Authentication.dart

Import is utilized to import the default tools used by flutter SDK. Import is also utilized to import information from other files, this is used constantly as every new file requires the flutter functionality.

```
import 'dart:convert';
import 'dart:async';
import 'package:flutter/widgets.dart';
import 'package:http/http.dart' as http;
import '../models/http_exception.dart';
class Auth with ChangeNotifier {
  String _token;
  DateTime _expiryDate;
  String _userId;
  Timer _authTimer;

  bool get isAuth {
    return token != null;
  }
  String get token {
    if (_expiryDate != null &&
        _expiryDate.isAfter(DateTime.now()) &&
        _token != null) {
      return _token;
    }
    return null;
  }
  String get userId {
    return _userId;
  }
  Future<void> _authenticate(
      String email, String password, String urlSegment) async {
    final url =

'https://www.googleapis.com/identitytoolkit/v3/relyingparty/$urlSegment?key=AIzaSyA7_i6qDofa
aguFo5AmyIO0CcqeB7HymP0';
    try {
```

```
    final response = await http.post(
      url,
      body: json.encode(
        {
          'email': email,
          'password': password,
          'returnSecureToken': true,
        },
      ),
    );
    final responseData = json.decode(response.body);
    if (responseData['error'] != null) {
      throw HttpException(responseData['error']['message']);
    }
    _token = responseData['idToken'];
    _userId = responseData['localId'];
    _expiryDate = DateTime.now().add(
      Duration(
        seconds: int.parse(
          responseData['expiresIn'],
        ),
      ),
    );
    _autoLogout();
    notifyListeners();
  } catch (error) {
    throw error;
  }
}
Future<void> signup(String email, String password) async {
  return _authenticate(email, password, 'signupNewUser');
}
Future<void> login(String email, String password) async {
  return _authenticate(email, password, 'verifyPassword');
}
void logout() {
```

```
    _token = null;
    _userId = null;
    _expiryDate = null;
    if (_authTimer != null) {
      _authTimer.cancel();
      _authTimer = null;
    }
    notifyListeners(); - To notify the database
  }
  void _autoLogout() { - function to autologout user after the token expires.
    if (_authTimer != null) {
      _authTimer.cancel();
    }
    final timeToExpiry = _expiryDate.difference(DateTime.now()).inSeconds;
    _authTimer = Timer(Duration(seconds: timeToExpiry), logout);
  }
}
```

This file is used to create a token to be able to login. The token requires information such as time and userID which is specified by the device utilized. This is included due to requirement from the firebase database. Without the token creation the database will reject any method of signing in.

Cart.dart

```
import 'package:flutter/foundation.dart';
class CartItem {
  final String id;
  final String title;
  final int quantity;
  final double price;
  CartItem({
    @required this.id,
    @required this.title,
    @required this.quantity,
    @required this.price,
  });
}
```

```
class Cart with ChangeNotifier {
  Map<String, CartItem> _items = {};

  Map<String, CartItem> get items {
    return {..._items};
  }
  int get itemCount {
    return _items.length;
  }

  double get totalAmount {
    var total = 0.0;
    _items.forEach((key, cartItem) {
      total += cartItem.price * cartItem.quantity;
    });
    return total;
  }
  void addItem(
    String productId,
    double price,
    String title,
  ) {
    if (_items.containsKey(productId)) {
      _items.update(
        productId,
        (existingCartItem) => CartItem(
              id: existingCartItem.id,
              title: existingCartItem.title,
              price: existingCartItem.price,
              quantity: existingCartItem.quantity + 1,
            ),
      );
    } else {
      _items.putIfAbsent(
        productId,
        () => CartItem(
```

```
            id: DateTime.now().toString(),
            title: title,
            price: price,
            quantity: 1,
          ),
    );
  }
  notifyListeners();
}

void removeItem(String productId) {
  _items.remove(productId);
  notifyListeners();
}

void removeSingleItem(String productId) {
  if (!_items.containsKey(productId)) {
    return;
  }
  if (_items[productId].quantity > 1) {
    _items.update(
        productId,
        (existingCartItem) => CartItem(
              id: existingCartItem.id,
              title: existingCartItem.title,
              price: existingCartItem.price,
              quantity: existingCartItem.quantity - 1,
            ));
  } else {
    _items.remove(productId);
  }
  notifyListeners();
}
void clear() {
  _items = {};
  notifyListeners();
```

```
  }
}
```

This file controls the cart functionality. It reads the items that are added to cart, then based on the items the total is calculated. Within this file it also includes functionality to remove items from within the cart. ProductID is the primary key that controls each of the items. @required is used to ensure that all the product data is loaded before the screen shows up. If data is missing, then the product will not load within the cart. The required information has to be converted into a string in order to be functional.

Orders.dart

```
import 'dart:convert';
import 'package:flutter/foundation.dart';
import 'package:http/http.dart' as http; - to specify http instead of https
import './cart.dart';

class OrderItem {
  final String id;
  final double amount;
  final List<CartItem> products;
  final DateTime dateTime;

  OrderItem({
    @required this.id,
    @required this.amount,
    @required this.products,
    @required this.dateTime,
  });
}

class Orders with ChangeNotifier {
  List<OrderItem> _orders = [];
  final String authToken;
  final String userId;

  Orders(this.authToken, this.userId, this._orders);
```

```
List<OrderItem> get orders {
  return [..._orders];
}

Future<void> fetchAndSetOrders() async {
  final url = 'https://shop-app-ds.firebaseio.com/orders/$userId.json?auth=$authToken';
  final response = await http.get(url);
  final List<OrderItem> loadedOrders = [];
  final extractedData = json.decode(response.body) as Map<String, dynamic>;
  if (extractedData == null) {
    return;
  }
  extractedData.forEach((orderId, orderData) {
    loadedOrders.add(
      OrderItem(
        id: orderId,
        amount: orderData['amount'],
        dateTime: DateTime.parse(orderData['dateTime']),
        products: (orderData['products'] as List<dynamic>)
            .map(
              (item) => CartItem(
                    id: item['id'],
                    price: item['price'],
                    quantity: item['quantity'],
                    title: item['title'],
                  ),
            )
            .toList(),
      ),
    );
  });
  _orders = loadedOrders.reversed.toList();
  notifyListeners();
}

Future<void> addOrder(List<CartItem> cartProducts, double total) async {
```

```
    final url = 'https://shop-app-ds.firebaseio.com/orders/$userId.json?auth=$authToken';
    final timestamp = DateTime.now();
    final response = await http.post(
      url,
      body: json.encode({
        'amount': total,
        'dateTime': timestamp.toIso8601String(),
        'products': cartProducts
            .map((cp) => {
                  'id': cp.id,
                  'title': cp.title,
                  'quantity': cp.quantity,
                  'price': cp.price,
                })
            .toList(),
      }),
    );
    _orders.insert(
      0,
      OrderItem(
        id: json.decode(response.body)['name'],
        amount: total,
        dateTime: timestamp,
        products: cartProducts,
      ),
    );
    notifyListeners();
  }
}
```

