Shop App

Widget(Stateful, Stateless)

MyApp() - Stateless

- MultiProvider()
  - providers:
    - ChangeNotifierProvider.value()
      - **Auth**()
    - ChangeNotifierProxyProvider<Auth, Products>()
      - **Products**(token, userId, previousProducts)
    - ChangeNotifierProvider.value()
      - **Cart**()
    - ChangeNotifierProxyProvider<Auth, Orders>()
      - **Orders**(token, userId, previousOrders)
  - child:
    - Consumer<Auth>()
      - materialApp()
        - title: ‘MyShop’
        - theme: Themes
        - home: **Auth**.isAuthenticated
          - true: *ProductsOverviewScreen()*
          - false: FutureBuilder()
            - future: **Function *Auth.****tryAutoLogin()*
            - builder: ConnectionState.waiting
              - true: *SplashScreen()*
              - false: *AuthScreen()*
        - routes:
          - *ProductDetailScreen()*
          - *CartScreen()*
          - *OrdersScreen()*
          - *UserProductsScreen()*
          - *EditProductScreen()*

ProductsOverviewScreen() - Stateful

- Scaffold()
  - appbar: AppBar()
    - title: Text() -> ‘MyShop’
    - actions:
      - PopupMenuButton() => showOnlyFavorites
        - PopupMenuItem()
          - Text() > ‘Only Favorites’
        - PopupMenuItem()
          - Text() -> ‘Show All’
      - Consumer<**Cart**>()
        - builder: *Badge(widget, **cart**.itemCount)*
        - child: IconButton(shopping_cart) => Navigator -> *CartScreen()*
  - drawer: *appDrawer()*
  - body: isLoading
    - true: Center()
      - CircularProgressIndicator()
    - false: *ProductsGrid(showOnlyFavorites)*

SplashScreen() - Stateless

- Scaffold()
  - body: Center()
    - Text() -> ‘Loading…’

AuthScreen() - Stateless

- Scaffold()
  - body: Stack()
    - Container()
    - SingleChildScrollView()
      - SizedBox()
        - Column()
          - Flexible()
            - Container()
              - Text() -> ‘MyShop’
          - Flexible()
            - *AuthCard()*

ProductDetailScreen() - Stateless

- Scaffold()
  - CustomScrollView()
    - SliverAppBar()
      - FlexibleSpaceBar()
        - title: Text() -> loadedProduct.title
        - background: Hero()
          - Image.network() -> loadedProduct.imageUrl
    - SliverList()
      - SliverChildListDelegate
        - SizedBox()
        - Text() -> loadedProduct.price
        - SizedBox()
        - Container()
          - Text() -> loadedProduct.description
        - SizedBox()

CartScreen() - Stateless

- Scaffold()
  - appBar: AppBar()
    - Text() -> 'Your Cart'
  - body: Column()
    - Card()
      - Padding()
        - Row()
          - Text() -> 'Total'
          - Spacer()
          - Chip()
            - Text() -> **cart**.totalAmount
          - *<u>OrderButton</u>(**cart**)*
    - SizedBox()
    - Expanded()
      - ListView.builder()
        - itemCount: **cart**.items.length
        - itemBuilder: *<u>CartItem</u>(id, productId, price, quantity, title)*

OrdersScreen() - Stateless

- Scaffold()
  - appBar: AppBar()
    - title: Text() -> ‘Your Orders’
  - drawer: *AppDrawer()*
  - body: FutureBuilder()
    - future: **Function *Orders***.*fetchAndSetOrders()*
    - builder: ConnectionState.waiting
      - true:Center()
        - CircularProgressIndicator()
      - false: error
        - true: Center()
          - Text() -> ‘An error occurred!’
        - false: Consumer<**Orders**>
          - ListView.builder()
            - itemCount: **orders**.length
            - itemBuilder: *OrderItem(**orders**)*

