

Git-Based Platform for Distributed Learning Communities

by

Vahid Fazel-Rezai

Submitted to the Department of Electrical Engineering and Computer
Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

© Vahid Fazel-Rezai, MMXVIII. All rights reserved.

The author hereby grants to MIT permission to reproduce and to
distribute publicly paper and electronic copies of this thesis document
in whole or in part in any medium now known or hereafter created.

Author
Department of Electrical Engineering and Computer Science
May 25, 2018

Certified by
Philipp Schmidt
Research Scientist, Director of Media Lab Learning Initiative
Thesis Supervisor

Accepted by
Katrina LaCurts
Chairman, Department Committee on Graduate Theses

Git-Based Platform for Distributed Learning Communities

by

Vahid Fazel-Rezai

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 2018, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Instructors use software called learning management systems to administer online courses. Traditionally these digital platforms mimic physical classrooms, but new technology opens up potential for interactions not previously possible. In this thesis, we build a learning management system on top of Git to create a learning environment that encourages collaboration and exploration. We discuss how it was designed following established learning principles, the technical implementation, and the results of a pilot online class, with 100 learners lasting 6 weeks, as part of the Refugee Learning Accelerator. The platform architecture used in this study can be replicated and reused for other online courses with similar goals.

Thesis Supervisor: Philipp Schmidt

Title: Research Scientist, Director of Media Lab Learning Initiative

Acknowledgments

To do...

Contents

1	Introduction	11
1.1	Refugee Learning Accelerator	12
1.2	Methodology	13
2	Background	15
2.1	Traditional Learning Platforms	15
2.2	Foundational Theory	16
2.2.1	Self-determination Theory	16
2.2.2	Learning Outcomes	16
2.2.3	The 4 P's	17
2.3	Design Tensions	18
3	Related Work	21
3.1	Overview of Git	21
3.2	Communities of Practice	22
3.3	Other Uses of Git	22
3.4	Suitability of Git for Learning Platforms	24
4	Platform Development	27
4.1	Design	27
4.1.1	Balancing Tensions	27
4.1.2	Platform Interactions	29
4.2	Components	31

4.2.1	Git hosting service	31
4.2.2	Chat service	32
4.2.3	Webinar platform	32
4.2.4	Server infrastructure	33
4.3	Technical Infrastructure	33
4.3.1	Deployment	33
4.3.2	Server Configuration	35
4.3.3	Content Configuration	37
4.3.4	Git Architecture	37
4.3.5	Custom Scripts	38
4.4	Weekly Operation	38
5	Results of RLA Pilot	39
6	Conclusion	41
A	RLA Challenge Descriptions	43
B	RLA Online Course Survey	47

List of Figures

5-1	The course-central repository.	40
-----	--	----

Chapter 1

Introduction

New education technology, enabled by the Internet, has allowed learners to connect with their instructors and with each other digitally. Pieces of software known as learning management systems take the experience of a classroom online, giving remote learners more access to education and enriching their learning experiences with a variety of technology.

In this thesis I employ Git, a version control technology initially created for collaboration among software developers, as the foundation for an experimental learning platform. The decentralized nature of Git can be utilized to expand interactions beyond replicating a traditional course. New interactions can be designed to balance the benefits of a collective learning experience with those of independent open-ended projects. In this study, I develop a prototype of such a platform, deploy it as part of an online course pilot, and examine the results. With this work, I hope to contribute to lowering the barrier of entry to administer an online course using a Git-based learning platform. The architecture I designed can be built upon in future courses.

The following sections outline the research methodology of this study. Chapters 2 and 3 cover the theoretical background and other similar projects that were taken as guidance in designing the new platform. Chapter 4 steps through the design and implementation of the platform. Finally, results of a pilot run of using the platform in an online course are presented in chapter 5.

1.1 Refugee Learning Accelerator

The development of a Git-based learning platform is part of a larger project called the Refugee Learning Accelerator (RLA). RLA is a program run by the Learning Initiative, a group of the MIT Media Lab. The goal of the program is to gather and support technologists from the Middle East to work on education tools for refugee learners.

The program ran in three phases from fall of 2017 through spring of 2018. First, interested candidates applied as teams through an open process. Around 100 participants were selected for the first phase of the program, which consisted of a 6-week online course during November and December of 2018. The course operated on a weekly cycle:

- Challenge materials released every other week introducing a new technology. Teams were asked to design and build a prototype of a solution related to refugee education using that technology.
- Twice during the week a webinar was conducted where an expert in the field discussed the week's technology and answered questions.
- Teams were given independence to work as they liked over the course of the week.
- Teams received qualitative feedback every week from the instructors once a week, at the midpoint and the end of each challenge.

The topics chosen for the online course were chatbots, human centered design, and AR. A description of each challenge is included in Appendix A.

The second phase of the program brought 14 selected teams together for a week-long, in-person workshop in Amman, Jordan. There, teams forged their project ideas through rapid iterations of feedback and field observation. One important goal of the workshop was to foster a supportive community, where members encourage one another to pursue their projects and share resources.

Following the workshop, teams worked independently to bring their projects to fruition in partnership with local NGOs and with the continued support of the RLA community and Media Lab staff. [16]

RLA is the product of the collective effort by members of the Learning Initiative and others passionate about finding creative ways to nurture learning, including the motivated participants, many of whom were working on their projects meanwhile also taking on a full-time load at work or as a student. The team of instructors included researchers and students from MIT and other affiliated experts. Mentors and speakers from all backgrounds donated their time to assisting teams through the webinars and at the workshop. The development of a learning platform a key one component in the operation of creating and executing the RLA program.

