CS4039
SOFTWARE FOR
MOBILE DEVICES

SERIALIZATION AND DESERIALIZATION IN FLUTTER

ADD DEPENDENCIES USING PUBSPEC.YAML

📌Role of pubspec.yaml

- Configuration file for every Flutter project.
- Manages project metadata, dependencies, and assets.
- Acts as a central place to declare everything your app needs.

📌Types of Dependencies

1. **SDK Dependencies → Core Flutter/Dart libraries.**
   - **Example: sdk: flutter**
2. **Package Dependencies → Third-party packages from pub.dev.**
   - **Example: http, provider, etc.**
3. **Git Dependencies → Package hosted on Git repositories.**
   - **Example: git: https://github.com/user/repo.git**
4. **Path Dependencies → Local packages from your system.**
   - **Example: path: ../my_package**

📌Add Package in pubspec.yaml

```yaml
dependencies:
  flutter:
    sdk: flutter
  http: ^1.2.0
```

📌Add Image Assets in Flutter

- Place your images inside assets/ folder.
- Declare in pubspec.yaml:

```yaml
flutter:
  assets:
    - assets/images/logo.png
    - assets/icons/
```

HTTP PACKAGE

- Fetching/posting data from the internet is necessary for most apps.
- Luckily, Dart and Flutter provide tools, such as the http package, for this type of work.
- Add package in pubspec.yaml.

Android apps must declare their use of the internet in the Android manifest (AndroidManifest.xml)

PARSE JSON AS DART OBJECT

- JSON Encode and Decode Functions http packages.
- Simply converts string response in to key map.
- Set those value to your dart object field by creating your own methods.

JSON SERIALIZATION TYPES

- Manual serialization
- Automated serialization using code generation

MANUAL SERIALIZATION

- Manual JSON decoding refers to using the built-in JSON decoder in dart:convert
- Serializing JSON inline

```
Map<String, dynamic> user = jsonDecode(jsonString);
print('Howdy, ${user['name']}!');
```

- Serializing JSON inside model classes

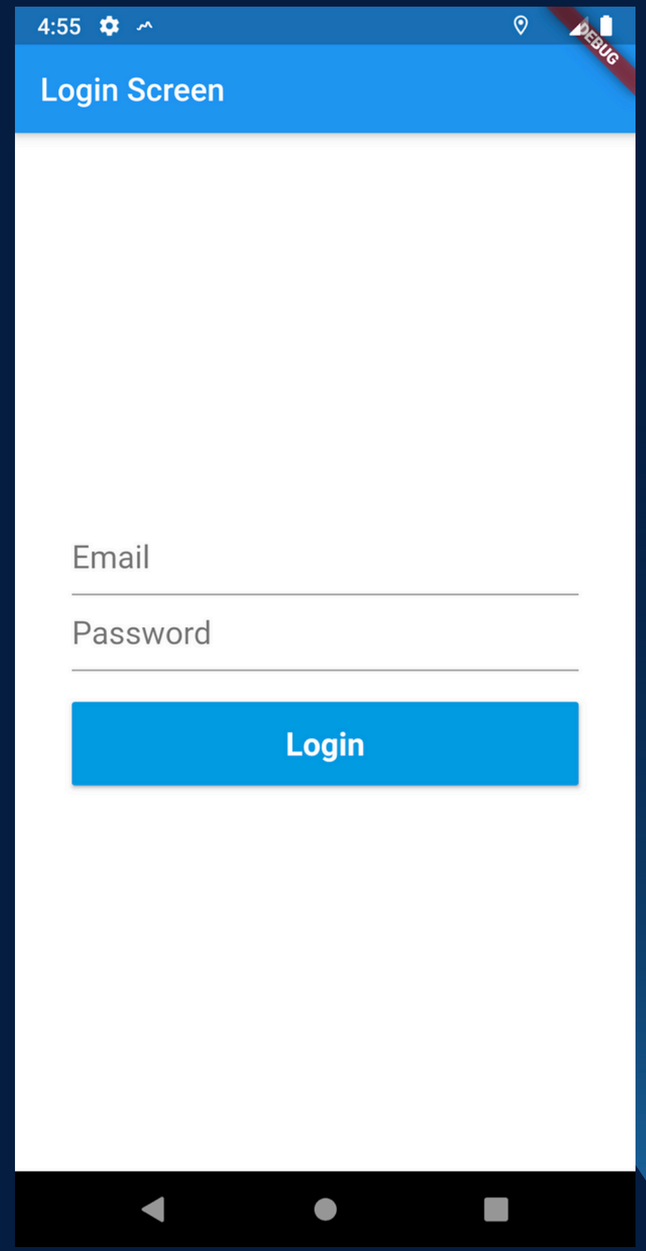
A User.fromJson() constructor, for constructing a new User instance from a map structure.
A toJson() method, which converts a User instance into a map.

