# Principi programskog inženjerstva na primjeru aplikacije za praćenje položaja računalnog miša

**Keser, Romano**

**Undergraduate thesis / Završni rad**

**2024**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **Polytechnic of Međimurje in Čakovec / Međimursko veleučilište u Čakovcu**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:110:838320

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-12-19**

*Repository / Repozitorij:*

Polytechnic of Međimurje in Čakovec Repository - Polytechnic of Međimurje Undergraduate and Graduate Theses Repository

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVA

**Romano Keser, 0313022737**

# Principi programskog inženjerstva na primjeru aplikacije za praćenje položaja računalnog miša

Završni rad

Čakovec, rujan 2024.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVA

**Romano Keser, 0313022737**

# Principi programskog inženjerstva na primjeru aplikacije za praćenje položaja računalnog miša

# Principles of software engineering on the example of application for tracking mouse position

Završni rad

Mentor:

Dr. sc. Bruno Trstenjak, prof. struč. stud.

Čakovec, rujan 2024.

# MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

## PRIJAVA TEME I OBRANE ZAVRŠNOG/DIPLOMSKOG RADA

Stručni prijediplomski studij:

☐ Računarstvo ☐ Održivi razvoj ☐ Menadžment turizma i sporta

Stručni diplomski studij Menadžment turizma i sporta: ☐

Pristupnik: ROMANO KESER, JMBAG: 0313022737
(ime i prezime)

Kolegij: Programsko inženjerstvo i informacijski sustavi
(na kojem se piše rad)

Mentor: dr. sc. BRUNO TRSTENJAK, prof. struč. stud.
(ime i prezime, zvanje)

Naslov rada: Principi programskog inženjerstva na primjeru aplikacije za praćenje položaja računalnog miša

Naslov rada na engleskom jeziku: Principles of software engineering on the example of application for tracking mouse position

Članovi povjerenstva:
1. dr.sc. Sanja Brekalo, prof. struč. stud., predsjednik
(ime i prezime, zvanje)
2. Jurica Trstenjak, v. pred., član
(ime i prezime, zvanje)
3. dr. sc. Bruno Trstenjak, prof. struč. stud., mentor
(ime i prezime, zvanje)
4. Marija Miščančuk, v. pred., zamjenski član
(ime i prezime, zvanje)

Broj zadatka: 2023-RAČ-8

Kratki opis zadatka: Cilj završnog rada je obraditi principe programskog inženjerstva na primjeru aplikacije za praćenje položaja računalnog miša. U sklopu Završnog rada treba dokumentirati početne zahtjeve i odrediti primarne značajke aplikacije bitne za način izvedbe, cijeli logički proces prikazati UML ili EPC dijagramima, na osnovu dijagrama postaviti konceptualni i detaljni dizajn aplikacije, u proces izrade aplikacije upotrebom vodopadnog ili agilnog modela, izraditi aplikaciju, opisati temeljne principe testiranja aplikacije te definirati uvjete koje aplikacija mora imati za njenu komercijalizaciju.

Datum: 18.9.2024.

Potpis mentora: dr.sc. Bruno Trstenjak, prof. struč. stud.

ZAHVALA

Želim se zahvaliti mentoru dr.sc Bruni Trstenjaku za strpljenje i navođenje tokom pisanja ovog rada. Također se želim zahvaliti profesoru dr.sc. Josipu Nađu koji me naučio kako voditi, pisati i strukturirati dokumentaciju projekta.

Romano Keser

## Sažetak

Ideja za praćenje i analizu pokreta računalnog miša potaknuta je iz knjige koja istražuje napredne tehnologije senzorskog praćenja i njihovu primjenu u svakodnevnom životu. Inspiriran konceptom prikupljanja i analize podataka (engl. *tracking*), gdje se koriste razni senzori za praćenje različitih parametara, razvijena je aplikacija koja prati i analizira pokrete računalnog miša na zaslonu. Ovo praćenje omogućava pohranjivanje koordinata miša te analizu najaktivnijih područja na ekranu, što može pružiti korisne uvide u korisničke navike.

Cilj je ovog završnog rada izraditi funkcionalnu aplikaciju koja bilježi pokrete miša, omogućuje analizu prikupljenih podataka, generiranje toplotne mape i prikaz najaktivnijih sektora korisničkog sučelja. Aplikacija omogućuje korisnicima ispis toplotne mape (engl. *heatmap*) koja vizualizira aktivna i neaktivna područja ekrana, pružajući vrijedan alat za razumijevanje interakcije s računalnim sučeljem. Vizualizacija se prikazuje pomoću boja, slično kao na vremenskom radaru prognoze: crvena boja označava najaktivnija područja, plava manje aktivna, dok crna boja predstavlja potpuno neaktivna područja.

U nastavku rada detaljno se objašnjavaju svi aspekti razvoja aplikacije, uključujući tehničke specifikacije, slučajeve primjene, plan testiranja te puštanja aplikacije u produkcijski rad.

Ovaj završni rad predstavlja konkretan primjer primjene principa programskog inženjerstva u razvoju aplikativnih rješenja, s fokusom na praćenje korištenja računalnog miša te s ciljem poboljšanja korisničkog iskustva i dobivanja novih saznanja o interakciji s računalnim sučeljem. Nadalje, rad će služiti za daljnje istraživanje i razvoj u ovom području, omogućavajući studentima i stručnjacima da bolje razumiju kako senzorske tehnologije i praćenje mogu biti primijenjeni u različitim kontekstima.



**Ključne riječi:** *Računalni miš, analiza pokreta, toplinska mapa*

## Abstract

The idea for tracking and analysing mouse movements was inspired by a book that explores advanced sensor tracking technologies and their applications in everyday life. Inspired by the concept of data collection and analysis, also known as tracking, where various sensors are used to monitor different parameters, an application was developed to track and analyse mouse movements on the screen. This tracking enables the storage of mouse coordinates and the analysis of the most active areas on the screen, providing valuable insights into user's behaviour.

The goal of this project is to create a functional application that records mouse movements, enables the analysis of collected data, generates heatmaps, and displays the most active sectors of the user interface. The application allows users to generate a heatmap that visualizes active and inactive areas of the screen, providing a valuable tool for understanding interactions with the user interface. The visualization is presented using colours, similar to a weather radar: red indicates the most active areas, blue less active areas, whereas black represents completely inactive areas.

The following sections of this work explain in detail all aspects of the application's development, including technical specifications, use cases, testing plans, and the release of the application into production.

This project serves as a concrete example of the application of software engineering principles in the development of application solutions, with a focus on tracking and a goal of improving the user experience and gaining new insights into interactions with computer interfaces. Furthermore, the work will serve as a basis for further research and development in this field, allowing students and professionals to better understand how sensor technologies and tracking can be applied in various contexts.

**Keywords*:*** *Computer mouse, motion analysis, heat map*

## Popis korištenih kratica

**UI** User Interface

# Sadržaj

# 1. UVOD

Ovaj završni rad bavi se razvojem aplikacije za praćenje i analizu pokreta računalnog miša na zaslonu računala. Ideja je proizašla iz proučavanja tehnika praćenja (engl. *tracking*) i njihovih primjena u različitim područjima. S obzirom na sve manju veličinu i poboljšanje performansa računala i senzora, tehnologija praćenja postaje sve značajnija za precizno praćenje različitih aktivnosti. Glavni cilj aplikacije razvijene u okviru ovog rada je pohranjivanje koordinata pokreta miša te analiza najaktivnijih područja monitora, kao i korisničkih navika.

Aplikacija omogućava snimanje pokreta miša i bilježenje pokreta dominantne ruke korisnika. Na kraju snimanja, korisnik može zaustaviti program i generirati toplotnu mapu (engl. *heatmap*) koja prikazuje aktivne i neaktivne sektore monitora računala. Razvojem ove aplikacije, istraženi su potencijalni slučajevi primjene tehnologije praćenja pokreta miša, uključujući prilagodbu radnog prostora korisničkim navikama i analizu općenitih korisničkih navika poput pretraživanja interneta ili upotreba alata u svrhu provođenja administrativnih poslova.

Tehnički aspekti razvoja aplikacije, uključujući dijagram procesa, plan testiranja, slučajeve primjene i planirane izvještaje, detaljno su opisani u radu. Analizirane su prednosti i izazovi primjene ove tehnologije, a rezultati pokazuju da praćenje pokreta miša može pružiti vrijedne uvide u korisničke interakcije s računalnim sustavima, što može pomoći u razvoju prilagođenih rješenja za optimalnije korisničko iskustvo.

# 2. PROGRAMSKO INŽENJERSTVO

Ovo poglavlje osvrnut će se na teorijsku osnovu programskog inženjerstva te će se pobliže opisati njezini principi i svrha nastanka. Programsko inženjerstvo multidisciplinarno je područje koje obuhvaća dizajn, razvoj, testiranje i održavanje softverskih sustava. Glavni je cilj programskog inženjerstva proizvodnja visokokvalitetnog softvera koji zadovoljava potrebe korisnika, uz optimalno korištenje resursa i minimiziranje grešaka.

## 2.1. Elementi softverske podrške

Da bi se razumjela definicija programskog inženjerstva i njegova važnost, potrebno je objasniti pojam softverske podrške. Softverska podrška, poznata i kao softver (engl. *software*), odnosi se na skup programa koji nemaju fizičke dimenzije, ali su ključni za izvođenje računalnih naredbi. Softverska podrška obuhvaća sve potrebne komponente, uključujući podatke, dokumentaciju i računalne programe, koji zajedno omogućuju učinkovito rješavanje računalnih procesa.

## 2.2. Povijest programskog inženjerstva

Razvoj programskog inženjerstva započeo je u pionirskim danima računarstva, kada su prvi programi bili ručno kodirani u strojnom jeziku. Kako su računalni sustavi postajali složeniji, pojavila se potreba za strukturiranim pristupom razvoju softvera.

U 1960-ima, suočeni sa "softverskom krizom", odnosno s problemima poput prekoračenja budžeta, kašnjenja projekata i nepouzdanog softvera, stručnjaci su počeli razvijati prve formalne metode programskog inženjerstva. Ovi rani pokušaji uključivali su koncepte kao što su životni ciklus razvoja softvera i osnovni principi upravljanja projektima (Bohem 1976)

Tijekom 1970-ih i 1980-ih programsko inženjerstvo značajno je napredovalo uvođenjem strukturiranog programiranja i objektno-orijentiranog pristupa. Ove metode omogućile su bolju modularnost i ponovnu upotrebu koda, dok su modeli poput vodopadnog i iterativnog razvoja donijeli veću disciplinu i organizaciju u proces razvoja softvera.

Eksplozija interneta 1990-ih donijela je nove tehnologije i promijenila paradigme razvoja softvera. Agilne metodologije, koje naglašavaju fleksibilnost, timski rad i brze iteracije postale su ključne za uspješno upravljanje softverskim projektima u ovom razdoblju.

Danas je programsko inženjerstvo složena i raznolika disciplina koja obuhvaća brojne prakse, metodologije i alate za razvoj, testiranje i održavanje softvera. Evolucija ovog područja

nastavlja se s trendovima poput *DevOpsa*[1], kontinuirane integracije i isporuke, te primjene umjetne inteligencije, koji oblikuju budućnost razvoja softvera.

## 3. POČETNA IDEJA I CILJ RADA

Ideja za razvoj aplikacije proizašla je iz knjige *The Inevitable* (Kelly 2016), koja istražuje dugoročne trendove u tehnologiji i njihov utjecaj na društvo. Točnije, inspiracija je došla iz poglavlja o *loggingu*[2], koje se fokusira na važnost praćenja i bilježenja korisničkih aktivnosti.

U knjizi je istaknuto kako s razvojem tehnologije i smanjenjem dimenzija senzora praćenje (engl. *tracking*) različitih događaja postaje sve lakše i učinkovitije. Kao rezultat toga otvaraju se nove mogućnosti za praćenje raznih događaja i promjena. Različiti senzori omogućuju raznovrsne primjene, a smanjenje njihovih dimenzija, zajedno s poboljšanjem kapaciteta baterija, značajno doprinosi ovom napretku. Primjerice, senzori za mjerenje temperature, monitori otkucaja srca i drugi medicinski uređaji sada su smanjeni na minimalne dimenzije, što olakšava praćenje stanja pacijenata, kako u bolnici, tako i izvan nje. Ova tehnološka poboljšanja omogućuju svakodnevno praćenje zdravstvenog stanja, pružajući detaljan uvid u stanje pojedinca, dok smanjenje dimenzija baterije omogućuje slobodnije kretanje.

Danas pametni satovi i pametni telefoni, koji imaju ugrađene senzore malih dimenzija, dodatno unapređuju *logging*. Ovi uređaji mogu pratiti fizičku aktivnost, otkucaje srca, spavanje i druge biometrijske podatke, omogućujući kontinuirano praćenje korisnika u realnom vremenu. Integracija ovih senzora u svakodnevne uređaje čini praćenje pristupačnijim i praktičnijim, otvarajući nove mogućnosti za personaliziranu zdravstvenu skrb i analizu životnih navika (Kelly 2016: 203-229).

Osim praćenja zdravstvenog stanja, senzori mogu pomoći u nadgledanju ponašanja i reakcija tijela. Unatoč indikacijama koje pružaju, ljudi imaju tendenciju ignorirati ili krivo interpretirati podatke zbog svoje prirode. Automatizirani sustavi koji ne zahtijevaju stalnu pažnju korisnika

[1] *DevOps* je praksa u razvoju softvera koja integrira razvoj (engl. *development*) i operacije (engl. *operations*) kako bi se ubrzala isporuka softverskih aplikacija. Glavna je ideja *DevOpsa* promicanje suradnje između razvojnih timova i operativnog osoblja, što omogućava bržu implementaciju, testiranje i puštanje softvera u produkciju. *DevOps* se oslanja na automatizaciju procesa kao što su izgradnja koda, testiranje, implementacija i nadzor, što rezultira bržim povratnim informacijama i mogućnošću brze prilagodbe promjenama. Korištenjem *DevOps* praksi organizacije mogu postići veću agilnost i pouzdanost u razvoju softvera, što je posebno važno u današnjem tehnološkom okruženju koje se brzo mijenja.

