Mobile Development :

13 : Revision + Testing Mobile Apps

Professor Imed Bouchrika
National School of Artificial Intelligence
imed.bouchrika@ensia.edu.dz

Outline :

- *Revision & Case Studies*
  - *Databases, State Management, Networking*
  - *Uploading Images*
- *Exception Handling in Flutter*
- *Testing Mobile Apps*
  - *Testing Concepts*
  - *Debugging & Profiling*
  - *Unit Testing*
  - *UX/UI Testing*
  - *Advanced Tools for Testing*

Topics being covered

- **Part 1:**
  - *Validating Ideas for Mobile Apps*
  - *UI and UX Design Principles*
- **Part 2:**
  - *Building Apps using Kotlin*
  - *Building Apps using Flutter*
  - *Flutter : State Management, Navigation, Databases, Networking, Firebase..*
- **Part 3:**
  - *Case Studies and Examples with Backends*
- **Part 4:**
  - *Testing the App*
  - *Publishing and Promoting the App.*

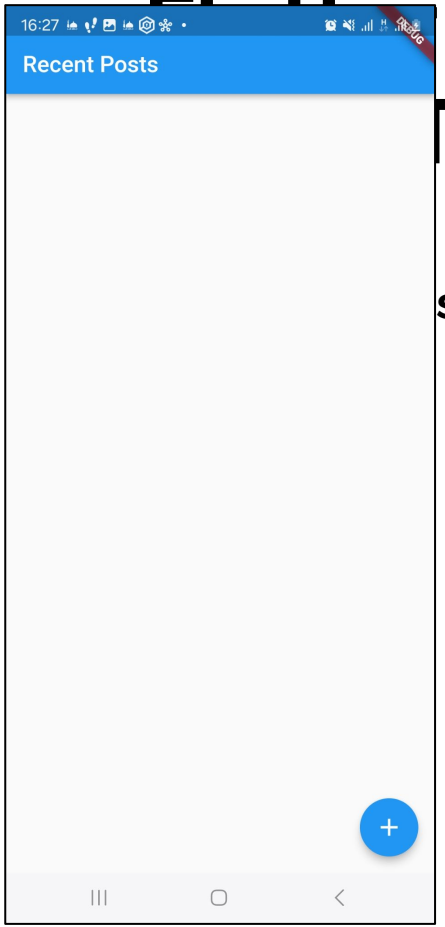
Section 1

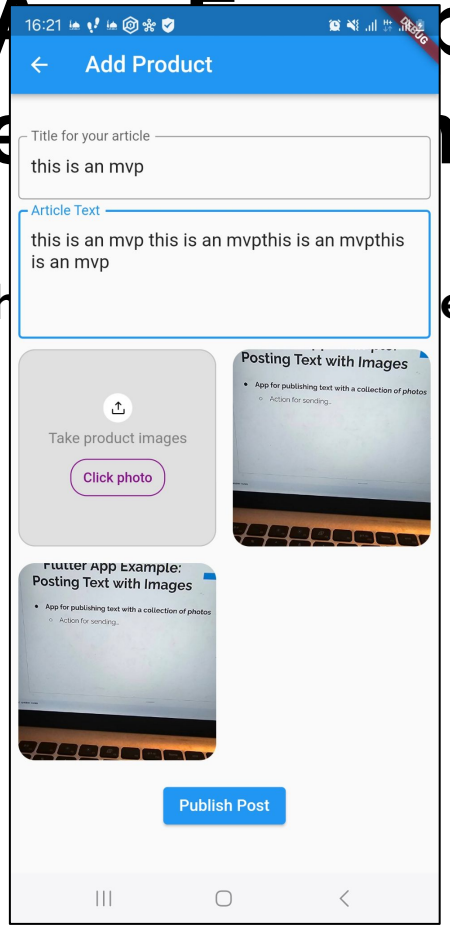
More case studies and examples

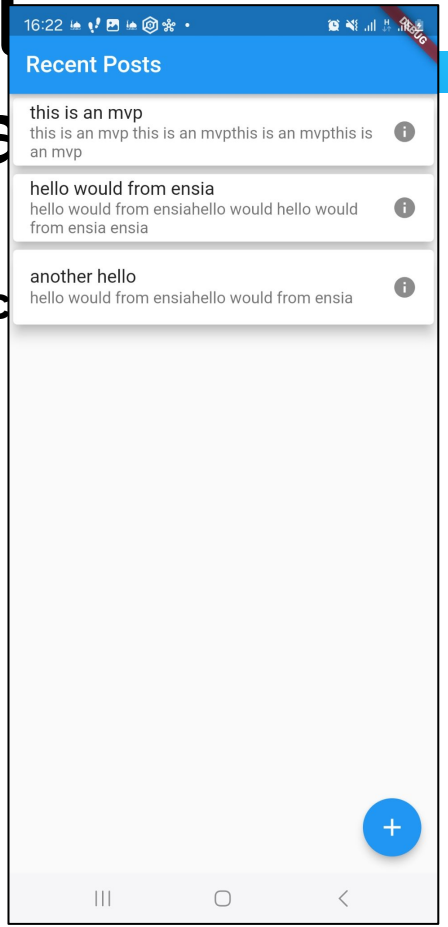
Flutter App Example: Posting Text with Images

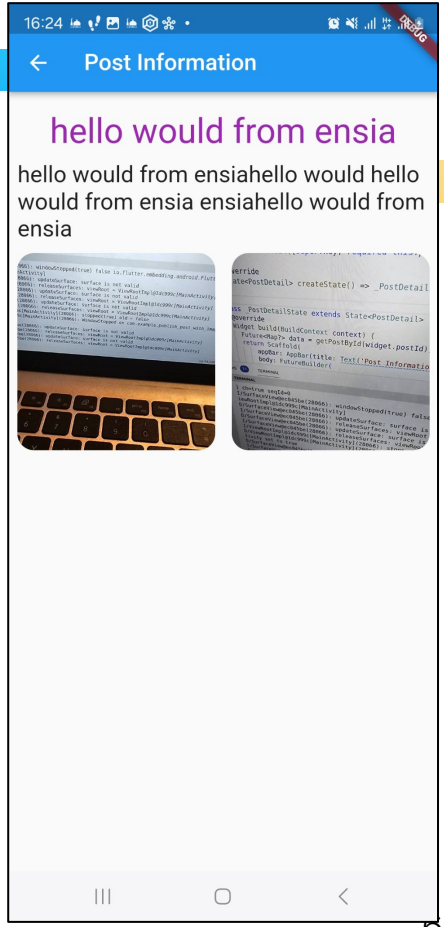
- **App for publishing text with a collection of photos**
  - There are cases where the images needs to be available to other uses
    - Therefore, there is a need to upload them online

MVP


16:27
Recent Posts

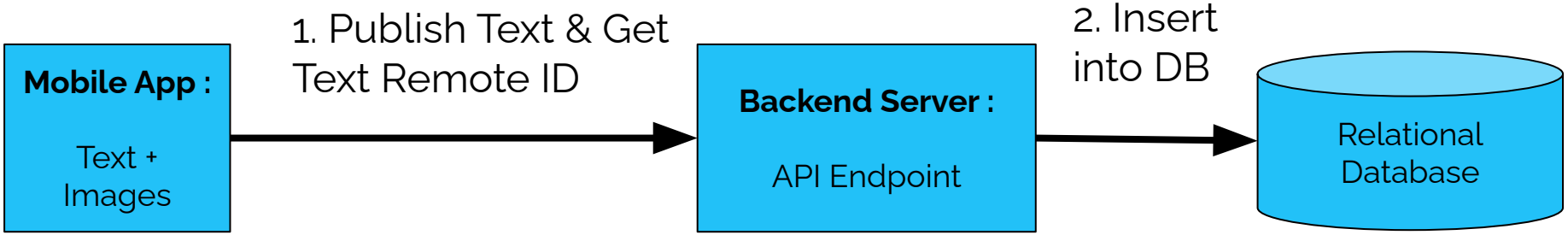

16:21
Add Product
Title for your article
this is an mvp
Article Text
this is an mvp this is an mvpthis is an mvpthis is an mvp
Take product images
Click photo
Posting Text with Images
App for publishing text with a collection of photos
Action for sending...
Flutter App Example: Posting Text with Images
App for publishing text with a collection of photos
Action for sending...
Publish Post


