**Internship Report**

**Report on Student Industrial Work Experience Scheme (SIWES) Training Programmed Held Between**

**January – August 2020**

**At**

****

**54 Demurin Street Ketu Lagos.**

**By**

**Adedokun Emmanuel Ayo**

**17/0984**

**Submitted to**

**FACULTY OF SCIENCE**

**DEPARTMENT OF COMPUTER SCIENCE, ADELEKE UNIVERSITY**

**P.M.B 250, LOOGUN-OGBERIN ROAD, EDE, OSUN STATE**

**IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF BACHELOR OF SCIENC (B.Sc) DEGREE IN COMPUTER SCIENCE**

**DEDICATION**

This report is dedicated foremost to God Almighty for given me the grace and opportunity to be able to complete my six (6) months SIWES program at OneKiosk despite the challenges posed by the covid-19 lockdown, curfew and others.

I would also like to dedicate it to my Dad Mr. Israel, Uncle Mr. Adeyemi and my Cousin Mr. Deji for their love and support all through my SIWES program period.

**AKNOWLEDGEMENT**

I am forever grateful to God almighty, the giver of all wisdom, knowledge and understanding, without whom I would have achieved nothing at all.

I wish to express my thanks to my beloved parents for their moral and financial support toward the completion of this program.

I also want to say a big thanks to my Industrial Base Supervisor at OneKiosk Mr. Adeshina John the CEO of OneKiosk for his continues word of motivation and moral support towards me.

And the entire staff of the Technical department for their relevant suggestion and contribution towards the completion of this program and I also thank the general staff and management of OneKiosk Africa for their understanding and opportunity given to me to work in their company.

**ABSTRACT**

This SIWES report presents the experience gained during my six (6) months of industrial training undertaken at OneKiosk Africa, 54 Demurin, street Ketu, Lagos State.

During my training I acquired practical knowledge on how to design graphics prototypes identity design, user interface design and user experience design, I also learnt how to build a mobile app, learnt how to use automation tools for programming, and also how to use and manage cloud computing and database.

This report discusses the technical skills gained during the training period and justifying the relevance of the scheme in equipping students with needed technical competence to thrive in the real world

**TABLE OF CONTENT**

[**DEDICATION** ii](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862023)

[**ACKNOWLEDGEMENT** iii](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862024)

[**ABSTRACT** iv](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862025)

[**CHAPTER ONE** 1](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862026)-3

[**INTRODUCTION** 1](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862027)

[1.0 SIWES: (STUDENT INDUSTRIAL WORK EXPERIENCE SCHEME) 1](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862028)

[1.1 PURPOSE OF SIWES: 1](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862029)

[1.2 AIMS AND OBJECTIVES OF SIWES: 1](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862030)-2

[1.3 DESCRIPTION OF THE ESTABLISHMENT OF THE ATTACHMENT 2](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862031)

[1.4 OBJECTIVES AND VISION OF MATT O’BELL 2](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862032)

[1.5 COMPANY’S AREA OF SPECIALIZATION 2-3](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862033)

[1.6 DEPARTMENTS IN THE COMPANY 3](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862034)

[**CHAPTER 2** 4-17](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862035)

[**INDUSTRIAL EXPERIENCE** 4](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862036)

[2.0 TECHNNICAL DEPARTMENT 4](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862037)

[2.1 HTML AND ITS PROPERTIES 4-5](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862037)

[2.2 BOOTSTRAP 5-6](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862038)

[2.3 PYTHON AND ITS PROPERTIES 6-10](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862039)

[2.4 DJANGO AND ITS PROPERTIES 10-16](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862040)

[2.5 MY FULL WEBSITE WITH HTML, BOOTSTRAP AND DJANGO 16-17](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862041)

[**CHAPTER 3** 18](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862042)

[3.0 PERSONAL INPUT TO THE COMPANY (MATT O’BELL) 18](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862041)

[**CHAPTER 4** 19](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862042)

[**SUMMARY, CONCLUSION AND RECOMMENDATION** 19](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862043)

[4.0 SUMMARY 19](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862044)

[4.1 CONCLUSION 19](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862045)

[4.2 RECOMMENDATION 19](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862045)

[**REFERENCES** 20](file:///C:\Users\DELL\Downloads\siwes%2520report%2520correction.docx#_Toc524862046)

**CHAPTER 1**

**INTRODUCTION**

**1.0 SIWES: (Student Industrial Work Experience Scheme)**

The early phase of science and technology in Nigeria was characterized by the theoretical lectures in polytechnics and universities which have proven to be an ill method of teaching. Students in Universities and Polytechnics graduate with little or no technical experience in their course of study.  
In the same vein, students’ inability to contribute to the society is hampering the growth and development of our country. It was in this view that SIWES was introduced to the Industrial and Educational sector.

SIWES is an acronym for Student’s Industrial Work Experience Scheme. SIWES was established in the year 1973 in order to improve the standard of education in Nigeria in order to achieve the needed technological advancement.

Economists being able to evaluate the role technology plays in a country’s economy concluded that for an economy to grow and develop there be advancement in the technology sector of the country.  
SIWES was solely funded by ITF (Industrial Training Funds) during it early stage not until it was difficult to continue for economic stress: then the responsibility was shared between Industrial Training Funds (ITF) and the Federal Government.

The Federal Government took over the funding of the scheme and Industrial Training Funds took over the managerial position by managing the funds given to them by the Federal Government in order to sustain the scheme.

SIWES (Student’s Industrial Work Experience Scheme) is a scheme for the duration of sixteen weeks (4 month) or more. SIWES is done after the first year in polytechnics (ND1); and done after Second year or third year in Universities depending on the institutions.

The effective management of Student’s Industrial Work Experience Scheme (SIWES) has been as a result of the cooperation and well played roles of the Federal Government, ITF, Supervising agencies.

**1.2 AIMS AND OBJECTIVES OF SIWES:**

The specific objectives of SIWES were summarized by the federal government as follows:

1. To provide an avenue for students to acquire industrial skill and experience in their course of study.
2. To prepare students for the work situation they are likely to meet after graduation.
3. To expose students to work methods and techniques in handling equipment and machinery that may not be available in the university / Institute.
4. Provide student an opportunity to apply their bridging the gap between Higher Education and actual practice.
5. Make transition from the university to the world of work easier and thus enhance students contact for later job placement after graduation.
6. Enhancing students’ contacts with potential employers while on training.

**1.3 DESCRIPTION OF MY IT ESTABLISHMENT**

OneKiosk Africa started as an idea in year 2018 as an ecommerce company and on 7th January 2019 OneKiosk was registered and started operating here in Nigeria. Founded by Mr. Adeshina John OneKiosk Africa aimed to build a sustainable ecommerce for Africa by leveraging the use of local retailers around to serves customers withing the range of the retailer under fifty-nine (59) minutes. OneKiosk was able to achieve this by creating a platform where local retailers can register their business, put their goods virtually online and give access to their location in order to be discovered by customers. Customers then go to the platform find what they want fro bm the nearest retailer close to them as recommend by the platform; customers then make a purchase of the item and then under fifty-nine minutes or less their items will be delivered to them by the local logistics that OneKiosk has already provided.

**1.4 OBJECTIVES AND VISION OF ONEKIOSK**

1. Increase operational efficiency in their business
2. Protect valuable assets such as human capital, physical assets, cash, receivables etc
3. Improve internal control systems.
4. To be a powerful force for the Economic Growth of Africa
5. To connect Africans with each other and the rest of the world through Technology & Commerce.
6. To make it easy to search, order on all platforms, and pay on delivery.
7. Access 100% Genuine Products from Local Vendors
   1. **COMPANY’S AREA OF SPECIALIZATION**

The company is specialized in the following core services.

1. Ecommerce
2. Support Services: Technical / Functional
3. Logistics
4. Deliveries
5. Customer Care

**1.6 DEPARTMENTS IN THE COMPANY**

1. Management Department
2. Technical Department: Software Developers, Web Developer, UI/UX Designers and Graphic Designers.
3. Business Analyst Department.
4. Accounting Department.
5. Advertising Department.
6. Administration Department

**CHAPTER 2**

**INDUSTRIAL EXPERIENCE**

**2.0 Technical Department**

This department was where my Industrial Training took place where I was grounded and expose to the web and software development world especially the creation of websites taking me step by step with practical all through the process.

