ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ ҒЫЛЫМ ЖӘНЕ БІЛІМ МИНИСТРЛІГІ

Қ.И. Сатпаев атындағы Қазақ ұлттық техникалық зерттеу университеті

Автоматика және ақпараттық технологиялар институты

Бағдарламалық инженерия кафедрасы

Зәкір Кенесары Тәжібайұлы

МАГИСТРЛІК ДИССЕРТАЦИЯ

Магистр академиялық дәрежесін алу үшін

Диссертация тақырыбы	Такси қызметіне арналған көлік құралдарын бақылау жүйесі
Мамандық бағыты	7M06101 – Software Engineering

Стандартизация/Нормоконтроль
PhD, ассистент профессор
_____А.Т. Ахмедиярова
«20» 05 2022 г.

Оппонент PhD, профессор
_____ М.Е.Мансурова
«01» 06 2022 г.

Ғылыми жетекшісі
PhD, профессор ассистенті
_____ Ж.М. Алибиева
«30» 05 2022г.

ҚОРҒАУҒА ЖІБЕРІЛДІ
Кафедра меңгерушісі
физ.-мат.ғыл кандидаты,
профессор
_____А.Н. Молдагулова
«02» 06 2022 г.

Алматы 2022

ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ ҒЫЛЫМ ЖӘНЕ БІЛІМ МИНИСТРЛІГІ

Қ.И. Сатбаев атындағы Қазақ ұлттық техникалық зерттеу университеті

Автоматика және ақпараттық технологиялар институты

Бағдарламалық инженерия кафедрасы

Мамандық: 7М06101 – Software engineering

БЕКІТЕМІН
БИ Кафедра меңгерушісі
физ.-мат. ғыл кандидаты,
профессор
_____ А.Н.Молдагулова
«02» 06 2022г.

Магистрлік диссертацияны орындауға арнлаған
ТАПСЫРМА

магистрант Зәкір Кенесары Тәжібайұлына
Диссертация тақырыбы: «Такси қызметіне арналған көлік құралдарын бақылау жүйесі».
Аяқталған диссертацияны тапсыру уақыты «___»_____

Магистрлік диссертацияның бастапқы деректері. Такси қызметіне арналған көлік құралдарын бақылау жүйесіне арналған бағдарламалық жасақтаманы әзірлеу кезіндегі алгоритмдер мен оң шешімін беретін модельдер ұсынылып, толығырақ сипатталды. Зерттеу жұмысының қойылған мақсаттарының ішінде: жүйедегі негізгі рөл атқаратын компоненттердің алгоритмдерін зерттеп, жеке алгоритм құрылды ; зерттелген жұмысты талдап, соған байланысты бағдарламалық жасақтамасы әзірленді.

Магистрлік диссертацияда әзірленетін сұрақтар тізімі немесе магистрлік жұмыстың қысқаша мазмұны: а) зерттеу алды жоба жұмысымен танысу; ә) бақылау жүйесінің әзірлеу модельдері мен алгоритмдері; б) ұтымды алгоритм жасау барысындағы зерттеу нәтижелері; в) зерттеу нәтижелерін талдап соған байланысты бағдарламалық жасақтаманы әзірлеу
Ұсынылатын негізгі қолданылған әдебиеттер: Taxi fuel consumption and emissions estimation model based on the reconstruction of driving trajectory / Jiancheng Weng, Quan Liang, Zhihong Chen // Sagepub Journals – 2017. – С. 8-11; Efficient Shortest Path Query Processing in Dynamic Road Networks/ Mengxuan Zhang // Control Theory and Control Engineering. – 2021. – С. 43-68.; Novel online routing algorithms for smart people-parcel taxi sharing services / Son Nguyen Van, Anton Dries, Behrouz Babaki // ETRI Journal Wiley. – 2022. – С. 1-12.; Taxi Operational Status Real Time Monitoring System based on seat sensing / Zhongqing

Huang // International Conference on Intelligent Systems Research and Mechatronics Engineering (ISRME 2015), 2015. – С. 1-4.

Магистрлік диссертацияны дайындауға арналған

ГРАФИК

Бөлімдердің атауы, әзірленетін сұрақтар тізімі	Ғылыми жетекшіге ұсыну мерзімі	Ескертулер
Бөлім 1.	30.10.2021 г.	Орындалды
Бөлім 2.	13.11.2021 г.	Орындалды
Бөлім 3.	20.11.2021 г.	Орындалды

Көрсетілген жоба бойынша бөлімдерге байланысты консультациялар

Бөлім	Консультант, (ғылыми дәрежесі)	Уақыт	Қолы
Стандартизация Нормконтроль	PhD, профессор ассистенті А.Т. Ахмедиярова	29.05.2022	
Программное обеспечение	PhD, қауымдастырылған профессор Н.К. Мукажанов	29.05.2022	

Тапсырманың берілген күні "02" 09 2021 г.

Кафедра меңгерушісі _____ А.Н. Молдагулова

Ғылыми жетекшісі _____ Ж.М. Алибиева

Тапсырманы орындауға магистрант қабылдады _____ К. Т. Зәкір

Күні "03" 06 2022 г.

АҢДАТПА

Бұл диссертациялық жұмыста такси қызметіне арналған көлік құралдарын бақылау жүйесін құру тұжырымдамасы таныстырылады. Көлік құралдарының заманауи және сенімді GPS мониторингі және спутниктік навигацияны қолдану көлік құралдарының қозғалысын, жүргізушілердің сапалы жұмысын бірыңғай басқаруға мүмкіндік береді. Такси қызметіне арналған көліктерге спутниктік бақылау технологияларын қолдану, оның барлық процесстерін автоматтандырып, басқару мәселелерін шешу арқылы көлік жұмысының тиімділігін арттыруды қамтамасыз етеді. Көлік құралдарына қатысты технологияларды қолдану арқылы көліктің нақты жағдайын бақылау жүйесімен, яғни мобильді қосымша немесе веб қосымша арқылы тұрақты түрде қадағалап отыруға болады. Жалпы айтқанда, бұл диссертациялық жұмыста нарықтағы такси қызметіндегі көліктерді қадағалаудағы өзекті мәселелерді шешетін бірегей бақылау жүйесін құру туралы толық ашып айтылатын болады. Жұмыс барысында нарықтағы такси кәсіпорындарында қолданылатын алгоритмдер қарастырылады және сәйкесінше оптимальді шешім ретінде жүйеде жұмыс істейті жаңа алгоритм жасалынады.

Түйінді сөздер**:** мониторинг, такси, агрегатор, Javascript, Framework.

АННОТАЦИЯ

В данной диссертации представлена концепция создания системы управления транспортным средством для службы такси. Использование современного и надежного GPS-мониторинга автотранспорта и спутниковой навигации позволяет осуществлять комплексный контроль дорожного движения, качественную работу водителей. Использование технологий спутникового мониторинга такси позволит повысить эффективность работы транспорта за счет автоматизации всех его процессов и решения управленческих задач. Используя технологии, связанные с транспортными средствами, фактическое состояние транспортного средства можно постоянно контролировать с помощью системы мониторинга, т.е. мобильного приложения или веб-приложения. В целом в данной диссертации будет подробно освещено создание уникальной системы управления, которая позволит решать насущные вопросы на рынке услуг такси. В ходе работы рассматриваются алгоритмы, используемые в таксомоторных компаниях рынка и, соответственно, разрабатывается новый алгоритм, работающий в системе, как оптимальное решение.

Ключевые слова: мониторинг, такси, агрегатор, Javascript, Framework.

ANNOTATION

This dissertation presents the concept of creating a vehicle management system for a taxi service. The use of modern and reliable GPS-monitoring of vehicles and satellite navigation allows for a comprehensive control of traffic, quality work of drivers. The use of satellite monitoring technology in taxis will improve the efficiency of transport by automating all its processes and solving management problems. Using vehicle-related technologies, the actual condition of the vehicle can be constantly monitored by a monitoring system, i.e. mobile application or web application. In general, this dissertation will cover in detail the creation of a unique management system that will allow solving pressing issues in the taxi services market. In the course of the work, the algorithms used in the taxi companies of the market are considered and, accordingly, a new algorithm is developed that works in the system as an optimal solution.

Keywords: monitoring, taxi, aggregator, Javascript, Framework.

МАЗМҰНЫ

КІРІСПЕ

Тақырыптың өзектілігі. Жүргізушілердің жұмысын бақылау үшін және бизнестегі процесстерді статистикалық мәліметтер арқылы есеп алып, кәсіпті қолайлы және оңай түрде басқару үшін көптеген автопарк иелері көлік құралдарын бақылау жүйесін пайдаланады. Бұл такси көліктерін бақылау мен тиімділігін арттыруға, жанармай мен техникалық қызмет көрсету шығындарын азайтуға, сондай-ақ жолаушылармен мен такси қызметкерлерін қорғауға мүмкіндік береді. Көлік құралдарын бақылау жүйесін пайдалану такси нарығындағы бәсекеге қабілеттілікті арттыруға көмектеседі.

Жобаның мақсаты: Көлік құралдарының заманауи және сенімді GPS мониторингі және спутниктік навигацияны қолдану арқылы көлік құралдарының қозғалысын, жүргізушілердің сапалы жұмысын бірыңғай басқаруды қамтамасыз ету. Көлік құралдарының жұмыс уақытындағы іс-әрекеттерді бақылайтын бірегей жүйе әзірлеу.

Жоба тапсырмалары:

- зерттеу барысында жасалынатын жүйені нарықтағы басқа да такси және такси көлік құралдарын бақылайтын бағдарламаларымен салыстырып айырмашылықтарын айқындау ;
- жүйедегі негізгі рөл атқаратын компоненттердің алгоритмдерін зерттеп, жеке алгоритм құру ;
- зерттелген жұмысты талдап, соған байланысты бағдарламалық жасақтамасын әзірлеу.
- әзірленген модель мен алгоритм көрсеткіштері бойынша эксперименттік зерттеулер жүргізу;
- магистрлік жұмысты дайындау және рәсімдеу.

Зерттеудің ғылыми жаңалығы келесідей:

Жобаның бірегейлігі ретінде көлік құралдарын бақылайтын жаңа автоматтандырылған жабдықтар мен қоса сол бақыланған ақпараттарды қарауға арналған мобилді қосымшаны айтуға болады. Жүргізуші жол үстінде белгілі бір командаларды орындау арқылы көлік туралы барлық ақпараттарды біле алады, және сол ақпараттар бұлттық технологияда жасалынған дерекқорында сақталынады, кейін сол мәліметтерді қосымшадан архив түрінле қарауға мүмкіншілігі бар. Бұл өз кезегінде такси қызметін жаңа деңгейге шығаратын инновациялық жүйе болады

Практикалық құндылығы

Жоба нәтижелері мыналарды жүзеге асыруға мүмкіндік береді:

- Компания мен таксист және жүргізіп жүрген көлігінін арасында біртекті үздіксіз байланыс
- Ыңғайлы интерфейстегі қосымша
- Көлік құралдарын бақылап , ақпараттарды ұқыпты сақтау

Теоретикалық құндылықтары такси қызметіне арналған көлік құралдарын бақылау жүйесінің жасақтамасын жасау және нарықтағы бар алгоритмдер мен жүйеге енгізілетін жаңа алгоритмді салыстырып, қолданысқа түсіру болып табылады. Тақырыптың өзекті мәселесін шешу үшін

модификацияланған Дейкстер және А* эвристикалық алгоритмдері қолданылды, зерттеу нәтижелеріне сәйкес екі алгоритмді біріктіретін гибридті алгоритм ұсынылды және ол жақсы нәтиже беретініне көз жеткізілді.

Жұмыс апробация. Диссертациялық зерттеу жұмысы «Сәтпаев оқулары – 2022. Қазіргі ғылыми зерттеулердің трендтері» атты халықарлық-ғылыми конференцияға «Такси қызметіне арналған көлік құралдарын басқару жүйесі» тақырыбымен қатысты.

Диссертация құрылымы. Диссертациялық жұмыс үш негізгі бөлімнен тұрады, кіріспе, негізгі бөлім, қорытынды, пайдаланылған әдебиеттер тізімі және қосымша (бағдарлама код листингі). Материалдар 63 бетке жазылды, 5 кестелер, 27 суреттен, және де 1 қосымша листинг кодынан тұрады. Қолданылған әдебиеттер саны 30.

Зерттеу алды жоба жұмысы

1.1 Такси қызметіне арналған көлік құралдарын бақылау жүйесінің жұмыс істеу принципі

Такси қызметіне арналған көліктерге спутниктік бақылау технологияларын қолдану, оның барлық процестерін автоматтандырып, басқару мәселелерін шешу арқылы көлік жұмысының тиімділігін арттыруды қамтамасыз етеді. Заманауи навигациялық және телекоммуникациялық технологиялар классикалық қоғамдық көлік диспетчерлік жүйесінің жұмысында қолданылады.

Бақылау жүйесінің негізгі мақсаты

Жұмыс кестесін сақтау және маршруттар бойынша бүкіл парктің өндірістік бағдарламасын орындау мақсатында ағымдағы жағдайдың факторларын ескере отырып, көлік құралдарын жедел басқару және диспетчерлеу. Жекелеген көліктердің ақаулығы немесе олардың техникалық ақаулығы жағдайындағы мәселелерге жедел әрекет ету.

Бақылау жүйесінің негізгі міндеттері

Такси қызметіне арналған көліктерге бақылау жүйесін енгізу келесі міндеттерді шешуге ықпал етеді:

- диспетчерлік бақылау және такси көліктерін орталықтандырылған басқару
- жолаушыларды тасымалдау саласындағы қызметтердің сапасын бақылау
- көлік құралдарының жұмысы туралы жедел деректерді алу арқылы клиенттермен және қызметкермен даулы жағдайларды шешу мүмкіндігі
- көлік құралдарына қызмет көрсету және техникалық қызмет көрсету шығындарын минимизациялау
- такси көлік тасымалдаушыларының пайдасын максимизациялау
- кәсіпорынның бәсекеге қабілеттілігін арттыру
- тоқтап қалудың, жұмыс уақытын жоғалтудың алдын алу арқылы кәсіпорынның экономикалық көрсеткіштерін арттыру
- басқару мен қызмет көрсету тиімділігін арттыру
- көлік құралдарының қозғалу заңдылығын арттыру
- жолаушыларға көрсетілетін көлік қызметінің сапасын арттыру
- атқарушы органдармен өзара әрекеттесуді оңтайландыру

Бақылау жүйесінің жұмыс істеу принципі

- Бақылау объектілері - такси көлігі навигациялық жүйелерден және GPS-тен алынған географиялық координаттар туралы ақпаратты қабылдайтын және өңдейтін мамандандырылған борттық навигациялық-коммуникациялық жабдықтармен жабдықталады.
- Жабдық барлық деректерді қабылдағаннан және өңдегеннен кейін, ақпарат нақты уақыт режимінде, тәулік уақыты мен ауа-райының жағдайына қарамастан, сымсыз байланыс арқылы телематикалық серверге жіберіледі.