While one of my goals in building the platform was certainly to support the online course of RLA, I also hoped to explore the possibilities of a Git-based platform. In this way, the online course of RLA served also as a pilot run for the platform in order to evaluate its utility and shortcomings.

1.2 Methodology

Ultimately the research question I hope to answer in this thesis is: how can Git be used and adapted to support effective online learning communities?

This can be approached in three steps:

- First, I turn to established theory on digital learning pedagogy and principles to guide the characteristics and ultimately define interactions of the platform. In particular, I derive these interactions from first principles of learning and target learning outcomes rather than by attempting to move traditional classroom interactions online.
- Second, I implement a technical platform that supports these interactions. The bulk of this work is done before the pilot course, but also includes the technical operation of the course. As much as possible, I rely on third-party software to

increase reliability, reproducibility, and familiarity.

- Third, I use the pilot run to evaluate not only how these interactions work in the manifested platform, but also how well they work. In order to gather feedback, I conduct an anonymous online survey among course participants. The survey questions are included in Appendix B. Responses to the survey as well as anecdotal evaluation are reported in Chapter 5.

The code and course content are all freely available online for use in future iterations of such research. [16]

Chapter 2

Background

Educational theory and pedagogy comprise a vast field and have been subjects of study since long before the advent of online education. Here we briefly touch on frameworks that went into consideration for this thesis.

2.1 Traditional Learning Platforms

An online platform to support learning is not a new concept. A learning management system (LMS) is a tool that helps instructors supervise an online course, and popular commercial LMSs exist such as Blackboard, Moodle, and Canvas.

A traditional LMS typically provides features including tracking learner grades, file management for assignments and corresponding submissions, and communication mechanisms for instructor announcements or learner questions. These features are often convenient duplicates of the same in-person interactions of traditional classrooms. Furthermore, as LMS is designed to complement traditional classrooms, it usually does not fully encompass the activities of a class, and an all-online course would require a medium of instruction external to the LMS. [29]

Of note, the issue of learner collaboration, which is often a component of more creative, project-based assignments, is often not fully addressed in traditional LMS platforms. This intersection of collaboration and technology is instead known as computer-supported collaborative work (CSCW), and families of tools exist to sup-

port online collaboration. Learners thus would have to take to these external tools to collaborate, depending on the particular assignment.

2.2 Foundational Theory

2.2.1 Self-determination Theory

I begin with the assumption that humans have innate desire and ability to pursue their interests, and treat the role of the learning environment as fostering the right conditions. This is formalized in self-determination theory (SDT), which delineates three intrinsic psychological needs:

- Autonomy: to see oneself as the causal agent of one's life.
- Relatedness: to interact and connect with others in one's environment.
- Competence: the accumulation of skills and ability to accomplish tasks.

SDT suggests that when these three feelings are present, humans are optimally disposed to grow. [27][9]

The implication of SDT for design of a learning environment is to provide opportunities and encourage learners to be self-driven and motivated to explore their interests. Taking this perspective, not only can the learning environment cultivate ways of providing for the three basic needs prescribed by SDT, but can also channel learners' assumed individual motivation into achieving the desired learning outcomes. [10][19]

2.2.2 Learning Outcomes

We chose learning outcomes that align with the larger purpose of the RLA program and with consideration of the backgrounds of instructors and participants.

On a broad level, we borrow objectives of learning from what is commonly used at the Media Lab and in particular often referenced by Mitch Resnick of the Lifelong Kindergarten group: design creatively, reason systematically, and work collaboratively. These are not specific to RLA, but we can refer back to these general principles

when defining specific deliverables. In particular, we can frame specific objectives in terms of the abilities of learners to think, learn, and work, as opposed to reaching a pre-defined state. [25]

In the case of RLA, we are aiming for participants to continue beyond the course in building technologies for refugee learners, which involves outcomes in all three categories:

- Think creatively about new ways of utilizing technology for learning in unconventional settings.
- Learn systematically how to find and employ the latest technologies.
- Work collaboratively with other technologists, NGOs, and instructors to put a solution into practice.

The spirit of these outcomes can be found in the choice of challenge topics and deliverables (see Appendix A).

2.2.3 The 4 P's

We've established through SDT that learners can be self-driven, and that we would like to somehow cultivate a learning environment that channels this towards certain broad outcomes. Here we fill that gap. We lay out a framework for fostering creative learning based on four core components, the four P's of creative learning [24][23][21]:

- **Projects.** Project-based courses forces learners to design creatively and provides space for learners to realize their own vision.
- **Peers.** Learning as part of a community not only feeds the relevance component of SDT, but also sets up learners to work collaboratively and build upon each other's work.
- **Passion.** From SDT we take that learners are motivated by the ability to pursue areas of their own interest. We build this into the learning experience by allowing learners to work on what they care about.

- **Play.** In our learning outcomes, there is no preset finished product of learning. The same applies to the process. Allowing learners to tinker and experiment with ideas can help them find the best learning experience for themselves.

We center the design of the learning platform on these 4 P's going forward.

