VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
INFORMATION TECHNOLOGIES STUDY PROGRAM

Problem-Based Project

Leisure Map Application

Done by:
Arminas Attas
Dominykas Baronas
Martynas Muižys

Supervisor:
dr. Agnė Brilingaitė

Vilnius
2024

Preface

This project was done during the 4th semester of the study programme *Information Technologies* in the specialization of Innovative studies. We chose the topic *Blockchain multiplayer game* suggested during the project market on the first lecture of the subject ''Problem-Based Project''. The topic of blockchain gaming seemed relevant to us as we knew there is a growing interest in blockchain technology and its potential uses. People are looking for new and exciting ways to engage with blockchain, and gaming is a natural fit. We expected to learn more about the data integration of blockchain technology and algorithms to create a unique and enjoyable multiplayer gaming experience.

January 3, 2024

Arminas Attas

Dominykas Baronas

Martynas Muižys

Contents

Abstract

The aim of this project was to develop an application for Android devices that would ensure that the most suitable leisure places and activities would be proposed based on the user's selected preferences. Essentially, there is an implemented filtering feature that can be applied to find the most appropriate places and significantly facilitate the decision while choosing where to spend the free time. A filtering feature will include properties such as rating, place type and the distance to POI based on the user's current location or the city where the place is located. Moreover, the application will analyze the user's data during the session while using collaborative filtering. Pearson Correlation Coefficient (PCC) will be used to find the most similar user and depending on that recommendations will be offered. Additionally, there will be features added such as GPS, navigation and the possibility to save favorite places. Other useful functionalities as displaying weather forecasts of POI and the ability to use a search box to find desired objects manually will be enabled. With the support of a web server that manages the database, makes API calls and executes algorithmic calculations, the security and performance of the application are significantly increased.

Keywords: Web server [1], Android application [2], Recommendation algorithm [3], POI recommendation [4], Collaborative filtering [5]

Santrauka

"Maps.Relax" laisvalaikio mobili programėlė

Šio projekto tikslas buvo sukurti programėlę Android išmaniems įrenginiams. Programėlės paskirtis - užtikrinti kad vartotojui būtų rekomenduotos tinkamiausios vietos laisvalaikio praleidimui pagal pasirinktus kriterijus ir turimą objektų informaciją, kaip įvertinimas ir atstumas iki vietos, ženkliai palengvinant naudotojo apsisprendimą kurią vietą būtų geriausia pasirinkti."Maps.Relax" taip pat siūlo funkcionalumus kaip objektų filtravimas, žemėlapis, rekomendacijos pagal mūsų sukurtą objektų įvertinimo sistemą, rekomendacijos pagal panašių vartotojų paieškas ir mėgstamiausių vietų išsaugojimas. Taipogi, mūsų programėlėje yra įrašyta navigacijos sistema, kad vartotojai galėtų rasti greičiausią ir optimaliausią kelią iki pasirinktos vietos. "Maps.Relax" suteikia galimybę peržvelgti savaitinę miestų orų prognozę bei papildomai gali pasiūlyti išskirtinių renginių vykstančių tik Vilniuje sąrašą. Objektų filtravimas apima tokius kriterijus kaip vietos įvertinimas, tipas, miestas, atstumas nuo naudotojo esamos lokacijos iki laisvalaikio vietos. "Maps.Relax" naudojimo sesijos metu rinks informaciją apie vartotojo naudojamus filtrus ir objektų paiešką, kurią mūsų žiniatinklio serveris analizuos ir panaudos naudotojų su panašiais interesais laisvalaikio vietų rekomendacijoms. Jeigu žmogus yra konkrečiai apsisprendęs dėl vietovės į kurią norėtų nuvykti - yra galimybė pasinaudoti paieškos laukeliu. Šis funkcionalumas leis naudotojams ieškoti vietovių žemėlapyje nesinaudojant rekomendacijomis ir iškart suteiks informaciją būtent apie šią vietą. Norint užtikrinti progamėles patogumą, vartotojams galima išsaugoti tikrai patikusias laisvalaikio vietas į mėgstamiausių vietovių sąrašą. Žemelapyje paspaudus ant objekto žymeklio bus galimą jį išsaugoti, o iš mėgstamiausių vietų sąrašo paspaudus ant išsaugotos vietos laukelio objektas bus pažymėtas žemėlapyje. Pasitelkiant žiniatinklio serverio pagalbą, kuris valdo duomenų bazę, palaiko aplikacijų programavimo sąsają ir vykdo skaičiavimus specialiai parašytais algoritmais, taip yra garantuojamas stabilus programėlės veikimas ir vartotojų informacijos saugumas. Norint užtikrinti stabilų programėlės veikimą taip pat naudojami instrumentiniai ir vienetiniai testai.

Introduction

The goal of this project was to develop an Android application that allows users to find the most acceptable leisure place to spend their free time and find it in the least time-consuming way by using algorithms that analyze information about the user and the objects. Users will find functionalities such as filtering, searching, checking weather forecasts, saving favorite locations, finding what similar users search for, quickest routes to objects and distance to them. These functionalities will significantly help users to make fast decisions where to spend their time free time.

1 Analysis

The following analysis consists of advantages, disadvantages and a general sense of feeling of the user with the aim to compare the competitiveness of other similar Android applications in the market. Three different Android applications were taken into account to determine this situation. Essentially, analyses that were taken into consideration were completed by our team. Therefore, there is a possibility that our opinions will not match users' opinions.

1.1 Analyzed applications

1.1.1 "Google Maps" by Google LLC

Advantages

- Large selection of categories - objects are categorized into different categories that make searching for locations quicker and easier.
- "Google Maps" provide a street view that allows users to check the visual view of the location they are searching for.
- Objects have star classifications. This helps users to visually see the ratings of objects and to form expectations according to other people's reviews.