/////

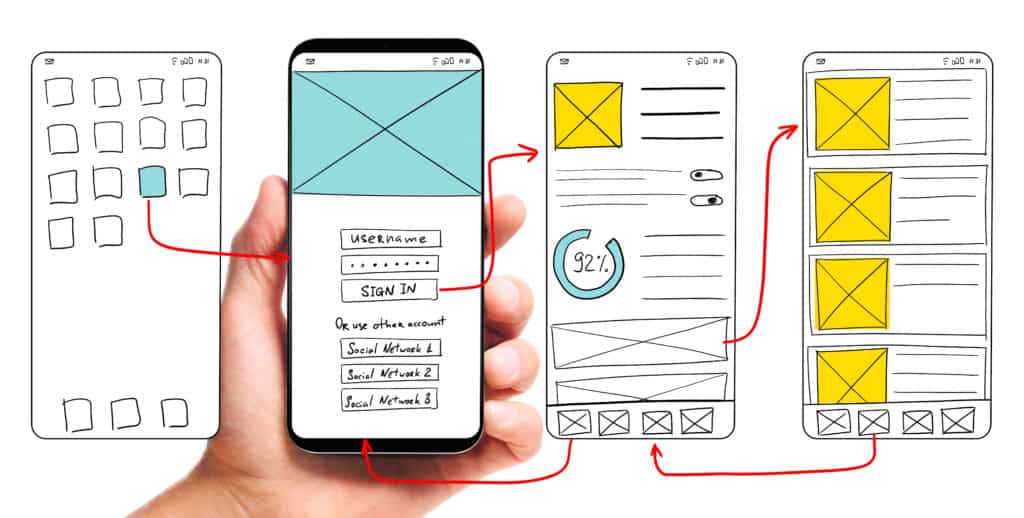
The technical department was where I did my Industrial training. I was taught everything I needed to know about UI/UX (User interaction & User Experience) and Graphic designs. At a certain point in time OneKiosk was in need of a mobile app for android and ios and I was asked if I could make research on a cross-platform solution, I found flutter and I was given the go ahead to learn flutter and make a prototype app . This also lead me to join the software engineers where I was also taught about software automation tools (like Git and Codemagic) and some key concept about API’s.

**2.1 UI/UX Design Overview**

UX design refers to the term “user experience design”, while UI stands for “user interface design”. Both elements are crucial to a product and work closely together. But despite their professional relationship, the roles themselves are quite different, referring to very different aspects of the product development process and the design discipline. Before I consider the key differences between UX and UI, I would like to first define what each term means individually.

**2.1 User Experience (UX) Design**

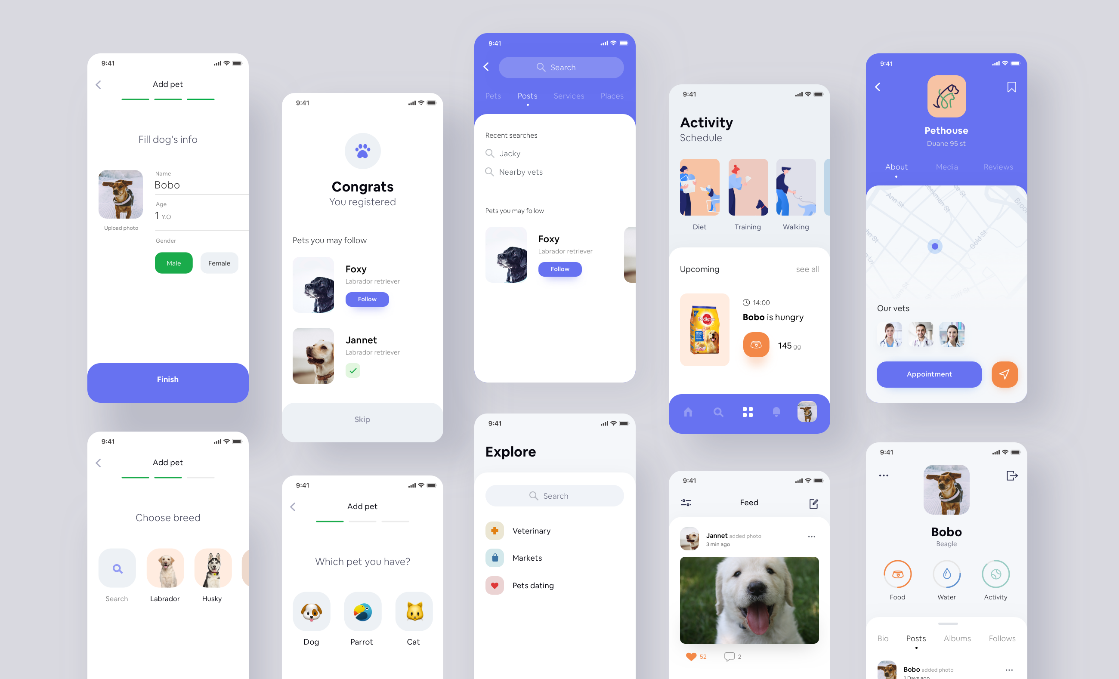
User experience (abbreviated as UX) is how a person feels when interfacing with a system. The system could be a website, a web application or desktop software and, in modern contexts, is generally denoted by some form of human-computer interaction (HCI).

UX designers explore many different approaches to solving a specific user problem and ensure that the product logically flows from one step to the next. One way that a UX designer might do this is by conducting in-person user tests to observe one’s behaviours. By identifying verbal and non-verbal stumbling blocks, they refine and iterate to create the “best” user experience. 

**2.2 User Interface (UI) Design**

Unlike UX designers who are concerned with the overall feel of the product (digital products e.g. apps), user interface designers are particular about **how the product is laid out**,their goal is to ensure user’s interaction is as simple and efficient as possible . They oversee designing of each screen or page with which a user interacts and ensuring that the UI visually communicates the path that a UX designer has laid out.

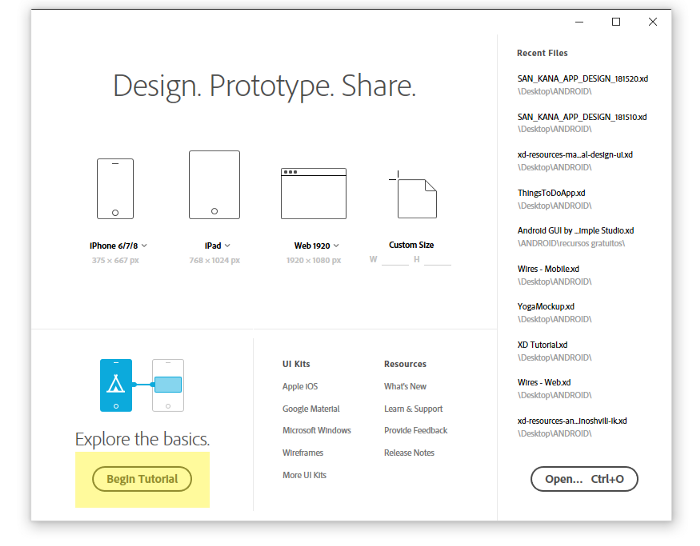
UI designers are also typically responsible for creating a cohesive style guide and ensuring that a consistent design language is applied across the product. Maintaining consistency in visual elements and defining behaviours such as color to use, typography and fonts, how to display error or warning states etc. Some of the tools UI designers use are: Illustrator, sketch, Adobe Xd etc.



**2.4 Adobe XD UI/UX Design Applications**

At OneKiosk I was taught Adobe XD, one of the most comprehensive and reliable UI/UX design applications used to build a variety of wireframes, prototypes, and graphics. It free to install on both Mac and Windows systems. There are other software that those similar things as Adobe XD e.g. Sketch, Figma, Gravity etc.

When launching the app, the welcome page provides different standard screen size templates and adds the file size you set yourself. In addition, the welcome page contains a number of resources; these resources can be used as a starting point for learning this application, as well as UI design elements such as iOS, Android, Web, and Desktop devices.



Adobe XD is divided into two (2) view, design view and prototype view, accessible from the top left of the app. In design view, designers can use many tools and features to create design layout pages and import resources from other application related to Adobe.

