	Download from website and install.
PostgresSQL[54]	Database using in the project, an open source object-relational database system.	On Ubuntu 13.10 to install: `sudo apt-get install postgresql*` To start the database: `Sudo service postgresql start`
Eclipse[55]	An Integrated Development Environment (IDE) for coding and running glassfish server.	Choose "Eclipse IDE for Java EE Developers" version "Juno" to download.
pgAdmin III[56]	An open source tool for administration and development for PostgreSQL.	On Ubuntu 13.10 to install: `Sudo apt-get install pgadmin3`

4.7.2 Configuration and database connection

This subsection explains the configuration necessary in order to run the Sharplet app locally.

1. To avoid the Eclipse development tool trying to perform a validation of js libs, in Window → Preferences →JavaScript →Include Path, add the following Exclusion patterns[57]:

```
  **/*bootstrap*, **/*jquery*, **/*jQuery*, **/acidJs*/,
js/libs/
```

2. Create a new folder for database files, and make sure the use/group "glassfish/glassfish" has read write permissions on the folder, or the current user who runs glassfish is the owner of the folder.

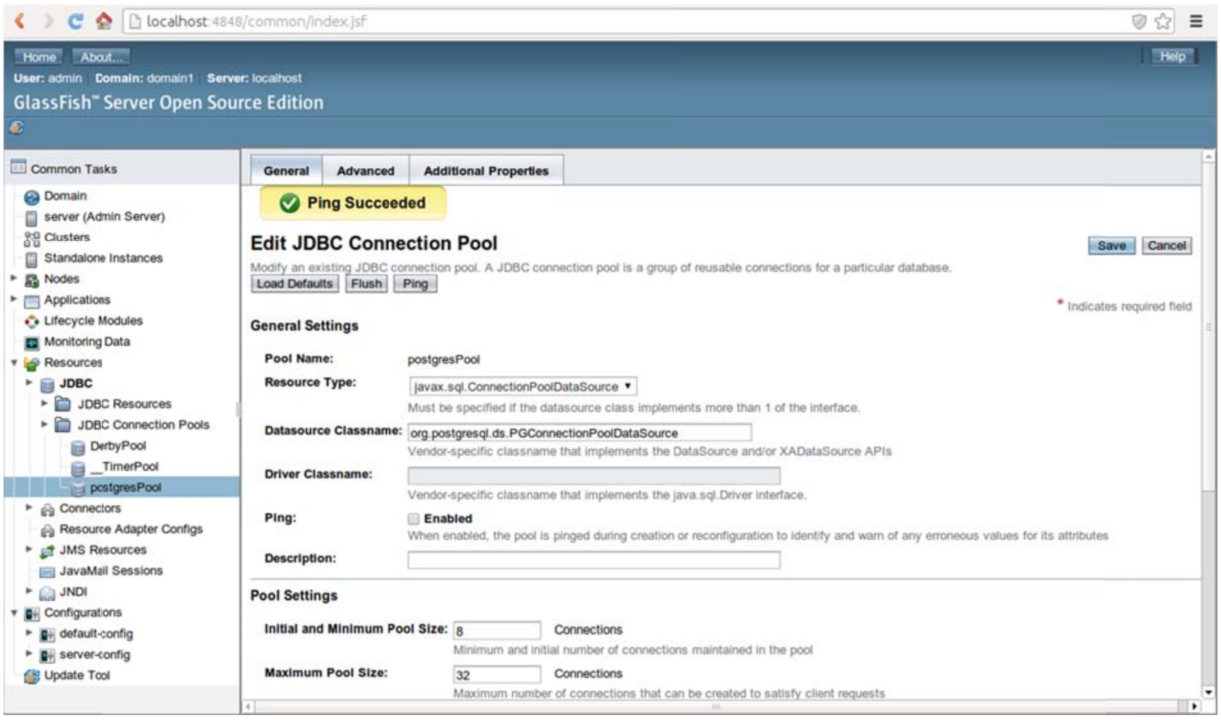
3. To create a database, first start pgAdminIII and connect to local PostgreSQL server. Then create a new database with the appropriate properties. In this project the database is named "flash". Enter the following query:

```
  OWNER = postgres
                              ENCODING = 'UTF8'
                              TABLESPACE = pg default
                              LC COLLATE = 'C'
                              LC CTYPE = 'C'
                              CONNECTION LIMIT = -1;
```

   Then click the "SQL" button to run the query to create the database. (The complete query file is Appendix B).

4. To start the glassfish server in Eclipse, under javaEE perspective in Eclipse, in the "servers" tab → add a new server, select Glassfish v.3.1.2 and right click the server then choose "start".

5. To configure Glassfish, in the browser open the glassfish admin page at localhost:4848. Under resources /JDBC/JDBC Connection pools, add new pool named "postgresPool". Under "JDBC Resources", create a new resource named "jdbc/flash" with the postgres pool. Make sure "ping" on pool is successful. This is shown in Figure 4-11.



Ping Succeeded

Edit JDBC Connection Pool

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.

* Indicates required field

General Settings

Pool Name: postgresPool

Resource Type: javax.sql.ConnectionPoolDataSource
Must be specified if the datasource class implements more than 1 of the interface.

Datasource Classname: org.postgresql.ds.PGConnectionPoolDataSource
Vendor-specific classname that implements the DataSource and/or XADataSource APIs

Driver Classname:
Vendor-specific classname that implements the java.sql.Driver interface.

Ping: Enabled
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes

Description:

Pool Settings

Initial and Minimum Pool Size: 8 Connections
Minimum and initial number of connections maintained in the pool

Maximum Pool Size: 32 Connections
Maximum number of connections that can be created to satisfy client requests

Figure 4-11: GlassFish configuration page

6. To run the sharplet app, in the Eclipse server's tab, right click the glassfish server, add the sharplet project to the server from the menu, then start the server from the same menu. In the browser go to localhost:8181 and the sharplet front page will be shown (as shown in Figure 4-12).