2.3 Design Tensions

Traditional courses emphasize delivering content through lectures, assignments, readings, and evaluations. The 4 P's call for some of the same aspects of traditional courses, but also an aspect of community that must be reconsidered in an online setting. Pulling these together creates tension between “course” and “community” and requires reconciling opposing characteristics of the learning experience:

- **Individual pathways vs shared experiences.** Learners prosper when given independence to really focus on their passions, but would then potentially miss out on experiences on which community can be built.
- **Team work vs solitary work.** Working in a team adds a feeling of belonging, but takes away from the autonomy component of SDT.
- **Content vs connection.** Some experiences are great for conveying information that is ultimately needed to learn systematically, but would take away from the unstructured time in which learners can form their own connections.

For each of these tensions, there are clear benefits on both sides to learning. Ideally, a learning experiences mixes and balances both sides of each tension.[22]

There are other tensions where our pedagogy stands clearly on one side of the debate:

- **Projects vs puzzles.** Projects are open-ended and allow learners to be creative and work along the best path for them. With puzzles, the path to a solution has been decided by the instructor and creativity is removed from the experience.

- **Open vs closed.** Collaboration and self-driven discovery are much easier when course content and others' work is visible to everyone.
- **Qualitative feedback vs objective rubric.** A rubric inherently assume that there is a set of checkboxes towards which learners work, again removing creativity and limiting exploration. On the other hand, constructive feedback not only tailors evaluation to the type of work that is best for the learner, but also can be used to encourage further play. Additionally, peer feedback connects learners with each other and enhances the community feeling.

We consider features of the platform and the facilitation of the RLA course along these dimensions.

Chapter 3

Related Work

In this chapter we survey existing technologies and platforms that we can learn or borrow from in developing a Git-based learning platform. Git has been applied to many fields, and its application to education has recently come under study by researchers.

3.1 Overview of Git

Git is a technology that is used for online collaboration, particularly in software engineering. Created in 2005 by Linus Torvalds as a tool to support the development of the Linux kernel, Git allows users to each have a local copy of a set of files and sync between their copy and a central server in a coordinated fashion. Git is also a version control system, hence allowing users to backtrack changes or split into multiple branches of work from the same starting state. [8]

Git is used as a command line tool, in which users type text commands to perform the desired git actions, but there are also graphical interfaces available which serve as a wrapper and replacement for the command line. On top of Git, hosting services such as Github, GitLab, and Bitbucket provide a range of features that are as integral to some workflows as Git itself. [1]

Overall, the benefit of Git derives from its powerful capabilities as a collaboration tool, but comes with the cost of an upfront time investment to learn its operation.

3.2 Communities of Practice

A community of practice consists of a group of people, a common interest that they share, and interactions or activities where the group acts on their interest. Members of the community learn from each other and improve their practice. Software development is an example of a community of practice, and Git was created to address the issue of collaboration on code in such a community. [6]

Git has gone on to prove useful in other communities of practice and has been adapted for collaboration in areas other than code. [20] The community of an online course can itself be seen as a community of practice, so other applications of Git may bear lessons on how to adapt it for learning.

3.3 Other Uses of Git

While Git was initially created for and is actively used by the software development community, its application has spread to many other use cases:

- **Programming-based courses.** Initially used for collaborating on work, Git is making its way into classrooms as a form of submitting work as well. Instructors may even set up repositories for students for their use or collaboration with teammates. [28] Github itself has even added features to streamline this use case. [14] However, note that this is not a LMS and usually is not used as a complete learning platform.
- **Contest submission.** DevArt by Google ran a contest for which they provided a template repository and submissions were accepted by taking advantage of Github's fork functionality. The template repository was forked over 2000 times. [11] Similarly, Mozilla accumulated submissions for its Global Sprint event using the issues feature on Github. [18]
- **Project management.** Communities of practice centered around a Git repository complete the workflow by using issue tracking (built into Github) and

real-time chat services in conjunction with the codebase. [3]

- **Datasets and legislation.** Git hosting platforms have the level of availability and navigation that make them suitable for hosting various collections of information including datasets and pieces of legislation. [20]
- **Blogging.** The version control aspect of git is often used as a content management system. Github provides support for serving files in Markdown format as static websites, which makes this even simpler. [28]
- **Recipes.** A service called Fork the Cookbook allows users can fork, or copy, an existing recipe and tweak it to make it their own. This creates a collaborative and decentralized platform for sharing cooking recipes. [5]
- **Travel logging.** A Github user has a repository for his personal travels, where onlookers can submit suggestions for attractions in a city he is planning to attend by suggesting edits for that particular city's file. [12]

Synthesizing these applications, we summarize several key techniques in the adaptation of Git to power communities of practice:

- Using Git hosting platforms as a link-able and easily navigable host of information.
- Using of version control and Markdown as a content management system.
- Parsing files managed by Git as a static web page, as done by Github Pages.
- Empowering members to contribute to a central file by submitting a suggested edit.
- Creating a submission of work by forking a template repository and adding work to the forked repository.
- Using group chats such to complement Git projects with more real-time communication.

3.4 Suitability of Git for Learning Platforms

While Git was not originally designed as a learning platform, we found in it suitable qualities. Here we discuss the benefits of git as a tool for fostering positive educational interactions.