16:22
Recent Posts
this is an mvp
this is an mvp this is an mvpthis is an mvpthis is an mvp
hello would from ensia
hello would from ensiahello would hello would from ensia ensia
another hello
hello would from ensiahello would from ensia


16:24
Post Information
hello would from ensia
hello would from ensiahello would hello would from ensia ensiahello would from ensia

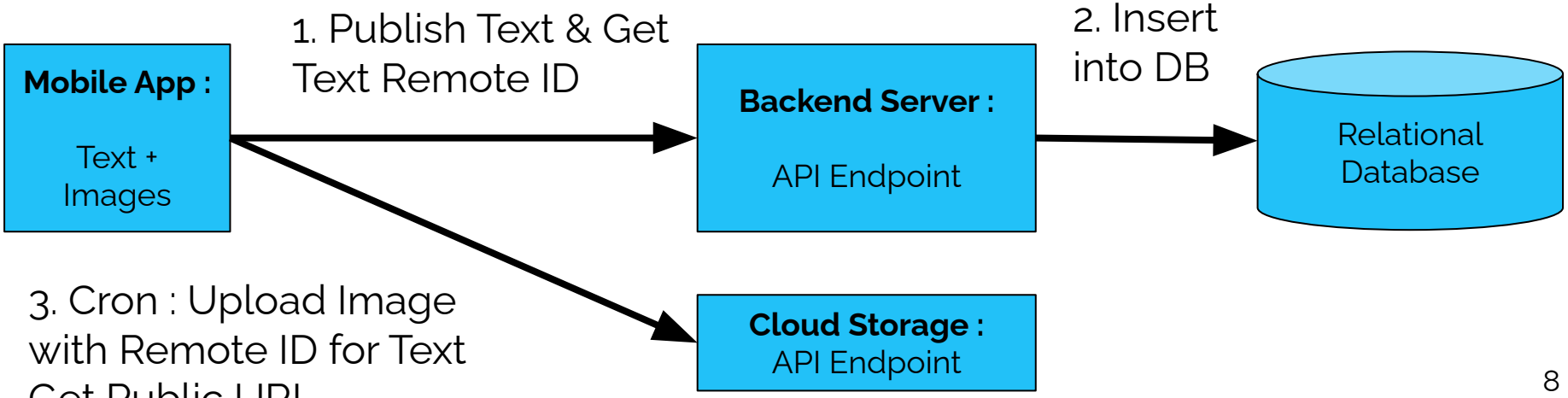
Flutter App Example: Posting Text with Images

- **App for publishing text with a collection of photos**
  - Architecture



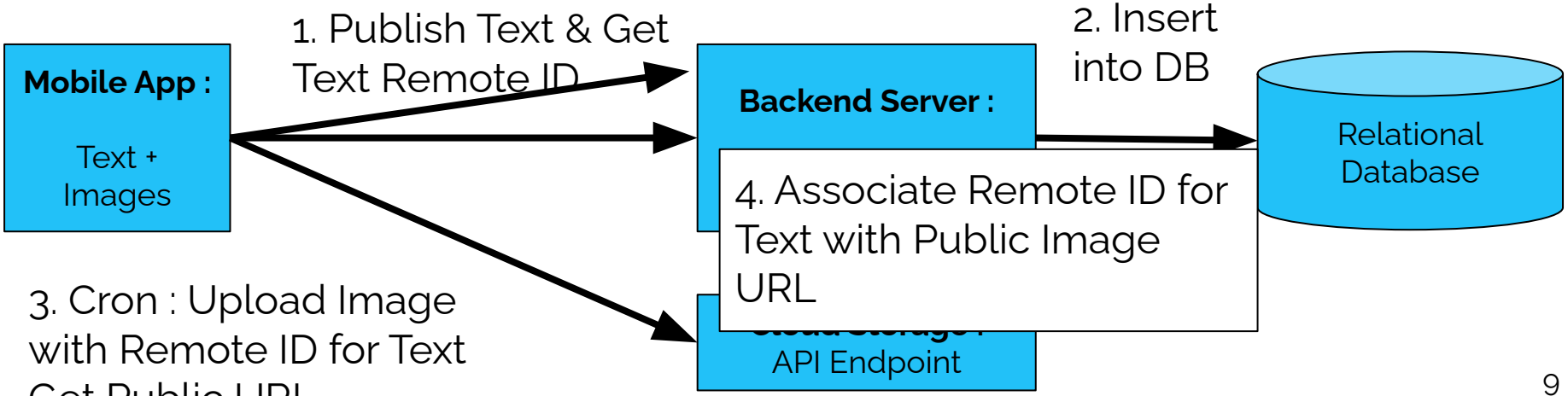
Flutter App Example: Posting Text with Images

- **App for publishing text with a collection of photos**
  - Architecture



Flutter App Example: Posting Text with Images

- **App for publishing text with a collection of photos**
  - Architecture



Flutter App Example: Posting Text with Images

- **App for publishing text with a collection of photos**
  - Interaction :
    - Clicking Post
      - Send Directly to the Remote Server (Obligatory to have internet Connection)
      - Save Image to local DB
        - Sequentially send Images to Cloud storage
        - Associate Remote Post with Image URL

Flutter App Example: Posting Text with Images

- **App for publishing text with a collection of photos**
  - Interaction :
    - Clicking Post
      - Save Post in Local Database
      - When there is internet connection:
        - Sync newly added posts to the Remote Server
      - Save Image to local DB
        - Sequentially send Images to Cloud storage
        - Associate Remote Post with Image URL

Flutter App Example: Posting Text with Images

- **App for publishing text with a**
  - Main file

```dart
import 'package:flutter/material.dart';
import 'package:publish_post_with_images_mvp/screens/listing.dart';

import 'utils/posts.dart';

void main() async {
 init_my_data();
 runApp(const MainApp());
}

class MainApp extends StatelessWidget {
 const MainApp({super.key});

 @override
 Widget build(BuildContext context) {
   return const MaterialApp(
     home: ListingPost(),
   );
 }
}
```

Flutter App Example: Posting Text with

- **App for publishing text with a**
  - posts.dart for handling data
    (Using this utility File to Decouple UI from DB)

```dart
late List<Map> post_data;

Future<bool> init_my_data() async {
 post_data = [];
 return true;
}

Future<Map> publishPost(Map data) async {
 int max_id = 0;
 for (Map line in post_data) {
   if (line['id'] > max_id) max_id = line['id'];
 }
 data['id'] = max_id + 1;
 post_data.add(data);

 return {'status': 1, 'message': 'data is added successfully.'};
}

Future<List<Map>> getPosts() async {
 return post_data;
}

Future<Map?> getPostById(int id) async {
 for (Map line in post_data) {
   if (line['id'] == id) return line;
 }
 return null;
}
```

Flutter App Ex
Posting Text with

- **App for publishing text with**
  - Widget for Listing

    All posts

```
class ListingPost extends StatefulWidget {
 const ListingPost({super.key});

 @override
 State<ListingPost> createState() => _ListingPostState();
}

class  ListingPostState extends State<ListingPost> {
 @override
 Widget build(BuildContext context) {
   Future<List<Map>> allpost = getPosts();
   return Scaffold(
       appBar: AppBar(title: Text('Recent Posts')),
       floatingActionButton: FloatingActionButton(
         tooltip: 'New Post',
         onPressed: () {
           Navigator.of(context)
               .push(MaterialPageRoute(builder: (context) =>
NewPostWidget()));
         },
         child: const Icon(Icons.add, color: Colors.white, size: 28),
       ),
       body: Center(
           child: FutureBuilder(
               future: allpost,
               builder: (context, snapshot) {
                 if (snapshot.hasData) {
                   List<Map> myData = snapshot.data!;
                   return getListOfPostsWidget(myData);
                 } else if (snapshot.hasError) {
                   return Text("${snapshot.error}");
                 }

                 return CircularProgressIndicator();
               }))));
 }
```

Flutter App Exa
Posting Text with

- **App for publishing text with**
  - Widget for Listing

    All posts

Will the screen refresh automatically when I publish a post ?

```
class ListingPost extends StatefulWidget {
 const ListingPost({super.key});

 @override
 State<ListingPost> createState() => _ListingPostState();
}

class  ListingPostState extends State<ListingPost> {
 @override
 Widget build(BuildContext context) {
   Future<List<Map>> allpost = getPosts();
   return Scaffold(
       appBar: AppBar(title: Text('Recent Posts')),
       floatingActionButton: FloatingActionButton(
         tooltip: 'New Post',
         onPressed: () {
           Navigator.of(context)
               .push(MaterialPageRoute(builder: (context) =>
NewPostWidget()));
         },
         child: const Icon(Icons.add, color: Colors.white, size: 28),
       ),
       body: Center(
           child: FutureBuilder(
               future: allpost,
               builder: (context, snapshot) {
                 if (snapshot.hasData) {
                   List<Map> myData = snapshot.data!;
                   return getListOfPostsWidget(myData);
                 } else if (snapshot.hasError) {
                   return Text("${snapshot.error}");
                 }

                 return CircularProgressIndicator();
               }))));
 }
```