This is the functionality in order to place orders. It notifies the database, so the orders also show up on the database. It sends the id of transaction, the title, the quantity and the total price of the cart.

Product.dart

```
import 'dart:convert';
```

```dart
import 'package:flutter/foundation.dart';
import 'package:http/http.dart' as http;

class Product with ChangeNotifier {
  final String id;
  final String title;
  final String description;
  final double price;
  final String imageUrl;
  bool isFavorite;

  Product({
    @required this.id,
    @required this.title,
    @required this.description,
    @required this.price,
    @required this.imageUrl,
    this.isFavorite = false,
  });

  void _setFavValue(bool newValue) {
    isFavorite = newValue;
    notifyListeners();
  }

  Future<void> toggleFavoriteStatus(String token, String userId) async {
    final oldStatus = isFavorite;
    isFavorite = !isFavorite;
    notifyListeners();
    final url =
        'https://shop-app-ds.firebaseio.com/userFavorites/$userId/$id.json?auth=$token';
    try {
      final response = await http.put(
        url,
        body: json.encode(
          isFavorite,
```

```
        ),
      );
      if (response.statusCode >= 400) {
        _setFavValue(oldStatus);
      }
    } catch (error) {
      _setFavValue(oldStatus);
    }
  }
}
```

This functionality adds the information required to showcase information within the individual product screen. Including loading the price, title, picture and description. It gathers the information from the URL.

Products.dart

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import '../models/http_exception.dart';
import './product.dart';

class Products with ChangeNotifier {
  List<Product> _items = [
  ];
  final String authToken;
  final String userId;

  Products(this.authToken, this.userId, this._items);

  List<Product> get items {
    return [..._items];
  }

  List<Product> get favoriteItems {
    return _items.where((prodItem) => prodItem.isFavorite).toList();
  }
```

```
Product findById(String id) {
  return _items.firstWhere((prod) => prod.id == id);
}
Future<void> fetchAndSetProducts([bool filterByUser = false]) async {
  final filterString = filterByUser ? 'orderBy="creatorId"&equalTo="$userId"' : '';
  var url =
      'https://shop-app-ds.firebaseio.com/products.json?auth=$authToken&$filterString';
  try {
    final response = await http.get(url);
    final extractedData = json.decode(response.body) as Map<String, dynamic>;
    if (extractedData == null) {
      return;
    }
    url =
        'https://shop-app-ds.firebaseio.com/userFavorites/$userId.json?auth=$authToken';
    final favoriteResponse = await http.get(url);
    final favoriteData = json.decode(favoriteResponse.body);
    final List<Product> loadedProducts = [];
    extractedData.forEach((prodId, prodData) {
      loadedProducts.add(Product(
        id: prodId,
        title: prodData['title'],
        description: prodData['description'],
        price: prodData['price'],
        isFavorite:
            favoriteData == null ? false : favoriteData[prodId] ?? false,
        imageUrl: prodData['imageUrl'],
      ));
    });
    _items = loadedProducts;
    notifyListeners();
  } catch (error) {
    throw (error);
  }
}
```

```
Future<void> addProduct(Product product) async {
  final url =
      'https://shop-app-ds.firebaseio.com/products.json?auth=$authToken';
  try {
    final response = await http.post(
      url,
      body: json.encode({
        'title': product.title,
        'description': product.description,
        'imageUrl': product.imageUrl,
        'price': product.price,
        'creatorId': userId,
      }),
    );
    final newProduct = Product(
      title: product.title,
      description: product.description,
      price: product.price,
      imageUrl: product.imageUrl,
      id: json.decode(response.body)['name'],
    );
    _items.add(newProduct);
    notifyListeners();
  } catch (error) {
    print(error);
    throw error;
  }
}
Future<void> updateProduct(String id, Product newProduct) async {
  final prodIndex = _items.indexWhere((prod) => prod.id == id);
  if (prodIndex >= 0) {
    final url =
        'https://shop-app-ds.firebaseio.com/products/$id.json?auth=$authToken';
    await http.patch(url,
        body: json.encode({
          'title': newProduct.title,
```

```
            'description': newProduct.description,
            'imageUrl': newProduct.imageUrl,
            'price': newProduct.price
          }));
      _items[prodIndex] = newProduct;
      notifyListeners();
    } else {
      print('...');
    }
  }
  Future<void> deleteProduct(String id) async {
    final url =
        'https://shop-app-ds.firebaseio.com/products/$id.json?auth=$authToken';
    final existingProductIndex = _items.indexWhere((prod) => prod.id == id);
    var existingProduct = _items[existingProductIndex];
    _items.removeAt(existingProductIndex);
    notifyListeners();
    final response = await http.delete(url);
    if (response.statusCode >= 400) {
      _items.insert(existingProductIndex, existingProduct);
      notifyListeners();
      throw HttpException('Could not delete product.');
    }
    existingProduct = null;
  }
}
```

This is the functionality to add a new item into the database, it takes the input from the user. The information is added to the correct fields of the database. Furthermore, this also includes the functionality to delete a product and to edit a product. It is also required that the token is accurate and that it is the admin token, to ensure that only admins can edit or add a new product. It also checks to ensure that the product hasn't been added yet, to ensure no duplication occurs

Authentication_Screen.dart

