GETTING STARTED WITH YOUR

FINAL YEAR PROJECT

BY SWASTIK GOWDA

SEPTEMBER 2021

TABLE OF CONTENTS

ABOUT

Think of this as an ebook or pdf or a suggestion handout, on how to get started with your final year project. This ebook mainly focuses on providing information on which technologies to choose and workflow or architecture to use while developing your final year project.

This ebook is written in such a way that anyone with basic programming skills can easily understand and skyrocket their final year project development.

This ebook just states my suggestions and my suggestions only, but I have done my fair share of research on this topic since my hobby is building projects, sometimes alone and sometimes with my friends.

HOW IS THIS GONNA HELP ME?

I know that colleges are not teaching us how to build projects or tools and technologies needed to build one, but they expect us to know about these things, which might be difficult for some of us, so this ebook will streamline your options and provide/equip you with the info on which tools and technologies to use and will also provide you with the resources to learn about them further, of course, you might need to do some extra searching and learn about those things and learn to work with them, this ebook is just gonna introduce you to them , rest is up to you.

This ebook is not going to be very technical, just the right amount of info is shared, for you to understand and get started with your project.

ABOUT MYSELF

My name is Swastik Gowda, 4th year CSE student here at Dr. AIT Banglore.

I am a passionate programmer/geek/techie, most of the time glued to my computer, and sometimes here and there I enjoy to teach or blog about my learnings, you can find me on Instagram or on GitHub / LinkedIn / Portfolio website.

I am not good at writing nor explaining, just to help my friends get started with their project, I am writing this and that too in a hurry so there might be a ton of grammar or spelling mistakes or technical errors so, please do ignore those.

IMPORTANT SIGNS TO LOOK OUT FOR

★ Personnel recommendation

★ Second Best recommendation

⚠ Warning / Not recommended

2 | WHICH TECHNOLOGIES TO CHOOSE ?

In the tech field, there is no one to rule them all or no one silver bullet, the technologies you must choose boils down to one thing and that is what are you trying to build?

★ ■ If you are trying to develop a **web-based project** - then I would recommend that you go with the normal popular tech stack, that is HTML, CSS, & JavaScript, only these would be enough for you to build the basic applications but using only these you cannot build complex applications, you can just trick the college to think that it is but it will not be one.
The problem is that there is no database involved so you cannot store data for doing complex applications, let's say for eg - you want to develop a library management system or railway booking or movie booking system, etc,.., these all require you to store data somewhere and access it later and perform some level of CRUD operation on it, but none the less you can choose any one of the following methods you -

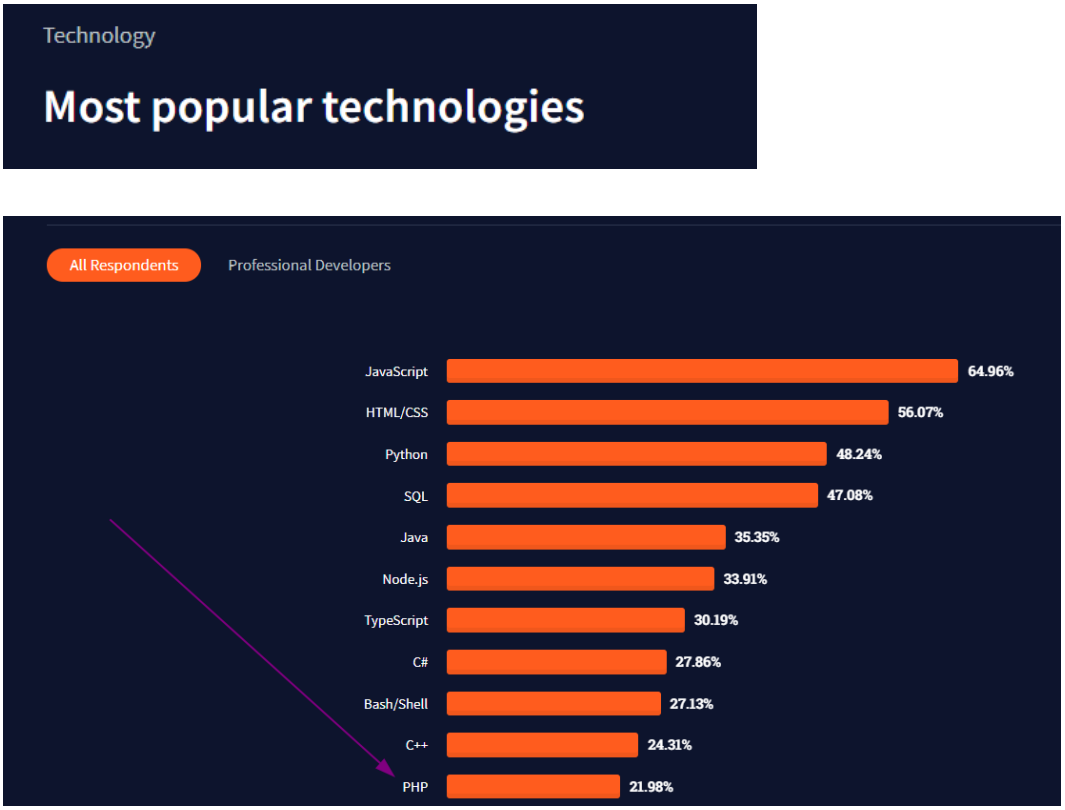
★ **1 -** You can store the data in the browser itself using any one of these -

1. Local Storage
2. IndexedDB

which is super easy and we can do complex stuff with it and with a well-prepared presentation and well-written report you can successfully trick your project mentor into thinking that this is a complex application and this will be sufficient to get good marks for your project.