Flutter App Exa
Posting Text with

- **App for publishing text with**
  - Widget for Listing

    All posts

```
class ListingPost extends StatefulWidget {
 const ListingPost({super.key});

 @override
 State<ListingPost> createState() => _ListingPostState();
}

class  ListingPostState extends State<ListingPost> {
 @override
 Widget build(BuildContext context) {
   Future<List<Map>> allpost = getPosts();
   return Scaffold(
       appBar: AppBar(title: Text('Recent Posts')),
       floatingActionButton: FloatingActionButton(
         tooltip: 'New Post',
         onPressed: () {
           Navigator.of(context)
               .push(MaterialPageRoute(builder: (context) =>
NewPostWidget()))
               .then((value) {
             setState(() {});
           });
         },
         child: const Icon(Icons.add, color: Colors.white, size: 28),
       ),
       body: Center(
           child: FutureBuilder(
               future: allpost,
               builder: (context, snapshot) {
                 if (snapshot.hasData) {
                   List<Map> myData = snapshot.data!;
                   return getListOfPostsWidget(myData);
                 } else if (snapshot.hasError) {
                   return Text("${snapshot.error}");
                 }
                 return CircularProgressIndicator();
```

Flutter App Example: Posting Text with Images

- **App for publishing text with a collection of photos**
  - Simple Steps to turn the MVP into a fully functional App:
    - ?
    - ?
    - ?
    - ?
    - ?

Upon clicking **Publish Post**, we will post **immediately** to the server. User must be connected.

Flutter App Example: Posting Text with Images

- **App for publishing text with a collection of photos**
  - Simple Steps to turn the MVP into a fully functional App:
    - Mobile App:
      - Use Dio to get posts and publish posts from the EndPoints
      - Add Local Database to store images (Image[id,path,type)
      - Cron Job : Upload Images to Cloud Storage + Link Images to posts
    - Implement Backend Service**s**

Flutter App Exa
Posting Text with

- **App for publishing text with**
  - Simple Steps to turn the
    - Mobile App:
      - Use Dio to get p
      - Add Local Datal
      - Cron Job : Uploa posts
    - Implement Backend

```
Future<List<Map>> getPosts() async {
 try {
   var response = await dio.get(api_endpoint_post_get);
   Map ret_data = json.decode(response.toString());
   List tmp = ret_data['data'];
   post_data = [];
   for (dynamic line in tmp) post_data.add(Map.of(line));

   return post_data;
 } catch (e, stack) {
   print('Error >>>>>>>> $e \n $stack');
   return post_data;
 }
}

Future<Map?> getPostById(int id) async {
 try {
   var response = await dio.get('$api_endpoint_post_get_byId/$id');
   Map ret_data = json.decode(response.toString());
   Map tmp = Map.of(ret_data['data']);
   return tmp;
 } catch (e, stack) {
   print('Error >>>>>>>> $e \n $stack');
   return null;
 }
```

Flut
Postin

- App for
  - Sim
    -
    -

```
Future<Map> publishPost(Map<String, dynamic> data) async {
 try {
   var response =
       await dio.post(api endpoint post publish, data: FormData.fromMap(data));
   Map ret data = json.decode(response.toString());
   if (ret data['status'] == 200) {
     DBImage.scheduleImages(
         ret data['data']['id'], data['images'] as List<Map<String, dynamic>>);
     return {'status': 1, 'message': 'data is added successfully.'};
   } else {
     print('Error >>>> $ret_data');
   }
 } catch (e, stack) {
   print('Error >>>>>>>> $e \n $stack');
 }
 return {'status': 0, 'message': 'Error somewhere....'};
}
```

Flut
Postin

- App for
  - Sim
    -
    -

```dart
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';
import 'db_images.dart';

class DBHelper {
 static const  database name = "POST_PUBLISH_V4.db";
 static const _database_version = 1;
 static var database;

 static Future getDatabase() async {
   if (database != null) {
     return database;
   }
   List sql_create_code = [
     DBImage.sql_code,
   ];
   database = openDatabase(
     join(await getDatabasesPath(), _database_name),
     onCreate: (database, version) {
       for (var sql_code in sql_create_code) database.execute(sql_code);
     },
     version: _database_version,
     onUpgrade: (database, oldVersion, newVersion) {
       print(">>>>>>>>>>>>>$oldVersion vs $newVersion");
       if (oldVersion != newVersion) {
         database.execute('DROP TABLE IF EXISTS images');
         for (var sql_code in sql_create_code) database.execute(sql_code);
       }
     },
   );
   return database;
 }
}
```

Flut
Postin

- App for
  - Sim
    -
    -

```dart
import 'package:sqflite/sqflite.dart';
import 'db_helper.dart';
class DBImage {
 static const tableName = 'images';
 static const sql_code = '''CREATE TABLE IF NOT EXISTS images (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            remote_id INTEGER,
            imagePath TEXT,
            type TEXT,
            tried INTEGER,
            create_date TEXT
          )      ''';
 static Future<Map<String, dynamic>?> getSingleImageToUpload() async {
   final database = await DBHelper.getDatabase();
   List<Map<String, dynamic>> res = await database.rawQuery('''SELECT
           id, imagePath,remote_id from ${tableName}  where
           tried=0  order by id  ASC   limit 1 ''');
   if (res == null || res.isEmpty) return null;
   return res[0] ?? null;
 }
 static Future<bool> scheduleImages( int post_id, List<Map<String, dynamic>> images) async {
   final database = await DBHelper.getDatabase();
   for (Map<String, dynamic> img in images) {
     print("$img");
     img['remote_id'] = post_id.toString();
     img['tried'] = '0';
     await database.insert(DBImage.tableName, img,
         conflictAlgorithm: ConflictAlgorithm.replace);
   }
   return true;
 }
 static Future<bool> deleteRecord(int id) async {
   final database = await DBHelper.getDatabase();
   database.rawQuery("""delete from  ${tableName}  where id=?""", [id]);
   return true;
 }
```

Flut
Postin

- App for
  - Sim
    -
    -

```dart
Future<bool> cron_upload_images_for_posts() async {
 try {
   print('Getting Image to upload to the WEB......>>>>');
   Map<String, dynamic>? data_upload = await DBImage.getSingleImageToUpload();
   if (data_upload == null) return false;
   print("Data to upload $data_upload ");
   String? url = await image_upload_to_firebase(data_upload['imagePath']);
   if (url != null) {
     //Upload to Server...
     postAddImage(data_upload['remote_id'], url);
     //delete from mobile...
     DBImage.deleteRecord(data_upload['id']);
   }
 } catch (e, stack) {
   print('>>>> Cron Error $e \n $stack ');
 }
 return true;
}

Future<String?> image_upload_to_firebase(String local_path) async {
 try {
   var uuid = Uuid();
   String filename = uuid.v4() + '.jpg';
   final spaceRef = storageRef.child("myimages/$filename");
   await spaceRef.putFile(File(local_path));
   String imageUrl = await spaceRef.getDownloadURL();
   print("Image uploaded ---> $imageUrl");
   return imageUrl;
 } catch (e, stack) {
   print('Error >>>> $e $stack');
   return null;
 }
}
```

Flut
Postin

- App for
  - Sim
    -
    -

```dart
Future<String?> image_upload_to_s3_storage (String local_path) async {
 try {
   final formData = FormData.fromMap({
     'file': await MultipartFile.fromFile(local_path, filename: 'anyname.jpg'),
   });

   final response =
       await dio.post('$api_endpoint_image_upload ', data: formData);
   Map ret_data = json.decode(response.toString());
   if (ret_data['status'] == 200) {
     return ret_data['URL'];
   }
 } catch (e, stack) {
   print('Error >>>> $e $stack');
 }
 return null;
}
```