First of all, Git aligns with the design tensions discussed above:

- **Individual pathways vs shared experiences.** Git gives individuals the freedom to take a self-defined path of work by being flexible in how it merges back into the main body of work. Learners can use their preferred tools and platforms, since any file type is supported. At the same time, the project repository acts as a shared space that learners share and can interact with each other in.
- **Team work vs solitary work.** Using Git allows learners to work independently on their own time, yet at the same time collaborate with teammates by building on each other's work. After all, its *raison d'être* is to be able to build upon each others' work without conflict.
- **Content vs connection.** Git can serve as a means of delivering educational content. As discussed in the previous section, Git can also be adapted as the underlying mechanism of a content management system. As a versioning system, Git enables publishing new content, updates, and revisions in a fashion that can be distributed. Popular git hosting websites such as Github and GitLab have built-in support for Markdown-formatted files and support and static web pages. At the same time, learners, can form connections with each other by reviewing each other's commits or suggesting edits to someone else's work.

Git also aligns well with the preferences for the type of experience RLA is designed to be:

- **Projects vs puzzles.** Git gives learners room to play, since branches can be made to explore new ideas and unsuccessful changes can be easily reverted.

- **Open vs closed.** By default, everyone's work is openly documented and visible. Commit messages log a complete history of changes.
- **Qualitative feedback vs objective rubric.** As a version control system, the currency is in units of iterative work rather than a final deliverable, reinforcing the idea that there is no preset rubric for what is considered good or bad work.

Finally, because Git is already widely-used, it is reliable in a technical sense and help material is readily available online for easing adoption. On the flip side, Git's wide use makes it an important skill. Code collaboration is practically a requirement in the software development industry in particular, and software is increasingly relevant in all industries, so learning Git as part of a participating in course can itself be an additional benefit. [15]

Chapter 4

Platform Development

In this chapter we walk through the design and implementation of the Git-based platform.

4.1 Design

In this section I lay out the high-level design for the RLA course. The platform and pedagogy are closely linked, but one does not necessarily imply the other. In other words, while a platform can be built to support a desired pedagogy, the content and facilitation are also critical to achieving it. On the other hand, a particular pedagogical approach may not be possible without a platform that can support it.

Hence, the design must encompass the whole course, from which the platform design will follow. Though the design targets the goals of the RLA program, it is intended to be adaptable for other future courses.

4.1.1 Balancing Tensions

We considered the tensions outlined above in several design decisions:

- **Individual pathways vs shared experiences.**
 - We allowed learners to apply as teams so that they self-select into groups around work styles, technical interests, and local refugee context. This

allows learners to work in teams, but still feel autonomy over who their team is and how they work within their teams.

- In webinars, sometimes we asked teams to split up for breakout sessions with smaller groups. This gave learners a chance to meet others outside their team and contribute to the shared experience of the overall course.

- **Team work vs solitary work.**

- Parts of the challenges involved teams agreeing upon a direction and producing a single result, such as identifying a problem to solve or creating a product pitch.
- Other parts of the challenges allowed teams to divide work by expertise or interest, such as building a prototype or having different roles in testing with users.

- **Content vs connection.**

- One of the secondary goals of the online courses is to foster a community that extends into the later phases of the Accelerator, and communication laid the groundwork for the type of community formed. We aimed for instructors to clearly communicate how to use the platform and be available to guide students throughout the course. General course information was be readily accessible to students.
- While this addressed content, we also wanted to address connection. We allow for natural and structured opportunities to ask questions and give feedback. We encouraged introductions at the beginning of the course and small talk outside the topic of the challenge was normal.

Along other dimensions we made intentional decisions to promote creative learning:

- **Projects vs puzzles.** Each week's challenge was intentionally left open-ended and phrased as a question rather than a directive.

- **Open vs closed.** We had to decide how to control the authority of performing actions and the visibility of content. On one hand, we believed that the platform should be as open as possible, but we also wanted to prevent either intentional or accidental actions that may destroy or hinder other participants' work. Anticipating that sensitive topics would not be expressed without a private way of doing so and thus would like to allow for that.
- **Qualitative feedback vs objective rubric.** We incorporated a weekly review in the form of comments on submissions. Each gave three pieces of feedback following the traffic light approach: one positive, one neutral, and one negative.

4.1.2 Platform Interactions

The details of the design were chosen to support the needs of RLA's weekly cycle. In each week of the cycle, content is released at the beginning and learners worked towards a particular milestone by the end of the week. Throughout the week learners can collaborate and participate in seminars.

To achieve these requirements, there are four interactions that the core Git-based platform needs to support:

- **Onboarding.** We want learners to be able to register for the course as easily as possible, yet still provide basic information to introduce themselves to the rest of the learner community.
- **Content distribution.** New material posted at the beginning of the week includes a description of the week's challenge and targeted milestones. In addition, resources relevant for the week are gathered and provided for learners' reference. This content may also sometimes include skeleton code or template documents in which learners can report their progress. These pieces of content require the additional ability of being pulled into learners' working directories.
- **Collaboration.** Throughout the week, teams work together on a single project or codebase in some sort of team work space.

- **Submission.** At the end of the week, learners need a mechanism to put together a package representing the state of their work on which to receive feedback. We also encourage learners to use these submissions as a resource to learn from other learners' approach to the challenge.
- **Feedback.** We believe frequent personalized feedback can be helpful to learners and also cultivates a culture of collaboration. Hence we'd like our platform to provide an interface where a reviewer can easily peruse a learner's work and leave comments.

Note that all these interactions have similarity in structure with feature development and code review used in software engineering, so we can utilize corresponding features in the Git hosting service.