```
import 'dart:math';
```

```dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

import '../providers/auth.dart';
import '../models/http_exception.dart';

enum AuthMode { Signup, Login }

class AuthScreen extends StatelessWidget {
  static const routeName = '/auth';

  @override
  Widget build(BuildContext context) {
    final deviceSize = MediaQuery.of(context).size;
    return Scaffold(
      body: Stack(
        children: <Widget>[
          SingleChildScrollView(
            child: Container(
              height: deviceSize.height,
              width: deviceSize.width,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                crossAxisAlignment: CrossAxisAlignment.center,
                children: <Widget>[

                  Flexible(
                    flex: deviceSize.width > 600 ? 2 : 1,
                    child: AuthCard(),
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
```

```
    );
  }
}

class AuthCard extends StatefulWidget {
  const AuthCard({
    Key key,
  }) : super(key: key);

  @override
  _AuthCardState createState() => _AuthCardState();
}

class _AuthCardState extends State<AuthCard> {
  final GlobalKey<FormState> _formKey = GlobalKey();
  AuthMode _authMode = AuthMode.Login;
  Map<String, String> _authData = {
    'email': '',
    'password': '',
  };
  var _isLoading = false;
  final _passwordController = TextEditingController();

  void _showErrorDialog(String message) {
    showDialog(
      context: context,
      builder: (ctx) => AlertDialog(
          title: Text('An Error Occurred!'),
          content: Text(message),
          actions: <Widget>[
            FlatButton(
              child: Text('Okay'),
              onPressed: () {
                Navigator.of(ctx).pop();
              },
            )
```

```
          ],
        ),
  );
}

Future<void> _submit() async {
  if (!_formKey.currentState.validate()) {
    return;
  }
  _formKey.currentState.save();
  setState(() {
    _isLoading = true;
  });
  try {
    if (_authMode == AuthMode.Login) {
      await Provider.of<Auth>(context, listen: false).login(
        _authData['email'],
        _authData['password'],
      );
    } else {
      await Provider.of<Auth>(context, listen: false).signup(
        _authData['email'],
        _authData['password'],
      );
    }
  } on HttpException catch (error) {
    var errorMessage = 'Authentication failed';
    if (error.toString().contains('EMAIL_EXISTS')) {
      errorMessage = 'This email address is already in use.';
    } else if (error.toString().contains('INVALID_EMAIL')) {
      errorMessage = 'This is not a valid email address';
    } else if (error.toString().contains('WEAK_PASSWORD')) {
      errorMessage = 'This password is too weak.';
    } else if (error.toString().contains('EMAIL_NOT_FOUND')) {
      errorMessage = 'Could not find a user with that email.';
    } else if (error.toString().contains('INVALID_PASSWORD')) {
```

```
      errorMessage = 'Invalid password.';
    }
    _showErrorDialog(errorMessage);
  } catch (error) {
    const errorMessage =
        'Could not authenticate you. Please try again later.';
    _showErrorDialog(errorMessage);
  }

  setState(() {
    _isLoading = false;
  });
}

void _switchAuthMode() {
  if (_authMode == AuthMode.Login) {
    setState(() {
      _authMode = AuthMode.Signup;
    });
  } else {
    setState(() {
      _authMode = AuthMode.Login;
    });
  }
}

@override
Widget build(BuildContext context) {
  final deviceSize = MediaQuery.of(context).size;
  return Card(
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(10.0),
    ),
    elevation: 8.0,
    child: Container(
      height: _authMode == AuthMode.Signup ? 320 : 260,
```

```
constraints:
    BoxConstraints(minHeight: _authMode == AuthMode.Signup ? 320 : 260),
width: deviceSize.width * 0.75,
padding: EdgeInsets.all(16.0),
child: Form(
  key: _formKey,
  child: SingleChildScrollView(
    child: Column(
      children: <Widget>[
        TextFormField(
          decoration: InputDecoration(labelText: 'E-Mail'),
          keyboardType: TextInputType.emailAddress,
          validator: (value) {
            if (value.isEmpty || !value.contains('@')) {
              return 'Invalid email!';
            }
          },
          onSaved: (value) {
            _authData['email'] = value;
          },
        ),
        TextFormField(
          decoration: InputDecoration(labelText: 'Password'),
          obscureText: true,
          controller: _passwordController,
          validator: (value) {
            if (value.isEmpty || value.length < 5) {
              return 'Password is too short!';
            }
          },
          onSaved: (value) {
            _authData['password'] = value;
          },
        ),
        if (_authMode == AuthMode.Signup)
          TextFormField(
```

```
    enabled: _authMode == AuthMode.Signup,
    decoration: InputDecoration(labelText: 'Confirm Password'),
    obscureText: true,
    validator: _authMode == AuthMode.Signup
        ? (value) {
            if (value != _passwordController.text) {
              return 'Passwords do not match!';
            }
          }
        : null,
  ),
SizedBox(
  height: 20,
),
if (_isLoading)
  CircularProgressIndicator()
else
  RaisedButton(
    child:
        Text(_authMode == AuthMode.Login ? 'LOGIN' : 'SIGN UP'),
    onPressed: _submit,
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(30),
    ),
    padding:
        EdgeInsets.symmetric(horizontal: 30.0, vertical: 8.0),
    color: Theme.of(context).primaryColor,
    textColor: Theme.of(context).primaryTextTheme.button.color,
  ),
FlatButton(
  child: Text(
      '${_authMode == AuthMode.Login ? 'SIGNUP' : 'LOGIN'} INSTEAD'),
  onPressed: _switchAuthMode,
  padding: EdgeInsets.symmetric(horizontal: 30.0, vertical: 4),
  materialTapTargetSize: MaterialTapTargetSize.shrinkWrap,
  textColor: Theme.of(context).primaryColor,
```