Flutter App Example: Posting Text with Images

- **App for publishing text with a collection of photos**
  - Backend Services:
    - Posts Service: (PHP, Flash/Vercel, Render..., JS...)
      - Publishing New Post
      - Getting Posts
      - Get Post By ID
      - Link Image to Post
    - Images Uploading Service (Firebase, S3,,.)

```
from flask import Flask, request
import json
from supabase import create_client, Client
app = Flask(__name__)

url="https://rmuifxmxzskeidlcnnmw.supabase.co"
key="eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InJtdWlmeG14enNrZWlkbGNubm13Iiwicm9sZSI6InNlcnZpY2Vfcm9sZSIsImlhdCI6MTcwMjQwMjk0OCwiZXhwIjoyMDE3OTc4OTQ4fQ.6HnmTNfoPh3I5LggqZEgKInd-Vmyn2ZEyyNSA8GY-YA"
supabase: Client = create_client(url, key)

@app.route('/post.get',methods=['GET','POST'])
def api_post_get():
    response = supabase.table('posts').select("*").order('id', desc=True).limit(20).execute()
    return json.dumps({'status':200,'message':'','data':response.data})


@app.route('/post.get.byId/<int:post_id>',methods=['GET','POST'])
def api_post_get_byId(post_id):
    response = supabase.table('posts').select("*").eq('id',post_id).execute()
    if len(response.data)==0:
        return json.dumps({'status':500,'message':'No data found'})

    ret=response.data[0]
    ret['images']=[]
    response = supabase.table('post_images').select("*").eq('post_id',post_id).execute()
    for line in response.data:
        ret['images'].append({
            'type':'network',
            'imagePath':line['url'],
        })


    return json.dumps({'status':200,'message':'','data':ret})
```

```python
@app.route('/post.publish',methods=['GET','POST'])
def api_post_publish():
    title= request.form.get('title')
    description= request.form.get('description')
    #For simplicity, no checks will be performed.
    response = supabase.table('posts').insert({"title": title, "description":
description}).execute()
    if len(response.data)==0:
        error='Error creating the user'
        return json.dumps({'status':500,'message':error})
    return json.dumps({'status':200,'message':'','data':response.data[0]})


@app.route('/post.add.image/<int:post_id>',methods=['GET','POST'])
def api_post_add_image(post_id):
    img_url= request.form.get('url')
    response = supabase.table('post_images').insert({"post_id": post_id, "url":
img_url}).execute()
    if len(response.data)==0:
        error='Error ....'+str(response)
        return json.dumps({'status':500,'message':error})
    return json.dumps({'status':200,'message':'','data':response.data[0]})


@app.route('/')
def home():
    return 'Hello Ensia Students!'
```

Section 2

Exceptions & Errors Handling in Flutter

Exception & Error Handling

- **When writing code :**
  - If there is a possibility that there will be an error or Exception in your code:
    - Enclose the code inside try catch
    - Print the Error Description and Stack Trace of the Error
  - Dart : has two concepts :
    - Errors : represents a program failure that the programmer should have avoided.
    - Exceptions : An Exception is intended to convey information to the user about a failure,

```
try {
  // Some code Here
} catch (e, stack) {
  print('Error >>>> $e $stack');
}
```

ll be an error,

- Enclose inside try catch
- Pri the Sta

Avoid this

```
try {
  // Some code Here
} on Exception catch (e, stack) {
  print('Error >>>> $e $stack');
}
```

Exception & Error Handling

- **Possible Scenarios to have errors :**
  - Getting data from an external source : Network, Database …
  - Converting or decoding data between different formats:
    - Json to Map or to List…
    - Map<dynamic,dynamic> to Map<String,,,,>

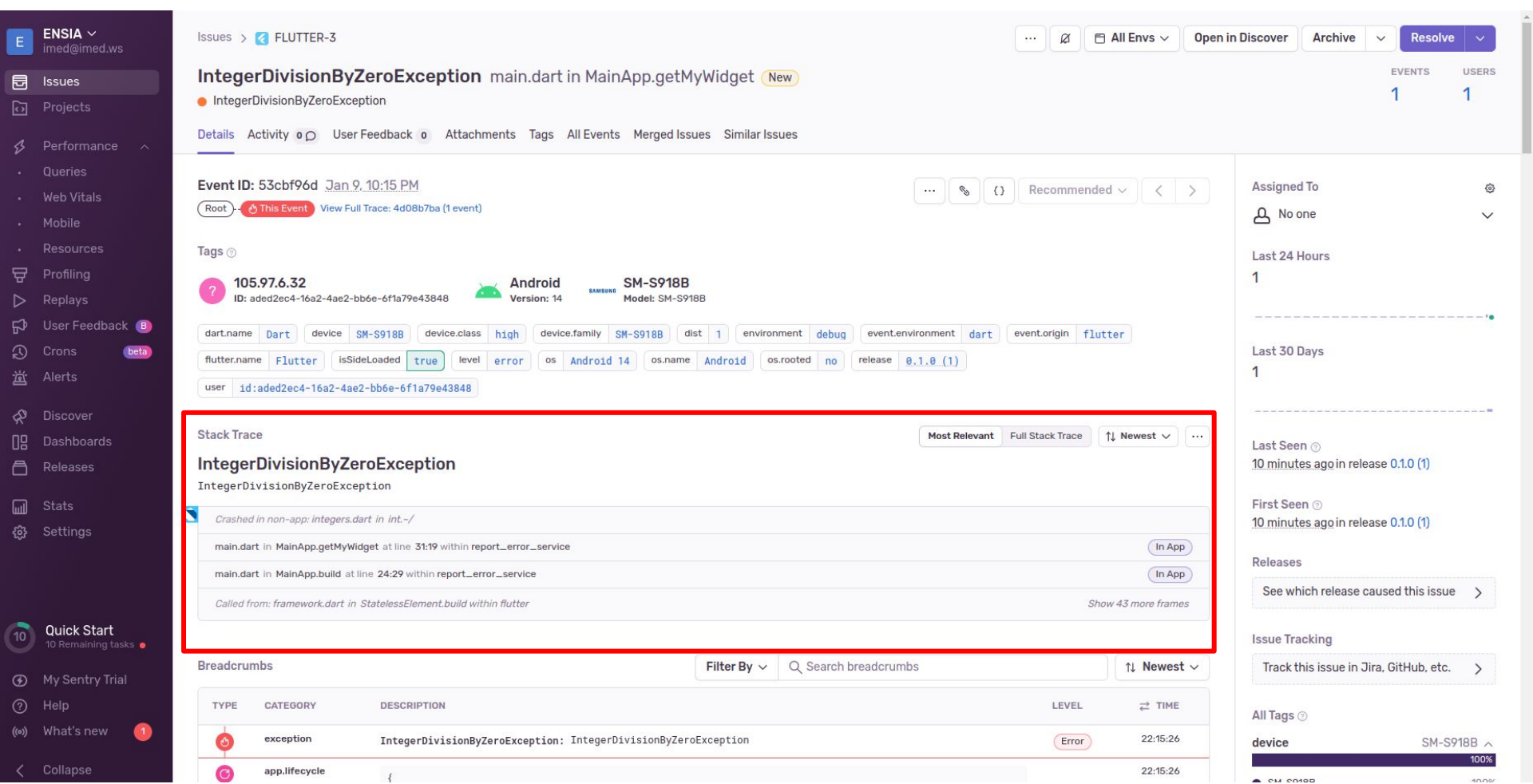
Exception & Error Handling

- **Logging and Monitoring Errors :**
  - Use Print or Log to the console as Much as YOU CAN
  - How to debug or trace an error for app not running on your phone away from you?