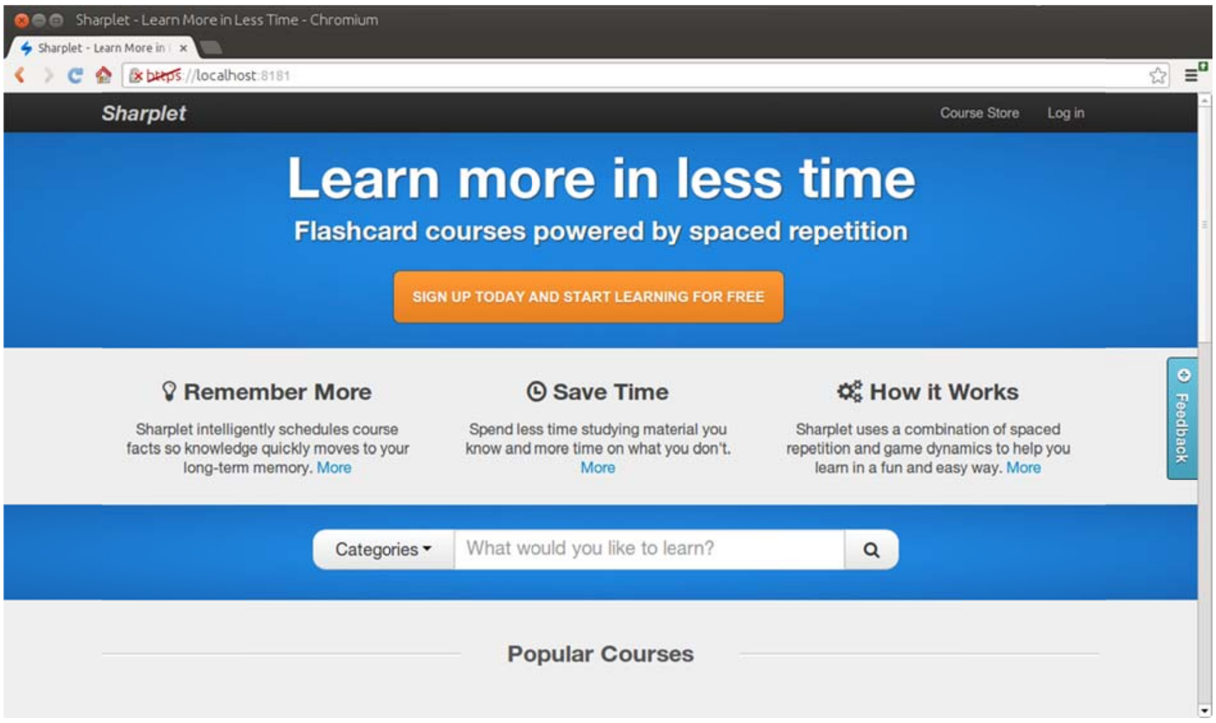

Sharplet - Learn More in Less Time - Chromium
https://localhost:8181
Sharplet
Course Store
Log in
Learn more in less time
Flashcard courses powered by spaced repetition
SIGN UP TODAY AND START LEARNING FOR FREE
Remember More
Sharplet intelligently schedules course facts so knowledge quickly moves to your long-term memory. More
Save Time
Spend less time studying material you know and more time on what you don't. More
How it Works
Sharplet uses a combination of spaced repetition and game dynamics to help you learn in a fun and easy way. More
Feedback
Categories
What would you like to learn?
Popular Courses

Figure 4-12: Sharplet front page on localhost

7. To make your account in the sharplet an administrator, go to pgAdminIII and run the query:

```
  insert into user roles (userkey, username, rolename) values
({your user id from the id column in users}, {your username
from the username column in users}, 'admin');
```

4.7.3 Design consistency

The proposed new feature was developed in pure HTML/CSS/Javascript with almost no styling (such as the font color, typeface, etc.) or using only the default Bootstrap theme. This made it very easy to keep the design consistency when applying an existing style sheet. The video editing mode basically replaces the original Sharplet course create page and retains the same header and footer.

The video study mode (shown inFigure 4-13) is similar to normal study mode as when the learner is in the video study page, he (or she) will see the question first just as in the other study mode, but a video window will be shown to play a video answer when the button "Show answer" is clicked. In addition, the "Know", "Familiar", and "Unknown" feedback buttons appear for learner to decide how well he (or she) remembers the answer. An example of how this feedback is gathered is shown in Figure 4-14.

The range of EF (E-Factor) in Sharplet has been modified to use the values in the range 0 ~ 6, where "Know" = 6, "Familiar" = 3, and "Unknown" = 0, the default value is 3.

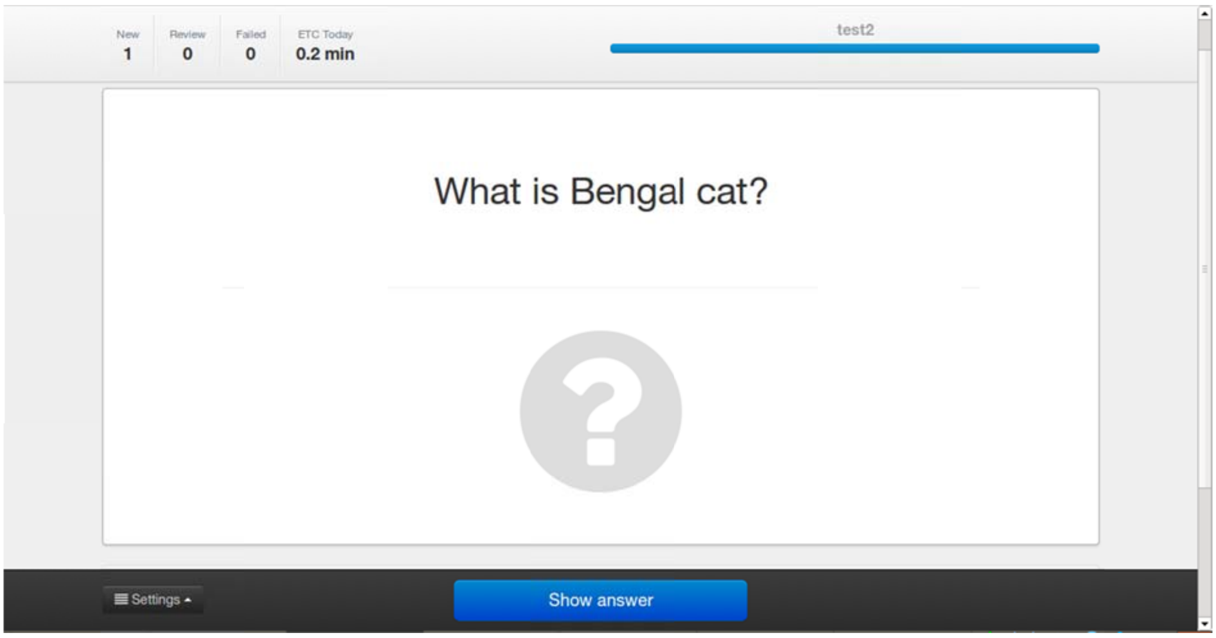


Figure 4-13: Studying mode asking the question.