Disadvantages

- No disadvantages were found.

Conclusion

Overall "Google Maps" has great functionality and covers most of the users' needs. The application also provides users with a lot of information about objects such as pictures, ratings and reviews which helps to form an overall opinion about particular places before going there.

1.1.2 "Bikemap" by Bikemap GmbH

Advantages

- A function to see different layers of the map (3D, Night, Satellite). That helps to plan your routes depending on terrain and environment.
- A function to save routes is very useful and allows users to go through that route many times.
- A function to share routes with other users.

Disadvantages

- Most of the functions are paid. Due to this situation, an application lacks some important features like seeing different layers of the map or using offline maps.
- Annoying advertisement which promotes "Bikemap premium" every time you open the application. This is definitely the worst feature of the app.

Conclusion

Overall "Bikemap" delivers a clear example on how the application should look like. The functionality of the application is great, although most of the features are paid. The function layout is clear and comfortable for everyday use. "Bikemap" gave a few great ideas for our project like using a few map layers instead of one.

1.1.3 Automatic recommendation system based on hybrid filtering algorithm

We analyzed study by Sharma Sunny, Rana Vijay and Malhotra Manisha which is about collaborative filtering. While reading this study we understood how user similarity is used, how is it calculated. The article clearly defined the advantages of this algorithm and problems as well. This study has also encouraged us to research more about collaborative filtering and that helped us to decide whether we should use it in our application or not.

Advantages

- Better recommendations - higher chance that user will like the recommendation since it is based on similar user's preferences
- Application improvement - our recommendation algorithm which is based on score system gives very objective recommendations. User similarity algorithm with more personalized recommendations greatly improves the quality of the recommendations, therefore, giving users better experience while using the application.

Disadvantages

- Cold start - algorithm does not work well when there is little data about users. Problem occurs in applications which just started.
- Scalability - as our community will grow, there will be more data to analyze therefore making the algorithm work much slower.

Conclusion

Even though collaborative filtering is bad at cold start we still decided to go for it. It helps us to give more subjective recommendations bringing better user experience.

1.2 Analysis conclusion

Analyzed applications helped us to generate more ideas for "Maps.Relax" application. Also, it allowed us to see how similar applications look from users' perspective and which functionalities fit users' needs the best. Additionally, we found great ideas such as user similarity algorithm which improves the quality of our application.

2 High-level overview

2.1 Users

- **Non-registered user** - has the ability to use most of the functions like search locations, use maps and navigation, read reviews and check the weather in any location of Lithuania. They also will have the ability to create an account.
- **Registered user** - can use all functions available in the application including saving locations, leaving reviews and also getting recommendations from application special algorithm.
- **Admin** - can use all functions available and manage users. An admin inherits use case models from the registered user.

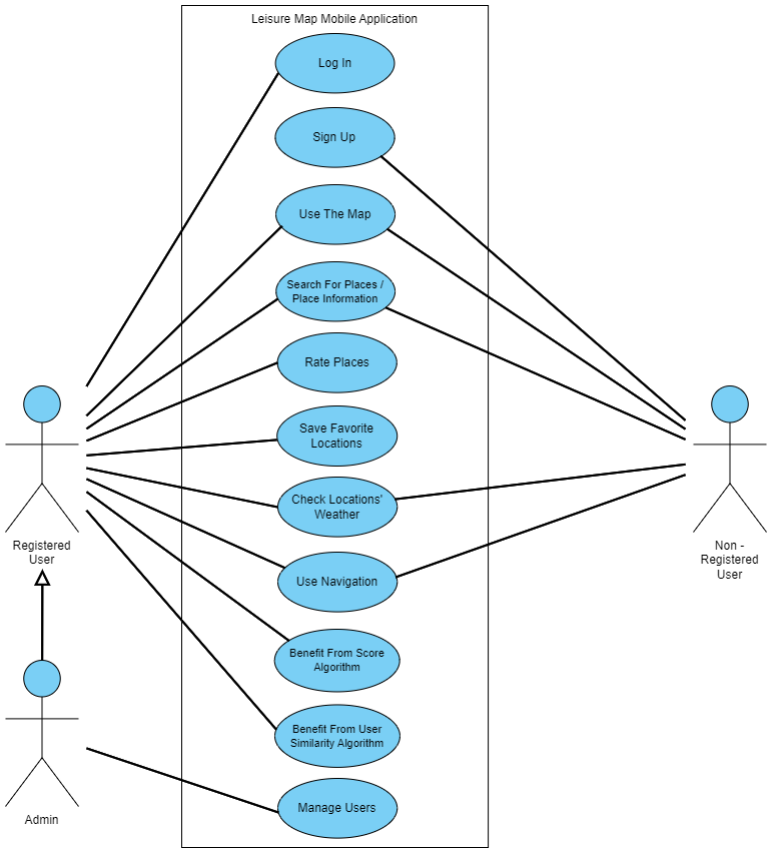

Figure 1. Use case diagram

2.2 Location

- Source code will be uploaded on GitLab repository available at: https://git.mif.vu.lt/mamu8341/gis

2.3 Responsibilities

- Provide users with information about all leisure places around Lithuania.
- Recommend leisure places based on collected data and preferences of users.
- Provide users with the quickest routes to reach desired leisure places.

2.4 Need

The problem with today's applications is that they do not provide any recommendations for users that they might be interested in. Our application solves this problem, essentially, the most suitable leisure place will be picked while taking into consideration users' preferences or collected data about them. Moreover, users will be navigated through the quickest route to reach the chosen leisure place. Additionally, enabling customers to check weather forecasts for chosen leisure places will certainly enhance convenience.