- Арнайы бағдарламалық жасақтаманың көмегімен бақылау жүйесі жедел және нақты уақыт режимінде бақылау процестерін және жүргізушілердің жұмысын басқарады, төтенше жағдайлар кезінде дабыл хабарламаларын және реттеуші шаралар қабылдайды, объектілердің техникалық параметрлерін бақылайды, статистика мен есепке алуды жүргізеді, толық жағдайды сараптап, талдайды.

Деректерді қабылдау және беру технологиялары

- Таксиге арнаған көліктің бақылау жүйесі деректерді қабылдау және беру үшін инновациялық технологияларды қолданады:
- Навигациялық жүйелер мен GPS-тің ғаламдық орналасу жүйесінің спутниктері объектінің географиялық координаттары туралы ақпаратты есептеуге және өңдеуге мүмкіндік беретін сигналдар жібереді.
- Ұялы байланыс желілері басқарылатын объектілер мен сервердің навигациялық жүйелері арасында мәліметтер алмасуды қамтамасыз етеді
- Дүниежүзілік ғаламтор желісі телематикалық сервер, пайдаланушының компьютері және байланыс жүйелері арасында мәліметтер алмасуды қамтамасыз етеді

1.2 Көлік құралдарын бақылау жүйесін қолдана отырып, автомобиль паркін тиімді пайдаланудың негізгі мүмкіндіктері

- Жанармай шығынын бақылау. Жанармай ағызуды жою.
- Көлікті және басқа мақсаттарды рейстерді пайдалануды жою
- Берілген маршруттан ауытқуды бақылау
- Жылдамдық шектерінің сақталуын бақылау
- Көлік құралдарының бос тұрып қалуын азайту
- Көлік кептелісі және айналма жолдар туралы жүргізушіні хабардар ету мүмкіндігі
- Кептелістерді ескере отырып, маршруттарды ұтымды жоспарлау

Көліктік бақылау жүйесі көлік құралдарының визуалды графикалық дисплейі мен қозғалыс статистикасын ескере отырып, қозғалыс жағдайын (мысалы, кептелістер) және басқа да факторларды ескере отырып, қозғалыс маршруттарын ұтымды құруға және туындайтын жағдайларға жедел тұрақты және анормальды түрде ден қоюға мүмкіндік береді.

Көлік құралдарын бақылау жүйесі диспетчер мен көлік құралы арасында хабарлама алмасуға мүмкіндік береді, бұл жүктелген міндеттерді шешу тиімділігін едәуір арттырады және жүргізушінің де, жүктің де қауіпсіздігін арттырады. СМС хабарламаларын пайдалана отырып, бақылау жүйесінің диспетчері тапсырма қоюдан бастап трафик жағдайына дейін барлық қажетті ақпаратты жүргізушіге бере алады. Хабарламаларды бақылау жүйесінің SMS шлюзі арқылы жүргізушінің телефонына тікелей жіберуге болады, бұл қосымша

жабдық шығындарын қажет етпейді және диспетчерлік орталықтың дауыстық және SMS трафигінің құнын төмендетеді.

Персонал өздерінің барлық әрекеттерін көліктік бақылау жүйесімен басқарылатындығын түсініп, жұмыс уақытында өзін-өзі тәртіпті ұстауға мәжбүр болады.

Кәсіпорында енгізілген көліктік бақылау жүйесі көлік құралын мақсатсыз пайдаланумен байланысты шығындарды (есепке алынбаған сапарлар) және көрсетілген маршруттардан ауытқуды айтарлықтай төмендетеді, бұл өз кезегінде жоғарыда сипатталған пайдалану шығындарының төмендеуіне әкеледі. Сонымен қатар, көлік құралдарын бақылау жүйесі жанармай құю мен ағызуды бақылауға мүмкіндік беретіндіктен, персонал бұдан былай «қосымша кірістің» түріне жүгінбейді.

Көлік жүргізу қауіпсіздігі факторлары

- Көлік құралының қозғалысын және аялдамаларын бақылау
- Көліктің көрсетілген маршруттардан ауытқуын жедел бақылау
- Көлік құралдарының көрсетілген геоаймақтарға кіруін және шығуын бақылау
- Қосымша датчиктерді қосу арқылы көлік жүйелері мен жүктердің күйін бақылау
- Бір жүргізушінің үздіксіз жүруін қоса алғанда, жұмыс уақытының мониторингі
- Драйвер кез келген уақытта SOS ескерту хабарламасын жібере алады
- Жүргізушімен екі жақты байланыс мүмкіндігі

Бақылау жүйесі жүргізушінің де, жүктің де қауіпсіздігін жақсартады. Көлік қауіпсіздігі бірінші кезекте персоналдың тәртіптілігіне және белгіленген тасымалдау ережелерін сақтауға байланысты. Тек белгілі бір жанармай құю бекеттерінде жанармай құю, қорғалатын аумақтарда тоқтау және түнеу сияқты талаптар қауіпсіздікті қамтамасыз ету үшін міндетті болып табылады және компанияның көлік паркін пайдалану кезінде қатаң түрде орындалуы керек, өйткені көлік құралдарын бақылау жүйесі тұрақ пен жанармай бекеттерін басқаруға мүмкіндік береді.

Корпустың тоңазытқыш бөліміне температура сенсорын орнату арқылы температура режимін басқаруға болады және қажет болған жағдайда осы параметрдің басқару мәнінен төмен немесе жоғары өзгергені туралы хабарлама алады, бұл өз кезегінде жүктің зақымдану қаупін төмендетеді. Автокөлік құралдарын бақылау кез-келген көлік жүйесіне қосымша датчиктер орнатуға және пайдалануға мүмкіндік береді: дәл отын датчиктері, есіктерді ашу немесе жабу, багаж, корпус аудару, толтыру және төгу мойындары, температураны бақылау және тағы да басқалар.

Жүргізу уақытын бақылау және жүргізушілердің қатты шаршауын болдырмау жол-көлік оқиғасын айтарлықтай азайтады және осылайша көлік құралы мен жүк жоғалту қаупін азайтады.

1.3 Көлік құралдарын бақылау жүйесінің өзектілігі (нақты статистикалар)

GPShome бақылау жүйесін пайдаланушылардың статистикасы көрсеткендей, көлік құралдарын сауатты және дәйекті бақылаумен жанармай шығыны 25-30% -ға төмендеді. Осы сандарды пайдалана отырып, бақылау жүйесін енгізудің артықшылықтарын анықтауға болады. Мысалы, аз тонналы көлік құралдары үшін есептеуді қарастырайық. Статистика көрсеткендей, жанармай шығыны 100 километрге орташа есеппен 20-дан 16 литрге дейін азаяды. 22 жұмыс күні ішінде 300 км орташа тәуліктік жүру кезінде айына шамамен 264 литр жанармай үнемдеуге болады, бұл литріне 45 рубль болатын айына 47520 теңге рубльді құрайды, бұл 570240 теңгені жылына үнемдеуге болады. Осылайша, тек жанармай үнемдеу көліктік бақылауды төлеуден гөрі көп нәрсе алады және көлік бөлімінің жетістігіне әсер ететін көптеген факторлар бар, оларға көліктік бақылау тікелей әсер етеді.

Қазақстанда такси нарығы белсенді дамып келеді. Оның өсуіне республикада түрлі такси агрегаторларының пайда болуы ықпал етеді. Соңғы бірнеше жылда Yandex.Taxi, Uber, Indriver және басқалары сияқты агрегаторлар қазақстандық нарықта өз жұмысын бастады. Олардың келуі тұтынушыларға автокөлікке өздеріне ыңғайлы түрде тапсырыс беруге, өздері үшін ең қолайлы тарифтерді таңдауға, жүргізушілерге тасымалдау маршруттарын оңтайландыруға, бос жүрістер мен тоқтап қалуларды азайтуға мүмкіндік берді.
2014-2018 жылдары Қазақстанда такси сапарларының саны шамамен 50% өсті: 415,2 миллион сапардан 619,7 миллион сапарға. Көрсеткіштің ең үлкен өсімі 2017 және 2018 жылдары байқалды және 19,4 және 27,7% құрады. Такси агрегаторларының таралуынан басқа, мобильді қосымшалардың танымал болуы, тасымалдаушыларға қосымша қызметтердің пайда болуы мен дамуы нарыққа оң әсерін тигізді. (1 – сурет)

1 – сурет. Такси қызметіндегі басты көрсеткіштік статистикалары, BusinesStat

[1] [1] BusinesStat – құрылымдық бизнестердің аналитикалық статистикаларын сараптайтын ұйым

BusinesStat[1] бағалауы бойынша, Қазақстандағы такси нарығы белсенді қарқынмен дами береді. 2019-2023 жылдары сапарлар санының өсу динамикасы өткен жылдармен салыстырғанда орташа 6,7% құрайды. 2023 жылға қарай такси сапарларының саны 856 миллион адамға дейін артады, бұл 2018 жылғы деңгейден 38,1% асады.

Бөлімге қорытынды

Қорытындылай келе бұл бөлімде зерттеу алды жұмысы ретінде тақырыпты ашу үшін жалпылама сипаттамалар мен статистикалық мәліметтер келтірілді. Нақтырақ айтар болсақ такси қызметіндегі көлік құралдарын басқарудағы негізгі мақсаттарының бірі: жұмыс кестесін сақтау және маршруттар бойынша бүкіл парктің өндірістік бағдарламасын орындау мақсатында ағымдағы жағдайдың факторларын ескере отырып, көлік құралдарын жедел басқару және диспетчерлеу; жекелеген көліктердің ақаулығы немесе олардың техникалық ақаулығы жағдайындағы мәселелерге жедел әрекет ету болып табылады.

Бақылау жүйесінің жұмыс істеу принципі ретінде Бақылау объектілері - такси көлігі навигациялық жүйелерден және GPS-тен алынған географиялық координаттар туралы ақпаратты қабылдайтын және өңдейтін мамандандырылған борттық навигациялық-коммуникациялық жабдықтар мен серверлер қарастырылды. Арнайы бағдарламалық жасақтаманың көмегімен бақылау жүйесі жедел және нақты уақыт режимінде бақылау процестерін және жүргізушілердің жұмысын басқарады, төтенше жағдайлар кезінде дабыл хабарламаларын және реттеуші шаралар қабылдайды, объектілердің техникалық параметрлерін бақылайды, статистика мен есепке алуды жүргізеді, толық жағдайды сараптап, талдайды.

Бақылау жүйесі жүргізушінің де, жүктің де қауіпсіздігін жақсартады. Көлік қауіпсіздігі бірінші кезекте персоналдың тәртіптілігіне және белгіленген тасымалдау ережелерін сақтауға байланысты. Тек белгілі бір жанармай құю бекеттерінде жанармай құю, қорғалатын аумақтарда тоқтау және түнеу сияқты талаптар қауіпсіздікті қамтамасыз ету үшін міндетті болып табылады және компанияның көлік паркін пайдалану кезінде қатаң түрде орындалуы керек, өйткені көлік құралдарын бақылау жүйесі тұрақ пен жанармай бекеттерін басқаруға мүмкіндік береді.

2 Көлік құралдарын бақылау жүйесінің әзірлеу модельдері мен алгоритмдері

Күнделікті жүргізушілер тас жолдарда, белгісіз бағыттар мен әртүрлі рельефтерде жүру кезінде қиындықтарға тап болады. Нақтырақ айтқанда, автомобильді қосалқы қоныстандыру, көлік ұрлау немесе жазатайым көлік оқиғалар болуы мүмкін. Белгілі бір такси сапарға шыққанда, ол межелі жерге жеткенше, көлік құралдарын бақылаушы архивтегі статистикалық есептемелерге немесе журналдарға сүйене отырып таксидің жағдайын анықтай алады және ол өз кезегінде нақты уақыт ремжимінде көліктің қауіпсіздігін қамтамасыз ете алады.

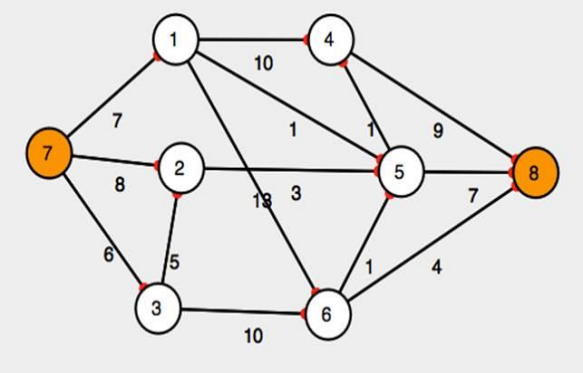
Қазіргі уақытта апаттар мен тонау санының өсуіне байланысты көлік құралын нақты уақытта бақылау маңызды. Мұндай оқиға орын алған сайын, көлік әкімшісі нақты уақыт режимінде бақылап отырса, ол хабарландырылатын жағдайға сәйкес жақын маңдағы ауруханаларға немесе полиция бөлімшелеріне дереу хабарлай алады.

Бұл диссертациялық жұмыста ұсынылатын алгоритмдер такси қызметіндегі көлемді қаражат өсімін арттыруға және такси көлік құралдарының қауіпсіздіктерін сақтауға ықпал етеді. Қысқаша айтқанда, көлік құралдарын бақылау жүйесі нақты уақыт режимінде көліктерді оңтайлы түрде бақылауды қамтамасыз ете алады. Барлық факторларын ескере отырып, ұсынылған алгоритмді қолдану арқылы драйверге ең қысқа жол көрсетіледі. Қауіпсіз жылдамдықты асырған кезде жүргізушіге ескертіледі. Көлік құралының әкімшісіне жазатайым оқиға болған жағдайда әрекет ету үшін жылдам және уақтылы ескертіледі.

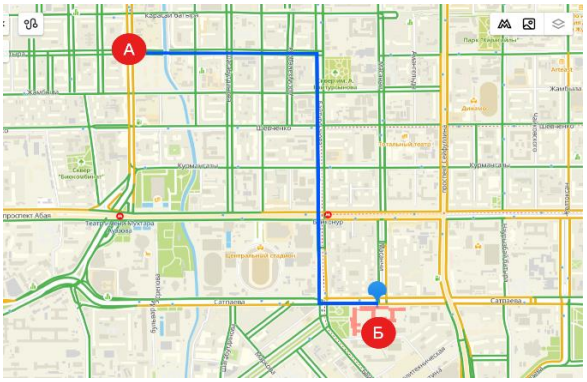
2.1 Такси көлік құралдарын тиімді маршрутизациялау алгоритмдері

Көлік құралдарын тиімді маршрутизациялау кезіндегі жол карталарындағы қиылыстарымен көрсетілетін жолдар алгоритмі, әдеттегі қысқа жолды іздеудің алгоритмінен өзгеше болып келеді.