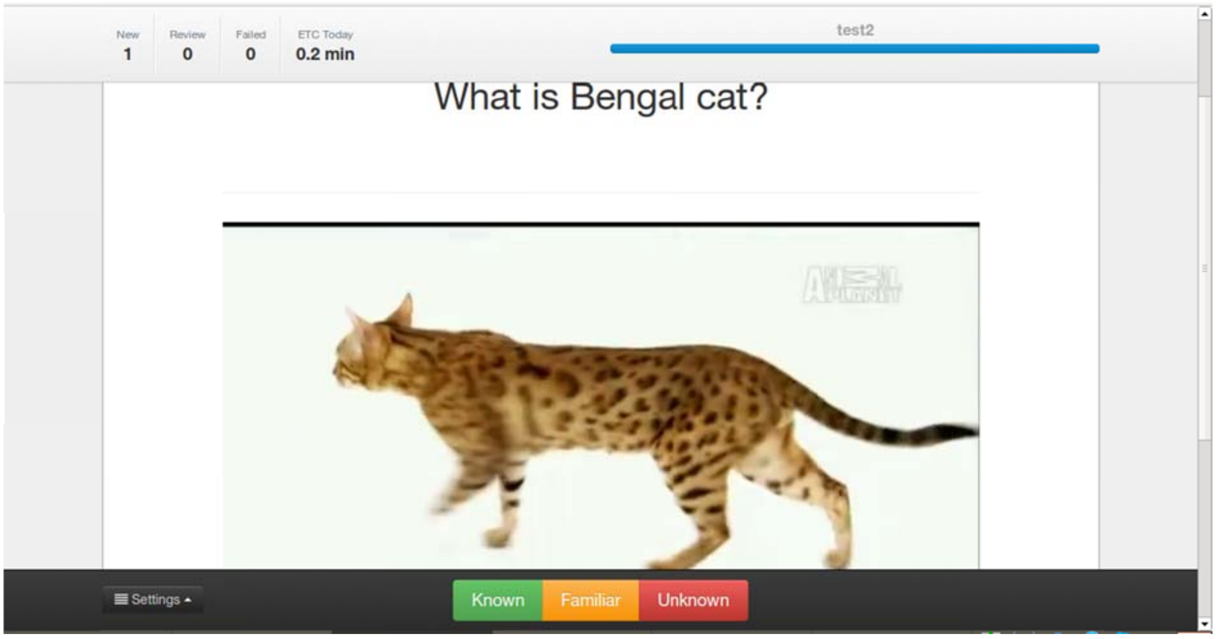


Figure 4-14: Studying mode shows the video answer.

4.8 Analysis of quantitative and qualitative data collected during actual site usage

This section analyzes data collected during actual site usage. The first subsection describes the data collected using Google Analytics concerning the visits to the application's website. This data is presented to show the number of visitors per day and the different operating systems and devices that these users are using. These statistics support the choice of design decisions made during the implementation such as the use of Flash will be a problem for iOS so a native application will be a must because of the number of iOS users, data are shown section 4.9. This is followed by a description of the loading testing that was performed to determine how long it would take to load the pages that the user is going to encounter when using the newly introduced video study mode.

4.8.1 Google Analytics

This subsection shows the data from Google Analytics that indicates the number of visitors per day to the site broken down by different operating systems and devices. A plot of the data is shown in Figure 4-15 and tabular data is shown in Table 4-6 (by OS) Table 4-7 (by type of device). Due to very limited advertisement and limited investment funds during this time, the number of visitors is quite low.

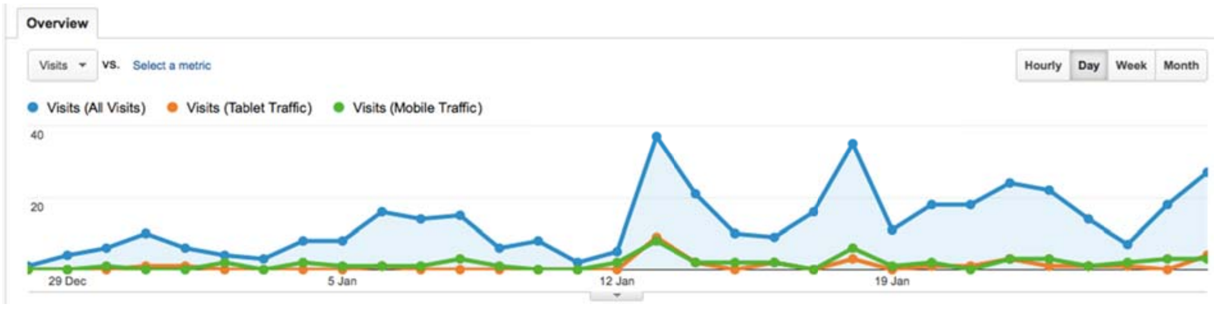


Figure 4-15: Google Analytics showing visits (28 December to 27 January 2014)

Table 4-6: Breakdown of visits by operating system (Data from 28 December 2013 to 27 January 2014)

Operating System	**Visits**	**% Visits**
1. Windows		
All Visits	195	48.39%
Tablet Traffic	0	0.00%
Mobile Traffic	0	0.00%
2. Macintosh		
All Visits	92	23.83%
Tablet Traffic	0	0.00%
Mobile Traffic	0	0.00%
3. iOS		
All Visits	67	16.63%
Tablet Traffic	30	96.77%
Mobile Traffic	37	71.15%
4. Linux		
All Visits	31	7.69%
Tablet Traffic	0	0.00%
Mobile Traffic	0	0.00%
5. Android		
All Visits	8	1.99%
Tablet Traffic	0	0.00%
Mobile Traffic	8	15.38%

Table 4-7: **Breakdown of visits by browser (Data from 28 Days)**

Browser	Visits	% Visits
1. Chrome		
All Visits	161	39.95%
Tablet Traffic	2	6.45%
Mobile Traffic	8	15.38%
2. Safari		
All Visits	93	23.08%
Tablet Traffic	6	19.35%
Mobile Traffic	11	21.15%
3. Firefox		
All Visits	63	15.63%
Tablet Traffic	0	0.00%
Mobile Traffic	0	0.00%
4. Safari (in-app)		
All Visits	44	10.92%
Tablet Traffic	22	70.97%
Mobile Traffic	22	42.31%
5. Internet Explorer		
All Visits	24	5.96%
Tablet Traffic	0	0.00%
Mobile Traffic	1	1.92%

4.8.2 Page loading time testing

The page load time testing in this section is done using Chrome for Linux 64-bit on the development machine. The loading time of the video editor page and the loading time after a click to show the answer on studying page were tested 5 times. These results are shown in Figure 4-16. The average time for loading video clip after clicking the “Show answer” button is 1670 ms and this is too long. The user will find waiting this long undesirable. Because the page loading times were all much too long to be acceptable to users, there was no need to perform these tests large numbers of times.

While efforts have been made to reduce the waiting time, as described in section 4.5.2, this effort failed. Therefore, a major first step in future work will be reducing the waiting time.

