

**Project proposal**

**Name and surname of the student:**

Mahdiye Rashidi

**Student number:**

9712126027

**Name of the Course:**

Software Engineering

**Degree and name of the field of study:**

Undergraduate computer engineering

**Supervisor:**

Dr. Haji Ghorbani

Autumn and winter 1400

**Introduction**

Developing a software product from scratch, launching it, and then guiding users to adopt it can all be extremely challenging and unpredictable. To keep everyone on the same page, and make an otherwise overwhelming journey easier to navigate, the concept of software documentation emerged. Thus, knowing how to write software documentation is super-important.

Software development teams, testers, and users alike (and everyone else related to the project) need some guidance to help them with their goals. With effective documentation, everyone wins.

Writing software documentation is tricky. While workflows vary from company to company, there are certain best-practices which, if adhered to, can make the process a lot smoother (and yield the ideal results).

I prepared a documentation of all my experiences in this project which we read together in the following.

Let's first of all start with this question that What is Software Documentation?

**What is Software Documentation?**

Software documentation refers to the documents, tutorials, and other material that describe the development, functionality, and/or the use of a software product. It is one of the many forms of technical documentation.

The purpose of software documentation is simple – to streamline the communication between all the parties involved with the product.

Within an org where the software is being developed, a technical document can be considered a sort of a *wiki* page – a guiding blueprint that the development team can refer to when working on it. Additionally, it can also help those who use the finished version of the product.

To be more specific, effective software documentation can help:

* **Align Expectations with Understanding** – one of the main concerns of any software company is to ensure that the software engineers work towards bringing the vision of the stakeholders to life. An error in documentation can cause discrepancies between what’s required and what’s being developed.
* **Aid in Helping the End-User** – in addition to guiding the programmers in implementing requirements, software documentation also helps the audience with setting up the software, understanding the user-interface, and following the best use-cases.
* **Record Progress** – another internal use of software documentation is to keep track of the software development lifecycle (SDLC). This can help a company measure progress, learn from mistakes, and make better decisions in the future.

Every tech company – from small startups to well-established giants like Microsoft, Amazon, and Google – uses some form of software documentation.

All in all, programmers, stakeholders, and users alike benefit from this form of technical communication.

**The Different Types of Software Documentation**

Before learning how to write software documentation, it’s important to understand the different types of technical documents you might be required to work on.

They are mainly distinguished on the basis of the specific goals they accomplish.

Depending on the methodology/approach it uses, a company may not use every type of document (those that follow an agile approach usually don’t engage in heavy documentation in the beginning).

With that out of the way, software documentation can be split into two broad categories.

**Product Documentation**

When talking about software documentation, people mostly refer to product documentation.

As the name suggests, this category includes all the documents/material that contain essential details about the product. Of course, it can be for both the software developers and the end-users.

We can further classify product documentation into the following types:

* **Requirements Documents** – these are created at the very beginning of the project. As the name suggests, they’re intended to clearly communicate what is expected from the software – the functionality, features, goals, etc. – to the software engineering teams. They are also known as “ReadMe” documents.
* **Architecture/Design Documents** – these provide an overview of the software’s architecture and highlight the design principles to be used.
* **Source Code** – this includes documents containing the actual code (Python, HTML, etc.) of the software product.
* **End**-**User** – this includes all the user documentation, such as user guides, user manuals, reference manuals, etc. The purpose is to ensure a smooth user experience. If the solution is an API, the material is referred to as API documentation.

In addition to the above, a document detailing the marketing strategy/research can also be filed under product documentation.

**Process Documentation**

This category includes all the documents describing the underlying processes that bring a product from ideation to the launch phase.

The purpose of process documentation is to break down the software development journey (and provide a North-Star for all the teams involved in the project).

Process documentation can include:

* **Plans** – not to be confused with the *requirements*, these are also typically created before the development starts. They include cost estimates, schedules, etc.
* **Progress Reports** – these help measure the success of the software’s development (by using certain metrics) and ensuring that the development team is on the right path.
* **Working Papers** – these special documents record the ideas, thoughts, and notes that the engineers, developers, and systems administrators have during development.
* **Standards** – finally, development teams need to specify the standards of coding and design that they stick with to keep things consistent.