[2] *Logging* je proces bilježenja ili praćenja podataka o aktivnostima sustava ili korisnika.

mogli bi riješiti ovaj problem, omogućujući otkrivanje zdravstvenih problema, loših navika i drugih anomalija nakon duljeg perioda praćenja pokreta računalnog miša.

Jedan je zanimljiv primjer upotrebe senzora projekt Uda Wachtera (Kelly 2016: 208-243), koji je izradio pojas koji vibrira u smjeru sjevera. Nakon nekoliko tjedana nošenja pojasa Wachter je razvio novi osjećaj za prostornu orijentaciju, instinktivno osjećajući smjer sjevera. Ovaj primjer pokazuje kako pravilna upotreba malih senzora može dovesti do stvaranja potpuno novog osjeta, u ovom slučaju za orijentaciju (Kelly 2016: 14-29).

Kao drugi primjer praćenja ističe se 'prijenos uživo', poznat i kao engl. *lifestreaming*, ideja o pohranjivanju svih događaja i iskustava u kronološki tok informacija. Ovo je praćenje elektroničkog života u kronološkom poretku, gdje se sve informacije, umjesto u mape, pohranjuju redoslijedom događanja. Ovaj način arhiviranja možemo vidjeti na vremenskoj crti (engl. *timeline*) na društvenim mrežama, gdje se informacije uglavnom prikazuju i sortiraju prema vremenskom slijedu (Kelly 2016: 76-95).

Koncept *lifestreaminga* i *lifeloggінga* pruža dodatnu dimenziju ovoj ideji, gdje se bilježe svi aspekti korisničkog iskustva u kronološkom redoslijedu. Ove tehnologije omogućuju dvadesetčetverosatno praćenje vitalnih funkcija i bilježenje svih događaja u svakodnevnom životu.

U kontekstu projekta ovog završnog rada, praćenje kretanja računalnog miša koje se bilježi u obliku toplotne mape omogućuje prikaz interakcije korisnika s računalnim sučeljem. Uz stručnu analizu podaci prikazani toplotnom mapom mogli bi dovesti do razumijevanja načina na koji korisnik upotrebljava računalno sučelje ili određenu aplikaciju. U vezi s tim, postavljeni su ciljevi rada:

a) praćenje pokreta miša, to jest evidentiranje svih pokreta miša na zaslonu računala i pohranjivanje koordinata miša
b) analizu podataka, odnosno prepoznavanje i analizu korisničkih obrazaca i navika
c) prilagodbu radnog prostora, kao pomoć korisnicima u optimizaciji njihovog radnog prostora, radnih navika i uvid u to kako korisnik može poboljšati svoje grafičko sučelje (na primjer kako rasporediti ikone po radnoj površini)
d) izradu i generiranje toplotnih grafova (engl. *heatmap*).

# 4. METODOLOGIJA IZRADE APLIKACIJE

U ovom poglavlju detaljno su opisane metodologije korištene za razvoj aplikacije za praćenje računalnog miša. Cilj je prikazati kako se vodopadni, spiralni i agilni model mogu primijeniti u izradi softverskog rješenja, te kako svaka metodologija doprinosi uspješnom razvoju projekta.

Za aplikaciju su korišteni sljedeći programski alati:

a) alat za razvoj .NET C# koda *Visual Studio*

b) *WPF* ili *Windows Presentation Foundation*, grafički podsustav unutar alata *Visual Studio* za prikaz korisničkih sučelja u aplikacijama *Windowsa,* upotrijebljen za strukturiranje korisničkog sučelja (engl. *user interface* ili kraće *UI*)

c) programski jezik C# (engl. *C Sharp*) za objektno-orijentirano programiranje, korišten za generalni 'kostur' aplikacije i implementaciju korisničkog iskustva, upravljanje navigacijom unutar aplikacije, pokretanje *Python* skripti, povezivanje na bazu podataka te zapisivanje koordinata miša

d) programski jezik *Python* korišten za generiranje toplotnih mapa (engl. *heatmap*), analizirajući prikupljene podatke i vizualizirajući aktivna područja na ekranu

e) baza podataka *SQLite* korištena kao lokalna baza podataka, koja omogućuje pohranu i pristup podacima o pokretima miša na računalu, te uvid u prijašnja snimanja.

Aplikacija je osmišljena u skladu s određenim postavkama s ciljem zadovoljavanja korisničkih očekivanja. Prva je postavka vezana uz jednostavnost upotrebe aplikacije koja treba imati intuitivno korisničko sučelje koje omogućava snimanje i pregled pokreta računalnog miša. Druga se postavka odnosi na precizno bilježenje koordinata pokreta računalnog miša i generiranje pouzdanih grafova. Posljednja je postavka orijentirana na to da aplikacija omogućuje generiranje detaljnih izvještaja o pokretima miša i korisničkim navikama.

U tehničkom pogledu, aplikacija koristi senzore ugrađene u računalni miš i računalne algoritme za prikupljanje i obradu podataka. Glavne komponente uključuju integrirane senzore u računalnom mišu koji precizno bilježe sve pokrete, softverske algoritme koji služe za analizu prikupljenih podataka, generiranje izvještaja te intuitivno korisničko sučelje.

Opseg završnog rada obuhvaća razvoj aplikacije za praćenje i analizu pokreta računalnog miša. Razvoj aplikacije može se podijeliti u nekoliko točaka:

a) definiranje funkcionalnosti aplikacije**,** odnosno što aplikacija treba raditi i kako će to raditi
b) razvoj tehničkih specifikacija**,** to jest određivanje tehnologija koje će se koristiti za izradu aplikacije
c) izradu dijagrama, točnije izradu UML i EPC dijagrama za vizualizaciju logičkog procesa aplikacije
d) implementaciju i testiranje aplikacije, odnosno razvoj aplikacije, provođenje testiranja jedinica, te provođenje integracijskih i sistemskih testova.

## 5. IZGLED APLIKACIJE

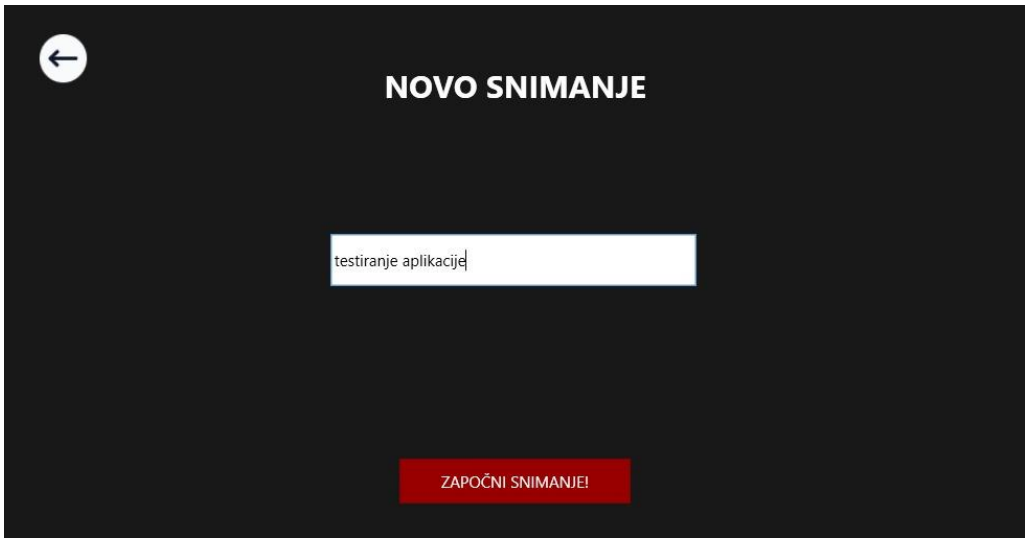
Dizajn sučelja aplikacije jednostavan je i omogućuje laku navigaciju. Pokretanjem aplikacije prikazana su dva navigacijska gumba. Korisnik s početnog ekrana može krenuti s novim snimanjem ili otići na pregled prijašnjih snimki.

**Slika 1.** Početni ekran aplikacije


Izvor: Autor

Klikom na gumb 'Novo snimanje' otvara se prozor gdje se od korisnika traži da dodijeli ime nove sesije snimanja, slika 2.

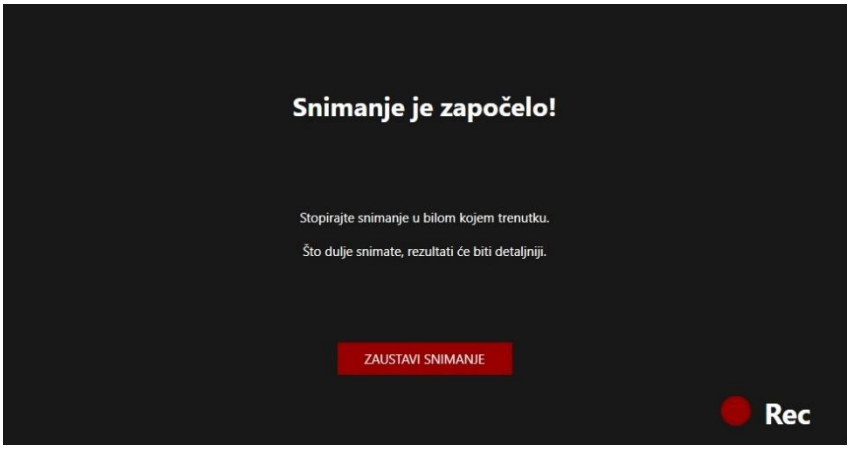
**Slika 2.** Ekran za imenovanje nove sesije snimanja




Ivor: Autor

Klikom na gumb 'Započni snimanje' aktivira se ekran s indikatorom aktivnog snimanja računalnog miša te je s tim korakom započelo snimanje i upis koordinata miša u bazu podataka, slika 3.

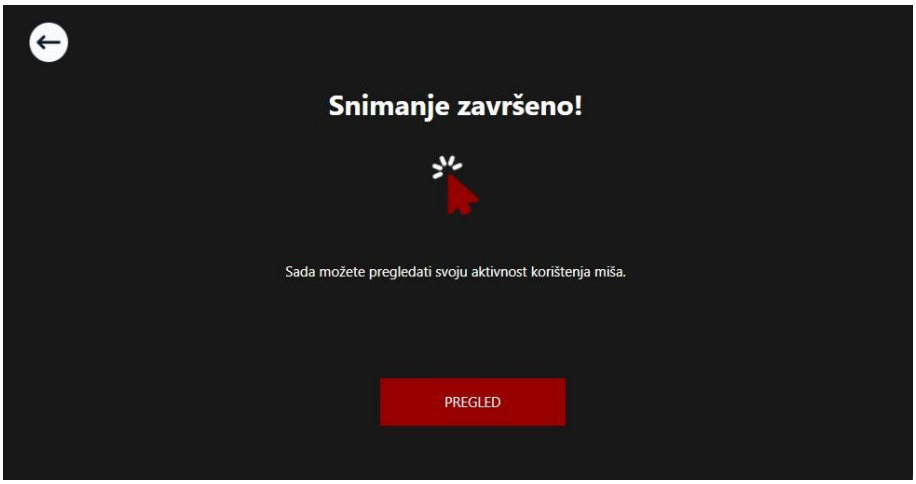
**Slika 3.** Ekran aktivnog snimanja računalnog miša




Izvor: Autor

Za vrijeme snimanja, korisnik može aplikaciju ostaviti da radi u pozadini. Kada korisnik odluči zaustaviti snimanje, klikom na 'Zaustavi snimanje' upis koordinata miša u bazu završava te se otvara novi prozor s naznakom da je snimanje završeno, slika 4. Ovaj prozor jasno prikazuje završetak procesa i omogućava korisniku da vidi potvrdu da su podaci uspješno zabilježeni.

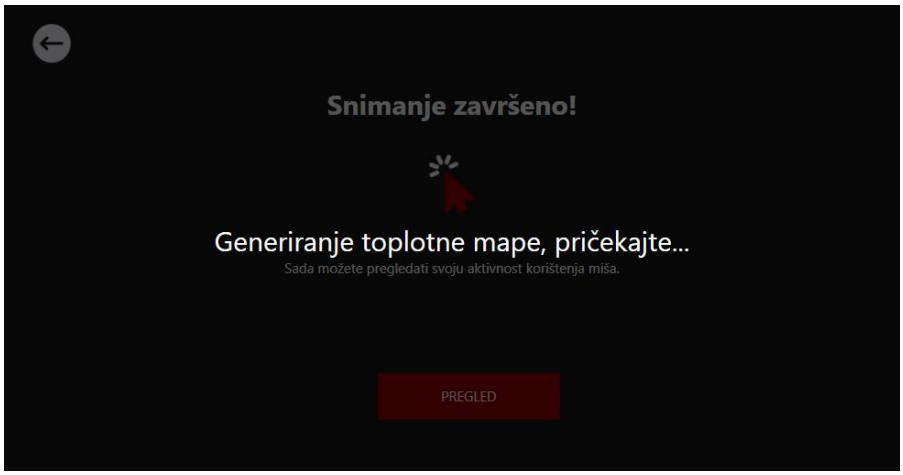
**Slika 4.** Ekran završene sesije snimanja




Izvor: Autor

Korisnik odavde može otvoriti snimljeni rezultat ili pritiskom na bijeli gumb sa strelicom u lijevom gornjem kutu vratiti na početni ekran. Ovaj gumb je aktivan u svakom prozoru aplikacije i njegova funkcionalnost je uvijek ista (povratak nazad).