```
              ),
            ],
          ),
        ),
      ),
    ),
  );
 }
}
```

This file includes the requirements to design the authentication page, including all the buttons, their functionality and the text boxes. It also includes the correct error messages to show the user if they don't input something correctly, or if the user is not signed up. This file also includes the authentication technique to ensure that the user is registered, and that duplication doesn't occur.

Designs and the location of each asset and text box are also included. But also, what functionality should occur when a user presses them. For example, opening a textbox will trigger the keyboard.

Cart Screen.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../providers/cart.dart' show Cart;
import '../widgets/cart_item.dart';
import '../providers/orders.dart';

class CartScreen extends StatelessWidget {
  static const routeName = '/cart';

  @override
  Widget build(BuildContext context) {
    final cart = Provider.of<Cart>(context);
    return Scaffold(
      appBar: AppBar(
        title: Text('Your Cart'),
      ),
```

```
body: Column(
  children: <Widget>[
    Card(
      margin: EdgeInsets.all(15),
      child: Padding(
        padding: EdgeInsets.all(8),
        child: Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: <Widget>[
            Text(
              'Total',
              style: TextStyle(fontSize: 20),
            ),
            Spacer(),
            Chip(
              label: Text(
                '\$${cart.totalAmount.toStringAsFixed(2)}',
                style: TextStyle(
                  color: Theme.of(context).primaryTextTheme.title.color,
                ),
              ),
              backgroundColor: Theme.of(context).primaryColor,
            ),
            OrderButton(cart: cart)
          ],
        ),
      ),
    ),
    SizedBox(height: 10),
    Expanded(
      child: ListView.builder(
        itemCount: cart.items.length,
        itemBuilder: (ctx, i) => CartItem(
            cart.items.values.toList()[i].id,
            cart.items.keys.toList()[i],
            cart.items.values.toList()[i].price,
```

```
                cart.items.values.toList()[i].quantity,
                cart.items.values.toList()[i].title,
              ),
            ),
          )
        ],
      ),
    );
  }
}

class OrderButton extends StatefulWidget {
  const OrderButton({
    Key key,
    @required this.cart,
  }) : super(key: key);

  final Cart cart;

  @override
  _OrderButtonState createState() => _OrderButtonState();
}

class _OrderButtonState extends State<OrderButton> {
  var _isLoading = false;

  @override
  Widget build(BuildContext context) {
    return FlatButton(
      child: _isLoading ? CircularProgressIndicator() : Text('ORDER NOW'),
      onPressed: (widget.cart.totalAmount <= 0 || _isLoading)
          ? null
          : () async {
              setState(() {
                _isLoading = true;
              });
```

```
            await Provider.of<Orders>(context, listen: false).addOrder(
              widget.cart.items.values.toList(),
              widget.cart.totalAmount,
            );
            setState(() {
              _isLoading = false;
            });
            widget.cart.clear();
          },
      textColor: Theme.of(context).primaryColor,
    );
  }
}
```

This file includes how the cart screen should look like, how the different products in the cart should be showcased alongside with their information. It includes information on where each asset should be located on the screen. Furthermore, it includes the buttons that add the functionality from the cart.dart file, such as placing a new order, calculating total and removing items from the cart.

Edit Product Screen.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../providers/product.dart';
import '../providers/products.dart';

class EditProductScreen extends StatefulWidget {
  static const routeName = '/edit-product';

  @override
  _EditProductScreenState createState() => _EditProductScreenState();
}

class _EditProductScreenState extends State<EditProductScreen> {
  final _priceFocusNode = FocusNode();
```

```
final _descriptionFocusNode = FocusNode();
final _imageUrlController = TextEditingController();
final _imageUrlFocusNode = FocusNode();
final _form = GlobalKey<FormState>();
var _editedProduct = Product(
  id: null,
  title: '',
  price: 0,
  description: '',
  imageUrl: '',
);
var _initValues = {
  'title': '',
  'description': '',
  'price': '',
  'imageUrl': '',
};
var _isInit = true;
var _isLoading = false;

@override
void initState() {
  _imageUrlFocusNode.addListener(_updateImageUrl);
  super.initState();
}

@override
void didChangeDependencies() {
  if (_isInit) {
    final productId = ModalRoute.of(context).settings.arguments as String;
    if (productId != null) {
      _editedProduct =
          Provider.of<Products>(context, listen: false).findById(productId);
      _initValues = {
        'title': _editedProduct.title,
        'description': _editedProduct.description,
```

```
        'price': _editedProduct.price.toString(),
        'imageUrl': '',
      };
      _imageUrlController.text = _editedProduct.imageUrl;
    }
  }
  _isInit = false;
  super.didChangeDependencies();
}

@override
void dispose() {
  _imageUrlFocusNode.removeListener(_updateImageUrl);
  _priceFocusNode.dispose();
  _descriptionFocusNode.dispose();
  _imageUrlController.dispose();
  _imageUrlFocusNode.dispose();
  super.dispose();
}

void _updateImageUrl() {
  if (!_imageUrlFocusNode.hasFocus) {
    if ((!_imageUrlController.text.startsWith('http') &&
            !_imageUrlController.text.startsWith('https')) ||
        (!_imageUrlController.text.endsWith('.png') &&
            !_imageUrlController.text.endsWith('.jpg') &&
            !_imageUrlController.text.endsWith('.jpeg'))) {
      return;
    }
    setState(() {});
  }
}

Future<void> _saveForm() async {
  final isValid = _form.currentState.validate();
  if (!isValid) {
```

```
  return;
}
_form.currentState.save();
setState(() {
  _isLoading = true;
});
if (_editedProduct.id != null) {
  await Provider.of<Products>(context, listen: false)
      .updateProduct(_editedProduct.id, _editedProduct);
} else {
  try {
    await Provider.of<Products>(context, listen: false)
        .addProduct(_editedProduct);
  } catch (error) {
    await showDialog(
      context: context,
      builder: (ctx) => AlertDialog(
            title: Text('An error occurred!'),
            content: Text('Something went wrong.'),
            actions: <Widget>[
              FlatButton(
                child: Text('Okay'),
                onPressed: () {
                  Navigator.of(ctx).pop();
                },
              )
            ],
          ),
    );
  }

