**Advanced Programming**

John Mobile App Report - (Flutter)

Phillip Cole – 1507146

University of Technology Jamaica

Tutor: Gilroy Gordon

Contents

[Singleton & Factory Pattern 3](#_Toc6225319)

[Package Management Tool 4](#_Toc6225320)

[Unit Testing 5](#_Toc6225321)

[Source Control Management Tool 6](#_Toc6225322)

[Code Generation Tool 7](#_Toc6225323)

[Model-View-Controller & Repository Pattern 9](#_Toc6225324)

[Continuous Integration Server 10](#_Toc6225326)

[References 11](#_Toc6225327)

### Singleton & Factory Pattern

Both of these are utilized hand in hand, to make sure there is only one instance of a firebase is created and used.

E.g.

**static final** FirebaseFirestoreService *\_instance* = **new** FirebaseFirestoreService.internal();  
  
 **factory** FirebaseFirestoreService()=> *\_instance*;  
  
 FirebaseFirestoreService.internal();

### Package Management Tool

In flutter it has a built-in package management tool. For a package to be installed for use in the project the dependencies have to be added to the pubspec.yaml file and then imported for use in the required .dart files.

E.g.

**dependencies**:  
 **flutter**:  
 **sdk**: flutter  
 **carousel\_pro**: ^0.0.13  
 **cloud\_firestore**: ^0.9.7  
 **location**: ^1.4.1  
 **google\_maps\_flutter**: ^0.5.3  
 **geoflutterfire**: ^2.0.3+2  
 **numberpicker**: ^1.0.0