In addition to managing the course content, we also want to support a few additional types of communication:

- For seminars, we needed a way to broadcast video to learners. Additionally, we would like to engage learners to drive the conversation according to their passions and have face-time with their peers.
- For technical support and other miscellaneous discussion, we would like to have an open, real-time, group chat. In particular, the group aspect will expose the entire community to conversations as appropriate so that everyone feels included and in-the-know.
- For showcasing teams' work, especially to parties outside the course, it would be ideal to have a static website summarizing the challenges and the progress coming out of the course. Anyone could then share the link and in this way feel ownership over their work.

4.2 Components

4.2.1 Git hosting service

In order to have a group of distributed learners participate in the course using Git, we needed to have an server accessible by the Internet running Git where the content would be hosted. We looked at two of the most popular Git hosting services, Github and GitLab. Both offer a SaaS model where they host Git on their servers and simply provide an interface, as well as a self-hosted version where they provide the software but leave running a server to the customer.

We compared the two in terms of features, branding, and technical performance:

- **Features.** The two options had very similar sets of features. Github has organizations, teams, repositories, and pull requests, while GitLab has groups, subgroups, projects, and merge requests. Both expose a webhook for Git commits, allow in-browser editing of files, and can serve hosted files as static websites. Both offer free or discounted private repositories/projects for educational uses. On a few points GitLab had additional features, such as being able to request to join a group and being able to trigger scripts that run before serving static websites.
- **Branding.** We took into account what implications the branding of the sites would have for learners. Many software companies use Github to host either all their code or even just their open source projects, and as a result a Github profile feels very much like a professional portfolio. We wanted to make sure learners are comfortable experimenting and making mistakes, and took into consideration that some may feel less inclined to do so if the work would permanently be associated with their professional portfolio.
- **Performance.** We took into consideration the technical reliability of each option. We found that GitLab's SaaS offering, GitLab.com, was often slow to load. For both, a self-hosted instance would include considerable setup and costs to run a server, but would allow customizations such as the domain name.

Finally, though both are welcoming homes to open source projects, Gitlab has made the web platform itself open source, whereas Github's is not.

For our course, we ended up opting for GitLab, and chose to mitigate performance issues by using a self-hosted instance of the free tier. However, it would certainly be possible to adapt our approach for either GitLab or Github, in the both SaaS and self-hosted models.

4.2.2 Chat service

To allow the development of a community, we wanted to make communication among learners as well as between learners and instructors as open as possible. GitLab alone, with commit messages and comments on merge requests, did not offer as fluid of a setting as we wanted. We decided to augment GitLab with a team-based chat service where learners could communicate in real time.

We looked at existing cloud messaging services including Slack, Gitter, and Mattermost, all of which have a very similar design. They provide open channels where any user can join and chat, as well as messaging between users and in private groups for more direct communication. We hoped to use these spaces as a forum for discussing ideas, asking each other for help, and even just to meet other learners.

From our options we chose to run a self-hosted instance of Mattermost, largely due to our choice of self-hosted GitLab. Mattermost and GitLab have a built-in integration for authentication and backend configuration. Mattermost is also open source, continuing in our spirit of relying on open source tools.

4.2.3 Webinar platform

We wanted to incorporate seminars into the content of our course, and for this we needed a video-streaming platform, ideally also allowing learner interaction. The Learning Initiative at the MIT Media Lab has developed exactly such a platform called Unhangout [4], so we used this for our course. An Unhangout session has a

main video stream and allows participants to break out into smaller groups in a video conference format. Unhangout is also open source.

4.2.4 Server infrastructure

In order to host the components above, to serve websites, and to run any additional scripts, we required an Internet-accessible server. For addressing this need we looked at cloud computing platforms and domain name registration services. There are many vendors in these markets and we didn't find significant benefits in a particular one over any of the others. We ended up using DigitalOcean for the platform server and Namecheap to register the domain.

We also sought to enable HTTPS secure connections to the platform server. To do so, we utilized Let's Encrypt, a certificate authority that provides free automatically generated certificates.

4.3 Technical Infrastructure

In this section I walk through the steps of setting up a platform to meet the needs of the design described above. I assume a technical background, in particular familiarity with Internet architecture and Git. My approach was very much iterative, where the most restrictive components were configured first followed by smaller features within that component.

4.3.1 Deployment

First of all, I created a droplet in DigitalOcean, which is essentially a virtual machine that would serve as the platform server, with the following specifications:

- The server ran Ubuntu Linux as the operating system.
- Following GitLab's recommendation, I allocated 4 GB of RAM initially. Later in the course, as the server supported more programs including Mattermost

and static webpages, this had to be increased. To do so, I shut down the server temporarily and rebooted the image on a larger droplet.

- I initially allocated 60 GB of disk space. As with the RAM, this had to be increased later in the course, especially when participants were storing AR projects with graphics in their workspaces on the server.
- DigitalOcean provides an option to create the droplet using pre-defined images, one of which has an instance of GitLab. I chose this image to streamline installing GitLab. Using a different cloud computing provider without such a feature would alternatively require downloading the source code to the server and compiling on the machine manually.