Exception & Error Handling

- **Logging and Monitoring Errors :**
  - Use Print or Log to the console as Much as YOU CAN
  - How to debug or trace an error for app not running on your phone away from you?
    - Use Error Monitoring Services like **sentry.io**:
      - **flutter pub add sentry_flutter**

```dart
import 'package:flutter/material.dart';
import 'package:sentry_flutter/sentry_flutter.dart';
void main() async {
 await SentryFlutter.init(
   (options) {
     options.dsn =
         'https://1401ebe9f95ad66dee18b95285e22a96@o4506543497674752.ingest.sentry.io/4506543501082624';
     options.tracesSampleRate = 1.0;
   },
   appRunner: () => runApp(MainApp()),
 );
}
class MainApp extends StatelessWidget {
 const MainApp({super.key});
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: Scaffold(
       body: Center(child: getMyWidget('Hello')),
     ),
   );
 }
 Widget getMyWidget(String title) {
   try {
     final f = 1 ~/ 0;
   } catch (exception, stack) {
     print('Reporting Error>>>>>>>>>>>>>>. $exception $stack');
     Sentry.captureException(
       exception,
       stackTrace: stack,
     );
     return Text('Error....');
   }
   return Text('Hello World !');
 }
}
```

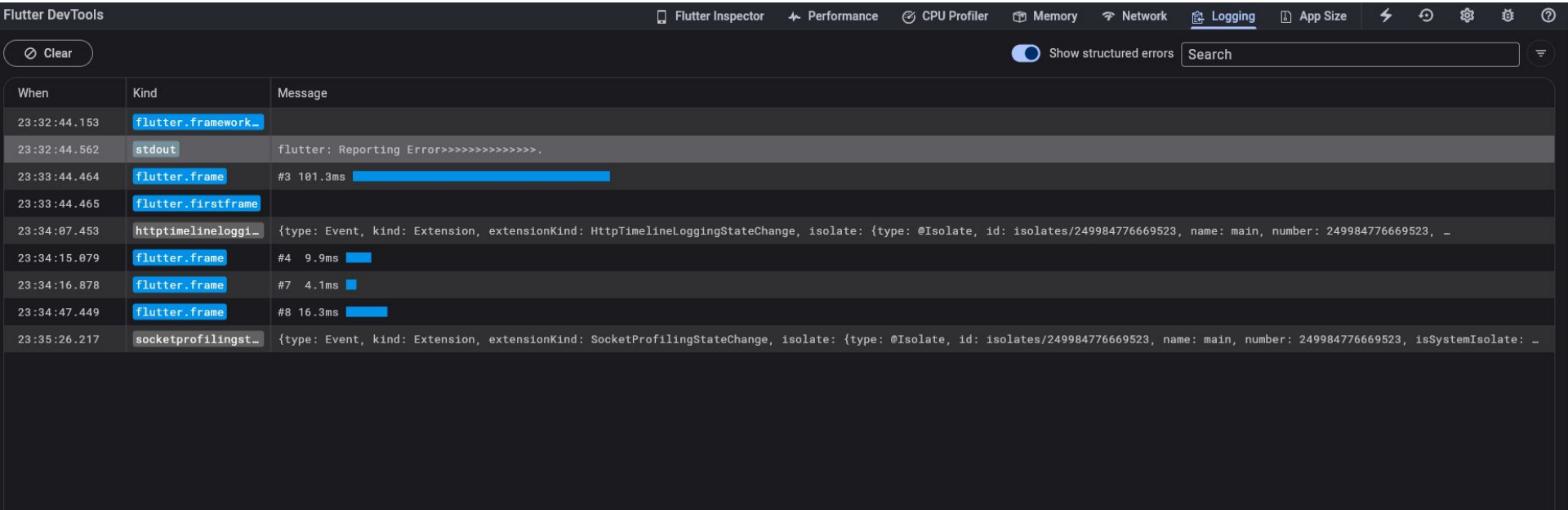

ENSIA
imed@imed.ws
Issues
Projects
Performance
Queries
Web Vitals
Mobile
Resources
Profiling
Replays
User Feedback
Crons
Alerts
Discover
Dashboards
Releases
Stats
Settings
Quick Start
10 Remaining tasks
My Sentry Trial
Help
What's new
Collapse
Issues > FLUTTER-3
All Envs
Open in Discover
Archive
Resolve
IntegerDivisionByZeroException main.dart in MainApp.getMyWidget New
IntegerDivisionByZeroException
EVENTS 1
USERS 1
Details
Activity 0
User Feedback 0
Attachments
Tags
All Events
Merged Issues
Similar Issues
Event ID: 53cbf96d Jan 9, 10:15 PM
Root
This Event
View Full Trace: 4d08b7ba (1 event)
Recommended
Tags
105.97.6.32
ID: aded2ec4-16a2-4ae2-bb6e-6f1a79e43848
Android
Version: 14
SM-S918B
Model: SM-S918B
dart.name Dart
device SM-S918B
device.class high
device.family SM-S918B
dist 1
environment debug
event.environment dart
event.origin flutter
flutter.name Flutter
isSideLoaded true
level error
os Android 14
os.name Android
os.rooted no
release 0.1.0 (1)
user id:aded2ec4-16a2-4ae2-bb6e-6f1a79e43848
Stack Trace
Most Relevant
Full Stack Trace
Newest
IntegerDivisionByZeroException
IntegerDivisionByZeroException
Crashed in non-app: integers.dart in int.~/
main.dart in MainApp.getMyWidget at line 31:19 within report_error_service
In App
main.dart in MainApp.build at line 24:29 within report_error_service
In App
Called from: framework.dart in StatelessElement.build within flutter
Show 43 more frames
Breadcrumbs
Filter By
Search breadcrumbs
Newest
TYPE
CATEGORY
DESCRIPTION
LEVEL
TIME
exception
IntegerDivisionByZeroException: IntegerDivisionByZeroException
Error
22:15:26
app.lifecycle
22:15:26
Assigned To
No one
Last 24 Hours
1
Last 30 Days
1
Last Seen
10 minutes ago in release 0.1.0 (1)
First Seen
10 minutes ago in release 0.1.0 (1)
Releases
See which release caused this issue
Issue Tracking
Track this issue in Jira, GitHub, etc.
All Tags
device
SM-S918B
100%

Exception & Error Handling

- **Logging and Monitoring Errors using DevTools**
  - On the main.dart App
    - You click Run with debugging
    - Open the command palette (Ctrl + shift + p)
      - Type Devtools and choose :
        - Open Flutter DevTools
        - Open in Browser..

Exception & Error Handling

- Logging and Monitoring Errors using DevTools



Section 3

Testing Your Mobile App

Introduction to Testing & Debugging Mobile Apps

- **Testing is :**
  - Defined as the the process of :
    - Examining that the software does what it is supposed to do
    - Uncovering errors and defects during the previous stages of the software development life cycle including requirements, design and implementation.

Introduction to Testing & Debugging Mobile Apps

- **Objectives of Software Testing** :
  - Uncover all software bugs / defects
  - Improve the performance, usability and other quality attributes.
  - Establish a confidence that the software is **fit for use.**

Introduction to Testing & Debugging Mobile Apps

- **Objectives of Software Testing** :
  - The aim of software testing is never to demonstrate that the software has no bugs or defects . As it is impossible.

Introduction to Testing & Debugging Mobile Apps

- **Objectives of Software Testing** :

> **“Testing can only show the presence of errors, not their absence ”**
>
> *Dijkstra, E. W. 1972. “The Humble Programmer.”*

Introduction to Testing & Debugging Mobile Apps

- **Why Software Testing** is **EXTREMELY IMPORTANT**:
  - Reduce Costs in both : Development and Maintenance
  - Customer Satisfaction and Confidence in using your software
  - Reputation for your brand
  - Avoid Disasters, Stressful nights,...etc.

Introduction to Testing & Debugging Mobile Apps

- **Why Software Testing** is **EXTREMELY IMPORTANT**:
  - **How Expensive to Find a bug and fix it after sometime whilst the system is in production ?**

Introduction to Testing & Debugging Mobile Apps

- Terminology :
  - **Software Debugging vs Software Testing**
    - **Software Testing** :
      - Find the errors in the software
      - Done by all people including a random person
    - **Software Debugging** :
      - Is the process to **locate** a defect and resolve it.
      - Done only by Software Developers.
      - Very Difficult and Very Costly

Introduction to Testing & Debugging Mobile Apps

- **Static vs. Dynamic Testing**
  - **Static Testing :**
    - This is conducted by **reviewing and inspecting** the actual source code or design models without executing or running the code
    - Is it really efficient ?
      - Depending on the experience of the inspector and their dedication to find errors.
      - Very tedious process. (Because ?)

Introduction to Testing & Debugging Mobile Apps

- **Static vs. Dynamic Testing**
  - **Static Testing :**
    - This is conducted by **reviewing and inspecting** the actual source code or design models without executing or running the code
    - Is it really efficient ?
      - Depending on the experience of the inspector and their dedication to find errors.
      - Very tedious process.(Because **Reading Code takes more time**)

Introduction to Testing & Debugging Mobile Apps

- **Static vs. Dynamic Testing**
  - **Static Testing :**

urce

e

Fagan (1976) reported that more than 60% of the errors in a program can be detected using informal program inspections.

