## **Final Project Documentation**

**Application Name:** *Instaham*

**Author:** *Eduardo Aguilar*

**Date:** 10/06/2025

### **1. Project Overview**

#### **1.1 Problem Statement**

Provide a 2-3 sentence description of the problem your application solves or the need it fulfills.

**Instaham will be an app that provides the user with a way to express themselves by being able to post, comment, and socialize with other users.**

#### **1.2 Target Audience**

Describe the intended user of your application.

**For example**: "This app is designed for anyone that is 18 and older and are looking for another way to socialize with friends, family, and/or co-workers.

#### **1.3 Core Features**

Provide a bulleted list of the main features of your application.

* *Feature 1: Users will be able to post publicly.*
* *Feature 2: Users will be able to comment under their posts and other users’ posts.*
* *Feature 3: User will have a personal profile.*
* *Feature 4: User can create an account with an email address and password.*

### 

### 

### **2. Technical Design & Architecture**

#### **2.1 State Management Strategy**

Describe the state management approach you chose and why it was appropriate for your application's complexity.

**I used Flutter’s built-in Stateful Widget and Stateless widget for text fields, navigation toggles, and buttons. I included per page its own state like controllers for the email and password fields and a couple of booleans for if they had an account or not.**

#### **2.2 Data Model**

Describe the main data structures or classes in your application. Include a code block for your primary model(s).

**I tried using the firebase authentication data model to capture user input that way i didn’t have to use a local database or API backend. But i was confused on installing the firebase.**

class LoginPage extends StatefulWidget {

final Function()? onTap;

const LoginPage({super.key, required this.onTap});

@override

State<LoginPage> createState() => \_LoginPageState();

}

final emailTextController = TextEditingController();

final passwordTextController = TextEditingController();

void signIn() async {

showDialog(

context: context,

builder: (context) => const Center(

child: CircularProgressIndicator(),

),

);

try {

await FirebaseAuth.instance.signInWithEmailAndPassword(

email: emailTextController.text,

password: passwordTextController.text,

);

if (context.mounted) Navigator.pop(context);

} on FirebaseAuthException catch (e) {

Navigator.pop(context);

displayMessage(e.code);

}

}

void displayMessage(String message) {

showDialog(

context: context,

builder: (context) => AlertDialog(

title: Text(message),

),

);

}

#### 

#### **2.3 Persistence / API Strategy**

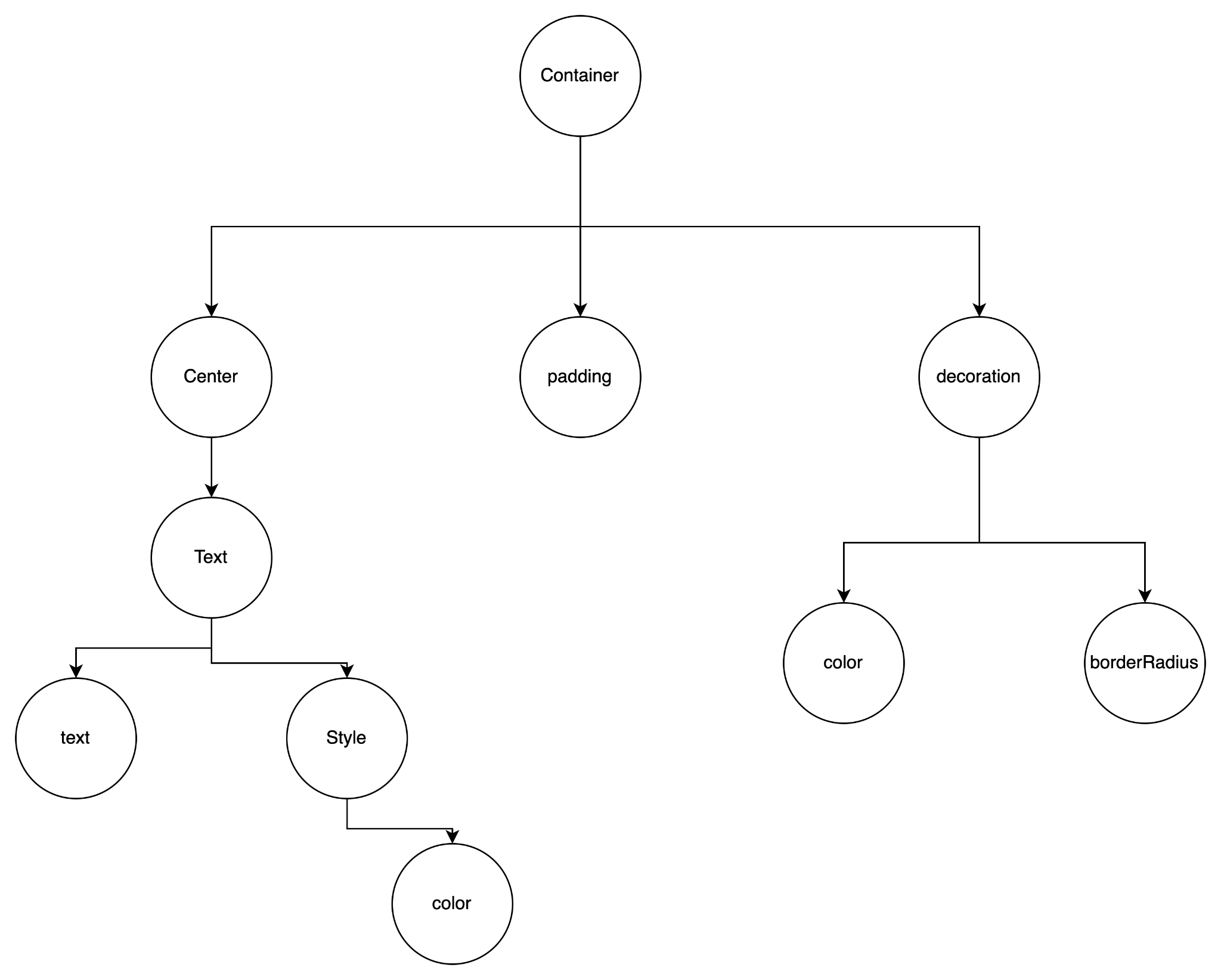
Explain how your application handles data. If you used local persistence, describe the method and data format. If you used a web API, list the endpoints you consumed.