Егер графикте жол торабының ұқсастығы ретінде қарастыратын болсақ, онда жол қиылыстары түйіндер, ал жолдар жиектер болып табылады, ал егер қашықтықты екі түйін арасындағы ең қысқа жолды табу әдісі ретінде қарастырылса, бұл мақсатта Дейкстра алгоритмі ең жақсы алгоритм екендігі дәлелденді. Бірақ нақты уақыт режимінде жол маршруты қарастырылған кезде, жол кептелісі, жол жүру уақыты және тағы да басқа факторларды ескеру қажет. Ол кезде біз нақты уақыт режимінде жол желісіндегі ең қысқа жолды табуға міндетті басқа факторларды ескеретін модификацияланған Дейкстра алгоритмін тұжырымдай аламыз.(2, 3 – сурет)



2 – сурет. Дейкстер алгоритмі үлгісі



3 – сурет. Дейкстер алгоритмін маршрутизацияда қолдану үлгісі

Тиімді жолды ұсыну үшін қол жетімді уақыт жол желісінің шектеулеріне байланысты шектелуі керек. Балама жолды ұсынуға аз уақыт кетуі керек.

2.2 Ең қысқа жол табудағы модификацияланған Дейкстер алгоритмі

Бұл алгоритм нақты уақыттағы жол желісіндегі ең қысқа жолды табу алгоритмінен гөрі жақсы нәтиже көрсетеді. Бұл алгоритм уақыт, қашықтық және жүктеме сияқты бірнеше параметрлерді жоспарланған жерге жеткізудің ең қысқа маршрутын табуға мүмкіндік береді. Алгоритмнің псевдокоды келесідей:

```
// Негізгі айнымалыларды бастапқы түйін ретінде ЖүрілгенТүйін және ҚысқаЖол[u] деп белгілеп аламыз
ЖүрілгенТүйін := {x}
ҚысқаЖол[x] := 0
// Пайдаланушыға қашықтық, уақыт және кептеліс факторлары арасында кез-келген мәнді таңдауға мүмкіндік беріледі
ACCEPT CHOICE
// Жоспарланған жерге ең қысқа ара қашықтықты табу
    МинимумҚашықтық := INFINITE
    FOR EACH b-ға дейінгі жүрілген түйіндер
        IF ҚысқаҚашықтық[b] < МинимумҚашықтық
            ҚысқаҚашықтық[b] = МинимумҚашықтық
        END IF
    END FOR
// Барлық түйіндер арасындағы қашықтықты олардың сәйкес коэффициенттеріне байланысты жаңартылады
FOR EACH түйіндер жұбы [x, a]
    Түйіндер арасындағы уақыт:      // Уақыт факторына байланысты жаңарту
        Қашықтық [x, a] := Қашықтық [x, a] * уақыт
    Түйіндер арасындағы кептеліс: // Кептеліске байланысты жаңартулар
        Қашықтық [x, a] := Қашықтық [x, a] * кептеліс уақыты
END FOR
```

```
FOR EACH түйіндер жұбы [a, b]
        Түйіндер арасындағы уақыт:        // Уақыт факторына байланысты жаңарту
                Қашықтық [a, b] := Қашықтық [a,b] * уақыт
        Түйіндер арасындағы кептеліс: // Кептеліске байланысты жаңартулар
                Қашықтық [a, b] := Қашықтық [a,b] * кептеліс уақыты
END FOR
//Жоспарланған жерге кептеліс факторларын ескеріп жалпы қысқа қашықтықты анықтау
        FOR EACH a, b түйіндер жұбы
                ҚысқаҚашықтық[x, a] + ҚысқаҚашықтық [a, b]
        END FOR
END WHILE
```

2.3 Эвристикалық алгоритм

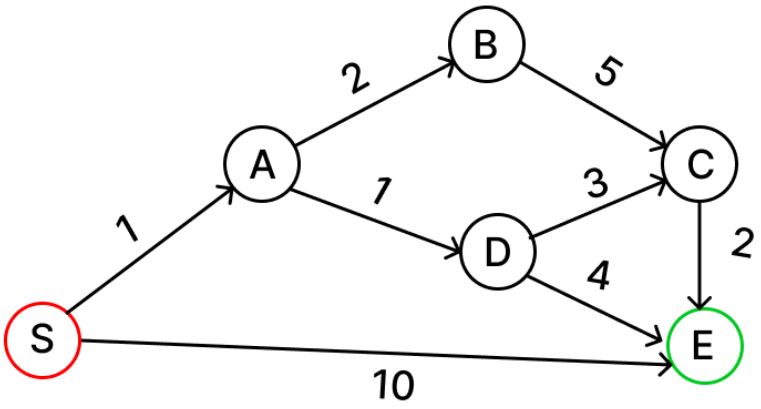
Барлық эвристикалық алгоритмдер туралы айтатын болсақ, A* алгоритмі оларды нақты уақыт режимінде жол желісінде жүзеге асыруға келгенде басқаларға қарағанда жақсы нәтиже бере алады, өйткені A* алгоритмі ең жақсы оңтайлы маршрутты табу үшін эвристиканы қолданады. Түйінді таңдау бастапқы түйіннен бастап жоспарланған түйінге дейінгі жердің құнын есептеу негізінде жүзеге асырылады. A* алгоритімінің жұмыс істеу принципін қарастыратын болсақ, процестін басында бастапқы түйіндерге іргелес түйіндер қарастырылады, қандайда бір іргелес түйінге қатысты минимальды мән(біздің жағдайымызда қашықтық) f(x) тандалынады, содан кейін табылған минимальді арақашықтықағы түйінге жол жүріледі. Алгоритм минимальді құнды тапқанша, іргелес жатқан түйіндерге дейінгі мәндерді басымдылықпен қарастырады. Жолдың басымдылығы **F(x) = g(x) + h(x)** мәні бойынша анықталады. Алгоритм жоспарланған түйінге жету барысында, түйіндер арасындағы ең минимальді құнды тапқанша, басқа түйіндер арасындағы құндармен салыстырылады. Көптеген шешімдердің ішінен ең аз құны бар шешім таңдалады.(4 – сурет)
H(x) эвристикасы құны неғұрлым аз болса, соғұрлым сол түйінге жүруге басымдық беріледі, әрбір минимальді құны бар жолды жүру үшін сұрыптайтын ағаштарды пайдалануға болады.

Көптеген қарастыралтын түйіндер jaqyn айнымалысында сақталынады, ал жоспарланған түйінге жету барысындағы басқада түйіндер — kezek айнымалысында кезекте тұрады. Жолды қысқа қашықтықпен жүріп өту үшін f(x) функциясын пайдаланамыз.

A* алгоритмін сипаттайтын псевдокод төменде көрсетілген:

```
function A*(first_node, last_node, f)
        // жүріп өтілген түйіндер жиыны
        var jaqyn := the empty set
        // қысқа жол жүрудегі кезекте тұрған түйіндер
        var kezek := make_queue(f)
```

```
        enqueue(kezek, path(start))
        while kezek is not empty
                var p := remove_first(kezek)
                var x := the last node of p
                if x in jaqyn
                        continue
                if x = goal
                        return p
                add(jaqyn x)
                foreach y in success(x)
                        enqueue(kezek, add_to_path(p, y))
return fail
```



4 – сурет. Модификацияланған Дейкстер алгоритмі үлгісі

Түйін	h(x)
S	0
A	0
B	0
C	0
D	0
E	0

F(x) = g(x) + h(x) өрнегі бойынша қарастыратын болсақ, **F(x)** бастапқы түйіннен жоспарланған түйінге дейінгі жүрілетін қысқа қашықтық. **h(x)** әрбір түйіннің өзіндік құны, біздің жағдайда барлығын 0-ге теңестіріп аламыз, ал **g(x)** түйіндер арасындағы ара қашықтық.

S➞E бастапқы пунктен соңғы пунктке қысқа жолмен бару алгортимі

S➞A➞B➞C➞E: F(x) = 1 + 2 + 5 + 2 = 10 (кезек)

S➞E: F(x) = 10 (кезек)

S➞A➞D➞C➞E: F(x) = 1 + 1 + 3 + 2 = 7 (кезек)

S➞A➞D➞E: F(x) = 1 + 1 + 4 = 6

g(x) + h(x) ≤ g(y) + h(y)

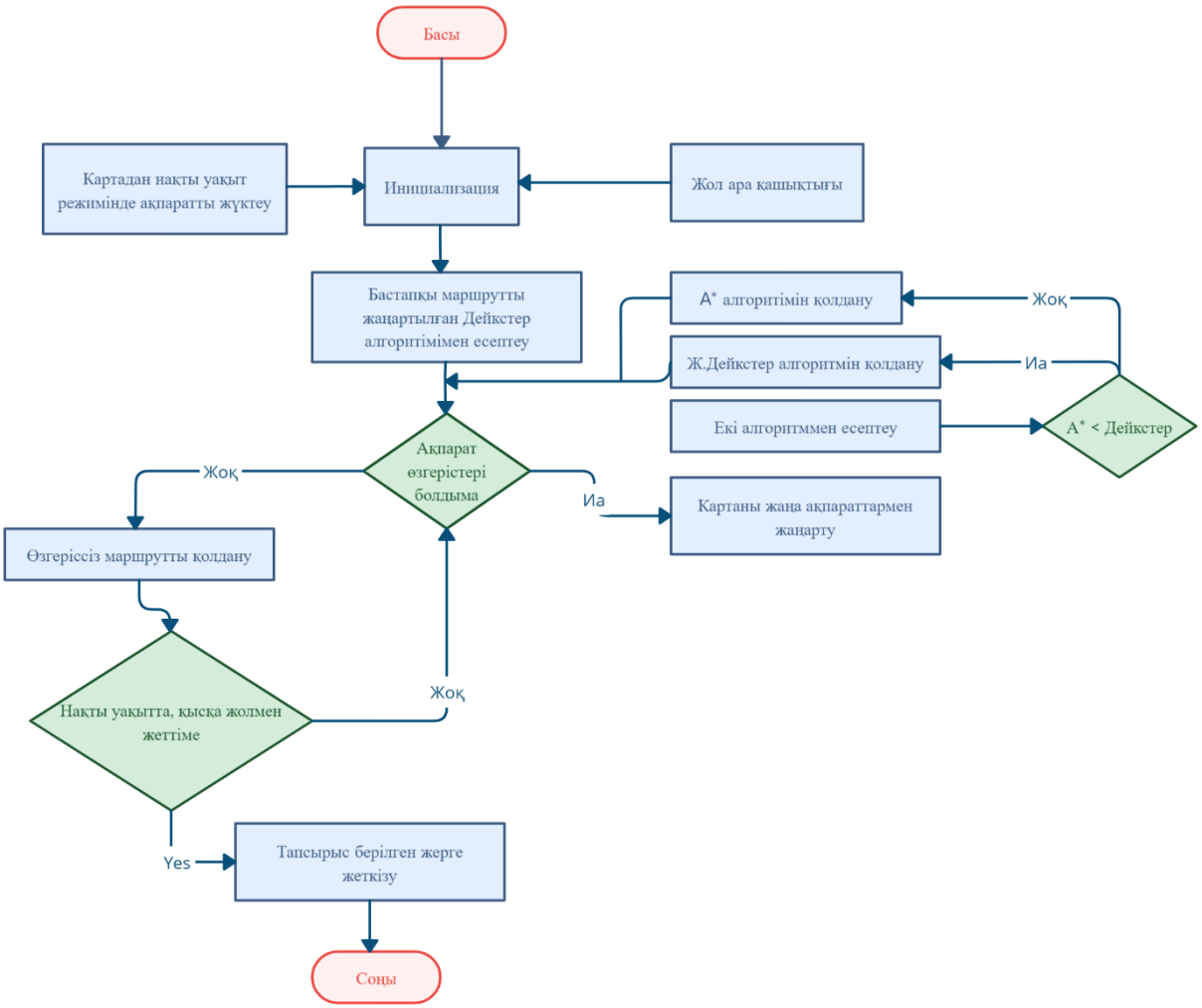
осы өрнек арқылы кезектегі жүріп өтілген жол мен жүріп өтілетін жолдардың құны салыстырылады

2.4 Жүйеге кіріктірілетін алгоритм және оның нәтижелері

Алгоритм А * түйіндер арқылы өтетін жолды "оптимистік" бағалаумен жұмыс істейтіндіктен, түйіндер арасындағы жолдың ең қысқа қашықтығын жүріп өтеді.

А * қысқа жолды іздеуді аяқтаған кезде, алгоритмге сәйкес, ол кез-келген түйін арқылы барлық бағыттағы жол жүріп, жүрілген түйіндер арасындағы ара қашықтықтарды бір бірімен салыстырады. Қысқаша айтқанда, А * жолдың ұзындығын азайту мүмкіндігін ешқашан жіберіп алмайды, сондықтан ол қолайлы алгоритмдердің бірі болып табылады.

Жаңартылған Дейкстер алгоритмі мен А* алгоритмі бір біріне қатты ұқсайды. Көп жағдайда екі алгоритмнің нәтижелері бірдей шығады. Соған байланысты көлік құралдарын басқаруда такси жүргізушілерінің уақытын үнемдеу үшін және жанармай шығының үнемдеп, көптеген сапарларды орындау үшін жүйеде екі алгоритм колданылады.(А* және модификацияланған Дейкстер алгоритмі). Жүйеде екі алгоритмнің нәтижелері салыстырылынып, ең қысқа жолды және ең қысқа уақытта жетуді көрсеткен алгоритмді сапар барысында жүргізушіге ұсыналады. Жүйеде қолданылатын алгоритмдердің диаграммасы төменде көрсетілген. (5 – сурет)



5 – сурет. Жүйеге кіріктірілетін алгоритмдер

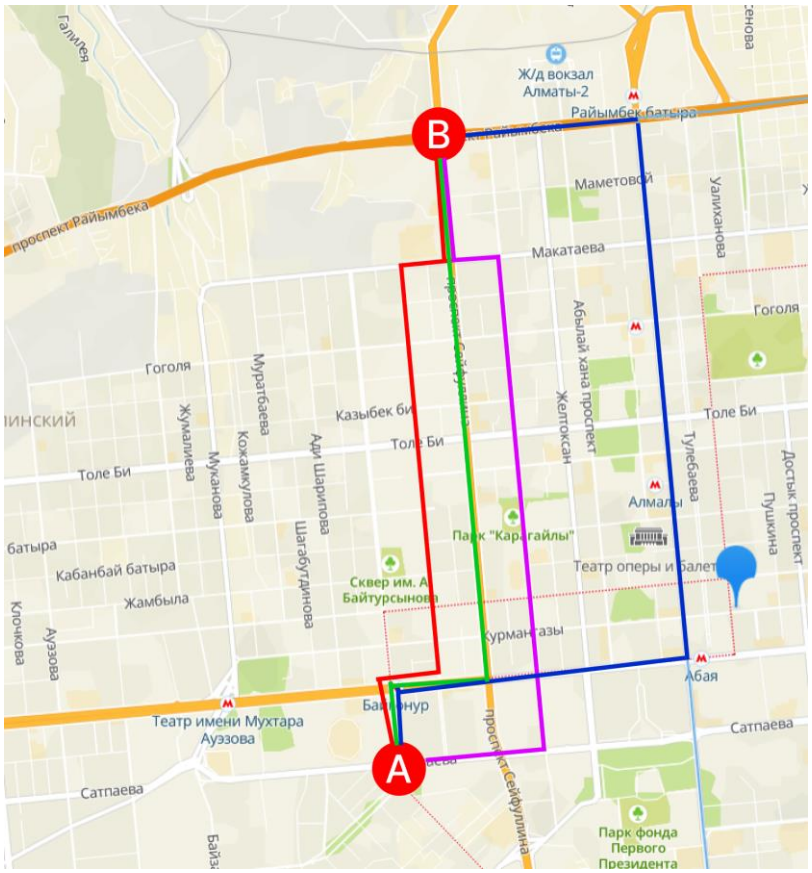
Бұл гибридті модель модификацияланған Дейкстер және эвристикалық алгоритмдерді (A* іздеу әдісі) толық қолданады және нақты уақыт режимінде жол маршруттарын кептеліс уақыт шектеулерін шешетін тиімді стратегия.