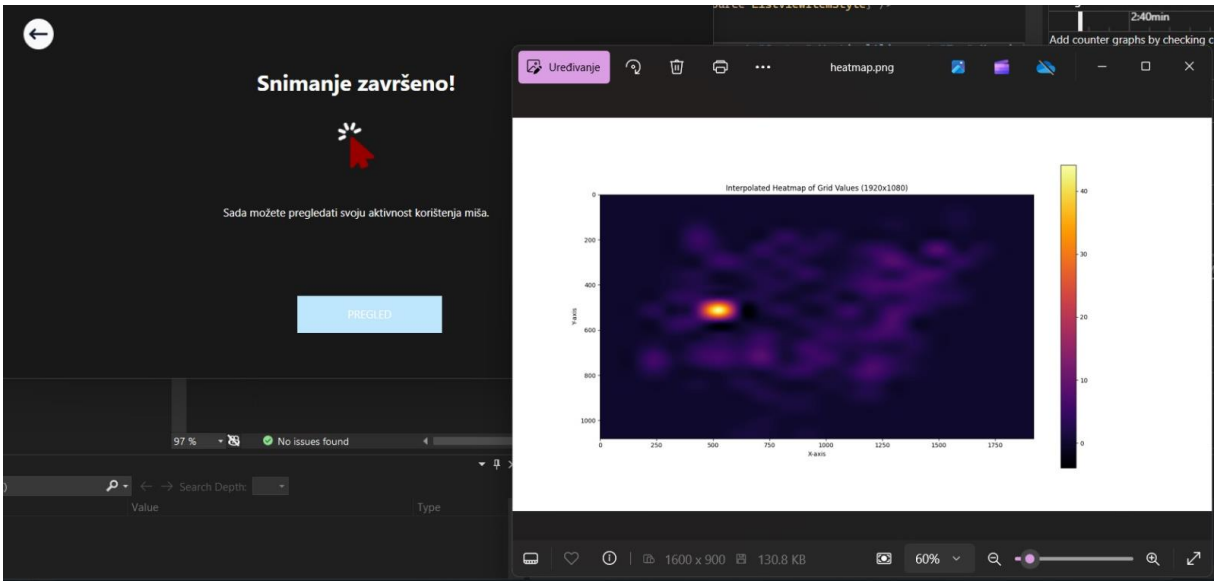
**Slika 5.** Generiranje toplotne mape u proces




Izvor: Autor

Pritiskom na gumb 'PREGLED', *Python* skripta se aktivira i započinje generiranje toplotne mape, nakon čega se automatski otvara generirana slika rezultata. Fotografija toplotne mape automatski se pokreće u standardnom *Windows* pregledniku za fotografije, kao što je prikazano na slici 6.

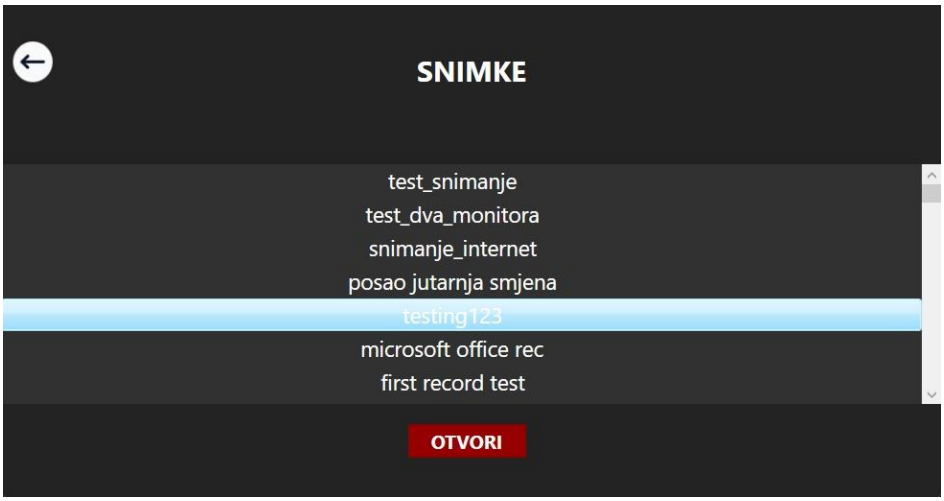
**Slika 6.** Otvaranje fotografije toplotne mape nakon snimanja pokreta računalnog miša




Izvor: Autor

S početnog ekrana, ako korisnik odluči pregledati prijašnje snimke, otvara se ekran s listom svih snimljenih sesija. Lista prikazuje imena svih snimki iz baze podataka.

**Slika 7.** Lista snimljenih sesija




Izvor: Autor

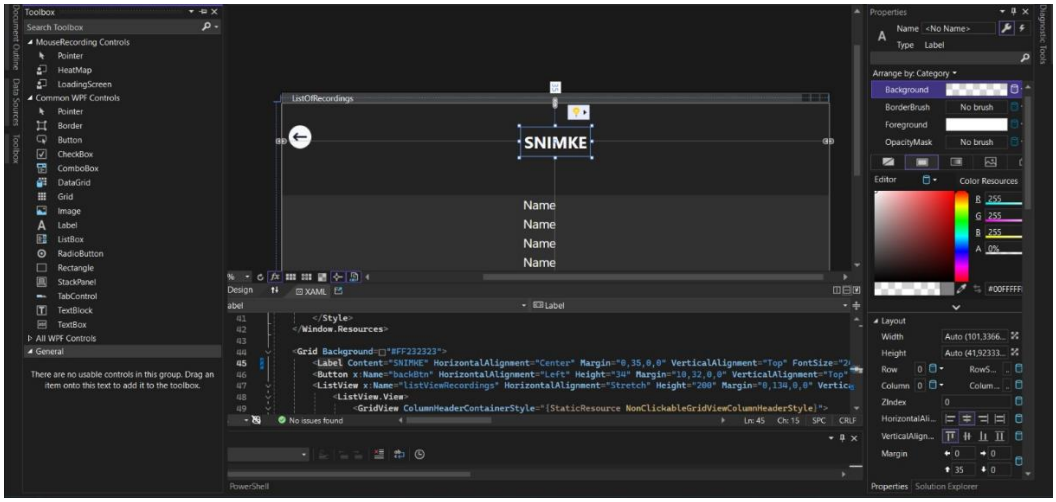
Snimke svih snimljenih sesija prikazane su u jednoj listi. Klikom na jednu od snimaka, korisnik može nanovo generirati toplotnu mapu odabrane snimke iz liste.

### 5.1. WPF alat

Dizajn i kostur aplikacije napravljen je pomoću WPF alata (engl. *Windows Presentation Foundation*) unutar Visual Studio Code-a. Ovaj alat izdan je 2006. godine te omogućava na jednostavan način postaviti bazične elemente aplikacije poput gumbova, listi, boja, slika itd. (Nathan 2007: 9-21)

Svakom dodanom elementu može se pristupiti u kodu i dodatno njime manipulirati.

**Slika 8.** WPF alat




Izvor: Autor

### 5.2. Korelacija između WPF alata i C# koda

Struktura i dizajn aplikacije napravljena je pomoću WPF alata. Svakom dodanom elementu u WPF-u može se pristupiti u kodu. Recimo, gumb za prikaz generirane toplotne mape, dvostrukim klikom na njega unutar WPF alata automatski se generira asinkrona metoda u klasi odgovorne za taj ekran. U ovom slučaju to je metoda *ReviewHeatmapBtn_Click* u klasi *SuccessfulRecordingWindow.cs*.

Klikom na gumb „PREGLED“ na početnom ekranu sa slike 1, aktivira se `ReviewHeatmapBtn_Click` metoda koja prvo poziva `loading.screen = Visibility.Visible` čime se aktivira ekran učitavanja. Zatim, u *Try Catch* metodi, poziva se asinkrona funkcija `_heatmap.Render(imagePath)` koja za argument prima *imagePath,* odnosno, ime naslova korisničkog trenutnog snimanja računalnog miša. Za vrijeme izvršavanja procesa renderiranja aktivan je ekran učitavanja *(*engl. *loading screen*), slika 5., koji se zatvara

nakon uspješnog renderiranja grafa. Nakon generiranja toplotne mape, aktivira se ekran s porukom uspješnog renderiranja i mogućnosti pregleda snimljenoga rada. Odabirom pregleda novo snimljene snimke, otvara se fotografija *png* formata. Fotografija je automatski spremljena u lokalni direktorij aplikacije, pod nazivom kojeg je korisnik zadao prije početka snimanja nove sesije.

**Kod 1.** Primjer C# koda za navigacijski gumb za generiranje toplotne mape

```
1 reference
private async void ReviewHeatmapBtn_Click(object sender, RoutedEventArgs e)
{
    loadingScreen.Visibility = Visibility.Visible; // Show loading screen

    try
    {
        string imagePath = RecordingNameHolder.CurrentRecordingName;    //Get the current recording from Util script
        await Task.Run(() => _heatMap.Render(imagePath));   //load Python script
    }
    catch (Exception ex)
    {
        System.Windows.MessageBox.Show($"An error occurred: {ex.Message}");
    }
    finally
    {
        loadingScreen.Visibility = Visibility.Collapsed; // Hide loading screen
    }
}
```

Izvor: Autor

# 6. LOGIKA RENDERIRANJA

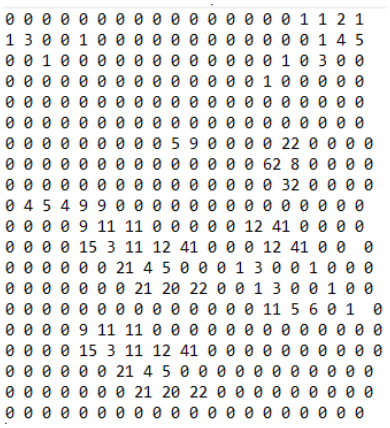
Renderiranje se izvršava u dva različita koraka. Prvi korak je prikupljanje koordinata iz lokalne baze podataka koristeći .NET C# kod i pomoćne klase za komunikaciju s lokalnom bazom podataka. Drugi korak je pozivanje *Python* skripte koja pokreće funkcije *Matplotlib* biblioteku za generiranje toplotne mape.

## 6.1. C# .NET kod

Prikupljanje Podataka: Korištenjem .NET aplikacije, podaci o koordinatama miša se povlače iz baze podataka. Podaci se zatim obrađuju, svaka koordinata se zapisuje u zasebnu ćeliju, kako bi se izračunali brojevi točaka unutar definirane mreže (engl. *grid*) od 20 x 20 ćelija.

Upis koordinata u ćeliju: Rezultati izračuna pohranjuju se u tekstualnu datoteku. Svaki broj predstavlja jednu od 400 ćelija na monitoru, gdje 0 označava potpuno neaktivnu ćeliju, odnosno onu ćeliju preko koje računalni miš nije prešao za vrijeme snimanja ni u jednom trenutku.

**Slika 9.** Tekstualna datoteka s vrijednostima ćelija monitora 20 x 20




Izvor: Autor

Nakon što su podaci zapisani u tekstualnu datoteku, WPF (.NET C#) aplikacija pokreće *Python* skriptu koja dalje obrađuje podatke i generira vizualni prikaz (toplotnu mapu). Skripta se izvršava pomoću programa *render.exe*, koji izvršava sastavljeni (engl. compiled) *Python* kod (kod 3). Ukratko, *Python* skripti prosljeđuju se podaci koordinata, koji se procesiraju pomoću *Matplotlib* biblioteke. Proces generiranja toplotne mape opisan je u poglavlju 7. *Generiranje toplotne mape*.

**Kod 2.** Render metoda

```
/// <summary>
/// Renders the heatmap based on the mouse coordinates recorded in the specified recording.
/// </summary>
/// <param name="currentRecordingName">The name of the current recording.</param>
2 references
public void Render(string currentRecordingName)
{
    int numCellsX = 20;
    int numCellsY = 20;

    try
    {
        DataTable mouseCoordinates = DatabaseHelper.GetMouseCoordinates(currentRecordingName);
        List<(int, int)> recordedCoordinates = ExtractCoordinates(mouseCoordinates);

        string filePath = $"{currentRecordingName}.txt";
        int[,] counts = CalculatePointCounts(recordedCoordinates, numCellsX, numCellsY);
        WritePointCountsToFile(counts, filePath);

        ExecutePythonScript(filePath);
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error during rendering: {ex.Message}");
    }
}
```

Izvor: autor

### 6.2. Python kod

*Python* skripta obrađuje podatke o brojevima točaka kako bi generirala toplotnu mapu (engl. *heatmap*) koja vizualno predstavlja učestalost točaka. U *Python* kodu (kod 3), koriste se sljedeće komponente:

**Torch i NumPy**: Za numeričku obradu podataka, *Python* koristi biblioteke kao što su *torch* i *numpy*. *Torch* se koristi za interpolaciju mreže, dok *numpy* olakšava manipulaciju matricama.

**Matplotlib**: Za vizualizaciju podataka koristi se *matplotlib*, koja omogućuje generiranje grafova i toplotnih mapa.

**Argument Parser**: *Python* skripta koristi *argparse* za primanje ulaznog argumenta koji specificira putanju do datoteke s brojevima točaka.

**Postprocesiranje**: Skripta vrši rotaciju i horizontalno okretanje podataka kako bi se osigurala pravilna orijentacija konačne vizualizacije. Rezultirajući podaci se zatim interpoliraju na razlučivost od 1920 x 1080 piksela.

**Kod 3.** Python kod

main.py

```python
import torch
import numpy as np
import matplotlib.pyplot as plt
import argparse

# Set up argument parser
parser = argparse.ArgumentParser(description='Generate heatmap from point counts.')
parser.add_argument('file_path', type=str, help='Path to the point counts file')
args = parser.parse_args()

# Read the points from the file specified by the command-line argument
points_file_path = args.file_path
with open(points_file_path, 'r') as file:
    lines = file.readlines()
    point_counts = [[int(num) for num in line.split()] for line in lines]

# Rotate the grid by 90 degrees clockwise
def rotate_clockwise(matrix):
    return [list(reversed(col)) for col in zip(*matrix)]

# Flip the grid horizontally
def flip_horizontal(matrix):
    return [list(reversed(row)) for row in matrix]

rotated_grid_values = rotate_clockwise(point_counts)
flipped_grid_values = flip_horizontal(rotated_grid_values)

# Convert the flipped grid to a numpy array
```

Izvor: Autor

**Izvoz Rezultata**: Na kraju, generirana toplinska karta se pohranjuje kao PNG datoteka, koja se kasnije može otvoriti i pregledati.