}
setState(() {
  _isLoading = false;
});
Navigator.of(context).pop();
```

```
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Edit Product'),
      actions: <Widget>[
        IconButton(
          icon: Icon(Icons.save),
          onPressed: _saveForm,
        ),
      ],
    ),
    body: _isLoading
        ? Center(
            child: CircularProgressIndicator(),
          )
        : Padding(
            padding: const EdgeInsets.all(16.0),
            child: Form(
              key: _form,
              child: ListView(
                children: <Widget>[
                  TextFormField(
                    initialValue: _initValues['title'],
                    decoration: InputDecoration(labelText: 'Title'),
                    textInputAction: TextInputAction.next,
                    onFieldSubmitted: (_) {
                      FocusScope.of(context).requestFocus(_priceFocusNode);
                    },
                    validator: (value) {
                      if (value.isEmpty) {
                        return 'Please provide a value.';
                      }
                      return null;
```

```
  },
  onSaved: (value) {
    _editedProduct = Product(
        title: value,
        price: _editedProduct.price,
        description: _editedProduct.description,
        imageUrl: _editedProduct.imageUrl,
        id: _editedProduct.id,
        isFavorite: _editedProduct.isFavorite);
  },
),
TextFormField(
  initialValue: _initValues['price'],
  decoration: InputDecoration(labelText: 'Price'),
  textInputAction: TextInputAction.next,
  keyboardType: TextInputType.number,
  focusNode: _priceFocusNode,
  onFieldSubmitted: (_) {
    FocusScope.of(context)
        .requestFocus(_descriptionFocusNode);
  },
  validator: (value) {
    if (value.isEmpty) {
      return 'Please enter a price.';
    }
    if (double.tryParse(value) == null) {
      return 'Please enter a valid number.';
    }
    if (double.parse(value) <= 0) {
      return 'Please enter a number greater than zero.';
    }
    return null;
  },
  onSaved: (value) {
    _editedProduct = Product(
        title: _editedProduct.title,
```

```
        price: double.parse(value),
        description: _editedProduct.description,
        imageUrl: _editedProduct.imageUrl,
        id: _editedProduct.id,
        isFavorite: _editedProduct.isFavorite);
  },
),
TextFormField(
  initialValue: _initValues['description'],
  decoration: InputDecoration(labelText: 'Description'),
  maxLines: 3,
  keyboardType: TextInputType.multiline,
  focusNode: _descriptionFocusNode,
  validator: (value) {
    if (value.isEmpty) {
      return 'Please enter a description.';
    }
    if (value.length < 10) {
      return 'Should be at least 10 characters long.';
    }
    return null;
  },
  onSaved: (value) {
    _editedProduct = Product(
      title: _editedProduct.title,
      price: _editedProduct.price,
      description: value,
      imageUrl: _editedProduct.imageUrl,
      id: _editedProduct.id,
      isFavorite: _editedProduct.isFavorite,
    );
  },
),
Row(
  crossAxisAlignment: CrossAxisAlignment.end,
  children: <Widget>[
```

```
Container(
  width: 100,
  height: 100,
  margin: EdgeInsets.only(
    top: 8,
    right: 10,
  ),
  decoration: BoxDecoration(
    border: Border.all(
      width: 1,
      color: Colors.grey,
    ),
  ),
  child: _imageUrlController.text.isEmpty
      ? Text('Enter a URL')
      : FittedBox(
          child: Image.network(
            _imageUrlController.text,
            fit: BoxFit.cover,
          ),
        ),
),
Expanded(
  child: TextFormField(
    decoration: InputDecoration(labelText: 'Image URL'),
    keyboardType: TextInputType.url,
    textInputAction: TextInputAction.done,
    controller: _imageUrlController,
    focusNode: _imageUrlFocusNode,
    onFieldSubmitted: (_) {
      _saveForm();
    },
    validator: (value) {
      if (value.isEmpty) {
        return 'Please enter an image URL.';
      }
```

```
                  if (!value.startsWith('http') &&
                      !value.startsWith('https')) {
                    return 'Please enter a valid URL.';
                  }
                  if (!value.endsWith('.png') &&
                      !value.endsWith('.jpg') &&
                      !value.endsWith('.jpeg')) {
                    return 'Please enter a valid image URL.';
                  }
                  return null;
                },
                onSaved: (value) {
                  _editedProduct = Product(
                    title: _editedProduct.title,
                    price: _editedProduct.price,
                    description: _editedProduct.description,
                    imageUrl: value,
                    id: _editedProduct.id,
                    isFavorite: _editedProduct.isFavorite,
                  );
                },
              ),
            ),
          ],
        ),
      ],
    ),
  ),
);
 }
}
```

This file controls the edit/add product screen. Alongside with the data validation to ensure that the data entered is accurate and correct. Ensuring that everything works well within the database. This file also includes the information to update a product on the

database. This file also includes the formatting and the various assets included within this screen.

Orders Screen.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

import '../providers/orders.dart' show Orders;
import '../widgets/order_item.dart';
import '../widgets/app_drawer.dart';

class OrdersScreen extends StatelessWidget {
  static const routeName = '/orders';

  @override
  Widget build(BuildContext context) {
    print('building orders');
    return Scaffold(
      appBar: AppBar(
        title: Text('Your Orders'),
      ),
      drawer: AppDrawer(),
      body: FutureBuilder(
        future: Provider.of<Orders>(context, listen: false).fetchAndSetOrders(),
        builder: (ctx, dataSnapshot) {
          if (dataSnapshot.connectionState == ConnectionState.waiting) {
            return Center(child: CircularProgressIndicator());
          } else {
            if (dataSnapshot.error != null) {
              return Center(
                child: Text('An error occurred!'),
              );
            } else {
              return Consumer<Orders>(
                builder: (ctx, orderData, child) => ListView.builder(
                      itemCount: orderData.orders.length,
```

```
                  itemBuilder: (ctx, i) => OrderItem(orderData.orders[i]),
                ),
              );
            }
          }
        },
      ),
    );
  }
}
```