UserProductsScreen() - Stateless

- Scaffold()
  - appBar: AppBar()
    - title: Text() -> ‘Your Products’
    - actions: IconButton(+) => Navigate -> *EditProductScreen()*
  - drawer: *AppDrawer()*
  - body: FutureBuilder()
    - future: **Function** *refreshProducts(context)*
    - builder: ConnectionState.waiting
      - true: Center()
        - CircularProgressIndicator()
      - false: RefreshIndicator() => **Function** *refreshProducts(context)*
        - Consumer<**Products**>()
          - Padding()
            - ListView.builder()
              - itemCount: **Products**.items.length
              - itemBuilder: Column()

- *UserProductItem(id, title, imageUrl)*
- Divider()

EditProductScreen() - Stateful

- Scaffold()
  - appBar: AppBar()
    - title: Text() -> 'Edit Product'
    - actions: IconButton(save) => **Function** *saveForm*
  - body: isLoading
    - true: Center()
      - CircularProgressIndicator()
    - false: Padding()
      - Form()
        - ListView()
          - TextFormField() => saveTitle
          - TextFormField() => savePrice
          - TextFormField() => saveDescription
          - Row()
            - Container()
              - imageUrl.isEmpty
                - true: Text() -> 'Enter a URL'
                - false: FittedBox() => Image.network() -> imageUrl
            - Expanded()
              - TextFormField() => saveImageUrl

Badge(widget, cart.itemCount) - Stateless

- Stack()
  - widget
  - Positioned()
    - Container()
      - Text() -> **cart**.itemCount

AppDrawer() - Stateless

- Drawer()
  - Column()
    - AppBar()
      - title: Text() -> 'Hello Friend!'

    - Divider()
    - ListTile() => Navigate -> '/'
        - leading: Icon(shop)
        - title: Text() -> 'Shop'
    - Divider()
    - ListTile() => Navigate -> <u>*OrdersScreen()*</u>
        - leading: Icon(payment)
        - title: Text() -> 'Orders'
    - Divider()
    - ListTile() => Navigate -> <u>*UserProductsScreen()*</u>
        - leading: Icon(edit)
        - title: Text() -> 'Manage Products'
    - Divider()
    - ListTile() => Navigate -> *pop(), '/';* **Function *Auth***.<u>*logout()*</u>
        - leading: Icon(exit_to_app)
        - title: Text() -> 'Logout'

ProductsGrid(showFavorites) - Stateless

- GridView.builder()
    - itemCount: **Products**.length
    - itemBuilder: ChangeNotifierProvider.value()
        - <u>*ProductItem()*</u>

AuthCard() - Stateful

- Card()
    - AnimatedContainer()
        - Form()
            - SingleChildScrollView()
                - Column()
                    - TextFormField() => saveEmail
                    - TextFormField() => savePassword
                    - AnimatedContainer()
                        - FadeTransition()
                            - SlideTransition()
                                - TextFormField() => saveAuthMode
                    - SizedBox()

        - isLoading
            - true: CircularProgressIndicator()
            - false: ElevatedButton()
                - Text() -> ‘Signup’ / ‘Login’
        - TextButton(authMode) => **Function** *switchAuthMode()*

OrderButton(Cart) - Stateful

- TextButton() => **Function *Orders***.*addOrder(**Cart**.items, totalAmount),* **Function *Cart***.*clear()*
    - isLoading
        - true: CircularProgressIndicator()
        - false: Text() -> ‘ORDER NOW’

CartItem(id, productId, price, quantity, title) - Stateless

- Dismissible() => **Function *Cart***.*removeItem(productId)*
    - background: Container()
        - Icon(delete)
    - confirmDismiss: showDialog()
        - AlertDialog()
            - title: Text() -> ‘Sure’
            - content: Text() -> ‘Description’
            - actions:
                - TextButton(‘No’)
                - TextButton(‘Yes’) => Navigate -> Pop()
    - child: Card()
        - Padding()
            - ListTile()
                - leading: CircleAvatar()
                    - Padding()
                        - FittedBox()
                            - Text(0 -> price
                - title: Text() -> title
                - subtitle: Text() -> total
                - trailing: Text() -> quantity

OrderItem(Order) - Stateful

- AnimatedContainer()
  - Card()
    - Column()
      - ListTile()
        - title: Text() -> **Order**.amount
        - subtitle: Text() -> **Order**.dateTime
        - trailing: IconButton(expand_more/expand_less) => expanded
      - AnimatedContainer()
        - ListView()
          - Row()
            - Text() -> **Product**.title
            - Text() -> **Product**.quantity * **Product**.price