3 Preliminary design

After starting the application users will be redirected to the main menu. Activities can be changed by clicking buttons. "Sign In/Sign Up" activities allow users to log in or register. "Map" activity shows map and users have the ability to search for leisure places directly. "Find Activities" activity proposes recommendations where to spend free time depending on users' selected preferences. "Log out" button terminates the session of logged-in user. However, the application will keep running and a user will be considered as not logged-in.

Figure 2. Design of the Android application

4 System architecture

4.1 UML Deployment diagram

The UML deployment diagram illustrates the execution architecture of the system, representing nodes as hardware components. Software components are operating inside of hardware components and are noted as artifacts.

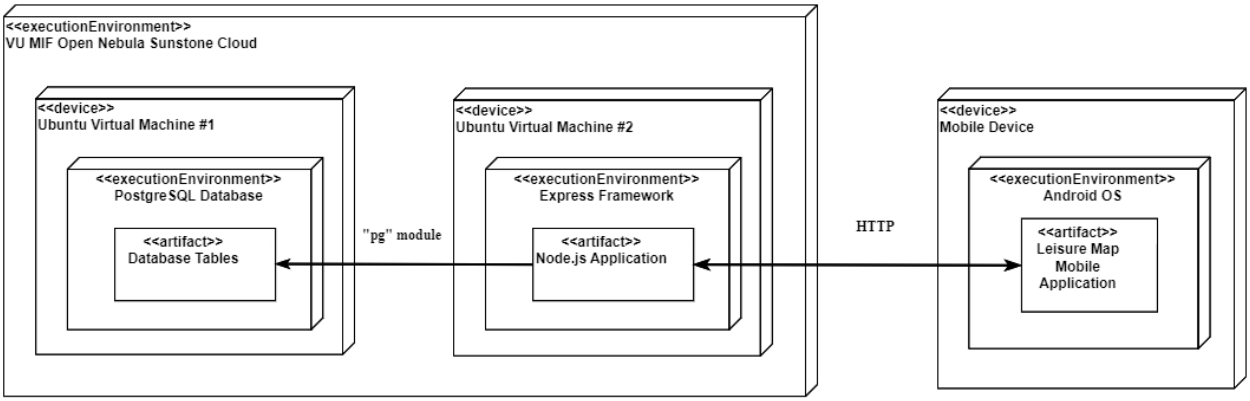

Figure 3. UML Deployment diagram

- **Ubuntu Virtual Machine #1.** A version 20.04 of Ubuntu is installed on the virtual machine which is on a cloud computing platform Open Nebula. The virtual machine with a PostgreSQL database created where log in credentials and saved favorite places. Moreover, a quantity of tags of selected objects during the session will be stored in the database.
- **Ubuntu Virtual Machine #2.** A version 20.04 of Ubuntu is installed on the virtual machine. The web server uses Express framework which runs Node.js application. Furthermore, the web server requests data from external APIs. Additionally, it manages user logins, account creations.
- **Android Mobile Device.** Leisure Map Mobile application is used on the Android Mobile Device.

4.2 Web server

A web server will improve Android application's performance by executing functions in web server rather than in application. Web server consists of functions:

- **API calls** - all API calls will be made from the web server. This ensures the security of user's IP since Android application will not be directly connected to the API.
- **Caching API data** - LRU cache will be used to make reduce amount of API calls. It will increase application's performance as well it will lower chances of getting IP blocked from API in case a web server exceeds maximum allowed requests.
- **User similarity algorithm** - function to calculate find most similar user and based on that recommend places to visit.
- **Database management** - function to execute CRUD operations with database.

4.3 Database

PostgreSQL remote database is used to store data from Overpass API since the API is known to be slow. Also, a database allows to store information about users that includes usernames, passwords and data about user interests.

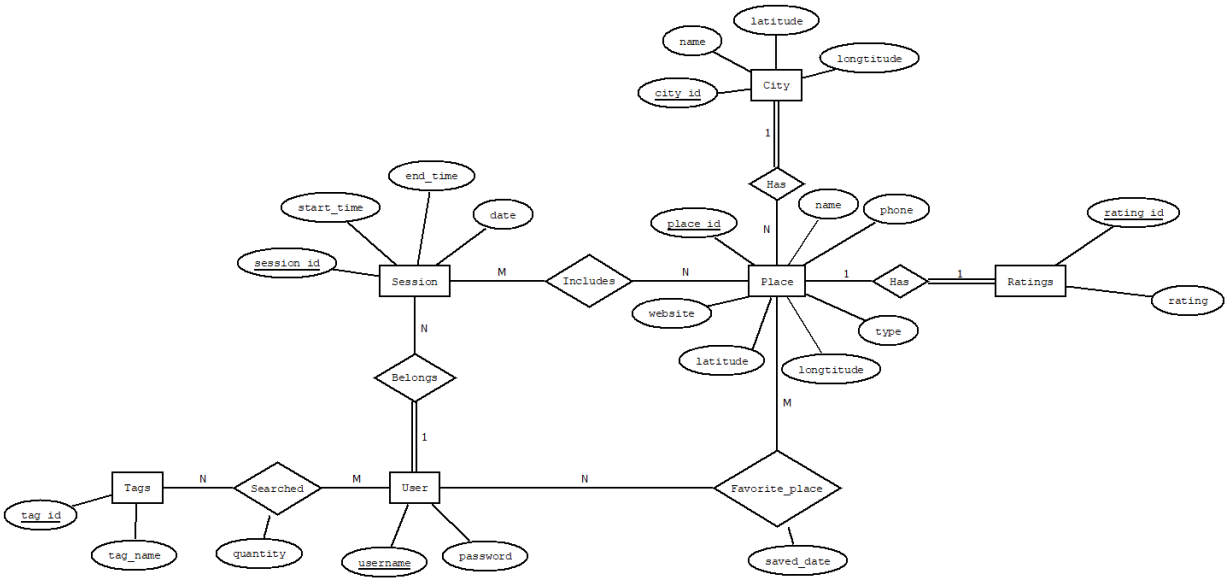

Figure 4. Remote database ER diagram

4.4 System context diagram

The context diagram represents external factors that are interacting with our internal system. A system context diagram is used to depict defined boundaries between the system and its environment. The internal software is connected with external entities that interact with the system.