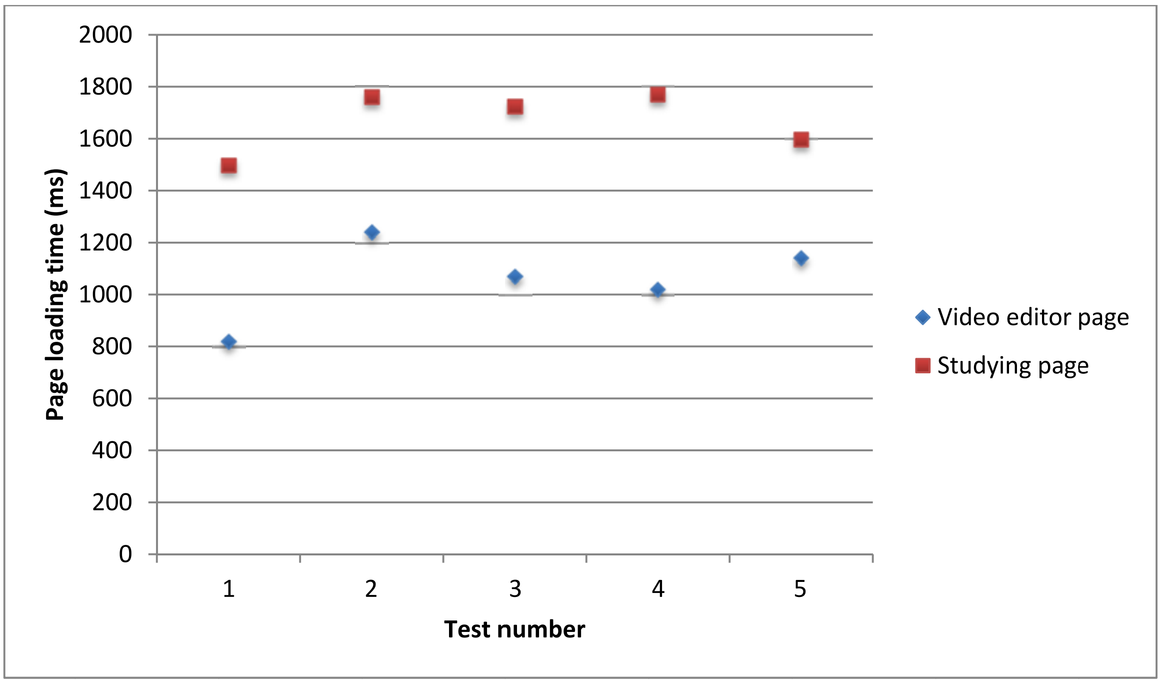


Figure 4-16: Load testing in development environment

4.9 Technical requirements and initial design regarding a mobile application

There are two approaches to developing a mobile application: (1) a web app made for a phone or (2) a native mobile app. A native mobile app can take maximal advantage of the device's functionality, but requires more time and more professional experience in mobile development that required when developing a web app. Since developing a web app is easier, in this project we have adopted this approach. Thanks to the use of responsive design the layout can fit the phone screen without any problem, but there are two concerns:

1. Apple does not allow Flash on any of their portable devices, including iPhones[58]. Despite Google's intentions, Google's YouTube is still in the process of transitioning from Flash to HTML5; hence the project could not avoid using Flash. As a result the YouTube feature in the web app that has been developed is impossible to implement on any of Apple's mobile devices.

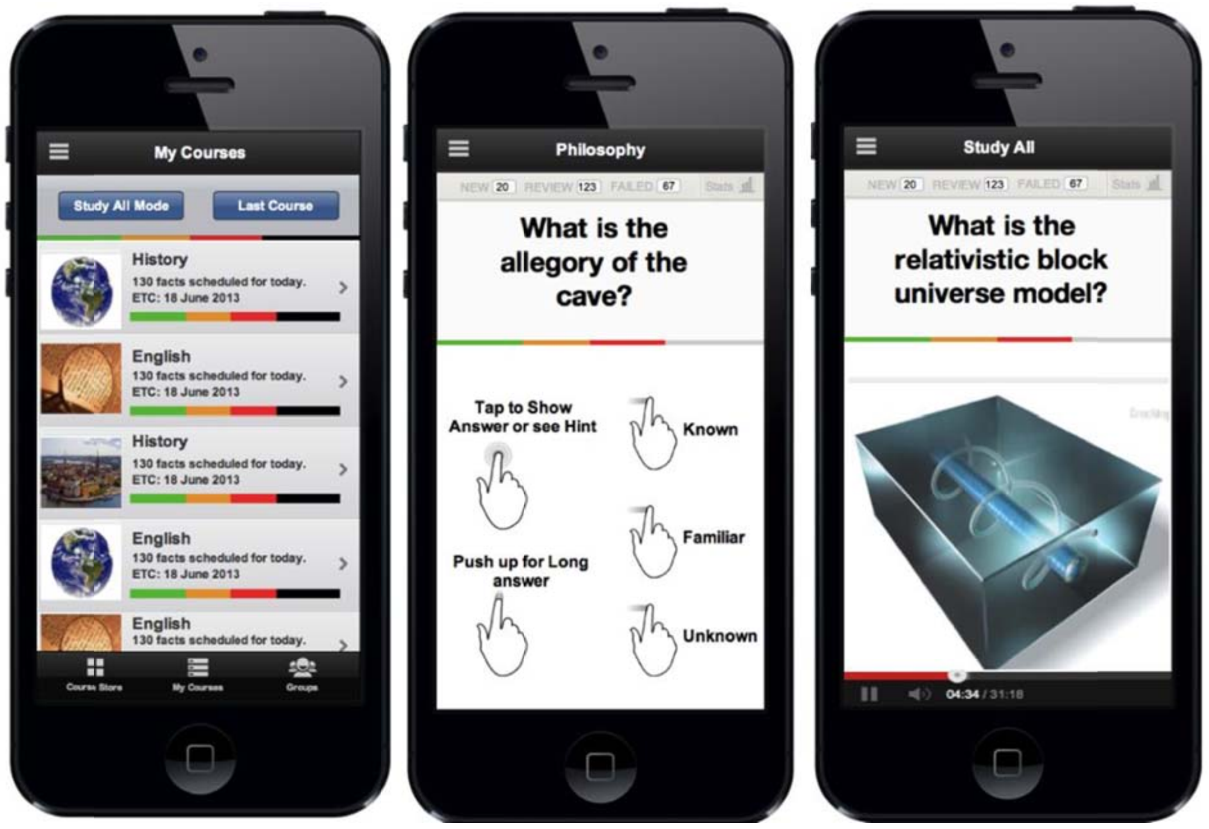
2. A lot of plugins (such as the drag bar from the Bootstrap themes) have not been tested on mobile devices. Unfortunately, just because the mouse works in the desktop development environment does not mean that fingers will work on a mobile device. More investigation and testing will be needed to investigate the operation of the web app on a variety of different handsets.

However, as part of this project a *design* of an iPhone app has been done. Figure 4-17 shows the iPhone App design (left to right):

(a) **My courses page** - in this page the user can browser the courses edited by him or others on computer browsers. The courses editing feature seems to be a bad idea on a such small screen.

(b) **Studying page** - in this page the user can perform all sorts of operations he can do on a computer with gestures show in the figure.

(c) Video studying page - in this page the user can watch YouTube clips just as on a computer.


My Courses
Study All Mode
Last Course
History
130 facts scheduled for today.
ETC: 18 June 2013
English
130 facts scheduled for today.
ETC: 18 June 2013
History
130 facts scheduled for today.
ETC: 18 June 2013
English
130 facts scheduled for today.
ETC: 18 June 2013
English
130 facts scheduled for today.
Course Store
My Courses
Groups
Philosophy
NEW 20 REVIEW 123 FAILED 67 Stats
What is the allegory of the cave?
Tap to Show Answer or see Hint
Push up for Long answer
Known
Familiar
Unknown
Study All
NEW 20 REVIEW 123 FAILED 67 Stats
What is the relativistic block universe model?
04:34 / 31:18

(a) **My courses page** (b) **Studying page** (c) **Video studying page**

Figure 4-17: iPhone App design.

