RoadMovie

Authors: Margarita Mayer, Polina Romanenkova

What is it?

RoadMovie is an application that allows people who are in Moscow to explore the filming locations of Moscow films.

Project glossary:

In the glossary you can find the working definitions for the RoadMovie application.

Pinch and stretch - a "compressing" and "stretching" the movement of the fingers, allowing, respectively, to reduce or increase the scale of the map.

Map viewing - zooming in and out of the map.
Pen - a movement on the screen of the enlarged image, allowing you to see the parts hidden behind the edge of the screen.
Scrolling - view relevant information by scrolling down the ribbon.
Help about the movie - a relevant information that appears when you click on the icon.
Double-tap - touching the screen twice in a row in the same place.
Biopic - a biographical film.
Advice on taking photos (ADOT) - some tips for those who want to take pictures or shoot exactly like in the movie - camera angle, weather, lens, time of day when the shot will come out the most similar.
Way of reaching POI/route - the way the user can visit points on the route - information on whether he will need a car or public transport, how long the route will take.
Route - the geographical locations in which/where certain scenes of the film were filmed.
POI (point of interest) - a landmark or other object marked with a point on a map.
Scene importance - a percentage of the importance of a scene shot on a particular POI for the film. The percentage of importance is calculated according to the length of the scene, its role in creating the plot, its recognizability, and the number of main characters involved in it.
Notes on a scene - certain important information that characterizes a particular scene - the weather in which the scene was filmed, the music that was played, how many takes were done.
FiLo (or filo) - film location
Android Studio - an integrated development environment (IDE) for working with the Android platform.
Flutter - a development kit and open-source framework for building Android mobile apps.
Dart - the programming language used to code Flutter apps.

Stakeholders:

1. Developers
   Responsibilities:
   - Developing an app
2. Maintainers
   Responsibilities:
   - Test the system, check it for errors
   - Fix mistakes, keep the system working
3. Users

Responsibilities:
- Use the app
- Give feedback if they would like to

User stories:

User Type	User Story Title	User stories
Application User	Questionnaire	1.1. As a user, I want to know which movies were made in Moscow. 1.2. As a user, I want to know what part of Moscow the film I choose was shot in. 1.3. As a user, I want to distinguish movie scenes by priority so that I can skip less important places if I want to.
	Graphic Design	2.1. As a user, I want the design to be minimalistic and intuitive so that I can quickly learn how to use the app.
	Registration	3.1. As a user, I don't want to sign up for the app to save my time and not give away my personal information.
	Walking	4.1. As a user, I want to see all the locations marked on the map where a movie was shot. 4.2. As a user I want to have a route of walking along all the places of a particular movie. 4.3. As a user, I want to see my location marked on the map.
	Discovering	5.1. As a user, I want to select the movie I want to explore. 5.2. As a user, I want to see a shot from the movie and a description of it when I click on the POI so I know the information about the shot. 5.3. As a user I want to quit the route at any time so that I can take a different route. 5.4. As a user I want to discover new places in the center of Moscow. 5.5. As a user I want to see the history of my movie discoveries so that I can analyze my walks and make them better.
	Learning	6.1. As a user I want to get additional information about the movie so that I can learn more about things that interest me. 6.2. As a user I want to see recommendations of routes with ratings so that I can choose the most convenient option for myself. 6.3. As a user I want to learn additional information about the process of filming in particular places.
	Free using	7.1. As a user I want to use the app without any payments.

Non-Functional Requirements:

Category	Explanation	How will we achieve it?
Usability	The app has to have a convenient, minimalistic and intuitive interface so that the user can quickly and effectively learn how to use the app.	We will poll users for their opinion regarding the interface and collect statistics. At least 60% of beta users must anonymously confirm that the RoadMovie app is convenient enough.
Speed	The user expects the app to respond to his taps very quickly.	We will make an app that reacts after tapping on the screen after a maximum of 2 seconds.
Speed	The user wants the application to start as quickly as possible after clicking on it.	The application should start with a latency of no greater than 5 seconds.
Capacity	The app should be small because users don't want to take up much space on their phone.	Maximum application size should be 15 MB.
Availability	The app should contain a list of films that will be of interest to users.	We will poll users for their opinion regarding the interface and collect statistics. At least 80% of beta users must anonymously confirm that the movie list is interesting enough and they like it.