**import 'package:location/location.dart'**;  
**import 'package:geoflutterfire/geoflutterfire.dart'**;

**import 'package:cloud\_firestore/cloud\_firestore.dart'**;

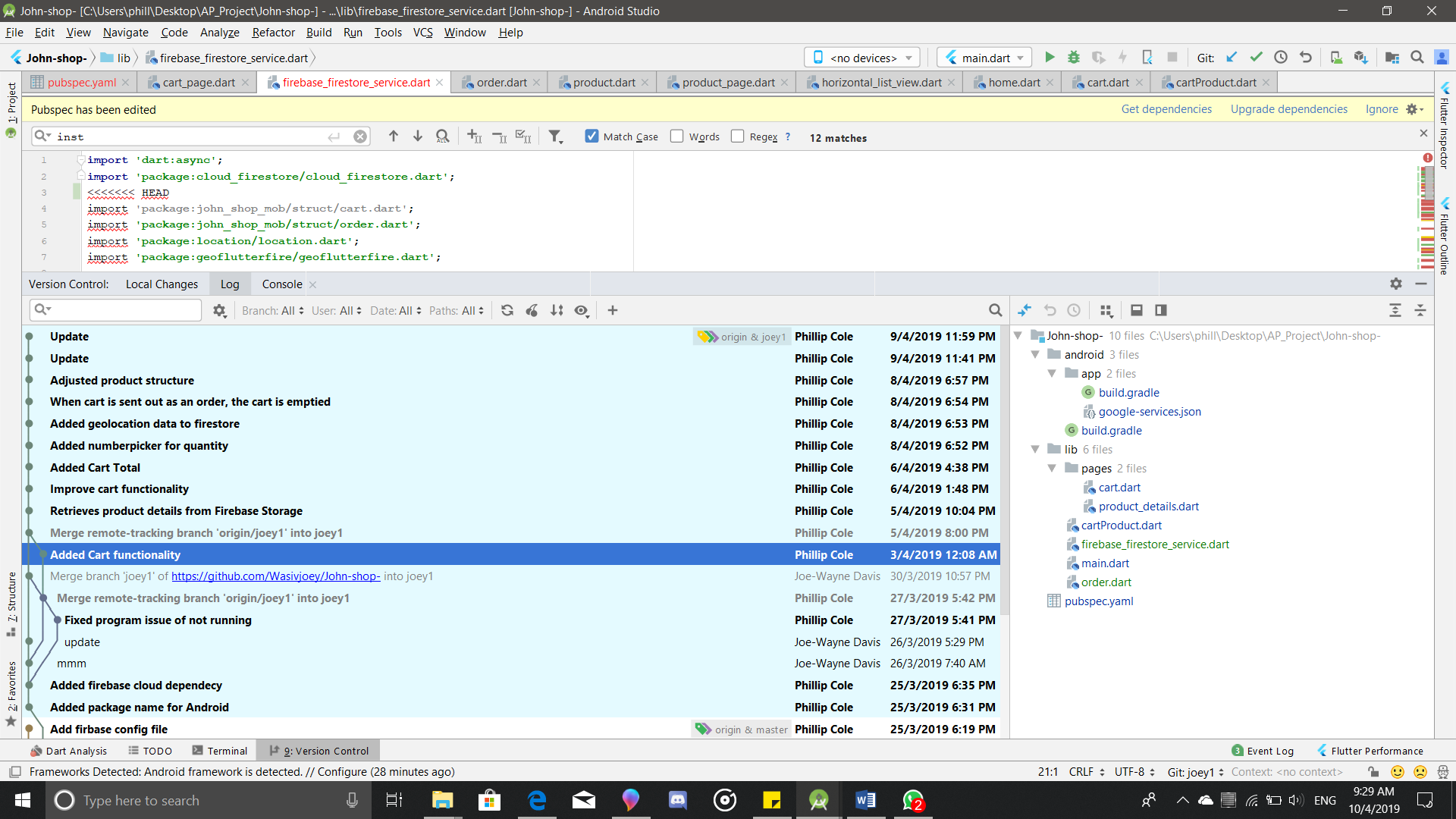
### Unit Testing

The instance of Unit Testing used within this project would be for a password and email form validation which include making sure that the fields are not empty as well as making sure email field matches the email pattern among others.

