

SKIBIDI Traffic

**Your AI companion for managing traffic.
Create and optimize your intersection using AI.**

Proiect realizat de catre:
[Cezara Zamfir](#)

Cristian Dima
David Muscalu
Cozma Rares

Cuprins

Descrierea aplicatiei	4
User stories, backlog creation	14
Diagrame.....	15
Source control cu Git.....	16
Teste automate.....	18
Raportare bug si rezolvare cu pull request	20
Design patterns	22
Prompt engineering: documentarea folosirii toolurilor de AI in timpul dezvoltarii software	26

Descrierea aplicatiei

Overview

Skibidi Traffic este o aplicatie web de management, vizualizare, simulare si optimizare a traficului. Aplicatia ofera utilizatorului un mediu in care acesta poate:

- desena intersectii
- simula traficul rutier pe acestea
- utiliza un model de AI bazat pe reinforcement learning pentru a imbunatati performanta intersectiei
- salva configuratiile create intr-un cont personal

Ideea aplicatiei

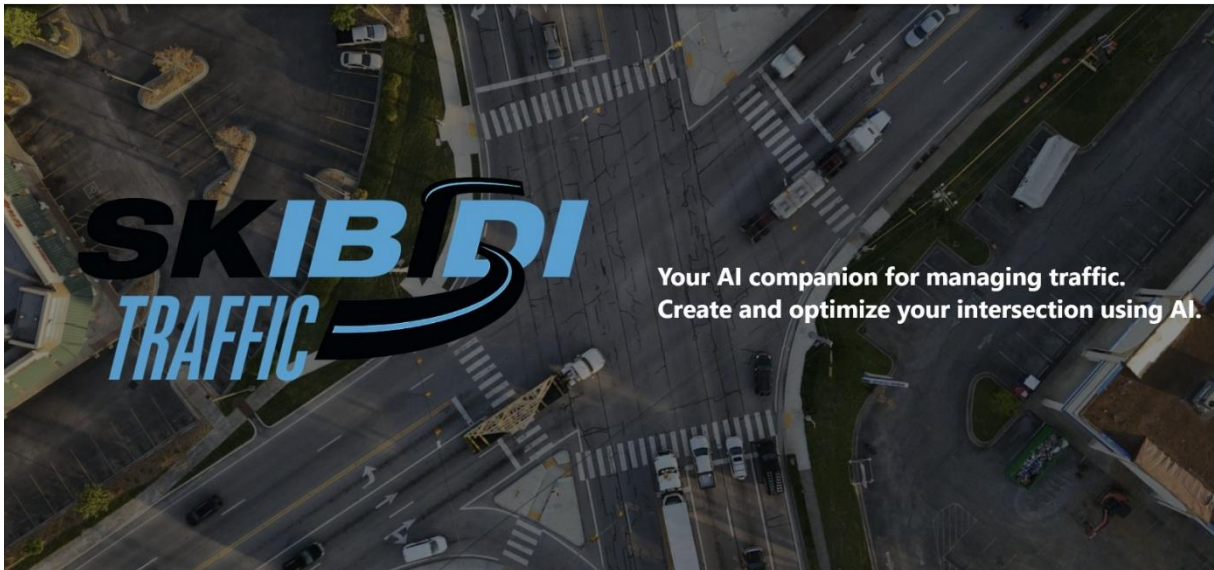
Promisiunea primarului nostru, domnul Nicusor Dan, de a rezolva categoric problema traficului bucurestean a adus speranta in sufletele noastre. El a trebuit, insa, sa devina presedinte pentru a salva tara de fascism, asa ca problema traficului a ramas pe capul nostru. Aiciam luat problema in propriile maini: ne-am propus sa cream o aplicatie de fluidizare a traficului care sa duca la indeplinire visul presidential.

Paginile aplicatiei

Principalele pagini ale aplicatiei web Skibidi Traffic sunt urmatoarele:

1. Home
2. Login
3. Sign up
4. Create
5. Simulate
6. My account
7. About us

Utilizatorul intra in aplicatie prin Home page. Pe aceasta pagina sunt prezentate functionalitatile generale ale site-ului. Tot aici are optiunea de a accesa cateva intersectii predefinite fara a fi nevoit sa isi creeze un cont. Astfel, el poate decide daca aplicatia este potrivita scopurilor sale.





About Us

We're a young and passionate team on a mission to untangle traffic, one intersection at a time. Our love for AI and traffic lights brought us here—and we're determined to help Nicușor free Bucharest! While he battles the real estate mafia, we handle the signal timings.



About Skibidi Traffic

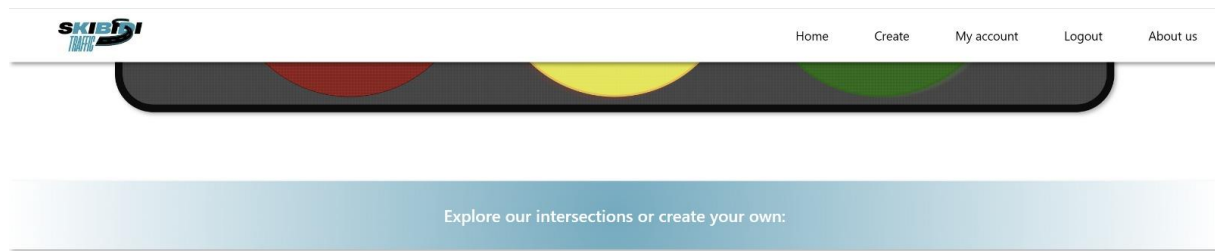
On this site, you can explore pre-built intersections like Piața Romană, Universitate, or Titan, and watch as our AI calculates the optimal green light intervals to maximize vehicle flow. Feeling creative? Log in and design your own intersections!



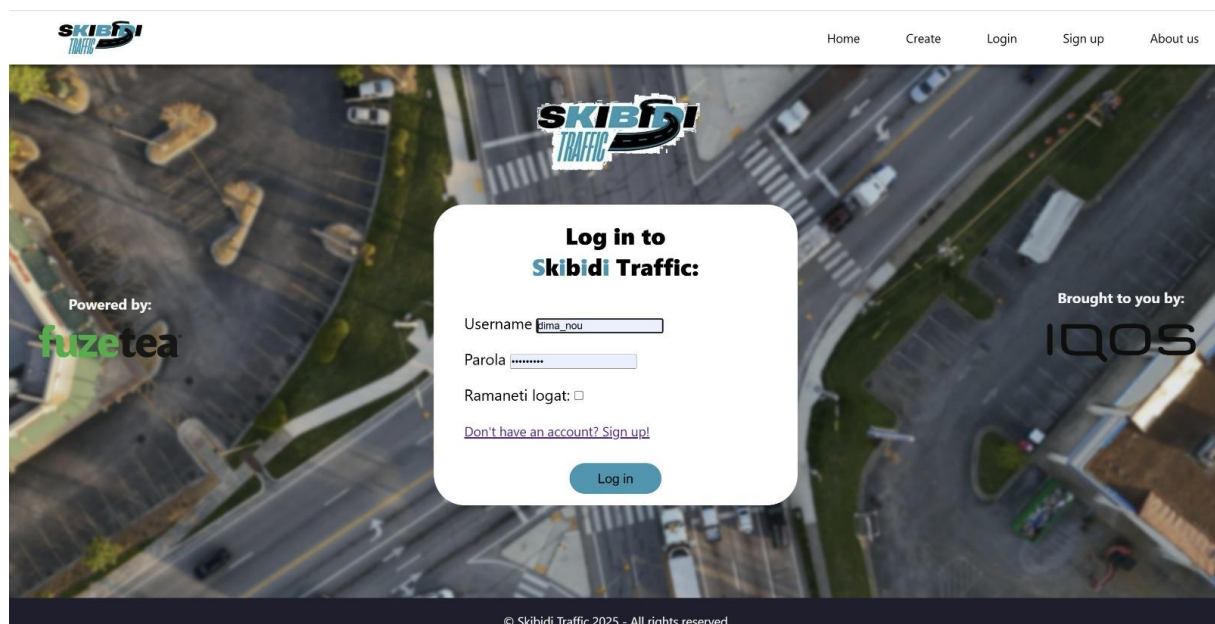
Let's create!

Ready to take control? Log in and start designing your own intersections. Then use our AI to optimize traffic flow, set green light durations, and simulate real-time traffic conditions.