Product Detail Screen.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

import '../providers/products.dart';

class ProductDetailScreen extends StatelessWidget {
  static const routeName = '/product-detail';

  @override
  Widget build(BuildContext context) {
    final productId =
        ModalRoute.of(context).settings.arguments as String;
    final loadedProduct = Provider.of<Products>(
      context,
      listen: false,
    ).findById(productId);
    return Scaffold(
      appBar: AppBar(
        title: Text(loadedProduct.title),
      ),
      body: SingleChildScrollView(
        child: Column(
          children: <Widget>[
            Container(
              height: 300,
```

```
            width: double.infinity,
            child: Image.network(
              loadedProduct.imageUrl,
              fit: BoxFit.cover,
            ),
          ),
          SizedBox(height: 10),
          Text(
            '\$${loadedProduct.price}',
            style: TextStyle(
              color: Colors.grey,
              fontSize: 20,
            ),
          ),
          SizedBox(
            height: 10,
          ),
          Container(
            padding: EdgeInsets.symmetric(horizontal: 10),
            width: double.infinity,
            child: Text(
              loadedProduct.description,
              textAlign: TextAlign.center,
              softWrap: true,
            ),
          )
        ],
      ),
    ),
  );
 }
}
```

This file includes the way the various orders for each user are gathered from the database and showcased within the app. This includes the different methods to showcase price, quantity, time of the order and the individual products. This information

is user specific, so it has to be gathered from the user that is logged in using the token that was generated. Furthermore, it also includes the styling details for this screen.

Product Overview Screen.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../widgets/app_drawer.dart';
import '../widgets/products_grid.dart';
import '../providers/cart.dart';
import './cart_screen.dart';
import '../providers/products.dart';

enum FilterOptions {
  Favorites,
  All,
}

class ProductsOverviewScreen extends StatefulWidget {
  @override
  _ProductsOverviewScreenState createState() => _ProductsOverviewScreenState();
}

class _ProductsOverviewScreenState extends State<ProductsOverviewScreen> {
  var _showOnlyFavorites = false;
  var _isInit = true;
  var _isLoading = false;

  @override
  void initState() {
    super.initState();
  }

  @override
  void didChangeDependencies() {
    if (_isInit) {
      setState(() {
```

```
      _isLoading = true;
    });
    Provider.of<Products>(context).fetchAndSetProducts().then((_) {
      setState(() {
        _isLoading = false;
      });
    });
  }
  _isInit = false;
  super.didChangeDependencies();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Tyres Online AE'),
      actions: <Widget>[
        PopupMenuButton(
          onSelected: (FilterOptions selectedValue) {
            setState(() {
              if (selectedValue == FilterOptions.Favorites) {
                _showOnlyFavorites = true;
              } else {
                _showOnlyFavorites = false;
              }
            });
          },
          icon: Icon(
            Icons.more_vert,
          ),
          itemBuilder: (_) => [
                PopupMenuItem(
                  child: Text('Only Favorites'),
                  value: FilterOptions.Favorites,
                ),
```

```
              PopupMenuItem(
                child: Text('Show All'),
                value: FilterOptions.All,
              ),
            ],
        ),
        Consumer<Cart>(
          builder: (_, cart, ch) => Badge(
                child: ch,
                value: cart.itemCount.toString(),
              ),
          child: IconButton(
            icon: Icon(
              Icons.shopping_cart,
            ),
            onPressed: () {
              Navigator.of(context).pushNamed(CartScreen.routeName);
            },
          ),
        ),
      ],
    ),
    drawer: AppDrawer(),
    body: _isLoading
        ? Center(
            child: CircularProgressIndicator(),
          )
        : ProductsGrid(_showOnlyFavorites),
  );
 }
}
```

This file controls how the various products in the database should be showcased within the main screen. This includes styling of the boxes and being able to gather the required information from the database. This file also includes the filtering option that is use specific, utilizing the user's token. As well as being able to filter the products based on

whether they are a user's favorites or no. Lastly, this also includes on being able to add products to cart and being able to undo them within a 3 second time period.

User Products Screen.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

import '../providers/products.dart';
import '../widgets/user_product_item.dart';
import '../widgets/app_drawer.dart';
import './edit_product_screen.dart';

class UserProductsScreen extends StatelessWidget {
  static const routeName = '/user-products';

  Future<void> _refreshProducts(BuildContext context) async {
    await Provider.of<Products>(context, listen: false)
        .fetchAndSetProducts(true);
  }

  @override
  Widget build(BuildContext context) {
    print('rebuilding...');
    return Scaffold(
      appBar: AppBar(
        title: const Text('Your Products'),
        actions: <Widget>[
          IconButton(
            icon: const Icon(Icons.add),
            onPressed: () {
              Navigator.of(context).pushNamed(EditProductScreen.routeName);
            },
          ),
        ],
      ),
      drawer: AppDrawer(),
```

```
      body: FutureBuilder(
        future: _refreshProducts(context),
        builder: (ctx, snapshot) =>
            snapshot.connectionState == ConnectionState.waiting
                ? Center(
                    child: CircularProgressIndicator(),
                  )
                : RefreshIndicator(
                    onRefresh: () => _refreshProducts(context),
                    child: Consumer<Products>(
                      builder: (ctx, productsData, _) => Padding(
                            padding: EdgeInsets.all(8),
                            child: ListView.builder(
                              itemCount: productsData.items.length,
                              itemBuilder: (_, i) => Column(
                                    children: [
                                      UserProductItem(
                                        productsData.items[i].id,
                                        productsData.items[i].title,
                                        productsData.items[i].imageUrl,
                                      ),
                                      Divider(),
                                    ],
                                  ),
                            ),
                          ),
                    ),
                  ),
      ),
    );
  }
}
```

This is the screen that showcases all the products available on the database, just to show them within an organized environment. Furthermore, it also includes the styling for the page such as dimensions for the picture, font for the product title. As well as icons that are gathered from Google's Material design library which was imported at the start.

App Drawer.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

import '../screens/orders_screen.dart';
import '../screens/user_products_screen.dart';
import '../providers/auth.dart';

class AppDrawer extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Drawer(
      child: Column(
        children: <Widget>[
          AppBar(
            title: Text('Hello Friend!'),
            automaticallyImplyLeading: false,
          ),
          Divider(),
          ListTile(
            leading: Icon(Icons.shop),
            title: Text('Shop'),
            onTap: () {
              Navigator.of(context).pushReplacementNamed('/');
            },
          ),
          Divider(),
          ListTile(
            leading: Icon(Icons.payment),
            title: Text('Orders'),
            onTap: () {
              Navigator.of(context)
                  .pushReplacementNamed(OrdersScreen.routeName);
            },
          ),
```

```
          Divider(),
          ListTile(
            leading: Icon(Icons.edit),
            title: Text('Manage Products'),
            onTap: () {
              Navigator.of(context)
                  .pushReplacementNamed(UserProductsScreen.routeName);
            },
          ),
          Divider(),
          ListTile(
            leading: Icon(Icons.exit_to_app),
            title: Text('Logout'),
            onTap: () {
              Navigator.of(context).pop();
              Provider.of<Auth>(context, listen: false).logout();
            },
          ),
        ],
      ),
    );
  }
}
```

