# ***Refactoring Report***

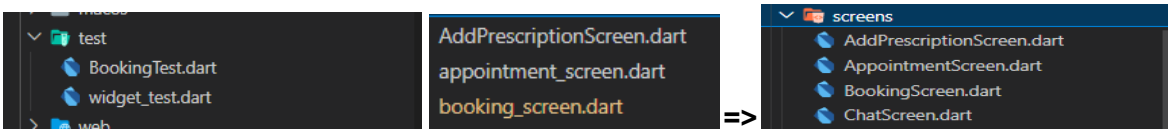
Refactoring is the process of restructuring code, while not changing its original functionality. Throughout our development process, there were many refactorings we did and below is a report of what we changed and why it was a bad smell.

**Design Pattern Refactoring:**

We had mismatches with our file/class naming conventions(CamelCase, Capitalise, lowercase with underscores etc.) With pushes into our develop branches, we had merge conflicts with names of classes.

### *Refactoring method:*

We made sure to come to one file naming convention and stick to it.




**Commented Code:**

During development, there were many segments of code that were commented out when we worked on our individual user stories, and some parts of the said code hadn't been implemented yet and it would cause errors. There were other times where another member would either fix the code or comment that section out or create the same method with the right functionality needed but still not delete the commented code in case the member who wrote it needed it for some specific reason.

### *Refactoring method:*

Every so often, during our code reviews, we would decide which methods or do some round off cleanups of commented code once we have the functionality implemented.

**Dead Functions:**

Some of the commented code above, have been dead functions as explained before.This, if left in, will create confusion when running and just not look nice when we look at our code to review it.

***Refactoring method:***

We look over our commented code, and if we see functions that we don't have the use for, we make sure to delete the function to avoid clutter and confusion of code.

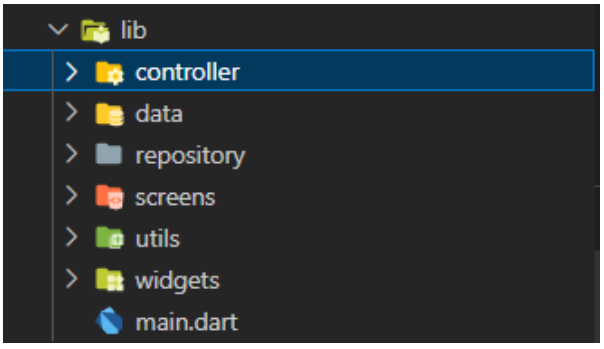
```dart
// // Deletes doctor with theit id
// Future<http.Response> deleteDoctor(int? id) async {
//   var url = Uri.parse('$_dataURL/doctor/$id');
//   var response = await http.delete(url);
//   return response;
// }

// Creates new account and returns account id
Future<int> createAccount(String email, String password) async {
  var url = Uri.parse('$_dataURL/account');

  // json body
  var body = jsonEncode({
    "email": email,
    "passwordHash": password,
    "userType": _usrType
  });

  var response = await http.post
    (
      url,
      headers: {"Content-Type": "application/json"},
      body: body
    );
```

=>

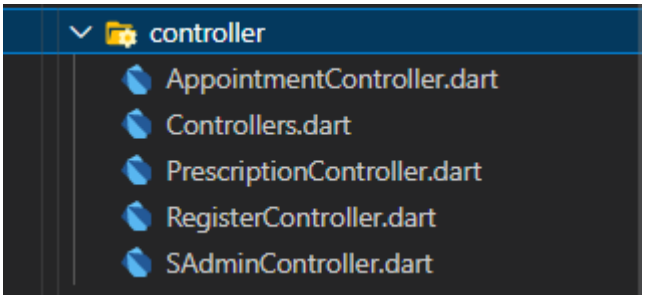
```dart
// Creates new account and returns account id
Future<int> createAccount(String email, String password) async
  var url = Uri.parse('$_dataURL/account');

  // json body
  var body = jsonEncode({
    "email": email,
    "passwordHash": password,
    "userType": _usrType
  });

  var response = await http.post
    (
      url,
      headers: {"Content-Type": "application/json"},
      body: body
    );

  int accountId = json.decode(response.body); // convert

  return accountId;
}
```

Our final file naming conventions:

**Folders** :




**Files**:



**Variables**:

```
class Chat {
  String message;
  String doctor;
  String dateTime;
  String image;
  bool isOnline;
  bool hasStory;
  String messageType;
```

**Functions**:

```
Future<List<Doctor>> getDoctorsList() async {
  List<Doctor> doctorList = [];
```