Explore our intersections or create your own:

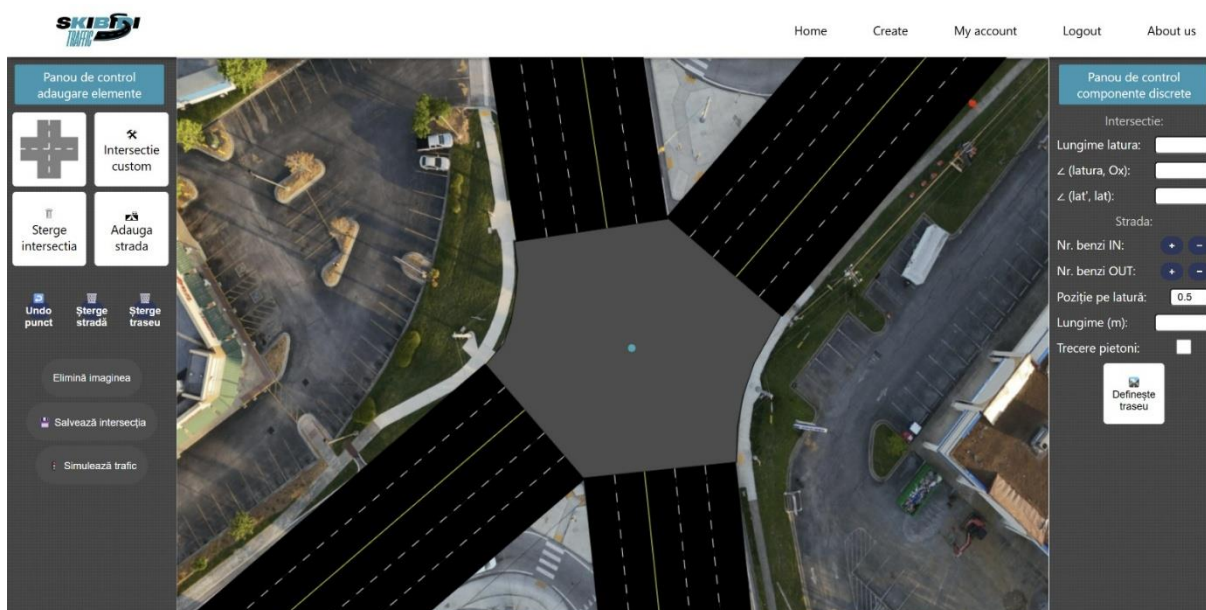


La click pe butonul Login sau Sign Up din navbar, utilizatorul este trimis la pagina de Login/Sign Up. Daca acesta nu este logat si incearca sa navigheze catre pagina Create, este redirectionat automat catre Login. In paginile Login/Sign Up se fac verificari corespunzatoare pe toate campurile formularelor.

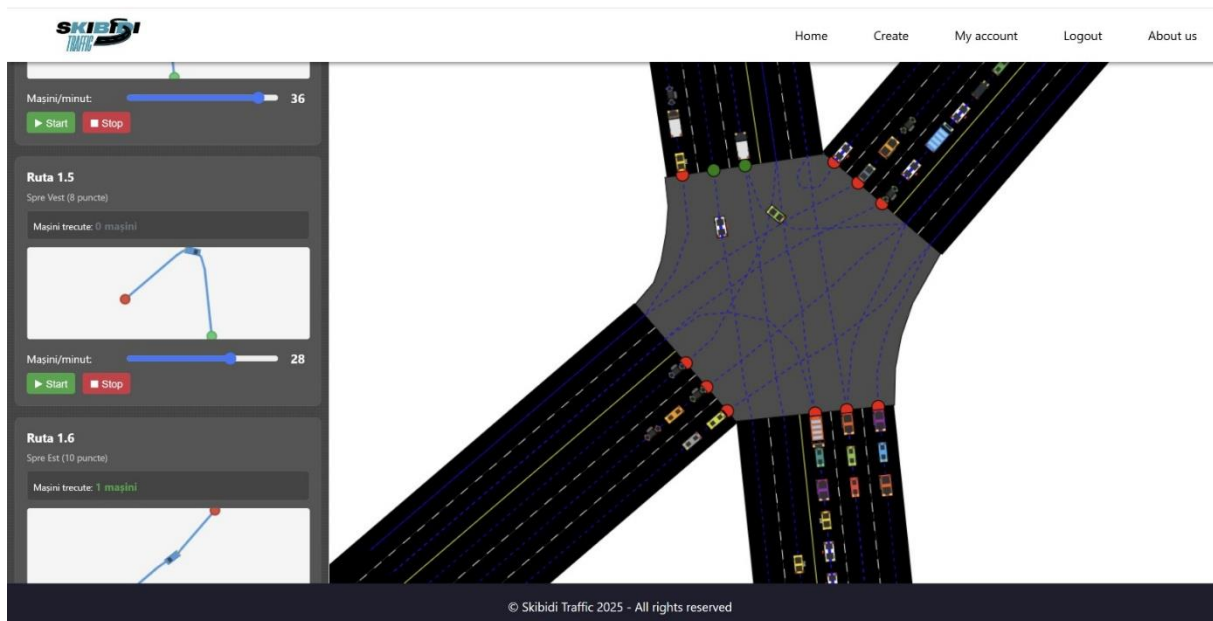




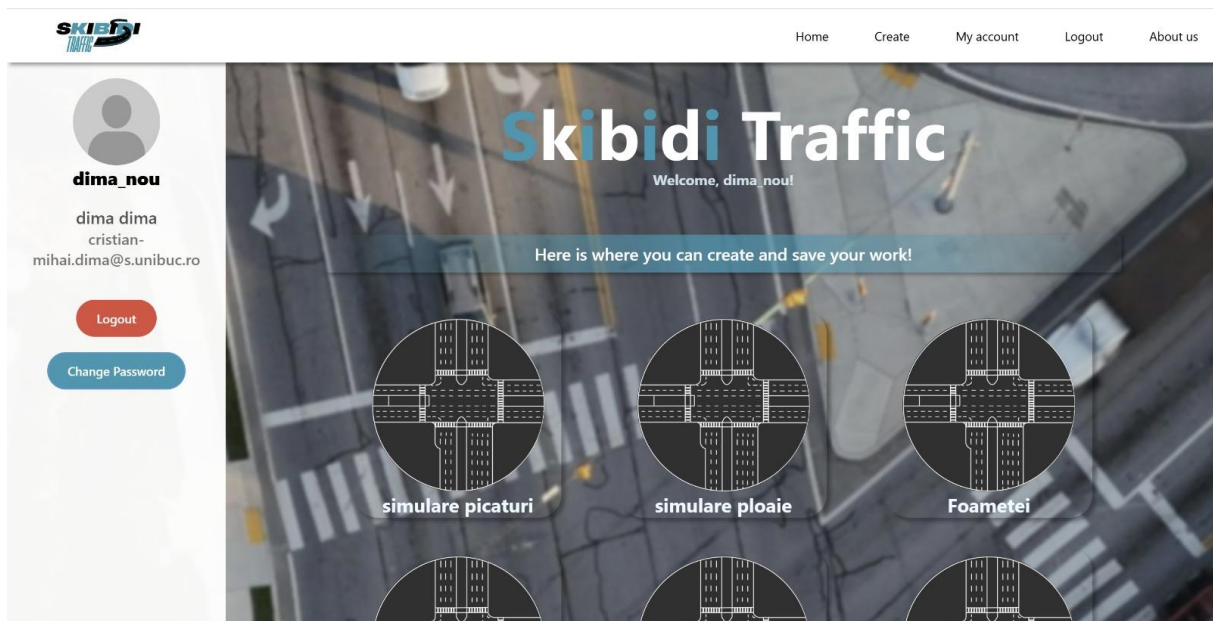
Pe pagina Create utilizatorul isi poate desena, dupa ce s-a logat, intersecții. Pagina dispune de doua panouri de control al elementelor, iar nivelul de interactiune user – element este ridicat, acesta putand manipula efectiv elementele intersecției reprezentate pe canvas cu ajutorul mouse-ului.



Pe pagina Simuleaza, unde este redirectionat de pe Create, cand apasa pe “Simuleaza trafic”, utilizatorul poate rula o simulare a traficului in timp real, pe intersecția create de el. Poate modifica timpii semafoarelor manual sau poate opta pentru timpii sugerati de modelul nostru de Inteligenta Artificiala bazat pe Reinforcement Learning.