Ovaj pristup omogućuje korištenje snage oba jezika: .NET za integraciju s bazom podataka i kontrolu tijeka aplikacije, te *Python* za naprednu obradu podataka i vizualizaciju.

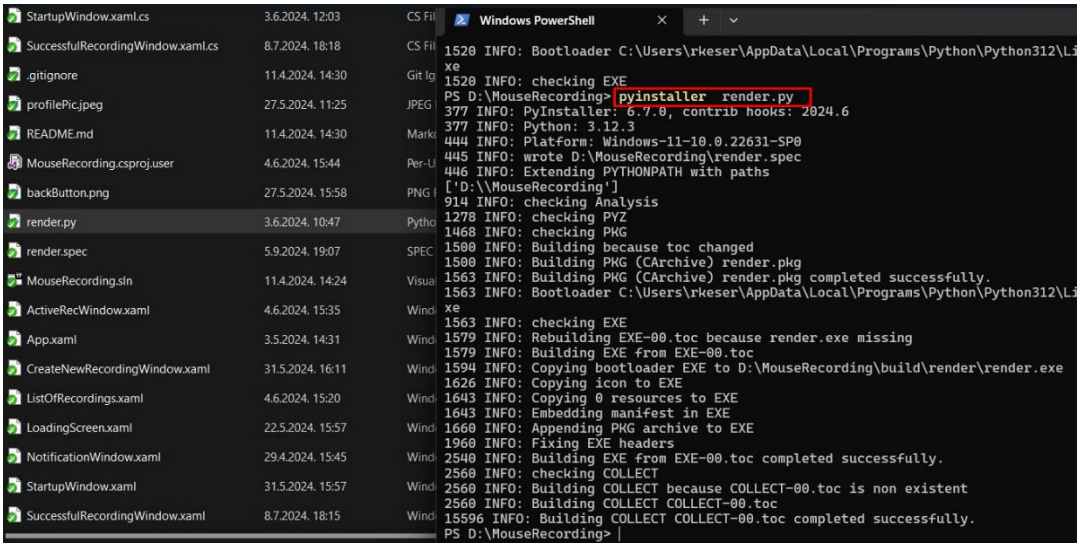
### 6.3. Prevedeni *Python* kod

Generiranje toplotne mape izvršava se pomoću *Python* skripte (kod 3). S obzirom da korisnik možda nema instalirani *Python* na svojem računalu, potrebno je pre-kompajlirati *Python* skriptu u izvršnu datoteku (engl. e*xecutable*). Tako aplikacija postaje samostalna i funkcionalna bez potrebe za instalacijom dodatnog softvera ili postavljanjem *Python* okruženja. Izvršna datoteka *Python* skripte je zatim spremljena u lokalni direktorij pod nazivom *render.exe*, slika 11.

#### 6.3.1. Izrada izvršne datoteke

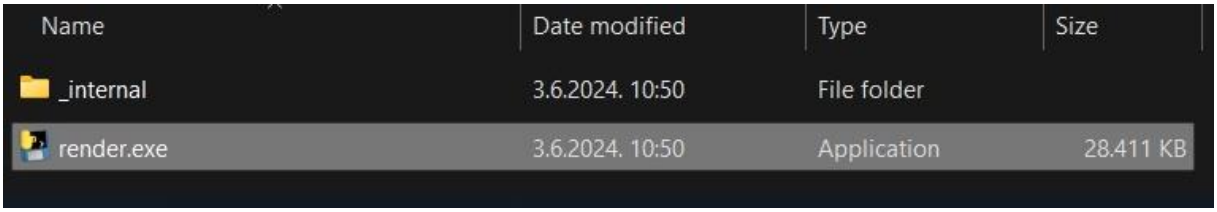
Izrada izvršne datoteke *render.exe* napravljena je pomoću alata *PyInstaller*. Ovaj alat omogućava pakiranje *Python* skripte i svih biblioteka i modula koji su korišteni u kodu. *PyInstaller* alat potrebno je instalirati pomoću alata *pip*. To je alat za instalaciju i upravljanje *Python* paketa (modula i biblioteka). Pip omogućuje korisnicima jednostavno preuzimanje i instalaciju dodatnih *Python* biblioteka iz *Python Package Indexa*, najvećeg repozitorija *Python* paketa.

**Slika 10.** Kreiranje izvršne datoteke




Izvor: Autor

**Slika 11.** Izvršna datoteka




Izvor: Autor

*PyInstaller* preuzima se pomoću komande `pip install pyinstaller`. *PyInstaller* spreman je za korištenje nakon uspješne instalacije, te se pokreće komanda za izradu izvršne datoteke `pyinstaller render.py`. Komanda se pokreće preko terminal konzole u istom direktoriju gdje i *Python* skripta.

Izvršna datoteka *render.exe* spremna je za korištenje nakon uspješnog kreiranja. Ona se zatim prebacuje u izvorni direktorij aplikacije koja je s ovim završnim korakom spremna za distribuciju.

# 7. GENERIRANJE TOPLOTNE MAPE

Generiranje toplotne mape glavna je inačica ove aplikacije. Korištenjem boja prikazuje se koncentracije aktivnosti računalnog miša u određenim dijelovima računalnog sučelja.

## 7.1. *Matplotlib* biblioteka

*Matplotlib* prvi je put razvio John D. Hunter 2003. godine. Biblioteka je prvobitno bila namijenjena za repliku MATLAB-ovih grafikona, kako bi inženjeri, developeri i istraživači koji su se odlučili na prijelaz iz MATLAB okoline na *Python* imali slične alate za vizualizaciju podataka. (Hunter 2007: 90-95)

S vremenom je *matplotlib* postao sve popularniji zbog svoje fleksibilnosti i sposobnosti da kreira visokokvalitetne grafike. Danas se široko koristi na akademijama, u industriji, podatkovnoj znanosti (engl. *data science*), inženjeringu, i mnogim drugim sektorima.

*Matplotlib* je izuzetno važan alat zbog svoje fleksibilnosti, kompatibilnosti i prirode otvorenog koda (engl. open source code). Fleksibilnost *matplotliba* omogućuje korisnicima da prilagode svaki aspekt svojih grafikona, uključujući boje, fontove, stilove linija i oznake, čime se

osigurava potpuna kontrola nad vizualizacijom podataka. Pored toga, biblioteka je kompatibilna sa širokim spektrom *Python* paketa, kao što su *NumPy, Pandas, SciPy* i *PyTorch*, što omogućava jednostavnu integraciju vizualizacije u različite analitičke tokove rada. Konačno, kao projekt otvorenog koda (engl. *open source*), *matplotlib* ima snažnu zajednicu korisnika i programera koji kontinuirano unapređuju biblioteku, dodajući nove funkcionalnosti i ispravljajući greške, što dodatno doprinosi njenoj popularnosti i širokoj primjeni.

Upravo zbog svoje jednostavnosti, u ovom se projektu koristi *Matplotlib* biblioteka za vizualizaciju toplotne mape, slika 12. Ova biblioteka jedna je od najpopularnijih u *Pythonu* za kreiranje raznih grafika i vizualizacija, uključujući histograme, 3D grafike, linijske grafikone te toplotne mape.

### 7.2. Funkcije *matplotlib* biblioteke za generiranje toplotne mape

Iz numeričkih podataka spremljenih u tekstualnu datoteku generira se toplotna mapa koja predstavlja vrijednost matrice u različitim bojama. Za vizualizaciju u ovoj aplikaciji koriste se nijanse crvene i ljubičaste boje te crna boja. Detalji o svakom koraku u ovom procesu objašnjeni su u nastavku.

**Kod 4.** Uvoz potrebnih podula

```
Import matplotlib.pyplot as plt
```

Izvor: Autor

Ova linija uvozi „*pyplot“* modul iz *Matplotlib* biblioteke, koji pruža jednostavno sučelje za kreiranje grafikona i drugih vizualizacija.

**Kod 5.** Kreiranje figure

```
plt.figure(figsize=(16, 9))
```

Izvor: Autor

Ova funkcija kreira novu figuru s dimenzijama 16 x 9 inča. Figura predstavlja grafičku ploču na koju se postavljaju grafički elementi poput grafikona, osi, naslova itd. Veličina figure određuje krajnju veličinu slike toplotne mape.

**Kod 6.** Prikazivanje toplotne mape

```
plt.imshow(resized_grid, cmap='inferno', interpolation='nearest')
```

Izvor: Autor

Funkcija „*imshow“* koristi se za prikazivanje dvodimenzionalne matrice kao slike, prikazano u slici 9. U ovom slučaju, „*resized_grid“* predstavlja interpolirane podatke. Argument *„cmap='inferno'“* određuje da će se koristiti toplotna mapa „inferno“, koja prikazuje vrijednosti od tamnih (niže vrijednosti) do svjetlijih tonova (više vrijednosti), pri čemu tamni tonovi predstavljaju niže vrijednosti, a svjetliji više vrijednosti. „*interpolation='nearest'“* znači da će svaki piksel biti prikazan bez dodatnog izračunavanja između susjednih vrijednosti, čime se osigurava jasan prikaz.

**Kod 7.** Dodavanje bočne trake

```
plt.colorbar()
```

Izvor: Autor

Ova funkcija dodaje bočnu traku (engl. *colorbar*) koja prikazuje skalu boja korištenih na toplotnoj mapi. Ona korisnicima omogućuje da razjasne određene boje te da ih povežu s numeričkim vrijednostima iz matrice.

**Kod 8.** Postavljanje naslova osi

```
plt.title('Interpolated Heatmap of grid values (1920x1080)')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
```

Izvor: Autor

Ove linije postavljaju naslov grafikona „Interpolated Heatmap of Grid Values 1920 x 1080“, kao i oznake X i Y osi, nazvane „*X-axis“* i „*Y-axis*“

**Kod 9.** Spremanje podataka

```
#Save the heatmap to a file
Output_file = 'heatmap.png'
plt.savefig(output_file)
```

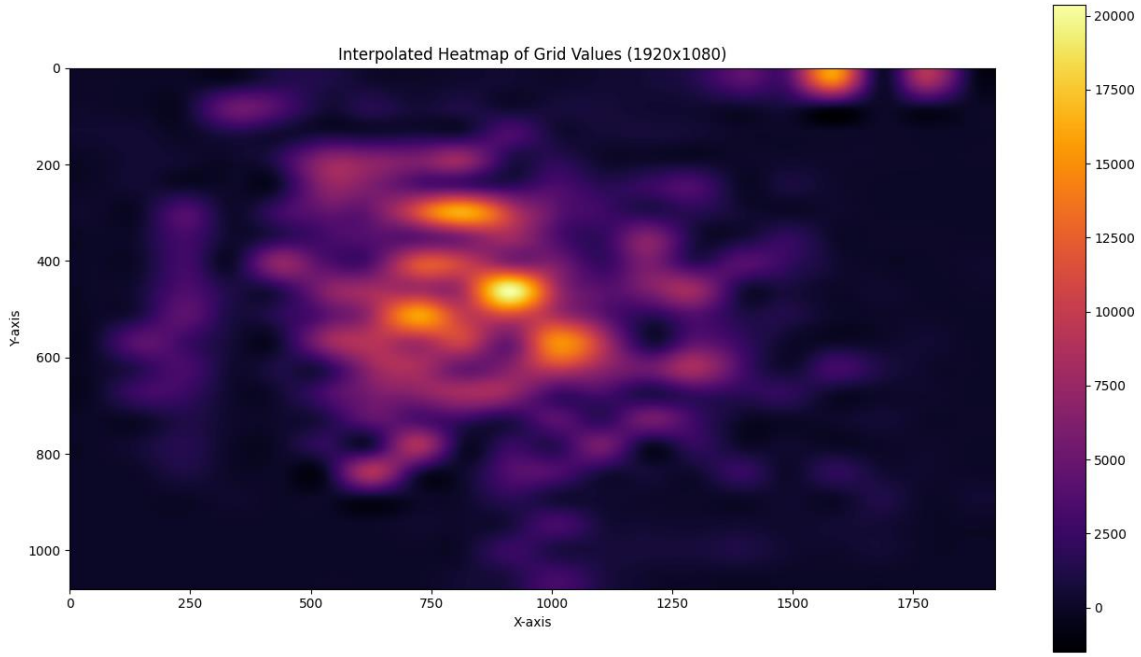
Izvor: Autor

Na kraju, nakon što je grafikon kreiran, funkcija *savefig* omogućava spremanje podataka kao fotografije (*.png*), pod imenom 'heatmap.png'. Fotografija grafikona pohranjuje se u lokalnom direktoriju pokrenute aplikacije. Iako je potrebno dati ime datoteci „*heatmap.png“*, ona se u

procesu pohranjivanja podataka od strane .NET-a preimenuje u naslov kojeg je korisnik zadao prilikom početka snimanja, prikazano u slici 2.

### 7.3. Finalni prikaz generirane toplotne mape

**Slika 12.** Generirana toplotna mapa




Izvor: Autor

Slika 12. prikazuje generiranu toplotnu mapu nastalu tijekom višesatnog rada u programu *Visual Studio Code*.[3] Na prvi pogled, graf se može činiti kao skup nasumičnih boja, no detaljnijom analizom moguće je uočiti jasnu povezanost između boja na mapi i elemenata sučelja koje je korisnik koristio tijekom rada.

