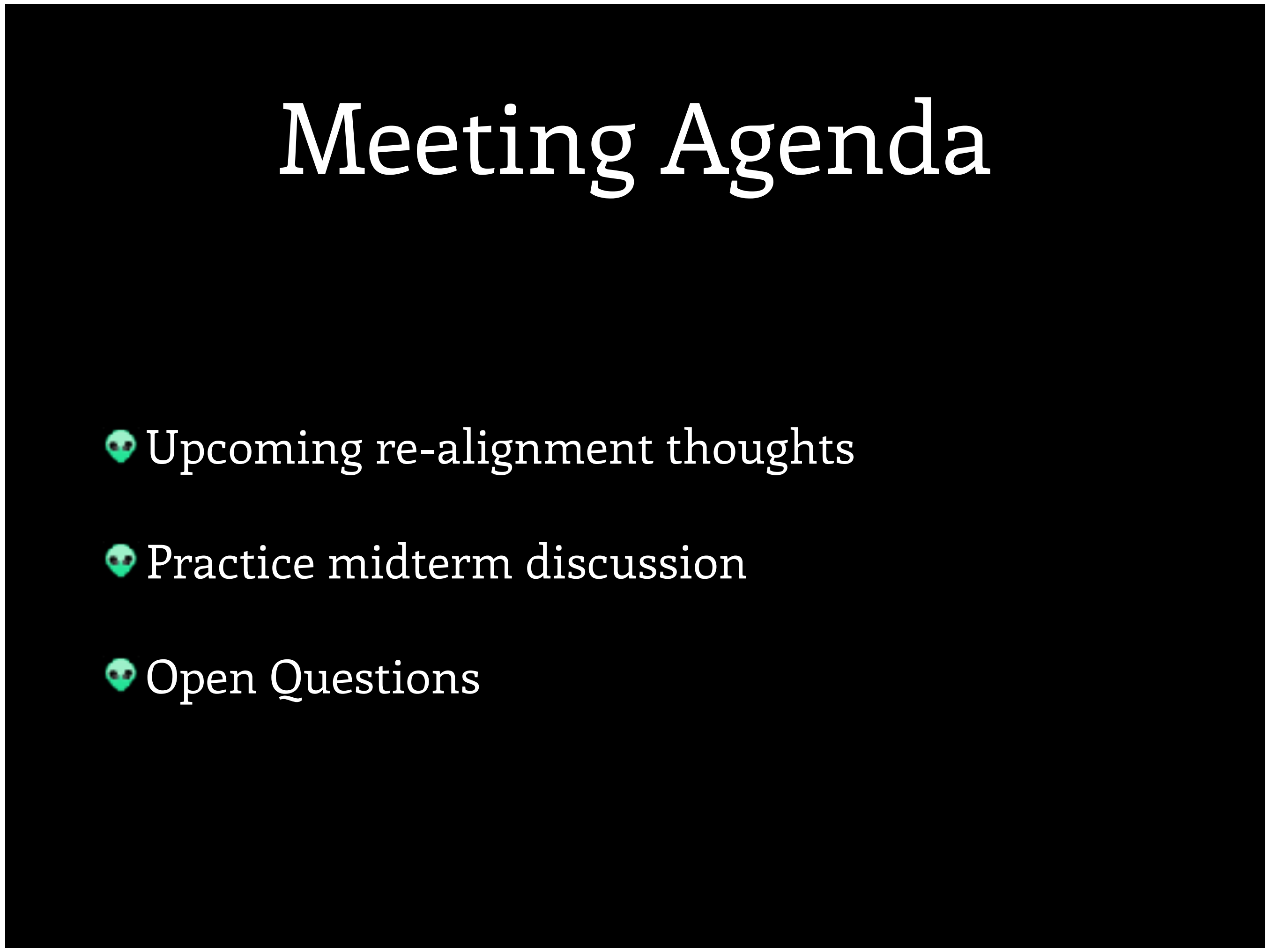
Working Session

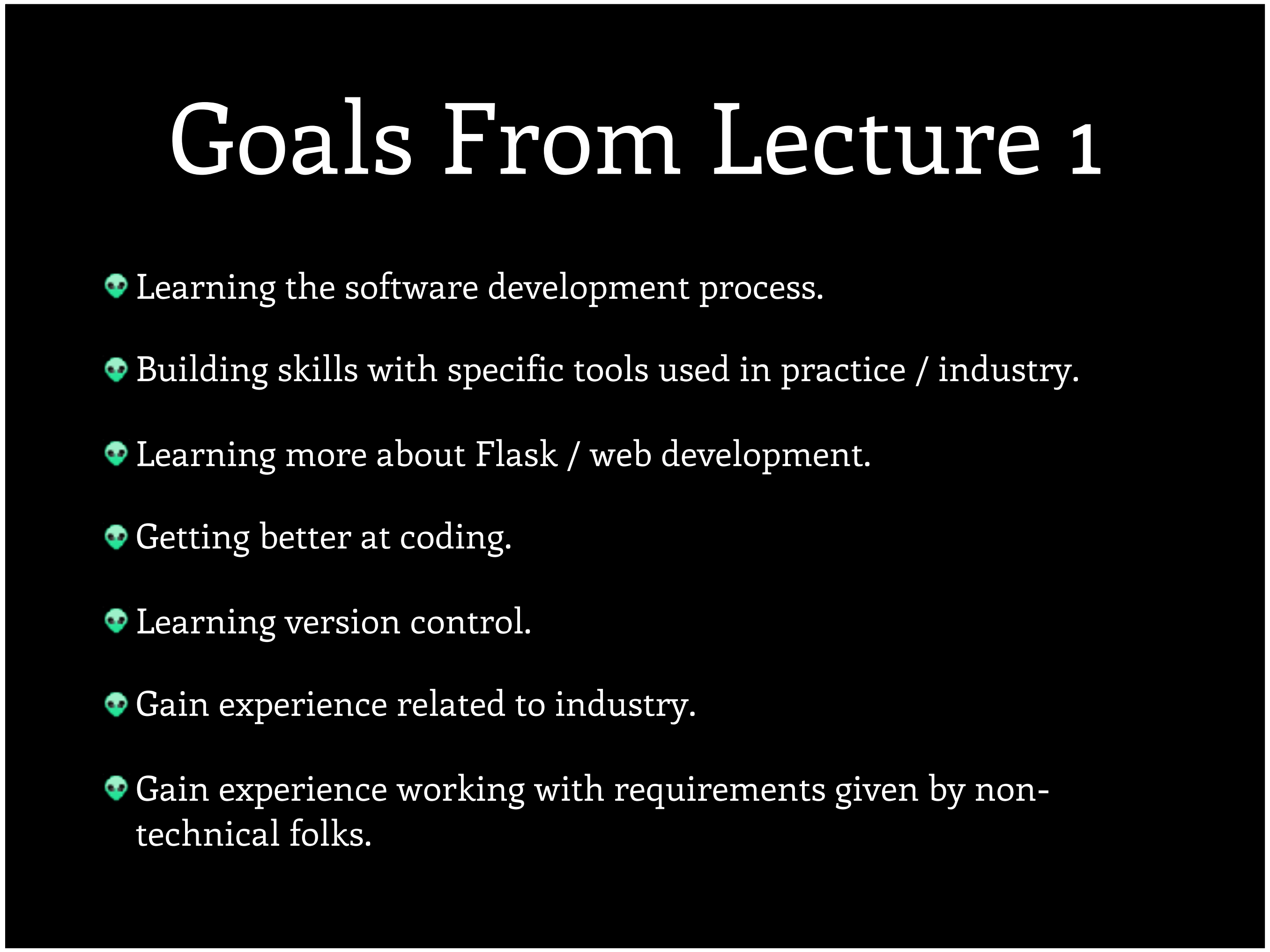
Feb 8



Meeting Agenda

- Upcoming re-alignment thoughts
- Practice midterm discussion
- Open Questions

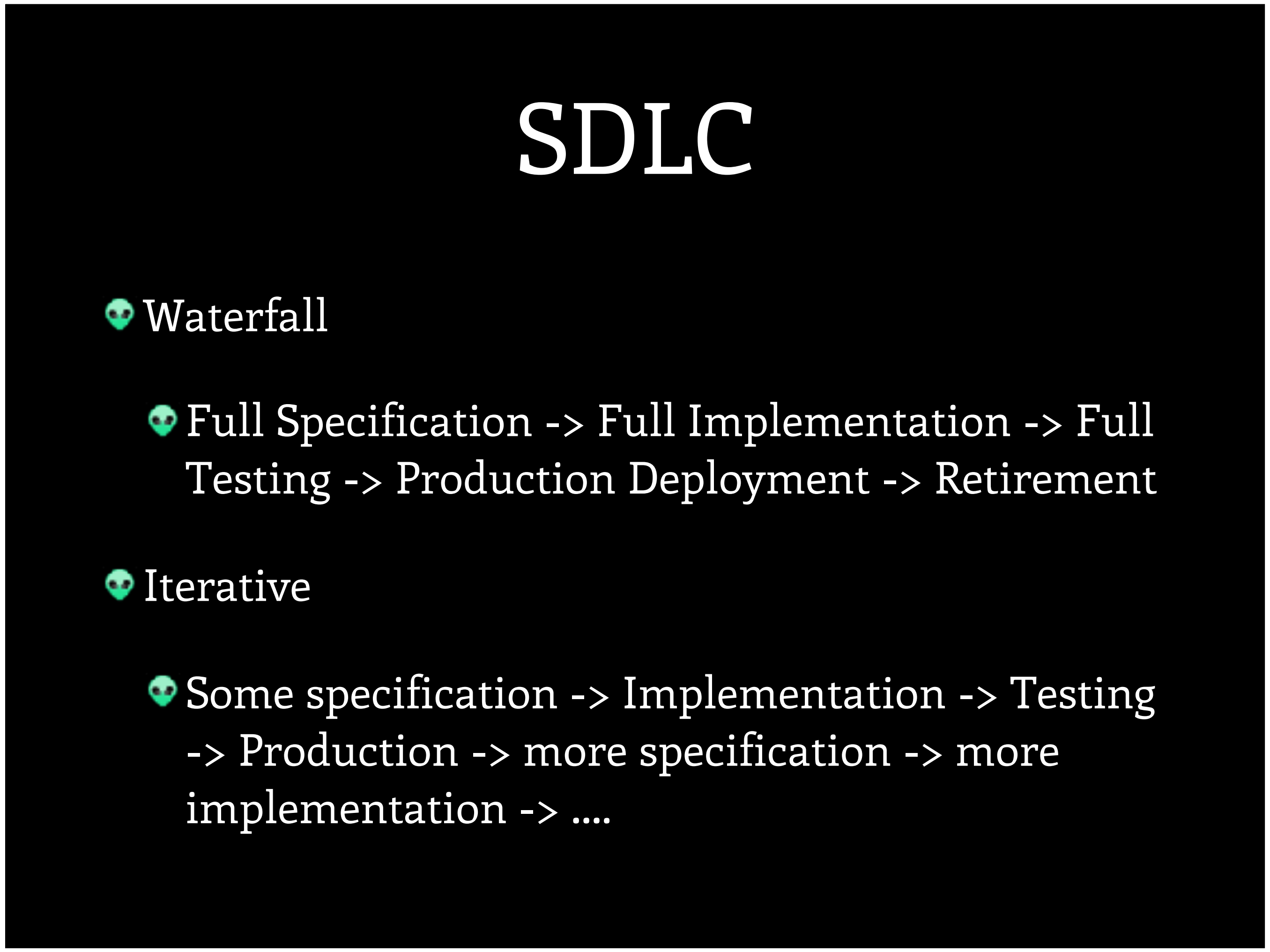
Early on we’d talked about running the project like I’ve run project. I’m not scaling well. The practices that work with teams of 8-12 full-time developers do not scale to 30+ developers. Similarly, the way requirements, objectives, and solutions are posed do not scale under the limited number of contact hours. I feel like I need to make a pretty drastic realignment in the way the course is being taught and the term project. I figure the middle of the term is a good place for that.