Pe pagina My account, utilizatorul are o zona dedicata informatiilor contului sau, precum si o zona in care poate vedea toate intersectiile create de el, listate descrescator cronologic.




Pagina About Us este cea dedicata prezentarii echipei noastre.


Meet the Team




Dima



Cezara



Muscalu



Rares

Our Goal

To create a platform where you can visualize and improve traffic flow through smart intersection design. Whether you want to experiment with Piața Romană or create a new roundabout in Cucuieți, we've got you covered.

POWERED BY: IQOS **fuzetea**

Flow-ul aplicatiei

Cand utilizatorul ajunge pe pagina Create, i se prezinta un canvas insotit de 2 panouri de control. Folosindu-se de clasele de elemente listate in sidebar-uri si prin interactiunea cu canvas-ul, el poate reprezenta efectiv intersectii realiste. Desenarea este insotita de verificarea permanenta a regulilor ce vor permite ulterior simularea adecvata a traficului pe intersectia definita de utilizator.

Utilizam o baza de date stocata in cloud ce permite ca intersectiile create de user sa fie salvate si asociate contului sau. La salvare, userul e rugat sa atribuiască un nume intersectiei.

Alta optiune posibila este apasarea unui buton de simulare a traficului ce redirectioneaza utilizatorul catre pagina Simuleaza. Aici ii este prezentata o alta interfata, cu un meniu cu optiuni specifice customizarii traficului, precum si un canvas pe care se desfasoara simularea, cu mentiunea ca gradul de interactiune cu acesta este acum, redus, pentru a permite simularea corecta si realista a traficului. Initial, simularea ruleaza cu timpii optimi ai semafoarelor prezisi de modelul de RL, insa userul poate introduce si manual inputuri pentru timpii fazelor, iar simularea se va updatea in timp real.

Manual de instructiuni pentru paginile Create, Simuleaza si Play

In aceasta sectiune este descrisa interactiunea userului cu paginile aplicatiei web ce implica definirea intersectiilor si simularea traficului:











Utilizarea paginii Create

Pe canvas apare intersectia cu configuratia Dorita de utilizator. Panourile laterale contin butoane si campuri ce pot seta componentele intersectiei.

Panoul de control adaugare elemente:

Gestioneaza elementele mari ale intersectiei:

Contine:

- butonul  → la click pe el, apoi pe canvas se insereaza o intersectie predefinita de tip SQUARE, cu 4 strazi incidente, fiecare avand cate 1 banda pe sens.
- Butonul  → la click pe el, apoi pe canvas, se intra in modul de definire a intersectiei custom
- Butonul  → la click pe el se curata tot canvasul, any unsaved work *it's gone* :)
- Butonul  → la click pe o intersectie deja desenata pe canvas, apoi pe butonul „Adauga strada”, se intra in modul ce permite adaugarea strazilor
- Butonul  → daca ne aflam deja in modul de definire al intersectiei custom si am plasat pe canvas unul dintre punctele ce reprezinta varfurile poligonului intersectiei, apasarea butonul „Undo punct” va elimina din lista varfurilor poligonului ultimul punct adaugat.
- Butonul  → la click pe o strada deja adaugata, apoi pe „Sterge strada”, se elimina strada selectat din lista strazilor incidente in intersectie
- Butonul  → la click pe el, daca am intrat in prealabil in modul „Definire traseu” si mi se afiseaza traseele existente asociate intersectiei si selectez unul dintre aceste trasee, traseul respectiv se sterge din lista rutelor asociate intersectiei
- Butonul  → permite incarcarea din Fisiere a unei imagini de fundal ce poate fi utilizata pentru ghidaj in timpul desenarii intersectiei custom
- Butonul  → salveaza intersectia sau modificarile aduse unei intersectii deja existente in baza de date, cu userul current drept owner.
- Butonul  → redirectioneaza userul pe pagina de simulare. Daca intersectia nu este salvata, I se solicita userului sa isi salveze munca inainte de simulare.



Desenarea unei intersectii custom:

Dupa intrarea in modul “Desenare intersectie custom”, utilizatorul plaseaza puncte pe canvas reprezentand varfurile unui poligon ce defineste intersectia. Ordinea punctelor determina forma poligonului. Cand toate punctele dorite au fost adaugate, se da click pe

“Stop and save” și se iese din modul “Desenare intersecție”, pe canvas aparand intersecția desenată.

Adaugarea strazilor:

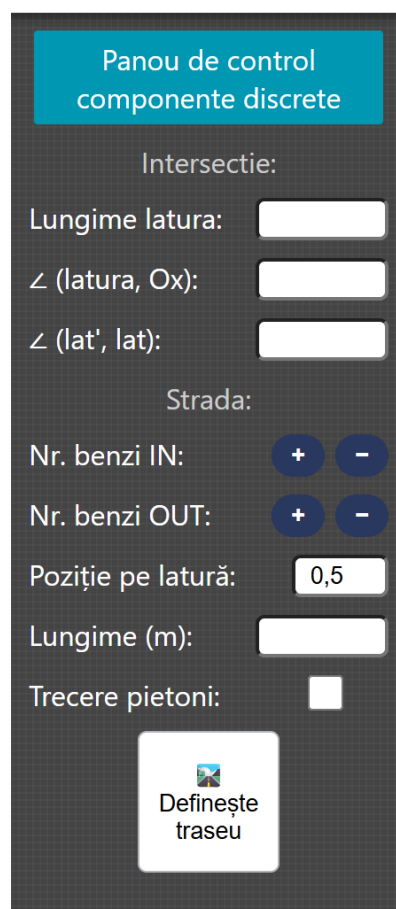
Se selectează o intersecție, apoi se apasă butonul “Adaugare strada”. Se da click în proximitatea unei dintre laturile intersecției, iar pe acea latură va apărea o stradă incidentă.

Panoul de control componente discrete:

Gestionează subcomponentele sau proprietățile unor elemente majore ale intersecției.

Conține:

- Referitor la intersecție:
 - Campul “Lungime latură”, unde, în timpul desenării intersecției, utilizatorul poate preciza o dimensiune reală în metri (ce va fi convertită în pixeli pentru a fi reprezentată proporțional pe canvas). Laturii determinate de ultimul punct curent desenat și punctul anterior i se va atribui această dimensiune.
 - Campul “unghi(latură, Ox)”, ce manipulează înclinarea ultimei laturi în curs de desenare față de axa Ox
 - Campul “unghi(latură', latură)”, ce manipulează înclinarea ultimei laturi în curs de desenare față de latură desenată anterior
- Referitor la strada:
 - Butoane ce cresc sau scad numărul de benzi ce intra/ies în/din intersecție, asociate unei strazi deja selectate
 - Campul “Pozitie pe latură”, ce primește o valoare între 0 și 1 reprezentând poziționarea centrului strazii (centrul strazii este punctul de intersecție al axului strazii cu latură intersecției) pe latură.
 - Campul “Lungime”, unde se setează lungimea în metri (ulterior convertită în pixeli) a strazii
 - Campul boolean “Trecere pietoni”. Dacă este bifat, strada are trecere de pietoni în capătul dinspre intersecție.
 - Butonul “Defineste traseu”, care, la click, activează modul de definire a unui traseu prin intersecție. Un traseu este un drum urmat de o mașină care vrea să ajungă de pe banda x a strazii i pe banda y a strazii j.



Definirea unui traseu:

Se apasa butonul “Defineste traseu”. Pe canvas, in centrul fiecarei benzi de intrare in intersectie, vor aparea puncta verzi ce indica punctele de start posibile pentru definirea unui traseu. Se da click pe unul dintre ele, apoi se deseneaza puncte prin intersectie prin click ce vor forma o linie franta deschisa reprezentand traseul. Se inainteaza cu puncte din apropiere in apropiere, pana catre un punct de finalizare posibil, semnalat prin buline verzi. La click pe un punct de END, se adauga traseul in vectorul de trasee al intersectiei.

Stergerea unui traseu:

Se intra in modul “Defineste traseu”, se da click pe butonul din panoul stang “Sterge traseu”, apoi se da click pe un traseu care se vrea a fi sters.