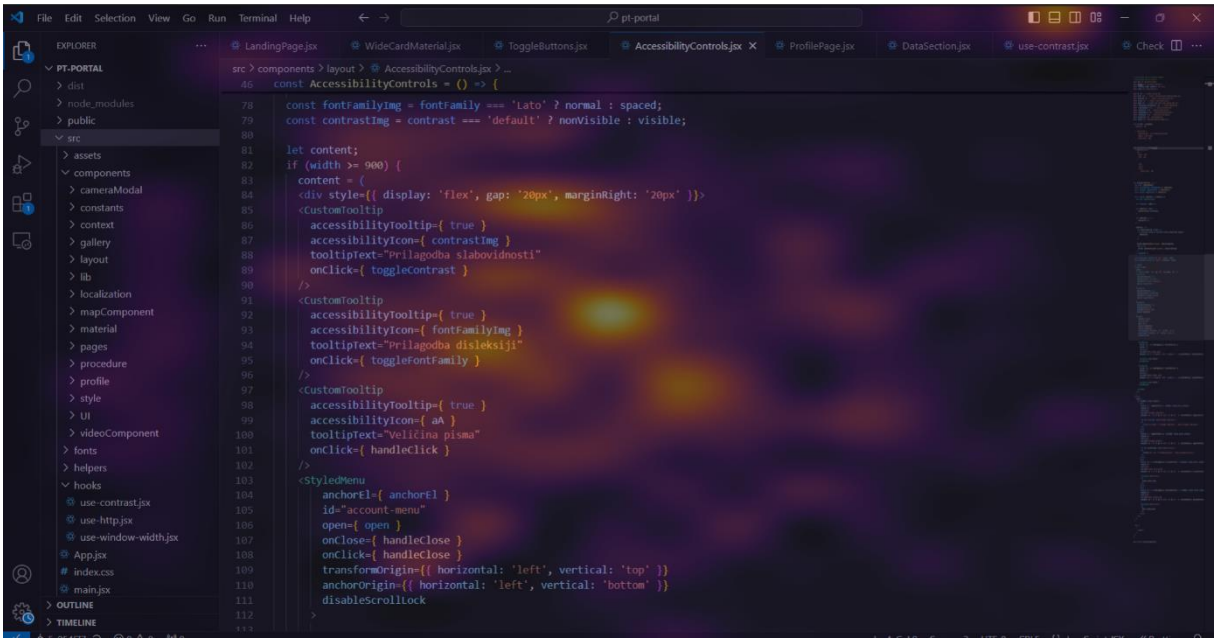
Primjerice, područja na toplotnoj mapi koja su najintenzivnije obojena odgovaraju lokacijama na ekranu gdje su se odvijale ključne aktivnosti korisnika, poput pisanja koda, navigacije kroz datoteke (lijevi izbornik), korištenja terminala, te u gornjem desnom kutu korištenje funkcija za podešavanje responzivnosti prozora, zatvaranje i smanjivanje prozora *Visual Studio Code-a*. To su područja na kojima je računalni miš najčešće bio pozicioniran ili gdje je korisnik najviše puta kliknuo mišem. Na ovaj način toplotna mapa pruža vizualni prikaz korisnikovih interakcija

[3]*Visual Studio Code* je alat za programiranje i pisanje koda.

s *Visual Studio Code-om*, otkrivajući obrasce u njegovom radu, poput često korištenih dijelova sučelja ili navigacijskih putanja.

Ako se napravi preslika ove generirane toplotne mape sa sučeljem *Visual Studio Code-a*, jasno se vidi koji su sektori sučelja najaktivniji. Ova usporedba dodatno potvrđuje korisnost toplotnih mapa u analizi korisničkog ponašanja, omogućujući detaljan uvid u to kako korisnik koristi određeni softver.

**Slika 13.** Vizualna korelacija između toplotne mape s Visual Studio Code sučeljem

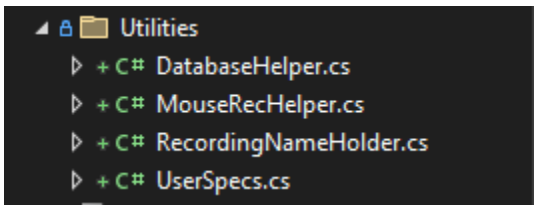



Izvor: Autor

## 8. POMOĆNE KLASE

Pomoćne klase (engl. *helper classes*) prikazane na slici 14, klase su koje pružaju dodatne funkcionalnosti i olakšavaju rad s određenim komponentama aplikacije.

**Slika 14.** Pomoćne klase unutar mape 'Utilities'




Izvor: Autor

Koriste se za enkapsuliranje zajedničke logike koja se ponavlja u više dijelova aplikacije, čime se smanjuje duplikacija ili redundantnost koda te se poboljšava čitljivost i održivost aplikacije. Pomoćne klase doprinose boljem tijeku rada (engl. *workflow*) jer omogućuju modularnost i ponovno korištenje koda, što ubrzava razvoj i olakšava održavanje.

### 8.1. 'Database helper' pomoćna klasa

Jedna od bitnih pomoćnih klasa u ovoj aplikaciji je klasa *DatabaseHelper*, koja upravlja svim aspektima interakcije s bazom podataka *SQLite*, slika 15. Ova klasa omogućuje inicijalizaciju baze podataka kod umetanje podataka (kod 12), te dohvaćanje podataka iz baze, čime se centralizira i pojednostavljuje rad s bazom podataka.

### 8.1.2. Inicijalizacija baze podataka

Inicijalizacija baze podataka obavezna je kako bi se osiguralo postojanje svih potrebnih struktura za pohranu podataka prije nego što aplikacija pokuša pristupiti ili pohraniti bilo kakve informacije. Ovaj proces kreira datoteku baze podataka i potrebne tablice unutar nje, u slučaju da one već ne postoje, čime se omogućuje ispravan rad aplikacije. Prvi je korak u ovoj metodi provjera postoji li već datoteka baze podataka na predviđenoj lokaciji (*DatabaseFilePath*). Ako datoteka ne postoji, metoda nastavlja s njenim kreiranjem. Ako datoteka baze ne postoji, koristi se metoda *SQLiteConnection.CreateFile* za kreiranje nove *SQLite* datoteke baze podataka na definiranoj lokaciji. Lokalna datoteka baze podataka nazvana je *mouse_coordinates*.db. Nakon što je datoteka baze kreirana, metoda otvara vezu s bazom koristeći *SQLiteConnection*.

**Kod 10.** Inicijalizacija baze podataka

```
1 reference
public static void InitializeDatabase()
{
    if (!File.Exists(DatabaseFilePath))
    {
        SQLiteConnection.CreateFile(DatabaseFilePath);

        using (var connection = new SQLiteConnection($"Data Source={DatabaseFileName};Version=3;"))
        {
            connection.Open();

            using (var command = new SQLiteCommand(connection))
            {
                command.CommandText = @"
                CREATE TABLE mouse_coordinates (
                    record_name TEXT,
                    coordinates BLOB
                )";
                command.ExecuteNonQuery();
            }
        }
    }
}
```

Izvor: Autor

Ova veza omogućuje izvršavanje SQL naredbi za kreiranje tablica i manipulaciju podacima unutar baze. Unutar otvorene veze metoda koristi *SQLiteCommand* za kreiranje SQL naredbe koja stvara tablicu *mouse_coordinates* s dva stupca: *record_name* (tekst) i *coordinates* (BLOB). U ovu tablicu pohranjuju se imena snimaka i koordinate miša. Konačno, SQL naredba izvršava se pomoću *ExecuteNonQuery*, što rezultira kreiranjem tablice unutar baze podataka ako ona već ne postoji. Time je baza podataka spremna za pohranu podataka o snimljenim koordinatama miša.

### 8.1.3. BLOB tip podataka

BLOB (engl. *Binary Large Object*) tip je podatka koji omogućuje pohranu velikih količina binarnih podataka u bazi podataka. Ovaj tip koristi se za spremanje podataka koji nisu u tekstualnom formatu, poput slika, videozapisa, zvuka ili, u ovom slučaju, koordinata miša. Za razliku od tekstualnih podataka, BLOB pohranjuje sirove binarne podatke, omogućujući tako fleksibilnu pohranu složenih ili velikih datoteka unutar baze podataka.

U ovoj aplikaciji podaci o koordinatama miša prikupljaju se svakih 300 milisekundi. S obzirom na to da se podaci prikupljaju tako često, brzo se akumulira velika količina informacija. Kako bi se te koordinate pohranile na učinkovit način, koristi se BLOB tip podatka, koji omogućuje spremanje koordinata kao binarnih podataka. Ti podaci se zatim serijaliziraju pomoću metode *SerializeCoordinatesList* (kod 11).

### 8.1.4. Serijalizacija podataka

Metoda *SerializeCoordinatesList* serijalizira listu koordinata miša u binarni format koji se može pohraniti u BLOB polje baze podataka. Ova metoda radi na način da prvo kreira memorijski tok (engl. *memory stream),* koji se koristi za privremeno pohranjivanje podataka prije nego što se pretvori u niz bitova. Za privremeno pohranjivanje podataka koristi se radna memorija (engl. *RAM – Random Access Memory*), koja omogućuje brzo zapisivanje i čitanje podataka.

Zbog činjenice da se podaci o koordinatama miša prikupljaju svakih 300 milisekundi, program akumulira veliku količinu podataka u vrlo kratkom vremenu. Svaka koordinata se pohranjuje u memorijski tok, što znači da se u RAM-u pohranjuje niz bitova za svaki pokret miša. S obzirom na visoku frekvenciju prikupljanja podataka i količinu koordinata koje se obrađuju, program

troši značajnu količinu memorije. Dakle, može se zaključiti da je ovaj program '*memory heavy*'[4] jer koristi RAM memoriju za kontinuiranu obradu i privremeno pohranjivanje velike količine podataka prije nego što ih trajno spremi u bazu podataka.

**Kod 11.** Serijalizacija koordinata računalnog miša

```
// Method to serialize coordinates into byte array
1 reference
private static byte[] SerializeCoordinatesList(List<(int, int)> coordinatesList)
{
    using (MemoryStream ms = new MemoryStream())
    {
        using (BinaryWriter writer = new BinaryWriter(ms))
        {
            // Write the number of coordinates to the stream
            writer.Write(coordinatesList.Count);

            // Write each coordinate pair to the stream
            foreach (var (x, y) in coordinatesList)
            {
                writer.Write(x);
                writer.Write(y);
            }
        }
        // Return the byte array containing serialized coordinates
        return ms.ToArray();
    }
}
```

Izvor: Autor

Metodu *InsertMouseCoordinates* okida metoda *StopMouseTimer* (kod 13). Metoda *StopMouseTimer* ima nekoliko ključnih funkcionalnosti:

a) zaustavljanje *timera*. Metoda prvo zaustavlja *_mouseTimer*, koji se koristi za periodično prikupljanje koordinata miša tijekom snimanja. Ako je *timer* aktivan, metoda *Stop* zaustavlja njegovo daljnje izvođenje.

b) preuzimanje naziva snimke. Ime snimke preuzima se iz tekstualnog polja *recordNameTextbox*. Ovo ime se koristi za identifikaciju snimke u bazi podataka.

c) umetanje koordinata u bazu pomoću *DatabaseHelper* klase. Poziva se metoda *InsertMouseCoordinates* iz *DatabaseHelper* klase, koja ubacuje trenutno zabilježene

[4]'Memory heavy' su programi koji koriste veliki postotak resursa, radne memorije ili računalnog procesa.

koordinate miša (*_recordedCoordinates*) u bazu podataka pod odgovarajućim nazivom snimke.

d) postavljanje trenutnog naziva snimke. Trenutni naziv snimke postavlja se u *RecordingNameHolder.CurrentRecordingName*. Ovaj naziv koristan je za daljnje operacije i referenciranje trenutne snimke u aplikaciji.

Ova metoda osigurava da su svi podaci o kretanju miša pravilno pohranjeni u bazu podataka nakon zaustavljanja snimanja, omogućujući kasniji pristup i analizu tih podataka.

**Kod 12.** Metoda za upis koordinata računalnog miša

```
1 reference
public static void InsertMouseCoordinates(string recordName, List<(int, int)> coordinatesList)
{
    using (var connection = new SQLiteConnection($"Data Source={DatabaseFileName};Version=3;"))
    {
        connection.Open();

        using (var command = new SQLiteCommand(connection))
        {
            // Serialize coordinates into a single byte array
            byte[] coordinatesBytes = SerializeCoordinatesList(coordinatesList);

            command.CommandText = "INSERT INTO mouse_coordinates (record_name, coordinates) VALUES (@recordName, @coordinates)";
            command.Parameters.AddWithValue("@recordName", recordName);
            command.Parameters.AddWithValue("@coordinates", coordinatesBytes);

            command.ExecuteNonQuery();
        }
    }
}
```

Izvor: Autor

**Kod 13.** Metoda za stopiranje aktivnog snimanja

```
/// <summary>
/// Stops the mouse recording timer and saves the recorded coordinates to the database.
/// </summary>
1 reference
public void StopMouseTimer()
{
    _mouseTimer?.Stop();
    string recordName = recordNameTextbox.Text;
    DatabaseHelper.InsertMouseCoordinates(recordName, _recordedCoordinates);
    RecordingNameHolder.CurrentRecordingName = recordName;
}
```

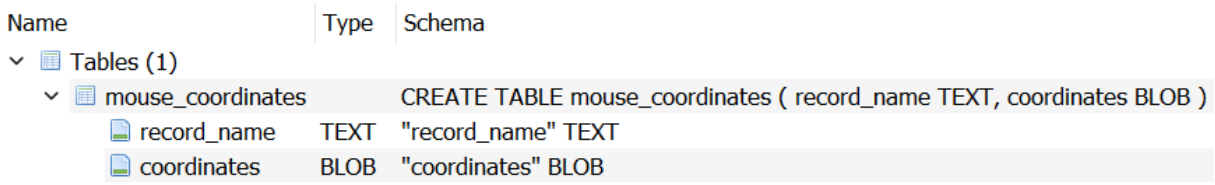
Izvor: Autor

### 8.1.6. Struktura baze podataka

Struktura je baze podataka u ovoj aplikaciji jednostavna. Sastoji se od jedne tablice naziva *mouse_coordinates*, koja ima dva stupca:

- *record_name* (TEXT): naziv sesije snimanja pokreta računalnog miša.
- *coordinates* (BLOB): binarni podaci koji predstavljaju serijalizirane koordinate pokreta miša.

