Final Project Report

Overview

As one of the premier education institutions in the world, Yale offers a number of lecture courses on its online platform, Open Yale Courses. While its target audience is the general public, many Yale students taking those offered classes also benefit from the materials available by using them to catch up on missed lectures, reading transcripts for review, and “time-shifting” lectures.

However, only 42 of over 2,000 undergraduate courses at Yale are available on Open Yale Courses, meaning that the number of students who can benefit from the resources is very limited. From the institution’s point of view, it is against its original purpose to expand its current platform since Open Yale Courses is not intended for its own students in the first place. Additionally, According to an article on the New York Times, Yale has spent $30,000 to $40,000 for each course it puts online, making it highly uneconomical and impractical to expand the platform in a large scale.

In light of the above problem, our group came up with the idea of CourseShare, a crowdsourced and context-aware platform that allow Yale students to record, upload and share lectures in the web. While students benefit from the platform through reviewing missed lectures uploaded by peers, they reciprocally contribute to the database by recording and uploading.

Product Description

CourseShare is a mobile/web application developed on the Android platform that targets Yale students in all offered classes (especially lecture courses). The mobile application allows students to record the audio of the lecture, take snapshots of the lecture slides, and upload to the server the package containing the audio file and image file(s) for each session. To ensure that recordings stay in the Yale community, students must login to the Yale network through our CAS-integrated interface before uploading. The web application fetches the package of each session from the server and synchronizes the photos and the audio using timestamps. Students
can then view the lecture “montage” for catch-up and review purposes. CourseShare is the first course-sharing platform using the crowdsourcing concept, and its “montage” creation synchronizing audio with images is another notable novelty point in the product.

**Application workflow**

![Application workflow diagram]

**Design Points**

**Pictures + Audio vs. Video**

The group chose to send a package of audio and image files instead of video due to use-case and data transfer considerations. In terms of use cases, recording a video would require the student to have his phone on video mode and hold the position for a full 75 minutes. Not only would it drain an immense amount of energy from the phone battery, this is also a great inconvenience and a major disincentive to the usage of the application. Moreover, video packages would be much larger than packages with only audio and images, and thus would greatly lengthen the upload time and increase energy consumption. Table 1 shows figures about the size of a package for a 75-minute lecture.

<table>
<thead>
<tr>
<th>Video</th>
<th>Audio + Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit size</td>
<td>Size for 75-min lecture</td>
</tr>
<tr>
<td>500 kbits/s</td>
<td>275MB</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td><strong>275MB</strong></td>
</tr>
</tbody>
</table>

*Assume 100 slides/lecture
Therefore, by opting to use a “montage” package, we provide better user experience and minimize energy consumption (see below).

Energy Consumption

Another major concern of ours, and of mobile application development in general, is power consumption. As users run our application on mobile devices, we would like to ensure that the application does not consume a major portion of the energy available. A few features of our application will help us achieve the goal. First, following the design point, we opted to upload much smaller packages of images and audio recordings. Furthermore, our app, while running a background service to record the audio of the whole lecture, allows the user to turn off or lock their screens without destroying the activity and service, ensuring continuity in the recording and minimization of energy consumption.

Technical Challenges

Media Recorder and Camera API Adaptation

One surprising technical difficulty experienced during the application development was the management of the activity lifecycle. Since our application required both the constant usage of a camera and a background recording service while the screen was off, the activity lifecycle had to extremely well regulated and consistent. We discovered that when the user locks the screen, Android automatically destroys and recreates the activity on start-up since the lock screen acts as a “change in orientation”. While this has its own benefits for activities to adapt to new screens, it was extremely detrimental for our use case (over the course of a single recording, they would be locking the camera multiple times). The constant destruction of our activity would make keeping consistent data in the SQLite database difficult. In order to address this issue, we designed the application to handle orientation and screen changes on its own, rather than relying on the built-in Android lifecycle handler, so that the recording and camera services can remain consistent. In this manner, we were able to manage the activity lifecycle so that the activity only pauses and the service keeps on running while the screen was locked.
HTML5 playback

HTML 5 playback is restricting in many respects:

- Browser support is non-standard and non-universal. As a result, currently the application may only be run on more recent Firefox and Chrome versions.
- Browsers can only play certain file formats. Especially considering that the phones usually encode using AMR and 3GPP, the server needed to transcode the videos.
- Existing solutions were suitable for playlists, but not particularly suitable for sequential, gapless playback of multiple audio files. We had to write our own Javascript to ensure that pre-loading happened.

Authentication services

In order to make sure that only Yale students could access the data, we used Yale’s Central Authentication Service (CAS). However, CAS was made mostly with an eye towards web application-based authentication, and it is not possible to authenticate with the CAS server without a third server. Fortunately, we needed another server to upload pictures and audio to anyways, but getting authentication to work involved writing a custom client based off of DefaultHttpClient that interacted with the CourseShare server, which then contacted the CAS servers. The device does not make a direct connection to the CAS servers.

Future Enhancements

In the section below, we highlight some main features that can further improve the application’s functionality, user experience and marketability.

Optimization using image and audio enhancement technologies

On Open Yale Courses, there is only one camera angle and a single audio input for the lectures. The major problems with this are that the viewers may not be able to hear a question raised by a student far away from the camera, and the fixed recording angle does not guarantee a clear view of the lecture slides or the blackboard. Since CourseShare is crowdsourced, there will be multiple recordings and snapshots from different angles. When these packages are uploaded using the same course label, on the server end we can merge the recordings together to reduce
background noise, and pick the snapshot of the highest quality for each time frame. This feature will fully utilize the crowdsourcing idea and greatly enhance the quality of the “montage”.

Location awareness

The current version of CourseShare prompts the user to enter the course label (e.g. CPSC 434, ECON 115) on upload, which may cause some inconvenience because not every student memorizes all the course numbers. To further enhance the user experience, we are looking to construct a database, from Yale’s Online Course Information, of all courses with their respective course numbers, location, and meeting time. Since Wi-Fi is available in all classrooms in most universities, it is possible to make use of Wi-Fi location services to determine the user’s location while recording, and then query the database for the course number together with the timestamp of the package. This will allow the addition of an auto-suggest feature for the course label in the upload stage.

However, the initial construction of the database may be very costly given the large number of courses offered. Another way to construct the database is through initial crowdsourcing, meaning that when a user uploads a package, metadata containing the location and time is also added to the database. As a result, each course label will be associated with multiple location-time pairs within a certain radius from a center and time frame. This implementation will enhance the application’s portability across schools because it will not need authentication to obtain the information of courses.

Expansion to other institutions in the US and in the world

Currently, CourseShare is only available to Yale students with its CAS Login activity. For the application to be used in other schools, since recording is done locally on the phone and upload can be supported by school-independent servers, we will only need to adapt to the authentication system for login and playback. If CourseShare turns out successful at Yale, we believe that the project is portable enough to expand its service to other schools as well. It will be incredibly exciting to see CourseShare benefiting all college students in the US and even in the world.

Conclusion
CourseShare at this stage is by no means a completed product. Given the time constraints, our group was able to create a prototype that supports the three fundamental functionalities: record, upload, and playback. We strongly believe that CourseShare has a high potential for future development and public release. First, the idea of crowdsourcing academic resources is unprecedented. Second, from the perspective of software development, the project is technically challenging yet certainly achievable. Third, since there are 150 million college students globally, CourseShare is highly marketable once mature. We are excited to continue CourseShare’s development and realize its potential in the future.