Project Approach

Please don’t panic!! We know how to do it. But we just need to take care of additional factors before starting a project.

What are those additional factors? Let's list them out:

1. Scalability
2. Designs
3. Performance
4. Coding Standards
5. GIT/ CI/ CD
6. More Interaction

These are the few points which I think we always need to remember. But if you have any other point let’s add it and discuss it.
So now i will go one by one on each point, why these all points are important before starting Angular Project.

As we all know Angular is a big framework in itself. It has more capabilities which we still need to know and practice. But we can study the basic ones and can start our work. But always remember that less knowledge always ends up in spider webs. So we must study these points:

Scalability

This is the main factor before starting any project in Angular. You must get the idea of scalability of your project.
How do you get it?
You must participate in the team meetings or have a good discussion with your manager. Ask about business logics they want to include in the project. After that take a break and think full flow according to that logic or discussions.From this flow you can get the idea of how big or small an application you can create. And that will decide the scalability of the project.
Now you must ask, if I have decided that my application will be big or small. So what should i do from that scalability.So!!

What do I do with that scalability?
Now when you have the scale factor for your project you can decide following things:

1. Monorepo Concept
2. Modules Management/ Sharing Concept

Monorepo Concept: This is the concept in which projects have separate websites/ panels. Eq. If you are developing a delivery application project. Then that project will have following web applications:

1. Website
2. Customer Panel
3. Merchant Panel
4. Driver Panel
5. Super Admin Panel

So if you will go with creation of individual Angular Applications then you will end up with 5. But with monorepo you can create one Angular Project and you can handle all the 5 webs in 1.
You can study monorepo concept and set up by using this: Click Here
Modules Management/ Sharing Concept: This is the main part where you have to decide how many and which modules I have to create in my project. If you would understand the requirements from discussion properly then you will be able to do this step also.
This is the main point where your past experience of projects also works. Because you will get to know about the sharing modules, services, pipes etc.

Designs

This is the more precious part of web development. Design is the key factor for users who are interacting with our websites. We call it UI/UX.
Why are designs important?
If we have good designs of our web application then the following things will happen:

1. Easy to understand each part.
2. Users will spend more time on our application.
3. Users will get exactly what they want.
4. With less clicks users can reach easily to end.

If we don't have good designs users will do and get vice versa of above points.

How can we plan for good designs?
So for this we have UI/UX developers. Who develop each screen before we start developing. These developers provide us the designs on following platforms:

1. Zeplin: Click here
2. Invisions: Click here
3. Mockups: Click here

Sometimes these UI/UX developers are not available in project then you must take reference from these:

1. Any similar project in the company.
2. Other online applications available in the market.
3. Online themes available which you can choose from free/ paid.

But for the design you must have good knowledge of HTML5 and CSS3. Which you can learn basics from following:
HTML5: Click here
CSS3: Click here
What extra do we need from UI/UX?
This is the section where we have to ask from UI/UX developers some extra things in advance. This is actually one thing but it has more than enough for web applications.So the thing is ***Design Guidelines.*** This is the most important sheet for all frontend developers. Because it will contain the following items:

1. Color codes which we are going to use.
2. Button styles to follow.
3. Font family we have to use.
4. Box Shadows we have to give.
5. Card view.
6. List view.
7. Tabs view etc.

So many more we can define which is required in the project. These all types of guidelines will be provided in the ***Design Guidelines Sheet.***
Bootstrap design guidelines link: Click here
Why would I care about these design guidelines?
If you will not care for these guidelines. Then you will suffer in future for your project. Because these guidelines will help us to create our theme structure in the project.
Now what is the theme? So the theme is basically the directory or number of files which we create from these guidelines help. So themes can be created from CSS, SCSS, SASS etc.

Theme from CSS you have to define the variables in the root directory of the CSS. Which we represent it as:

```
:root {
        --varName: value;
}
```

For using this variable in any class you can use it as:

```
.className {
        propertyName: var(--varName);
}
```

If these CSS variables are defined you can also change its value through Javascript. You can study: Click here

When you run it in the browser then these *:root* variables will get defined in the *<html>* tag. There you can see all the listed variables of *:root*.