Simularea traficului:

Dupa ce utilizatorul a apasat pe “Simuleaza trafic”, din pagina Create, si a ajuns pe pagina simularii, in partea stanga are la dispozitie un panou de control al simularii, iar in dreapta paginii, canvasul pe care ruleaza simularea.

Controlul simularii se poate realiza prin:

- Modificarea timpilor fazelor (la hover peste o faza din sidebar se evidentiaza rutile ce fac parte din faza respective). Default, timpii fazelor sunt cei prezisi ca optimi de modelul AI.



- Setarea fluxului de masini pe fiecare ruta in parte, gratie casetelor dedicate preview-ului rutelor, precum si simularea traficului doar pe anumite rute (se da start din caseta dedicate rutei respective)

Control Flux Trafic

Setează numărul de mașini pe minut pentru fiecare rută

Ruta 1.1
 Spre Nord (9 puncte)

Mașini trecute: **24 mașini**

Mașini/minut: **24**

▶ Start

■ Stop

Ruta 1.2
 Spre Nord (8 puncte)

Mașini trecute: **24 mașini**

Mașini/minut: **10**

▶ Start

■ Stop

- În partea inferioara a panoului de control se regaseste sectiunea:

Total Statistici

Total mașini trecute: **446 mașini**

Vreme Rea: Dezactivată

Pornește Tot Traficul

Oprește Tot Traficul

Reset Contoare

Export Stats

× Închide Simularea

Aici, utilizatorul poate da start tuturor rutelor simultan, poate opri tot traficul, si poate active un buton de vreme rea, ce face masinile sa adapteze viteza la conditiile meteo.

User stories, backlog creation

Publicul țintă:

1. **Primăriile și autoritățile locale**, instituțiile din teritoriu implicate în gestionarea traficului rutier
 - Vor să optimizeze traficul în oraș pentru a reduce blocajele și poluarea.
 - Pot folosi datele generate pentru a justifica investiții în semafoare inteligente sau infrastructură nouă.
2. **Inginerii de trafic și urbaniști**
 - Vor să analizeze impactul diferitelor configurații ale intersecțiilor.
 - Pot testa soluții pentru a fluidiza circulația înainte de a le implementa în realitate.
3. **Pasionații de jocuri de strategie și simulatoare**
 - Vor să își testeze abilitățile de gestionare a traficului și să concureze cu AI.
 - Le plac provocările bazate pe optimizare și luarea deciziilor rapide.

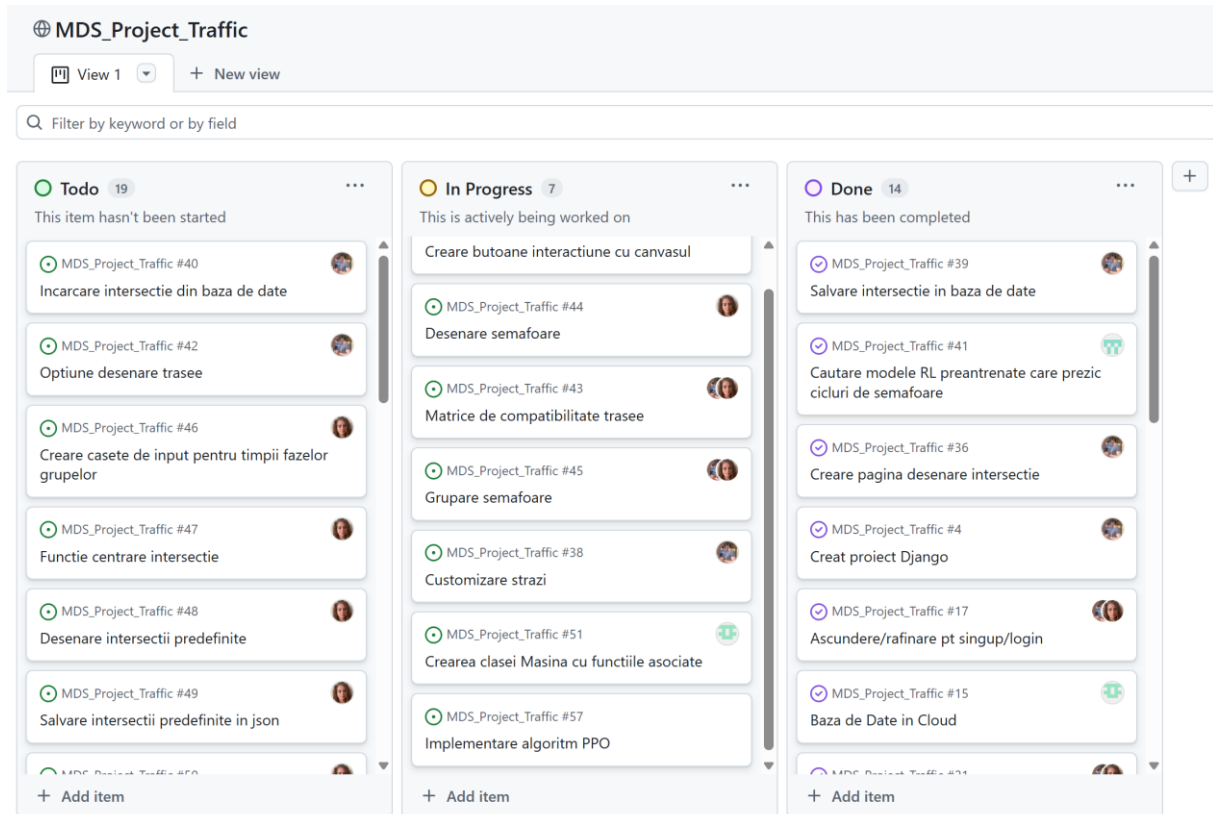
User stories:

1. Ca utilizator, doresc să pot accesa un site cu mai multe pagini, pe care să pot vizualiza intersecții reale, transpuse într-un format grafic (incluzând elemente precum semafoare, semne de circulație, marcaje rutiere, benzi de circulație etc.)
2. Ca utilizator, doresc să pot configura propriile intersecții.
3. Ca utilizator, doresc să pot optimiza fluxului de trafic atât manual, cât și cu ajutorul AI
4. Ca utilizator, doresc să pot accesa câteva intersecții default (ex. Piața Romană, Universitate etc.) fără a fi nevoie să îmi creez un cont pe site, astfel încât să îmi dau seama dacă acest produs software este potrivit pentru mine.
5. Ca utilizator, doresc un grad mare de interacțiune cu intersecția atunci când o configurez.
6. Ca utilizator, doresc să pot simula trafic în timp real peste o intersecție creată.
7. Ca utilizator, în urma sign up-ului pe site, doresc să îmi pot crea și salva intersecțiile personalizate, astfel încât să pot reveni oricând pentru a le modifica sau testa noi setări de semaforizare.
8. Ca utilizator, doresc să pot ajusta durata semaforizării pentru fiecare direcție de trafic în parte, astfel încât să pot experimenta cu diferite scenarii de optimizare a fluxului de trafic.
9. Ca utilizator, doresc să pot seta fluxul de trafic (numărul de mașini) din toate sensurile unei intersecții, astfel încât să pot simula scenarii complete de trafic și să optimizez semafoarele în funcție de nevoile fiecărei direcții.
10. Ca utilizator, doresc să pot vedea statistici ale simulării precum numărul de mașini care au trecut în total prin intersecție.

11. Ca utilizator, doresc o simulare cât mai aproape de realitate a condițiilor de trafic (mașini care merg la viteze diferite, reacții întârziate ale șoferilor, condiții meteo care influențează deplasarea autovehiculelor).