- Бастапқыда ең қысқа жол модификацияланған Дейкстер алгоритмін қолдана отырып жасалады. Жаңа маршрут, жүктемені өзгерту немесе жол учаскесін бұғаттау сияқты кез-келген жаңа жаңартуларды алған кезде балама маршрутты A* іздеу әдісін қолдана отырып есептеледі және екі тәсіл үшін де жол уақыты салыстырылады. (6 – сурет)
- Қысқа жол маршруты, қысқа уақыт тандалынады және маршрут сәйкесінше жаңартылады.
- Бұл процесс тапсырыс берілген жерге кептеліс кезінде қысқа уакыт ішінде қысқа жолмен жетуді тапқанша қайталанады.

Нәтижелерді талдау үшін Android қосымшасы жасалды. Әр түрлі алгоритмдерді қолдана отырып, атап айтқанда: модификацияланған Дейкстра алгоритмі, A* алгоритмі және сол қосымшаның құрамына кіретін алгоритм, қашықтық пен жол уақыты сияқты ең қысқа маршрут статистикасы төменде көрсетілгендей осы алгоритмдердің әрқайсысын қолдана отырып жазылды:

1 – кесте. Маршрутизация сынақтамасы 5км-ге дейін
(Райымбек-Сейфуллин ➞ Сатпаев-Байтұрсынов)

Алгоритм	Ара қашықтық *(км)*	Уақыт *(м)*
Қарапайым Дейкстер	6.4	17
Модификацияланған Дейкстер	5.5	12
A*	5.2	10
Жүйеміздегі алгоритм	5.2	10



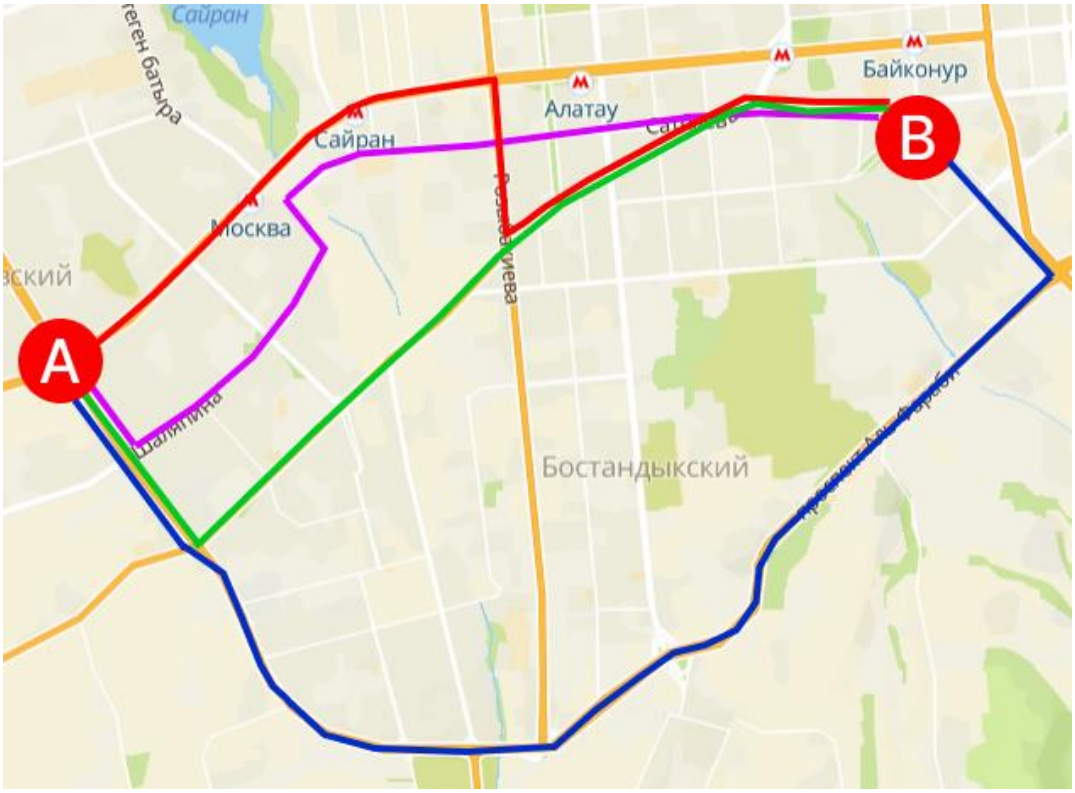
6 – сурет. Кесте 1-дегі алгоритмдерді қолдану арқылы жүрілген маршрут графигі
(**Жүйедегі алгоритм:** жасыл; **A*:** қызыл; **М.Дейкстер:** күлгін; **Қ.Дейкстер:** көк)

Жоғарыда көрсетілген алгоритмдер 5км-ге дейінгі қашықтықта сынақтан өткізілді. Алгоритмдер арасында жоспарланған жерге жету барысында қарапайым Дейкстер ең қысқа жол табуда ауытқушылық көрсетті және ұзақтау уақыт арасында жетті (Кесте 1). Ал модификацияланған Дейкстер, A* және жүйемізде қолданылатын алгоритм салыстырмалы түрде болжамды нәтиже көрсетті, нақтырақ айтқанда A* және жүйедегі алгоритм өте жақсы нәтижеге қол жеткізді. Бұл екі алгоритм өз кезегінде 5км-ге дейінгі қашықтықта минимальді

уақыт ішінде қысқа жолды жүріп келді. Келесі сынақтама ұзақтау (10-12км) ара қашықтыққа сыналатын болады. (7 – сурет)

2 – кесте. Маршрутизация сынақтамасы 10-12км-ге дейін
(Абай-Саин ➞ Сатпаев-Байтұрсынов)

Алгоритм	Ара қашықтық *(км)*	Уақыт *(м)*
Қарапайым Дейкстер	11	24
Модификацияланған Дейкстер	10,7	26
A*	10,8	25
Жүйеміздегі алгоритм	10,2	22



7 – сурет. Кесте 2-дегі алгоритмдерді қолдану арқылы жүрілген маршрут графигі
(**Жүйедегі алгоритм:** жасыл; **A*:** қызыл; **М.Дейкстер:** күлгін; **Қ.Дейкстер:** көк)

Жоғарыда көрсетілген графикте қарапайым Дейкстер алгоритмі ғана жолдағы көптеген факторларды негізге алмайды, сондықтан нәтижесі басқа алгоритмдерге қарағанда тиімсіз көрсеткішті көрсетеді, соған орай көлік құралдарын бақылау жүйесінде бұл алгоритмді қолданбайтынымыз анық. Ал қалған алгоритмдерді бір біріне ұқсас көрсеткіштер көрсеткендіктен, алдағы уақытта басқада сынақтарға нәтижелерін салыстыру үшін қолданамыз. Жүйемізде қолданалатын гибридті алгоритм (М.Дейкстер + A*) қысқа және ұзақ

қашықтықта қысқа уақыт ішінде межелеген жерге жетті және бұл оптимальді жақсы нәтиже болып табылады.

Жүйеде қолданылатын алгоритмдерді қорытындалай келе, зерттеу барысында ұсынылған алгоритмдер Android құрылғысында жасалынған такси қызметіндегі көлік құралдарын бақылау жүйесіне оң әсерін тигізді. Жүргізушілер мен жолаушылардың қауіісіздігінен бастап , қысқа уақыт ішінде межелеген жерге жету және сапар барысынды жанармайды үнемдеу секілді процесстер автоматты түрде орындалатын болғандықтан такси саласындағы кәсіпке қомақты пайда табуға ықпал етеді. Бұл жүйеде таксидің орналасқан локациясын және белгілі бір тапсырыста жүргендігін нақты уақыт режимінде бақылауға болады. Сонымен қатар, жүйе бақылаушысы жүйедегі процесстер туралы статистикалық мәліметер арқылы есеп алып, алдағы уақытта такси қосымшасын дамыту үшін өзгерістер енгізе алады, такси көліктерінің нақты уақыт режимінде жағдайын бақылап, жолдағы орын алған жол апатын жедел түрде шешуге көмектеседі.

2.5 Көлік құралдарын бақылау кезіндегі жанармайды тиімді үнемдеу моделі

Жұмыс істеп тұрған көліктердің жанармай шығыны мен шығарындыларын бақылау – жанармай шығының азайту үшін қажет. Такси – жолаушылар көлігі жүйесінде энергия тұтынуын бақылауды қажет ететін негізгі нысандардың бірі. Дегенмен, автокөліктің жанармай тұтынуы мен шығарындылары туралы деректерді жинаудың дәстүрлі әдістерінің құны жоғары және техникалық қызмету көрсетуде ыңғайсыздық тудырады.

Әлемдік позициялар жүйесінің (GPS) траектория деректеріне негізінде такси жанармайы шығыны мен шығарындыларын нақты түрде бақылап, қадағалауға болады.

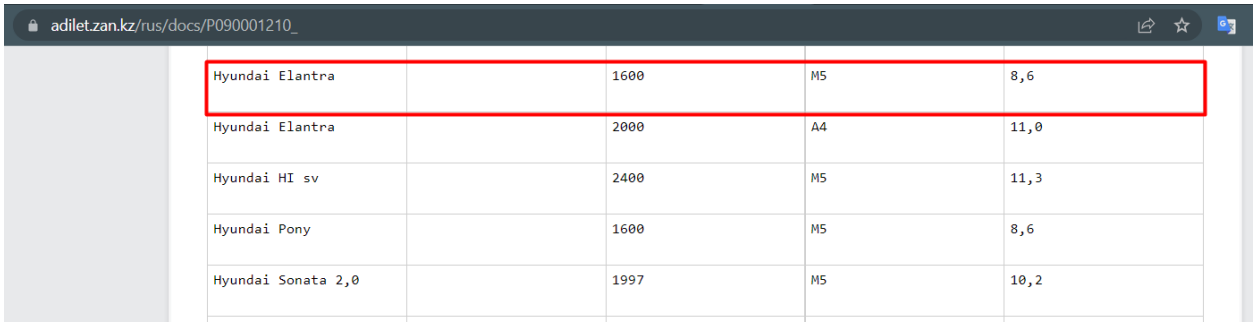
Экспериментті алдымен үш түрлі жүргізу циклімен жүргізіледі: круиз, жеделдету және баяулау және осы екеуін қоса алғанда, құрамдас жүргізу циклі. Содан кейін қозғалыс траекториясын қайта құру негізінде отын шығыны мен шығарындыларды есептеу модельдері ұсынылады. Сондықтан таксилердің отын шығыны мен шығарындыларын осы үш жүргізу цикліне сәйкес келетін GPS траектория деректері арқылы алуға болады. Модельдің дәлдігі отын шығыны (92%) және CO2 эмиссиясы (95%) CO, NOx және HC шығарындылары үлгілерінен (60%–70%) өлшемдерге сәйкес келетіні тексерілді.

Сонымен қатар, тәуелді айнымалы ретінде 100 км-ге отын шығынын ескере отырып, модельдің нәтижелері мен өлшемдер арасындағы салыстырмалы қателер қалалық жерлерде 1,9% және жан-жақты жұмыс жағдайында (яғни, қалалық және қала маңындағы аудандарда) 11,2% құрады.

Бақытымызға орай, жаһандық позиция жүйесі (GPS) көлік құралдарын динамикалық бақылау, қауіпсіздікті бақылау және нақты уақыт кестесін құрудың басқару талаптары ретінде таксилерде және басқа да жұмыс істейтін көліктерде кеңінен орнатылды. Осылайша, автокөліктің отын тұтынуы мен

шығарындыларын бақылаудың негізгі мәселесін нақты уақыт режиміндегі GPS траектория деректеріне негізделген автокөліктің отын тұтынуы мен шығарындыларының үлгілерін құру арқылы шешуге болады.

Бұл экспериментте сынақ көлігі ретінде Hyundai Elantra таксиі таңдалды, себебі ол Алматы қаласының такси жүргізушілерінің арасында кең тараған. Сынақтан өтетін көлік түрі 1.6 L қозғалтқышы бар. Көліктің жасы - 5 жыл. Қазақстан Республикасының жанар-жағар майларды тұтыну нормаларын бақылайтын орган басшылыққа алған стандартты отын шығыны бойынша қалалық жағдайларда отын шығыны 8.6 л/100 км, ал кешенді жағдайларда 6.9 л/100 км құрайды. (8 – сурет)



Hyundai Elantra		1600	M5	8,6
Hyundai Elantra		2000	A4	11,0
Hyundai HI sv		2400	M5	11,3
Hyundai Pony		1600	M5	8,6
Hyundai Sonata 2,0		1997	M5	10,2

8 – сурет. Қазақстан Республикасының жанар-жағар майларды тұтыну нормалары (https://adilet.zan.kz/rus/docs/P090001210)

Алматы қаласында жылдамдық шектеулері қалалық жерлердегі әртүрлі жолдарда 30-дан 120 км/сағ-қа дейін ауытқиды. Әдетте, қала жағдайында такси жүргізу жылдамдығы 95 км/сағ-тан аз. Сондықтан сынақ жылдамдығы 0–95 км/сағ шектелді.

Зерттеуімізде көлік құралының жанармай шығыны және үлгі шығарындылары осы болжамға негізделген: әрбір қозғалыс траекториясын төрт жұмыс жағдайына бөлуге болады, атап айтқанда, бос жүріс, крейсерлік, жеделдету және баяулау. Содан кейін, GPS деректеріне және траекторияны қайта құруды жүзеге асыруға сәйкес, көліктің отын шығыны мен шығарындыларын көлік жылдамдығы мен отын шығыны және осы төрт қозғалыс циклiндегі шығарындылар арасындағы сандық қатынас арқылы бағалауға болады. Әдетте, бос жүріс күйіндегі көліктер бүкіл саяхат кезінде шағын үлесті құрайды, сонымен қатар жанармай шығыны мен шығарындылар деңгейі төмен; бұл ретте GPS траектория деректерін талдау арқылы бос жүрістің басталу және аяқталу уақытын өлшеу қиын. Осылайша, бос жүріс жағдайы осы зерттеуде төмен жылдамдықтағы бір круиздік бөлік ретінде қарастырылатын болады.