5 Conclusions and Future work

This chapter gives a conclusion and discusses future improvements. The chapter concludes with a discussion of some of the social, ethical, and economic aspects of this thesis project.

5.1 Conclusions

This thesis has given an overview of those aspects of the problem that have been studied and a proposed solution has been presented.

The thesis began with an overview of spaced repetition systems and brief reviews of existing services based on spaced repetition. Details of the relevant parts of the YouTube API were explained along with some bugs found when carrying out the implementation part of this thesis project.

Some of the relevant front-end technologies were explained, such as responsive web design, HTML5, CSS3, Javascript, and jQuery. These technologies were used to create a user interface for the prototype.

The prototype was evaluated during the development process and feedback of volunteers was used to improve the functionality of the prototype. New volunteers without involvement in the previous testing tested the prototype features in a limited integration environment (i.e., not completely integrated into the live Sharplet site). Feedback from these volunteers includes:

- The feature is interesting and new, and the feature is preferable to the normal textual flashcards.
- The video editor page is not user friendly; sometimes users did not know what to do next and need to ask for instructions. Textual instructions are needed to provide a more clear and friendly experience.
- The advertisements on the top of video are very annoying and it should be possible to remove them.
- The waiting time for switching flashcards is too long.
- The search function is not perfect – especially due to the limitation of 10 search results. The result is that sometimes users needed to go to YouTube to search for the right video to learn the correct keyword to search for in the prototype.

The volunteers took on average 160 second per question to edit each answer clip. The answer clips averaged 16 seconds in duration, with a minimum of 5 seconds and a maximum of 30 seconds. (Detailed statistical analysis is shown in Appendix C.)

The prototype developed and evaluated as part of this master's thesis project has been completed and will be available online at sharplet.com. This new video clip editing and study mode seems to make studying more fun (based on the feedback described above) and bring greater efficiency to spaced repetition based online learning world.

5.2 Future work

Development of mobile apps and web apps for smart phones and tablets is common place nowadays. Application of these techniques will not only increase the size of the user base, but also will help the space repetition learning method achieve its full potential by utilizing the convenience of mobile devices and exploiting the fact that people have these devices with them nearly all of the time. In the future, we expect that a user will be able to study using his or her mobile devices on the go with learning materials edited via a web app on his or her computer. Thus perhaps rather than playing games to waste time when waiting for the bus, users will take advantage of this time to learn a new language, increase their vocabulary, improve their memory, etc.

As described in section 4.8.2, one of the most important future activities is to improve the page loading times as the page loading times are currently too long to be acceptable to users. This is especially true because we would like users to make use of the system to study when they have time available (which may occur sporadically throughout the user's day). Until there is a native app, there is a need for future research to solve either the problem with the YouTube API, to solve the problem of this preloader solution, or the parallel processing to do preloading and caching could be built into the web browser.

In this thesis project, because of the effort put into a responsive design, most of mobile devices can display the web pages for both video clip editing and studying with video clips. However, because of different input methods (mouse, touch screen, or finger) further user studies are needed. Additionally, because the ideas of web app UX design and native mobile app UX design are fundamentally different and presently the web based app cannot be used on Apple's mobile devices, there is a need to develop a native mobile app - both for IOS and Android (as these represent the two major portions of the smartphone market).

Eventually using Bootstrap for the front-end should be abandoned. Although Bootstrap provided easy way for a developer with limit design training to make an adequate design, the result is that all of the web app developed using Bootstrap look familiar. This is expected to confuse users and not make the website unique enough to be remembered (and frequently revisited). A professional web designer rather than a front-end developer will be needed for the future of the UI.

The prototype should be tested in a production environment with a limited number of online users. However, this will only be realistic when the proposed solution is fully integrated into the application and when its performance is high enough to not detract from the user's experience.

5.3 Required Reflections

The result of this master thesis project is a working prototype of an online learning system based on a spaced repetition system with YouTube integration. Instead of being limited to using a learning method that one can use with a pen and paper, the introduction of multimedia functionality gives the learner a means to expand upon the old way of spaced repetition studying and exploits the new potential of the increasingly ubiquitous smartphone technology. The use by learners of the production version of the online learning system based on a spaced repetition system with YouTube integration is expected to have beneficial social effects as learners will be able to make better use of their time, while expanding their skills. These expanded skills will also have social and economic effects, particularly in a multilingual society.

This expansion of spaced repetition with multimedia will inspire a lot of developers and investors to address this underdeveloped market for online spaced repetition services. More and more people will bring more great ideas to this area, and eventually learners will benefit from these efforts. These investments and the resulting systems are expected to have positive social and economic effects.

It is expect that the use of multimedia will attract learners to use their computer, tablet, or smart phone to study rather than pen and paper and that this can reduce the impact of spaced repetition system on the environmental by eliminating the physical media of flashcards and by making greater use of the users' existing electronic devices.

The volunteers who involved have been contributed a lot for this thesis project. Although no formal informed consent was given, they were fully aware that their answers and suggestions might be used to provide a better product. They were assured of the privacy of their responses as these responses would only be known to the project developers and that no personally identifying information would be retained regarding their participation or responses.

Bibliography

[1] ‘Sharplet’. [Online]. Available: https://sharplet.com/. [Accessed: 10-Sep-2013]

[2] ‘Spaced repetition’. sharplet.com, 30-Apr-2013 [Online]. Available: http://www.gwern.net/Spaced repetition. [Accessed: 07-May-2013]

[3] J. H. Reynolds and R. Glaser, ‘Effects of repetition and spaced review upon retention of a complex learning task.’, *J. Educ. Psychol.*, vol. 55, no. 5, pp. 297–308, 1964. DOI:10.1037/h0040734

[4] D. L. Schacter, S. A. Rich, and M. S. Stampp, ‘Remediation of memory disorders: Experimental evaluation of the Spaced-Retrieval technique’, *J. Clin. Exp. Neuropsychol.*, vol. 7, no. 1, pp. 79–96, Feb. 1985. DOI:10.1080/01688638508401243

[5] H. P. Bahrick and L. K. Hall, ‘The importance of retrieval failures to long-term retention: A metacognitive explanation of the spacing effect’, *J. Mem. Lang.*, vol. 52, no. 4, pp. 566–577, May 2005. DOI:10.1016/j.jml.2005.01.012

[6] Mark Otto, Jacob Thornton, Chris Rebert, Julian Thilo, and et al., ‘Twitter Bootstrap’. [Online]. Available: http://getbootstrap.com/getting-started/. [Accessed: 29-Aug-2013]

[7] Jakob Nielsen and Don Norman, ‘User experience design’. Nielsen Norman Group, 02-May-2013 [Online]. Available: http://www.nngroup.com/articles/definition-user-experience/. [Accessed: 07-May-2013]

[8] 37signals, *Getting Real*. 2006 [Online]. Available: https://gettingreal.37signals.com/. [Accessed: 22-Feb-2014]

[9] Hermann Ebbinghaus, *Memory: A Contribution to Experimental Psychology*. Columbia University, 1885 [Online]. Available: http://psy.ed.asu.edu/~classics/Ebbinghaus/index.htm

[10] H. F. Spitzer, ‘Studies in retention’, *J. Educ. Psychol.*, vol. 30, no. 9, pp. 641–656, Dec. 1939. DOI:10.1037/h0063404

[11] C. A. Mace, *Psychology of Study*. Pelican Books, 1932 [Online]. Available: http://archive.org/stream/psychologyofstud00mace/psychologyofstud00mace_djvu.txt

[12] C. A. Mace, *Psychology of study.*, New impression edition. Penguin, 1968.