Next, I purchased an easy-to-remember domain name using Namecheap that we could connect to the platform server. This would serve as the landing page and home of the course. To connect this domain to the platform server, I configured nameservers with A records, which point a domain to an IP address. In particular, I created subdomains as follows, all of which were connected to the IP address assigned for the platform server:

- *: The * subdomain indicates the root domain, or the absence of a subdomain. The server was initially configured to redirect requests here to the GitLab subdomain, and later this became the address for the course homepage.
- **gitlab**: Initially this was called “code,” but was renamed to reduce confusion since we started referring to the GitLab instance simply as “GitLab.” This was configured to serve the home page of GitLab.
- **mattermost**: Initially this was called “chat,” but again was renamed to reduce confusion. This was configured to serve the home page of Mattermost.
- **pages**: This was reserved for the possibility of teams wanting to serve their own webpages, but in the end we didn’t reach this point. Instead, we automatically generated websites for each team based on their submitted work and served them as part of the course homepage at the root domain.

This could be equivalently be accomplished either in Namecheap’s console or by configuring Namecheap to redirect to DigitalOcean’s nameservers and configuring the records in DigitalOcean’s console. I chose the second to consolidate all configuration to one console.

On the server’s side, GitLab has built-in configuration for nginx, which is a tool that directs how web requests to a server are routed and responded to. While GitLab requires certain settings in the nginx configuration files, it also allows for customization that I later used to add capabilities including HTTPS, static sites, and Mattermost.

4.3.2 Server Configuration

Throughout this section, I will cover modifications made to the server through SSH in the command line.

The droplet was created with an instance of GitLab already present, so I then went through the process of configuring individual settings for the instance. This was done by changing variables in a specific Ruby file. GitLab provides comprehensive documentation on this configuraton [2], but I touch on a few key settings here:

- The external URL was set to match the registered domain, so that GitLab would automatically configure nginx to route requests to GitLab properly.
- Email notifications were enabled. To do so, first I created a GMail account for the course administrators and configured GitLab with the appropriate SMTP authentication details.
- Custom nginx configuration was enabled. GitLab autogenerates the main nginx files running on the server, so to insert custom configuration requires pointing GitLab to a file containing the additional configurations.
- Integrating with additional services such as Mattermost and GitLab Runner, which are discussed in more detail below.

I then restarted the GitLab service, which triggered GitLab to update its own configuration files in the background. For each key change, I manually tested the functionality; for example, I triggered a test email notification to my own account and checked my inbox for the email. Additionally, on a couple occasions, users needed their passwords reset or accounts deleted, which was accomplished by applying updates to the underlying database through the command line.

By default, GitLab does not support SSL connections and hence accessing the website via HTTPS would not work. To fix this, I used Let's Encrypt's command line tool to issue certificates for each of the subdomains listed above. This involved inserting nginx configuration that would respond to Let's Encrypt's servers' cryptographic challenges. Once set up, these certificates would be valid until expiration 90 days later. This exceeded the length of the course, but were configured to autorenew when close to expiration anyway.

At this point, the bare version GitLab was fully functional and could be used via the web interface.

The next large component to set up was Mattermost. As with GitLab, Mattermost is open source and was installed by downloading and running the source code on the platform server. Furthermore, GitLab allowed for integration with Mattermost on the level of the server code. This was accomplished with further adjustments in GitLab's configuration. Other configuration could also be made in this file, including:

- Setting the external URL to properly route requests to the Mattermost instance.
- Telling Mattermost to recognize HTTPS connections using the issued certificate.
- Permissions for Mattermost users to create and delete teams.

Mattermost also has a mobile client that can be used to access chat channels through a native mobile app. I tested this using my personal phone and wrote instructions for how learners could download and log in to the client.

The most important result of the integration between GitLab and Mattermost was the shared authentication. That is, a user could create an account on the GitLab

instance, and then use the same credentials to log in to the Mattermost instance. This greatly simplified the onboarding process for learners.

Finally, a similar process as for setting up Mattermost was required for setting up GitLab Runner. This is a service separate from GitLab that runs the appropriate scripts to prepare and serve static websites using the files in a GitLab-managed Git repository, among other uses. I downloaded the source code onto the platform server and started the service, then adjusted GitLab's configuration file yet again to properly route requests to the service. An additional step here was to register the instance of GitLab Runner with GitLab through GitLab's web interface. After these steps, GitLab users could use the GitLab web interface to trigger jobs for the GitLab Runner instance to run.

4.3.3 Content Configuration

To do...

4.3.4 Git Architecture

To do...

- **Divided by folder.** All teams commit to one shared master branch, but in different folders. The benefit of this approach is its simplicity, but the downside is the lack of permissions enforcement to avoid one team interfering with another's work.
- **Divided by branch.** All teams commit to different branches of a single class repository. This way teams can even rebase on master to gain access to new course materials. Though tricky, it is also possible to enforce appropriate permissions so teams can only push to their own branch. The downside is that it becomes messy for teams to create their own branches for development.
- **Divided by repository.** All teams work in their own repositories, which are pulled together each as a subtree within a course repository. This allows for

complete freedom for teams within their own repository with simple permission settings, but creates complexity in having each team's work in multiple locations, namely in two separate repositories.

4.3.5 Custom Scripts

As much as possible, Git-related operations of the platform were automated to ease the burden on the instructors.

- Group membership.
- Onboarding group creation
- Health check with email notifications
- Create subtrees
- Create merge request
- Perform merge
- Analyze progress

To do...

4.4 Weekly Operation

Chapter 5

Results of RLA Pilot

For six weeks in November and December of 2017, approximately 100 learners participated in an online course using the Git-based learning platform as the first phase of the RLA program. Following the course, an optional survey was conducted, the questions of which are included in Appendix B. In this section we report the experience, results, and feedback we observed in the course and survey responses.

To do...