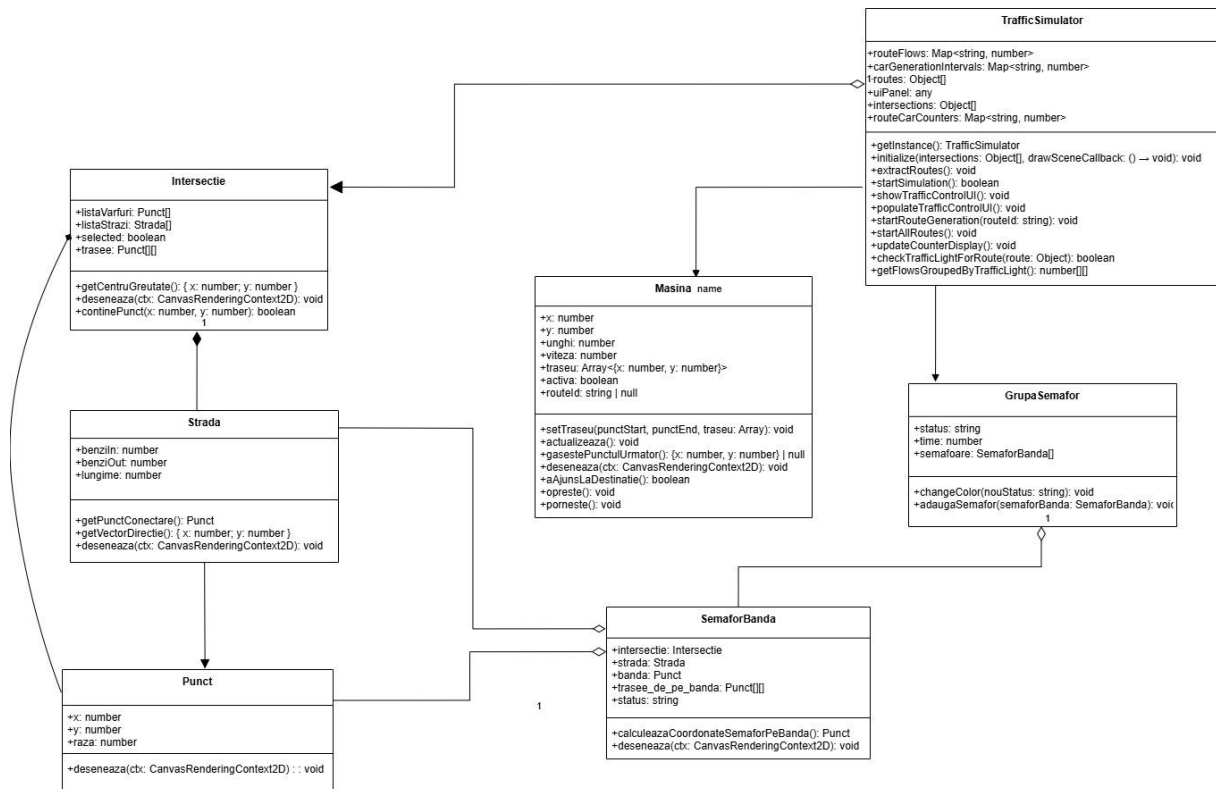
Backlog:

Având în vedere dimensiunea proiectului și timpul scurt avut la dispoziție pentru implementarea sa, a fost important să ne formulăm clar task-urile și să știm clar care este următorul pas în procesul de dezvoltare software. În acest sens, ne-am creat un tabel de task-uri pe GitHub:



Diagrame

Diagrama UML de clase a aplicației noastre este reprezentată mai jos. Clasa Singleton TrafficSimulator gestionează simularea traficului. O intersecție este alcătuită din puncta și din strazi. Fiecare banda are asociat un SemaforBanda, iar mai multe semafoare banda corespunzătoare traseelor compatibile unele cu altele sunt grupate într-un obiect GrupaSemafor. Masinile ce se deplasează prin intersecție în timpul simulării au dedicată clasa Masina.

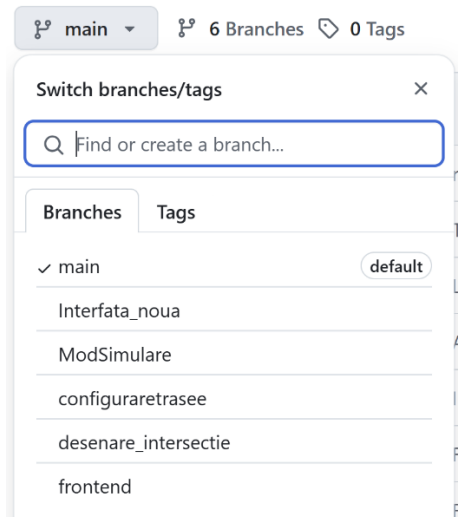


Source control cu Git

Crearea branch-urilor

Fiecare functionalitate majora a fost dezvoltata pe un branch separat, pentru a permite lucru paralel si testare independenta:

- ModSimulare – contine logica de baza pentru simularea traficului (afisarea, pornirea si resetarea simularii)
- Interfata_noua – contine logica de desenare si circulare a masinilor
- configuraretrasee – logica pentru desenarea si stergerea traseelor in intersectie
- desenare_intersectie – logica pentru desenarea intersectiilor din pagina Create, cu clase javascript pentru elementele folosite(Punct, Strada)
- frontend – crearea proiectului Django si a paginilor aplicatiei web(Home, Login, SignUp, My Account, About Us)



Commit-uri

Pentru a mentine un istoric clar si usor de urmarit, fiecare membru al echipei a realizat commit-uri frecvente si descrieri relevante. In total, proiectul contine peste 10 commit-uri semnificative. Exemple de commit-uri relevante:

[“Butoane stergere/undo, pagina simuleaza”](#)

[“Adaugare rute intersectie, salvare obiecte intersectie user in BD”](#)

[“Pagina create cu functii noi de desenare”](#)

[“Functionalitati de login/logout/signup/changepassword”](#)

Pull Requests si integrare

Dupa finalizarea lucrului pe un branch, modificarile au fost integrate in ramura main prin pull request-uri. Acestea au fost utilizate pentru verificarea si testarea codului inainte de integrare si pentru evitarea integrarii accidentale a unor modificari incomplete.

Pentru integrarea modificarilor s-a utilizat merge si rebase, asigurand coerenta codului si pastrarea unui istoric clar al contributiilor.

<input type="checkbox"/>	0 Open	9 Closed	Author	Label	Projects	Milestones	Reviews	Assignee	Sort
<input type="checkbox"/>			Interfata noua						
			#32 by dimacul was merged 2 weeks ago						
<input type="checkbox"/>			Bug - traseul nu incepe si nu se termina in capetele strazilor	bug					
			#31 by dimacul was merged 2 weeks ago						
<input type="checkbox"/>			Adaugare rute intersectie, salvare obiecte intersectie user in BD						
			#30 by dimacul was merged 2 weeks ago						

Teste automate

Am creat teste automate pentru verificarea corectitudinii functionalitatilor de creare si salvare a intersectiilor in baza de date, inclusiv desenarea acestora pe canvas.

Tehnologii folosite

- Selenium WebDriver: automatizare browser Chrome

Test de autentificare

- Testeaza logarea automata pe site cu un cont de test.
- Verifica accesul catre pagina de creare a intersectiilor.

Testarea crearii intersectiilor

Pentru fiecare configuratie dintr-un fisier JSON cu mai multe puncte:

- Deschide pagina de creare.
- Activeaza modul de desenare.
- Deseneaza punctele pe canvas.
- Dezactiveaza modul de desenare (finalizare).
- Salveaza intersectia cu un nume specific.
- Confirma popup-urile de alerta.

Testarea resetarii si incarcarii paginii

Se reincarca pagina de creare dupa fiecare intersectie pentru a preveni erorile din starea anterioara.