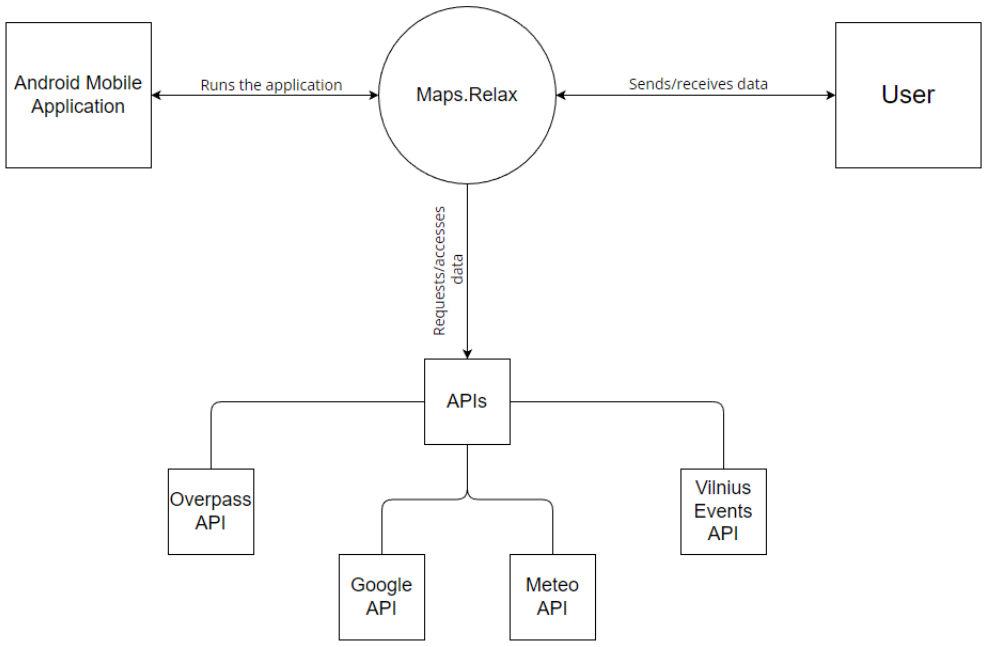

Figure 5. System context diagram

4.5 Data flow

The diagram above represents data flow with API. An Android application calls the web server with the specific route (e.g. ip:port/weather?city=vilnius), then the function of the web server takes parameters provided in url query and makes API call based on given parameters. After that the web server pulls data from API and sends that data back to the Android application.

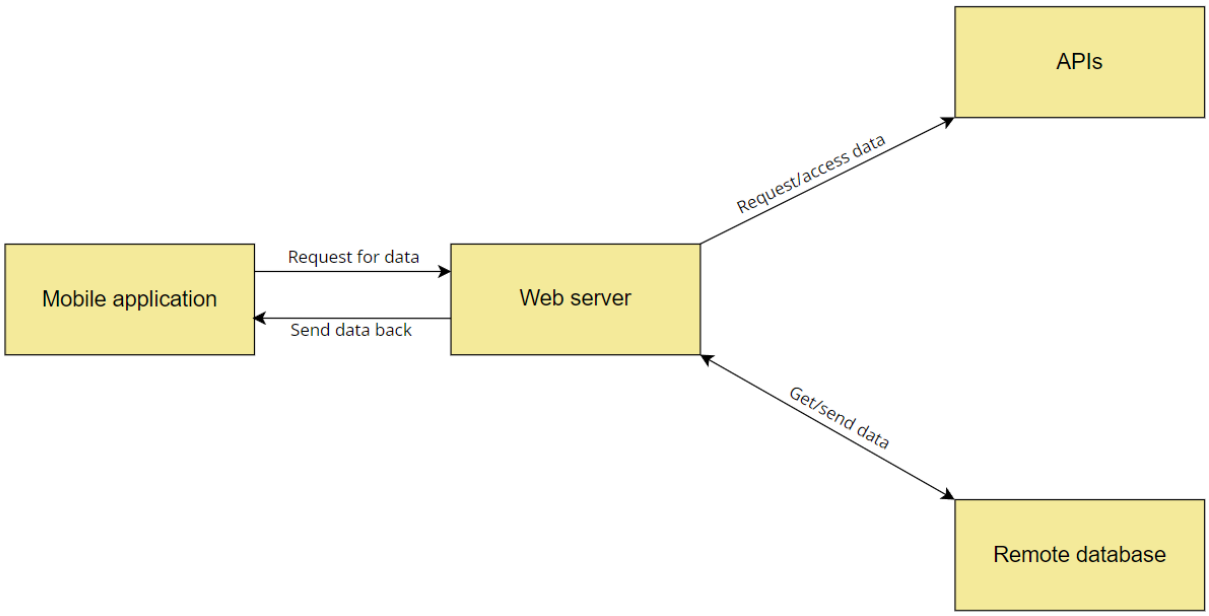

Figure 6. Data flow diagram

5 Log In and Sign Up features

5.1 Users registration

In order to gather the data of registered users and differentiate them, "Maps.Relax" takes advantage of the registration system. Primarily, account creation allows users to obtain access to all possible features inside the application. To accomplish that users ought to enter the following details such as a username, password and confirm the password. During the process of registration, the password is hashed with an adaptive Bcrypt hash function which is accessible through the Bcrypt library specifically compatible with JAVA.