[13] S. Leitner, *So lernt man lernen. Angewandte Lernpsychologie – ein Weg zum Erfolg*. Freiburg im Breisgau, Germany: Verlag Herder, 1972.

[14] supermemo.com, ‘What is SuperMemo?’ 23-Nov-2013 [Online]. Available: http://www.supermemo.com/english/smintro.htm. [Accessed: 02-Feb-2014]

[15] P. Woźniak, ‘Optimization of learning: A new approach and computer application’, Master’s thesis, University of Technology in Poznan, Computer Science Center, Poznań, Poland, 1990 [Online]. Available: http://www.supermemo.com/english/ol.htm. [Accessed: 15-Feb-2014]

[16] P. Woźniak, ‘SuperMemo 2: Algorithm’, *3.2. Application of a computer to improve the results obtained in working with the SuperMemo method*, 10-May-1998. [Online]. Available: http://www.supermemo.com/english/ol/sm2.htm. [Accessed: 15-Feb-2014]

[17] Damien Elmes, ‘Anki - powerful, intelligent flashcards’. 04-Nov-2013 [Online]. Available: http://ankisrs.net/

[18] Cram, *Cram*. 2014 [Online]. Available: http://www.cram.com/. [Accessed: 10-Feb-2014]

[19] Cram Team, ‘About Cram.com’, 2014. [Online]. Available: http://www.cram.com/about. [Accessed: 11-Feb-2014]

[20] Quizlet, *Quizlet*. 2014 [Online]. Available: http://quizlet.com/. [Accessed: 10-Feb-2014]

[21] Google, ‘YouTube API overview’. [Online]. Available: https://developers.google.com/youtube/getting_started. [Accessed: 10-Sep-2013]

[22] E. Hammer-Lahav, ‘Introduction — OAuthn’, *OAuth*, 2007. [Online]. Available: http://oauth.net/about/. [Accessed: 15-Feb-2014]

[23] E. Hammer, ‘Explaining OAuth’, 05-Sep-2007. [Online]. Available: http://hueniverse.com/2007/09/explaining-oauth/. [Accessed: 15-Feb-2014]

[24] ‘The OAuth 1.0 Guide « hueniverse’, 15-Jul-2011. [Online]. Available: http://hueniverse.com/oauth/guide/. [Accessed: 15-Feb-2014]

[25] E. Hammer-Lahav, ‘The OAuth 1.0 Protocol’, *Internet Req. Comments*, vol. RFC 5849 (Informational), Apr. 2010 [Online]. Available: http://www.rfc-editor.org/rfc/rfc5849.txt

[26] D. Hardt, ‘The OAuth 2.0 Authorization Framework’, *Internet Req. Comments*, vol. RFC 6749 (Proposed Standard), Oct. 2012 [Online]. Available: http://www.rfc-editor.org/rfc/rfc6749.txt

[27] MDN developer team, ‘DOM Storage guide’. Mozilla Developer Network, 13-Nov-2013 [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Storage. [Accessed: 24-Dec-2013]

[28] Ian Hickson, ‘Web Applications 1.0’. Opera Software, 01-Sep-2005 [Online]. Available: http://www.whatwg.org/specs/web-apps/2005-09-01/. [Accessed: 24-Dec-2013]

[29] ‘Web Storage’. W3C, 30-Jul-2013 [Online]. Available: http://www.w3.org/TR/webstorage/. [Accessed: 24-Dec-2013]

[30] MDN developer team, ‘HTTP cookies’. Mozilla Developer Network, 26-Jan-2012 [Online]. Available: https://developer.mozilla.org/en-US/docs/Web_Development/HTTP_cookies. [Accessed: 24-Dec-2013]

[31] M. Otto, ‘Bootstrap from Twitter’. [Online]. Available: https://dev.twitter.com/blog/bootstrap-twitter. [Accessed: 24-Sep-2013]

[32] Alexis Sellier and LESS core team, ‘LESS - The Dynamic Stylesheet language’. [Online]. Available: http://lesscss.org/. [Accessed: 08-Oct-2013]

[33] E. Marcotte, ‘Responsive Web Design · An A List Apart Article’, 25-May-2010. [Online]. Available: http://alistapart.com/article/responsive-web-design. [Accessed: 15-Feb-2014]

[34] E. Marcotte, *Responsive Web Design*, Paperback. A Book Apart, 2011.

[35] O. Reichenstein, ‘Responsive Typography: The Basics’. 01-Jun-2012 [Online]. Available: http://ia.net/blog/responsive-typography-the-basics/. [Accessed: 08-May-2013]

[36] J. Pepitone, ‘Worst PC sales drop in history’, Apr. 2013 [Online]. Available: http://money.cnn.com/2013/04/10/technology/pc-sales/index.html. [Accessed: 08-May-2013]

[37] V. Salminen, ‘Adaptive vs. Responsive, what’s the difference?’ 23-Feb-2012 [Online]. Available: http://viljamis.com/blog/2012/adaptive-vs-responsive-whats-the-difference.php. [Accessed: 08-May-2013]

[38] ‘CSS media queries’. Mozilla Developer Network, 26-Aug-2013 [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries?redirectlocale=en-US&redirectslug=CSS/Media_queries. [Accessed: 29-Aug-2013]

[39] Mozilla developer, ‘<iframe>’. 10-Feb-2014 [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe. [Accessed: 22-Feb-2014]

[40] ‘W3Counter - Global Web Stats’, *Awio Web Serv. LLC*, Oct. 2013 [Online]. Available: http://www.w3counter.com/globalstats.php?year=2013&month=10. [Accessed: 14-Nov-2013]

[41] Md. Iqbal Hossain and Md. Iqbal Hossain, ‘Dynamic scaling of a web-based application in a Cloud Architecture’, Master’s thesis, KTH Royal Institute of Technology, School of Information and Communication Technology, Stockholm, Sweden, TRITA-ICT-EX-2014:13, February 2014.

