Extending a Learning Platform with Cross-Device Functionality

Maria Husmann
Department of Computer Science
ETH Zurich
husmann@inf.ethz.ch

Nicola Marcacci Rossi
Department of Computer Science
ETH Zurich
nicolamr@student.ethz.ch

Moira C. Norrie
Department of Computer Science
ETH Zurich
norrie@inf.ethz.ch

Abstract
We report on our experience of adding cross-device functionality to a learning platform with a substantial user base. The extension allows students to display an exercise sheet on one device (typically a notebook) and use a paired handheld device to submit photographed solutions to the exercise. We outline the design and implementation of the feature and discuss lessons learned in the process of working outside of a controlled lab environment.

Author Keywords
cross-device; in-the-wild; education.

ACM Classification Keywords
H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous

Introduction
Today's device ecologies offer exciting opportunities for interactions between multiple devices [3]. Tools and frameworks facilitate the design [4, 2] and implementation [5, 6, 1] of cross-device applications. Nevertheless, we have encountered few such applications in the wild. Two notable exceptions have been extensions to Google Maps[1] and YouTube[2]. Both of these are small extensions to appli-

[1] https://support.google.com/maps/answer/6081481
[2] https://support.google.com/chromecast/answer/2995235

cations with a huge user base. Building up a user base is challenging for any application. Having access to an existing base is thus an interesting opportunity to explore cross-device designs in the wild, but comes with the challenge of having to integrate with an existing system.

We had the opportunity to extend a web-based education platform called Taskbase[3] with cross-device functionality. Taskbase manages collections of theory and exercise materials and allows exercises to be grouped into exercise sheets and published to a class of students. Students can give feedback on the exercises and rate their difficulty. Taskbase is currently in use at several schools and universities in Switzerland, including ETH Zurich with over 3500 users in the mathematics department and the University of St. Gallen with over a 1000 users. ETH Zurich has its own installation[4] and this was used in the work described here. The platform was developed by the startup company edTechLab in cooperation with the mathematics department of ETH. The authors of this paper were not involved with development prior to the cross-device project that we describe in this paper. However, one of the authors is acquainted with the CEO of edTechLab and the company expressed interest in exploring cross-device functionality in Taskbase.

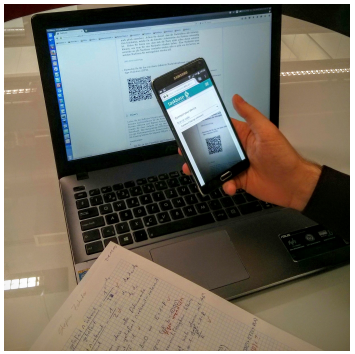

Figure 1: The student scans a QR code on their notebook.

Cross-Device Extension

We started by analysing how Taskbase is used and where there is potential added value with a cross-device extension. We then designed a cross-device feature and, after consulting with edTechLab, implemented it in Taskbase.

[3] http://www.edtechlab.ch/taskbase
[4] https://e-lectures.ethz.ch/

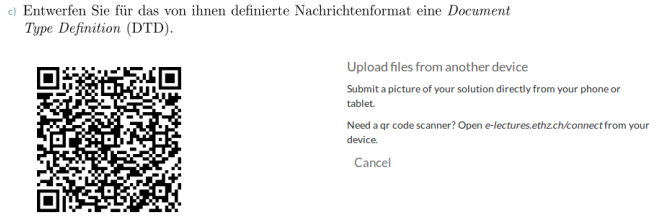

Figure 2: For each exercise, a QR code is displayed that takes the student to the exercise submission site when scanned with a phone.

Analysis

While the exercise sheets can be printed, instructors observed that students often use their devices (notebooks or tablets) to access them during exercise sessions. This was confirmed when we analysed access patterns to the system. We also noted that many students used more than one device to access the platform. Within one week, we observed roughly 2700 users accessing the platform with two different devices, while fewer than 400 users only used a single device. Fewer than 100 users accessed Taskbase with three or more devices.

In our own teaching (independent of Taskbase), we are increasingly experiencing students submitting either scanned or photographed versions of their handwritten assignments via email. The teaching assistant then prints the submissions, marks them on paper, and returns them in the next exercise session.

