Chapter 1

Introduction

Welcome to Mendicant University! Over the next few weeks, you'll learn a lot while working on fun projects. Our courses are different than most other trainings out there, so we've put together this guide to help you find your way.

Rather than using the traditional lecture-homework-review cycle, our courses are entirely project based. Between your personal project, the community service project and the three exercises we provide, you'll have plenty to think about and work on as soon as the course begins.

As you work on these projects, you will inevitably come up with questions or run into unexpected problems. We provide large amounts of office hours and regularly do code reviews so that we can give you personalized feedback that will help you improve. Mendicant University is as much a mentoring program as it is a training program, and this shines through in the way we run our core skills course.

Chapter 2

Personal Project

Your personal project is your chance to work on something that's useful to you while still learning new things in the process. It can be pretty much anything you'd like, as long as you keep the following guidelines in mind.

- This is going to be a very busy course, and you'll need to balance your time between this project and the other assignments. You should pick a project that's challenging enough to generate some interesting conversations, but not overwhelmingly difficult for you. Keep in mind that only work done between 1/9 and 1/30 will be evaluated!
- Web applications often involve a lot of non-Ruby work and time consuming boilerplate chores. We tend to be a bit more picky about approving web application ideas for this reason. But if you've got a web-based idea that you really want to work on, don't let this discourage you from proposing it.
- Due to the current incompatibilties between Shoes and different operating systems, we are not approving any project proposals that make use of Shoes.
- Up to three students can work on the same project, but each student needs to send in their own proposal outlining what specifically they plan to work on.
- Your project needs to be something you can work on publicly, and release under a free software license.

To submit a proposal, log into university-web, and click on your submission for the Personal Project assignment to reach the comments thread for your project. Then enter your proposal select "Feedback Request" at the bottom and click "Comment & Update Status". Your initial proposal just needs to give a rough overview of what sort of project you want to work on and what you expect to be able to complete within the three week session. A couple paragraphs should be enough to get the conversation started, we'll ask for extra details if needed.

Chapter 3

Community Service Project

Contributing to existing open source projects can be an amazing learning experience. For this reason, and because we want to encourage students to value community service, we have made this activity a requirement for passing the core skills course.

The list of available projects/mentors are listed below:

- University Web, PuzzleNode, Mission of Mercy (Jordan Byron)
- RubyMoney, cashrb, Rubinius, Mendibot (Shane Emmons)
- Ivory Tower, Bookie, Stack Wars (Gregory Brown)
- Prawn, Crawdad (Brad Ediger)
- rstat.us (Carol Nichols)
- AgoraOnRails (Alberto Fernández Capel)
- Ruby Documentation Project (Andrea Singh)

You can find out more about each of these projects by looking at the Community Service assignment description in university-web. While we expect most folks will work on the projects listed here, if you have a really good idea for a project from a maintainer not listed, just let us know and we may be able to work something out.

Once you've chosen a project, update your community service submission with which project you'd like to work on and choose "Request Feedback". This lets us know that your proposal is ready for review.

Multiple students may work on the same project as long as each student is doing a significant amount of original work on their own.

Chapter 4

Exercises

You will be given three exercises to work on, covering the following themes: software integration, complex modeling/requirements discovery and academic topics. Each exercise will typically be open ended, and so you're encouraged to pitch your project ideas before beginning work on them so that we can help you decide if they'll be a good fit for the course. You'll be expected to work on at least two of the exercises. One of them must meet the quality standards of the course, while the other need only meet the functionality requirements of the exercise.

We strongly prefer realistic projects to contrived ones where possible, and will almost always require you to write code that actually runs, even if it means stubbing out some things that you won't have time to build out during the course. We expect you to put some effort into making sure that your code is easy to review: consider each formal review of your project to be a mini-release, and don't expect us to review rushed or unnecessarily messy code. That having been said, perfection is not expected, we only ask that your code reflects the best of your own abilities.

The exercises will be designed in such a way that you can and should collaborate with the other students in your session. You are free to share common bits of code or learn from each other's implementations, as long as the code you end up submitting still contains a large amount of your own original work. For a number of reasons, we ask that you do not share any materials from your session (including your own work) with anyone outside of the course without Gregory's permission. You will be able to post your work publicly as soon as the course ends.