Theme from SCSS/ SASS. For this you have to be a good knowledge of that which you can learn it from here: Click here

So in SCSS there are many things with which you can create your own process of theme. Mostly commonly used things are:

1. Variables with *$* sign
2. Mixins with @*mixin*
3. Extend the class @*extend*
4. Include mixin functions @*include*

Out of these you can also write *if-else, foreach, for, while and functions* to create the theme.(We will create a mini project to understand the theme better.)

The above two are those which we have to develop from scratch. But in the market there are also auto generated themes available: Bootstrap, Material etc. But these all are also only to help us by using predefined classes and components like table, tab, grid, carousel etc. To shape these classes and components into our design guidelines you must know how to override the default values defined in these themes with our guidelines.

Customize Bootstrap Theme: Click here

Customize Material Theme: Link-1, Link-2

Performance

Performance is the main factor to run the business through web applications in the market. If you have a poor performance website, the user has to wait for a long time and that will end up in trash.
Refer this document to understand better. Click Here.
Performance can be improved by following:

1. Use CDN links.
2. Minify and compress your CSS and javascript files. Eq. Webpack, Grunt, Gulp, Babel etc.
3. Optimize the image size by keeping quality the same. Use SVG for more or JPEG.
4. Reduce the number of plugins we are using in the project.
5. Use font family only with extension WOFF, WOFF2. These have less size.
6. Reduce the number of redirections while visiting any pages.
7. Prefetch Technique.
8. GZIP compression on the hosting side.
9. Implement cache on the hosting side.
10. Use HTTP.2 protocol for fetching and sending.
11. Use visual representation of elements through *aria, title, alt, id.*Study this from here.

You can study about these points anywhere on the internet. But you must follow some coding fundamentals which will also not cause performance problems. So we will discuss here for Angular.
How can we increase performance through Angular?
So there are various methods available. We will list it out:

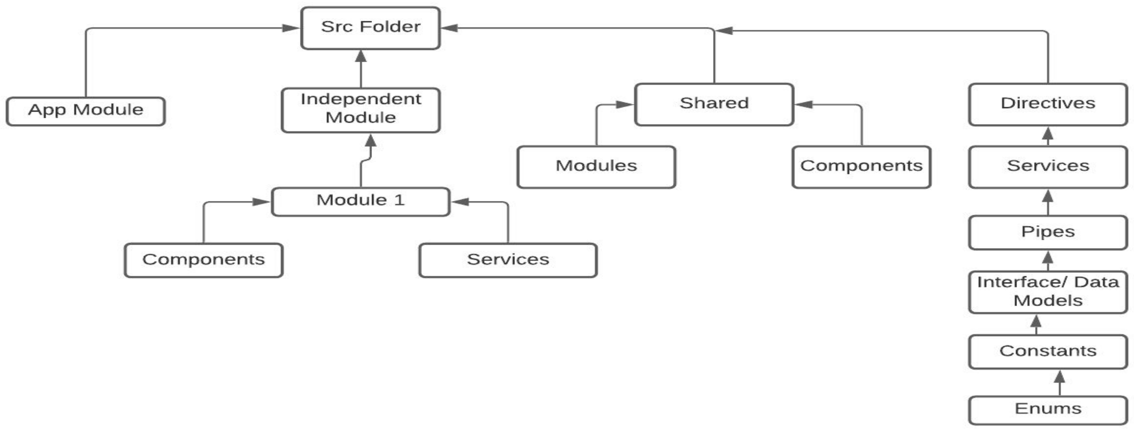
1. Break every module into minimal small components.
   For eq. In Header Module we can create these small components:
   a. Brand Logo
   b. Search Component for searching globally
   c. Profile options component
   d. Setting options component
   e. Notifications Component etc.
2. Don't create duplicate modules and components.For learning refer these links: Component, Module