In the Cleanroom process (Prowell et al. 1999), it is claimed that more than 90% of defects can be discovered in program inspections.

dedication to find errors.

The cleanroom software engineering process is a software development process intended to produce software with a certifiable level of reliability

Introduction to Testing & Debugging Mobile Apps

- **Static vs. Dynamic Testing**
  - **Static Testing :**

urce

e

A lot of AI-based tools are showing up recently to detect code smells and bugs via automated verification of source code.

dedication to find errors.

Introduction to Testing & Debugging Mobile Apps

- **Static vs. Dynamic Testing**
  - **Dynamic Testing :**
    - Inversely to static testing, this is done during the execution of the software (whether in a local development environment, or staging or even a production environment).
    - Testers would interact with the system and its components to find errors during the execution of the software.

Introduction to Testing & Debugging Mobile Apps

- **Passive vs. Active Testing**
  - **Passive Testing :**
    - The software is verified indirectly without any interaction with the actual software.
    - This is done via analysing the software logs and traces
    - Extremely useful when software is actually deployed and you need to know what's going on. (A user cannot log in ? why)
  - **Active Testing :**
    - Testers interact with the software product directly to examine its functionality and quality attributes.

Introduction to Testing & Debugging Mobile Apps

- **Positive vs. Negative Testing**
  - **Positive Testing :**
    - The software system is validated against the valid input data
    - The main intention of this testing is to check whether a software application does what it is supposed to do based on valid paths/user scenarios
  - **Negative Testing :**
    - Commonly referred to as error path testing or failure testing
    - Its aim is to ensure the stability of the application.
    - Done by applying as much creativity as possible and validating the application against invalid data as input : Example, Age from birth, input : **02/29/2021**

Introduction to Testing & Debugging Mobile Apps

- **Black Box vs. White Box Testing**
  - **Black Box Testing :**
    - The tester would treat the software as a black box where the functionalities of the system are tested without any knowledge of how they are implemented.
    - Testers need to know the test case in addition to the input and expected output or behaviour. No need to have any programming experience.
    - Occasionally called as functional testing as it is mostly used for the functional requirements (but can be also for non-functional reqs)

Introduction to Testing & Debugging Mobile Apps

- **Black Box vs. White Box Testing**
  - **White Box Testing :**
    - Conducted by a software developer where they can see and test internal structure of the code.
    - The tester needs to have a good understanding of the code so that they can examine different code paths and scenarios using tests.
    - White box tests are mainly used to detect logical errors in the program code.
    - Called also : Clear-box testing, structural testing, code-based testing or Glass box testing

Introduction to Testing & Debugging Mobile Apps

- **Functional vs. Non-Functional Testing**
  - **Functional Testing :**
    - The tester would focus on testing the primary use cases of the software product or user stories of a specific function of the code
    - Functional tests would answer the questions :
      - Can the user do this ?
      - Does this particular feature work ?
    - For instance: *Registration ? Login ? Posting , editing and completing a to-do item ? Resetting a forgotten password ? Changing the email address ?*

Introduction to Testing & Debugging Mobile Apps

- **Functional vs. Non-Functional Testing**
  - **Non-Functional Testing :**
    - It is related to the quality attributes, **how well** does the software perform its functionalities in terms of : performance ? reliability ? availability ? fault tolerance ? usability ? Security ? ….

Introduction to Testing & Debugging Mobile Apps

- **Functional vs. Non-Functional Testing**
  - **Non-Functional Testing :**
    - Usability Testing : Is it easy to use : SUS can be used.
    - Security Testing : Authorization ? Authentication ? Confidentiality?
    - Stress Testing : Testing how much strain the system can take before it fails. Considered to be a type of non-functional testing.
    - Destructive Testing
    - Performance Testing
    - Acceptance Testing
    - Accessibility Testing
    - Localization Testing
    - Concurrent Testing
    - ...

Introduction to Testing & Debugging Mobile Apps

- **Regression Testing**
  - Is to **re-run all test** cases including functional and non-functional to ensure that the software (which was previously tested) still performs as expected after a change. The change includes:
    - Fixing a bug
    - Adding new functionality
    - Configuration change
    - Substitution of hardware

Introduction to Testing & Debugging Mobile Apps

- **Manual vs. Automated Testing**
  - **Manual Testing :**
    - A human tester would go manually to test or inspect the software.
    - The main drawback of manual testing is the need to retest and redo all tedious tasks again in case of a new functionality or new release

Introduction to Testing & Debugging Mobile Apps

- **Manual vs. Automated Testing**
  - **Automated Testing :**
    - Testing code/scripts are written using/for Testing automation Tools or frameworks to ensure that the software at different levels behaves correctly.
    - Test Data needs to be prepared to run on the test scripts which simulate the execution of the software product
    - It can help to automate some or most repetitive tasks in a formalized test process.
    - For larger projects, it is becoming **a must**

Introduction to Testing & Debugging Mobile Apps

- **Manual vs. Automated Testing**
  - **Automated Testing :**
    - It is vital for Continuous delivery and Integration systems where there is a frequent release of the software product.
    - Fast and Efficient
    - Improved Test Coverage
    - Re-usability of the testing automation scripts.
    - Cost and Time savings.

Introduction to Testing & Debugging Mobile Apps

- **Automated Testing**
  - **Test-Driven Development :**
    - Test scripts are written before they implement new code or even change existing code.
      - Forces the Developers or Test Engineers to achieve a better understanding of software requirements
    - Developers write code for testability.
    - It limits the scope of development.
    - Every feature or function written is tested.

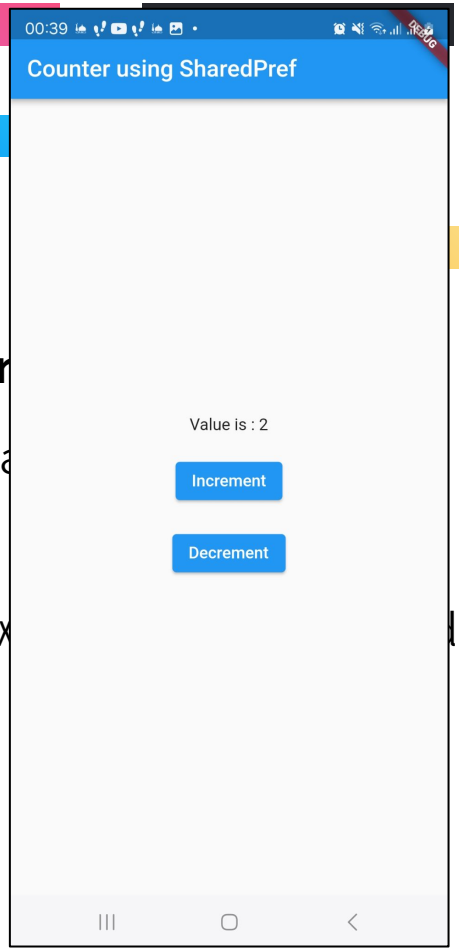
Introduction to Testing & Debugging Mobile Apps

- Different Types of Testing based on the level or phase
  - Unit
  - Component/Service (Widget Testing)
  - Integration
  - Release/Deployment
  - Post-Release
  - Acceptance Test

Unit Testing for Mobile Apps

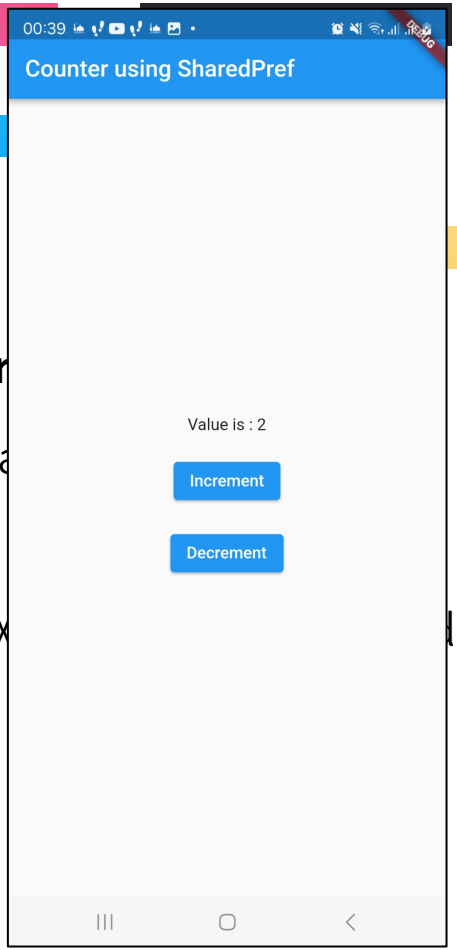
- **Unit Testing : Case Study for the Increment/Decrement App**
  - Unit tests are **very useful** for verifying the behavior and business logic inside a single function, method, or class.
  - The test package provides the core framework for writing unit tests, and the flutter_test package provides additional utilities for testing widgets.
    - **flutter pub add dev:test**