5.2 Bcypt password hashing

Bcrypt hashing function was used to hash and salt passwords securely. The salting process guarantees that a random sequence of characters is generated for each new hash. It was incorporated to prevent the predictability of the password because for the same password precisely the same hash would be produced every single time. The iteration count can be increased to make Bcrypt slower allowing it to scale with computing power. Due to Bcrypt slowness, it reduces the number of passwords per second an attacker could hash while arranging a dictionary attack. The web server uses a module called "bcryptjs" which can get the salt from the hashed password in the database. The password that is input by the user in the application during login is hashed by the module with the same salt that was used in the sign-up process. Then the module checks whether the hashes match.

Figure 7. Bcrypt hashed password example

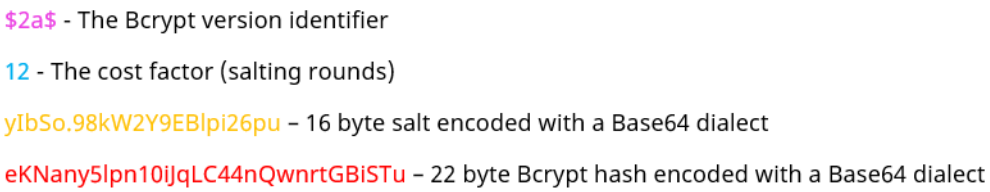

Figure 8. Bcrypt hashed password explanation

6 Technologies and Tools

This section describes programming languages, tools, APIs used in the project.

6.1 Programming languages

- Java programming language will be used in the process of Android application development.
- SQL is a standard language, it will be used for storing and retrieving data in databases.
- JavaScript will be used to create a web server application and implement its functions.

6.2 Other languages

- XML will be used for the user interface part of the mobile application.

6.3 Tools

- OpenNebula is an open-source cloud computing platform that will act as a hypervisor. It will allow to monitor and manage virtual machines on a single shared environment.
- Android Studio is used for the development of Android application.
- PostgreSQL is an object-relational database management system that will be used for data storage.
- Visual Studio Code is software used for a web server code editing.
- Putty is software used to connect to web server's and database's virtual machines
- Firebase Test Lab is a service which helps to test android application activities
- Thunder Client is extension used to test web server responses

6.4 Modules and Libraries

- "lru-cache" is a JavaScript module which will be used to implement least recently used cache.
- "pg" is a JavaScript module which will be used to connect web server to database.
- "chai" is a JavaScript module which will be used for web server unit testing.
- "bcryptjs" is a JavaScript module which checks whether passwords match.
- Volley is a library to facilitate networking for Androids apps that enables HTTP requests such GET or POST, essentially, send and receive data from the web server.

6.5 APIs

- Overpass API - provided by OpenStreetMap which gives the ability to see different locations on the map.
- Google Maps API - for getting maps, distance, routes, getting reviews and ratings of locations.
- Vilnius Events API - providing information about events in Vilnius city.
- Meteo API - providing information about the weather in Lithuania.

7 Functional Requirements

7.1 Sign In/Sign Up features

- Users will have the ability to create accounts.

7.2 Interactive Lithuania map

- Users will have the ability to see their current location and look for leisure places.

7.3 Search of leisure places

- Users will have the ability to search for specific objects particularly in Lithuania and will be provided with additional information such as working hours and ratings.

7.4 Quickest routes

- Users will have the ability to search for the quickest routes to reach their desired leisure place.

7.5 Rating leisure places

- Registered users will have the ability to rate leisure places from one to five stars.

7.6 Saving favorite locations

- Registered users will have the ability to save locations.

7.7 Leisure place recommendations

- Corresponding to the current location of a user, preferable type of activity, the rating of leisure place and similar users preferences, the algorithm will find the most suitable options for users to spend their free time.

7.8 About

- Users will have the ability to familiarize with general information about the app.
- Users will have the ability to look through security and privacy policies.

8 Non-functional Requirements

8.1 Compatibility

- The app will be compatible with Android 9.0 and greater. No future plans to make the application compatible with iOS.

8.2 Internationalization

- A supported language on the app by default will be English.

8.3 Usability

- Any user who has Android mobile phone with Android version 9.0 or greater will be able to fully utilize applications features.

8.4 Reliability

- The application should not have any performance issues or any bugs that would prevent user from using the application.

8.5 Security

- Sign up feature will not use any real names, e-mail addresses, phone numbers in order to prevent users from using personal information.
- Passwords will be hashed, thus, the the original password cannot be decrypted.

9 Algorithms

9.1 User similarity algorithm

Our application will keep track of what users are searching so that we could provide with the best possible recommendations. To accomplish this goal we are tracking what type of objects users are searching and using Pearson Correlation Coefficient we are able to find the most similar person for each user. After we find similar user we recommend based on other user's last weeks searches.

9.2 Overpass API data updating algorithm

```
function(request, response){
  Request data from Overpass API
  Parse received data to JSON type
  FOR(run loop until i equals to received data array length - 1){
    From received data get object`s[i] latitude and longtitude
    Using 'local-reverse-geocoder' module find and assign city that is closest to the object[i]
    Send query to remote database{
      IF object[i] id from data does not exist in the table 'place' THEN
        Send query to remote database{
          Insert object[i] into the table 'place'
        }
    }
  }
  Send response that update is completed
}
```

Figure 9. Overpass API updater algorithm

Algorithm used to find new objects that do not exist in the database. The function starts by calling web server route '/update/overpassapi/leisure'. Firstly, algorithm requests leisure place data from Overpass API. Then loop starts and using 'local-reverse-geocoder' node module it assigns closest city to the object[i]. Next algorithm checks if object[i] exists in our remote database. Lastly, if the object[i] does not exits then it gets added to the database table 'places'.

