Monday Deliverables

Our goal for today will be to form teams and come up with a plan for the product development period. We will start by pitching projects. Review the Deliverables section each day for the day's expectations.

Deliverables

- Product Ideation and Business Profile Workshop for application ideas (see next lesson)
- Pitch project ideas and form teams
- Review daily deliverables for the Product Development period (see content below)
- Establish team roles (see content below)
- Define project scope (see content below)
- Define MVP with user stories (see content below)
- Establish git workflow (see content below)
- Sprint planning session (see content below)

Team Formation

After you and your classmates have pitched your ideas, choose an idea that sounds interesting, and form teams. Teams may have a minimum of 2 people and a maximum of 4.

Daily Deliverables

Daily Team Stand Up: We will begin every day with a team stand up. In class, our stand up meetings served as a status check, where we communicated information about the coursework and any other announcements we have. Announcements were open for discussion.

During the project period, we will run our stand up meetings differently. We will have a stand up meeting every morning with our team, where we will share our priorities for the day, as well as anything getting in the way of our work. Stand up should be short, maybe 5-10 minutes. This time is not meant to be discussion, but rather, the structure of the meeting is meant to promote follow-up conversation, as well as to identify issues before they become too problematic. If there is anything you would like to discuss further, do it after stand up.

For the daily team stand up, go to a location as a team and stand up in a circle. Go around and have each team member answer the following questions:

- What did I accomplish yesterday?
- What are my objectives for they?
- Which blockers am I facing? Who or what is blocking me?

To get used to this format, it may be helpful for the person speaking to hold an item, and then pass it to the next person when finished.

Lunch break: This is mandatory. It acts as a health check point: stretch your legs and socialize with food and human beings.

Merging: This will be done everyday at 5pm local time making it easier to check for errors and merge conflicts, which will help the team prioritize objectives for the following day.

Deployment: After merging your work, deploy the project and make sure it's running well. Give the project to a fellow classmate from a different team use it and give constructive feedback.

Remember to check the Moringa Core Google calendar for the day's schedule as well.

Establish Team Roles

Task:

To stay organized, assign each member of the team at least one role.

- **Product owner (one person):** The person in charge of the whole project, has the bigger picture in mind and will be ultimately responsible for the success and failure of the team.
- **Scrum Master (one person):** Makes sure that the team works together to achieve the daily objectives, in charge of running stand-ups and making sure the project is on track. They will remind and steer the team to different activities on the slotted time.
- **Software Engineer:** Everyone in the team will take this title.

Define the Project Scope

Before we start thinking about the product we would like to build, it is important to define the project scope. Understanding the scope lays the foundation for managing any project changes down the line.

To define a project scope, we must first identify the following things:

- **Goals**: What is this project trying to do? Goals are the long-term, bigger picture items that set the tone and direction for our work.
- **Project objectives**: What are the specific steps we need to take to reach our goals?
- **Resources**: What do we have available to us? What strengths, skills, knowledge, mentors can we use?
- **Budget**: What are our resource constraints? Budget can mean a lot of things, and often implies financial constraints. For this project, we need to consider how to budget our time.
  - It may be tempting to learn a fancy new framework to develop this project. But doing so would take valuable time away from developing the actual app.
- **Schedule**: How are we going to manage our time?

Task:

Take some time as a group (minimum 30-45 minutes) to outline your project scope using the guidelines above.

The Minimum Viable Product (MVP)

Now that we have defined our project scope, we can start thinking about what we are going to create. A great place to start is the minimum viable product, or MVP. As the name implies, this is the *minimum* or *most basic* version of an application that fulfills its goals. That is, the problem the application is meant to solve, or the service it is supposed to offer to the user.

Let's look at an example of an MVP. Consider the following example from Gerry Clapp's response to the question **"What is a Minimum Viable Product?" on Quora (https://www.quora.com/What-is-a-minimum-viable-product)** :

Let's pretend you're building a startup with the goal of creating the ***best* donut ever**.

The product team starts off by building a plain donut. At this point it's considered an MVP. The product works, but it's probably not quite the *best* donut product out there. Now the team can ask their customers questions about the donut, like:

- What do like the most about the donut?
- If you could choose any topping, what topping would you add?
- Would you prefer a donut in a different shape?
- *And, so on*.

Using this newfound *validated learning from their customers*, the team can create a better donut. But, depending on the context of the customers that provided feedback, the team can have wildly varying results:

- In this particular case, it's to add candy sprinkles.
- In a different market, with different customers, those customers may [have] wanted a chocolate donut.
- If the team spoke to customers in another country, they may [have] wanted a strawberry donut.