Unit Testing for Mobile Apps

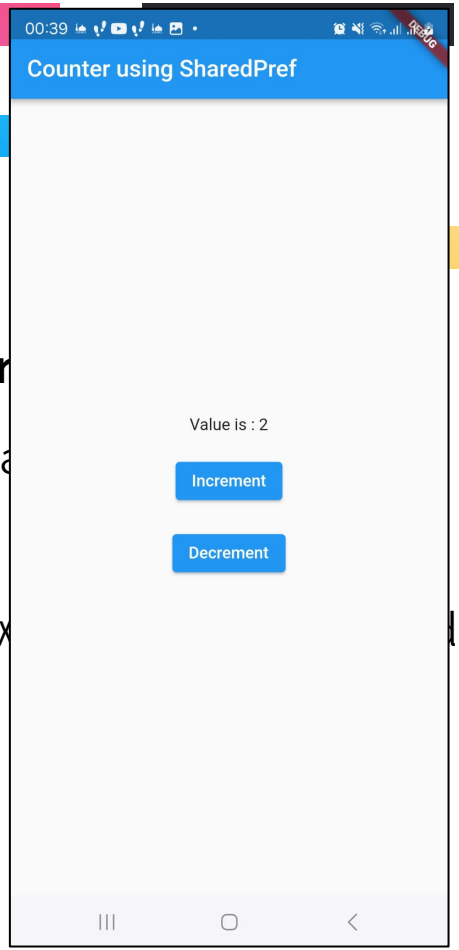


- **Unit Testing : Case Study for the Increment/Decremer**
  - Unit tests are **very useful** for verifying the behavior a
    inside a single function, method, or class.
  - The test package provides the core framework for w
    the flutter_test package provides additional utilities
    - **flutter pub add dev:test**

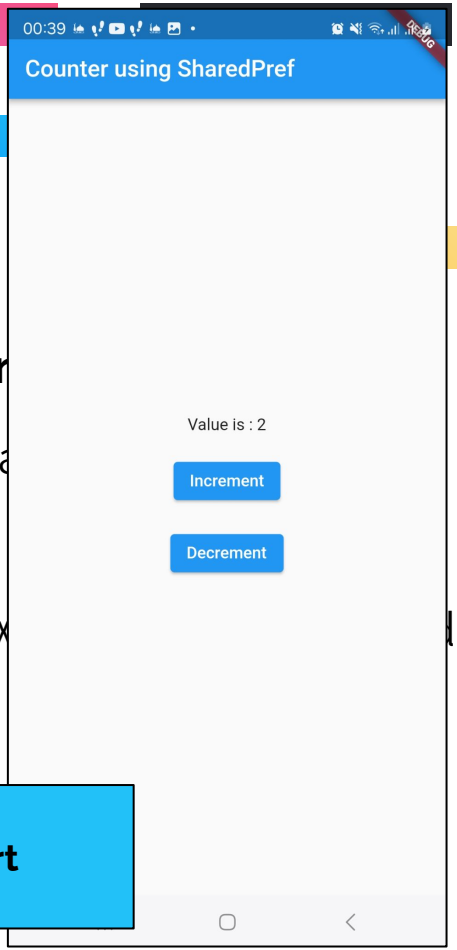
```dart
import 'package:shared_preferences/shared_preferences.dart';

class Counter {
 late SharedPreferences prefs;

 init() async {
   prefs = await SharedPreferences.getInstance();
 }

 int getValue() {
   var increment = prefs.getInt("incrementNumber") ?? 0;
   return increment;
 }

 Future<bool> increment() async {
   var increment = await prefs.getInt("incrementNumber") ?? 0;
   increment = increment + 1;
   await prefs.setInt("incrementNumber", increment);
   return true;
 }

 Future<bool> decrement() async {
   var increment = await prefs.getInt("incrementNumber") ?? 0;
   increment = increment - 1;
   await prefs.setInt("incrementNumber", increment);
   return true;
 }

 Future<bool> reset() async {
   await prefs.setInt("incrementNumber", 0);
   return true;
 }
}
```



```dart
class  HomeScreenState extends State<HomeScreen> {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
       appBar: AppBar(title: Text("Counter using SharedPref ")),
       body: Center(
           child: Column(
         mainAxisAlignment: MainAxisAlignment.center,
         crossAxisAlignment: CrossAxisAlignment.center,
         children: [
           Text("Value is : ${counter.getValue()}"),
           SizedBox(height: 20),
           ElevatedButton(
               onPressed: () async {
                 counter.increment().then(
                   (value) {setState(() {});},);
               },
               child: Text("Increment")),
           SizedBox(height: 20),
           ElevatedButton(
               onPressed: () async {
                 counter.decrement().then(
                   (value) {setState(() {});},);

               },
               child: Text("Decrement")),
         ],
       )));
 }
}
```

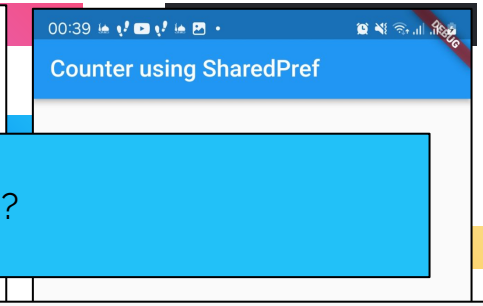


```dart
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:flutter_unit_testing/main.dart';
import 'package:flutter_unit_testing/utils/counter.dart';
import 'package:shared_preferences/shared_preferences.dart';

void main() {
 test('Counter value should be incremented', () async {
   WidgetsFlutterBinding.ensureInitialized();
   SharedPreferences.setMockInitialValues({});
   final counter = Counter();
   counter.init();
   counter.reset();
   counter.increment();

   expect(counter.getValue(), 1);
 });
}
```



To run the tests : **flutter test test/unit_test_counter.dart**

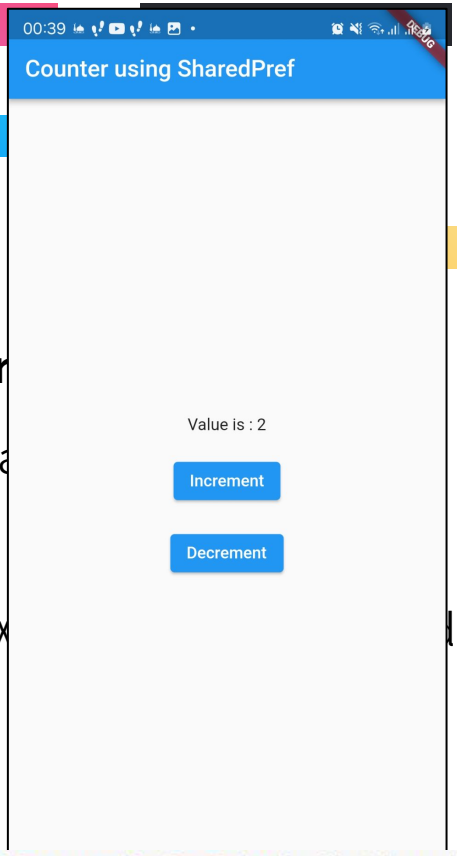
```dart
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:flutter_unit_testing/main.dart';
import 'package:flutter_unit_tes
import 'package:shared_preferenc

void main() {
 test('Counter value should be
   WidgetsFlutterBinding.ensure
   SharedPreferences.setMockIni
   final counter = Counter();
   counter.init();
   counter.reset();
   counter.increment();

   expect(counter.getValue(), 1
 });
}
```