MANUAL SERIALIZATION

```dart
class Movie {
  String title;
  String imageUrl;
  String releaseDate;

  Movie.fromJson(Map<String, dynamic> json)
      : title = json['title'],
        imageUrl = json['poster_path'],
        releaseDate = json['release_date'];
}
```

POSTING DATA ON INTERNET

- Post API Implementation
- Login Flow



SERIALIZING JSON USING CODE GENERATION LIBRARIES

- Add Json Serializable package.
- Add Build runner
- Add annotation on your class
- Add factory methods
- Execute command on your terminal “**flutter pub run build_runner build**”

SERIALIZING JSON USING CODE GENERATION SAMPLE

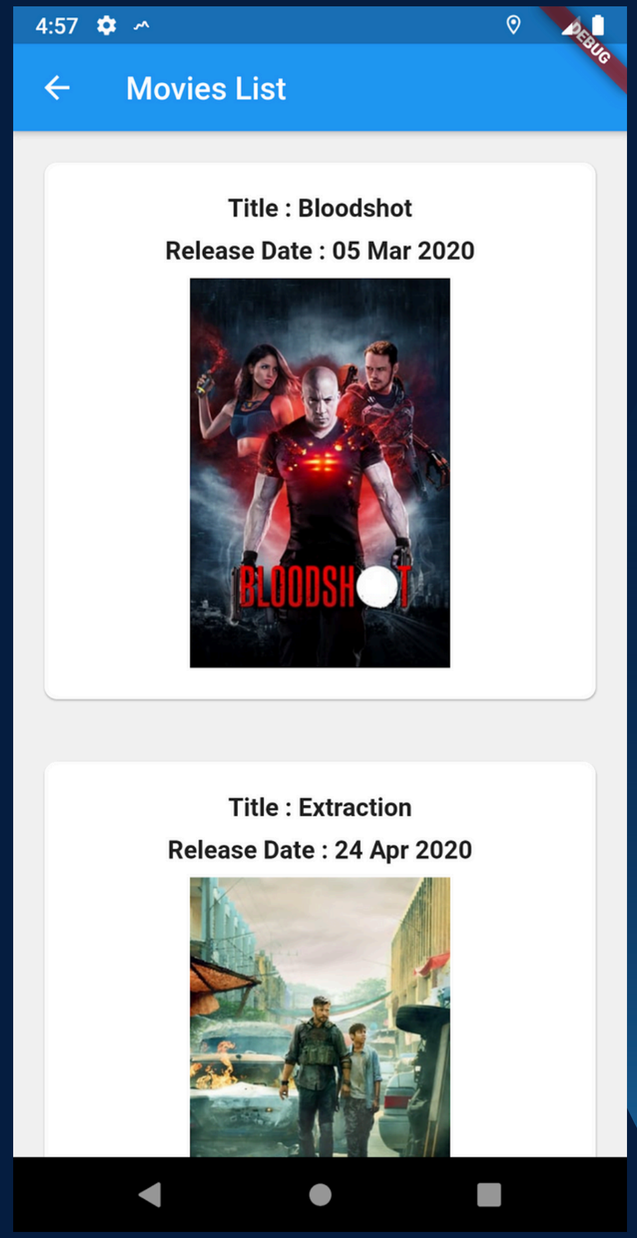
```dart
part 'movie.g.dart';

@JsonSerializable()
class Movie {
  String title;
  @JsonKey(name: 'poster_path')
  String imageUrl;
  @JsonKey(name: 'release_date')
  String releaseDate;

  Movie(this.title, this.imageUrl, this.releaseDate);
  factory Movie.fromJson(Map<String, dynamic> json) => _$MovieFromJson(json);
```

JSON SERIALIZABLE SUPPORT

- @JsonKey(defaultValue: false)
- @JsonKey(required: true)
- @JsonKey(ignore: true)
- @JsonKey(name: ‘keyName’)

FETCHING DATA FROM INTERNET

- GEt API Implementation
- Movie Listing Flow



JSON PARSE IN BACKGROUND

- If this work takes more than 16 milliseconds, your users experience jank
- Isolate
- Thread based concept in dart
- compute() : function runs expensive functions in a background isolate and returns the result