test(**'empty email returns error string'**, () {  
  
 **final** result = EmailFieldValidator.*validate*(**''**);  
 expect(result, **'Email can\'t be empty'**);  
});  
  
test(**'non-empty email returns null'**, () {  
  
 **final** result = EmailFieldValidator.*validate*(**'email'**);  
 expect(result, **null**);  
});  
  
test(**'empty password returns error string'**, () {  
  
 **final** result = PasswordFieldValidator.*validate*(**''**);  
 expect(result, **'Password can\'t be empty'**);  
});  
  
test(**'non-empty password returns null'**, () {  
  
 **final** result = PasswordFieldValidator.*validate*(**'password'**);  
 expect(result, **null**);  
});

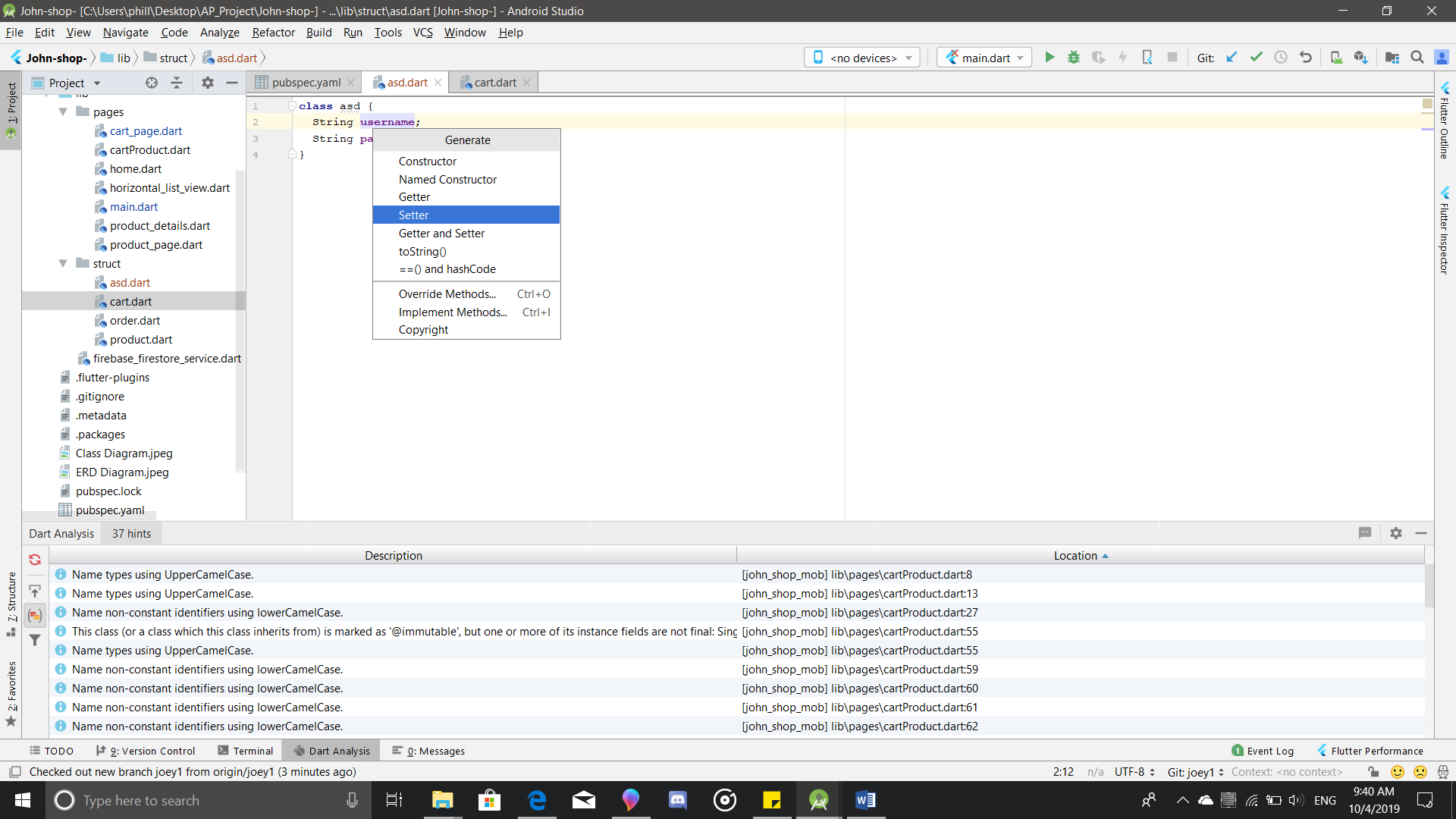
### Source Control Management Tool

Github was used as the source control management tool to easier facilitate cooperation and ease of testing.