[42] MDN developer team, ‘Working with objects’. 2014 [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects. [Accessed: 11-Feb-2014]

[43] MDN developer team, ‘JSON’. 2014 [Online]. Available: https://developer.mozilla.org/en/docs/JSON. [Accessed: 11-Feb-2014]

[44] MDN developer team, ‘JSON.stringify()’. 2014 [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify. [Accessed: 11-Feb-2014]

[45] MDN developer team, ‘JSON.parse()’. 2014 [Online]. Available: https://developer.mozilla.org/en-

US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse. [Accessed: 11-Feb-2014]

[46] Adobe Systems Incorporated, ‘Control access to scripts’. Adobe Systems Incorporated, 2014 [Online]. Available: http://helpx.adobe.com/flash/kb/control-access-scripts-host-web.html. [Accessed: 11-Feb-2014]

[47] Aran Rhee, ‘SWFObject’. 18-Jun-2013 [Online]. Available: http://code.google.com/p/swfobject/wiki/documentation. [Accessed: 11-Feb-2014]

[48] Google, ‘gapi.client.youtube.search.list’. 05-Dec-2013 [Online]. Available: https://developers.google.com/youtube/v3/docs/search/list. [Accessed: 11-Feb-2014]

[49] Google, ‘YouTube JavaScript Player API Reference’. 04-Nov-2013 [Online]. Available: https://developers.google.com/youtube/js_api_reference

[50] Browserscope, ‘Browserscope’. [Online]. Available: http://www.browserscope.org/?category=network. [Accessed: 05-Mar-2014]

[51] Canonical Ltd, *The world’s most popular free OS | Ubuntu*. 2014 [Online]. Available: http://www.ubuntu.com/. [Accessed: 11-Feb-2014]

[52] Oracle, *Java SE Development Kit 7 Downloads*. [Online]. Available: http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html. [Accessed: 11-Feb-2014]

[53] Oracle, *Oracle GlassFish Server*. [Online]. Available: http://www.oracle.com/us/products/middleware/cloud-app-foundation/glassfish-server/overview/index.html. [Accessed: 11-Feb-2014]

[54] The PostgreSQL Global Development Group, *PostgreSQL*. 2014 [Online]. Available: http://www.postgresql.org/. [Accessed: 11-Feb-2014]

[55] *Eclipse*. The Eclipse Foundation [Online]. Available: https://www.eclipse.org/. [Accessed: 11-Feb-2014]

[56] pgAdmin, *pgAdmin PostgreSQL Tools*. [Online]. Available: http://www.pgadmin.org/. [Accessed: 11-Feb-2014]

[57] Ritesh M Nayak, ‘How do I remove javascript validation from my eclipse project?’ 28-Jun-2010 [Online]. Available: http://stackoverflow.com/questions/3131878/how-do-i-remove-javascript-validation-from-my-eclipse-project. [Accessed: 11-Feb-2014]

[58] Steve Jobs, ‘Thoughts on Flash’, Apr-2010. [Online]. Available: http://www.apple.com/hotnews/thoughts-on-flash/. [Accessed: 11-Feb-2014]

Appendix A Source code for the standalone feature without integration

The code for the standalone feature is available in GitHub. Links to the repositories are provided in Appendix Table A-1. The standalone feature without Sharplet styling can be tested at http://youbei.github.io/sharplet/video.html. The integrated feature will be available at sharplet.com in the future when the optimization is completed.

Appendix Table A-1: Links to source code

Studying page	https://github.com/youbei/youbei.github.io/blob/master/sharplet/learning.html
Studying page with preloader	https://github.com/youbei/youbei.github.io/blob/master/sharplet/learning_preloader.html
Video editor page	https://github.com/youbei/youbei.github.io/blob/master/sharplet/video.html

Appendix B SQL query to create the initial database and add tables for video feature

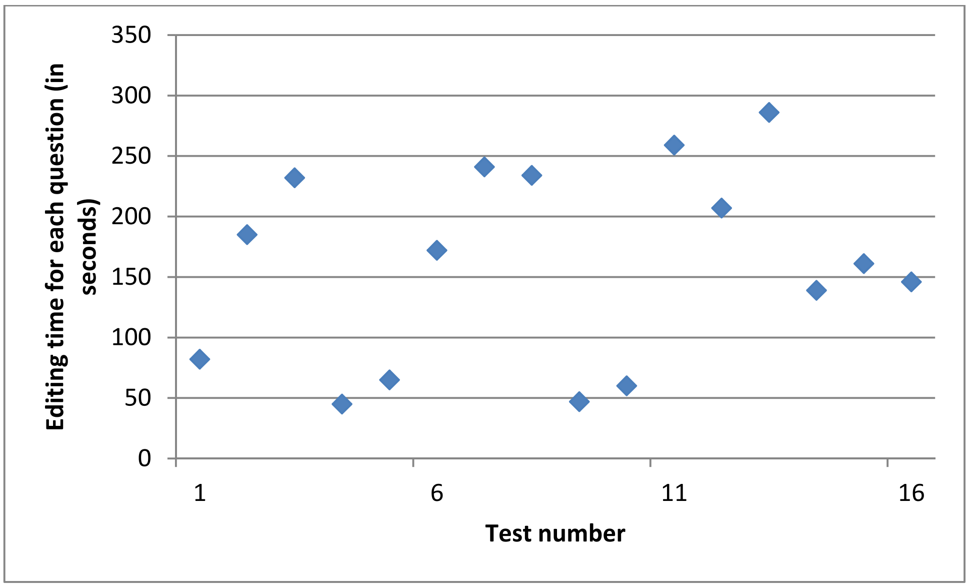
Appendix Table B-1: Github links to the SQL queries

SQL query to create the initial database	https://github.com/youbei/youbei.github.io/blob/master/sharplet/postgres_full_latest.sql
SQL query to add the data for video feature	https://github.com/youbei/youbei.github.io/blob/master/sharplet/video.sql

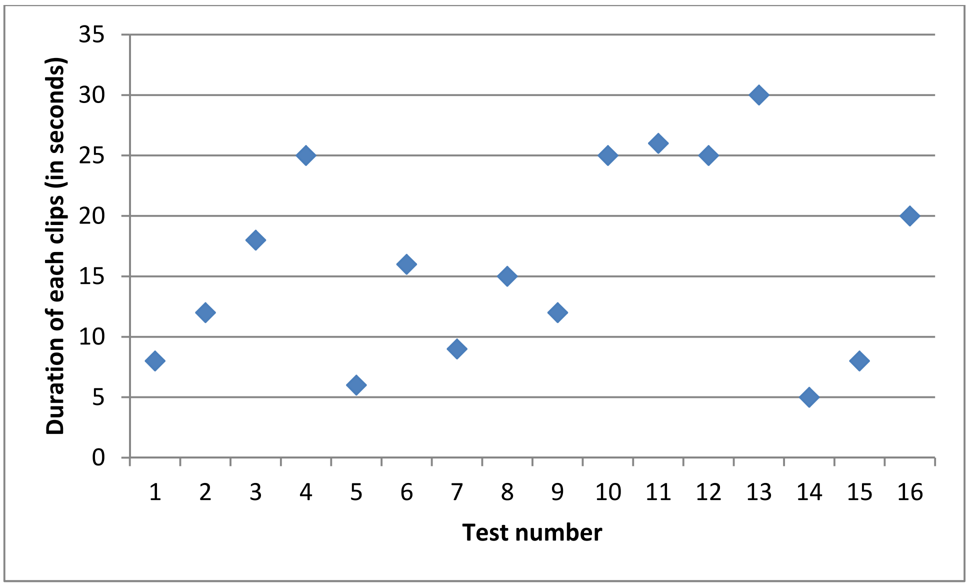
Appendix C Statistical analysis of editing times and clip duration

Editing time for each question (in seconds)	**Duration of each clips (in seconds)**	
82	8	
185	12	
232	18	
45	25	
65	6	
172	16	
241	9	
234	15	
47	12	
60	25	
259	26	
207	25	
286	30	
139	5	
161	8	
146	20	
45	**5**	**Min**
286	**30**	**Max**
166.5	**15.5**	**Median**
78.11	**7.86**	**Std**
160.06	**16.25**	**Average**

Appendix Figure C-1 and Appendix Figure C-2 plot the data from the above columns.