Біздің қазіргі зерттеуімізде круиздік жылдамдық 0-ден 95 км/сағ аралығында болды. Алдымен біз жылдамдықты 10 бірлік диапазонға бөлдік, олар 0–5, 5–15, 15–25, 25–35, 35–45, 45–55, 55–65, 65–75, 75–85 және тиісінше 85–95 км/сағ. Қысқаша суреттеу үшін, бұл жылдамдық диапазондары R әрпімен және 0–5 км/сағ қоспағанда медианалық жылдамдықпен жеңілдетілді. Мысалы, 15–25 км/сағ үшін ол R_{20} ретінде жеңілдетілді. Атап айтқанда, 0–5 км/сағ үшін ол R_5 -ке ауыстырылды. Осыған сүйене отырып, әртүрлі жылдамдық диапазонындағы круиздік қозғалыс циклiндегі жанармай шығыны мен

шығарындылары орташа жылдамдықтағы сәйкес мәнмен ұсынылуы мүмкін деп болжалды. 0–5 км/сағ диапазондағы отын шығыны мен шығарындылары R_5 мәніне ауыстырылды. Круиздік қозғалыс цикліндегі отын шығынының және шығарындыларының стандартты мәндері жинақталған

Біздің қазіргі зерттеуімізде круиздік жылдамдық 0-ден 95 км/сағ аралығында болды. Алдымен біз жылдамдықты 10 бірлік диапазонға бөлдік, олар 0–5, 5–15, 15–25, 25–35, 35–45, 45–55, 55–65, 65–75, 75–85 және тиісінше 85–95 км/сағ. Қысқаша суреттеу үшін, бұл жылдамдық диапазондары R әрпімен және 0–5 км/сағ қоспағанда медианалық жылдамдықпен жеңілдетілді. Мысалы, 15–25 км/сағ үшін ол R_{20} ретінде жеңілдетілді. Атап айтқанда, 0–5 км/сағ үшін ол R_5 -ке ауыстырылды. Осыған сүйене отырып, әртүрлі жылдамдық диапазонындағы круиздік қозғалыс цикліндегі жанармай шығыны мен шығарындылары орташа жылдамдықтағы сәйкес мәнмен ұсынылуы мүмкін деп болжалды. 0–5 км/сағ диапазондағы отын шығыны мен шығарындылары R_5 мәніне ауыстырылды. Круиздік қозғалыс цикліндегі отын шығынының және шығарындыларының стандартты мәндері жинақталған

Жоғарыда қорытындылай келе, көліктің отын шығыны мен шығарындыларын әртүрлі жылдамдық диапазонындағы круиздік, жеделдету және баяулаудың есептеу мәндерін қосу арқылы бағалауға болады. Жүргізу траекториясын қайта құруға негізделген жалпы отын шығыны мен шығарындылардың үлгілері төмендегідей тізімделген

$$T_a = \sum_k S_k \times t_k + \sum_i \sum_j C_{ij} \times P_{ij} \qquad (1)$$

- T_a – жалпы жанармай шығыны мен шығарындылары (L немесе g);
- S_k – әртүрлі жылдамдық диапазондары арасындағы жанармай шығыны және шығарындылары (L/s – литр/секунд немесе g/s – газ/секунд).
- k мәндері 5, 10, 20, 30, 40, 50, 60, 70, 80 және 90;
- t_k – GPS траекториясының деректерінен алынған k жылдамдық диапазонындағы уақыт ұзақтығы.
- C_{ij} – 2-кестеде көрсетілгендей, R_i диапазонындағы бастапқы жылдамдықпен және R_j ауқымындағы соңғы жылдамдықпен орташа жанармай шығыны мен шығарындылары (L немесе g).
- P_{ij} – GPS траектория деректерінен алынған жылдамдықтың R_i диапазонынан R_j диапазонына дейін өзгеретін уақыты.
- i, j мәндері 5, 10, 20, 30, 40, 50, 60, 70, 80 және 90.

T_a – жалпы жанармай шығыны мен шығарындылары екені анық. (1) теңдеу көліктің үдеуі мен баяулауы бір уақытта аяқталды деген болжаммен ұсынылғандықтан, есептеу нәтижелеріне сәйкес жанармай шығыны көптеу шығады. Осылайша, көліктің жылдамдатуы мен баяулату процесінде жанармай шығыны мен шығарындыларды жою үшін (1) формулаға кейбір өзгертулер жасалынуы қажет болады.

$$T = T_a - T_b = \sum_k S_k \times t_k + \sum_i \sum_j C_{ij} \times P_{ij}$$
$$- \sum_i \sum_j (S_i + S_j)/2 \times |j - i| \times 0.46 \times P_{ij} \quad (2)$$
$$= \sum_k S_k \times t_k + \sum_i \sum_j (C_{ij} - (S_i + S_j)/2 \times |j - i| \times 0.46) \times P_{ij}$$

3 – кесте. Әр түрлі үдеу және баяулау процестеріндегі жанармай шығыны.

Жол жүру типі	Жылдамдық(км/сағ)	Жанармай (л)
Жылдамдатып жүру	5-10	0.0032
	10-20	0.0039
	20-30	0.0065
	30-40	0.0067
	40-50	0.0096
	50-60	0.0136
	60-70	0.0159
	70-80	0.0204
	80-90	0.0247
Баяулатып жүру	90-80	0.0033
	80-70	0.0025
	70-60	0.0022
	60-50	0.0017
	50-40	0.0013
	40-30	0.0009
	30-20	0.0007
	20-10	0.0006
	0-5	0.0002

(2) теңдеуде T_a – (1) теңдеуден алынған мән. T_b – жеделдету және баяулау үшін бірнеше рет есептелетін круиздік процестегі жанармай шығыны және шығарындылар. Біз үдеу мен баяулау жылдамдығы сәйкесінше 0,6 және -0,6 $м/с^2$ болды деп есептейміз. Сонда 0,46 мәні үдеу және баяулау кезінде 1 км/сағ жылдамдықты өзгертуге кететін уақыт. Әлбетте, (2) теңдеу (1) теңдеуіне қарағанда тиімдірек және жанармайдын біраз бөлігін үнемдейді.

Модельдеу эксперименті үшін модельдің дәлдігі сынақтардағы құрамдас жүргізу циклдері мен Қазақстан Республикасының жанар-жағар майларды тұтыну статистикаларынан алынған есептеу мәндері арасындағы нәтижелерді салыстырып, жанармайды үнемдеудің тиімді моделі формула арқылы ұсынылды. Такси GPS деректерінен алынған белгілі бір көлік траекторияларына сүйене отырып, әртүрлі жұмыс жағдайларында 100 км-ге жанармай шығыны

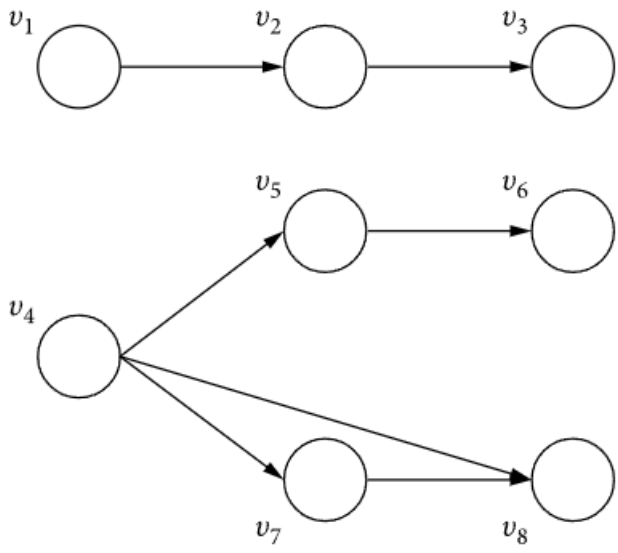
есептелді, содан кейін модельдің дәлдігі ресми статистикада көрсетілген 100 км-ге жанармай шығынымен салыстыру арқылы тексерілді.

2.6 Жолаушалар арасындағы такси сапарының бөлісуі және оның жуықтау алгоритмі

Осы бөлімде біз n жолаушылардың онлайн режиміндегі деректері үшін барлық мүмкін такси сапарларын тізімдей алатынымызды атап өтеміз. Оларды тізімдеу көп уақытты қажет етеді. Осылайша, біз алдымен мүмкін болатын барлық такси сапарларын табудың ең жақсы әдісін ұсынамыз.

Белгілі бір m жолаушылар санын ескере отырып, олар туралы барлық ақпараттарды білуге болады, соның ішінде отырғызылу орындары, отырылғызу уақыты, баратын орындар және қысқа жолдар жүрісі. Қандай да бір екі жолаушы бір таксимен жүре ала ма, жоқ па, соны тексеру қиын емес, яғни отырғызу шарттары мен бару шарттарын тексеру арқылы. Осылайша, біз барлық жолаушылар үшін осы байланысты ұсынып график жасай аламыз, бұл бізге барлық мүмкін такси сапарларын табуға көмектеседі.

G(P) = {V,E} жолаушылар графигі үшін, мұндағы *V* – әр жолаушы ұсынатын түйіндер жиынтығы, ал *E* — бағытталған жиектер жиынтығы. Жолаушыларды отырғызу тәртібін ескеру қажет болғандықтан, бұл қабырғалардың бағыты бар. Егер p_j жолаушысын p_i жолаушысының бірлескен сапарына қанағаттандыру мүмкін болса, онда v_j-ден v_i-ге дейінгі жиек орнатылады . Осылайша, біздің жолаушылар санын анықтай аламыз. Әр түйіннің екі атрибуты бар: тиеу уақыты және келу уақыты. Бұл келу уақытты такси жылдамдығымен және ең қысқа жолмен есептеледі. Осылайша, біз жолаушының қашан отырғызылғанын білеміз. (9 – сурет)



9 – сурет. Ең қысқа жол табудағы жуықтау алгоритмі

Содан кейін біз *G(P)* графигіндегі барлық мүмкін маршруттарды табу үшін терең іздеуді және топологиялық сұрыптауды қолдана аламыз. 2-суретте көрсетілгендей, біз v_1, v_2 және v_3 түйіндерін қарастырамыз, бұл p_1 жолаушысымен жүру p_2 жолаушысын алып кетуі мүмкін, бірақ p_3 жолаушысын ала алмайтындығын көрсетеді. v_4-ден v_5-ге дейінгі және v_7-ге дейінгі қабырғалар бар, бірақ v_5 және v_7 жолаушылары бір таксимен бөлісе алмайтындығын білдіреді, олардың арасында қабырға жоқ. Осылайша, p_4 таксиімен жүру p_5 және p_7 жолаушыларының біреуін ғана ала алады . p_6 жолаушысының жағдайы p_3 жолаушысымен бірдей. Енді v_2 және p_4 жолаушылары сапарды p_7-мен таксиге бөле алатындығын көрсететін v_8 шыңына назар аударылады. Сондықтан p_8 уақытын тексерудің қажеті жоқ .

Бұл іздеу процесінде біз топологиялық сұрыптау техникасын қолдануымыз керек. Егер біз осы шыңды қосқымыз келсе, жиынтықтағы кейбір шыңдарды алып тастау үшін келу уақыты мен қабылдау уақытын тексеруіміз керек. Содан кейін, егер жиынтықтың барлық шыңдары осы шыңға бағытталған болса, онда бұл шыңды қосуға болады. Бұл ретте таксидегі жолаушылардың ең көп санын да ескеруіміз қажет. Мұны қолдана отырып, біз әртүрлі жолаушылар құрамы бар такси сапарларының барлық түрлерін таба аламыз, бұл процесті тездетеді. Әрине, бұл іздеу процесінде уақытша күрделілікті жақсарту үшін кейбір жад әдістерін қолдануға болады.

T таксиінің барлық мүмкін сапарларының жиынтығына сүйене отырып, біз *w(T)*-ге барлық жолаушыларды қанағаттандыру үшін такси сапарларының ең аз санын таңдауымыз керек. *T* таксимен бөлісудің әр сапары үшін біз оған қанағаттануы мүмкін жолаушылар санын белгілейміз. Содан кейін біз таксимен бөлісу сапарларын бір-бірлеп таңдау үшін жуықтау алгоритмін жасаймыз. Әр итерацияда біз күткен жолаушыларды қанағаттандыра алатын такси сапарын таңдаймыз. Немесе біз әр такси сапарының жиынтығы деп елестете аламыз. *G(P)* жолаушылар графигінің барлық шыңдарын жабу үшін жиынтықтардың ең аз санын таңдау керек. Әр итерацияда біз ашылмаған шыңдардың максималды санын жаба алатын жиынтықты таңдаймыз. Енді біз бұл алгоритм *O(log m)* жуықтау екенін көрсетеміз, мұндағы *m*-жолаушылар саны, бұл қол жеткізуге болатын ең жақсы жуықтау болып табылады (егер P=NP болса).

Дәлел алгоритмнің бір қадамын қарастырыңыз. Осы қадамда такси сапарын таңдамас бұрын c күтетін жолаушылар саны болсын. Алгоритммен таңдалмаған таксимен бөлісетін рейстердің ішінде оңтайлы шешім күтілетін жолаушыларды қанағаттандыру үшін осындай таксимен бөлісетін рейстердің кейбір жиынтығын қолданады. R_{OPT} алдыңғы таңдау негізінде бірлескен сапарлардың оңтайлы жиынтығы болсын.

OPT оңтайлы шешімде такси сапарларының саны болсын. $|R_{OPT}| \leq OPT$ екені түсінікті, өйткені бұл таксиді бірлесіп пайдалану сапарлары оңтайлы шешімнің жиынтығы болып табылады, мұнда $|R_{OPT}|$ - бұл R_{OPT} жиынтығының мөлшері. Сонымен қатар, біз $\sum_{T \in R_{OPT}} w(T) \geq c$ бар екенін байқаймыз, өйткені бұл сапарлар қалған күткен жолаушыларды қанағаттандыра алатын өрнек.

$$max_{T \in R_{OPT}} w(T) \geq \frac{\sum T \in R_{OPT} w(T)}{|R_{OPT}|} \geq \frac{c}{OPT} \times O(log\ m) \qquad (3)$$

мұнда бірінші теңсіздік орташа дәлелден, ал екінші теңсіздік біздің алдыңғы бақылауларымыздан туындайды. Сонымен, *c* күткен жолаушылар қалған бұл қадамда *c/OPT*-тен кем емес орташа санды қанағаттандыра алатын таксиді бірлесіп пайдалану керек.

Біз әрбір жолаушыға таксимен жол жүру құнын бөлуге тырысамыз. Таксиді бірлесіп қолданатын әр сапардың құны-бұл барлық нақты шығындар. Осылайша, әр жолаушы *1/w(T)* заряд алады. Енді менің қанағаттанатын жолаушым OPT/(m – j + 1) - дан артық төлем ала алады деп дау айта аламыз. Бұл (2) теңдеуден және j-ші күтілетін жолаушы қанағаттанған кезде, бұл қадамда күтілетін жолаушылардан кем дегенде (m – j + 1) болуы керек.