The second view in Adobe XD is the prototype view, which allows designers to create connections between artboards (pages) and to define the connections between them through interaction. Once the Project is complete, the designer can preview it by clicking the preview icon in the top right corner or share it with the team via a web-based link. Designers can also save different pages in a bitmap or vector-based format.

**2.4 Adobe XD Tools and Features**

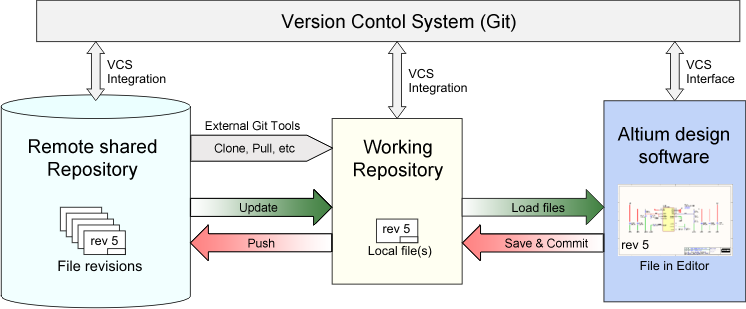
Adobe XD has very few but power full tools that can be used to make any prototype, wireframes specifically for UI/UX design. Many of the tools looks similar to other adobe product but like photoshop but they have been reimagined.

* **To Pan:** use your trackpad, or press spacebar + mouse, or press the mouse wheel.
* **To Zoom:** two-finger pinch trackpad, or Ctrl(Cmd)+mouse wheel, or Ctrl(Cmd)+(+/-). You can also zoom to a selected item by pressing Ctrl(Cmd)+3.
* **To Select:** use the left click of the mouse. To select groups inside a rectangular area, click and drag.
* **Pen Tool:** located on the tool panel, it is used to draw customs shapes of different sizes.
* **Text Tool:** located on the tool panel, it is used to select, edit, and write texts.
* **Rectangle Tool:** located on the tool panel, it is used to create rectangle shapes.
* **Ellipse Tool:** located on the tool panel, it is used to create oval and ellipse shapes.
* **Polygon Tool:** located on the tool panel, it is used to create rectangles, and shapes with more than four (4) corners.
* **Line Tool:** located on the tool panel, it is used to create oval and ellipse shapes.
* **Artboard Tool:** located on the tool panel, it is used to create new layout page of custom shapes and sizes.

**2.4 Git and GitHub (Automation tools)**

Git is a free, open-source **version control software**. It was created by Linus Torvalds in 2005. This tool is a version control system that was initially developed to work with several developers on the Linux kernel.

This basically means that Git is a content tracker. So Git can be used to store content — and it is mostly used to store code because of the other features it provides. Real life projects generally have multiple developers working in parallel, so they need a version control system like Git to make sure that there are no code conflicts between them. Also, the requirements in such projects change often. So a version control system allows developers to revert and go back to an older version of their code. The branch system in Git allows developers to work individually on a task (For example: One branch -> One task OR One branch -> One developer). Basically, think of Git as a small software application that controls your code base, if you’re a developer.



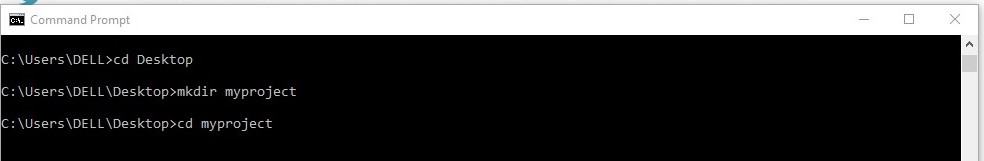
**2.4.1 Install Git and create a GitHub account**

Follow the instructions [here](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git) to install git (if it's not already installed). Please note that for this explanation I will be using git on the command line only. While there are some great git GUIs (graphical user interfaces), I was thought via the command line and I think it's easier to learn git using git-specific commands first and then to try out a git GUI once you're more comfortable with the command. After that we then create a GitHub account [here](https://github.com/join).  (Accounts are free for public repositories, but there's a charge for private repositories.)

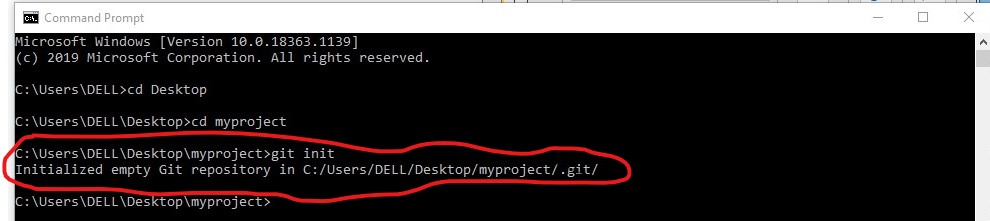
**2.4.1 Creating a local git repository**

When creating a new project on a local machine using git, you'll first create a new **repository**(or often, '**repo**', for short).

To use git, we'll be using the terminal. To begin, open a terminal and move to where you want to place the project on your local machine using the cd (change directory) command. For example, if you have a 'projects' folder on your desktop, you'd do something like:



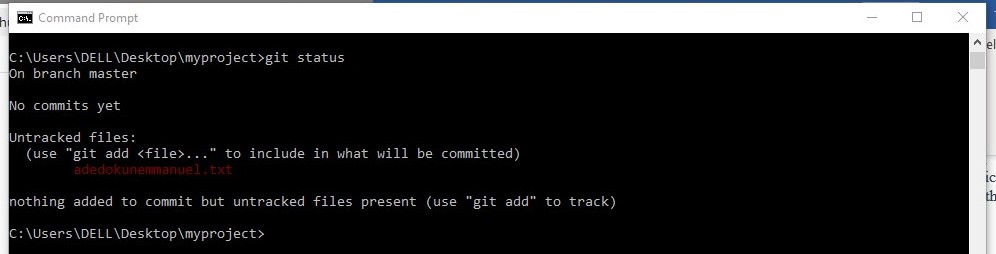
To initialize a git repository in the root of the folder, run the “git init”command:



Step 2: Add a new file to the repo:

Go ahead and add a new file to the project, using any text editor you like or running a **touch** command. Once you've added or modified files in a folder containing a git repo, git will notice that changes have been made inside the repo. But git won't officially keep track of the file (that is, put it in a commit - we'll talk more about commits next) unless you explicitly tell it to.

After creating the new file, you can use the git status command to see which files git knows exist.



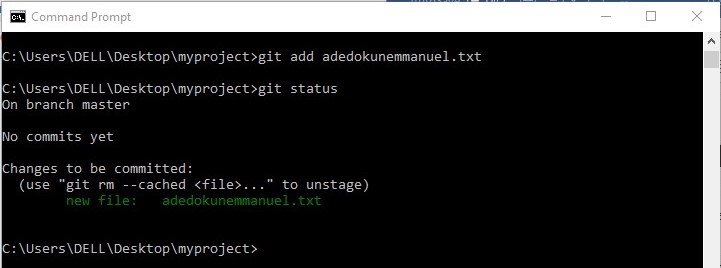
What this basically says is, “that git noticed you added a new file name adedokunemmanuel.txt, but unless you use the ‘git add’ command we are not going to do anything with it.”

A **commit**is a record of what files you have changed since the last time you made a commit. Essentially, you make changes to your repo (for example, adding a file or modifying one) and then tell git to put those files into a commit. Commits make up the essence of your project and allow you to go back to the state of a project at any point.

To add a file to a commit, you first need to add it to the staging environment. To do this, you can use the [**git add**](http://git-scm.com/docs/git-add)**<filename>**command (see Step 3 below).

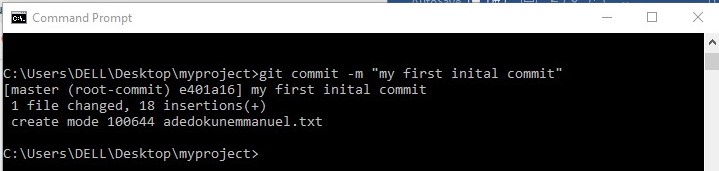
Step 3: Add a file to the staging environment:

Add a file to the staging environment using the **git add** command. If you rerun the git status command, you'll see that git has added the file to the staging environment (notice the "Changes to be committed" line).