Design

Based on our analysis, we decided to design a feature that allows students to submit their handwritten solutions to exercises using the camera of their phones or tablets. An exercise sheet may consist of multiple exercises and a stu-

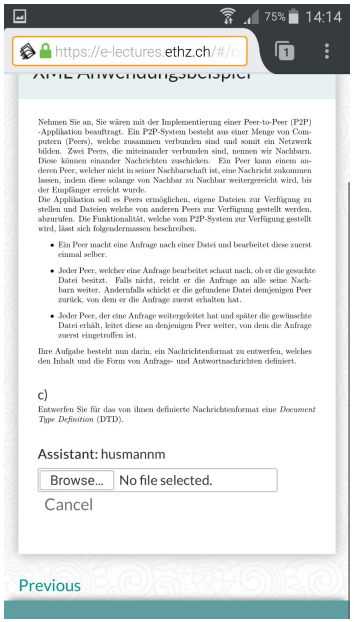

Figure 3: After scanning the QR code, the student is taken to the corresponding exercise on their phone.

dent can submit a solution for each exercise. The interface offers options to directly upload a file from the current device (typically a notebook or a desktop computer) or to use another device for uploading. If the latter option is chosen, the system displays a QR code (Fig. 2). When the code is scanned with a phone or tablet (Fig. 1), it opens a URL that points to the same exercise as that currently opened on the first device (Fig. 3) and the student is automatically logged in. Now they can simply photograph their solution (Fig. 5) and it will be associated with the exercise and uploaded to Taskbase. The responsible teaching assistant can access all submissions from their students and provide feedback as annotations on the pictures (Fig. 4) which can then be viewed by the student.

Implementation

As Taskbase is already in use, it was a clear goal not to disrupt users in any way and to keep the implementation as lightweight as possible, while integrating with the current architecture based on Java on the backend and AngularJS on the frontend. As current cross-device frameworks are still experimental, it was deemed too large a risk to integrate one of them and the changes were kept to a bare minimum. Consequently, no direct communication between the devices was implemented. Consequently, when a students scans their submission with their mobile phone, they need to manually refresh the notebook to see their submission. While we would have preferred synchronisation between the two devices, we had to compromise for the sake of complying with the current architecture which does not support any push messages from the server.

Deployment

As the electronic hand-in is only useful if assistants are willing to accept submissions made through the system, the feature was disabled by default and had to be enabled per

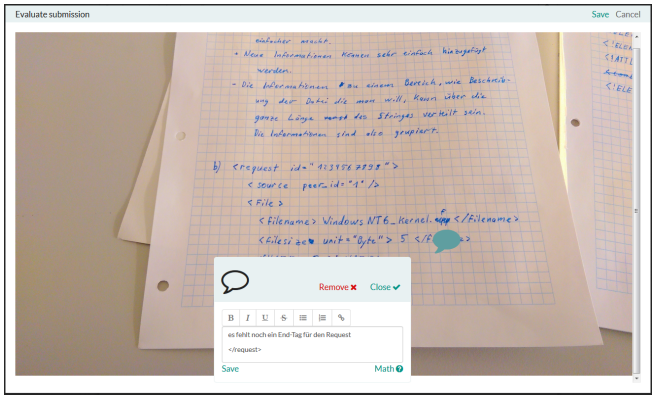

Figure 4: The UI for evaluating submitted exercises. Teachers or assistants can add comments and give feedback.

course. Taskbase then sent an email to professors informing them of the change and asking for volunteers to opt in. Unfortunately, there were few responses and the ones that we got were negative. The feature was rolled out in the second half of the semester and professors were reluctant to change the process half-way in. One professor worried that the feature would decrease personal interaction between assistants and students and feared dehumanisation of the process. We therefore decided to do a pilot test of the feature in one of the author's classes which had not been using Taskbase previously. The platform was introduced and the feature was demonstrated in class. Students were encouraged to use it, but were still allowed to hand in their solutions in person or by email. In total 44 students were enrolled in that class and roughly 30 students typically attended the exercise sessions. Handing in assignments was voluntary (a master solution was provided) and generally done by 6 to 10 students to get feedback. After the introduction of the cross-device feature, two students submitted