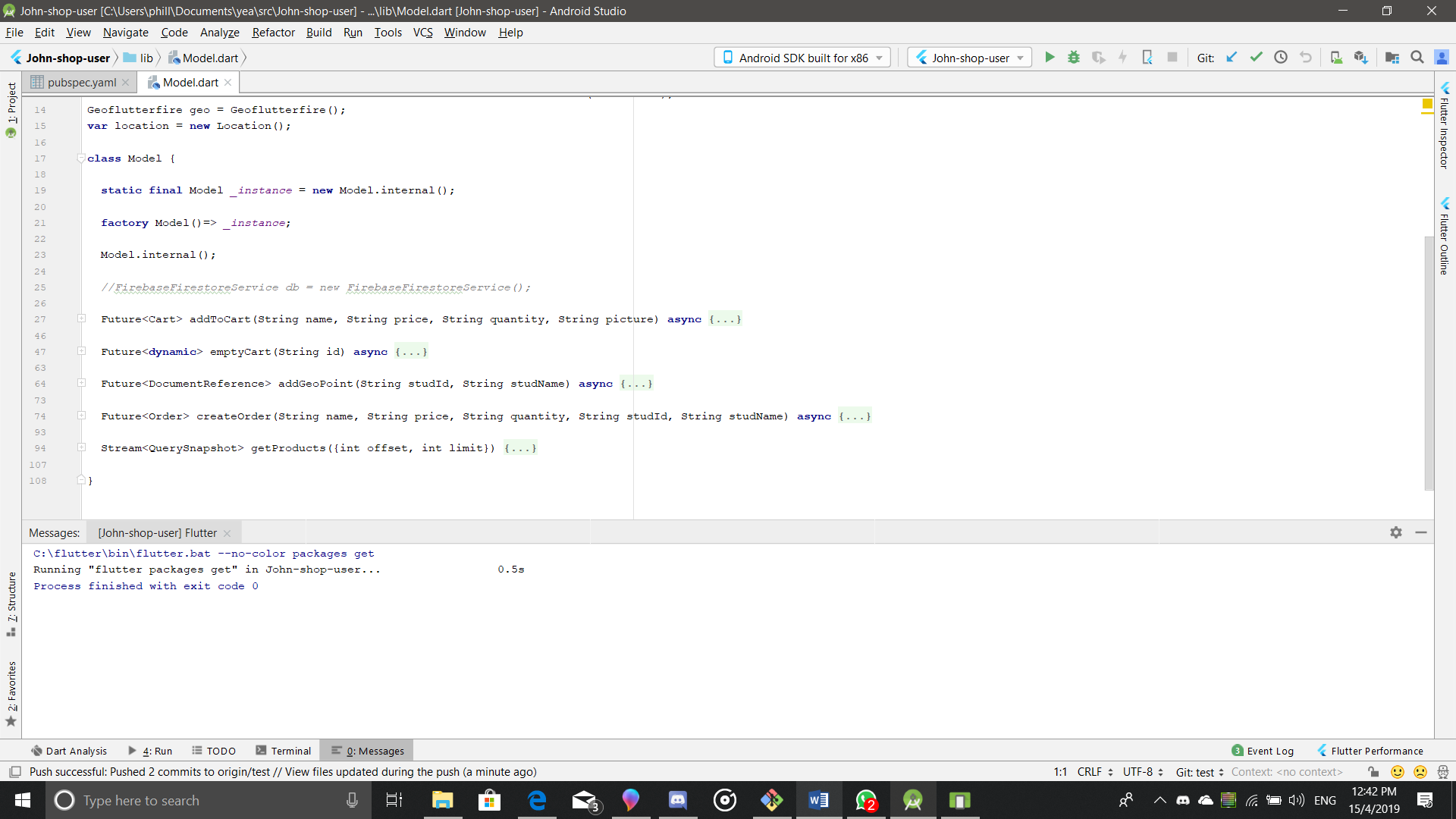


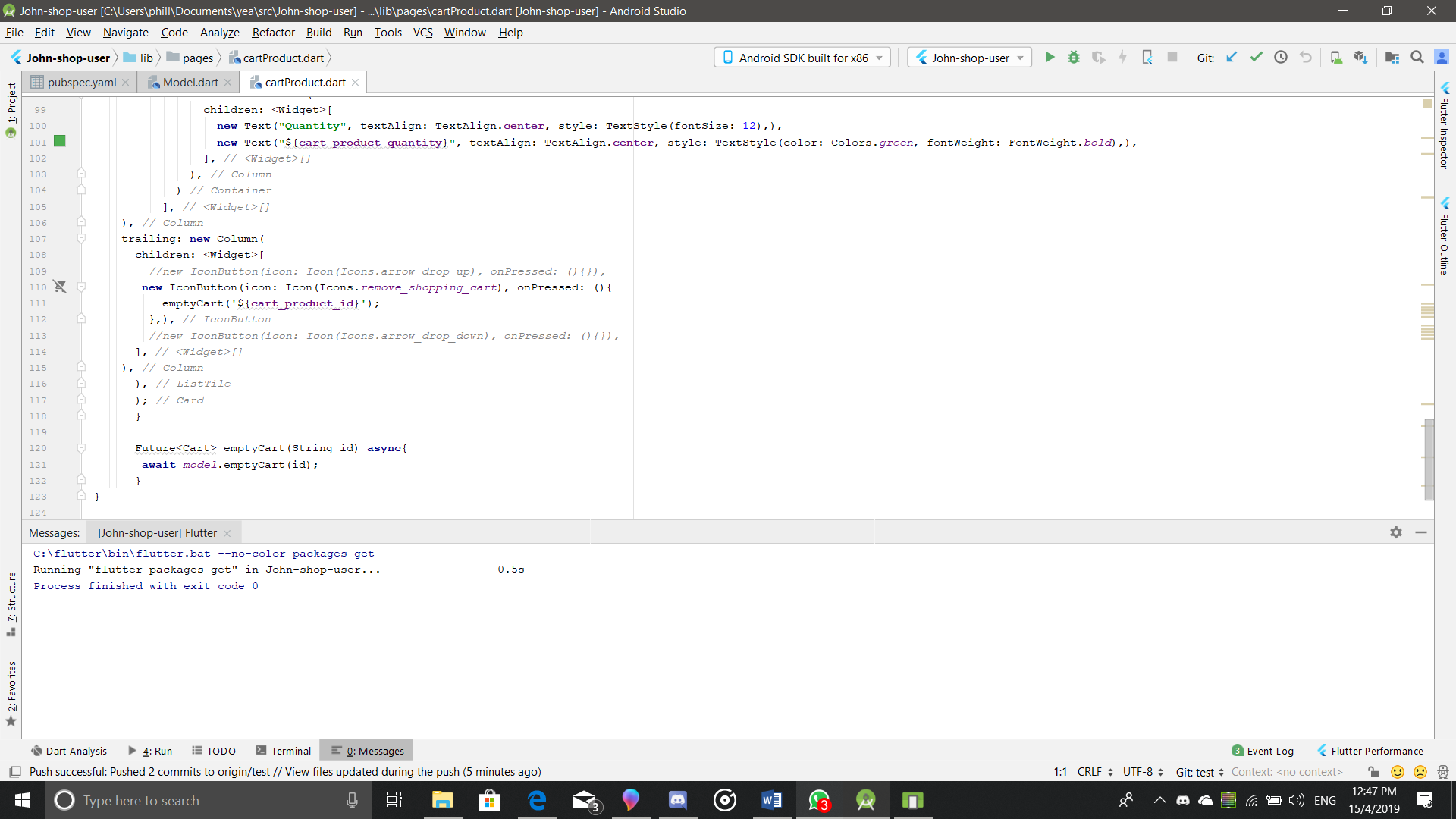
### Code Generation Tool

The code generation tool was used within the environment of Android Studio to make code such as constructors and getters and setters throughout this project.