Pasi pentru rulare:

- descarcare chromeDriver
- pornire server Django
- pip install selenium
- python teste_intersectii.py

Trebuie inlocuita aceasta bucata de cod din [create.js](#):

```
if (!idIntersectie) {  
    nume = prompt("Dă un nume intersecției:");
```

```

if (!nume || nume.trim() === "") {
    alert("Numele este necesar.");
    return;
}
}

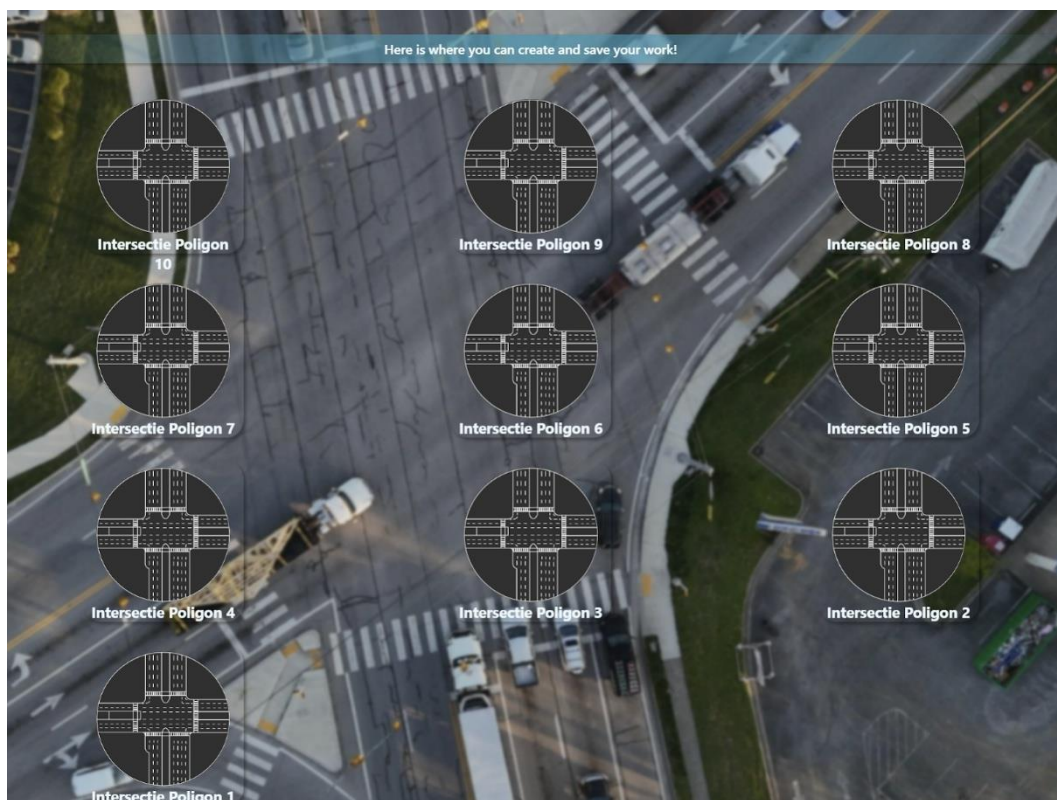
```

Cu:

```

nume = prompt("Dă un nume intersecției:");
if (!nume || nume.trim() === "") {
    alert("Numele este necesar.");
    return;
}

```



```

Incep testarea...
* Testez: Intersectie Poligon 1
ARNING: All log messages before absl::InitializeLog() is called are written to STDERR
0000 00:00:1749947048.662655 26088 voice_transcription.cc:58] Registering VoiceTranscriptionCapability
* Alertă: Intersecția a fost salvată cu succes!
* Testez: Intersectie Poligon 2
reated TensorFlow Lite XNNPACK delegate for CPU.
ttempting to use a delegate that only supports static-sized tensors with a graph that has dynamic-sized tensors (tensor
-1 is a dynamic-sized tensor).
* Alertă: Intersecția a fost salvată cu succes!
* Testez: Intersectie Poligon 3
* Alertă: Intersecția a fost salvată cu succes!
* Testez: Intersectie Poligon 4
* Alertă: Intersecția a fost salvată cu succes!
* Testez: Intersectie Poligon 5
* Alertă: Intersecția a fost salvată cu succes!
* Testez: Intersectie Poligon 6
* Alertă: Intersecția a fost salvată cu succes!
* Testez: Intersectie Poligon 7
* Alertă: Intersecția a fost salvată cu succes!
* Testez: Intersectie Poligon 8
* Alertă: Intersecția a fost salvată cu succes!
* Testez: Intersectie Poligon 9
* Alertă: Intersecția a fost salvată cu succes!
* Testez: Intersectie Poligon 10
* Alertă: Intersecția a fost salvată cu succes!
* Testele s-au terminat.

```

Raportare bug si rezolvare cu pull request

Bug #31 - Traseul nu incepe si nu se termina in capetele strazilor

Descriere:

La configurarea unui traseu, acesta nu incepea din primul punct al strazii si nu se termina in ultimul punct, ceea ce genera o reprezentare incompleta a traseului pe harta.

Fisiere afectate:

Semafor.js, Strada.js, create.js

Cauza probabila:

Logica initiala nu adauga explicit punctele de inceput si final ale fiecărei strazi in traseu. Se preluau doar punctele unde intra o masina in intersectie.

Comportament asteptat:

Traseul trebuie sa includa atat primul, cat si ultimul punct al fiecărei strazi selectate, pentru a putea simula traficul de la inceputul pana la sfarsitul unei strazi.

Actiuni efectuate:

- Am modificat codul pentru a adauga manual punctele de capat in traseu.
- Commit-ul contine modificarile necesare pentru ca traseul sa fie generat complet.

Branch:

configuraretrasee

Pull Request: #31

Status: Pull request merged into main

Commit: [7f31333](#)

Bug #33 - Contentul paginii "Simulare" apare partial sub meniu (fara pull request)

Descriere:

La accesarea paginii "Simulare", o parte din content apare ascuns sub bara de meniu. Acest comportament afecteaza lizibilitatea si interactiunea cu pagina.

Fisier afectat:

simuleaza.js

Cauza probabila:

CSS-ul asociat clasei behindNavBar, care nu tine cont de inaltimea meniului fix.

Comportament asteptat:

Continutul paginii sa inceapa imediat sub meniu, fara sa fie suprapus.

Actiuni efectuate:

- In CSS-ul paginii sau direct in simuleaza.js, am modificat stilurile pentru a adauga un padding-top corespunzator (ex: 80px) la containerul principal.
- Astfel, continutul este impins sub meniu si nu mai este ascuns.

Commit: [a8b7ee6](#)

Bug #33 - Contentul paginii "Simulare" apare partial sub meniu (fara pull request)

Descriere:

La accesarea paginii "Simulare", o parte din content apare ascuns sub bara de meniu. Acest comportament afecteaza lizibilitatea si interactiunea cu pagina.

Fisier afectat:

simuleaza.js

Cauza probabila:

CSS-ul asociat clasei behindNavBar, care nu tine cont de inaltimea meniului fix.

Comportament asteptat:

Continutul paginii sa inceapa imediat sub meniu, fara sa fie suprapus.

Actiuni efectuate:

- In CSS-ul paginii sau direct in simuleaza.js, am modificat stilurile pentru a adauga un padding-top corespunzator (ex: 80px) la containerul principal.
- Astfel, continutul este impins sub meniu si nu mai este ascuns.

Commit: [a8b7ee6](#)

Bug #35 - Salvare intersectii duplicat în modul "Create" (fara pull request)

Descriere:

La salvarea unei intersectii in modul "Create", intersectia se salveaza in baza de date, dar URL-ul nu se actualizeaza cu id-ul acesteia. Astfel, la apasarea butonului "Simuleaza trafic", functia nu gaseste id-ul in URL si cere salvarea din nou, desi intersectia exista deja.

Fisier afectat:

create.js

Cauza probabila:

Lipsa actualizarii/redirectionarii URL-ului dupa salvare.

Comportament asteptat:

Id-ul intersectiei sa fie accesibil dupa salvare pentru a putea fi incarcata corect la simulare, fara a fi nevoie sa se creeze o noua intersectie duplicat.

Actiuni efectuate:

Am memorat id-ului intersectiei intr-o variabila globala dupa salvarea intersectiei si l-am pentru a accesa pagina corecta de simulare.

Commit: [ccb9845](#)

Design patterns

Singleton Pattern

Clasa TrafficSimulator este implementata folosind design pattern-ul Singleton, care asigura existenta unei singure instante a simulatorului de trafic pe durata rularii aplicatiei.

Am ales acest pattern pentru a avea un punct unic de control al intregului sistem de simulare a traficului. Astfel, toate componentele aplicatiei interactioneaza cu aceeasi instanta, asigurand consistenta datelor si evitand probleme legate de sincronizare.