While product documentation is intended for both internal and external audiences, process documentation is mainly intended for the people developing the product.

1. Understand the Purpose and Audience of the Document

Before anything else, We need to take a step back and ask ourself why We’re about to create said document.

Since there are so many software documentation types, even the most experienced technical writers are prone to mixing up the main purpose or the audience they’re addressing.

For that reason, We first need to highlight the purpose of our document. A simple tip is to open up a blank doc and typing up its purpose as the title.

For instance, if we’re creating a document that conveys the expectations of the stakeholders to the software developers, the title could be something along the lines of: “The Vision for XYZ Software.”

Furthermore, clearly highlight the audience we’re creating the document for. Go one step further and create personas of the type of people who would read our technical content.

These may sound like trivial things to do, but trust me – they help.

2. I wrote down the important questions

It’s pointless to create a technical document that doesn’t address the pain-points or answers the questions of the audience.

Once I’ve identified the goal and the audience for my technical document, the next step is to anticipate (or better yet, ask about) the questions the readers might have about the product.

This is where my personas will come in handy.

List down those FAQs somewhere. But don’t include them in my main document just yet.

The goal of identifying the questions is to collect my thoughts, design our document accordingly, and provide the most relevant information that delivers maximum value.

3. I created an Outline for my Document

Figuring out the outlining process is an important aspect of learning how to write software documentation.

That is why, the next step is to come up with an appropriate design for your document.

If you’re writing a particular type of document for the first time, you may need to prepare a structure from scratch.

As with everything else, there is no universal template for every type of software documentation. The design/outline of your document will be based on the specific goals you want to accomplish and the comprehension level of your audience.

Here’s what it should include, in order:

The title and the audience

An executive summary of the document

The problem statement and the scope

The goal(s) of the document

Requirements for the reader (if applicable)

Instructions/Steps/Solutions/Findings/Code (whatever is applicable)

A timeline (if applicable)

References (if applicable)

At the end of the day, you know your audience better than anyone. Include anything else in your outline that might help.

Structure the information in the most helpful way. You may need the assistance of a graphic designer.

You can then use the outline/design as a template for all future documents to maintain the consistency of communication and make small improvements along the way.

4. I gathered the Required Information

Depending on your level of familiarity with the subject, you may need to conduct some heavy researching to gather and validate all the relevant information for your document.

This may entail interviewing subject matter experts/users, talking to the stakeholders, going through existing documents, and/or conducting research over the internet.

Process the research data into usable information, compile it over your outline, and provide references wherever necessary to add credibility to your writing (if it applies).

5. At the beginning of the project I started Writing the Draft

If you are in this step you’ve laid a strong foundation for your technical document, all that’s left to do is to draft it.

If you’ve created a solid outline and gathered all the required information beforehand, the actual drafting process shouldn’t take very long.

Here are some quick tips on writing:

* Don’t write more than you need to
* Wherever possible, avoid using jargon
* Use plain English od Persian to convey your thoughts
* Avoid being unambiguous
* Don’t edit while writing

While drafting, keep referring to the goal and the audience to stay on track.

6. Leverage Visuals

After you’re done preparing your draft, you should also include a few visuals (flowcharts, illustrations, screenshots, etc.) to augment your document.

If they’re available, you may also choose to add the graphics as you write the draft. Some writers even prefer adding them during the outlining phase. You’re free to do whatever feels right/based on your special circumstances.

These visuals can be used to emphasize a point, further explain a technical concept, help out the reader, and just make your document look so much better.

**7. I performed Final Editing**

We’re still not done yet.

The only thing left to do now is to edit your technical document.

Depending on your researching, formatting, and writing skills, it can take anywhere from a single to multiple rounds of edits to get the final document.

This entails having an editor (if available), a subject matter expert, or just another pair of fresh eyes look at your document for any grammatical, technical, or contextual errors.