This file controls how the navigation/app drawers of the application works, the design of how it looks, the different links that allow it to access the various pages and the icons associated to each page. This is organized as a column within a grid.

Cart Item.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

import '../providers/cart.dart';

class CartItem extends StatelessWidget {
  final String id;
  final String productId;
```

```
final double price;
final int quantity;
final String title;

CartItem(
  this.id,
  this.productId,
  this.price,
  this.quantity,
  this.title,
);

@override
Widget build(BuildContext context) {
  return Dismissible(
    key: ValueKey(id),
    background: Container(
      color: Theme.of(context).errorColor,
      child: Icon(
        Icons.delete,
        color: Colors.white,
        size: 40,
      ),
      alignment: Alignment.centerRight,
      padding: EdgeInsets.only(right: 20),
      margin: EdgeInsets.symmetric(
        horizontal: 15,
        vertical: 4,
      ),
    ),
    direction: DismissDirection.endToStart,
    confirmDismiss: (direction) {
      return showDialog(
        context: context,
        builder: (ctx) => AlertDialog(
            title: Text('Are you sure?'),
```

```
          content: Text(
            'Do you want to remove the item from the cart?',
          ),
          actions: <Widget>[
            FlatButton(
              child: Text('No'),
              onPressed: () {
                Navigator.of(ctx).pop(false);
              },
            ),
            FlatButton(
              child: Text('Yes'),
              onPressed: () {
                Navigator.of(ctx).pop(true);
              },
            ),
          ],
        ),
    );
  },
  onDismissed: (direction) {
    Provider.of<Cart>(context, listen: false).removeItem(productId);
  },
  child: Card(
    margin: EdgeInsets.symmetric(
      horizontal: 15,
      vertical: 4,
    ),
    child: Padding(
      padding: EdgeInsets.all(8),
      child: ListTile(
        leading: CircleAvatar(
          child: Padding(
            padding: EdgeInsets.all(5),
            child: FittedBox(
              child: Text('\$$price'),
```

```
              ),
            ),
          ),
          title: Text(title),
          subtitle: Text('Total: \$${(price * quantity)}'),
          trailing: Text('$quantity x'),
        ),
      ),
    ),
  );
 }
}
```

This file controls each tile for each product within the cart platform, it shows the price, the quantity, the item and the total price. It is responsible for how each of these tiles work and show on the screen. It also controls being able to remove each item by swiping on their tile.

Order Item.dart

```
import 'dart:math';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import '../providers/orders.dart' as ord;

class OrderItem extends StatefulWidget {
 final ord.OrderItem order;

 OrderItem(this.order);

 @override
 _OrderItemState createState() => _OrderItemState();
}

class _OrderItemState extends State<OrderItem> {
 var _expanded = false;

 @override
 Widget build(BuildContext context) {
```

```
return Card(
  margin: EdgeInsets.all(10),
  child: Column(
    children: <Widget>[
      ListTile(
        title: Text('\$${widget.order.amount}'),
        subtitle: Text(
          DateFormat('dd/MM/yyyy hh:mm').format(widget.order.dateTime),
        ),
        trailing: IconButton(
          icon: Icon(_expanded ? Icons.expand_less : Icons.expand_more),
          onPressed: () {
            setState(() {
              _expanded = !_expanded;
            });
          },
        ),
      ),
      if (_expanded)
        Container(
          padding: EdgeInsets.symmetric(horizontal: 15, vertical: 4),
          height: min(widget.order.products.length * 20.0 + 10, 100),
          child: ListView(
            children: widget.order.products
                .map(
                  (prod) => Row(
                      mainAxisAlignment: MainAxisAlignment.spaceBetween,
                      children: <Widget>[
                        Text(
                          prod.title,
                          style: TextStyle(
                            fontSize: 18,
                            fontWeight: FontWeight.bold,
                          ),
                        ),
                        Text(
```

