How to Plan

Introduction

Whenever you start a new project it is tempting to just jump into the coding. A lot of smaller projects won't encounter any issues with this, but larger projects will. The aim of this document and lesson is to help give you the tools to plan for larger personal projects. The focus is on large projects done by you alone, but the methods and processes discussed here would also be applicable in your work and professional environments. Let's get started!

Project workflows

Let's start by having a look at professional project workflows. In the beginning of this course at Lifechoices you were introduced to some project management methodologies. In this section we'll dive a bit deeper into the workflows and processes of planning done with the agile project management methodology.

As you might recall we discussed that professional projects have people on-board who don't just code. You have project managers who do most of the planning, time management, user stories, and programme workflows (how and where the user goes). The other professional we'll look at is UI/UX designers. These two are distinct practices but mesh together quite often. UI designers design the look and feel of the site/programme. They choose colours, fonts, design elements, and so on. UX designers work on developing and refining the user's experience of the site/programme. They focus on where the user clicks, where buttons are, how accessible parts of the programme is, and similar aspects.

Since your personal projects is just you, it's important to pick up some skills of all of these professional practices. You need to design your own projects, manage the time yourself, and plan the project out on your own. In a professional environment, you will **not** do this alone, if at all. It's useful to have these skills so that you aren't entirely blind to these processes, but this is not your focus.

A professional timeline

Let's have a look a typical professional timeline using the agile methodology.

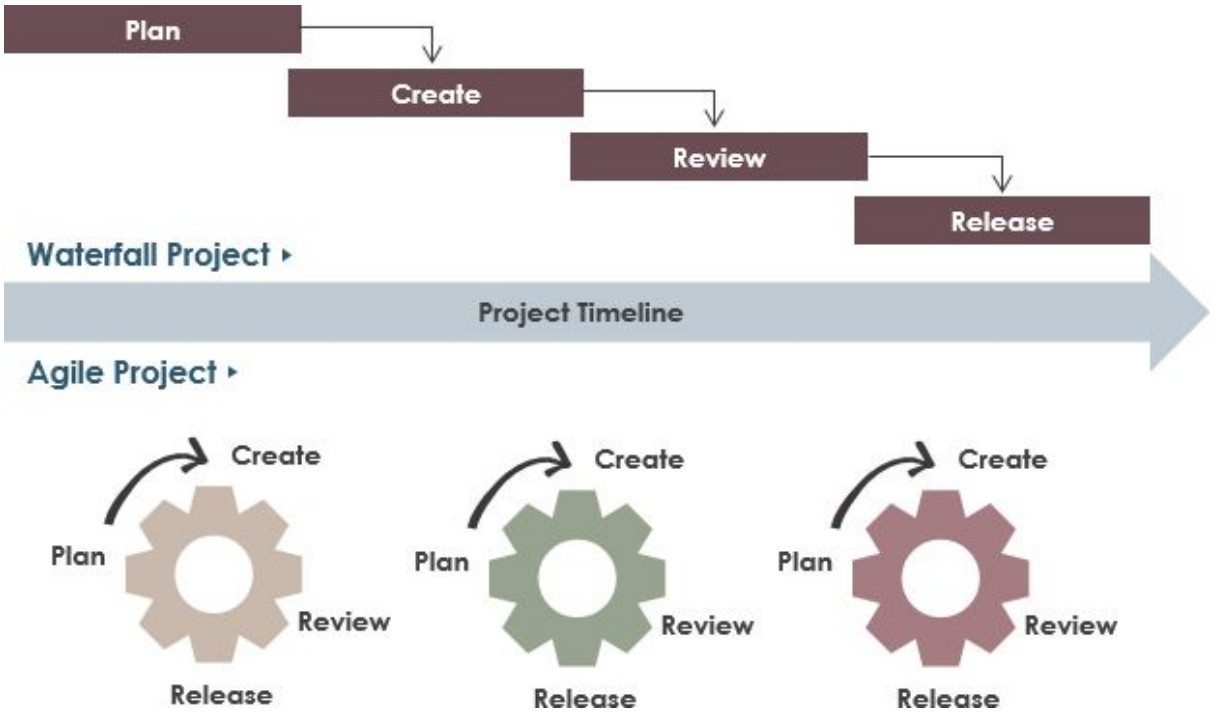

Diagram A: showcases the different workflows and timelines of waterfall and agile respectively.

As you can see with Diagram A, both methodologies have a planning phase. Your projects would look more similar to a waterfall project, but the essence is the same.

Plan -> Create -> Review -> Release

Planning is crucial for success of any project. You can ask any freelance developer, without a thorough planning phase, the project is bound to have hickups.

So let's start to look at *how* you plan.

Planning

Let's have a look at the different stages to planning. Just like the overall project lifecycle, and even coding cycles, there are phases to planning.