3. Lazy Loading Module through routing. This concept will break your each page into separate modules. And it will get called when you navigate to it. Please refer this link: Lazy Load
4. Preloading Strategy in routing. This will help us prefetch that module before actually going into it. Please refer this link: Preload
5. Unsubscribe every event, Observable before leaving that component.
6. Never add every component in the App Module. Make it more modular and create independent ones.
7. For change detections never use methods in HTML. Use Pipes instead of it. Please refer this link: Pipes
8. Use enableProd for production builds.
9. Use AOT (Ahead of Time) configuration for any environment build. Refer this link: AOT
10. Use trackBy always with ngFor. It will stop re rendering of already rendered items.
11. Server Side Rendering is also a technique for improving. Refer link: SSR.
12. Make Progressive Web Apps so that it can use cache on the client side. Refer link: PWA
13. Web Workers. Need Proof of Concept for this. Refer link: WEBW

Coding Standards

As every language, library and framework has their own coding standards. But we will discuss only General and Angular ones.you can study more about those standards from this link: Click here
The following points are those which I think is mostly required for this:

1. Proper use of NG CLI. This is the best part of Angular. Because with this we don’t have to spend our time on creating each file or folder. You just have to do it through NG CLI. You can learn it from here: Click here. This will give you the ability to create proper folder structure.
2. Folder Structure: The most important is the folder structure because it helps us to navigate between folders easily. The mostly structure I use is:



3. Don't add any links or paths in Angular JSON scripts and styles array. Because they will get loaded with an App. Which will increase the time of first paint. Use CDN links and directly add it to index.html or load them where you needed using dynamic loading scripts.
4. Use lazy load modules for each routing. If you will use components in routing paths then it will get added to the App again. It will increase the main.js size after build.
5. Use Commenting & Naming convention wisley in and for the files. Files name should be camelCase or - separated. Commenting and other function names must be camel case.
   Eq. Function name should be understandable by it's functionality or you can add some comment regarding that in your code. Use this type of commenting:
   ```
   /**
     * What will function do
     * @param arguments which you are passing into function
     */
   ```
6. Use of dataTypes in code. For this you have to define each variable data type what type of data it will take. You can do this with the help of Interfaces and Models.
   For eq.
   public name: string; // string is the data type for this variable
   public userData: {name: string;email: string;phoneNumber: string};
   // this name, email and phoneNumber are the keys or data in userData.
   Similarly in function arguments: *getUserDetails(id: number) {}*

7. Logical handling of data coming from the server. For this you can use any of the following libraries: RxJs, NGRX, Redux, Services.
   According to the integration of above libraries you can decide wether you want to create Dumb or Smart Component.
8. Typescript knowledge. You must get the knowledge of some typescript concepts.
   a. Interfaces
   b. Functions
   c. Classes
   d. Enum
   e. Generics
   f. Decorators
   g. Variable Declarations
9. TS Lint this is type to define code in proper format and symbols. This can be done with TypeScript. But you must install tslint in the editor.
10. Use buildOptimizer always.
11. Use webpack bundle analyzer for checking which npm module is consuming more space. Click here
12. Using background loading of heavy images or lazy loading of images. Click here.
13. Try to integrate these according to requirements. PWA (Progressive Web Apps), SSR (Server Side Rendering).
14. Create unit testing using spec files. Need to work on it.
15. Security Checks by Angular. This is something we need when we are using cross site URLs in our project and other more scenarios also. You can study this from here.
16. Create Environment files for each environment. Like: development, testing, beta, production etc. Then also define build config for these env in Angular JSON file.