**Note:**

The secret of good software documentation – whatever it may be – is ample planning.

Software documentation, just like any other form of technical writing, cannot be rushed.

Furthermore, it’s not always a one-man effort( like my project :) ). It requires close collaboration with the relevant stakeholders, software developers, and all the other parties directly or indirectly involved with the project.

By following the best practices, touching all the pain-points, and most importantly, staying within the scope, you can easily prepare a great software documentation in no time.

**What is README File?**

**As We know the main part of a good template for a README File is:**

|  |
| --- |
| **Project Title** |
| A little brief about what the project is about. It should be like a small summary format informing about the main purpose of the project. |
| **Motivation** |
| This section is for letting the reader know why you created this project, the reason behind pursuing such a project, and why you have decided to do it. |
| **Build Status** |
| This basically explains the current build status of the project. If there is a bug /error which needs addressing. This is done so for two different reasons The user understands that this is an issue and does not spend more time figuring if it was a mistake on their part. A developer who is familiar with the issue can suggest some solutions directly without going through the whole code. |
| **Code Style** |
| This lets the users know that you have used a particular code style and helps them when contributing to your project so that the whole project code style stays the same. Some common code styles: standard, xo, etc. |
| **Screenshots** |
| As the saying goes, a picture is equal to a thousand words. Most people will be interested if there is a visual representation of what the project is about. It helps them understand better. A visual representation can be snapshots of the project or a video of the functioning of the project. |
| **Tech/Framework used** |
| This is used to help the reader understand which tech or frameworks have been used to do the project. It helps the reader understand which all tech stack he has to be familiar with to understand the whole project. |
| **Features** |
| This is where you write what all extra features have been done in your project. Basically this is where you try to make your project stand out from the rest. |
| **Code Examples** |
| This is where you try to compress your project and make the reader understand what it does as simply as possible. This should help the reader understand if your code solves their issue. |
| **Installation** |
| If your project needs installation of certain software or configurations to the system. Do mention it in this section as it helps a lot for the reader to use your project. The steps mentioned should be precise and explanatory.  If possible, you can add links that can help them better understand how to configure/install the necessary files or software. |
| **API reference** |
| If your project is small, then we can add the reference docs in the readme. For larger projects, it is better to provide links to where the API reference documentation is documented. |
| **Tests** |
| This is the section where you mention all the different tests that can be performed with code examples |
| **How to Use?** |
| As I have mentioned before, you never know who is going to read your readme. So it is better to provide information on how to use your project. A step-by-step guide is best suited for this purpose. It is better to explain steps as detailed as possible because it might be a beginner who is reading it. |
| **Contribute** |
| This is where you let them know that they can contribute and help you out. A guideline on how to contribute is also helpful |
| **Credits** |
| Giving proper credit is most important. Mention any links/repos which helped you or inspired you to build this project. It can be a blog, another open source project, or just people who have contributed in building this project. |
| **License** |
| A short description of the license. (MIT, Apache, etc.) |