UserProductItem(id, title, imageUrl) - Stateless

- ListTile()
  - title: Text() -> title
  - leading: CircleAvatar()
    - NetworkImage() -> imageUrl
  - trailing: SizedBox()
    - Row()
      - IconButton(edit) => Navigate -> *<u>EditProductScreen</u>(id)*
      - IconButton(delete) =>
        - try: **Function *Products***.*<u>deleteProduct</u>(id)*
        - catch: ScaffoldMessenger.showSnackbar()
          - SnackBar()
            - Text() -> ‘Deleting Failed’

ProductItem() - Stateless

- ClipRRect()
  - GridTile()
    - child: GestureDetector() => *<u>ProductDetailScreen</u>(**Product**.id)*
      - Hero()
        - FadeInImage() -> **Product**.imageUrl
          - AssetImage()

- footer: GridTileBar()
  - leading: Consumer<**Product**>()
    - IconButton(favorite/favorite_border) => **Function** ***Product***.*toggleFavoriteStatus(token, userId)*
  - title: Text() -> **Product**.title
  - trailing: IconButton(shopping_cart) => **Function** ***Cart***.*addItem(id, price, title)*
    - ScaffoldMessenger.hideCurrentSnackBar()
    - ScaffoldMessenger.showSnackBar()
      - SnackBar() => SnackBarAction('UNDO') => **Function** ***Cart***.*removeSingleItem(**Product**.id)*
        - Text() -> 'Item Added'

Functions

Auth.tryAutoLogin()

- prefs.containsKey('UserData')
  - false: **false**
- expiryDate.isBefore(now)
  - false: **false**
- setToken
- setUserId
- setExpiryDate
- notifyListeners()
- **Function *Auth.***_autoLogout()_
- **true**

Orders.fetchAndSetOrders()

- extractedData.isNull
  - true: null
- extractedData.add(OrderItem*(id, amount, dateTime, products)*)
- notifyListeners()

refreshProducts(context)

- **Function *Products***.*fetchAndSetProducts()*

saveForm()

- ifFormValid

    - false: dontSave
- ifProductExist
    - true: **Function *Products*.*<u>updateProduct</u>(id, **Product**)***
    - false: try: **Function *Products*.*<u>addProduct</u>(**Product**)***
        - catch: showDialog()
            - AlertDialog() => TextButton('Okay') => navigate -> pop()
                - Text() - 'error'
- Navigate -> pop()

Auth.logout()

- authTimer = null
- notifyListeners()
- prefs.clear()

switchAuthMode()

- switchAuthMode

Orders.addOrder(cartProducts, total)

- orders.insert(*<u>OrderItem</u>(id, amount, dateTime, products)*)
- notifyListeners()

Cart.clear()

- items.clear
- notifyListeners()

Cart.removeItem(productId)

- items.remove(productId)
- notifyListeners()

Products.deleteProduct(id)

- items.removeAt(existingProductIndex)
- notifyListeners()
- fail: items.insert(existingProductIndex, existingProduct)
    - notifyListeners()

Product.toggleFavoriteStatus(token, userId)

- newStatus
- notifyListeners()
- fail: **Function Product**.*<u>setFavValue</u>(oldStatus)*

Cart.addItem(productId, price, title)

- items.containsProduct
    - true: items.update(productId, *CartItem(id, title, price, quantity++)*)
    - false: items.putIfAbsent(productId, *CartItem(id, title, price, 1)*)
- notifyListeners()

Cart.removeSingleItem(productId)

- items.quantity > 1
    - true: items.update(productId, *CartItem(id, title, price, quantity—)*)
    - false: items.remove(productId)
- notifyListeners()

Auth.autoLogout()

- setAuthTimers

Products.fetchAndSetProducts(filterByUser)

- Items = loadedFilteredProducts
- notifyListeners()

Products.updateProduct(id, newProduct)

- updateProduct
- notifyListeners()

Products.addProduct(Product)

- items.add(Product)
- notifyListeners()

Product.setFavValue(newValue)

- isFavorite = newValue
- notifyListeners()