Goals From Lecture 1

- Learning the software development process.
- Building skills with specific tools used in practice / industry.
- Learning more about Flask / web development.
- Getting better at coding.
- Learning version control.
- Gain experience related to industry.
- Gain experience working with requirements given by non-technical folks.

These are the goals Andy captured from our meeting on the first day of class. I think we've at least touched on all of these even if we haven't directly talked about them.



SDLC

- Waterfall
  - Full Specification -> Full Implementation -> Full Testing -> Production Deployment -> Retirement
- Iterative
  - Some specification -> Implementation -> Testing -> Production -> more specification -> more implementation -> ....

For really large projects, I think some variation of waterfall is needed. Having the specification upfront enables better segmentation of the problem and design of the interfaces so that several teams can work in parallel. Since interface changes ripple through teams, knowing before development starts what the interfaces will be is important.

For smaller projects iterative approaches are very effective. The “bring me a rock” game plays out to some level in almost all interactions between the people wanting a system and the people building the system. The short sprints and frequent meetings between technical teams and nontechnical stakeholders seen in most agile practices are about shortening the time spent in this cycle.

LOST started in the waterfall style. This allowed for some decoupling to order the development activities. Unfortunately, this was a huge mistake on my part since it removes a bunch of context from what is being built. The pieces are separable but why we want them and what they are intended to enable is too opaque. Conjoined with limited tool experience and experience working on related systems this has been a killer. The size/difficulty of the components has also been a huge problem.

I think the way forward may be to change life cycle models. We’ll finish assignment 5, capturing all of the existing work in a tagged branch. Then we restart development from a minimum viable product that we add features to for each of the remaining weeks. I think moving to the iterative model will also give us better traction and grounding for design discussions.