9.3 Storing weather forecast in LRU cache algorithm

```
function (request, response) {
   Request city name in query parameter
   Parse city name so that the first letter is capital
   Send query to remote database to check if the city exists{
   IF the city exists with name provided in query parameter THEN
      Send query to remote database to get city id
      Convert city id from string to integer
      IF cache is not empty with id of the city THEN
         Send weather forecast data to the user
      ELSE
         Insert city name into Meteo API URL
         Encode URL so that 'request' module accepts Lithuanian letters
         Make new API call with encoded URL
         Send weather forecast data to the user
         Save data in LRU cache and store it for one hour
   ELSE
      Send response that city does not exist
   }
}
```

Figure 10. Storing weather forecast in LRU cache algorithm

This algorithm stores Meteo API weather forecast data and figures in which cache to put which city. Function starts when route ’/weather?city=’ gets called with a parameter, e.g. ’Vilnius’. Then algorithm checks if the city exists in our database. If city does exists then query is sent to remote database to get the city id. Then algorithm check if cache with id of the city is not empty. If there is data in that cache then data is function returns data to the user. If cache is empty then algorithm makes new API call, sends data back to the user and saves it in cache with that city id for one hour.

9.4 Android Application object request algorithm

```
Instantiate new request to weberver using Volley library
onResponse(response) {
  try {
    Get array from JSON
      FOR(run loop until i is array length - 1)
        Get JSON object
        Get object[i] latitude, longtitude, type, name, city and rating
        IF objects does not have a name
          name is equal to type
        Location startPoint is equal to users' position
        Location endPoint is equal to objects' position
        Calculate distance between user and object
        Calculate score of object
        Create new object(name, distance, latitude, longtitude, rating, city, type, score)
        Add object to objects list
    }
  catch(JSON exception) {
    Print stack trace
  }
  Sort list using sortList() function
  FOR(run loop for each object in objects list)
    Create textView for object using createTextViews() function
}, response Error Listener {
  On Error Response displays error using Toast
}
Add request to queue
```

Figure 11. Android Application's request to the web server

Android Application requests data from the web server. Data is received in JSON format. Objects are taken from JSON array and data of each object such as name and city is taken, moreover, distance and score is calculated for each object. New object is created and added to list. Later the list is sorted using sorting algorithm.

9.5 Android Application object search algorithm

```
Marker currentMarker equals to null
searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        public boolean onQueryTextSubmit(String s) {
          location is equal to user's input from the search box
          for (run a loop through each city marker) {
            IF city marker title in lower cases equals to entered location name in lower cases {
              set listview as invisible
              pointToPosition(city marker position)
              animate camera on the map(update camera to new latitute and longtitude city position under zoom level 10)
            }
          }
          for (run a loop through each town marker) {
            IF town marker title in lower cases equals to entered location name in lower cases {
              set listview as invisible
              pointToPosition(town marker position)
              animate camera on the map(update camera to new latitute and longtitude town position under zoom level
13)
            }
          }
          IF currentMarker does not equal to null {
            remove current marker
            set currentMarker value to null
          }
          for (run a loop through each object) {
            IF object name in lower cases equals to entered location name in lower cases {
              IF currentMarker equals to null {
                currentMarker = add a marker on the map(set marker position(to latitute, longtitude, title and type of
object and set color to blue)))
                set listview as invisible
                pointToPosition(created current Marker position)
                animate camera on the map(update camera to new latitute and longtitude of current marker position under
zoom level 15)
              }
            }
          }
```

Figure 12. Android Application's search box and navigation to objects

A search box allows to search for cities and towns. Once, the user enter the full name of location the he will redirected to particular marker. However, for regular objects the marker is created only when user searches for the object. In a case a person enters a new name of the object, the previous marker will be removed from the map and the new marker for a new searched object will be added.

10 Testing

10.1 Android application testing

10.1.1 Firebase Test Lab

Android application tests were done using Firebase Test Lab. This service helps with automatic application activity change testing, button and text view layout on different android devices. Firebase Test Lab provides informative feedback with screenshot cluster and video material of testing progress as well as performance statistics. The service also allows uploading custom tests written by developers themselves. This option will be used to test user input information such as password and e-mail (e.g. whether the email or password user input has the correct format). Also, Test Lab gives insight about implementation issues and errors that were missed while testing ourselves.

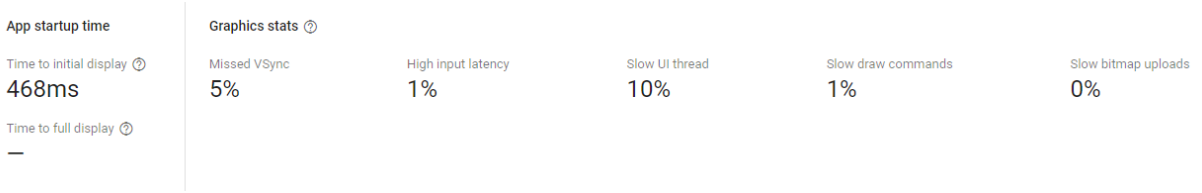

Figure 13. Android application performance statistics

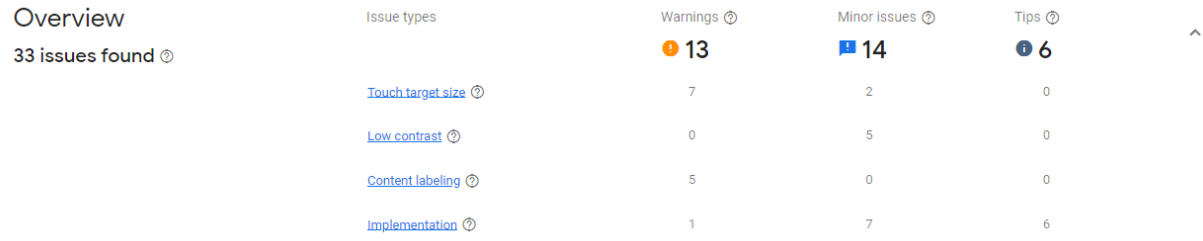

Figure 14. Test Lab AI review of applications' code

10.1.2 Instrumented tests

Activities have a set of tests which check whether all elements in the activity are created and work as intended.