```
                    '${prod.quantity}x \$${prod.price}',
                    style: TextStyle(
                      fontSize: 18,
                      color: Colors.grey,
                    ),
                  )
                ],
              ),
            )
            .toList(),
        ),
      )
    ],
  ),
 );
 }
}
```

This file is used to showcase the various different orders that have been made by a user, it controls how these orders will look on the screen but also the various logical problems needed to be solved to showcase the accurate information to the user.

Product Item.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../screens/product_detail_screen.dart';
import '../providers/product.dart';
import '../providers/cart.dart';
import '../providers/auth.dart';

class ProductItem extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  final product = Provider.of<Product>(context, listen: false);
  final cart = Provider.of<Cart>(context, listen: false);
  final authData = Provider.of<Auth>(context, listen: false);
  return ClipRRect(
   borderRadius: BorderRadius.circular(10),
```

```
child: GridTile(
  child: GestureDetector(
    onTap: () {
      Navigator.of(context).pushNamed(
        ProductDetailScreen.routeName,
        arguments: product.id,
      );
    },
    child: Image.network(
      product.imageUrl,
      fit: BoxFit.cover,
    ),
  ),
  footer: GridTileBar(
    backgroundColor: Colors.black87,
    leading: Consumer<Product>(
      builder: (ctx, product, _) => IconButton(
            icon: Icon(
              product.isFavorite ? Icons.favorite : Icons.favorite_border,
            ),
            color: Theme.of(context).accentColor,
            onPressed: () {
              product.toggleFavoriteStatus(
                authData.token,
                authData.userId,
              );
            },
          ),
    ),
    title: Text(
      product.title,
      textAlign: TextAlign.center,
    ),
    trailing: IconButton(
      icon: Icon(
        Icons.shopping_cart,
```

```
            ),
            onPressed: () {
              cart.addItem(product.id, product.price, product.title);
              Scaffold.of(context).hideCurrentSnackBar();
              Scaffold.of(context).showSnackBar(
                SnackBar(
                  content: Text(
                    'Added item to cart!',
                  ),
                  duration: Duration(seconds: 2),
                  action: SnackBarAction(
                    label: 'UNDO',
                    onPressed: () {
                      cart.removeSingleItem(product.id);
                    },
                  ),
                ),
              );
            },
            color: Theme.of(context).accentColor,
          ),
        ),
      ),
    );
  }
}
```

This file controls the functionality of saving a product to favorites and adding it to cart, alongside with the different buttons that show up on the various different product boxes. These icons are gathered from Material Design which is imported in the beginning.

Product Grid.dart:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../providers/products.dart';
import './product_item.dart';
```

```
class ProductsGrid extends StatelessWidget {
  final bool showFavs;

  ProductsGrid(this.showFavs);

  @override
  Widget build(BuildContext context) {
    final productsData = Provider.of<Products>(context);
    final products = showFavs ? productsData.favoriteItems : productsData.items;
    return GridView.builder(
      padding: const EdgeInsets.all(10.0),
      itemCount: products.length,
      itemBuilder: (ctx, i) => ChangeNotifierProvider.value(
            value: products[i],
            child: ProductItem(
              ),
          ),
      gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
        crossAxisCount: 2,
        childAspectRatio: 3 / 2,
        crossAxisSpacing: 10,
        mainAxisSpacing: 10,
      ),
    );
  }
}
```

This file controls the properties of the grid that the products will be showcased in. This includes on how many products should be displayed per row based on information such as user screen size. Furthermore, it will also take the number of products to ensure that enough rows are created to showcase all of them.

User Product Item.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../screens/edit_product_screen.dart';
import '../providers/products.dart';
```

```
class UserProductItem extends StatelessWidget {
  final String id;
  final String title;
  final String imageUrl;

  UserProductItem(this.id, this.title, this.imageUrl);

  @override
  Widget build(BuildContext context) {
    final scaffold = Scaffold.of(context);
    return ListTile(
      title: Text(title),
      leading: CircleAvatar(
        backgroundImage: NetworkImage(imageUrl),
      ),
      trailing: Container(
        width: 100,
        child: Row(
          children: <Widget>[
            IconButton(
              icon: Icon(Icons.edit),
              onPressed: () {
                Navigator.of(context)
                    .pushNamed(EditProductScreen.routeName, arguments: id);
              },
              color: Theme.of(context).primaryColor,
            ),
            IconButton(
              icon: Icon(Icons.delete),
              onPressed: () async {
                try {
                  await Provider.of<Products>(context, listen: false)
                      .deleteProduct(id);
                } catch (error) {
                  scaffold.showSnackBar(
```

```
                SnackBar(
                  content: Text('Deleting failed!', textAlign: TextAlign.center,),
                ),
              );
            }
          },
          color: Theme.of(context).errorColor,
        ),
      ],
    ),
  ),
 );
 }
}
```

This file is in charge of deleting products from the database and showing the correct icons for the user with the correct functionality. The edit button transports the user to the edit product screen and the trash button sends a command that is found on another file that sends a command to the database to remove the product permanently. This file also includes the correct messages to show a user if their request is met successfully.

Main.dart:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import './screens/products_overview_screen.dart';
import './screens/product_detail_screen.dart';
import './providers/products.dart';
import './providers/cart.dart';
import './providers/orders.dart';
import './providers/auth.dart';
import './screens/orders_screen.dart';
import './screens/user_products_screen.dart';
import './screens/edit_product_screen.dart';
import './screens/auth_screen.dart';

void main() => runApp(MyApp());
```

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider.value(
          value: Auth(),
        ),
        ChangeNotifierProxyProvider<Auth, Products>(
          builder: (ctx, auth, previousProducts) => Products(
                auth.token,
                auth.userId,
                previousProducts == null ? [] : previousProducts.items,
              ),
        ),
        ChangeNotifierProvider.value(
          value: Cart(),
        ),
        ChangeNotifierProxyProvider<Auth, Orders>(
          builder: (ctx, auth, previousOrders) => Orders(
                auth.token,
                auth.userId,
                previousOrders == null ? [] : previousOrders.orders,
              ),
        ),
      ],
      child: Consumer<Auth>(
        builder: (ctx, auth, _) => MaterialApp(
              title: 'Tyres Online AE',
              theme: ThemeData(
                primarySwatch: Colors.red,
                accentColor: Colors.grey,
                fontFamily: 'Lato',
              ),
              home: auth.isAuth ? ProductsOverviewScreen() : AuthScreen(),
              routes: {
```

```
            ProductDetailScreen.routeName: (ctx) => ProductDetailScreen(),
            CartScreen.routeName: (ctx) => CartScreen(),
            OrdersScreen.routeName: (ctx) => OrdersScreen(),
            UserProductsScreen.routeName: (ctx) => UserProductsScreen(),
            EditProductScreen.routeName: (ctx) => EditProductScreen(),
          },
        ),
      ),
    );
  }
}
```

Finally, the main.dart file is essentially the compiler for the application it brings all the pages together and tells the CPU what to load first and what should be loaded last. As well as include basic information about the application such as App Name and App Icon. It also includes the font and color scheme that should be used for all pages in the application to ensure consistency. It links the various functions to their desired screens.