Осылайша біз барлық жолаушылардың жалпы ағының жинақтай аламыз. Жалпы ағын:

$$\sum_{j=1}^{m} \frac{OPT}{m - j + 1} = OPT \times \sum_{j=1}^{m} \frac{1}{j} = OPT \times O(log\ m) \qquad (4)$$

Бұл жақындау жеткіліксіз болып көрінеді, өйткені сапарлар саны өте көп. Алайда, тәуелсіз шыңдар үшін, яғни басқалармен бөлісе алмайтын кейбір жолаушылар үшін жуықтау коэффициенті ескерілмейді. Ең көп кездесетін жағдайларды табу оңай, өйткені комбинациялар онша көп емес. Сонымен, осы алгоритмнің нақты қосымшадағы өнімділігі талаптарды қанағаттандыру үшін жеткілікті.

2.7 Такси қызметіне арналған көлік құралдарын бақылау жүйесіне арналған SWOT анализ

Күшті жақтары	Әлсіз жақтары
1. Қоғамдық көлікті бақылаудың тиімді әлемдік тәжірибесі 2. Нарықтағы бәсекелестіктің төмен деңгейі 3. Нарықты құрылымдық жобалаудың басталуы: компанияларды операторлар мен агенттерге бөлу 4. Нарық әлеуетті инвесторлар мен жеке қаржыландыру көздері үшін ашық	1. Аймақтарда, әсіресе орталықтан шағын және шалғай елді мекендерде такси көлігінің даму деңгейі жеткіліксіз 2. Қызмет көрсету деңгейінің төмендігі 3. Қоғамдық көлік кәсіпорындарының негізгі құралдар тозуының жоғары деңгейі 4. Білікті кадрлардың жеткілікті санының болмауы

Мүмкіндіктер	Қауіптер
1. Саланың сұранысқа көптігі 2. Навигациялық жабдықтың қайталама нарығының пайда болуы 3. Отандық навигациялық жабдық өндірушілерінің және мониторинг технологияларын жасаушылардың пайда болуы 4. Саланы мемлекеттік қолдаудың тиімді тетіктерінің болуы 5. Көліктік диспетчерлеудің маңызды және үнемді нәтижелеріне қол жеткізу 6. Халықаралық қаржы институттарынан келетін қоғамдық көліктің әлеуетін күшейту бағдарламаларын ауқымды қаржылық қолдау және іске асыру	1. Жоғары деңгей іскери тәуекелдер 2. Көлік құралдарын бақылауға арналған жабдықтар мен бағдарламалық қамтамасыз ету технологияларының жоғары құны 3. Ассортимент саясатының әлсіз деңгейі (нарықта ұсынылатын технологиялардың аз саны) 4. Халықтың терминалдарды пайдалану мәдениетінің төмендігі 5. Нарықтың төмен инвестициялық тартымдылығы 6. Төмен кәсіпкерлік белсенділік 7. Көлік саласындағы мемлекеттік қаржыландыру негізінен ұлттық жобаларға арналған (әуежайлар, порттар, автомобиль жолдары, теміржолдар)

Бөлімге қорытынды

Қорытындылай келе бұл бөлімде көлік құралдарын бақылау жүйесінің әзірлеу модельдері мен алгоритмдері туралы кеңінен айтылды. Нарықтаға бар алгоритмдер мен ұсынылатын өзіндік алгоритмді салыстыру арқылы жақсы нәтижеге жету болатынына көз жетті. Бұл диссертациялық жұмыста ұсынылатын алгоритмдер такси қызметіндегі көлемді қаражат өсімін арттыруға және такси көлік құралдарының қауіпсіздіктерін сақтауға ықпал етеді. Қысқаша айтқанда, көлік құралдарын бақылау жүйесі нақты уақыт режимінде көліктерді оңтайлы түрде бақылауды қамтамасыз ете алады. Барлық факторларын ескере отырып, ұсынылған алгоритмді қолдану арқылы драйверге ең қысқа жол көрсетіледі. Қауіпсіз жылдамдықты асырған кезде жүргізушіге ескертіледі. Көлік құралының әкімшісіне жазатайым оқиға болған жағдайда әрекет ету үшін жылдам және уақтылы ескертіледі.

Бұл алгоритмдердің арасында Дейкстер алгоритмін атап өткен жөн. Бұл алгоритм нақты уақыттағы жол желісіндегі ең қысқа жолды табу алгоритмінен гөрі жақсы нәтиже көрсетеді. Бұл алгоритм уақыт, қашықтық және жүктеме сияқты бірнеше параметрлерді жоспарланған жерге жеткізудің ең қысқа маршрутын табуға мүмкіндік береді.

Ал екінші алгоритм ретінде эвристикалық A* алгоритмі қарастырылды. Бұл гибридті модель модификацияланған Дейкстер және эвристикалық

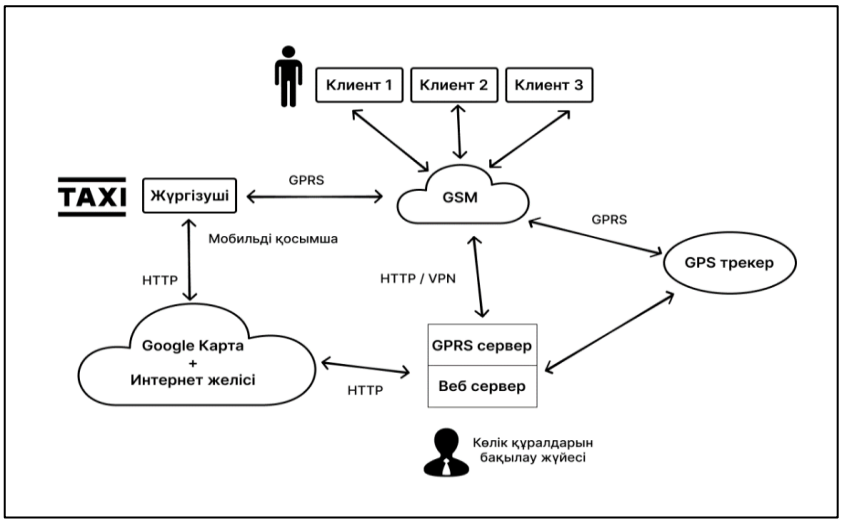
алгоритмдерді (A* іздеу әдісі) толық қолданады және нақты уақыт режимінде жол маршруттарын кептеліс уақыт шектеулерін шешетін тиімді стратегия.

Және де бұл бөлімде жағар жанармайдың үнемделуін қамтамасыз ететін тиімді формулалық модель ұсынылып зерттеу жұмысы жүргізілді. Формулаға бірқатар өзгертулер енгізу арқылы, қойылған мақсатқа жақсы нәтижемен жетті. Модельдеу эксперименті үшін модельдің дәлдігі сынақтардағы құрамдас жүргізу циклдері мен Қазақстан Республикасының жанар-жағар майларды тұтыну статистикаларынан алынған есептеу мәндері арасындағы нәтижелерді салыстырып, жанармайды үнемдеудің тиімді моделі формула арқылы ұсынылды.

3.0 Көлік құралдарын бақылау қосымшасының архитектурасы

Ұсынылған жүйе GPS модулінен, GSM модемінен, веб серверінен және GPRS серверінен тұрады. Жүйе құрылғыда GPS қосылған GSM құрылғысы арқылы көлікті үздіксіз бақылайды. Басқару жүйесі көліктен ақпаратты сервер жағында орнатылған SMS шлюз сервері арқылы алады. SMS клиенттерге кез келген кешігу немесе басқа маңызды хабарламалар туралы хабарлау үшін қолданылады. Жүргізуші көлік құралын бақылау әкімшісіне жолды тоскауыл қою кезінде немесе жолды жауып тастаған кез келген апат болған жағдайда балама бағытты сұрайтын SMS жібереді.

Жүйенің архитектурасы және жүйенің әртүрлі компоненттерінің бір-бірімен әрекеттесуі 10-суретте көрсетілген.



10 – сурет. Көлік құралдарын бақылау қосымшасының архитектурасы

Такси көліктерін бақылау үшін жүйелі түрдегі қосымша қажет болады. Яғни көлік жүргізушісінің арнайы интерфейсі және жасалынған іс әрекеттерді бақылайтын бақылау интерфейсі болады. Бақылаушы интерфейс түрлі платформада жасалынады(мобильді және вэб), бұл өз кезегінде такси көлік құралдарын әртүрлі құрылғыда кез келген уақытта тексеруге мүмкіндік береді.

Сапар барысында такси жүргізушілері онлайн режимде кәсіпорын бақылаушыларымен үздіксіз байланыста болады. Арадағы байланысты және іс әрекеттерді бақылауды қамтамасыз ету үшін RabbitMQ технологиясы қолданылады. RabbitMQ әртүрлі бағдарламаларға AMQP протоколы арқылы өзара әрекеттесуге мүмкіндік береді. RabbitMQ - SOA (қызмет көрсетуге бағытталған архитектура) құру және кейінге қалдырылған ресурстарды қажет ететін тапсырмаларды тарату үшін тамаша шешім болып табылады.

3.1 Байланыс технологияларына түсініктеме

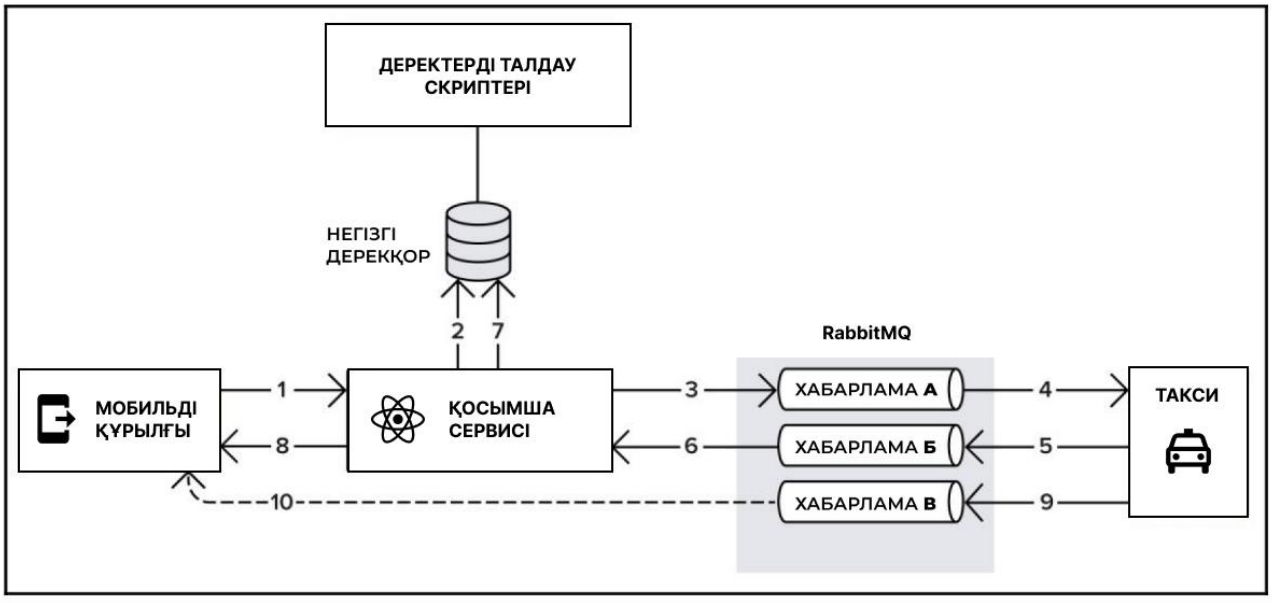
RabbitMQ – хабарламаларды кезекке қою жүйесі. Хабарлама кезегі, өз кезегінде, жүйенің әртүрлі құрамдас бөліктері арасында асинхронды түрде байланысу жолын қамтамасыз етеді. RabbitMQ AMQP протоколында жүзеге асырылады.

AMQP – жүйе хабарламалармен алмасу жолын анықтайтын ашық хаттама. Ол бір-бірімен өзара әрекеттесетін жүйелер ұстануға тиісті ережелерді белгілейді. Жүйелердің брокермен өзара әрекеттесу жолын сипаттаудан басқа, ол жүйелер арасында алмасатын командалар мен хабарламаларды стандарттайды.

AMQP тұжырымдамасы (концепция):

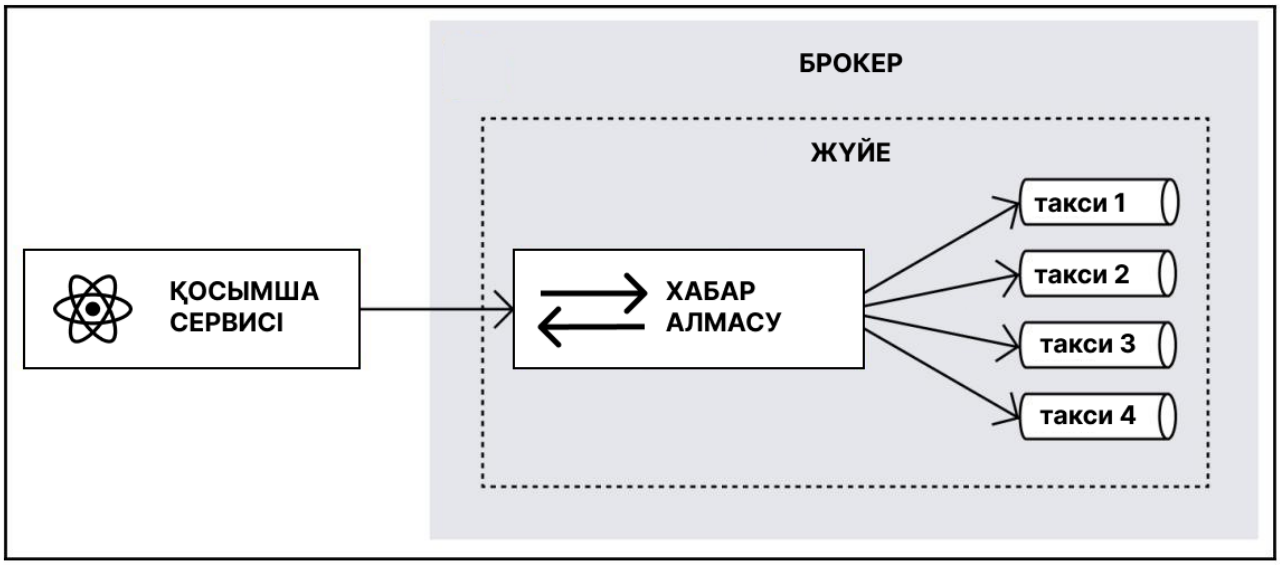
1. Хабарлама брокері – бір қолданбадан хабарламаларды қабылдайтын және оларды басқасына жеткізетін қолданбаның бөлігі.
2. Виртуалды хост – брокердің бөлігі болып табылады. RabbitMQ пайдаланатын қолданбаларды бөлу тәсілі.
3. Байланыс – қолданбалар мен брокер арасындағы физикалық TCP байланысы болып табылады. Клиент оффлайн болған кезде немесе жүйе қателікпен бұзылған кезде байланыс істен шығады.
4. Канал — байланыс ішіндегі виртуалды қосылым. Ол ресурстарды үнемдеу үшін бар каналды қайта пайдаланады және жүйелерді жаңа байланыс орнатуға мәжбүрлемейді. Жаңа хабар жарияланғанда немесе сұралғанда, бүкіл процесс бұрыннан бар канала орындалады.