Flutter framework:

We chose to use the Flutter framework in our project for a number of reasons:

1. Flutter contains its own graphics engine. Thus, there is no need to make the interface separate for Android and iOS.
2. The interface is easily broken down into separate modules. This allows you to apply the Single responsibility principle.
3. Availability of convenient packages to work with GoogleMaps. For example, google_maps_flutter.

How we implemented the graphical design of the app:

We created a custom icon for the app, as well as we made custom POI-markers on the map, we added custom fonts and evaluated a special colour pattern. A minimalistic design was applied for all parts of the app. And below you can see our brief of graphical design.

ROAD MOVIE

FUTURA BOOK C REGULAR

FFDA00	FFFFFF	A7CCDB	000000

ROAD MOVIE FUTURA DEMI C REGULAR

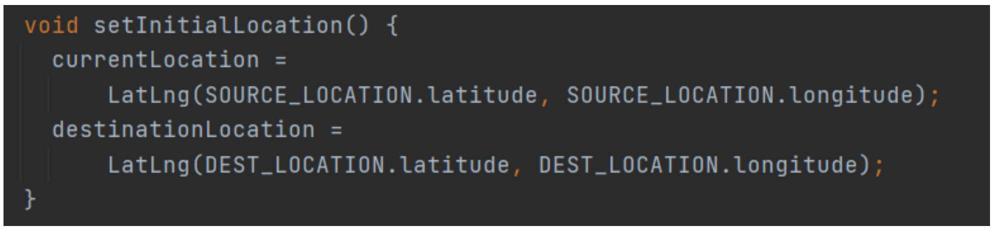
SOLID principles:

Single responsibility principle

When we wrote the code, we tried to make sure that each component had only one responsibility. The file system was divided into specific blocks, each of which performs a different function. For example, mappage.dart is the file responsible for the map. As a result, it was easier to understand exactly what each piece of code does.

Open–closed principle

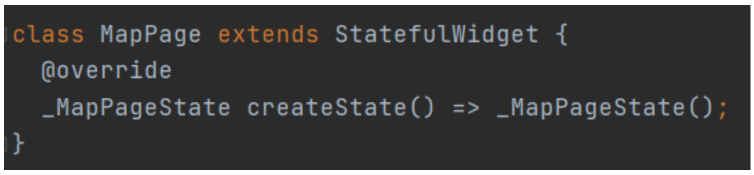
During the development of our project, we used the Agile philosophy. To make the code easy to use in the next sprints, we tried to use the Open/Closed principle from the very beginning when modifying it. Thus, in each new sprint, we practically didn't rewrite all the code, but simply added new classes and methods, thereby increasing the application's capabilities.



```
void setInitialLocation() {
  currentLocation =
      LatLng(SOURCE_LOCATION.latitude, SOURCE_LOCATION.longitude);
  destinationLocation =
      LatLng(DEST_LOCATION.latitude, DEST_LOCATION.longitude);
}
```

The task of the method is to set the initial location. It is available for extension, but closed for modification.

Liskov substitution principle
We applied the Liskov substitution principle only where it was appropriate. We used inheritance when our superclass was replaceable by a subclass in all the instances. Example usage:



```
class MapPage extends StatefulWidget {
  @override
  _MapPageState createState() => _MapPageState();
}
```

Interface Segregation Principle
Due to the size of our project the implementation of a such principle is not necessary as the application is small. We don't have many classes, each of which needs its own interface.

Dependency inversion principle
In our project, it was inappropriate to divide the modules into upper-level and lower-level modules because our project is not so big.

Design Patterns:

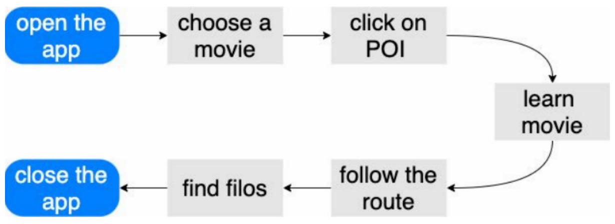
Creative pattern
This pattern helped to create objects by hiding the creation logic. As a result, it was easier for us to understand which objects should be created for each case. For example, the SelectedMoviePage class is inherited from the StatelessWidget class. Thus, the SelectedMoviePage hides some features of the StatelessWidget.

Behavior pattern
Different components of the code affect each other in different ways. We have tried to pay attention to this and to regulate the interaction of classes and methods. For example, the Utils class method returns a list of objects of the Category class.