**my README File**

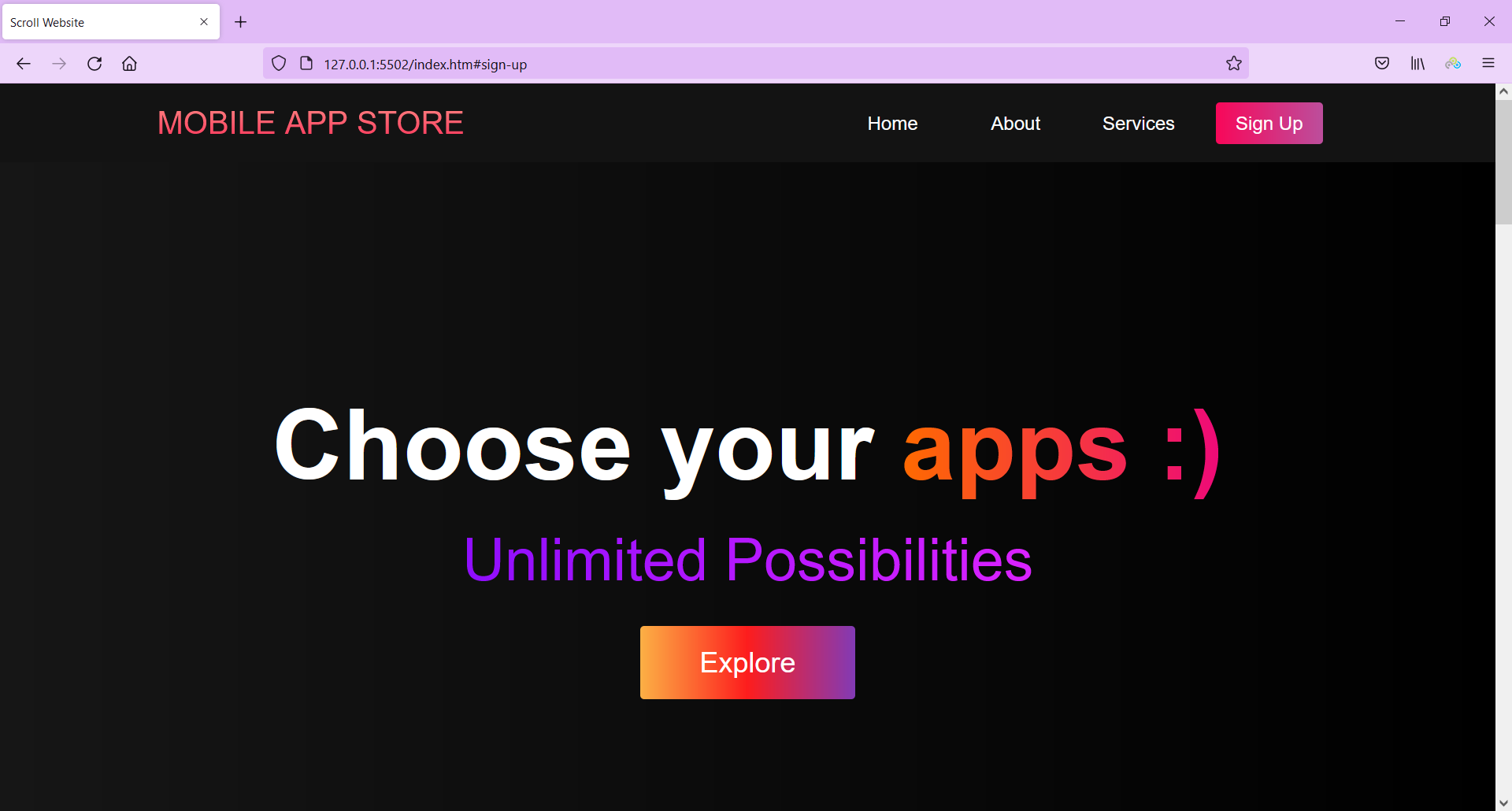
Project Title: Mobile App Store- An App store for searching applications.

Motivation: I started this project just because it was my software engineering course project. (I'm really a honest person 😊 )

Build Status: I am currently completing the back-end and front-end.

Code Style: I try my best to make my code as readable as possible, I use commenting as much as possible, I try to name my variables at least the necessary standard for readability, and so on.

Screenshots: This is a screenshot of my home tab of website during this time period of the project.



Tech/Framework used:

* Python, HTML, CSS, JavaScript
* Microsoft **SQL Server Management Studio(MSSMS)**
* GitHub, Trello

Features:

* Home
* About
* Services
* Categories list
* Search box
* Show single app
* Show articles
* ....

**Contributor:**

[MahdiyeRashidi](https://github.com/MahdiyeR)

**Support:**

You can contact us by sending mail to [Mahdiye5818@gmail.com](mailto:Mahdiye5818@gmail.com) Also you can follow us on [**Instagram**](https://instagram.com/imahdiye_r?utm_medium=copy_link)

**Free Software, and open source!**