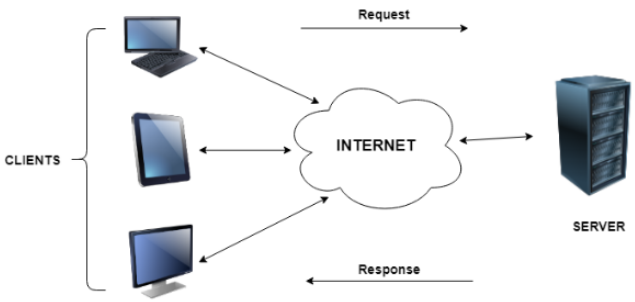
★ **2 -** The other thing you can do is instead of storing data in your browser you can store it in online databases like Zoho Creator, Supabase, ZenBase, StackBy, etc.., you can google "online database management system" for more tools like these, I am not much familiar with these tools but you can store data in them and do crud operation on them but none the less, you will have to know about API methods and how to send GET/POST requests and receive responses.

⚠️ **3 -** You can use HTML, CSS, JS, and PHP programming language and have it communicate with MySQL database, this is another previously widely used tech stack which is a good option if you already know PHP and MySQL but if you don't know PHP, then I wouldn't recommend learning it just for the sake of communicating with the database, since it's syntax is tedious to work with and is getting less popular day by day.
Below is the survey conducted by stack overflow in 2021-



★ **4 -** You can create your own Rest API endpoint, which means you create a web server, which accepts incoming requests and send the response back to the client, by this you can eventually communicate with your client/frontend code which might be written in JS or any programming language for that matter because each and every programming language will have some module implemented to work with Rest API's, for eg we use fetch in JS to send API requests likewise some module will be there for all languages, using those we can communicate with our backend and the

backend with Rest API endpoints can be written in node js or python along with Django framework, etc.., and can have a SQL or NoSQL database set up to communicate with our backend. This is how it's done in the industry so you might want to choose this option, but this might be hard for beginners but since we have one year to complete the project, and on top of that, we do it in a group there we will be support as well, so I guess this is perfect if you are ready to do a little bit hard work and learn the important stuff.



★ ■ If you are trying to develop a **Desktop application** - then you can go with python or java or if you know C# which is used to develop windows applications, or if you know JS then you can go with electron and many other options, anything is fine but none the less here also you might need to store data in the database and know how to communicate with that database, you can choose any database like Mysql or MongoDB or PostgreSQL, etc.., there will be some libraries in all popular languages which will help you to communicate with the database.
If you already know python and MySQL then it would be a great choice, or if you know java and MySQL, it would also be a good choice, but the other thing is that you will also have to build basic GUI using these languages for you to interact with the software because you cannot submit a project without GUI until or unless what you have developed is a super complex ML/AI-related algorithm. You can create GUI in java using applets/servlets, in python we can use libraries like kivy, Tkinter, etc..,

★ ■ If you are trying to develop an **IoT based project**- then you might be using IoT boards like Arduino Uno or Raspberry pi or Tiva launchpad, if you are using Arduino Uno or Tiva launchpad, they require you to use their own editors like Arduino IDE and Energia, and writing code in these IDE's demand you to have C/C++ skills and you should know about the various sensors that you might want to use in your IoT project, there will be datasheet for each and every sensor that is not the issue, but the dis-adv is that there is not a lot of community behind these things so when you get stuck, then it's up to you to solve it and move on, but the pro in here is that even a basic IoT project can be made to seem complex if explained and reported with the right amount of technical jargon and that will suffice you to get good marks.
If you are using Raspberry pi, I guess you'll have a lot of freedom on which programming language to choose because it can be programmed in C/C++ or python, etc.., by using the libraries that helps you to interact with Raspberry pi.

★ ■ If you are trying to develop an **ML/AI-based project**- then you might be using python / Rlang and TensorFlow library and whatnot, since its complex, I guess it's fine even if you don't create a GUI for interacting with the program but a working algorithm/program with an expected result should be shown if you are truly interested in ML/AI not for the hype, then only I recommend you start with ML/AI projects because it's gonna be hard.

My Final thoughts on this matter are, first read the "Must know technologies/architecture !" section of this ebook and then take your teammate's skills into account and then decide what type of project to build and then choose the technology appropriately.

3 | HOW TO COLLABORATE WITH TEAMMATES ?