Implementarea utilizeaza o proprietate statica instance in cadrul clasei TrafficSimulator si o metoda statica getInstance() care asigura accesul la aceeasi instanta pe toata durata aplicatiei. Constructorul este conceput astfel incat sa previna crearea mai multor instante, returnand instanta existenta daca aceasta a fost deja initializata.

```

export class TrafficSimulator {
  constructor() {
    if (TrafficSimulator.instance) {
      return TrafficSimulator.instance;
    }
    this.isSimulationActive = false;
    this.routeFlows = new Map();
    this.carGenerationIntervals = new Map();
    this.routes = [];
    this.uiPanel = null;
    this.intersections = [];
    this.drawSceneCallback = null;
    this.routeCarCounters = new Map();

    resetContorMasini();

    TrafficSimulator.instance = this;
  }

  static getInstance() {
    if (!TrafficSimulator.instance) {
      TrafficSimulator.instance = new TrafficSimulator();
    }
    return TrafficSimulator.instance;
  }
}

//instanta globala unica a simulatorului
export const trafficSimulator = TrafficSimulator.getInstance();

```

Daca TrafficSimulator nu ar fi fost implementat ca Singleton si ar fi fost create mai multe instante in aplicatie, s-ar fi putut intampla urmatoarele situatii problematice:

- Fiecare instanta ar gestiona separat fluxurile de trafic si contorul de masini, ceea ce ar duce la inconsistente in simulare. De exemplu, o instanta poate considera ca pe o ruta au trecut 10 masini, iar alta, in acelasi timp, poate avea un contor diferit pentru aceeasi ruta.
- Resursele folosite pentru generarea masinilor, timerele si animatiile ar fi duplicat, crescand consumul inutil de memorie si riscand conflicte intre setarile simultane ale mai multor simulatoare.

Model, view, template pattern

Proiectul este realizat in framework-ul Django, care se bazeaza pe design pattern-ul MODEL – VIEW – TEMPLATE.

- **Model:** definește structura și comportamentul datelor (sunt practic clase corespunzatoare tabelelor din baza de date)
- **View:** primește cererea, interacționează cu modelul și pregătește datele pentru prezentare.
- **Template:** definește “cum” arată rezultatul HTML (fișiere HTML in care se vor incarca dinamic la runtime elemente)

Model:

Proiectul nostru utilizeaza un singur model custom: Intersectie_salvata, clasa ce corespunde tabelului de intersectii salvate din baza de date. O prezentam mai jos:

```
class IntersectieSalvata(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    nume = models.CharField(max_length=100)
    data = models.JSONField() # PostgreSQL native JSON field
    data_adaugare = models.DateTimeField(auto_now_add=True)

    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name="intersectii", default = None)
```

(Modelul User este oferit default de Django si nu a necesitat implementarea de capuri custom suplimentare, asdar nu a trebuit extins printr-un model custom.)

Views/templates:

La introducerea unui URL specific in browser, este apelat view-ul care incarca pagina html corespunzatoare. URL-urile ce definesc proiectul sunt listate mai jos.

```
urlpatterns = [
    path('home', views.home, name='home'),
    path('profile', views.profile, name='profile'),
    path('signup', views.signup_view, name='signup'),
    path('login', views.login_view, name='login'),
    path('aboutus', views.aboutus, name='aboutus'),
    path('logout', views.logout_view, name='logout'),
    path('changepassword', views.change_password_view, name='changepassword'),
    path('game', views.game, name='game'),
    path('create', views.create, name='create'),
    path('saved/', views.salvare_intersectie, name='saved'),
    path('incarca/<uuid:id>/', views.incarca_intersectie, name='incarca'),
    path('simuleaza_intersectie/<uuid:id>/', views.simuleaza_intersectie, name='simuleaza_intersectie'),
    path("js_to_py", views.primeste_grupe_semafor, name="primeste_grupe_semafor"),
    path('simuleaza/', views.simuleaza, name='simuleaza')
]
```

View-urile importante ce definesc proiectul si **template**-urile incarcate de acestea sunt prezentate in continuare:

- home – incarca ‘home.html’
- profile – incarca ‘profile.html’, pagina corespunzatoare “My account”, in care se insereaza intersectiile utilizatorului curent, preluate din baza de date si filtrate dupa owner

```
@login_required
def profile(request):
    intersectii = IntersectieSalvata.objects.filter(user=request.user).order_by('-data_adaugare')
    return render(request, 'profile.html', {
        'intersectii': intersectii
    })
```

- about_us – incarca template-ul ‘about_us.html’
- create – incarca template-ul ‘create.html’ si forteaza pagina sa fie reincarcata la fiecare refresh, evitandu-se probleme cu cache-ul – vezi parametrul ‘timestamp’ din render()

```
@login_required(login_url='login')
def create(request):
    return render(request, 'create.html', {
        'timestamp': now().timestamp(), # sau int(time.time())
    })
```

- signup si login:

```
def signup_view(request):
    if request.method == 'POST':
        form = SignUpForm(request.POST)

        # Verificare validitate formular
        if form.is_valid():
            user = form.save()
            login(request, user) # autentificare automata dupa inregi
            return redirect('home')
        else:
            form = SignUpForm()

    print(form)
    return render(request, 'signup.html', {'form': form})
```

```
def login_view(request):
    if request.method == 'POST':
        form = CustomAuthenticationForm(data=request.POST, request=request)
        if form.is_valid():
            user = form.get_user()

            login(request, user)
            if not form.cleaned_data.get('ramane_logat'):
                request.session.set_expiry(0)
            else:
                request.session.set_expiry(24*60*60) # 1 zi

            request.session['user_data'] = {
                'username': user.username,
                'first_name': user.first_name,
                'last_name': user.last_name,
                'email': user.email
            }
            return redirect('profile')
        else:
            form = CustomAuthenticationForm()

    return render(request, 'login.html', {'form': form})
```

- `salvare_intersectie` – este un view ce nu incarca template-uri, ci are rolul de a transfera date de la frontend la backend. La click pe butoanele “Salvare intersectie” sau “Simuleaza” (in cazul in care intersectia nu e deja salvata), se creeaza un json cu datele intersectiei trimise in body-ul obiectului request, iar acesta se salveaza in baza de date. Daca intersectia a fost deja salvata, iar acum s-au efectuat doar modificari asupra ei, se modifica campul data al intersectiei deja salvate in baza de date.

```
def salvare_intersectie(request):
    if request.method != "POST":
        return JsonResponse({ "error": "Metodă nepermisă" }, status=405)

    try:
        body = json.loads(request.body)
        data = body.get("data")
        id_intersectie = body.get("id")

        if not data:
            return JsonResponse({ "error": "Lipsește câmpul 'data'" }, status=400)

        if id_intersectie:
            try:
                intersectie = IntersectieSalvata.objects.get(id=id_intersectie, user=request.user)
                intersectie.data = data
                intersectie.save()
                return JsonResponse({ "status": "actualizat", "id": str(intersectie.id) })
            except IntersectieSalvata.DoesNotExist:
                return JsonResponse({ "error": "Intersecția nu a fost găsită sau nu aparține utilizatorului." }, status=404)
        else:
            # optional: fallback dacă nu a fost niciodată salvată
            nume = body.get("nume") or "fara nume"
            intersectie = IntersectieSalvata.objects.create(user=request.user, nume=nume, data=data)
            return JsonResponse({ "status": "creat", "id": str(intersectie.id) })

    except Exception as e:
        return JsonResponse({ "error": f"Eroare internă: {str(e)}" }, status=500)
```

- incarca_intersectie – cauta intersectia ceruta in request in baza de date si o returneaza in format json

```
def incarca_intersectie(request, id):
    try:
        intersectie = IntersectieSalvata.objects.get(id=id, user=request.user)
        return JsonResponse(intersectie.data, safe=False)
    except IntersectieSalvata.DoesNotExist:
        raise Http404("Intersecția nu a fost găsită sau nu îți aparține.")
```

- simuleaza_intersectie – incarca template-ul 'simuleaza.html' cu datele intersectie pentru care se face simularea, primita in request

```
def simuleaza_intersectie(request, id):
    if request.method == 'POST':
        try:
            payload = json.loads(request.body)
            data = payload.get("data", {})

            return render(request, 'simuleaza.html', {
                'intersectie_id': id,
                'data': mark_safe(json.dumps(data))
            })
        except json.JSONDecodeError:
            return HttpResponseBadRequest('Invalid JSON')
```

- primeste_grupe_semafor – view folosit in simularea traficului pe baza predictiei modelului de RL, ce are nevoie sa primeasca timpii asociati starii de verde a unei grupe de semafoare ce controleaza trasee compatibile intre ele

```
def primeste_grupe_semafor(request):
    if request.method == "POST":
        try:
            data = json.loads(request.body)
            print("Am primit din JS:", data)

            model, env = load_model()
            results = test_utilizator(model, env, data)
            try:
                results = [int(round(x)) for x in results]
            except Exception as e:
                print("✗ Eroare la convertirea in int:", str(e))
                results = []
            print(results)
            return JsonResponse({"status": "ok", "results": results})
        except Exception as e:
            return JsonResponse({"error": str(e)}, status=400)

    return JsonResponse({"error": "Metodă invalidă (doar POST)"}, status=405)
```

Prompt engineering: documentarea folosirii toolurilor de AI in timpul dezvoltarii software

Proiectul nostru include intr-o proportie de aproximativ 50% cod generat sau corectat cu ajutorul unor LLM-uri precum ChatGPT, Copilot sau Claude.