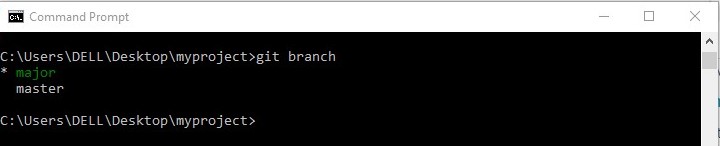
Step 4: Create a commit:

Run the command **git commit -m “Your message about the commit”**



Step 5: Create a new branch

Branches allow you to move back and forth between 'states' of a project. For instance, if you want to add a new page to your website you can create a new branch just for that page without affecting the main part of the project. Once you're done with the page, you can [**merge**](http://git-scm.com/docs/git-merge) your changes from your branch into the primary branch. When you create a new branch, Git keeps track of which commit your branch 'branched' off of, so it knows the history behind all the files. Let's say you are on the primary branch and want to create a new branch to develop your web page. Here's what you'll do: Run[**git checkout -b <my branch name>**](http://git-scm.com/docs/git-checkout). This command will automatically create a new branch and then 'check you out' on it, meaning git will move you to that branch, off of the primary branch.After running the above command, you can use the [**git branch**](http://git-scm.com/docs/git-branch) command to confirm that your branch was created:



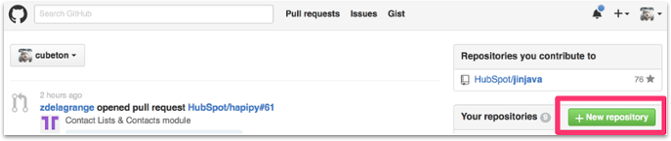
The branch name with the asterisk next to it indicates which branch you're pointed to at that given time.

Now, if you switch back to the primary branch and make some more commits, your new branch won't see any of those changes until you [**merge**](http://git-scm.com/docs/git-merge) those changes onto your new branch.

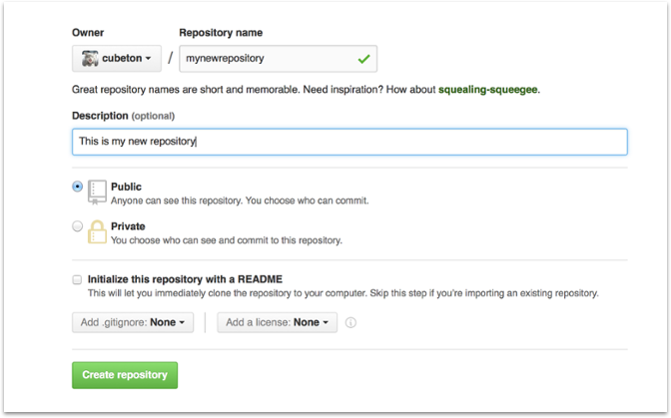
Step 6: Create a new repository on GitHub

If you only want to keep track of your code locally, you don't need to use GitHub. But if you want to work with a team, you can use GitHub to collaboratively modify the project's code.

To create a new repo on GitHub, log in and go to the GitHub home page. You should see a green '+ New repository' button:



After clicking the button, GitHub will ask you to name your repo and provide a brief description:



When you're done filling out the information, press the 'Create repository' button to make your new repo.

GitHub will ask if you want to create a new repo from scratch or if you want to add a repo you have created locally. In this case, since we've already created a new repo locally, we want to push that onto GitHub so follow the **'.... or push an existing repository from the command line'** section:

Step 7: Push a branch to GitHub

Now we'll **push** the commit in your branch to your new GitHub repo. This allows other people to see the changes you've made. If they're approved by the repository's owner, the changes can then be merged into the primary branch.

To push changes onto a new branch on GitHub, you'll want to run [**git push**](http://git-scm.com/docs/git-push)**origin yourbranchname**. GitHub will automatically create the branch for you on the remote repository:



 You might be wondering what that "origin" word means in the command above. What happens is that when you clone a remote repository to your local machine, git creates an **alias** for you. In nearly all cases this alias is called "[**origin**](https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes)." It's essentially shorthand for the remote repository's URL. So, to push your changes to the remote repository, you could've used either the command:**git push git@github.com:git/git.git yourbranchname** or **git push origin yourbranchname**

(If this is your first-time using GitHub locally, it might prompt you to log in with your GitHub username and password.)

If you refresh the GitHub page, you'll see note saying a branch with your name has just been pushed into the repository. You can also click the 'branches' link to see your branch listed there.

**2.4 Flutter and Dart Introduction**

Flutter is open source UI toolkit for building beautiful, natively compiled applications for [mobile](https://flutter.dev/docs), [web](https://flutter.dev/web), and [desktop](https://flutter.dev/desktop) from a single codebase. While Dart is a client optimized programing language that runs on any platform. Both Flutter and Dart are owned by Google.

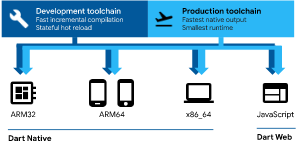
Dart shares features with many of other languages like Kotlin and Swift and it can be easily transpiled into JavaScript code. Thus, Flutter allows for a reactive and declarative style programming, it resembles React Native. There is no need to use a bridge in a Flutter to improve overall performance and start-up time. By using Dart, it will automatically achieve [Ahead-of-Time compilation](https://en.wikipedia.org/wiki/Ahead-of-time_compilation) (AOT).You will not be able to find any differences between an app build in Flutter and other languages.

**2.4.2 What is Dart**

Dart is a client-optimized language for fast app on any platform. You can use Dart to write simple scripts or full-featured apps. Whether you’re creating a mobile app, web app, command-line script, or server-side app, there’s a Dart solution for that.

Flexible compiler technology lets you run Dart code in different ways, depending on your target platform and goals:

* **Dart Native**: For programs targeting devices (mobile, desktop, server, and more), Dart Native includes both a Dart VM with JIT (just-in-time) compilation and an AOT (ahead-of-time) compiler for producing machine code.
* **Dart Web**: For programs targeting the web, Dart Web includes both a development time compiler (dartdevc) and a production time compiler (dart2js).



**2.4.2.1 Dart Language Samples**

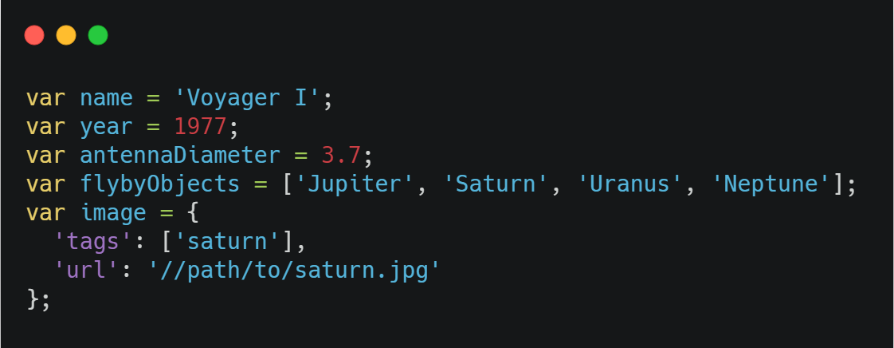
**Hello World**

Every app has a main() function. To display text on the console, you can use the top-level print() function:



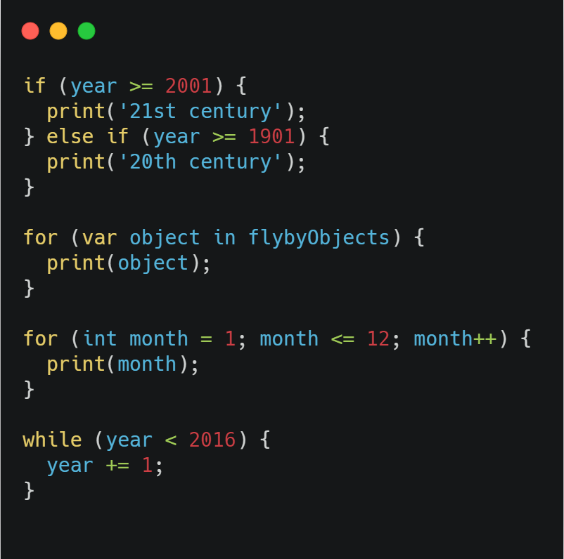
**Variables**

Even in type-safe Dart code, most variables don’t need explicit types, thanks to type inference:



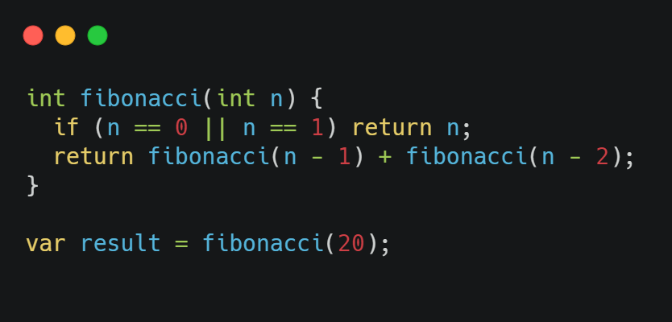
**Control flow statements**

Dart supports the usual control flow statements including break and continues, switch and case and assert.