The phases we'll look at are:

- ➢ Timeline planning
- ➢ Feature Design
- ➢ Code Design
- ➢ Visual Design

Timeline Planning

This step can be done both at the beginning, and end of the planning phase. Firstly you'd want to set up a rough timeline. When will planning be done? When will your first phase of code be done? When will testing be done? This could be more accurately described as the project timeline.

At the end of the planning phase you'd want to start looking at feature timelines. This will be when will feature A, B, and C be coded, tested, and rolled out.

So let's look at the project timeline.

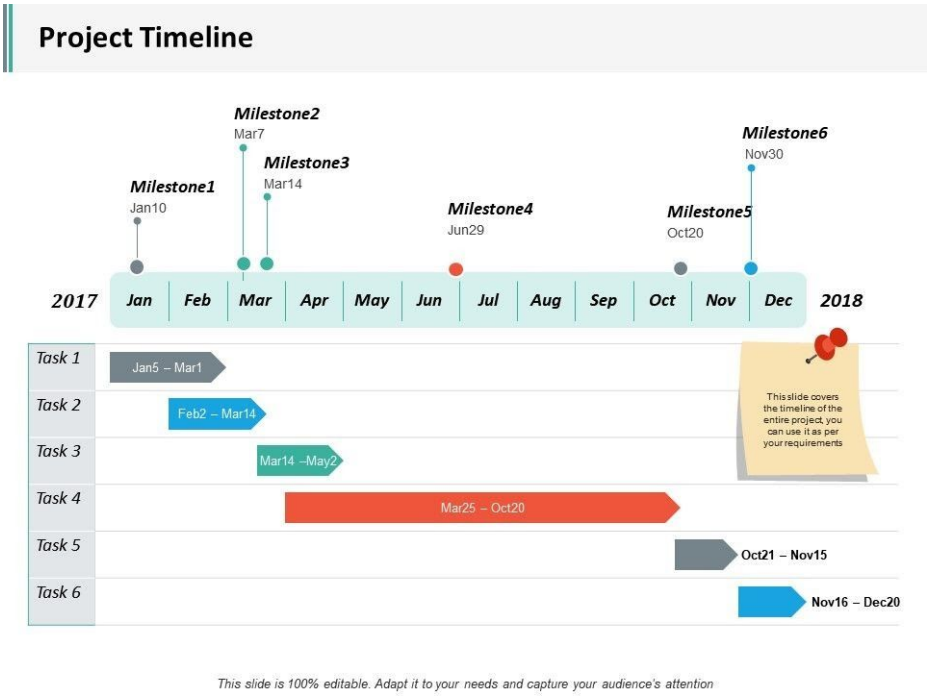

Diagram B: A visual representation of a project timeline.

The first step in your project timeline is defining milestones, their deadlines, and then the project due date. Milestones are large objectives to complete by a certain point in time. An example of a milestone is, having the database designed and coded, or example, having the UI of the programme finished. These large milestones will help keep your project on track and focused. In this step you'd also want to set aside the timeline for planning too. Once you have your milestones defines, you will have a better idea of how long each step should take.

The second step in your project timeline is actually moving on to defining the detail of those larger deadlines. Since you now have the milestones, and their deadlines, you can dig into the details of those. In this step you can start adding userflows, and coding designs. Don't focus on a visual design here yet.

The third step in your project timeline is setting the times (and sometime deligation) of the smaller aspects of the larger milestones. Example, here you can break the UI of the programme into different screens, and their individual deadlines.

The fourth step in your project timeline is setting aside time for your testing, bug fixing, and deployment. This is your final step in timeline planning since now you should have the larger milestone deadlines, and the tasks within these milestones broken down with their own deadlines.

As you can see the planning is an integrated, on-going process. Especially in agile development, your project manager would do sprint planning sessions, and more refined task planning. For your purposes, you can just revisit your timeline plan and adjust where it needs adjustment.

Feature Design

Feature design is the process of breaking down your project idea into different features. In this section you'd want to create milestones, and define the smaller features in these milestones.

You will take your project idea, example an ecommerce website and break it into milestones.

Example:

- Milestone A: Database designed and implemented
- Milestone B: Backend server designed and implemented
- Milestone C: Frontend website built
- Milestone D: Frontend features linked with the backend server
- Milestone E: Project tested, and rolled out

As you can see you can break your project into large sections. You don't have the design or code for your database, or server, but you can already have a rough idea of what needs to be done.

So now you want to focus on the actual features of your site. You would want to focus on the milestones, break them down, and add the features where you need. This process is an on-going and fluid process. You might jump between server and frontend features.

Let's look at Milestone C as an example of defining the features.