Activity launch tests

Purpose: test monitors whether the activity launches successfully.
Test steps:

- Start activity.
- Check whether the activity started and view is not null.

Tested activities: Main Menu, Login, Leisure Map, Find Activities.

Element tests

Purpose: test checks if all the elements in the activity XML file are created after the activity launches.
Test steps:

- Start activity.
- Chose element.
- Check if element is created (not null).
- Repeat the process with all the elements.

Tested activities: Main Menu, Login, Leisure Map, Find Activities.

Button tests

Purpose: test checks if clickable buttons open required activities
Test steps:

- Start activity.
- Chose element.
- Check if element is created (not null).
- Repeat the process with all the elements.

Tested activities: Main Menu, Login.

10.2 Web server testing

To test web server functions we decided to use Thunder Client extension in Visual Studio Code and Mocha framework. Extension gives the ability to send HTTP/HTTPS requests with methods GET, PUT, POST, DELETE. With Thunder Client we can manually test if SQL queries are working properly, is data we get from APIs is correct. Mocha framework tests our web server's routes and checks response codes and data type.

10.2.1 Web server connections testing

Web server has function assigned to route /testconnections that runs test function and gives back results in JSON format.

- **APIs** - to test connections with APIs we implemented a function to send 1 API call to each API. If the response status is 200 - connection with APIs was successful. If no response code or any other response code was sent - API is either down or not working.
- **Database** - to test connection with database we implemented function to connect and send simple query ("SELECT 1+1 AS SOLUTION") to the database. If the web server successfully sends a query to the database - the connection with the database was successful. Otherwise - connection with database failed.

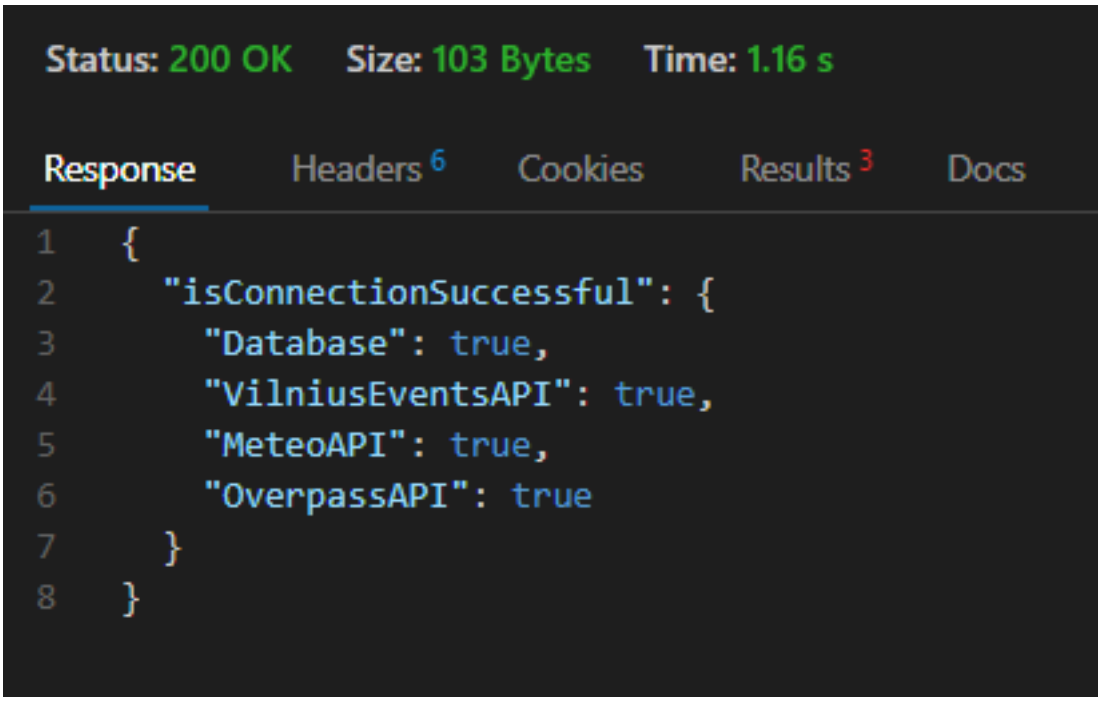

Figure 15. Testing connections with APIs and our remote database

10.2.2 Testing with Thunder Client extension

We are able to manually test our web server using Thunder Client extension in Visual Code Studio. With the extension we are able to see response time of functions that makes API calls or queries to the database. We can also check response code, what data we get, data size and type. Thunder Client extensions also provides with automated testing to check all these things mentioned before.

