Development Environment [15 minutes]

Server

1. Web Server *(Apache, Nginx, Node.js)*
2. Database Server *(MySQL, PostgreSQL, SQLite, MongoDB)*
3. Scripting Languages *(PHP, Python, Ruby)*
4. Support Libraries *(Mcrypt, OpenSSL, Mbstring)*
5. Access to server *(SSH, SFTP, Git)*
6. Access to database *(Database users & permissions)*

Developer

1. Text Editor / IDE *(Coda, Sublime Text, vim)*
2. File Transfer Client *(Coda, Cyberduck, WinSCP)*
3. SSH Client *(Terminal, PuTTy)*
4. Git Client *(GitHub app)*

Development Environment for this class:

1. Server: **Virtual Server**
2. Hypervizor: **VirtualBox**
3. Web Server: **Nginx**
4. Database Server: **MySQL**
5. Scripting Language: **PHP**
6. Support Libraries: **OpenSSL, PDO, Mbstring, Mcrypt, Tokenizer**
7. PHP Framework: **Laravel**
8. Access to server: **SSH**
9. Access to database: **MySQL user to local MySQL server**

Version Control [15 minutes]

1. ***Create new repo***
   Create new directory, open it and execute following command:

```
git init
```

2. ***Checkout a repo***
   You can create a local copy of a local repository:

```
git clone /path/to/repo
```

   You can also create a local copy of a remote directory:

```
git clone git@github.com:NYU-CS6015/Class-Materials.git
```

3. ***Workflow***
   Each git repo consists of three separate trees maintained by git: **working directory** that contains your code, **Index** that acts as a staging area and the **HEAD** that points to the recent commit of git repo.

   *Example: Think of mailing a package. First, you add your items to the box (**add**). Then you are putting a label (**commit**). Finally you bring the package to a shipping company, like FedEx or UPS (**push**).*

a. Add
Add specific file:

```
git add <filename>
```

Add all files within directory:

```
git add .
```

b. Commit
Commit the changes:

```
git commit -m “Commit description”
```

c. Push
To submit these changes to the remote repository:

```
git push origin master
```

4. Branching
Branches provide additional flexibility for your project by separating your features into isolated containers.

For example: You might want to work on a comments component for your blog. While your blog code is in the master branch, you might want to put comments into its own branch and later merge it after the feature has been completed and tested.

a. Create a new branch and switch to it:

```
git checkout –b <branch name>
```

b. Switch back to master branch:

```
git checkout master
```

c. Delete specific branch:

```
git branch –d <branch name>
```

d. Push branch to remote repository:

```
git push origin <branch name>
```

5. ***Update your repo***
   When you work with team members, you will need to make sure that the code in your local repo is up to date with others.

```
git pull
```

6. ***Tags***
   One of the best practices is to tag your code according to the release version. You tag your code with the following command:

```
git tag 1.0.0 <commit hash>
```

7. ***Logs***
   Logs are very useful when you are looking for history of the repository.

   a. Look up commit history of a specific person:

```
git log --author=<git author>
```

   b. Compressed log:

```
git log --pretty=oneline
```

8. ***Other useful stuff***
   a. ***Checkout specific file***
      Sometimes you might have code conflicts with specific file, and will need to revert back to previous version of that file. You can do the following:

```
git checkout -- <filename>
```

   b. ***Reset local repo***
      You might also be in a situation where you need to reset your local repo and use code from the committed code:

```
git fetch origin
git reset --hard origin/master
```

   c. ***Generating & adding SSH key to GitHub***
      i. ***OSX / Linux***
         https://help.github.com/articles/generating-an-ssh-key/#platform-mac

      ii. ***Windows***
         https://help.github.com/articles/generating-an-ssh-key/#platform-windows

   d. ***Creating GitHub repo***
      https://help.github.com/articles/create-a-repo/

e. Adding collaborators
https://help.github.com/articles/adding-collaborators-to-a-personal-repository/

f. Forking repos
https://help.github.com/articles/fork-a-repo/

Semester Project 45 minutes

Ideas
Old:

1. Social Network
2. Blog
3. Web Store
4. Dashboard

1. Generate new ideas 5 minutes
2. Sign up on Google Doc with final idea. 5 minutes
3. Assign groups. 10 minutes

Pre-development Process: 25 minutes

1. Web Application Requirements
    a. Purpose
    b. Goals for users / visitors
    c. Marketing
    d. Target audience
    e. Competition / Examples
    f. Technical Requirements
        i. Hosting
        ii. Database
        iii. Security / User Authentication
2. Sitemap
3. Diagrams
    a. Web Application Architecture
        i. Server – Hardware
        ii. Server – Software
        iii. Database
        iv. Database Abstraction
        v. Framework
        vi. Backend
        vii. Frontend
    b. User Flow Diagrams
4. Database Design