**Functions**

In Dart programing it is highly recommended to specify types of each function’s arguments and return types:



**Imports**

Dart has a reach comments system which are used to access APIs defined in other libraries:



Dart also includes libraries prefixes, show, hide, and lazy loading through the deferred keyword.

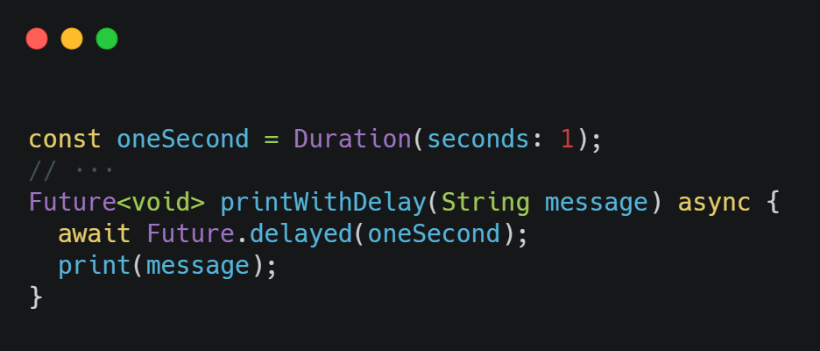
**Classes**

Here’s an example of a class with three properties, two constructors, and a method. One of the properties can’t be set directly, so it’s defined using a getter method (instead of a variable). Other properties about Dart classes not mentioned are initializer lists, optional new and const, redirecting constructors. Factory constructors, getters, setters and much more.



**Async**

Dart supports both synchronous and asynchronous events in a way that avoid callback hell and make your code much more readable by using async and await keywords.



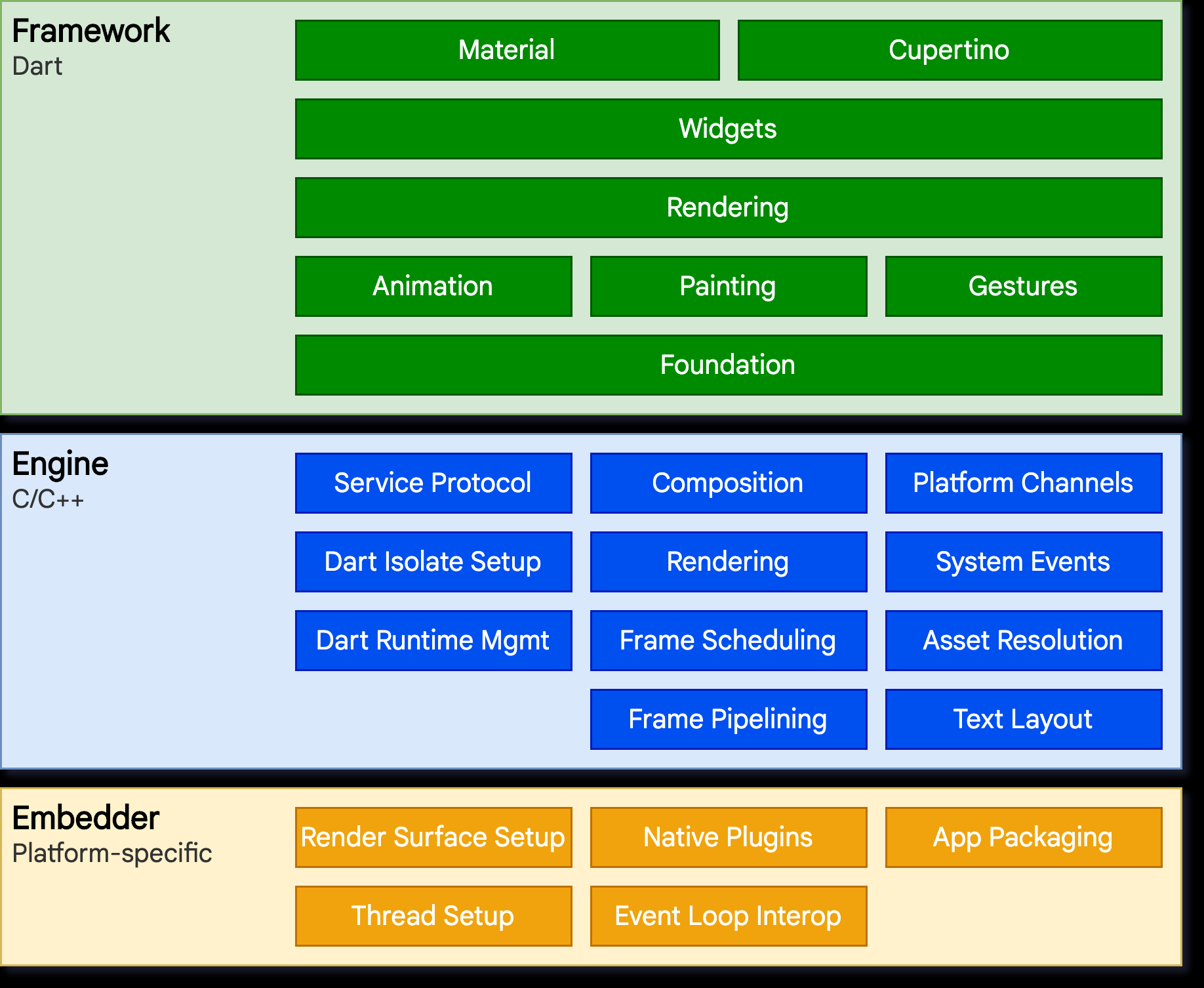
Dart has many more functionalities and properties which are not covered in this collections of samples, some of them include comments, inheritance, mixins, interfaces and abstract classes, exceptions e.t.c.

**2.4.3 Flutter and it’s Architectural**

Flutter is a cross-platform UI toolkit that is designed to allow code reuse across operating systems such as iOS and Android, while also allowing applications to interface directly with underlying platform services. The goal is to enable developers to deliver high-performance apps that feel natural on different platforms, embracing differences where they exist while sharing as much code as possible.

During development, Flutter apps run in a VM that offers stateful hot reload of changes without needing a full recompile. For release, Flutter apps are compiled directly to machine code, whether Intel x64 or ARM instructions, or to JavaScript if targeting the web. The framework is open source, with a permissive BSD license, and has a thriving ecosystem of third-party packages that supplement the core library functionality.

Flutter is designed as an extensible, layered system. It exists as a series of independent libraries that each depend on the underlying layer. No layer has privileged access to the layer below, and every part of the framework level is designed to be optional and replaceable.



To the underlying operating system, Flutter applications are packaged in the same way as any other native application. A platform-specific embedder provides an entry point; coordinates with the underlying operating system for access to services like rendering surfaces, accessibility, and input; and manages the message event loop. The embedder is written in a language that is appropriate for the platform: currently Java and C++ for Android, Objective-C/Objective-C++ for iOS and macOS, and C++ for Windows and Linux. Using the embedder, Flutter code can be integrated into an existing application as a module, or the code may be the entire content of the application. Flutter includes a number of embedders for common target platforms, but [other embedders also exist](https://hover.build/blog/one-year-in/).