**Slika 15.** Struktura baze podataka




Izvor: Autor

**Slika 16.** Lokalna baza podataka s nekoliko zapisa

| | record_name | coordinates |
|---|---|---|
| | Filter | Filter |
| 1 | test_snimanje | BLOB |
| 2 | test_dva_monitora | BLOB |
| 3 | snimanje_internet | BLOB |
| 4 | posao jutarnja smjena | BLOB |
| 5 | testing123 | BLOB |
| 6 | microsoft office rec | BLOB |
| 7 | first record test | BLOB |

Izvor: Autor

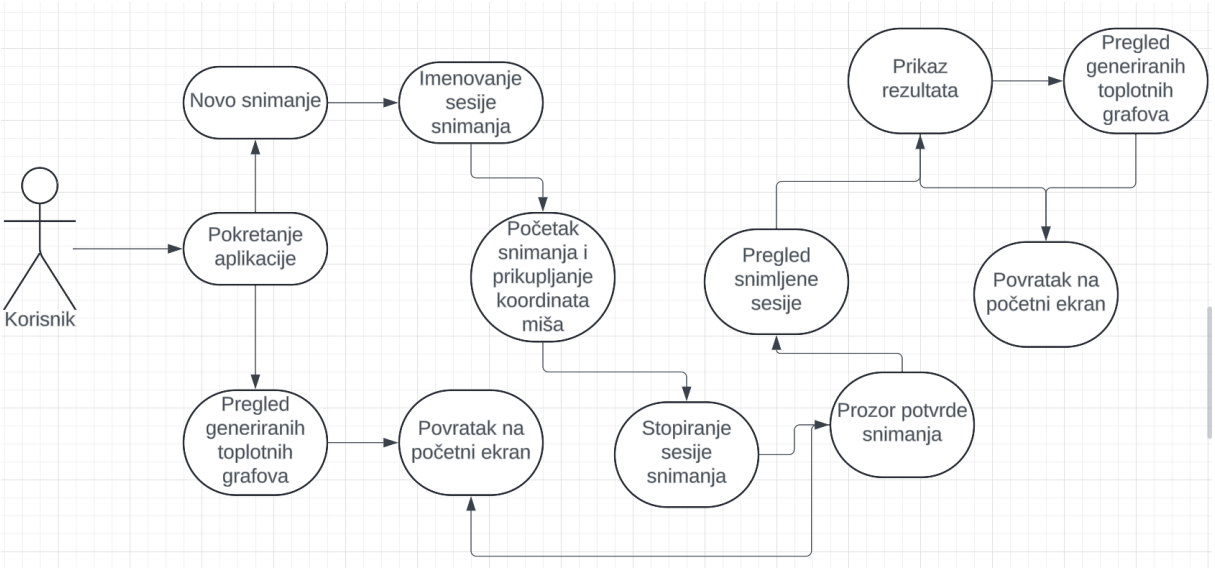
# 9. DIJAGRAM PROCESA

Dijagram procesa predstavlja vizualni prikaz toka aktivnosti unutar određenog sustava ili projekta. Cilj je ovog dijagrama omogućiti jasnije razumijevanje koraka koji se poduzimaju kako bi se postigao određeni rezultat, identificiranje ključnih faza u procesu, te prepoznavanje potencijalnih problema ili uskih grla u sustavu. Simboličkim prikazom različitih aktivnosti, odluka i tokova informacija, dijagram procesa omogućuje jednostavno praćenje i analizu svakog segmenta rada.

## 9.1. UML dijagram

Za ovaj je projekt dijagram procesa osmišljen u svrhu ilustracije koraka u razvoju i korištenju aplikacije, od početne interakcije korisnika do završnih akcija kao što su prikaz i analiza rezultata. Posebna pažnja posvećena je jasnom prikazu svih faza, uključujući i prijelaze između njih, kako bi se osiguralo da su svi aspekti sustava dobro definirani i lako razumljivi. Dijagram procesa služi kao temelj za razumijevanje logike iza implementiranih rješenja i omogućava lakše identificiranje mogućnosti za optimizaciju ili unapređenje sustava.

**Slika 17.** UML Dijagram

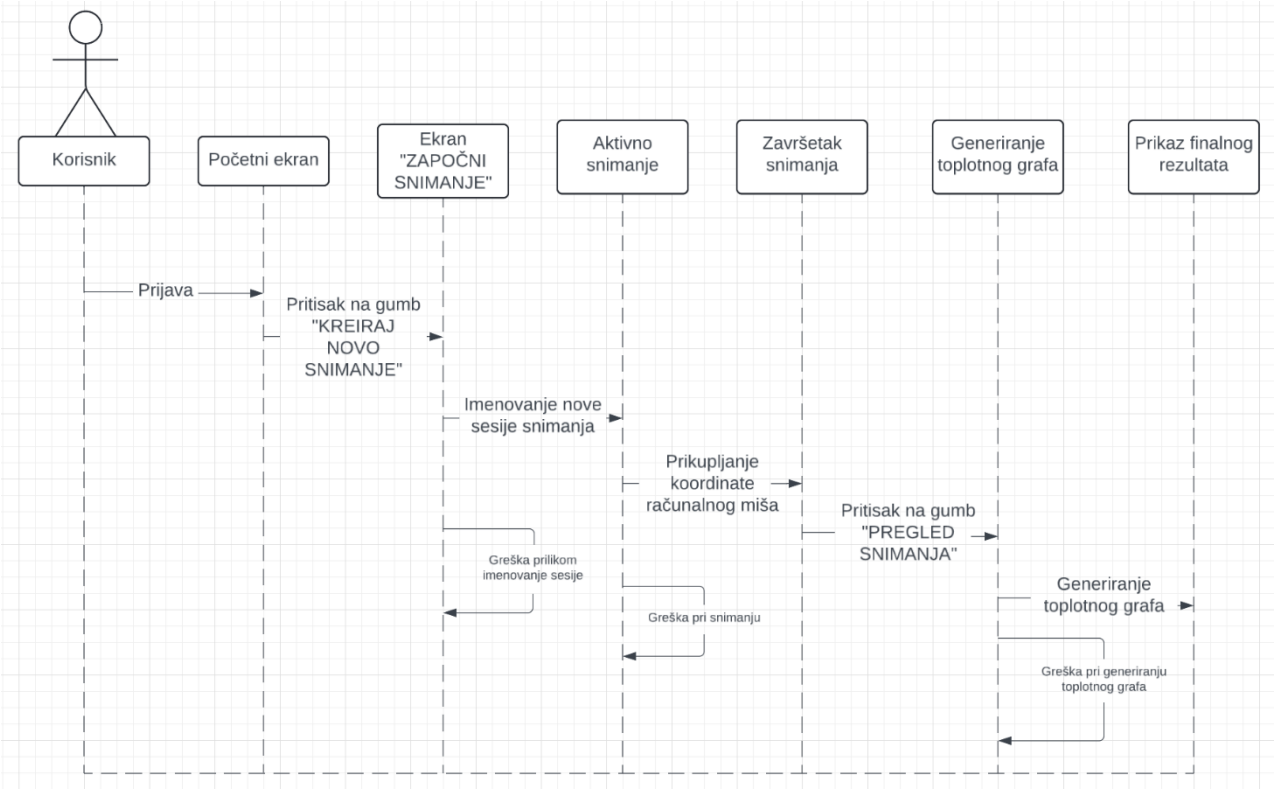



Izvor: Autor

### 9.2. Dijagram slijeda operacija

Dijagram slijeda operacija u ovom projektu prikazuje cjelokupni tok aktivnosti unutar aplikacije, počevši od prijave korisnika pa sve do prikaza finalnog rezultata. Ovaj dijagram služi kao vizualni vodič kroz sve bitne korake koje aplikacija provodi, omogućujući pregled cjelokupnog procesa korištenja aplikacije i interakcije korisnika s njom.

**Slika 18.** Dijagram slijeda operacija




Izvor: Autor

Korištenje aplikacije započinje dolaskom na početni ekran. Na početnom ekranu korisniku je dostupna opcija za započinjanje nove sesije snimanja ili pregled prijašnjih snimanja. Odabirom započinjanja uključuje imenovanje nove sesije. Ako se pojavi greška prilikom imenovanja sesije, korisniku je omogućeno vraćanje na prethodni korak kako bi pokušao ponovno.

Nakon uspješnog imenovanja korisnik prelazi na ekran aktivnog snimanja, gdje se aktivira prozor s indikatorom snimanja te se bilježe koordinate kretanja računalnog miša. U slučaju pogreške tijekom snimanja, aplikacija omogućuje povratak i ponovni pokušaj.

Po završetku snimanja aplikacija automatski prelazi na novi ekran s mogućnošću pregleda snimljene sesije ili povratak na početni izbornik. Prilikom pritiska gumba *Pregled snimanja*, aplikacija automatski prelazi na generiranje toplinskog grafa, koji vizualizira podatke

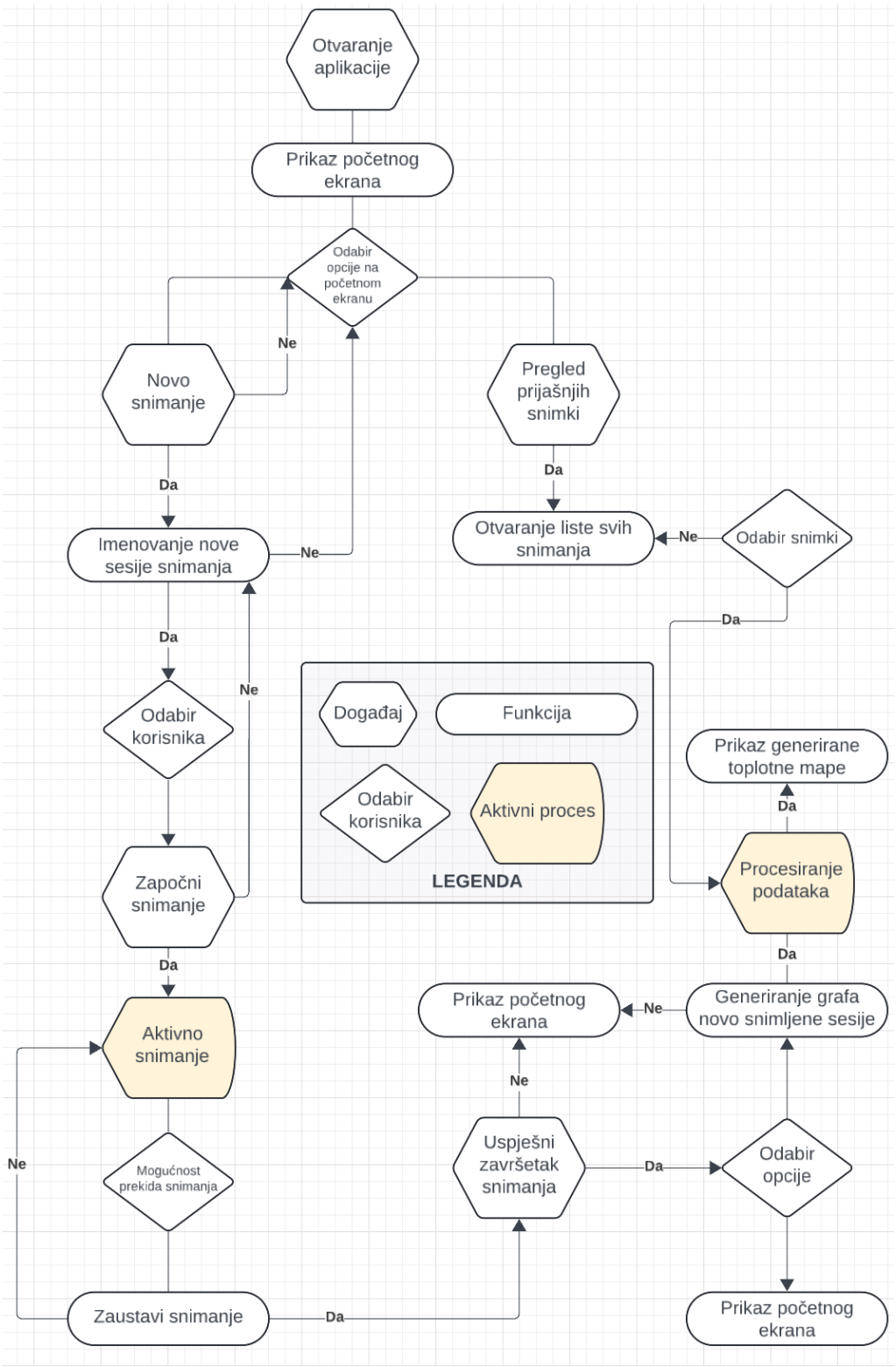
prikupljene tijekom snimanja. Ako dođe do greške u generiranju grafa, korisniku je pružena opcija povratka na prethodni korak kako bi pokušao ponovno generirati graf. Uspješno generirani toplinski graf prikazuje se kao konačan rezultat u obliku fotografije u *.png* formatu, čime se zaokružuje cijeli proces korištenja aplikacije.

Ovaj dijagram slijeda operacija osigurava da su svi koraci jasno definirani i prati interakcije između korisnika i aplikacije, čineći proces razumljivim i učinkovitim.

### 9.3. EPC dijagram

EPC dijagram (engl. *Event-Driven Process Chain*) jedan je od najvažnijih alata za modeliranje poslovnih procesa, a koristi se za vizualizaciju tijeka događaja i aktivnosti unutar sustava. EPC dijagrami omogućuju detaljan prikaz logičkog slijeda događaja i aktivnosti, prikazujući kako se pojedini procesi aktiviraju i međusobno povezuju. U kontekstu ove aplikacije EPC dijagram služi kao sredstvo za razumijevanje i optimizaciju tijeka operacija, osiguravajući da svaki korak u procesu ima jasno definirane ulaze i izlaze te logičke veze između njih, slika 22.