What does the project have in store for the future?

1. The user's location is marked on a map.
2. A route is created for multiple locations.
3. The user sees recommendations of routes with ratings and chooses the most suitable one.
4. More movies are available in the app.
5. The application works in other cities (not only in Moscow).
6. Users can get acquainted, unite and go exploring movie locations together.
7. We can collaborate with Kinopoisk or GoogleMaps. So, for example, in our app a user can go to Kinopoisk to watch a movie.

Task Flow Diagram:



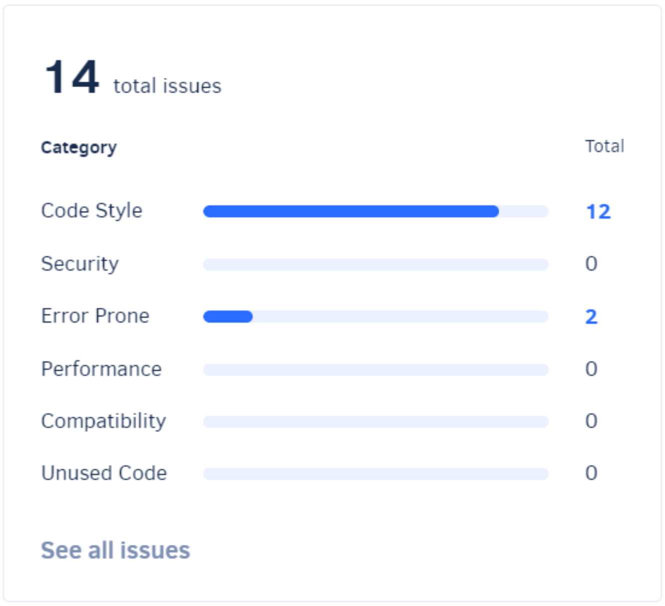
Version History			
Editor's Name	**Date** (DD/MM/YYYY)	**Reason for Changes/Sections Updated**	**Version**
Polina Romanenkova	03.09.2021	Initial version of the document	1.0.
Margarita Mayer	04.09.2021	Design architectural views	1.1.
Margarita Mayer, Polina Romanenkova	05.09.2021	Added information about Non-Functional Requirements	1.2.
Polina Romanenkova	11.09.2021	Created task flow diagram	2.0.
Margarita Mayer, Polina Romanenkova	14.09.2021	Created a list of movies	2.1.
Margarita Mayer	19.09.2021	Created demo of the project - android app, where a user can see the map and clickable points on the map	2.2.
Margarita Mayer, Polina Romanenkova	26.09.2021	Changed the name of the app to "RoadMovie"	2.2.1.
Margarita Mayer, Polina Romanenkova	29.09.2021	Found information about each location. We chose colors for the app and considered different interface designs.	2.2.2.
Margarita Mayer, Polina Romanenkova	01.10.2021	We chose the framework flutter (with dart language). The app had to be completely rewritten.	2.3.0
Polina Romanenkova	02.10.2021	Created an opening welcome page and a splashing page.	2.3.1.
Polina Romanenkova, Margarita Mayer	04.10.2021	Decided on the overlook of a movie selection page, created a movie selection page and a widget for a movie choosing button.	2.3.2.
Polina Romanenkova	04.10.2021	Started developing a map page,	2.3.3.
Polina Romanenkova	05.10.2021	Added the function to create custom clickable pins on the map. Started	2.3.4.

		developing a widget for POI, which is opened by clicking particular pins.	
Polina Romanenkova	06.10.2021	Added information on the project, updated the glossary and System Architecture descriptions.	2.3.5
Polina Romanenkova, Margarita Mayer	08.10.10	Improved the file system, got rid of redundancy, changed the mechanism of putting points to be dependent on a chosen movie.	2.3.6
Polina Romanenkova, Margarita Mayer	10.10.2021	Finished the code and updated the description	3.1.
Polina Romanenkova, Margarita Mayer	10.10.2021	Checked that the application works successfully, the project is well designed, created a demo.	4.1.

GitHub Repository: https://github.com/MargaritaMayer/movee

Codacy:

In our project we use Codacy to check whether we have errors or not. The results is here:


14 total issues

Category	Total
Code Style	12
Security	0
Error Prone	2
Performance	0
Compatibility	0
Unused Code	0