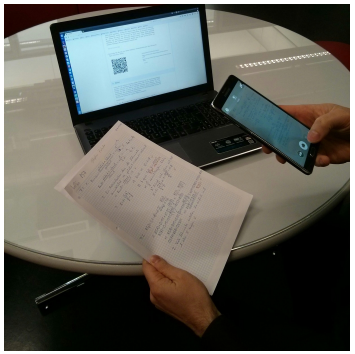

Figure 5: The student takes a picture of their solution and uploads it to the system.

their solution via their phone. 6 students used the new UI to upload a file from their notebook. No more assignments were submitted in person or by email. One student reported a problem with his installed QR code scanner that could not display the Taskbase website correctly. The rather low number of photographed submissions can be partly explained by the nature of the exercise which was better suited to being solved on a computer as well as the fact that it was the last exercise of the semester where the number of submissions of non-mandatory assignments typically drops.

Lessons Learned

While we had experience in developing cross-device applications, we typically work in a controlled environment where we can choose the architectures, technologies and devices and have full control over the applications that we build. Users are introduced to our applications in user studies or demonstrations where we, again, control many parameters. Working with Taskbase forced us to give up some of that control, confronted us with real users and stakeholders, and taught us the following lessons.

Focus on the user, not the devices. While our main interest was in integrating the phone, a significant amount of time was spent on the UI for giving feedback to the assignments. Had we skipped that part, students would have had no reason to use the system to hand in their solutions. It was important to develop a whole use-case, rather than just focusing on the cross-device part. All in all, a small fraction of time was spent on actual cross-device development.

Stakeholders are not necessarily excited about making something cross-device. Mainly students and teaching assistants benefited from our changes and the Taskbase team was very supportive. Professors however were more sceptical but important decision makers. Getting them on board will be critical for the success of the cross-device feature.

Cross-device frameworks may introduce a big change to the existing architecture. Cross-device frameworks typically introduce additional servers or services and protocols (for example WebSockets). Furthermore, companies may shy away from using frameworks that have not been proven to be production ready. In our case, the risk and effort was considered too high for a simple feature and, despite our experience with our own framework[5], we decided not to use it.

Don't make any assumptions about software and hardware. Provide alternatives to support a wide range of platforms and devices. We integrated a web-based QR scanner that could be accessed from a short URL for those users who had none installed. Our system required no installation on any device, thus maintaining a low barrier to entry. We did not force users to use their phones for submissions but rather supported file uploads from any device.

Carefully plan the introduction to the users. In our case, the timing of the introduction towards the end of the semester was not ideal. However, we were constrained by the timing of the student project and could not wait until the beginning of the next semester. Also, consider how users can learn to use cross-device features. The demonstration in class proved to be a good solution, however, it does not scale very well to a large user base.

Acknowledgements

We would like to thank edTechLab for giving us the opportunity to work with them and all their support. This project was supported by grant No. 150189 of the Swiss National Science Foundation (SNF).

[5] https://github.com/mhusm/XD-MVC

REFERENCES

1. Sriram Karthik Badam and Niklas Elmqvist. 2014. PolyChrome: A Cross-Device Framework for Collaborative Web Visualization. In *Proc. ITS*. DOI: http://dx.doi.org/10.1145/2669485.2669518
2. Steven Houben and Nicolai Marquardt. 2015. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1247–1256. http://doi.acm.org/10.1145/2702123.2702215
3. Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. The Proximity Toolkit: Prototyping Proxemic Interactions in Ubiquitous Computing Ecologies. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 315–326. http://doi.acm.org/10.1145/2047196.2047238
4. Michael Nebeling, Theano Mintsi, Maria Husmann, and Moira C. Norrie. 2014. Interactive Development of Cross-Device User Interfaces. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 2793–2802. http://doi.acm.org/10.1145/2556288.2556980
5. Mario Schreiner, Roman Rädle, Hans-Christian Jetter, and Harald Reiterer. 2015. Connichiwa: A Framework for Cross-Device Web Applications. In *Proc. CHI EA*. DOI:http://dx.doi.org/10.1145/2702613.2732909
6. Jishuo Yang and Daniel Wigdor. 2014. Panelrama: Enabling Easy Specification of Cross-Device Web Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 2783–2792. http://doi.acm.org/10.1145/2556288.2557199