Why the test is failing ?

```
00:00 +0 -1: Counter value should be incremented [E]
  LateInitializationError: Field 'prefs' has not been initialized.
  package:flutter_unit_testing/utils/counter.dart        Counter.prefs
  package:flutter_unit_testing/utils/counter.dart 30:11  Counter.reset
  test/unit_test_counter.dart 13:13                      main.<fn>
  ===== asynchronous gap ===========================
  dart:async                                             Future._asyncCompleteError
  test/unit_test_counter.dart 13:13                      main.<fn>

  LateInitializationError: Field 'prefs' has not been initialized.
  package:flutter_unit_testing/utils/counter.dart        Counter.prefs
  package:flutter_unit_testing/utils/counter.dart 16:27  Counter.increment
  test/unit_test_counter.dart 14:13                      main.<fn>
  ===== asynchronous gap ===========================
  dart:async                                             Future._asyncCompleteError
  test/unit_test_counter.dart 14:13                      main.<fn>

  LateInitializationError: Field 'prefs' has not been initialized.
  package:flutter_unit_testing/utils/counter.dart        Counter.prefs
  package:flutter_unit_testing/utils/counter.dart 11:21  Counter.getValue
  test/unit_test_counter.dart 16:20                      main.<fn>


To run this test again: /home/imed/snap/flutter/common/flutter/bin/cache/dart-sdk/bin/dart test /home/imed/Dropbox/Wo
rkspace/Teaching/ENSIA/MobileDev/MyLectures/W14/flutter_unit_testing/test/unit_test_counter.dart -p vm --plain-name '
Counter value should be incremented'
00:01 +0 -1: Some tests failed.
imed@imed-Inspiron:~/Dropbox/Workspace/Teaching/ENSIA/MobileDev/MyLectures/W14/flutter_unit_testing$
```

```dart
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:flutter_unit_testing/main.dart';
import 'package:flutter_unit_testing/utils/counter.dart';
import 'package:shared_preferences/shared_preferences.dart';

void main() {
 test('Counter value should be incremented', () async {
   WidgetsFlutterBinding.ensureInitialized();
   SharedPreferences.setMockInitialValues({});
   final counter = Counter();
   await counter.init();
   await counter.reset();
   await counter.increment();

   expect(counter.getValue(), 1);
 });
}
```

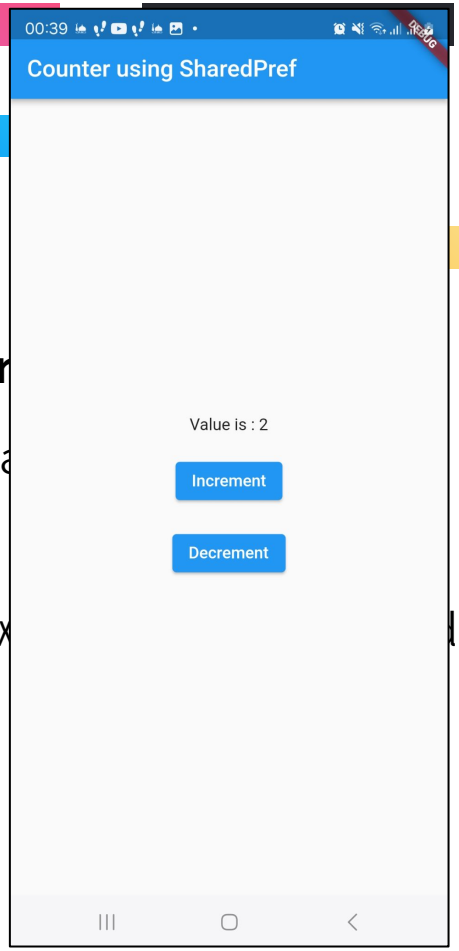



```
imed@imed-Inspiron:~/Dropbox/Workspace/Teaching/ENSIA/MobileDev/MyLectures/W14/flutter_unit_testing$ flutter test
t/unit_test_counter.dart
00:00 +1: All tests passed!
imed@imed-Inspiron:~/Dropbox/Workspace/Teaching/ENSIA/MobileDev/MyLectures/W14/flutter_unit_testing$
```

Widget Testing for Mobile Apps

- **Widget Testing for Flutter Apps**
  - It will allow you to create a simulated UI (invisible in the background) where you can simulate user interaction (entering text, tapping, sliding..) and search for results.

```dart
import 'package:flutter/material.dart';
import 'package:flutter test/flutter test.dart';
import 'package:flutter_unit_testing/screens/home.dart';
import 'package:flutter unit testing/utils/counter.dart';
import 'package:shared_preferences/shared_preferences.dart';

void main() {
 testWidgets('Clicking Increment button', (tester) async {
   WidgetsFlutterBinding.ensureInitialized();

   SharedPreferences.setMockInitialValues({});
   await counter.init();
   await counter.reset();

   // Build the widget.
   await tester.pumpWidget(MaterialApp(
     home: HomeScreen(),
   ));

   expect(find.text("Increment"), findsOneWidget);
   await tester.tap(find.text("Increment"));
   await tester.tap(find.text("Increment"));
   await tester.pump();
   expect(find.textContaining("Value is : 3"), findsOneWidget);
 });
}
```



UI/UX Testing for Mobile Apps

- **Usability Testing**
  - It is aimed to assess how easy and satisfied for a target user to perform a specific task using the mobile app.
  - Easy & Satisfied :
    - Short time
    - Effortless
    - Satisfied with the experience
    - No need for training and assistance
    - Can easily remember the instructions

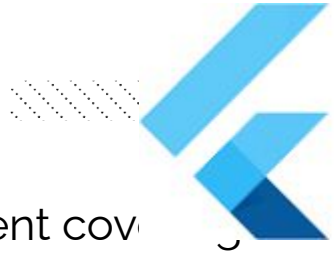
UI/UX Testing for Mobile Apps

- **Ways of Conducting Usability Testing**
  - Either :
    - Inviting real end-users to experiment the application
    - Hiring Usability Consultants
    - Automated Usability Testing
      - Still not yet mature yet and can be very expensive
    - Hybrid

UI/UX Testing for Mobile Apps

- **When to Conduct Usability Testing**
  - Primarily before the development using a prototype/MVP
  - But must be also done after completing the App for better improvment

UI/UX Testing for Mobile Apps



- **Usability Testing Guidelines**
  - A plan must be established detailing the testing experiment cov
    1. Types and numbers of users to be invited.
    2. Type of the experiment testing (Remote or inhouse)
    3. Define clearly the Tasks to be performed by the invited user (As atomic as possible)
       - Signup
       - Login
       - Create a Product
       - Take Picture
       - Publish Post
    4. Perform this tasks yourself or by an Expert user and estimate the usage metrics : duration for a task, number of interactions (taps,..)

UI/UX Testing for Mobile Apps

- **Usability Testing Guidelines**
  - A plan must be established detailing the testing experiment covering:
    5. Setup infrastructure needed to record the experiments whether for the remote or inhouse testing. This includes :
       - Screencast of the user-app interaction
       - If possible, record the face of the user
       - Record all events (Tapping, Typing…)

    6. A questionnaire must be constructed to be given to the user at the end of the experiment to express their subjective opinions about the utility and difficulty of the tasks given.

UI/UX Testing for Mobile Apps

- **Usability Testing Guidelines**
  - It is highly recommended that you sit next to the user and observe them without giving them any assistance and monitor their interaction for possible ways to improve the usability aspect
  - From the data you collect, analyse where you can possible infer usability issues and seek alternative designs to improve the usability.

Upgrade Testing for Mobile Apps

- **Releasing newer versions**
  - It is extremely important to conduct tests on simulation environments with existing data to ensure that :
    - The new features are compatible with the existing settings.
    - Customer Data will stay consistent and intact.

Lecture Demo Apps

- MVP for Posting an Article with Images
  - https://www.dropbox.com/scl/fo/adt9l7v1nsmil6xjlkssw/h?rlkey=plfl8byh4138a1xbk8mxpdjoy&dl=0
- Simple Implementation for the MVP using S3 Storage/Firebase Storage
  - https://www.dropbox.com/scl/fo/dep95qblyxylozxo2qk1x/h?rlkey=6v6bn06egr6vdohgby28ooi5k&dl=0
- Backend using Flask/Python
  - https://www.dropbox.com/scl/fo/difm10rg2jgc071x8k9fz/h?rlkey=rcoze03xxnzu5ds2ucig9gujk&dl=0
- Error Reporting using Sentry.io
  - https://www.dropbox.com/scl/fo/34ym04gqe6lq3liey21l8/h?rlkey=j4wxotajsipexi4bsxxp19p0k&dl=0
- Unit Testing in Flutter:
  - https://www.dropbox.com/scl/fo/a9q3ps4lt59394zku0lbu/h?rlkey=if5ajpraqwtscz9sc4todu8v2&dl=0