At the core of Flutter is the **Flutter engine**, which is mostly written in C++ and supports the primitives necessary to support all Flutter applications. The engine is responsible for rasterizing composited scenes whenever a new frame needs to be painted. It provides the low-level implementation of Flutter’s core API, including graphics (through [Skia](https://skia.org/)), text layout, file and network I/O, accessibility support, plugin architecture, and a Dart runtime and compile toolchain.

The engine is exposed to the Flutter framework through [dart:ui](https://github.com/flutter/engine/tree/master/lib/ui), which wraps the underlying C++ code in Dart classes. This library exposes the lowest-level primitives, such as classes for driving input, graphics, and text rendering subsystems.

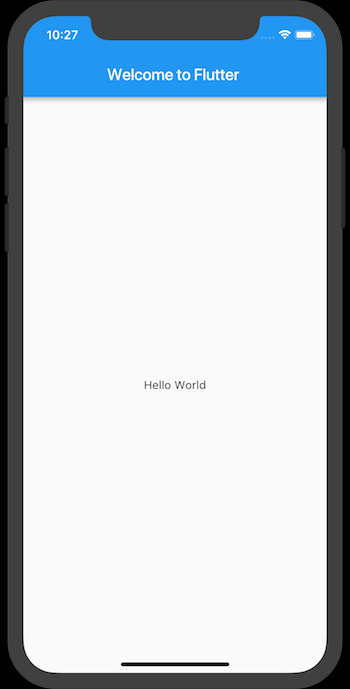
Typically, developers interact with Flutter through the **Flutter framework**, which provides a modern, reactive framework written in the Dart language. It includes a rich set of platform, layout, and foundational libraries, composed of a series of layers.

**2.4.5.1 Flutter Framework Samples**

Assuming all flutter installation processes are done or [click here](https://flutter.dev/docs/get-started/test-drive#create-app) to setup your computer to run flutter codes. By default, flutter generate some sample code in **lib/main.dart,** replace does code with the following codes, which displays “Hello world” in the center of the screen.



Run the app the way your IDE describes. You should see either Android, iOS or web output depending on your devices.



**Observations**

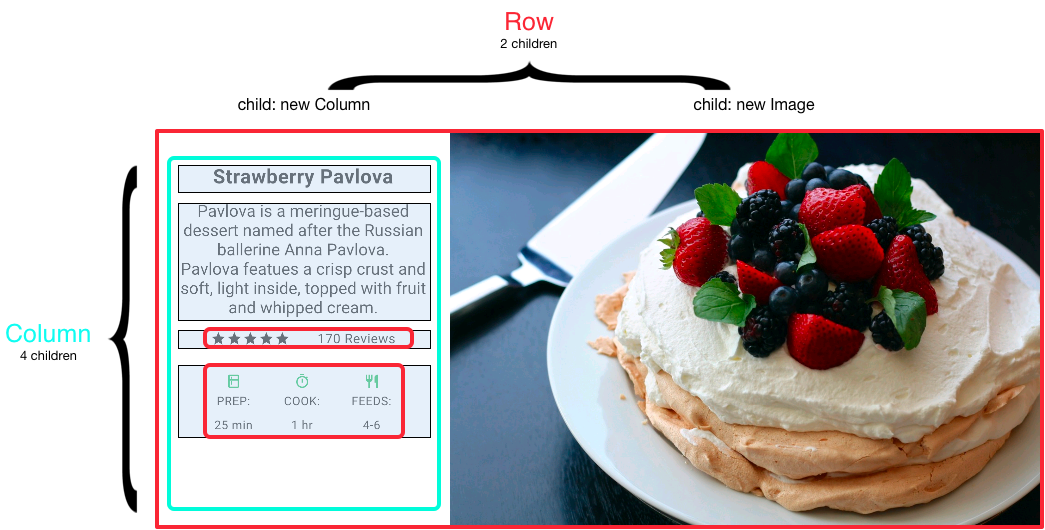
* This example creates a Material app. [*Material*](https://material.io/guidelines) is a visual design language that is standard on mobile and the web. Flutter offers a rich set of Material widgets. It’s a good idea to have a uses-material-design: true entry in the flutter section of your *pubspec.yaml* file. This will allow you to use more features of Material, such as their set of predefined [*Icons*](https://design.google.com/icons/)*.*
* The *main()* method uses arrow (=>) notation. Use arrow notation for one-line functions or methods.
* The app extends *StatelessWidget*, which makes the app itself a widget. In Flutter, almost everything is a widget, including alignment, padding, and layout.
* The Scaffold widget, from the Material library, provides a default app bar, and a body property that holds the widget tree for the home screen. The widget subtree can be quite complex.
* A widget’s main job is to provide a *build()* method that describes how to display the widget in terms of other, lower level widgets.
* The body for this example consists of a *Center* widget containing a Text child widget. The *Center* widget aligns its widget subtree to the center of the screen.

**Layouts in Flutter**

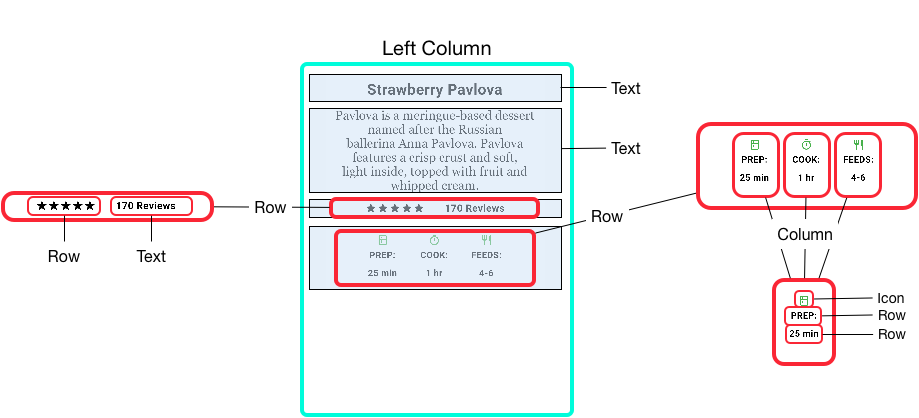
One of the most common layout patterns is to arrange widgets vertically or horizontally. You can use a Row widget to arrange widgets horizontally, and a Column widget to arrange widgets vertically.

To create a row or column in Flutter, you add a list of children widgets to a [*Row*](https://api.flutter.dev/flutter/widgets/Row-class.html) or [*Column*](https://api.flutter.dev/flutter/widgets/Column-class.html) widget. In turn, each child can itself be a row or column, and so on. The following example shows how it is possible to nest rows or columns inside of rows or columns.

This layout is organized as a *Row*. The row contains two children: a column on the left, and an image on the right:



The left column’s widget trees nest rows and columns.



Examples with codes.





 Note: *Row* and *Column* are basic primitive widgets for horizontal and vertical layouts—these low-level widgets allow for maximum customization. Flutter also offers specialized, higher level widgets that might be sufficient other needs. For example, instead of *Row* you might prefer [*ListTile*](https://api.flutter.dev/flutter/material/ListTile-class.html), an easy-to-use widget with properties for leading and trailing icons, and up to 3 lines of text. Instead of *Column*, you might prefer [*ListView*](https://api.flutter.dev/flutter/widgets/ListView-class.html), a column-like layout that automatically scrolls if its content is too long to fit the available space.