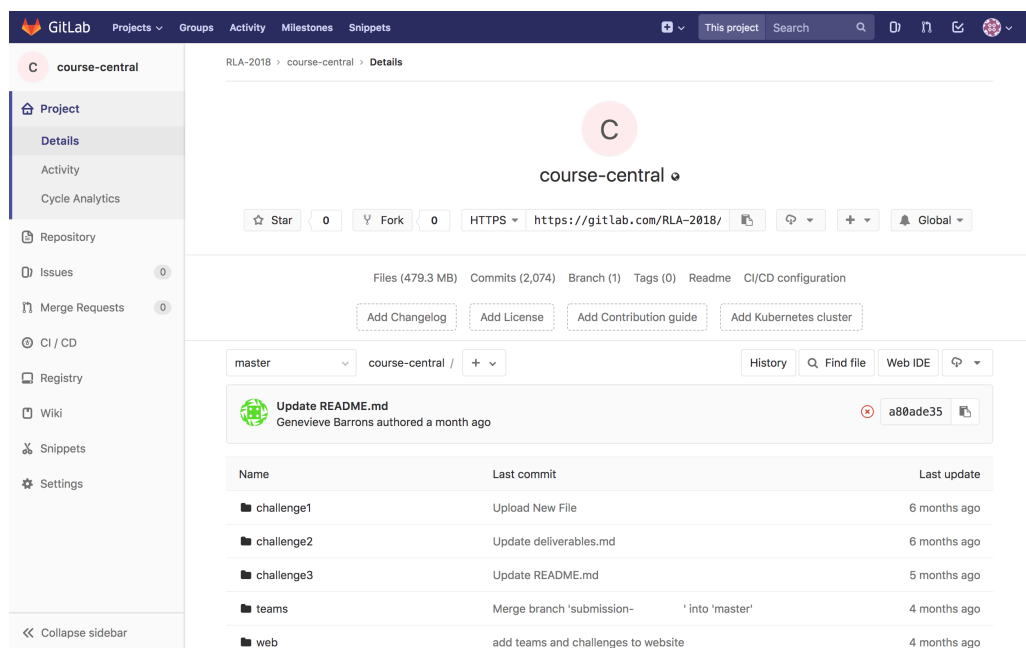


Figure 5-1: The course-central repository.

Chapter 6

Conclusion

To do...

Appendix A

RLA Challenge Descriptions

The online course was divided into three challenges, each of which had two sets of deliverables (one for each week). Here we provide brief descriptions of these. More information as well as various resources related to the topics of each challenge is available online. [16] All content was developed by members of the MIT Media Lab Learning Initiative unless otherwise noted.

1. **Challenge 1 - chatbots:** How might we use chatbots to support refugee learners inside or outside of the classroom?
 - Describe your concept for a chatbot in one or two paragraphs. Use visuals (mockups, diagrams) if possible. It is important to be as specific as possible while mapping out the scope of the project as this will greatly influence implementation and the tools to use. Answer the following questions:
 - What problem/ challenge will the chatbot solve?
 - How will the chatbot solve it?
 - Who is the primary user and how will the chatbot engage the user?
 - What activity does the chatbot facilitate that would not otherwise be possible?
 - What challenges do you expect to encounter?
 - Demonstrate how your chatbot works. Videos, screenshots or images are encouraged. Add a short description and answer the following questions:

- How did you build it? (Platform and technology)
- What challenges did you face?
- What aspect of the chatbot do you like best?
- What would you different in the future?
- Please also add a link to your code!

2. **Challenge 2 - human centered design:** How might we use everyday technologies as learning tools?

- Make an initial prototype. Questions to answer:
 - What solution are you testing? (and why did you choose it?)
 - Submit your prototype (use photos, video, diagrams etc.)
 - Describe the prototype and why you chose this prototyping method.
 - What did you learn during the prototyping process?
 - Who are your intended users for testing?
- Test it with users. Questions to answer:
 - How did you select your test users?
 - What was the setting of the test?
 - What were the main points of feedback you received (share a summary)?
 - What changes would make to your idea/project based on the feedback?
 - What parts of the Human Centered Design process were new to you?
 - What parts of the Human Centered Design process seemed most useful to you?

3. **Challenge 3 - augmented reality (AR):** How might we use Augmented Reality (AR) to support refugee language learning outside the classroom?

- Describe your concept for how low fidelity Augmented Reality could be used to support language learning in one or two paragraphs. Use visuals

(mockups, diagrams) if possible. It is important to be as specific as possible while mapping out the scope of the project as this will greatly influence implementation and the tools to use. Answer the following questions:

- What problem/challenge will the AR experience solve?
 - How will the AR experience solve it?
 - Who is the primary user and how will the AR experience engage the user?
 - What hardware does the user need? Is this realistic in the refugee context?
 - What activity does the AR experience facilitate that would not otherwise be possible?
 - What challenges do you expect to encounter?
- Demonstrate how your AR experience works. Videos, screenshots or images are encouraged. Add a short description and answer the following questions:
 - How did you build it? (Platform and technology)
 - What challenges did you face?
 - What aspect of the AR experience do you like best?
 - What would you do differently in the future?
 - Please also add a link to your code!

Appendix B

RLA Online Course Survey

The following questions were sent in an email to participants to gather feedback and evaluate how well the platform worked. The survey was anonymous and asked for consent for responses to be used in this research study. A total of 11 participants chose to respond.