Alegerea utilizarii lor a survenit nevoii de a lucra predominant in JavaScript (ce a venit la pachet cu dezavantaje precum sintaxa bazata pe functii lambda si lipsa de experienta cu acest limbaj a membrilor echipei). In acest sens, tool-urile de AI au fost de un real folos.

O alta motivatie pentru utilizarea LLM-urilor in generarea de cod a fost dimensiunea ambitioasa a proiectului si timpul scurt in care acesta a trebuit realizat.

Atasam mai jos cateva screenshot-uri din conversatiile cu LLM-uri in care acestea ne-au ajutat sa:

- cream functii

- tratam bug-uri
- descoperim sursa bug-urilor
- intelegem cum se utilizeaza anumite functii JavaScript
- transferam date intre frontend si backend
- instalam biblioteci necesare in mediul de lucru

ChatGPT 4o

cum as putea face testele cu simulări și salvări în UI? care sunt pașii pentru implementarea acestor teste? ce am de făcut? unde scriu cod? etc

Excelent! Hai să detaliez pas cu pas cum faci teste automate pentru **simulări** și **salvări** în UI, exact cum ai întrebat — folosind **Cypress**, unul dintre cele mai populare și simple tool-uri pentru testare E2E (end-to-end).

🟢 **1. Ce e Cypress și de ce e potrivit**

Cypress este un framework de testare frontend care:

- rulează direct în browser,
- poate simula clickuri, tastări, navigări,
- oferă feedback vizual (vezi testul rulând ca un user).

🛒 **2. Ce trebuie să ai înainte să începi**

- Aplicația ta frontend (ex: React, Vue, JS vanilla) să ruleze local, ex: `http://localhost:3000`

ChatGPT 4o

```
setTimeout(() => {
  window.location.href = `"% url 'simuleaza' %";
}, 1000);

} catch (error) {
  console.error("Eroare la încărcarea intersecției", error);
}
```

Eroarea apare doar când folosești intersecțiile predefinite salvate local, dintr-un motiv simplu: după redirecționare, nu ai un backend funcțional la ruta `/js_to_py`, iar `fetch("/js_to_py", {...})` returnează o pagină HTML (404/500) în loc de JSON.

🌟 **Ce se întâmplă pas cu pas:**

1. Tu apelezi `fetch("/static/intersections/num.json")` — totul OK.
2. Salvezi intersecția în `localStorage`.
3. Redirecționezi cu

js

↓

Copiază Editează

ChatGPT 4o

nu, nu vreau vectorul din fetch, vreau vectorul din funcția respectivă

Multumesc pentru clarificare! 🙏

Dacă vrei să folosești `results` din funcția `primește_grupe_semafor()` în `simuleaza_intersecție()`, ai două opțiuni reale:

🟢 **SOLUȚIA CORECTĂ: Folosește `request.session` — DAR CORESPUNZĂTOR!**

Ai făcut bine în principiu, dar probabil nu funcționează pentru că:

- nu ești autentificat,
- sau `primește_grupe_semafor()` e `@csrf_exempt`, deci sesiunea nu e salvată corespunzător.

🔗 **Pași corecți pentru a salva și recupera vectorul:**

1. 🛡️ **Scoate `@csrf_exempt` (dacă folosești token CSRF în `fetch`)**

ChatGPT o4-mini

pot face un eventListener ca la hover pe un element sa se intample ceva?

Da, sigur! Poți folosi fie CSS, fie JavaScript pentru asta. Dacă vrei neapărat un listener în JS, iată un exemplu simplu:

```
html
<div id="myElement" style="width:200px; height:100px; background:#ccc;">
  Trece mouse-ul peste mine
</div>

js
const el = document.getElementById("myElement");

// Când mouse-ul intră în element
el.addEventListener("mouseenter", () => {
  el.style.backgroundColor = "lightgreen";
  console.log("Hover start!");
});

// Când mouse-ul părăsește elementul
```

ChatGPT o4-mini

WARNING: ERROR: Could not find a version that satisfies the requirement stablebaselines3 (from versions: none) ERROR: No matching distribution found for stablebaselines3 PS C:\Users\cristi\OneDrive\Desktop\MD5_Project\Traffic>

S-a gândit timp de două secunde >

The error `ModuleNotFoundError: No module named 'stable_baselines3'` and the subsequent "no matching distribution" from pip both occur because you're trying to install the wrong package name and on a Python version for which no wheel has been published yet.

1. Install the correct package

The PyPi name is `stable-baselines3` (with hyphens), not `stablebaselines3`:

```
bash
pip install stable-baselines3
```

If you still get "no matching distribution," it means pip can't find any pre-built wheels for your Python version.

ChatGPT o4-mini

```
// Funcție pentru a curăța toate mașinile
export function stergeMasini() {
  clearMasini();
}

// Exporta funcția pentru simulare
export function simuleazaTrafic(intersecți, numarMasini = 5) {
  return genereareMasiniPeTraseeleSalvate(intersecți, numarMasini);
}

acesta e traficul
```

S-a gândit timp de două secunde >

Nu vei găsi vreun `while(true)` sau vreun `setTimeout` înfinit chiar în `traffic.js` — însă totmai din acest fișier legi `draiscene` în bucla de animație a mașinilor. Bucata vinovată e totmai apelul către:

```
js
// În traffic.js
export function initTrafic(draisceneFunc) {
  // ...
  setImmediate(() => draisceneFunc());
}
```

Github Copilot

Implementarea unui UI pentru setarea flow-ului de mașini în Traffic Simulator

Pentru a implementa un UI care să permită setarea flow-ului de mașini pentru fiecare rută definită în JSON-ul primit, poți urma acești pași:

1. Crearea unui panel de control în interfața utilizatorului

Adaugă acest HTML în template-ul pentru traficul simulator -->

```
<div id="traffic-flow-panel" class="control-panel">
  <h3>Setarea Flow Control</h3>
  <div id="routes-container">
    <!-- Acolo vor fi adăugate dinamic rările -->
  </div>
</div>
```

2. Adăugarea stilurilor CSS pentru nod panel

```
/* Stiluri CSS pentru panelul de control */
.control-panel {
  background-color: #f0f0f0;
  border-radius: 10px;
  padding: 10px;
  box-shadow: 0 5px 15px #ccc;
}
```

Dezavantajele utilizării tool-urilor de AI:

Deși prezintă toate beneficiile menționate anterior, utilizarea acestor instrumente a cauzat multe probleme de-a lungul proiectului.

Principala problemă care apare în momentul generării de cod cu AI este tentația de a introduce bucăți (uneori masive) de cod direct în proiect după o privire (trebuie să recunoaștem) superficială asupra sa.

Au fost multe momente în care ni s-a părut că înțelegem cod generat, însă acesta avea un comportament complet neașteptat la rulare. În astfel de momente a trebuit să revenim la forma anterioară a proiectului și să regândim logica sau să corectăm codul generat.

O altă problemă cu care ne-am confruntat a fost faptul că modelul Claude prezintă opțiunea de a insera automat, chiar în directorul de lucru cod generat. Ceea ce la început a părut un instrument salvator de timp și care va crește acuratețea predicției LLM-ului s-a dovedit a fi un mare dezavantaj.

Claude a creat, de exemplu o logică atât de complicată, haotică și... ciclică pentru simularea traficului (cu funcții din fișiere și clase diferite care se apelează în buclă, cu bucăți de cod inserate în zone irelevante pentru taskul curent etc. etc.) încât ne-a fost imposibil să înțelegem logica, să o corectăm și să o extindem. A trebuit așadar, cu 2 săptămâni înainte de deadline!, să refacem de la zero ceva ce părea terminat.

Sistematizare a problemelor întâmpinate:

- dificultatea de a depista sursa erorilor în codul generat de AI
- limitarea flexibilității pe termen lung