**Example (Persistence)**: "The application persists the user's list of trips locally using file I/O. The list of **Trip** objects is serialized into a JSON string and saved to a file named **trips.json** in the application's documents directory, managed via the **path\_provider** package."

**Example (API)**: "The application fetches real-time currency conversion rates from the ExchangeRate-API. The primary endpoint consumed is: **https://api.exchangerate-api.com/v4/latest/USD**."

#### **2.4 Widget Tree Diagram**

Provide **draw.io** diagram of the widget tree for **one** of your main screens. This is for button page



Example > <https://docs.flutter.dev/assets/images/docs/ui/layout/sample-flutter-layout.png>

### 

### **3. Setup & Installation**

#### **3.1 Prerequisites**

* Firebase Studio and Flutter SDK

#### **3.2 Installation Steps**

1. Clone the public repository you created: **git clone <your-repo-url>**
2. Navigate into the project directory: **cd <your-project-directory>**
3. Set up your Flutter files: **flutter create**
4. After adding dependencies: **flutter pub get**
5. Run the application: **flutter run**

### 

### **4. Automated Testing**

#### **4.1 Testing Strategy**

Briefly describe your approach to testing.

**For example**: "The testing strategy focused on validating the business logic in the data models (unit tests) and ensuring that user interactions on the most critical screens triggered the correct state changes (widget tests)."

#### **4.2 Test Case Examples**

Provide at least one unit test and one widget test example from your project. For each, describe its purpose, the AAA steps, and include the code snippet.

**Test Case 1: Unit Test**

* **Description:** This test verifies that the **Trip.fromJson()** factory constructor correctly parses a map into a **Trip** object.
* **Arrange:** A **Map<String, dynamic>** representing a valid trip is created.
* **Act:** The **Trip.fromJson()** constructor is called with the map.
* **Assert:** The properties of the resulting **Trip** object are checked to ensure they match the original map's values.

**Code Snippet:**  
 test('Trip.fromJson creates a valid Trip object from a map', () {

// Arrange

final tripJson = {

'id': 'trip1',

'destination': 'Paris',

'startDate': '2025-10-10T00:00:00.000',

'endDate': '2025-10-20T00:00:00.000',

};

// Act

final trip = Trip.fromJson(tripJson);

// Assert

expect(trip.id, 'trip1');

expect(trip.destination, 'Paris');

});

**Test Case 2: Widget Test**

* **Description:** This test verifies that tapping the "Add to Favorites" **IconButton** on the **TripDetailsScreen** correctly toggles the icon's state from outlined to filled.
* **Arrange:** The **TripDetailsScreen** widget is pumped with a sample **Trip** object.
* **Act:** The test finds the **IconButton** by its icon and simulates a tap. **tester.pump()** is called to rebuild the UI.
* **Assert:** The test verifies that the **Icons.favorite\_border** is no longer present and that **Icons.favorite** now is.

**Code Snippet:**  
 testWidgets('Tapping favorite button toggles icon', (WidgetTester tester) async {

// Arrange

await tester.pumpWidget(MaterialApp(home: TripDetailsScreen(trip: sampleTrip)));

// Act

await tester.tap(find.byIcon(Icons.favorite\_border));

await tester.pump();

// Assert

expect(find.byIcon(Icons.favorite\_border), findsNothing);

expect(find.byIcon(Icons.favorite), findsOneWidget);

});

### **5. User Guide**

Provide simple, step-by-step instructions for using the core features of your app.

1. **Home Page: You would notice the cover of the app.**
2. **Login Page: This is where you will log in and or create an account if you haven’t already. If you didn’t have an account it would redirect you to the registration page.**
3. **Registration Page: This is where you would register as a new user.**
4. **User profile: This is where you would go once logged in where you can post, comment, and like other users’s posts.**