GIT/ CI/ CD

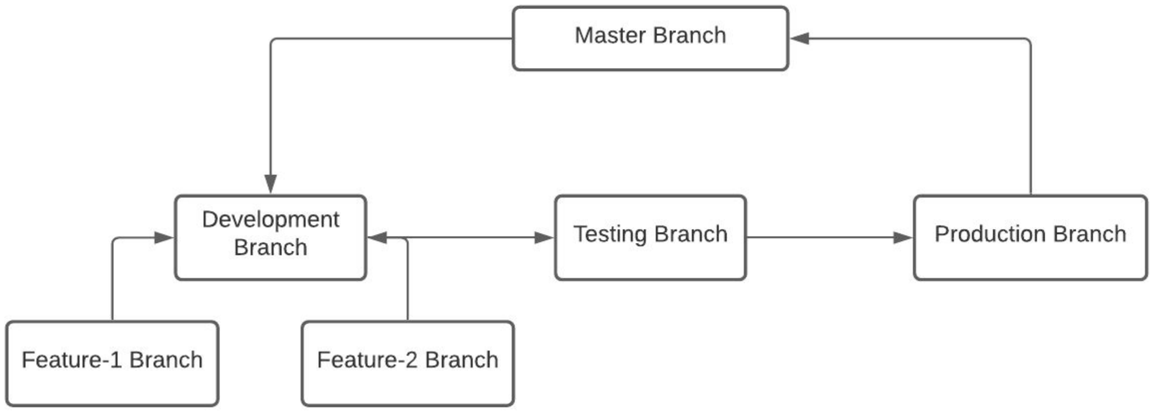
This is the part where we have to put all our hard work in a proper safe place from where we can get it in future also.This has two parts:

1. Deploying it in online platforms like: Github, Gitlab, Bitbucket etc.
2. Deploying it on an online server from where all over the world users can access it. We mostly use AWS(Apache Configuration). But there are others also.

These two setups are only accessible for development teams because in wrong hands will cause security breach and hacking.
We will discuss here only one approach which is for 1st point only. Because this will get more complex as the team size increases. So we must define some standards here:

To get an understanding of git use this link here. But the most commonly used commands are:
git init - To initialize the Git repository.
git remote origin -v: it will list out the URLs of GIT for this project.
git remote add origin "url": To add new remote repository url in project
git remote remove origin: To remove existing remote from project
git branch: It will let you know on which branch you are working
git checkout "branchName": To navigate to already existing branch
git checkout -b "branchName": To create and navigate to that branch
git status: It will show all the files you have changed, added and deleted in the project.
git add -A: To add temporary the changed files.
git commit -m"message you want": This will create a hash number to identify on git along with the message you typed.
git push origin "branchName": To push to git repository to that particular branch

Now how we will create branch structure:



Now if you will see the architecture above. We have firstly created a Master Branch then Development Branch. And from the Development branch we can create Feature branches on which our developers can work.
So we always rebase our code from Master branch to Development branch and then Feature branches also do rebase from Development branch.
For rebasing you must follow these steps:

1. If we have to rebase from Master branch to Development Branch. Fetch all latest code from git command: *git fetch origin && git reset --hard origin/branchName* by checking out those branches.
2. Now go to the Development branch and run command: *git rebase -i master.*
3. It will show conflicts according to each commit and resolve those commit wisely along with using commands: *git rebase --continue*
4. After completing conflict resolvement forces push to the Development branch.

What's the benefit of rebase?
Rebasing is the thing which will make your tree structure of commits proper and easy to understand. You can study it here
Precautions to take: This is something you need to take care. Because rebase is the thing for which you need to know the changes happening in code. Sometimes this will cause removal of code and causing problems in future. Please understand it properly.
As rebase is very good to have but comes with risk. Do this along with your seniors in the project.

After rebasing you will get the latest code on your branch and you can start the development. And now before merging directly into any branch do rebase. And then squash your commits into one commit and then merge.

As this merging and squashing you can also do it on a particular platform Like Github, Gitlab, Bitbucket by just raising a pull/ merge request for two branches.

More Interaction

This totally depends on you. Because you have to be active for this. You have to do following items:

1. Participate in Team interaction and discussions for making the project grow.
2. Read the latest technology and ideas which you can implement in the project. But always have proof of concept before applying that.
3. Share the ideas and problems with your team members and groups.

Thanks for Reading this. Please suggestion & improvements are always welcome for this document. But participate to make it more valuable.

Thanks By:
Angular Team
Crownstack Pvt. Ltd.
www.crownstack.com