WEEK-07 DAY-1
Monday - *Your GitHub Identity*

- WEEK-07 DAY-1
  Monday - *Your GitHub Identity*
- GitHub Profile and Projects Objectives
- Improving Your Profile Using GitHub
  - Wikis (pre-project)
  - README files (post-project)
  - Wrap up
- Your GitHub Identity

GitHub Profile and Projects Objectives

GitHub is a powerful platform that hiring managers and other developers can use to see how you create software.

- You will be able to participate in the social aspects of GitHub by starring repositories, following other developers, and reviewing your followers
- You will be able to use Markdown to write code snippets in your README files
- You will craft your GitHub profile and contribute throughout the course by keeping your "gardens green"
- You will be able to identify the basics of a good Wiki entries for proposals and minimum viable products
- You will be able to identify the basics of a good project README that includes technologies at the top, images, descriptions and code snippets

Improving Your Profile Using GitHub

By now you are likely familiar with certain aspects of GitHub. You know how to create repos and add and commit code, but there is much, much more that GitHub can do.

GitHub is an online community of software engineers - a place where we not only house our code, but share ideas, express feedback, gain inspiration, and present ourselves as competent, qualified software engineers. Yes, this is a place to manage version control and collaborate on projects, but in this module we are going to discuss how to harness the power of GitHub to your advantage.

Aside from your actual code repositories, there are several other sections that represent who you are as a developer.

- **Followers and Following**: Think of this as your "friends' list on GitHub. If you see a social media profile of a person with two friends, what are you going to think? Though you're not expected to have hundreds of followers on GitHub, you should follow your peers and encourage them to follow you to show industry engineers that there are people who would vouch for your skillset.
- **Stars**: These are the "likes" on your GitHub profile. Similar to encouraging others to follow you and following them, you should also "star" their popular repositories. By the end of the curriculum, you should have several stars on each of your portfolio projects.
- **Green Gardens**: This is birds-eye view of your "newsfeed" if your newsfeed reflected the number of commits that you were making. If you are very active on GitHub, your activity squares, AKA "green gardens" will be verdant and green. This tells anyone who views your profile that you are actively coding and engaged with your identity as a software engineer. However, if your "green gardens" are not green, this suggests that you are not actively coding and therefore perhaps not the best candidates for a development job.
- **Photo and brief intro**: This is fairly straight-forward and should mirror the photo from you LinkedIn. Choose a professional photo and brief description that highlights the fact that you are a software engineer.

Your followers, stars, green gardens, and photo represent who you are as a developer and should be constantly monitored and updated. It can be easy to fall into the trap of not pushing commits to GitHub or keeping your commits on your local machine - be careful to avoid these tendencies because they are counter-productive to your candidacy as a software engineer.

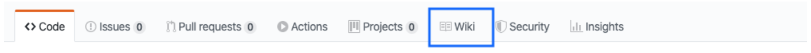
Not only is GitHub as representation of you as a developer, but it is also a place to present proposals, recaps, and provide additional project context. Two features that you will soon become familiar with are Wikis and READMEs.

Wikis (pre-project)

Wikis are features of PUBLIC repositories on GitHub and are where your design documents, explanation of technologies used and insight into what your repo contains will live.

Wikis are created at the beginning of any significant project and should be updated as your project evolves.

To create or update your repository's Wiki, click on the "Wiki" tab in your repo header and click "Edit" to update your home page and "New Page" to add a new section.



Best practices for Wiki creation:

- List of technologies in a visually accessible place
- Separate design documents into their own sections
- Write clearly and concisely - grammar and spelling matters!

Design documents will become very important when you begin your projects, but for now just know that you will spend time creating a solid Wiki in order to facilitate a smooth development process.

One of the most important aspects of your Wiki will be the outline of your project's features and the **minimum viable product** (MVP).

As you begin any project, you will consider what constitutes a "final product" and break down features into bite-sized "minimum viable products." For example, if you intend to create an a/A version of Twitter, your list of MVPs may include:

- Users are able to log into application
- Users are able to create, edit, and delete tweets
- Users are able to follow other users and see their tweets
- Users are able to like other users tweets
- Users are able to comment on other users tweets

In order for your project to stay on track, it is very important to break down your features and tackle one MVP at a time. All of the MVPs combined result in a final application.

Your MVP breakdown will live in your project's Wiki and is subject to evolution. As your project comes to life, your MVPs may merge, divide, or become irrelevant. Update your Wiki as things change and continue to referring to it as your go through the development process.

README files (post-project)

READMEs are text files that introduce and explain a project. Typically, READMEs are created and completed when you are ready to roll your application into production. READMEs should contain information about two impressive features that you implemented in your project, the technologies used, how to install the program, and anything else that makes you stand out as a software developer.

Think of READMEs as the "first impression" that prospective employers, colleagues, and peers will have of you as a developer. You want their first impression to be "wow, this person is thorough and this project sounds interesting," not "oh no, typos, missing instructions, and snores-galore."

When it is time to create your README, you should allocate about three hours to guarantee you have enough time to make your project shine.

README.md files are written using markdown syntax (.md) which makes them appear nicely on-screen. Markdown is a lightweight markup language with plain text formatting syntax. It's a very simple language used to create beautiful and presentable README and Wiki files for GitHub. There are many good resources out there for creating markdown documents, but here are two of our favorite:

- GitHub's guide to Mastering Markdown
- Repository with a collection of examples
- Browser side-by-side markdown and on-screen program (this is a favorite, code here and copy markdown into GitHub).

README Best Practices

- Divide your README into distinct sections
- List the technologies used at the top of your README for increased visibility
- Include nice pictures or Gifs to show and/or demonstrate how things work
- Include code snippets
- Provide instructions for how to install project (if applicable)
- Include link to the live site

Wrap up

The bottom line is that the way you represent yourself on GitHub matters! Take the time you need to write clearly, accurately reflect your process and applications, and immerse yourself in the diverse and interesting pool of software professionals who work and play on GitHub.

Your GitHub Identity

It is hard to write about yourself. But, today, you need to do that. This is a day of starting to establish how other software developers and hiring managers will perceive you.

Go to your GitHub profile page. Edit your profile to contain your description, "App Academy (@appacademy)" as your current company, your location (if you desire), and your Web site.

Now, make a personal Web site for your GitHub profile. You can do that using GitHub Pages. Follow the instructions at Getting Started with GitHub Pages to create your site, add a theme, create a custom 404, and use HTTPS (if you want).

Spend time writing about yourself. Like you read earlier, this is hard. But, tell the story of you in a way that will engage people.

Now, go follow all of your class mates and star their personal Web site repository, if they created one.

If you want to get really fancy and set up a blog, you can use a "static site generator" known as **Jekyll** to do that. It's a Ruby-based program; however, you don't need to know Ruby to use it. All you have to be able to do is use command line programs, something you're really getting to be a pro at! To do this, follow the well-documented instructions at Setting up a GitHub Pages site with Jekyll.