Most of them might not think that this would be an issue, since most of your teammates are your friends, and have gone along together very well, but when it comes to final year projects or projects in your companies, there are certain protocols they follow, and you will eventually have to adopt it later or sooner in the company, so again it's up to you whether you are gonna follow professional collaboration methods or not.

These are some of my suggestions -

1. Adopt Agile methodologies and preferably follow scrum workflow.
2. Learn about Kanban Board.
3. Use tools like Trello / Asana for assigning, scheduling tasks, and collaborating in teams.
4. Use Slack for communication, an alternative to your WhatsApp groups and you can do a lot with slack too.
5. Must use Git as version control and Github as code repository.
6. Follow feature branch git workflow.
7. Design the prototype or mockup of your project using designing/collaboration tools like Figma.
8. Review each other's code on Github, don't merge until at least one person has reviewed your code.
9. If possible, use Microservices Architecture, have explained this in the upcoming section of the ebook.
10. You can do pair programming using tools like vs code editor with live share extension or CodeSandbox etc..,
11. Use tools like zoom or gmeet for video conference.

4 | MUST KNOW TECHNOLOGIES/ARCHITECTURE!

If you want to develop complex projects, then you might need to know some of the technologies/architecture specified -

1. Git/Github - No matter the programming languages you need to know their technologies, to collaborate, and for code storage and version control.
2. Restful API methods - You need to know at least how to send get/post requests in any programming language and wait for the response and after receiving the response how to process it because pretty much the whole web works on this basis. But you can use other client-server interaction patterns like graphQL or RPC or message brokers but you don't need these, but just mentioned for those interested geeks out there.
3. Difference b/w prototype and mockup and wireframes - You need to know the difference between these because, whenever a project idea comes to your mind you shouldn't start coding it without a proper plan or wireframe or mockup, that is a formula for disaster. There are various tools out there like Figma or adobe xd etc.., for designing mockups or wireframes, use those!
4. Microservices architecture - This is very important, this is an architecture wherein we have multiple web servers responsible for handling the different tasks of our software let say for eg - we have an e-commerce application like amazon and imp features in that are the authentication, the payment system, and the recommendation system, etc.., here we can have one web server responsible for authenticating and other for handling payments and other for recommending products to users likewise it goes on .., because of this it

provides us with a lot of flexibility and several other advantages as well and the main reason I am telling you guys about this is that, let's say you are a team of two and one teammate knows python and the other teammate knows java it doesn't matter, one teammate can build one feature in python language and another teammate can build other feature in java language and both of them can communicate with the help of Rest API's but none the less both of the teammates should know how to implement and work with Rest API's and should know to communicate with anyone database, even if the code is hosted in completely different platforms, it doesn't matter until or unless you know how to talk to HTTP (basically Rest API's).

★ 5. Anyone database - It doesn't matter which database you guys choose, it may be SQL or NoSQL databases, you need to know at least how to do crud operations on anyone database because complex applications need data to work with and for that reason, they need to be stored somewhere.

5 | FINISHING UP!

That's it guys, this is what I wanted to share and help my friends who are struggling to develop projects or trying to understand what to learn and which technology to use. Below are just my quick few suggestions -

1. Don't use tools like WordPress or web-flow because you won't learn anything if you use tools like these, they are just a hacky alternative.
2. Follow best coding/programming design patterns.
3. Setup a linting tool to improve your code quality.
4. Refactor your code every often.
5. Do testing for your projects, like unit/integration or end-to-end testing, they are very useful because if you are always refactoring code like me, you'll break things along the way and tests are the one which helps us to realize which feature broke. If possible consider the TDD approach.
6. Follow the best Logging/Debugging practices.
7. Most of these suggestions are technology agnostic, it doesn't matter which programming language or technology you are using, it applies to all!
8. Document each and everything, the libraries you used or how to interact with the software or what does a specific function do, etc.., or make a flowchart of the data flow or make a system design of you software.

6 | BEYOND BASICS!

This section is just a bonus, it is helpful for those who want to go beyond basics but this is not necessary for final year projects, but if you want you can develop an application whose quality is of an enterprise-grade.
Below are my quick few suggestions -

1. If you want to make your applications balance load(incoming requests) efficiently, then use load balancers.
2. Build robust and high-tech GUI using different frontend frameworks or GUI libraries.
3. Setup Monitoring and Logging system for your applications.
4. Learn about best security practices and at least secure your application against the top 10 OWASP threats.
5. Configure your application to use SSL/TLS.
6. If your applications deal with media assets, then compression or optimization will increase the application's efficiency.
7. Perform caching to make your application faster.
8. Learn to work with database indexes and sharding.
9. Try to serve your static assets using CDN.

These are some of my quick suggestions, but even after applying all the above-mentioned suggestions, your application will still not be enterprise-grade, because a lot goes into making an application faster, robust, fault-tolerant, etc.., but nonetheless this is the first step towards making one.

7 | THANK YOU!

If you read till here guys, then I appreciate it a lot, so if there are any questions or if you want to appreciate it just drop a message or DM me guys and stay awesome.