From this example, we can see that creating an MVP has some benefits:

- **We complete a functional product sooner**. While it may not be the *best donut ever* or have all the features we would like, it is better to prioritize building a functioning prototype with fewer features than to attempt adding too many features and fail to produce a working project by the deadline. You can always add extra features after the MVP is finished.
- **We can collect user feedback before adding extra features**. After creating the MVP, we can beta-test it with sample users, and their feedback to understand what kinds of features they actually want. They may have thought of something you hadn't considered!

Define User Stories

A technique we will use to develop our MVP is defining **user stories**. After all, our users are the reason we are creating an app in the first place!

We've seen user stories before. In fact, many of our project requirements at Moringa School are structured in this way: **Food Tracker (http://moringacore-js.herokuapp.com/#20.html)** , **Craigslist (http://moringacore-js.herokuapp.com/#30.html)** , **Online Marketplace (http://moringacore-js.herokuapp.com/#31.html)** , to name a few.

A user story is a tool used in Agile software development to capture a description of the functional purposes of an app from an end-user perspective, in non-technical language. They are helpful in both guiding the product development, and for discussion of the the app with clients.

User stories often use this structure:

As a < *type of user* >, ***I want*** < *some goal* > ***so that*** < *some reason* >.

Here are what some user stories for a To Do app might look like:

- As a user, I want to add tasks to a list.
- As a user, I want to mark tasks as completed as I finish them.
- As a user, I want to see a welcome page that includes where I can go and what I can do.

- As a user, I want to see all of the lists that I have created so that I can manage them one at a time.
- As a user, I want to create new lists of different categories so that I can keep similar tasks together (phone calls, school work, house work, errands to run, bills to pay, etc)
- As a user, I want to select a single list and see the tasks for it.

Task:

As a team, spend 30-45 minutes and create at least 10 user stories to define the functionality of your MVP.

Establish Git Workflow

A consistent and established Git workflow is crucial to the success of any team project.

Task:

In order to stay organized, review the Git workflow outlined here:

- Main repository: You have a main repository for your project that everyone in the team works from through a fork + pull request workflow. Further explanation on how to do this can be found **here (http://blog.scottlowe.org/2015/01/27/using-fork-branch-git-workflow/)** .
- GitHub Housekeeping: At 5pm every day the whole team merges work and resolve any merge conflicts. Commit messages should **always** be short and precise, yet detailed enough for the rest of the team (and potential employers) to understand. This must be done in order to reduce the risk of having a project that is not working on the presentation day.
  - **Warning: Attempting to merge all of the team's code together on the last day has been the reason why several groups have failed to have complete projects.**
- Feature Branch: Used for project features that the app is expected to have. Further explanation on how to use it can be found here **(https://gist.github.com/forest/19fc774dde34f77e2540)** .
- Collaborators: The repository on GitHub will be made on one of the team members accounts and the rest of the team members will have collaborators access, which allows push, pull, and clone.

Product Management

Earlier, we mentioned that user stories are an Agile software development tool. But what is Agile, anyway?

Agile is a software development approach that is iterative and team-based . This approach emphasizes the quick delivery of an application in complete, functional components. Rather than organizing by tasks and schedules, all time is time-boxed into phases called sprints.

Each sprint has a defined duration with a running list of deliverables. The team defines deliverables at the start of the sprint. We rank deliverables by business value as determined by the customer. If all planned work for the sprint cannot be completed, work is reprioritized and the information is used for future sprint planning.

As work is completed, it can be reviewed and evaluated by the project team and customer, through daily builds and end-of-sprint demos. Agile relies on a very high level of customer involvement throughout the project, but especially during these reviews.

Note: You may remember talking about Scrum in class. Scrum is an implementation of Agile specific to software development. We've actually been using these principles throughout Moringa! There's nothing particularly new here, just the terminology.

Sprint Planning

In the world of Agile software development, a sprint is a set period of time with a running list of deliverables. The team defines these deliverables at the start of the sprint.

A sprint can be any length of time that makes sense for a project. At Moringa, we often worked in two-day sprints, and our deliverables were the lessons' objectives. In the workplace, it's more likely that you will experience sprints closer to a month long, with deliverables like creating features and finishing projects.

Let's think about our product development period at a glance. We have three weeks to complete a project, and will be presenting the status of our projects every Friday. It makes sense to divide this time into three week-long sprints (with the exception of this first sprint, which will run from now until Friday). Throughout the project period we will conduct market research to evaluate our product, so the planning period at the beginning of each sprint will be essential in helping the team re-evaluate and respond to any changes that come up. That way, we make sure that we have a functional product at the end of it all.

Task:

It's time to hold your first sprint planning session as a team. Spend half an hour and come up with deliverables with specific deadlines. Each deliverable needs to be assigned to a team member. As a reminder, by end of day Friday, we need well-defined user stories for our MVP, as well as the basic design of the app. We also need to have completed first iteration of user experience testing. How should we incorporate these deadlines into our deliverables?