Milestone C features:

- User can login
- User can browse the product list, no login required
- User can add a product to their basket, no login required
- User can checkout their basket, login required
- User can pay for their items, login required

- ❖ User can track their items' shipping, login required

So now you have the overall frontend features you want to implement. This list might not be exhaustive at the beginning of your planning but you might come back to this planning and update it. As you can see these features will span more than one milestone. So now we can go ahead and update the other milestones with these features.

Milestone B:

- ❖ Must handle a login request from the frontend
- ❖ Must handle an anonymous user browsing products
- ❖ etc...

Now you since you have the individual features, along with the overarching milestones, you can look at your code design.

Code Design

At this stage we should have finished our timeline planning, our milestone planning, and our feature planning.

This is a more technical design phase as you will start looking at code implementations, without coding. Here tools such as flowcharts and pseudocode would come massively in handy.

You would want to look at every feature on their own, and design them out.

Example Milestone B, login request:

Here you would start to add technical code flows. So you will look at where the data comes in, and what should handle the data. I would strongly urge you to use flowcharts or code stories here. These will allow for a visual representation of where the data goes, instead of jumping into pseudocode, since pseudocode is only helpful with technical implementations, but we want to look at the dataflow, and then the code we want to build around those.

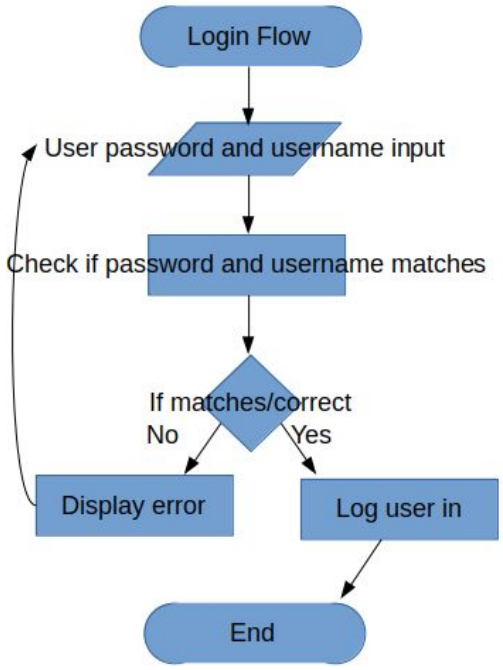

Diagram C: A practical example of how a flowchart of a login process

As you can see above is a practical example of how you might set up a login process flowchart. You can add lots more detail such as adding redirection processes, error displaying, etc. You might even want to add broad flowcharts, such as which screen the user is on, and what they do there, and direct those to other, more detailed flowcharts. Now you are equipped to start designing functions and classes around these designs.

You would handle databases a bit differently, here you would draw out the different tables and their different fields. You can add as much data here as you would feel useful. Example you can plan out which fields connect, what datatypes they are, how large they are, are they nullable or unique, etc. You can then draw lines between the database tables to show table relations. This will help greatly with a visual representation of the database.

You can use a similar process with functions and classes. Here you can draw them out and add rough processes. We'll look at an example of a login function. (this could also be a method on a class)

<u>Login Function:</u>

- Fetch user login data from request
- Check if user's username exists in the database
- (if yes) Check if the password they gave matched with the username's password in the database
- (if yes) Log user in by creating a login web token
- (if either is no) Return the user to the login screen with an appropriately descriptive error

As you can see above, we did no coding, but we have a good idea of what we need to do in this login function.

Visual Design

The visual design of the project is just as important for commercial projects as the experience and useability. Here a professional project would call in designers who would work closely with the project managers and stakeholders of the project to come up with a visual design, and a visual reference sheet that everyone is happy with.

In your own projects you don't have the luxury of having a visual designer so you will need to do this yourself.

In this phase you need to start setting up visual references, such as fonts, colours, design patterns (which type of button you use, material design, or flat design, etc) and where you would use these different colours and fonts.

When working on your visual design skills, it's often good to look at successful sites, and read up on different design strategies.

Conclusion

As you can see, the planning process tends to be an on-going process, with the bulk of it done at the beginning of a project. All of these different planning phases usually tend to mesh together, and are not done on their own.

In your professional work cycles, you wouldn't normally touch too much on these planning phases, except for the feature and code design, but you will definitely need to have these skills for larger personal projects.

Here are some concluding tips to help you keep on track:

- ➢ Always keep your timeline updated! And keep referencing it to make sure you're on time.
- ➢ Don't spend most of your time planning. Some coded features can just be done without going through a code planning phase if you know how to implement it. Keep your available time in mind.
- ➢ Keep your planning organized! A messy planning document, and messy organization of these documents will lead to a loss of time, and probably also a headache. So keep your planning documents organized, and make sure you can find them when you need them.