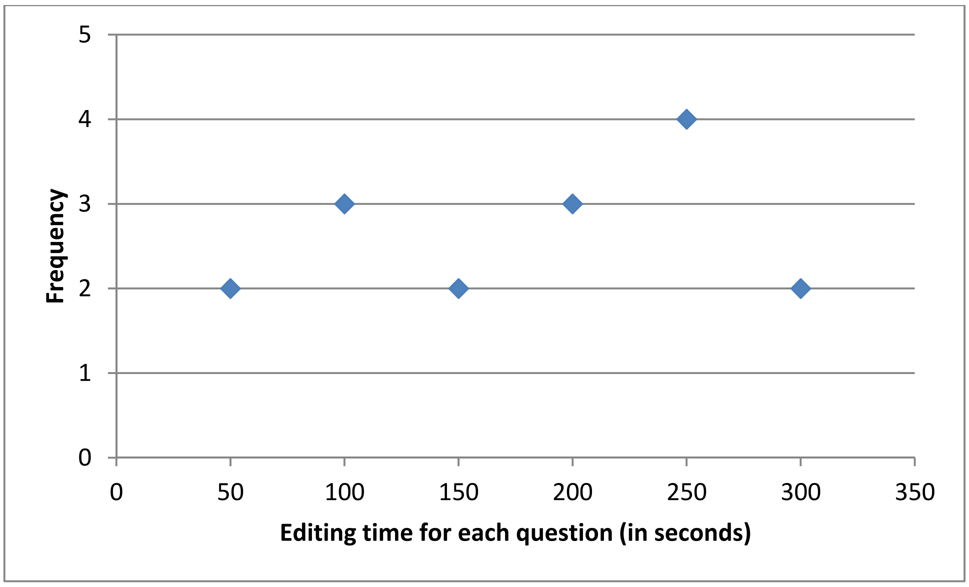


Appendix Figure C-1: Editing time for each question (in seconds)



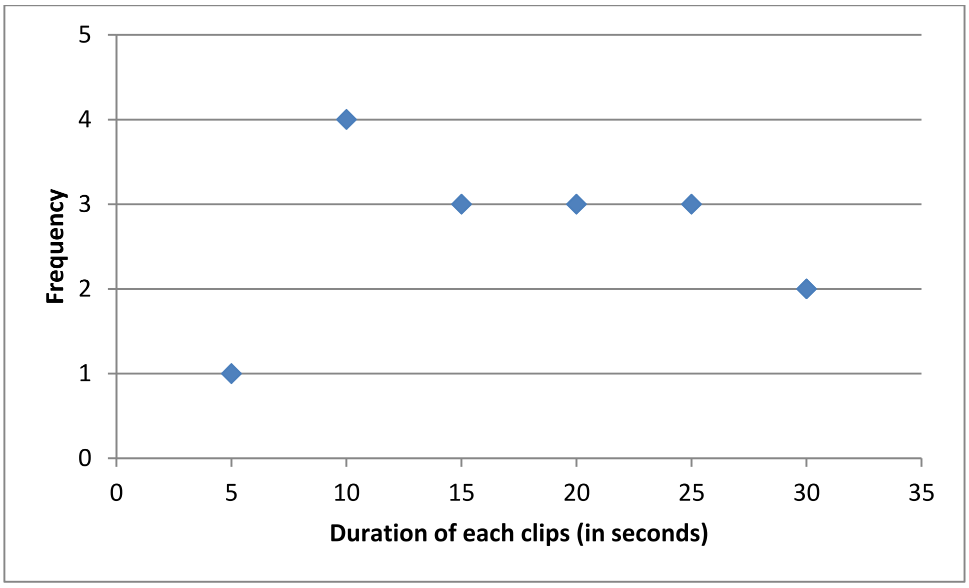
Appendix Figure C-2: Duration of clips (in seconds)

Appendix Figure C-3 shows a histogram of editing times. While there are only a small number of sample values, we can see that the distribution of editing times is rather flat.



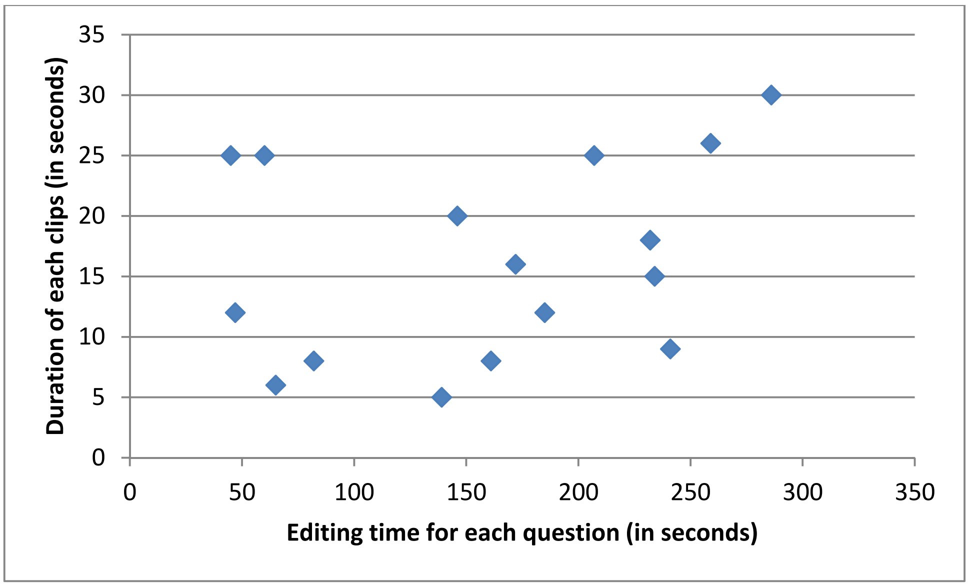
Appendix Figure C-3: Histogram of editing time (in seconds)

Appendix Figure C-4 shows the histogram of the clip durations. From this figure we can see that most clips are between 10 and 30 seconds long. This matches our intuitive notion that the answers tend to be short video clips, rather than long video clips.



Appendix Figure C-4: Histogram of Clip duration (in seconds)

We can see from Appendix Figure C-5 that there is no direct relationship between the time spent editing a clip and the duration of the clip. This contrasts with what might intuitively think that shorter clips would take longer to edit or that longer clips would take longer to edit. In fact computing a trend-line between editing time (x) and duration (y) is $y = 0.0291x + 11.597$ with $R^2 = 0.0834$. This very small R^2 shows that there is not a direction relationship between these two.



Appendix Figure C-5: Editing time for each question versus duration of the resulting clip

TRITA-ICT-EX-2014:17