Chapter 5

Feedback Requests

To request feedback on your work, add a comment to your submission and choose "Feedback Request" and click "Comment & Update Status". After doing so, you can also request feedback via IRC or the mailing list. Everyone is encouraged to participate in requests for feedback. You can request feedback at any time during the course on any assignment, you don't need to have a completed project to ask what others think about it. Of course, the following bits of advice are worth following to help you get the most out of your reviews:

- Make sure your code is cleaned up enough to be reasonably easy for a reviewer to understand. If you don't know how to do this, point that out in your first submission and ask for help.
- Include instructions in your feedback request that tell reviewers where to find the important parts of your project, and describe what works, what doesn't, and any obstacles that you've run into that may get in the way of their review.
- If you have specific questions for your reviewers, ask them up front! That will make it much more likely that you'll get the kind of feedback you need.
- Make sure to have either some tests that can be run, or some good examples that describe the project, and at least a semi-useful README. Ideally, have all of the above, but don't stress too much on testing every last thing, particularly if you're short on free time.
- Do not be afraid to ask for feedback early and often! This is where the learning happens in MU courses, and we love to give feedback.

The purpose of requesting feedback is to help you work through any problems you're having working on your project, and also to figure out how much more work you should do before you submit your code for evaluation.

Chapter 6

Evaluation Requests

Evaluations are conducted by Gregory and Shane, and determine whether your submission: a) meets the functional requirements of the assignment and b) meets the expected level of quality that would qualify you for joining our alumni network. These reviews are requested via university-web by selecting "Evaluation Request" and hitting the "Comment & Update Status" button on the comment thread for your submission. In addition to the guidelines that apply to requests for feedback, requests for evaluation should also keep the following points in mind:

- In order to be evaluated, your project needs to meet the functional requirements of the exercise, so if your project isn't 'done', then ask for general feedback instead.
- Your code does not need to be perfect, but it should be in a 'releasable' state, in which it can be used in environments other than your own machine, with a proper project structure and code that follows the most common conventions used in open source Ruby projects.
- Half-done features or defective features should be removed before requesting an evaluation. Only code that has been at least manually tested and shown to work should be included in submission. Delete any commented out code, or experimental code that is unused by the core features of your project.

Chapter 7

Scheduled Checkpoints

Please keep the following deadlines in mind as you work through the course.

- January 2: Initial ideas for individual and community service projects must be submitted via university-web.
- January 9: Individual and community service projects must be approved; course start date.
- January 16: At least one of your exercises must be functionally complete, and one meaningful feature needs to be implemented for your individual project. You should also have done any necessary research for your community service project, including discussing any potential blockers or obstacles with the maintainer of the project you are contributing to.
- January 23: The assignment you submitted on 1/16 must now meet quality standards. Significant progress must be made on a second assignment, as well as your individual and community Service projects.
- January 30: At this point, you need to have submitted at least two functionally complete assignments, one of which may be the one you submitted on 1/9. Your individual project needs to be cleaned up and useful, ideally matching what you originally proposed to get done, but if not, should at least be a well polished usable subset of your original plan. Your work on your community service project must be wrapped up and a pull request must be submitted to the maintainer.

Our deadlines are firm and you are expected to submit work on time throughout your course. We do give extensions from time to time when we fail to give feedback quickly enough for students to act on before a deadline. But if you think you'll miss a checkpoint for pretty much any other reason, you should talk to us as early as possible so that we can discuss how to reduce the scope of your assignments so that you can still deliver something useful on time.

Chapter 8

Time Management Advice

Our courses are meant to be challenging but should not feel like death marches, so if you feel overburdened by time pressures, be sure to talk to us and we'll help you.

We don't mind if you let our course fill up the time that you'd ordinarily use to work on your own side projects or hobbies. We also think it's fine if you've taken some time off from your day job so that you can focus on your session, even though you shouldn't feel pressured to do so. The one thing we don't want is for your sessions to get in the way of your sleep schedule or family responsibilities. If that's happening, you're doing something wrong! Part of the core skills course is a deliberate lesson in time management, so don't hesitate to ask for advice on this topic rather than being bowled over by it.

Chapter 9

Gaining Alumni Status

At the end of your session, if you have made it through all your checkpoints successfully, you may be invited to join the MU alumni network. We have a great community of learners who've been down the same road you have, and a number of alumni courses and activities that we're continuously working on. However, please focus on the journey and not the destination. The goal of an MU core skills session is to become a better software developer, not to become an alumnus.

This pretty much sums up what you need to know, so good luck, and happy hacking!