**Slika 19.** EPC dijagram




Izvor: Autor

# 10. EVALUACIJA I PERFORMANSE APLIKACIJE

Testiranje aplikacija postalo je ključan korak u razvoju softverskih rješenja u industriji, osiguravajući da proizvodi ispunjavaju visoke standarde kvalitete i funkcionalnosti. U današnjem dinamičnom poslovnom okruženju, gdje tehnološka rješenja moraju biti brza, pouzdana i sigurna, kvalitetno testiranje nije samo opcija, već nužnost. Industrija se sve više oslanja na sofisticirane aplikacije koje upravljaju složenim procesima, optimiziraju proizvodnju, poboljšavaju učinkovitost i omogućuju donošenje informiranih odluka. Svaka greška ili manjak performansi u tim aplikacijama može imati ozbiljne posljedice, uključujući financijske gubitke, narušavanje ugleda tvrtke ili prekide u proizvodnji.

Na primjer, učitavanje i procesiranje ne smije trajati dulje od nekoliko sekundi jer svaki dulji period odmah narušava korisničko iskustvo i može rezultirati padom produktivnosti te frustracijom korisnika. U industrijskim aplikacijama, gdje je brzina reakcije vrlo bitna, čak i kratka odgoda može uzrokovati ozbiljne probleme u radu i izvršenju zadataka. Zbog toga je važno pažljivo testirati sve aspekte performansi aplikacije, uključujući i vrijeme učitavanja, kako bi se osiguralo da aplikacija radi glatko i učinkovito u svim uvjetima.

Testiranje aplikacija u industriji obuhvaća širok spektar aktivnosti, od jednostavnih funkcionalnih testova do složenih stres-testova koji simuliraju stvarne uvjete rada. Cilj je identificirati i ispraviti sve potencijalne probleme prije nego što aplikacija dođe do krajnjih korisnika. Ovo uključuje testiranje različitih aspekata aplikacije, kao što su brzina, pouzdanost, sigurnost i korisničko iskustvo. Osim toga, testiranje pomaže u optimizaciji performansi aplikacije, čineći je efikasnijom i prilagodljivijom različitim uvjetima rada.

## 10.1. Testiranje aplikacije

Testiranje ove aplikacije provedeno je u nizu faza s ciljem osiguravanja stabilnosti, funkcionalnosti i optimalne performanse u stvarnim uvjetima korištenja. Proces testiranja uključivao je različite metode, uključujući funkcionalno testiranje, testiranje performansi, testiranje korisničkog sučelja te testiranje opterećenja (engl. *load testing*).

### 10.1.2. Funkcionalno testiranje

Na početku razvoja aplikacija je podvrgnuta temeljitom funkcionalnom testiranju kako bi se osiguralo da svi njezini moduli rade prema očekivanjima. Tijekom testiranja pažljivo su provjerene sve osnovne funkcionalnosti, poput navigacije između ekrana, ispravnosti prikaza

podataka, te mogućnosti unosa i obrade korisničkih podataka. Svaka bazična nepravilnost ove aplikacije je ispravljena, čime se osigurala stabilna osnova za daljnji razvoj.

### 10.1.3. Testiranje performansi

Kako bi se osiguralo da aplikacija može podnijeti očekivana opterećenja bez pada performansi, izvedeno je opsežno testiranje performansi. U ovoj je fazi posebna pozornost posvećena vremenu učitavanja pojedinih ekrana, s naglaskom na osiguranje da učitavanje ekrana ne traje dulje od nekoliko sekundi.

Tijekom testiranja primijećeno je da inicijalno vrijeme učitavanja nije zadovoljavalo postavljene standarde, zbog čega su provedene optimizacije u načinu učitavanja podataka i smanjivanju složenosti prikaza. Ove optimizacije uključivale su poboljšanje načina na koji aplikacija dohvaća i prikazuje podatke te smanjenje broja zahtjeva prema poslužitelju, što je značajno ubrzalo vrijeme reakcije aplikacije, slika 20.

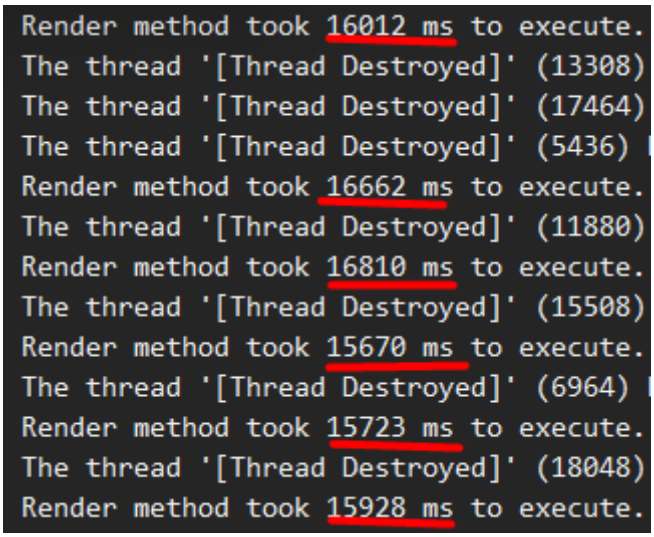
Kako bi se analizirao kod, koristi se štoperica (engl. s*topwatch*) kako bi se pratilo koliko vremena određena metoda troši na izvršavanje. U *C#* jeziku, *Stopwatch* iz *System.Diagnostics* klase omogućava precizno mjerenje proteklog vremena (kod 4).

**Kod 4.** Integriranje štoperice u metodu *Render*

```
public void Render(string currentRecordingName)
{
    Stopwatch stopwatch = new Stopwatch();
    // Start measuring time
    stopwatch.Start();

    int numCellsX = 20;
    int numCellsY = 20;

    try
    {
        DataTable mouseCoordinates = DatabaseHelper.GetMouseCoordinates(currentRecordingName);
        List<(int, int)> recordedCoordinates = ExtractCoordinates(mouseCoordinates);

        string filePath = $"{currentRecordingName}.txt";
        int[,] counts = CalculatePointCounts(recordedCoordinates, numCellsX, numCellsY);
        WritePointCountsToFile(counts, filePath);

        ExecutePythonScript(filePath);
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error during rendering: {ex.Message}");
    }
    finally
    {
        // Stop the stopwatch and log the elapsed time
        stopwatch.Stop();
        Debug.WriteLine($"Render method took {stopwatch.ElapsedMilliseconds} ms to execute.");
    }
}
```

Izvor: Autor

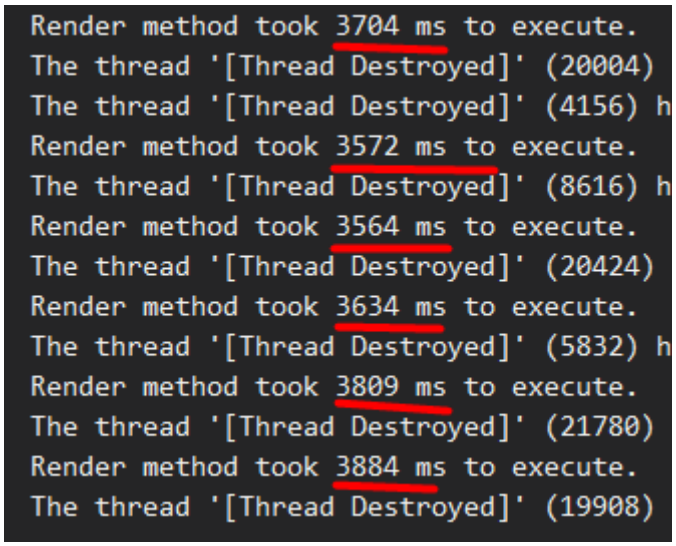
**Slika 20.** Vremenski period izvršavanja *Render* metode na neoptimiziranom kodu




Izvor: Autor

Slika 21 prikazuje vremenski period potreban za generiranje nekoliko različitih toplotnih mapa s optimiziranim kodom i dohvaćanjem koordinata iz radne memorije. Generiranje toplotne mape traje oko 3 sekunde, ovisno o količini snimljenih podataka. Nasuprot tome, slika 20 prikazuje generiranje istih toplotnih mapa, gdje se dohvaćanje koordinata vrši direktno iz baze podatka što produljuje procesiranje za otprilike 15 sekundi.

**Slika 21.** Vremenski period generiranja toplotne mape nakon optimizacije koda




Izvor: Autor

### 10.1.4. Optimizacija dohvaćanja podataka

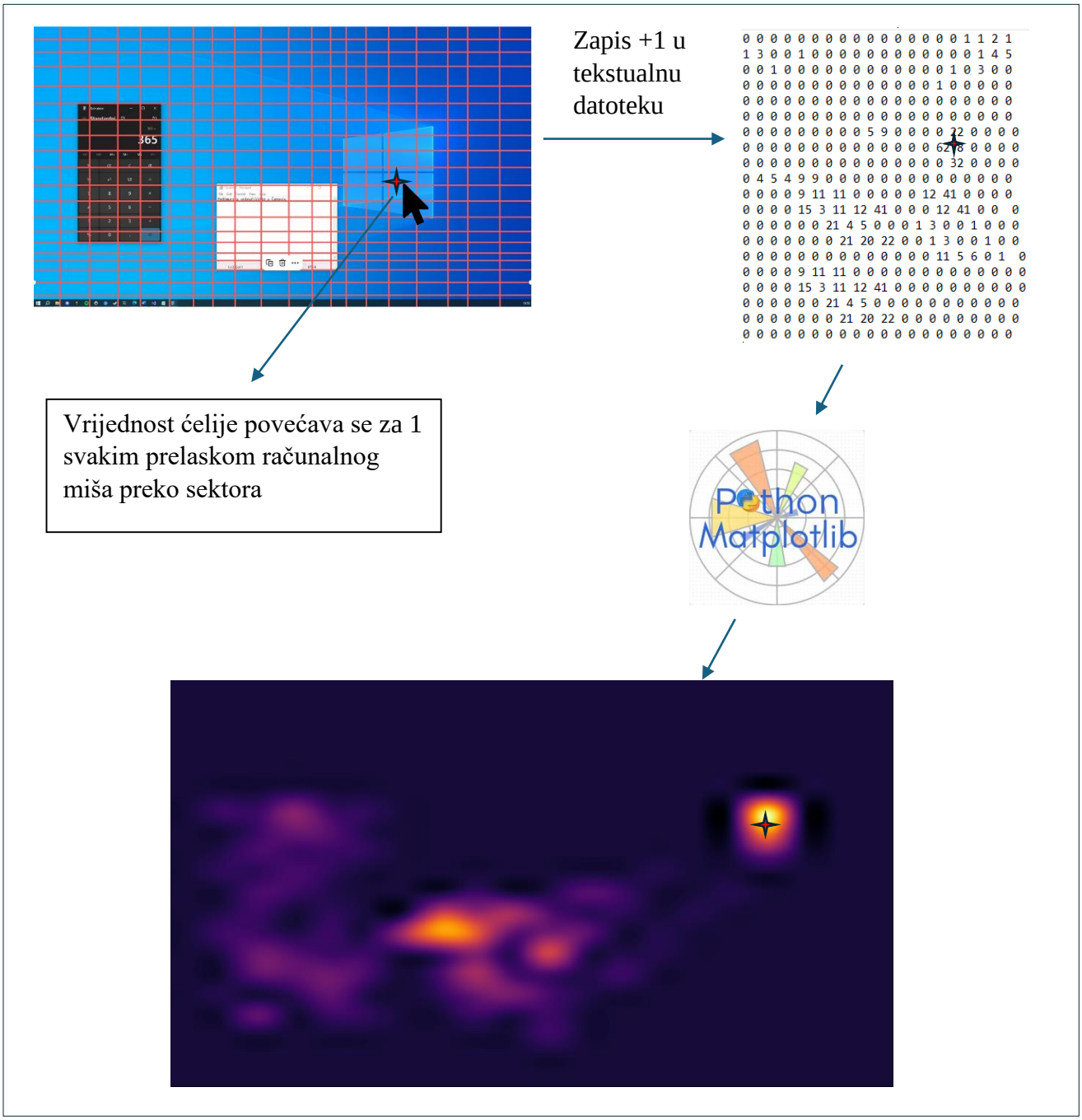
Veliki izazov u razvoju aplikacije bio je brzina dohvaćanja koordinata, koje su se u početnim fazama razvoja aplikacije dohvaćale direktno iz baze podataka. Ovaj se pristup pokazao sporim, posebno kada bi korisnik pokušao učitati složene ili velike mape koje su zahtijevale mnogo koordinata. Kao rješenje, implementirana je optimizacija gdje se, prije generiranja toplotne mape, koordinate privremeno pohranjuju u memoriju. Zahvaljujući brzini pristupa podacima iz memorije, vrijeme učitavanja ekrana značajno je skraćeno, što je rezultiralo fluidnijim korisničkim iskustvom. Ova optimizacija ne samo da je smanjila vrijeme reakcije aplikacije već je i omogućila brže generiranje i prikaz slike generiranog grafa, što je bitno za aplikacije koje zahtijevaju visoke performanse.

### 10.1.5. Korištenje tekstualne datoteke za renderiranje toplotne mape

Testiranje je pokazalo kako je, umjesto da se svaka koordinata zapisuje u bazu, brže svakoj ćeliji se vrijednost poveća za 1, čime se bilježi intenzitet interakcije na ekranu.

U ovom testiranju namjerno se fokusiralo na jednu točku na ekranu; kratko su se koristili kalkulator i blok za pisanje, a nakon toga se računalnim mišem pomicalo samo po manjem desnom kraju monitora. Gornja desna ćelija (označena zvjezdicom) primila je oko 60 prijelaza računalnog miša, dok je nekoliko susjednih ćelija imalo manji broj iteracija, što je rezultiralo crvenim oblakom na toplinskom grafu.

**Slika 22.** Slijed izrade toplotne mape




Izvor: Autor

## 11. NEDOSTACI I BUDUĆE NADOGRADNJE

Svaka aplikacija, ima svoje nedostatke i ograničenja. U ovom poglavlju rada analizirat će se trenutne slabosti aplikacije te identificirati područja koja bi mogla biti unaprijeđena. Nadalje, raspravljat će se o potencijalnim nadogradnjama koje bi mogle dodatno poboljšati

funkcionalnost, učinkovitost i korisničko iskustvo aplikacije u budućim verzijama. Ovaj pregled postavlja temelje za kontinuirani razvoj i prilagodbu aplikacije sve zahtjevnijim potrebama korisnika i tehnologija.

