Homepage

File structure

```
homepage/
| .next/
| components/
| | layouts/
| | ui/
| | utils/
| | .DS_Store
| | GeneralComponents.js
| node_modules/
| pages/
| .DS_Store
| .eslintrc.json
| .prettierignore
| README.md
| next.config.js
| package-lock 2.json
| package-lock.json
| package.json
| prettier.config.js
```

How the structure works

Directory and file purposes

Code snippets

Example Component `Header.js`

> In web development, components are reusable and modular pieces of code that encapsulate a specific functionality or part of the user interface. So basically it's a js file where you write your html code and later you can call this js (function) which will then output the code you've written in the js file in the components folder. Normally you would style this code also with a seperate CSS file but in this case we use ChakraUI where you can directly style the html code within the js file by wrapping them in so called boxes from ChakraUI. This type of Javascript is also called JSX.

```
// Header.js
import React from 'react';
// Since we are using ChakraUI external file is not needed
import './Header.css';  // Import the associated stylesheet
```

```
const Header = () => {
  return (
    <header className="header">
      <h1>My Personal Website</h1>
      <nav>
        <ul>
          <li><a href="/">Home</a></li>
          <li><a href="/about">About</a></li>
          <li><a href="/blog">Blog</a></li>
          <li><a href="/projects">Projects</a></li>
        </ul>
      </nav>
    </header>
  );
};

export default Header;
```

Types of exports

Named exports

```
export const SecondTest = () => {
  return <p> Test Worked !!! </p>
}
```

When you use the export statement in the declaration it counts as a named export which means when giving the path you also have to import this component with the curly braces.

Default exports

```
const Test = () => {
  return (
    <p> Test Worked !!! </p>
  )
}

export default Test
```

But that doesn't count here when exporting as a default and you want to import this component you just have to write for example: `import Test from '../path/Test'`

So those are named and default exports.

Example with ChakraUI

> Box, Heading, Flex, and Link are Chakra UI components that provide styling and layout functionality.

> The styling is achieved through the use of style props (e.g., bg, color, p, textAlign) and utility functions (e.g., mt for margin-top).

> Chakra UI allows you to build a consistent design by applying styles directly to components or by creating reusable styles using Chakra's theme. You can customize the appearance of these components by modifying their style props or by leveraging Chakra UI's theming system.

```
// ChakraHeader.js

import React from 'react';
import { Box, Heading, Flex, Link } from '@chakra-ui/react';

const ChakraHeader = () => {
  return (
    <Box bg="#333" color="white" p={4} textAlign="center">
      <Heading as="h1" size="lg">
        My Personal Website
      </Heading>
      <Flex mt={2} justifyContent="center">
        <Link mx={2} href="/">
          Home
        </Link>
        <Link mx={2} href="/about">
          About
        </Link>
        <Link mx={2} href="/blog">
          Blog
        </Link>
        <Link mx={2} href="/projects">
          Projects
        </Link>
      </Flex>
    </Box>
  );
};

export default ChakraHeader;
```

Example for framer motion

> Framer motion is also directly included into the js files

> motion.div is a Framer Motion wrapper around the regular div element, enabling animation features.

> The initial prop defines the initial state of the component before any animation.

> The animate prop defines the target state of the component after animation.

> The transition prop specifies the duration of the animation.

```
// ExampleComponent.js

import React from 'react';
import { motion } from 'framer-motion';

const ExampleComponent = () => {
  return (
    <motion.div
      initial={{ opacity: 0, x: -100 }}
      animate={{ opacity: 1, x: 0 }}
      transition={{ duration: 0.5 }}
    >
      <h1>Hello, Framer Motion!</h1>
    </motion.div>
  );
};

export default ExampleComponent;
```

Eample of importing css modules

```
// pages/about_me.js
import React from 'react';
import styles from './styles/about_me.module.css';

const AboutMe = () => {
  return (
    <div className={styles.container}>
      <h2 className={styles.title}>About Me</h2>
      {/* Other content */}
    </div>
  );
};

export default AboutMe;
```

Example of using component in pages/index.js

```
// pages/index.js

import React from 'react';
import Header from '../components/Header';

const HomePage = () => {
  return (
```

```
      <div>
        <Header />
        <h2>Welcome to the Home Page</h2>
        {/* Other content for the home page */}
      </div>
    );
};

export default HomePage;
```

Understanding Git

ChatGPT Understanding the basics of Git and version control is essential for effective collaboration and project management. Here are some key concepts and commands to get you started:

1. Initialize a Git Repository:

```
To start using Git in your project, navigate to your project's root
directory and run:
git init

This initializes a new Git repository in your project.
```

2. Check the Status of Your Repository:

```
To see which files are modified, untracked, or staged, use:
git status
```

3. Stage and Commit Changes:

```
To stage changes for commit:
git add filename

To stage all changes:
git add .

To commit changes:
git commit -m "Your commit message here"
```

4. View Commit History:

```
To view a log of commits:
git log
```

5. Create Branches:

```
To create a new branch:
git branch branch_name

To switch to a branch:
git checkout branch_name

Or combine both commands using:
git checkout -b branch_name
```

6. Merge Branches:

```
To merge a branch into the current branch:
git merge branch_name
```

7. Pull and Push Changes:

```
To fetch changes from a remote repository:
git pull origin branch_name

To push changes to a remote repository:
git push origin branch_name
```

8. Remote Repositories:

```
To add a remote repository:
git remote add origin https://github.com/your-username/your-repository.git

To clone a repository from a remote:
git clone https://github.com/your-username/your-repository.git
```

9. Ignoring Files:

```
Create a .gitignore file in your project's root directory to specify
files and directories that Git should ignore.
```

10. Undoing Changes:

```
To discard changes in your working directory:
git checkout -- filename
```

```
To unstage changes:
git reset HEAD filename
To undo the last commit (use with caution):
git reset HEAD~1 --hard
```

11. Resolve Conflicts:

```
If conflicts arise during a merge, resolve them manually, then use:
git add filename
git merge --continue
```

12. Tagging:

```
To tag specific commits:
git tag -a v1.0 -m "Version 1.0"
```

VCS See different versions

1. View Commit History:

```
To see a list of commits along with their details, run:
`git log`
This command shows you a chronological list of commits, including the
commit hash, author, date, and commit message.
```

2. View Specific Commit Details:

```
If you want to see the details of a specific commit, find the commit hash
in the git log output and use:

git show <commit-hash>
Replace <commit-hash> with the actual hash of the commit you want to view.
```

3. Navigate Through Commit History:

```
To navigate through the commit history, use arrow keys or press q to exit
the `git log` or `git show` view.
```

4. View Changes in a Specific Commit:

```
To see the changes introduced by a specific commit, you can use:
git diff <commit-hash>
This shows the differences between the specified commit and its parent.
```

5. View Changes Between Two Commits:

```
To view changes between two specific commits, use:
git diff <commit-hash-1>..<commit-hash-2>
```