Iteration 1

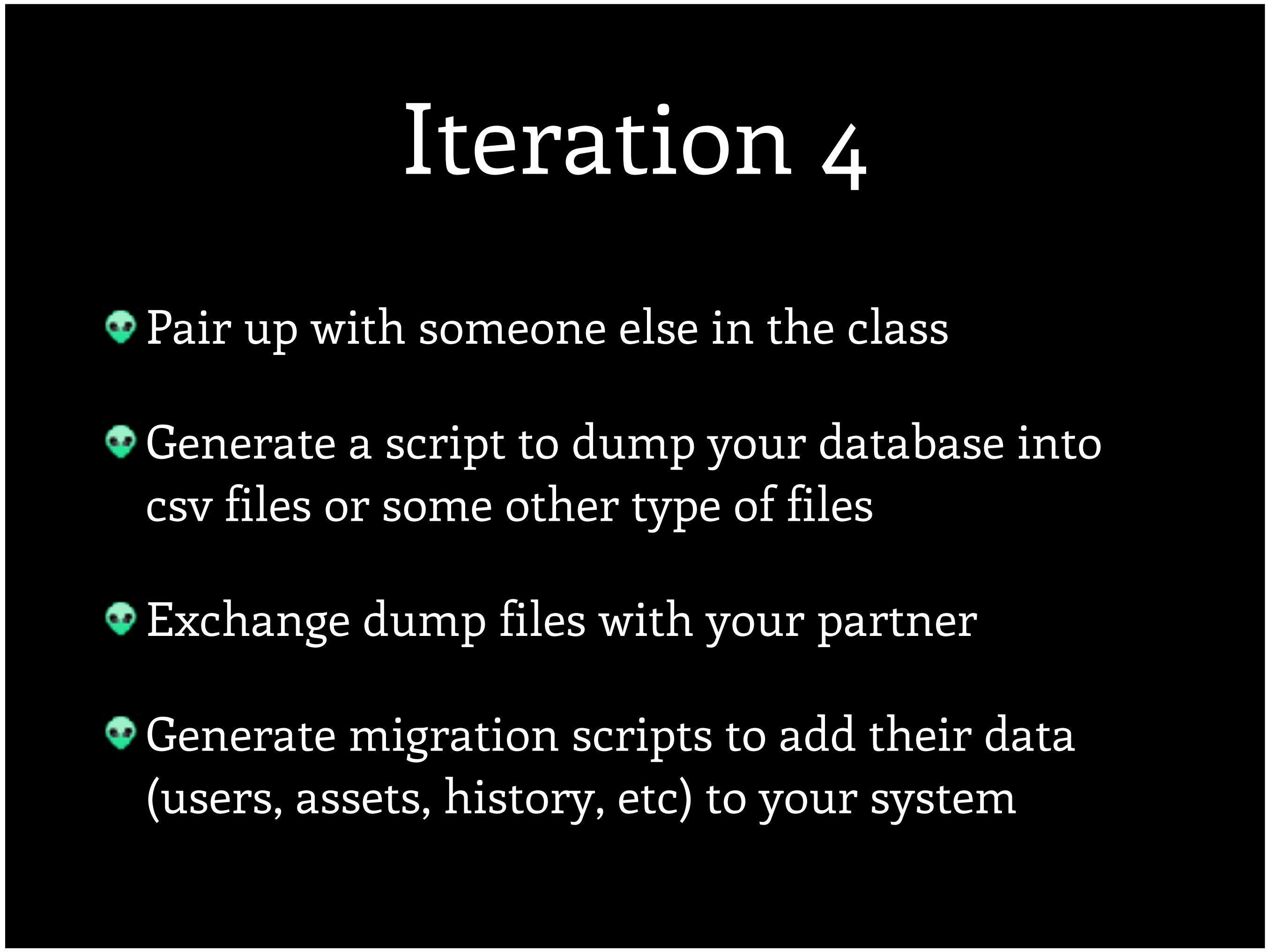
- Use git to create a branch with the current work and tag it
- Setup the master branch with a skeleton directory structure
- create_tables.sql for a simple database that handles username/password pairs
- Flask application that has a create user screen, login screen and dashboard screen
- preflight.sh script to install the tables and application

Iteration 2

- Augment create tables to add tables needed to enumerate facilities and assets as well as add roles for users
- Add setting the role to the create user screen
- Add screens to add facilities and assets to the application
- Add screen to dispose of assets
- Add screens to get reports on assets by facility and date

Iteration 3

- Augment create tables to add tables needed to track approvals and transferring assets between facilities
- Add screen for requesting a transfer, access controlled by role
- Add screen for approving a transfer, access controlled by role
- Add a report to the dashboard that shows transfers needing approval, displayed only for approvers
- Add a screen to report on asset transfer history