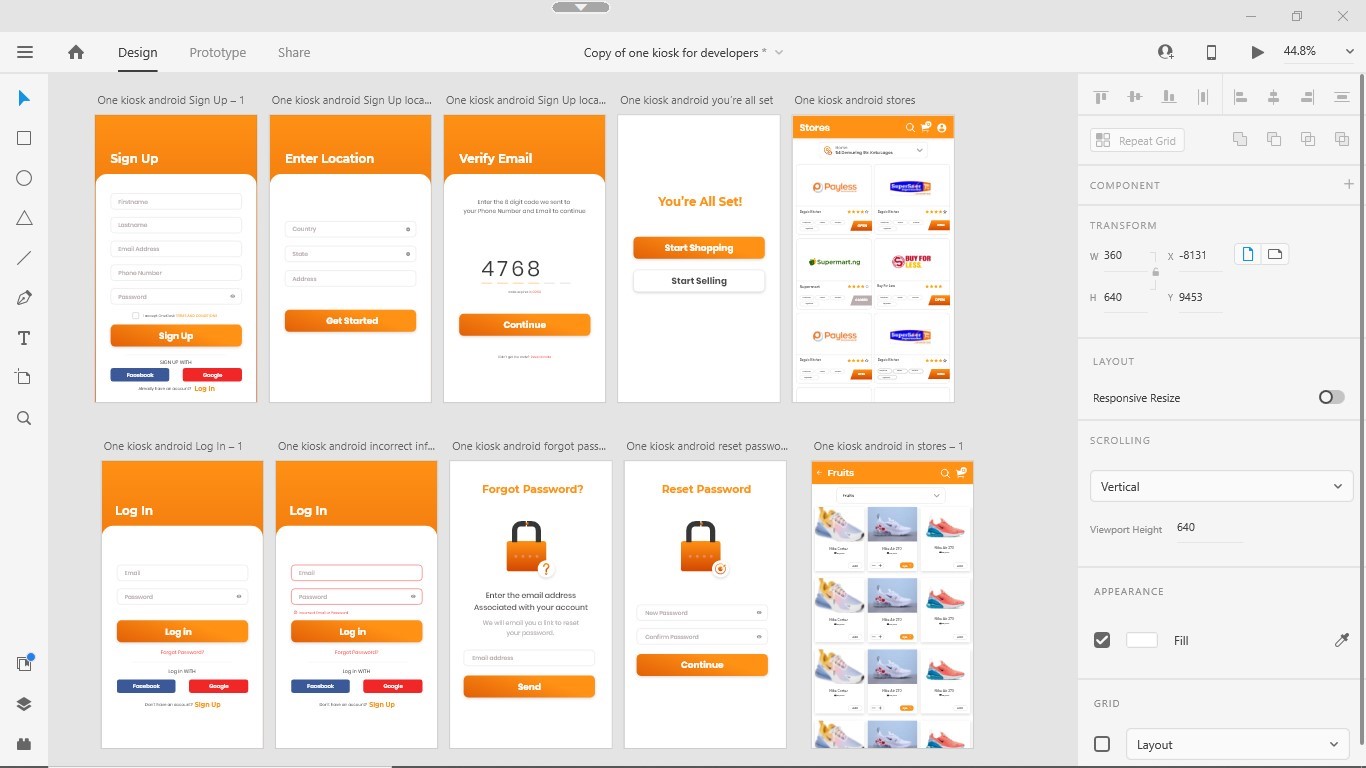
**CHAPTER 4**

**INDUSTRIAL EXPERIENCE**

**4.0 PERSONAL INPUT TO THE COMPANY (ONEKIOSK AFRICA)**

**4.0.1 Input as a UI/UX Designer**

After I became good at UI/UX Design, I was given the opportunity to design some UI for our mobile app division.



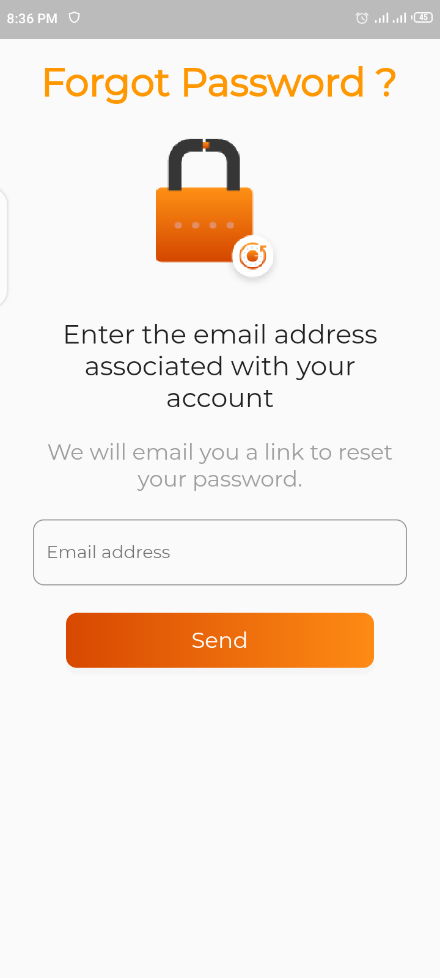
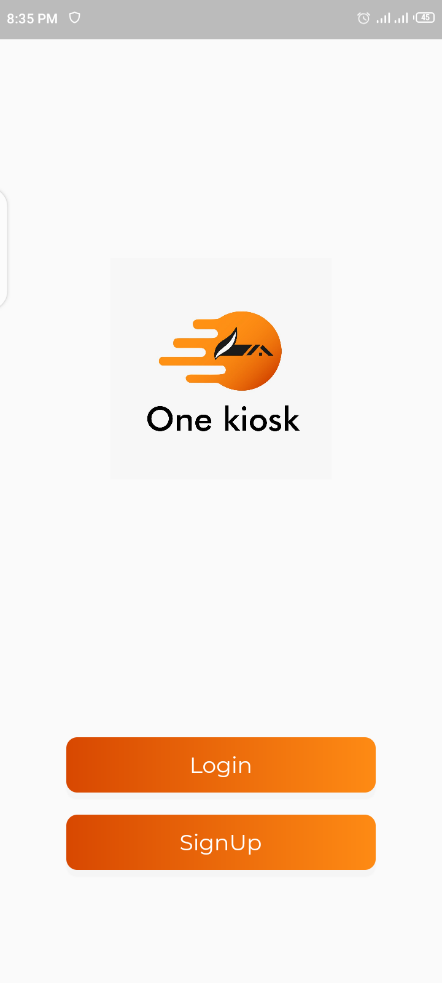
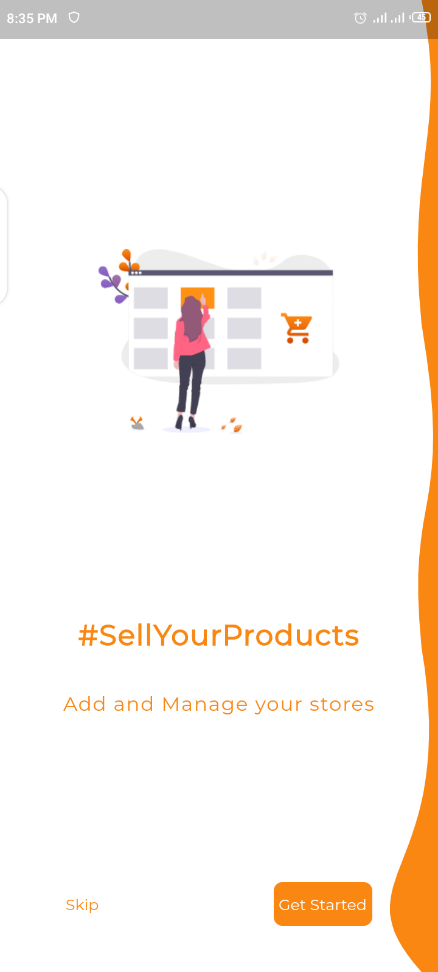
While the pressure was becoming much on our designer, I was tasked to design social media graphic During the lockdown while we work remotely.