1. Git

- How much had you used Git before the course?
- To what extent did you learn Git during the course?
- How useful would you say knowledge of Git was for collaborating with your team?

2. Course materials

- Did you have any difficulty accessing course materials? If so, what were they?
- Did you find the course materials helpful in working on your projects? If so, which materials were most helpful?

3. Signing up

- Did everyone on your team submit their own initial sign up on GitLab?

- How many people on your team used GitLab after the initial sign up? Was there a reason some did not?
- Did you find it useful to have a list of participants in the course?

4. Submission

- Would you have preferred an automatic, continuous submission (as opposed to your work being snapshot at a set time)?
- Did you run into any issues in submitting your work? If so, what were they?
- Were you able to see other team's submitted work? If so, did you find that useful?

5. Seminars

- What were your favorite and least favorite features of the Unhangout platform for seminars?
- Did you find the seminars useful in ideating for your projects? If so, were any seminars particularly useful?
- Did you find the seminars useful for the technical side of your projects? If so, were any particularly useful?

6. Team

- How was the technical work distributed among your team?
- Was your team physically located in the same place?

Bibliography

- [1] GitHosting. <https://git.wiki.kernel.org/index.php/GitHosting>. Accessed: 19-May-2018.
- [2] Omnibus GitLab documentation. <https://docs.gitlab.com/omnibus/>. Accessed: 19-May-2018.
- [3] Project management, made simple. <https://github.com/features/project-management>. Accessed: 19-May-2018.
- [4] Unhangout. <https://unhangout.media.mit.edu/>. Accessed: 19-May-2018.
- [5] About Fork The Cookbook. About Fork The Cookbook. <http://forkthecookbook.com/about>. Accessed: 19-May-2018.
- [6] Bates, A. W. (2015). *Teaching in a Digital Age: Guidelines for Designing Teaching and Learning*. Tony Bates Associates, Vancouver.
- [7] Brennan, K. and Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association*.
- [8] Chacon, S. Getting Started - A Short History of Git. <https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git>. Accessed: 19-May-2018.
- [9] Cherry, K. What Is Self-Determination Theory? <https://www.verywellmind.com/what-is-self-determination-theory-2795387>. Accessed: 19-May-2018.
- [10] David, L. Self-Determination Theory. <https://www.learning-theories.com/self-determination-theory-deci-and-ryan.html>. Accessed: 19-May-2018.
- [11] DevArt: Art Made with Code. DevArt: Art Made with Code. <https://devart.withgoogle.com/>. Accessed: 19-May-2018.
- [12] Egan, D. travel. <https://github.com/dylanegan/travel>. Accessed: 19-May-2018.
- [13] Feliciano, J. (2015). Towards a Collaborative Learning Platform: The Use of GitHub in Computer Science and Software Engineering Courses. Master's thesis, Department of Computer Science, University of Victoria. An optional note.

- [14] Github Classroom. Github Classroom. <https://classroom.github.com/>. Accessed: 19-May-2018.
- [15] Haaranen, L. and Lehtinen, T. (2015). Teaching git on the side: Version control system as a course platform. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, pages 87–92. ACM.
- [16] Initiative, L. Refugee Learning Accelerator. <https://refugeelearning.media.mit.edu/>. Accessed: 19-May-2018.
- [17] Kelleher, J. (2014). Employing git in the classroom. In *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*, pages 1–4. IEEE.
- [18] Mozilla’s Global Sprint. Mozilla’s Global Sprint. <https://mozilla.github.io/global-sprint/>. Accessed: 19-May-2018.
- [19] Niemiec, C. P. and Ryan, R. M. (2009). Autonomy, competence, and relatedness in the classroom: Applying self-determination theory to educational practice. *School Field*, 7(2):133–144.
- [20] Orsini, L. Seven Ways To Use GitHub That Aren’t Coding. <https://readwrite.com/2013/11/08/seven-ways-to-use-github-that-arent-coding/>. Accessed: 19-May-2018.
- [21] P. Schmidt, M. R. and Ito, J. Creative Learning and the Future of Work. *Disrupting*, page 147.
- [22] P. Schmidt, M. R. and Rusk, N. Learning Creative Learning: How we tinkered with MOOCs. <http://reports.p2pu.org/learning-creative-learning/>. Accessed: 19-May-2018.
- [23] Resnick, M. (2014). Give P’s a chance: Projects, peers, passion, play. In *Constructionism and creativity: Proceedings of the Third International Constructionism Conference*. Austrian Computer Society.
- [24] Resnick, M. (2017). *Cultivating Creativity through Projects, Passion, Peers, and Play*. The MIT Press, Cambridge.
- [25] Rheingold, H. Mitch Resnick: The Role of Making, Tinkering, Remixing in Next-Generation Learning. <https://dmlcentral.net/mitch-resnick-the-role-of-making-tinkering-remixing-in-next-generation-learning/>. Accessed: 19-May-2018.
- [26] Rocco, D. and Lloyd, W. (2011). Distributed version control in the classroom.
- [27] Ryan, R. M. and Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American psychologist*, 55(1):68.

- [28] Zagalsky, A. Why you should use GitHub: Lessons for the classroom and newsroom. <http://www.storybench.org/use-github-lessons-classroom-newsroom/>. Accessed: 19-May-2018.
- [29] Zagalsky, A., Feliciano, J., Storey, M.-A., Zhao, Y., and Wang, W. (2015). The emergence of github as a collaborative platform for education. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 1906–1917. ACM.