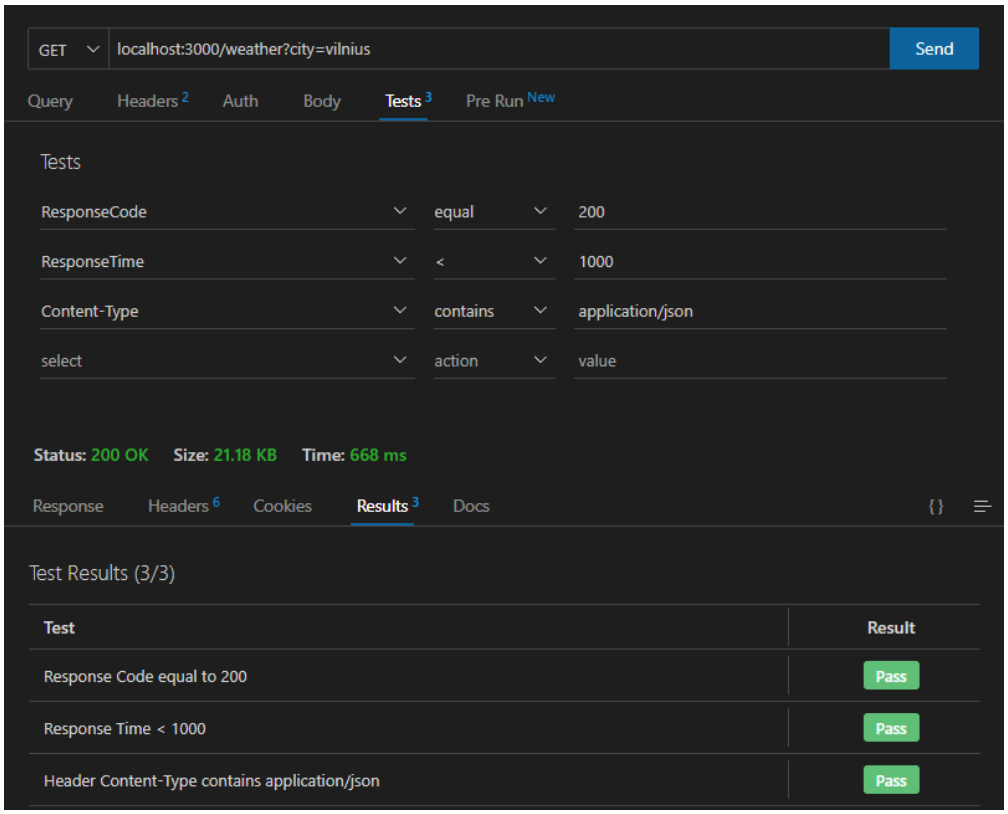

Figure 16. Thunder Client extension test example

10.2.3 Unit testing with Mocha framework

Mocha framework was used to test web server functions where API calls and SQL queries are made.

Test case 1: testing error handling We are able to test if web server's functions are able to handle errors. In this case we tested how function which sends queries to remote database handles errors. 2 queries were sent to the database. Test is expecting code 200 because function sends response code 200 if everything is fine. Otherwise response code is 404 and test fails.

```
/GET table place that exists
  √ it should GET all places from database (2116ms)

/GET table locations that does not exist
  1) it should GET all places from database


1 passing (2s)
1 failing

1) /GET table locations that does not exist
     it should GET all places from database:

    Uncaught AssertionError: expected { Object (_events, _eventsCount, ...) } to have status code 200 but got
404
```

Figure 17. Testing error handling with database queries

Test case 2: testing API calls With this test we can check if we get data back from API and if the data is JSON type. In this case three routes were called to get data from three different APIs. If response code is 200 and data type of response is JSON test succeeds. Otherwise test would fail.

```
/GET weather
  ✓ it should GET weather forecast for Vilnius city (659ms)

/GET events
  ✓ it should GET Vilnius Events data (907ms)

/GET restaurants
  ✓ it should GET restaurant data from Overpass API (3754ms)


3 passing (5s)
```

Figure 18. Testing APIs

Test case 3: testing account creation With this test we can see if account creation function works. In this case test tried to create 2 accounts with usernames 'gediminas' and 'arthur'. Response codes indicate account creation status. Account with username 'gediminas' was successfully created because username was available while account with username 'arthur' was not.

```
Creating user that does not exist
  ✓ it should should create new user (617ms)

Creating user that already exists
  1) it shouldnt create new account because user already exists


1 passing (1s)
1 failing

1) Creating user that already exists
     it shouldnt create new account because user already exists:
```

Figure 19. Testing account creation

Conclusions and Recommendations

Taking everything into consideration "Maps.Relax" Android application seeks to solve the issue of lack of suitable recommendations that are useful for the user and helps to save their time finding places where they can spend leisure time. The app relies on essentials algorithms such as collaborative filtering and object score algorithm to ensure and enhance convenience. Furthermore, "Maps.Relax" has many more useful functionalities like places filtering, searching, checking weather forecast and favorite locations.

References

[1] Lakshmi Prasanna Chitra and Ravikanth Satapathy. Performance comparison and evaluation of node.js and traditional web server (iis). In *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, pages 1--4, 2017.

[2] Man D. Nguyen, Thang Q. Huynh, and T. Hung Nguyen. Improve the performance of mobile applications based on code optimization techniques using pmd and android lint. In Van-Nam Huynh, Masahiro Inuiguchi, Bac Le, Bao Nguyen Le, and Thierry Denoeux, editors, *Integrated Uncertainty in Knowledge Modelling and Decision Making*, pages 343--356, Cham, 2016. Springer International Publishing.

[3] Sunny Sharma, Vijay Rana, and Manisha Malhotra. Automatic recommendation system based on hybrid filtering algorithm. *Education and Information Technologies*, 27(2):1523--1538, Mar 2022.

[4] Xican Wang, Yanheng Liu, Xu Zhou, Xueying Wang, and Zhaoqi Leng. A point-of-interest recommendation method exploiting sequential, category and geographical influence. *ISPRS International Journal of Geo-Information*, 11(2), 2022.

[5] Triyanna "Widiyaningtyas, Indriana Hidayah, and Teguh B." Adji. User profile correlation-based similarity (upcsim) algorithm in movie recommendation system. *Journal of Big Data*, 8(1):52, Mar 2021.