Бұл ағынды 10 кезеңмен түсіндіруге болады, олар алдыңғы (11 – сурет) диаграммада көрсетілген



11 – сурет. Қосымшаның негізгі архитектурасы.
(Такси көлік құралдарын бақылауға арналған)

1. Кәсіпорын қызметкері мобильді құрылғы арқылы нақты режимде (real-time) хабарлама алмаса алады және кәсіпорынға тиесілі барлық такси көлік құралы туралы ақпарат ала алады. Хабарламалар React технологиясы арқылы жасалынған қосымша сервисі арқылы жүзеге асады
2. Барлық хабарламалар мен ақпаратттар деректер қорында сақталынады.
3. Нақты режимдегі ақпараттар секунд сайын өзгеретін болғандықтан, RabbitMQ де кезекте тұрады
4. Желіде жұмыс жасап тұрған такси көліктері туралы кәсіпорын байланыста бола алады
5. Такси көлігінде болып жатқан ақпараттар кері байланыспен кәсіпорынға жіберіледі
6. Ол кезегінде қосымша сервисінен өтеді
7. Қосымша сервисі таксиден алынған жаңа ақпараттарды деректер қорына тіркейді
8. Кәсіпорын нақты такси туралы ақпараттарды қабылдайды
9. Такси қолданбасы RabitMQ-ге белгілі бір уақыт аралығында таксидің географиялық орнын автоматты түрде жібере бастайды
10. Содан кейін таксидің орналасқан жері WebSocket арқылы бақылаушының мобильді қосымшасына тікелей беріледі, осылайша олар таксидің қанша жанармай жұмсағанын, геодеректерін, жолаушылар тасымалдау тарихын және тағы басқа маңызды ақпараттарды біле алады.



12 – сурет. Кәсіпорын бір мезетте көптеген таксилермен хабар алмасып, ақпараттарды көре алады

3.2 Такси көліктерін бақылау жүйесінің модульдері

Бұл жүйе төрт негізгі модульдерден тұрады:

- Локацияны анықтау модулі
- Роутинг модулі

- Жылдамдық туралы ескерту модулі
- Оқыс оқиғалар туралы ескерту модулі

Локацияны анықтау модулі

Google Map API интерфейстері құрылғыдан локацияны табу үшін қол жетімді. Көлік құралдарын бақылау жүйесі Google Api арқылы орналасқан жерді анықтауға мүмкіндік беретін бағдарламалық жасақтамамен біріктірілген. Құрылғы орналасқан жерді ендік және бойлық тұрғысынан анықтайды. Орналасқан жерді анықтау модулі пайдаланушының арнайы енгізуін қажет етпестен GPS қолданатын көлікті автоматты түрде орының бақылай алады. Алынған ендік пен бойлық GIS (ғаламдық ақпараттық жүйе) арқылы белгілі бір жердің координатасы арқылы анықтай алады. Осы орналасқан жерді анықтап алғаннан кейін такси жүргізушінің бастапқы мекенжайдан тапсырыс берілген мекенжайға бара жатқан траектироясын картадан көре аламыз.

Роутинг модулі

Роутинг модулі – бұл жобаның ең маңызды модулі. Ол драйверге орналасқан жерден бастап жолаушы жету керек жерге дейінгі жолды ұсынады. Маршруттаудың негізгі міндеті – жай ғана жолды анықтау ғана емес, такси тапсырысындағы баратын жерге дейінгі ең қысқа қашықтықты көрсету болып табылады. Маршрутты бағыттау үшін модификацияланған Дейкстра алгоритмі қолданылады. Мақсатты пунктке дейінгі бағытты жолдағы кедергілер немесе кептелістер, қашықтық, уақыт және тағы да басқа әртүрлі факторларға байланысты жасалынады. Егер жол қозғалысының қалыпты барысында қандай да бір оқыс жағдайлар туындаса, роутинг модулі маршруттарды секунд сайын жиі жаңартып отырады. Маршрут модификацияланған Дейкстра алгоритмі және A* алгоритмдері арасында жақсы нәтиже беретін алгоритмді қолдану арқылы динамикалық түрде жаңартылады.

Жылдамдық туралы ескерту модулі

Жылдамдық туралы ескертулер жолаушылардың да, жүргізушінің де қауіпсіздігін сақтауға көмектеседі. Бұл модуль жолдың түріне байланысты алдын-ала анықталған шекті мәндерге ие болады. Жүргізуші жылдамдықтан асып, осы шекті мәннен өткен кезде, оның ұялы телефонына дабыл хабарламасы түрінде жылдамдық туралы ескерту келеді. Бұл жүргізушіге рұқсат етілген жылдамдық шегінен асып кеткен көлік құралын тежеу керек екендігі туралы хабарлайды. Жылдамдық туралы ескерту модулі әртүрлі шектерден тұрады. Бұл шектер әртүрлі факторларға байланысты өзгереді. Қала сыртындағы магистральдар жоғары мәнге ие, өйткені жолдар кеңірек және кептеліссіз; ал жергілікті көшелерде кептелістер көп, сондықтан шекті жол төмен болуы керек. Бұл көлік құралдарының қауіпсіздігін арттырады.

Оқыс жағдайлар туралы ескерту модулі

Егер жүйе апаттың жоғары ықтималдығын анықтаса, көлік құралының бақылаушысына шұғыл түрде көлік құралының ағымдағы орналасқан жері туралы SMS хабарлама жіберіледі. Содан кейін көлік құралының бақылаушысы жүргізушімен байланысып, тиісті әрекеттерді орындау арқылы себебін анықтайды деп күтілуде. Апаттың пайда болуын ықтималдығын көлік

құралының бақылаушысы жүйеден көліктің жағдайы туралы стастикалық мәліметтерден анализ жасап, бақылау алады. Сонымен қатар, көлік апаты болған жағдайда жүргізуші оқиға туралы көлік құралдарының бақылаушысымен байланысуы керек. Әкімші жазатайым оқиға болған жердің орналасқан жерін анықтап алып, көлік апатына байланысты тиісті әрекеттерді жасай алады.

Көлік жүйелерінің кеңістіктік құрылымын көлік желілері анықтайды. Көлік желісі жолаушылар мен жүк тасымалы жүзеге асырылатын көлік байланыстарының тығыздығы деп аталады.

Көлік желілерінің маңызды ерекшелігі, олар жалпы жағдайда, өнеркәсіптік көліктен басқа, көлік жүйесіне кірмейді, бірақ олар үшін сыртқы орта болып табылады. Сонымен қатар, көлік желілері көбінесе көлік жүйелерінің сандық және сапалық сипаттамаларын анықтайды. Сонымен бірге, егер біз көлік құралдарының қозғалысына қатысты мәселелерді қарастыратын болсақ, онда көлік желілері зерттелетін көлік жүйесіне қосылады[5].

Әдетте қалалық көлік жүйелерін модельдеудің негізі ірі қаланың көлік қозғалысының жалпы көлеміндегі үлесі 85-95% - ды құрайтын жолаушылар көлік корреспонденцияларын іске асырлады. Көлік моделі тұтастай алғанда бағдарламалық жасақтама болып табылады ақпараттық және есептеу блоктарынан тұратын кешен. Ақпараттық блоктар көлік ағындарын болжауға қажетті ақпаратты сақтауға және өңдеуге арналған бірыңғай дерекқорды құрайды. Есептеу блоктары алгоритмдерді орындайды қозғалыс қажеттілігін болжауға және іске асырушыларды есептеуге бағытталған математикалық бағдарламалау мәселелерін шешу.

Көлік ұсынысы мынандай элементтерден тұрады,көлік жүйесі(қала немесе аудан) қанағаттандыратын қолданыстағы көлік сұраныстарынан.Көлік сұранысы,қортындылай келгенде,көлік жүйесі қаншалықты сұранысқа ие және қаншалықтысапалы болатындығын анықтайды.

Көлік сұранысы қала тұрғындарының қозғалысқа деген қажеттілігін сандық және сапалық түрде анықтайды.

Көлік ұсынысы:

- Картографиялық ақпарат (Қаланың сандық жоспары)
- Әртүрлі көлік түрлеріне арналған қозғалыс жолдарының желісі, ұйымның техникалық құралдарын қоса алғанда, оның қасиеттері мен қозғалыс шарттары жол қозғалысы
- Көшелер мен жолдардың түрлері, орташа жылдық тәуліктік қарқындылық, аралықтар мен қиылыстардың өткізу қабілеті және т. б.

Көлік сұранысы:

- Статистика деректері: халық туралы, еңбекке қабілетті халық туралы, жұмыс орындары туралы, қызмет көрсету саласындағы жұмыс орындары туралы, студенттер саны және оқу орындары туралы мәліметтер
- Корреспонденцияларды бөлу бойынша сапарлар туралы статистика деректері
- Modal Split: көлік ағындарының түрлері бойынша жалпы бөлінуі зерттелетін аумақтағы көлік.

Мұндай көлік моделіндегі болжамды есептеу

төрт сатылы алгоритм, нәтижесінде мұндай болжамды көлік модельдері "төрт сатылы" деп аталады. Мұндай модельді құру және одан кейінгі жұмыс кезінде төрт кезеңді (қадамды)бөлуге болады:

1. Сұраныс генерациясы (Trip Generation)
2. Сұранысты бөлу (Trip Distribution)
3. Режимді таңдау(режимді таңдау)
4. Қайта Бөлу (Traffic Assignment).

Жол адамдар мен көліктердің қозғалысы үшін қатынас жолы, көлік құрылымының ажырамас бөлігі. Автомобиль жолы-бір жолды, көп жолды, қарама-қарсы, сондай-ақ механикалық көлік құралдары қозғалысының ілеспе бағыты бар жол. Термин мыналарды қамтиды қамтамасыз етуге арнайы арналған функционалдық байланысты құрылымдық элементтер мен жасанды инженерлік құрылыстар кешені автомобиль және басқа да көлік құралдарының есептелген жылдамдықтарымен, жүктемелерімен және габариттерімен, берілген қарқындылығымен қауіпсіз қозғалысы ұзақ уақыт бойы қозғалыс, сондай-ақ осы кешенді орналастыру үшін берілген жер учаскелері және белгіленген көлем шегіндегі кеңістік.

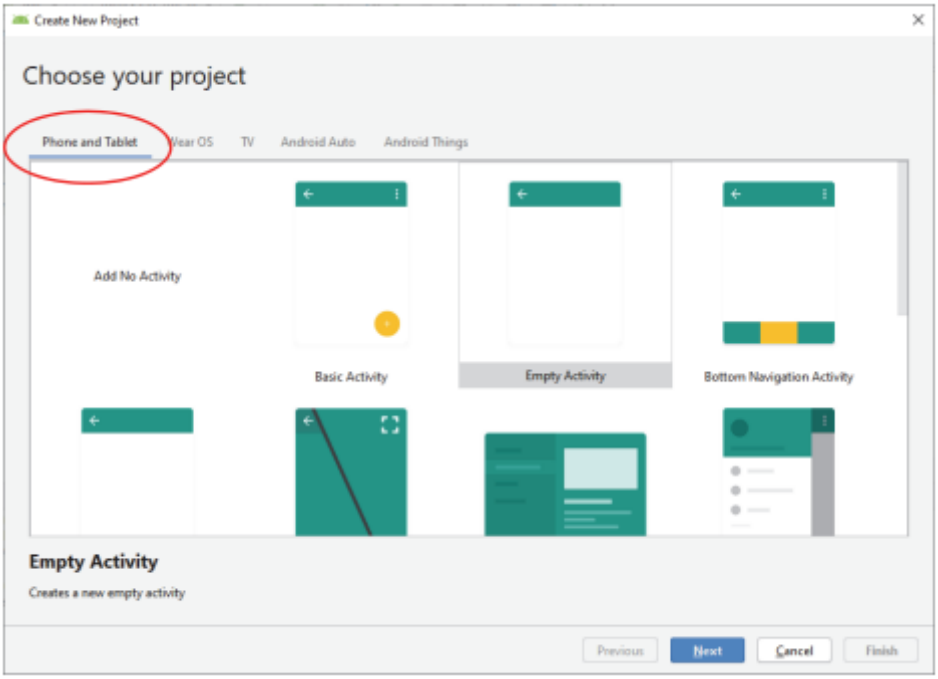
3.3 Такси қызметіне арналған Android қосымшасы

Android операциялық жүйесіне арналған қосымшаларды әзірлеу үшін SDK жүктеу және орнату қажет. Қазір бірнеше әзірлеу ортасы бар:

– NetBeans;
– Eclipse;
– Intellij IDEA;
– Android Studio.

Android Studio дәл Android ОЖ-ге бағытталған, сонымен қатар қосымша плагиндерді орнатуды қажет етпейді. Орындалу мысалдары бұл оқулықтағы тапсырмалар Android Studio-да қарастырылады. Жергілікті қосымшаларды әзірлеу тілдерін қарастырыңыз. Java-ресми бағдарламалау тілі Android Studio дамыту ортасы. Google-дің көптеген ресми құжаттары Java-ға сілтеме жасайды. Котлин-тіл 2017 жылдың мамыр айында Google I/O-да ресми түрде енгізілді және Google Java-дан кейін Android үшін екінші ресми бағдарлама тілі ретінде орналастырылды. Котлин Java-мен үйлесімді және өнімділіктің төмендеуіне және файл көлемінің ұлғаюына себеп болмайды. Бастап жеке java-дан ол аз қызметтік кодты қажет етеді, сондықтаноқу оңай. Төменгі деңгейдегі тілдерді Java NDK (Native Development Kit) пайдалануымен Android Studio қолдайды. Бұл жазуға мүмкіндік береді ойындар жасау үшін пайдалы болуы мүмкін жергілікті қосымшалар немесе басқа ресурстарды қажет ететін бағдарламалар.

Осы оқу-әдістемелік құралда келтірілген мысалдар, Java тілінде жазылған. Android Studio-да жаңа жоба жасау үшін (13 - сурет) белгілі бір қолданба үлгісін таңдау қажет.