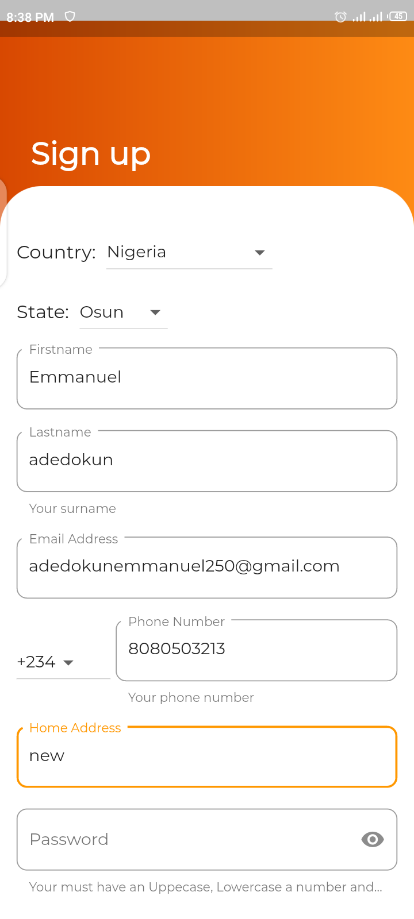
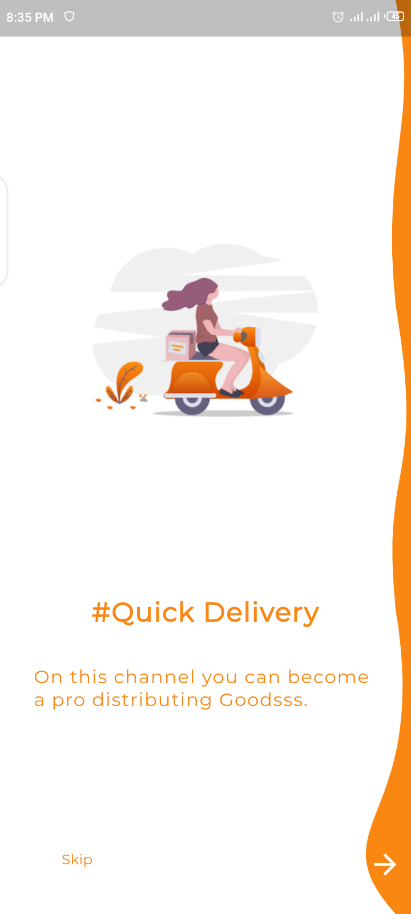
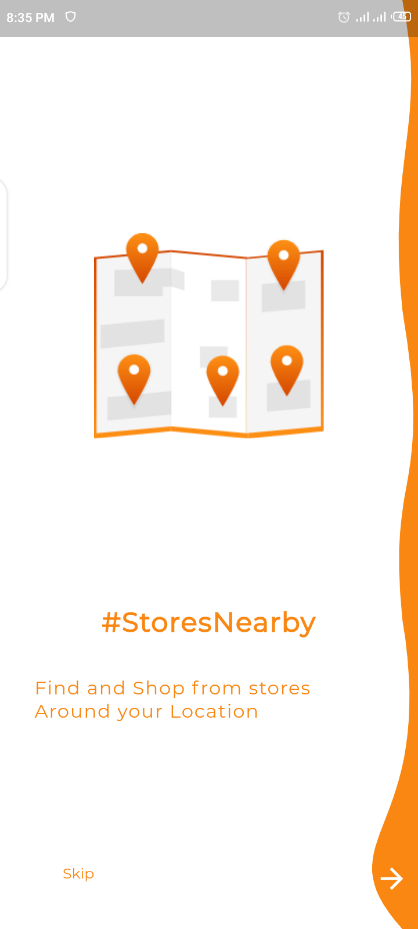


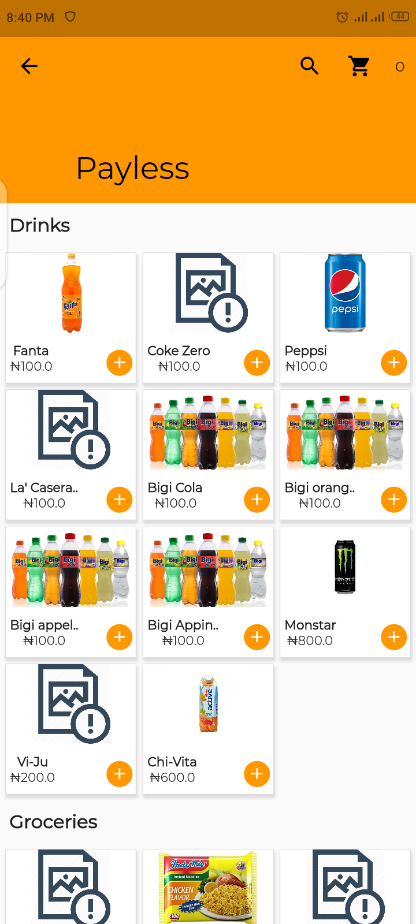
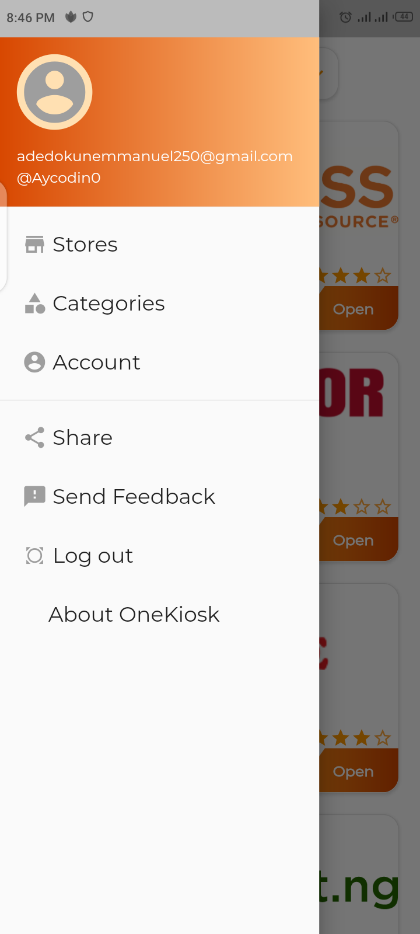
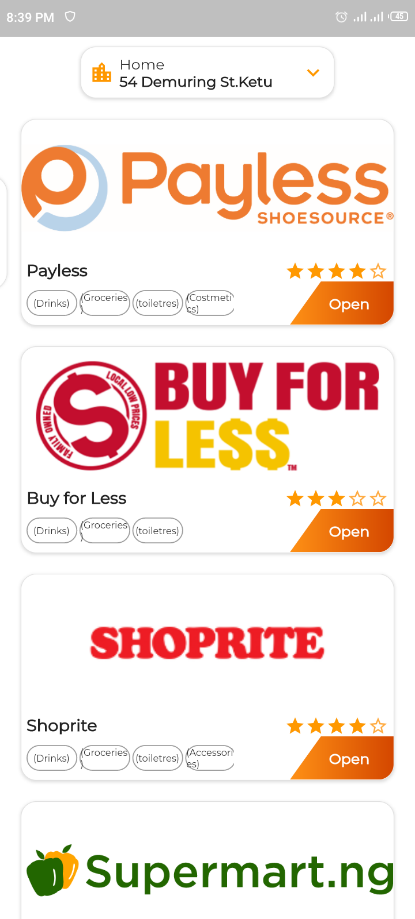
**4.0.1 Input as a Mobil app Developer**

As I have mentioned earlier, I was given the task to make research on Flutter framework and see if it was fit for cross platform app development on both iOS and Android. It turns out Flutter was good, far better, and easier learning curve than its competitors. After all my learning and research about Flutter I was able to come up with a four (4) screen prototype which I showed senior developers and tested virtually on my Android and iOS emulators. This success then leads me to continue making the full prototype application life cycle that we the designers had designed in Adobe XD.

Here are some screen shot of the working prototypes I made all in Flutter:







**CHAPTER 5**

**CONCLUSION AND RECOMMENDATIONS**

**5.0 SUMMARY**

Despite covid-19 lockdown and the challenges of working from home I was able to complete my student industrial work experience scheme at OneKiosk. The six-month program has exposed me to knowing more practical in-depth knowledge about UI/UX design, Software Engineering and most importantly how to work and communicating effectively with teams.

It has become obvious that a new generation with a new way of thinking about technology has emerged. Technology has helped us automate and solve problems we had think were impossible to solve 10 years back. One needs to constantly updates his/her knowledge about technology as it is always evolving with new drive to change and improve our ways of living.

**4.1 CONCLUSION**

The training experience is very mandatory and important for all students in tertiary institution like science and technology, engineering, etc. as it exposes us to new thinking about the real world. It should be encouraged at any time in all institution of learning. As this fields helps us to find solutions and if one can look for problem desperate in need of a solution in our society, chances are technology has way of solving the problem and enriching the brain behind it.

**4.2 RECOMMENDATIONS:**

I recommend that the (I.T) program its continuity in all tertiary institution because it help so many students in practical aspect and academic performance as well as work experience.

In other to make this SIWES training easy, student should look for interested place where they are practical orient.

I also recommended that the government and the school authority should assist the student in securing a good place for their (I.T) program, because some students found it difficult in securing a place.

////// I think we need to doo more in the flutter input section by atlist showing file structure and some code snippet /

//// recommendation section nmm