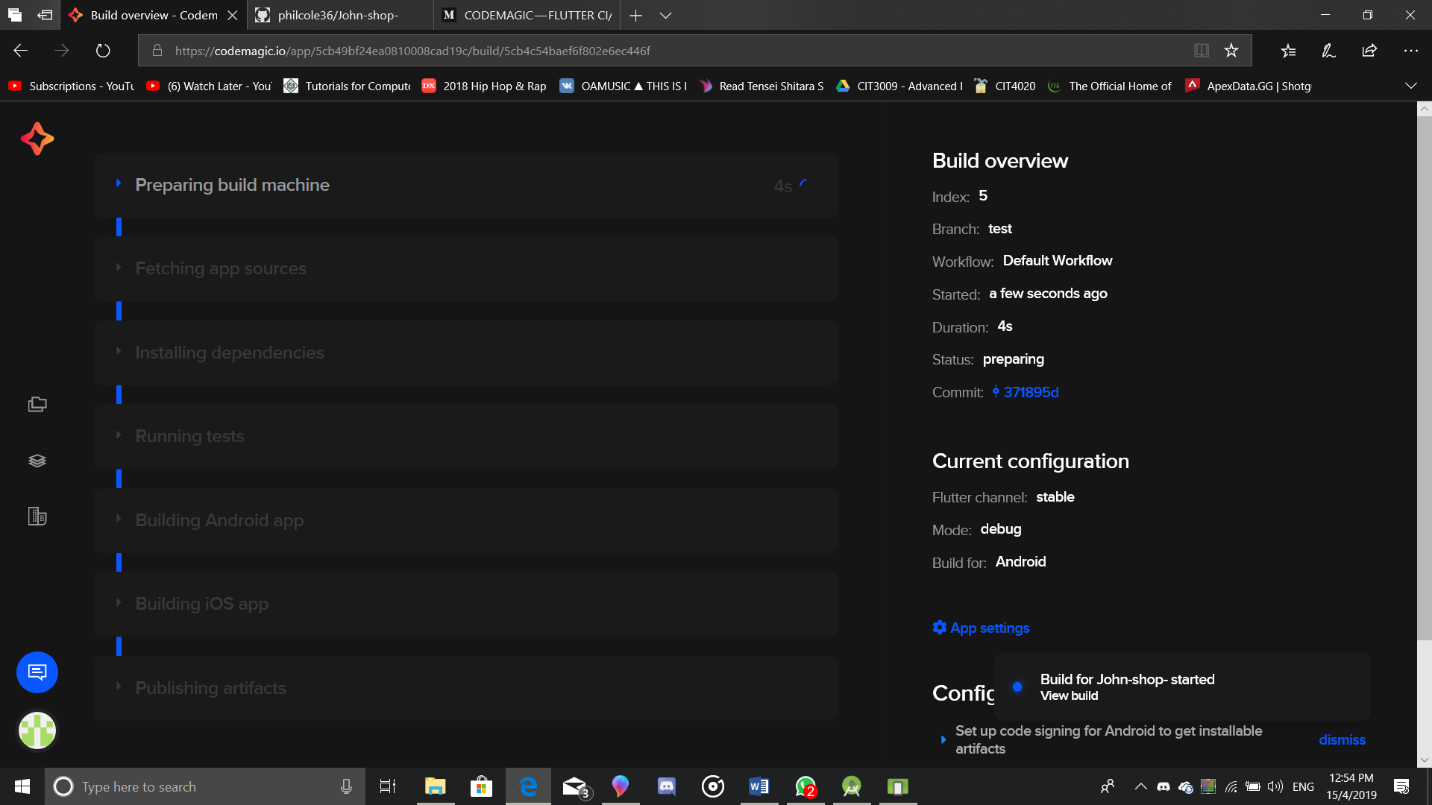
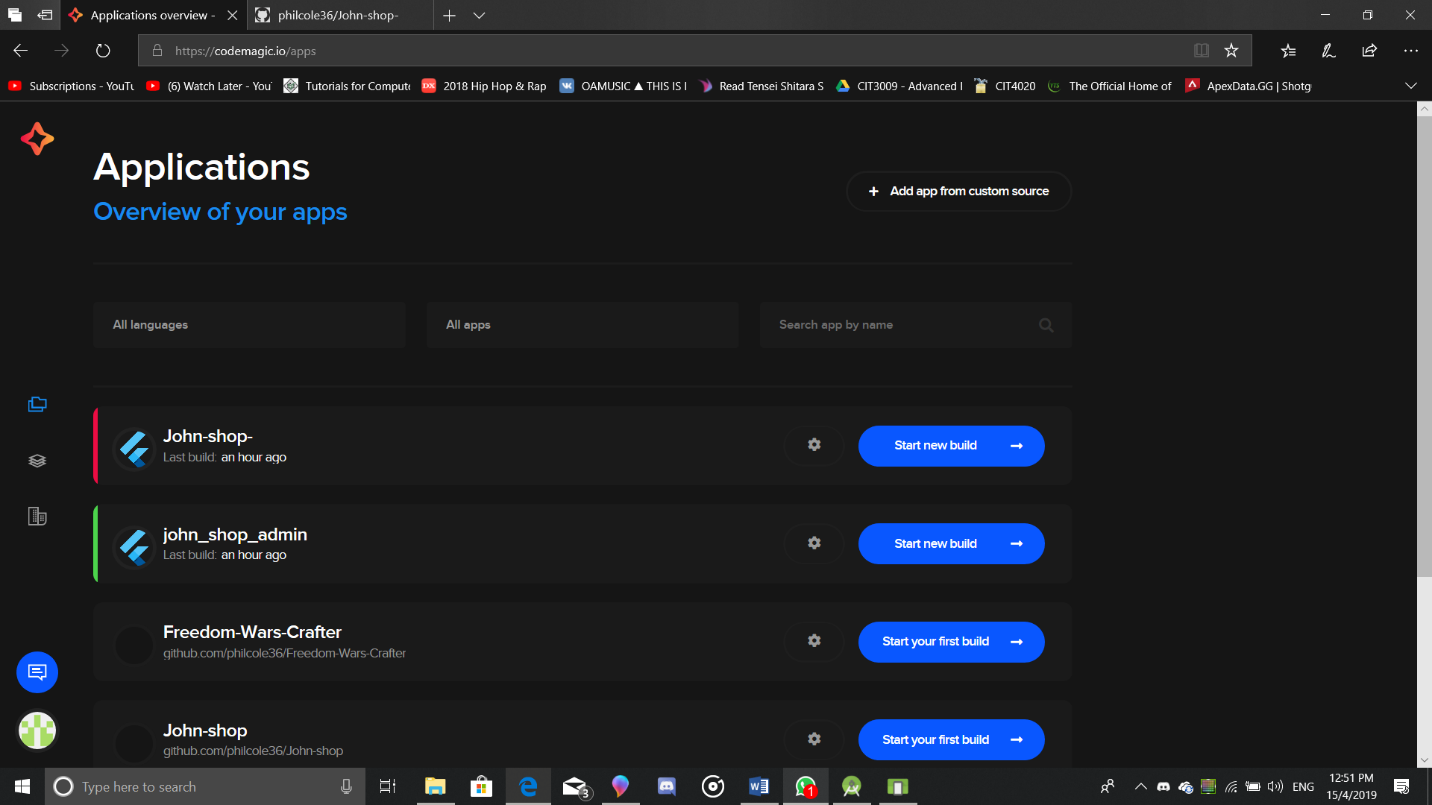
### Model-View-Controller & Repository Pattern

Model also acts as the repository pattern due to its uses as the only link to the database for saving and retrieve information.

A controller is placed at the end of each dart file that requires it and when called upon it await the response from the model file.

### Continuous Integration Server

For continuous integration the service of Code Magic was enlisted, it is the first CI/CD tool dedicated to Flutter apps, and it is developed by Nevercode. It links with the github repository and processes with testing, building and deploying the project selected.



# References

*Dart Packages*. (n.d.). Retrieved from Flutter Packages: https://pub.dartlang.org/flutter

*Flutter Documentation*. (n.d.). Retrieved from Flutter: https://flutter.dev/docs