### 11.1. Identificirani nedostaci

Tijekom korištenja i testiranja aplikacije, identificirani su određeni nedostaci koji ograničavaju njezinu funkcionalnost i upotrebljivost u širem spektru situacija. Iako korisnik može dijeliti rezultate kao generiranu fotografiju formata *PNG,* aplikacija trenutno nema integriranu značajku dijeljenja (engl. *Share)* koja bi omogućila jednostavno i brzo dijeljenje rezultata unutar same aplikacije. Nadalje, aplikacija je trenutno dostupna isključivo na operativnom sustavu *Windows*, što isključuje korisnike drugih platformi.

Kako je aplikacija dostupna isključivo za Windows operativni sustav, ograničena je njena upotreba na drugim platformama popup Linux ili MacOS operacijskih sustava. Još je jedan značajan nedostatak to što je aplikacija optimizirana samo za rezoluciju 1920 x 1080, što može rezultirati neadekvatnim korisničkim iskustvom na drugim ekranima drugačijih rezolucija.

Osim toga, aplikacija trenutno ne podržava snimanje aktivnosti unutar jednog specifičnog prozora ili aplikacije. Na primjer, kada bi korisnik htio snimiti kretanje računalnog miša samo unutar internetskog preglednika, aplikacija mu to ne bi mogla omogućiti – ona snima cjelokupno kretanje računalnog miša po ekranu.

Konačno, aplikacija ne prepoznaje ispravno rad u *dual-monitor* okruženju. Kada se korisnik u radu na računalu služi s više monitora, aplikacija ne može točno prepoznati koordinate miša pri prelasku s jednog monitora na drugi, što može dovesti do netočnih rezultata, posebno kada prelazi maksimalnu širinu ekrana od 1920 piksela.

Ovi otkriveni nedostaci pružaju smjernice za buduće nadogradnje, s ciljem unapređenja funkcionalnosti i fleksibilnosti aplikacije.

### 11.2. Potencijalna nadogradnja kao *web* ekstenzija

Jedna od nadogradnji mogla bi biti integracija s internetskim preglednicima kao ekstenzija. Ova bi funkcionalnost omogućila aplikaciji da prati i bilježi korisnikove kretnje računalnog

miša unutar specifičnih aplikacija, poput *Jire*[5] ili *Trella*[6], što bi bilo korisno za dizajnere i razvojne timove koji žele razumjeti kako se korisnici služe svojim softverom.

Ekstenzija za internetske preglednike omogućila bi korisnicima anonimno snimanje svoje aktivnosti dok rade unutar jedne aplikacije, na primjer, dok se koristi *Jira* alat za upravljanje projektima. Prikupljeni podaci o kretanju miša mogli bi se zatim automatski slati razvojnom timu *Jire*, koji bi analizom rezultata mogli steći uvide u to kako se njihova aplikacija koristi u stvarnom vremenu. Na temelju tih podataka dizajneri bi mogli identificirati najkorištenije sektore unutar *Jirina* sučelja, te optimizirati dizajn i korisničko iskustvo prema stvarnim potrebama korisnika.

### 11.3. Analiza klikova miša

Jedna od nadogradnji koja bi unaprijedila funkcionalnost aplikacije jest dodavanje mogućnosti odvojene analize klikova miša, a ne samo generalno praćenja pokreta. Klikovi miša predstavljaju točke interakcije korisnika s aplikacijom, te njihova analiza može pružiti uvide u to kako korisnici koriste sučelje.

Ova nadogradnja omogućila bi korisnicima da pomoću opcije '*toggle'* jednostavno "prebacuju" prikaz između obične toplotne mape analize pokreta miša i analize interakcije. Klikovi bi se prikazivali na zasebnoj toplotnoj mapi, koristeći različite boje ili intenzitet kako bi se jasno razlikovali od samih pokreta miša. Time bi korisnici mogli identificirati ključna mjesta na ekranu gdje se događaju najvažnije interakcije, što bi pomoglo u optimizaciji sučelja i poboljšanju korisničkog iskustva.

Primjerice, kombiniranjem ovih dvaju podataka, odnosno pokreta i klikova, dizajneri bi mogli prepoznati područja koja privlače pažnju korisnika, ali možda ne potiču na akciju, ili pak identificirati najčešće korištene funkcionalnosti unutar aplikacije. Takva bi integracija povećala preciznost analize, pružajući sveobuhvatnu sliku o načinu na koji korisnici komuniciraju s aplikacijom, što bi doprinijelo boljem dizajnu i funkcionalnosti.

[5]*Jira* je alat za praćenje i organizaciju projekata. Često se koristi u razvojnim timovima te se lako koristi za raspodjelu poslova unutar tima.
[6]*Trello* je alat za organizaciju razvoja projekata s fokusom na dodjeljivanje zadataka članovima preko radne ploče. Ima funkcionalnost vizualizacije napretka projekta preko radnih kartica, lista, labela itd.

# 12. MOGUĆE PRIMJENE I KORISTI

Primjena aplikacije može biti raznolika. U poslovnom okruženju menadžeri mogu koristiti podatke za optimizaciju radnih procesa svojih digitalnih proizvoda ukazujući na potencijalne nedostatke ili nefunkcionalnosti značajki aplikacije. Uz to, uvidom u navike korisnika dizajneri korisničkog sučelja mogu analizirati koje dijelove aplikacije korisnici najčešće upotrebljavaju te na temelju tih informacija optimizirati dizajn za bolju intuitivnost i učinkovitost.

U zdravstvenom sektoru može se koristiti za prevenciju i praćenje problema povezanih s dugotrajnim radom na računalu. Na temelju prikupljenih podataka, poput toplinskih mapa, stručnjaci mogu pregledati podatke i dati savjete za poboljšanje ergonomije i učinkovitosti. Stručnjaci mogu analizirati konzistentnost pokreta miša, njegovu pokretljivost te identificirati statičnost koja može biti znak loših radnih navika.

Osim stručnjaka, i sami korisnici mogu pratiti svoj napredak i prilagoditi svoje radne navike u skladu s preporukama stručnjaka. Upotrebom aplikacije korisnici mogu prilagoditi svoje radno okruženje kako bi poboljšali raspored otvorenih prozora na sučelju. Na temelju prikupljenih podataka o pokretima miša moguće je optimizirati postavke radne stanice.

Aplikacija bi se mogla integrirati s drugim softverskim alatima, kao što su sustavi za upravljanje zadacima ili softver za analizu produktivnosti. Na taj način podaci o pokretima miša mogu se koristiti za pružanje dodatnih uvida u efikasnost rada i optimizaciju korištenja softverskih alata.

Ovo aplikativno rješenje može biti koristan alat za istraživače koji proučavaju ponašanje korisnika pri radu na računalu. Praćenje i analiza pokreta miša može pružiti podatke za različite studije, od istraživanja produktivnosti do istraživanja navika korisnika pri korištenju digitalnih medija.

Ukratko, aplikacija za praćenje računalnog miša osmišljena je kako bi pomogla korisnicima da razumiju svoje interakcije s računalnim sučeljem na dubljoj razini. Praćenjem svakodnevnih pokreta miša aplikacija može prikupiti podatke o obrascima korištenja, identifikaciji učestalih pokreta i potencijalno prepoznati neučinkovitosti u radu. Ova se aplikacija može koristiti u više svrha, od optimizacije radnog procesa i poboljšanja ergonomije do edukacije njenih korisnika i stručnjaka iz različitih područja.

### 12.1. Koristi za dizajnere i inženjere aplikacija

Kao što je naznačeno u prethodnom poglavlju, podaci prikupljeni ovom aplikacijom mogu poslužiti dizajnerima i inženjerima aplikacija. Analizom toplotnih mapa, inženjeri i dizajneri mogu dobiti dragocjene uvide u to kako se korisnici integriraju s njihovim softverom.

Dizajneri mogu otkriti koje dijelove aplikacije korisnici najviše koriste, gdje se najviše zadržavaju, što im privlači pažnju, gdje nailaze na poteškoće. Nadalje, inženjeri bi mogli identificirati potencijalna uska grla ili nepotrebne korake. Ove se informacije mogu upotrijebiti za prilagodbu korisničkog sučelja kako bi ono bilo intuitivnije i učinkovitije te na taj način povisiti razinu zadovoljstva korisnika tijekom upotrebe njihova proizvoda. Također, podaci mogu otkriti potrebe i želje korisnika koje trenutno nisu zadovoljene. To može potaknuti razvoj novih značajki (engl. *feature*) ili proizvoda koji odgovaraju korisničkim očekivanjima.

Zaključno, aplikacija za praćenje računalnog miša predstavlja inovativan alat koji može doprinijeti boljem razumijevanju i poboljšanju svakodnevne interakcija s računalnim sučeljem. Uz pomoć stručne analize i uvida dizajnera, korisnicima i profesionalcima može pružiti vrijedne uvide u obrasce upotrebe digitalnih proizvoda i računalnog sučelja, sugerirajući jedan od načina optimizacije proizvoda, a time i korisničkog iskustva te radnih navika samih korisnika.

### 12.2. Izazovi u razvoju

Tijekom razvoja aplikacije suočilo se s nekoliko izazova. Jedan od glavnih izazova bio je optimizacija performansi, posebno u pogledu brzine učitavanja podataka i generiranja toplotnih mapa. Inicijalno, aplikacija je imala problema s dužim vremenom učitavanja zbog spore interakcije s bazom podataka. Ova poteškoća je riješena implementacijom privremenog pohranjivanja koordinata u memoriju, što je značajno poboljšalo brzinu i fluidnost rada aplikacije.

Još jedan izazov bio je rad u *dual-monitor* okruženju. Aplikacija nije ispravno prepoznavala koordinate miša prilikom prelaska između monitora, što je dovodilo do netočnih rezultata. Ovo pitanje zahtijeva dodatne prilagodbe kako bi se osigurala točnost u okruženjima s više monitora.

Pored toga, ograničenja u vezi s podrškom za različite operativne sustave i rezolucije također su predstavljala izazov. Trenutna verzija aplikacije podržava samo Windows operativni sustav i specifičnu rezoluciju, što može ograničiti njezinu primjenjivost u širem kontekstu.

### 12.3. Smjernice za budući razvoj

Na temelju trenutnih rezultata i identificiranih nedostataka, nekoliko smjernica za budući razvoj može biti istaknuto. Integracija s web preglednicima kao ekstenzija predstavlja priliku za proširenje funkcionalnosti aplikacije i poboljšanje korisničkog iskustva u specifičnim web aplikacijama kao što su *Jira* ili *Trello*.

Također, dodavanje mogućnosti za rad u različitim operativnim sustavima i podrška za različite rezolucije bit će ključne nadogradnje za povećanje univerzalnosti aplikacije. Razvijanje funkcionalnosti za snimanje aktivnosti unutar specifičnih prozora i unapređenje rada u *dual-monitor* okruženju također su područja koja bi trebala biti prioritet u budućim verzijama.

# 13. ZAKLJUČAK

Ovaj rad bavio se razvojem aplikacije za snimanje i analizu pokreta računalnog miša. Kroz analizu i razvoj, aplikacija je pokazala svoju funkcionalnost u praćenju i vizualizaciji korisničkih interakcija s računalnim sučeljem. Generiranjem toplotnih mapa korisniku je omogućeno vizualiziranje najaktivnijih i neaktivnih dijelova zaslona. Iako je funkcionalna, aplikacija ima nekoliko identificiranih nedostataka, poput rada na jednu razlučivost ekrana i rada isključivo na Windows operativnom sustavu. Ovi nedostaci otvaraju prostor za buduće nadogradnje i optimizaciju same aplikacije.

Projekt je prikazao ispravan način izrade i pisanje koda, uz pridržavanje programskih standarda poput korištenja pomoćnih klasa, raspodjele klasa i metoda, izbjegavanje redundancije u kodu te pravilnog imenovanje varijabli i metoda. Aplikacija se može pokrenuti na bilo kojem računalu Windows operativnog sustava bez potrebe za postavljanjem razvojnih okolina.

Potencijalne nadogradnje uključuju integraciju aplikacije u internetske preglednike kao alat za praćenje rada unutar specifičnih internetskih aplikacija. Također, klikovi miša bi isto mogli biti dio proširenja aplikacije za detaljniji ispis korisničkih navika.

# Izjava o autorstvu

**MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU**

**Bana Josipa Jelačića 22/a, Čakovec**

**IZJAVA O AUTORSTVU**

Završni/diplomski rad isključivo je autorsko djelo studenta te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, internetskih i drugih izvora) bez pravilnog citiranja. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom i nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, Romano Keser (ime i prezime studenta) pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog rada pod naslovom Principi programskog inženjerstva na primjeru aplikacije za praćenje položaja računalnog miša

te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:

(vlastoručni potpis)

# Literatura


1. Bohem B. W.: Software Engineering Economics, 1976.
2. Farrel J.: Programming with Microsoft Visual C#, 2019.
3. Géron A.: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2017.
4. Hunter J. D.: Matplotlib: A 2D graphics enviornment. Computing in Science & Engineering, 2007.
5. Kelly K.: The Inevitable Understanding the 12 Technological Forces That Will Shape Our Future, 2016.
6. MacDonald M.: Pro WPF in C#: Windows Presentation Foundation in .NET 4, 2010.
7. Martin R. C.: Clean Code: A Handbook of Agile Software Craftsmanship, 2008.
8. Martin R. C: Python Dana Science Handbook: Essential Tools for Working with Data, 2016.
9. Nađ J.: Programsko inženjerstvo i informacijski sustavi, Međimursko veleučilište u Čakovcu, 2020.
10. Nathan. A.: Windows Presentation Foundation Unleashed, 2006.
11. Price .J M.: C# 9 and .NET 5 – Modern Cross-Platform Development, 2020.
12. Yuen S.: Mastering Windows Presentation Foundation, 2017.

## Popis ilustracija