Iteration 4

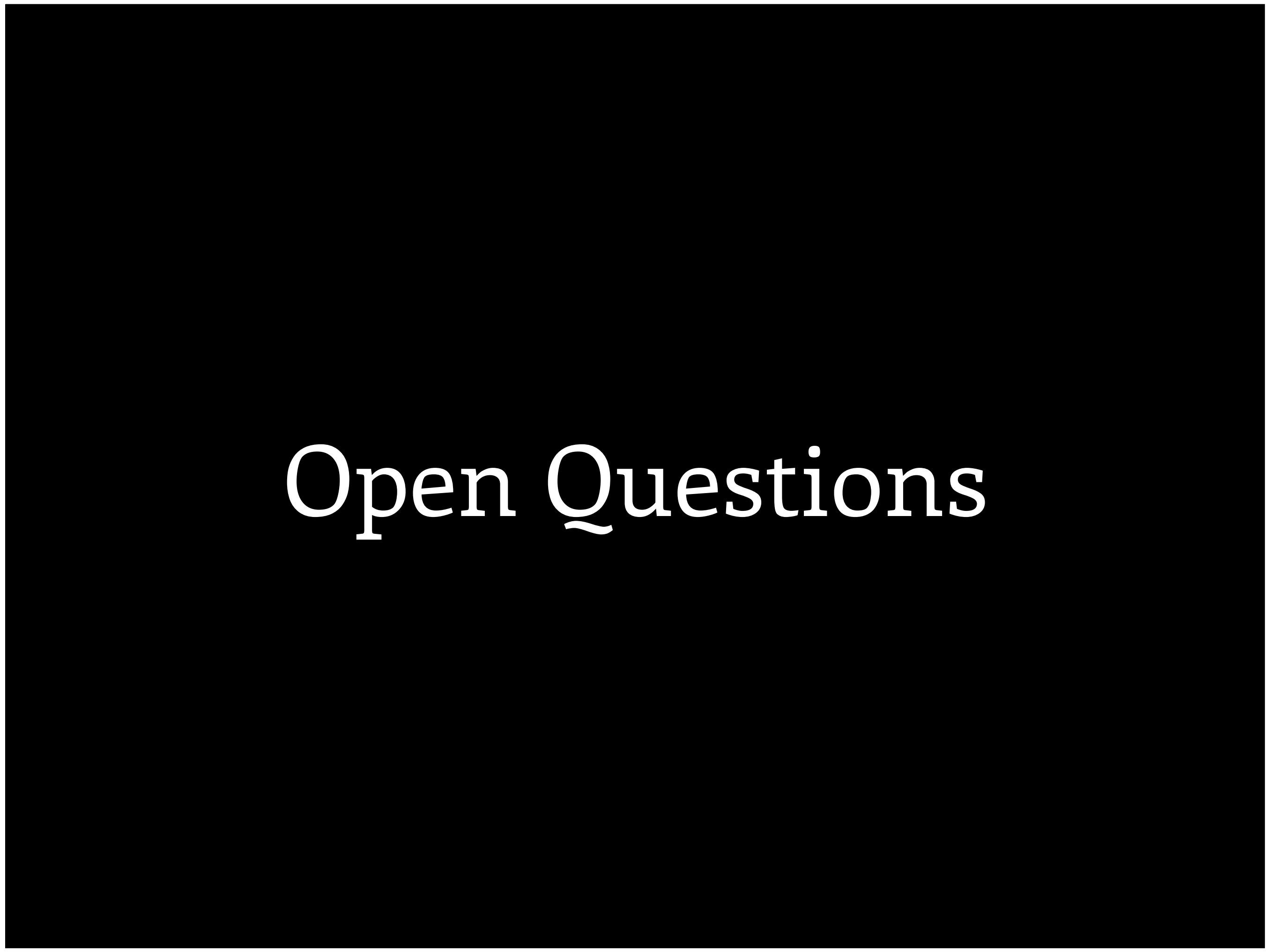
- Pair up with someone else in the class
- Generate a script to dump your database into csv files or some other type of files
- Exchange dump files with your partner
- Generate migration scripts to add their data (users, assets, history, etc) to your system

Iteration 5

- Pair up with someone else in class
- Write a test plan for your system
- Exchange and execute the test plans
- Create a web service and client to add/revoke users
- Disable the user create screen

Midterm on 2/14

- Have you had a chance to look at the practice midterm?
- Are you concerned about the questions?
- Are there things we should go over today?



Open Questions