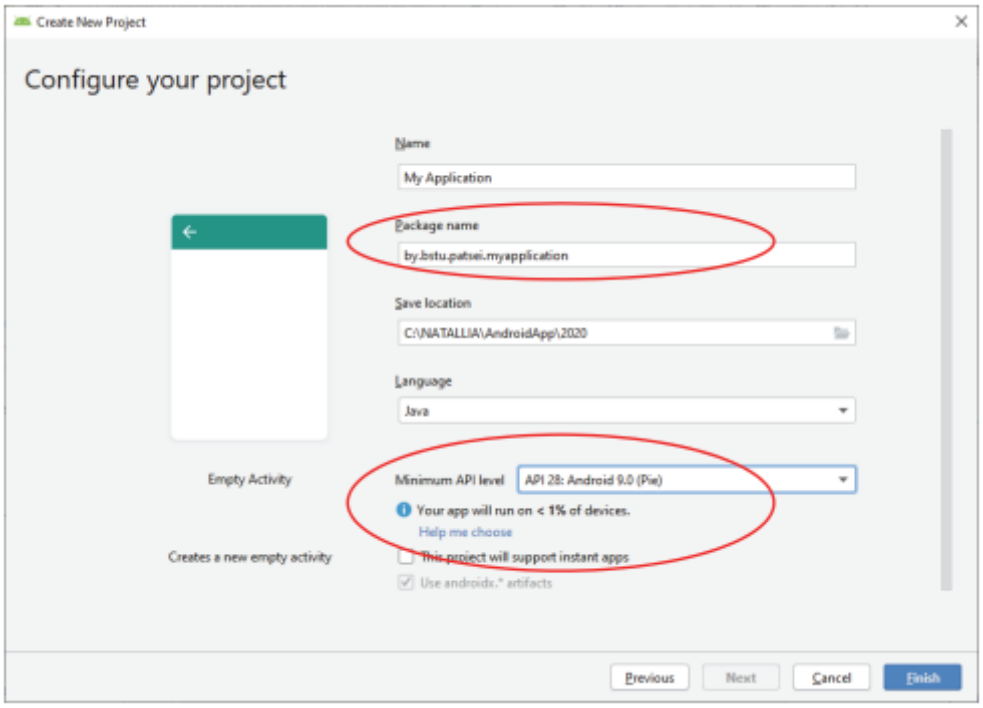


13 – сурет. Үлгіні анықтау терезесі

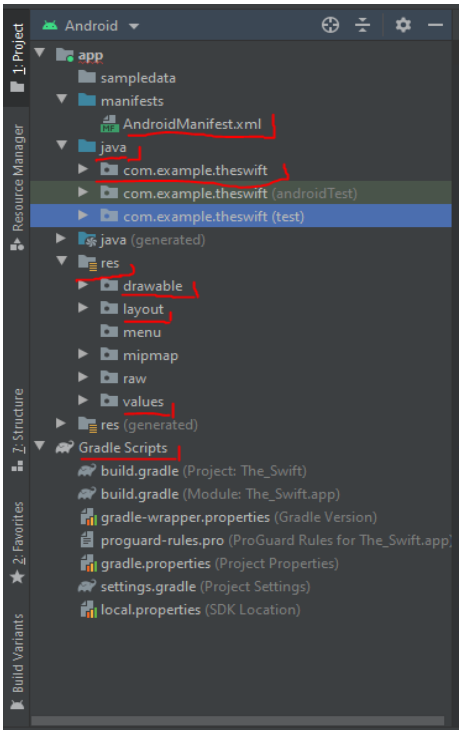
Содан кейін package name параметрін орнатқан кезде кері домен атауын пайдалану керек, мысалы by.sta.fit.аты-жөні.

Келесі қадам – API нұсқасын орнату (14 – сурет). Жаңа API таңдасаныз қолдау көрсетілетін құрылғылардың аз пайызы беріледі. Бағдарлама компоненттері-бұл қосымшаны құрайтын блоктар. Әрбір компонент-бұл жүйе бағдарламаға кіре алатын жеке нүкте. Барлық компоненттер пайдаланушы үшін кіру нүктелері бола бермейді. Алайда, әр компонент-бұл қосымшаның жұмысын тұтастай анықтайтын тәуелсіз құрылымдық бірлік. Қосымшаның компоненттерін бес түрдің біреуіне жатқызуға болады.

Жасалған жобаға оралайық . Оның құрылымында бар жалғыз модуль – app модулі. Код оның ішінде орналасқан модуль.



14 – сурет. Android қосымшасының жобасын жасау терезесі



15 – сурет. Android қосымшасының жобасын жасау құрылымы

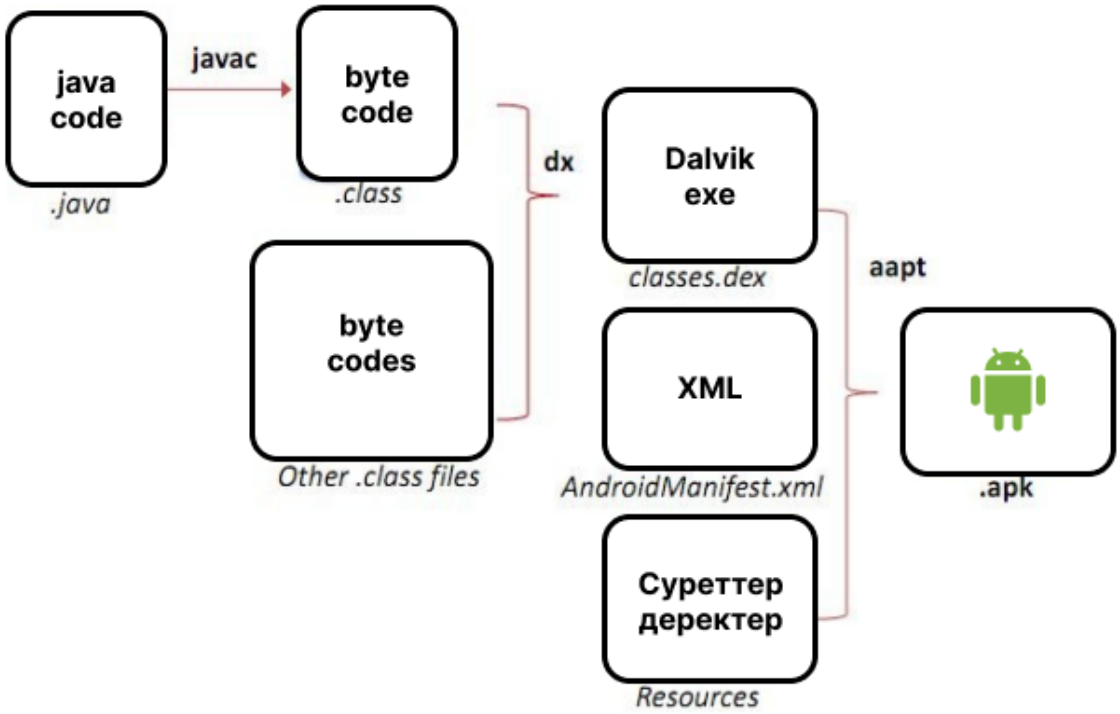
1.AndroidManifest.xml - Конфигурацияны анықтайды(15 – сурет)
2.java – Код Java
3.MainActivity - Белсенділік класын анықтайды
4.res - Жүйелік ресурстар
5.drawable - Суреттер қалтасы
6.layout - Жоба макеттері

7.values - Жол жұптарының файлдары "идентификатор-мән"

8.Gradle Scripts - Gradle сценарийлері қосымшалар құру кезінде пайдаланылады[9].

Бұл тек java да жазылған код емес (16 – сурет). Android қолданбасында қосымшаның жұмыс істеуі үшін көптеген басқа компоненттер мен ресурстар бар. Қосымшада қолданылатын жол тұрақтылары, сызбалар және басқа ресурстар артефакт деп аталады. Олар қосымшамен бірге жүктеледі. Осыған байланысты қосымшалар мен ресурстарды орау процесі бірқатар кезеңдерден тұрады. Java class және Java-code компиляторымен алынған javac элементтері DX компиляторының арнайы өңдегіші арқылы операциялық жүйе үшін арнайы Byte code файлдарына айналады. Бұл сонымен қатар қосымшаның контекстінде орналасқан арнайы конфигурация файлдарын қамтиды. Онда қосымшаның негізгі компоненттері туралы ақпарат сақталады. Мұнда арнайы құрал арқылы apk файлына оралған ресурстар бар. Негізінен, бұл файл zip мұрағаты, бірақ JAR емес, өйткені оның құрамында бірнеше компоненттер бар.

Тағы бір негізгі файл-бұл Android manifest, онсыз оны орнату, іске қосу және орындау мүмкін емес



16 – сурет. Қолданба файлының құрылым

android manifest-бұл әзірленіп жатқан қосымшаның негізгі конфигурация файлы. (17 – сурет) Әр бағдарламада manifest бар, онда Бағдарлама Атауы, package name және бағдарламада қолданылатын барлық компоненттер сияқты мәліметтер бар. Сондай – ақ, applications арнайы блогы бар-бұл ақпарат бағдарлама параметрлері мен android компоненттері, activity, service, receiver және мазмұн провайдерлері, сонымен қатар қосымшаларды іске қосу ерекшеліктері, оларға рұқсаттар туралы толық ақпарат.Манифест файлының мысалы суретте көрсетілген

17 – сурет. Android Manifest мысалы

manifest файлында қосымшаның барлық кіру нүктелерін сипаттайтын, Операциялық жүйе процесті қалай бастауға немесе шақыруға және өзара әрекеттесуді бастауға болатын төрт топ бар. Демек, Android жоқ manifest және осы топтардың сипаттамалары жүйе ештеңе істей алмайды.Бұл басқа бағдарламалардан айтарлықтай айырмашылық, бұл көптеген іске қосу жолдарын ескере отырып, қосымшаларды әзірлеу процесін айтарлықтай қиындатады.Компоненттердің өздері өз міндеттеріне қарай бөлінеді.

Әрекет-бұл негізгі элемент, ол көретін қосымшаның бөлігі сіз өзара әрекеттесетін пайдаланушы. Тегінде қосымшаның көрнекі бөлігі.

Қызмет-бұл интерфейсі жоқ арнайы компоненттер, олар пайдаланушымен тікелей араласпайды. Олар қолданбаның қандай да бір фондық тапсырмасын орындауға арналған.

Receiver-бұл жүйелер жасай алатын арнайы хабарламаларды алуға мүмкіндік беретін арнайы компоненттер таратуы. Мысалы, Операциялық желі сигналы жоғалған кезде жүйе арнайы хабарлама жібереді және бағдарлама оқиғалардың осы түріне жазыла алады.

3.4 React қосымшасының құрылымы

React-пен жұмыс жасағанда, әрқашан жобамыздың дұрыс құрылыммен жасалғанына мән береміз. Әрқашан дерлік, құру конфигурациясынсыз қосымшаны құру үшін **create-react-app** қолданбасы қолданамыз. Көлік

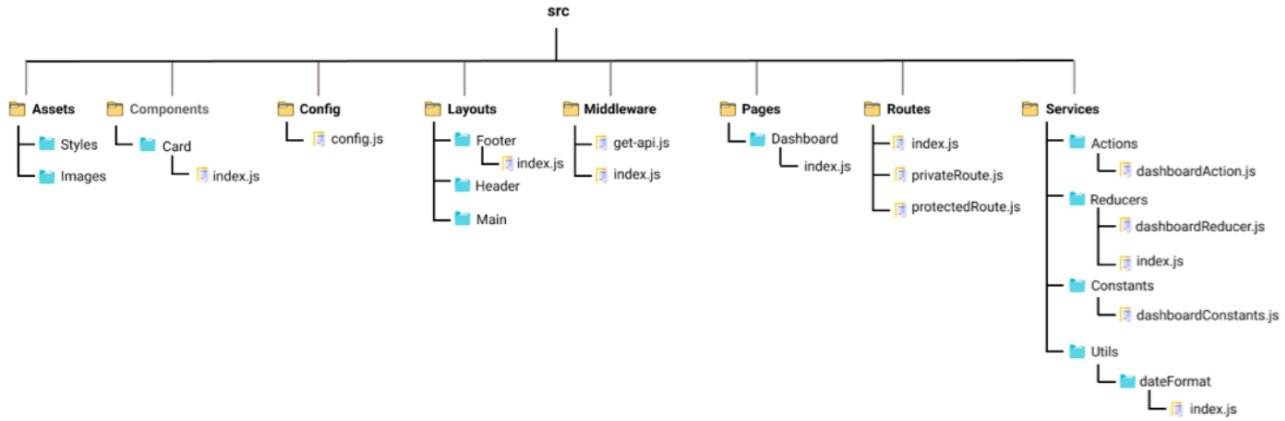
құралдарын бақылау жүйесін біз белгілі бір шаблонда жасаймыз. React қауымдастығының көптеген әзірлеушілері қолданатын ең жақсы шаблондық құрылымдарды қолданамыз. (18 – сурет)

Алдымен react js-пен жұмыс істеу үшін пайдалануға ыңғайлы даму ортасын құрамыз.

- Код редакторы ретінде әртүрлі кеңейтімдері бар VS Code.
- Google Chrome браузер ретінде әзірлеушілерге ыңғайлы жөндеу құралдары.
- Жөндеу құралы ретінде React және Redux DevTools әзірлеуші құралдары, өйткені олар өте тиімді болып келеді.

Енді біз жаңа жобаны жасау үшін реактивті CLI қолдана аламыз. Create-react-app көмегімен жасалған React js әдепкі кодты береді. Логотиптер, суреттер, стильдер және т.б. сияқты барлығын дерлік жойып, қосымшаның қаңқасын сақтайық, енді біз директориялар құрылымын жасаймыз.

React қосымшасының директориялар құрылымы



18 – сурет. React қосымшасының директориялар құрылымы

Assets директориясы

Атауынан көрініп тұрғандай, онда біздің жобамыздың активтері бар. Ол суреттер мен стиль файлдарынан тұрады. Мұнда жаһандық (global) стильдерімізді сақтай аламыз. Біз жобаны орталықтандырамыз, сондықтан мұнда бет немесе компоненттерге негізделген стильдерді сақтай аламыз. Бірақ сонымен қатар стильді бет директориясына немесе компоненттер директориясына сәйкес сақтай аламыз. Бірақ бұл әзірлеушінің ыңғайлылығына байланысты.

Layouts директориясы

Атауынан көрініп тұрғандай, ол бүкіл жоба үшін қол жетімді макеттерді қамтиды, мысалы, жоғарғы, төменгі деректеме және т.б. біз мұнда жоғарғы, төменгі деректеме немесе бүйірлік тақтаның кодын сақтап, оны шақыра аламыз.

Components директориясы

Компоненттер – кез-келген react жобасының құрылыс блоктары. Бұл папкада әр түрлі жобалық файлдарда қолдануға болатын түймелер, модельдер, кірістер, жүктеушілер және т.б. сияқты UI компоненттерінің жиынтығынан тұрады. Әрбір компонент блокты тестілеу үшін сынақ файлынан тұруы керек, өйткені ол жобада кеңінен қолданылады.

Pages директориясы

Беттер папкасындағы файлдар react қосымшаның бағытын көрсетеді. Осы папкадағы әр файлдың өз бағыты бар. Бетте оның ішкі папкалары болуы мүмкін. Әр парақтың өзіндік жағдайы бар және әдетте асинхронды операцияны шақыру үшін қолданылады. Ол әдетте топтастырылған әртүрлі компоненттерден тұрады.

Middleware директориясы

Бұл папкада бағдарламада жанама әсерлерді қолдануға мүмкіндік беретін аралық бағдарламалық жасақтама болады. Ол реакциямен редукцияны қолданғанда қолданылады. Мұнда біз барлық аралық бағдарламаларды сақтаймыз.

Routes директориясы

Бұл папка қосымшаның барлық маршруттарынан тұрады. Ол жеке, қорғалған және маршруттардың барлық түрлерінен тұрады. Мұнда біз тіпті кіші бағдарымызды атай аламыз.

Config директориясы

Бұл папка конфигурация файлынан тұрады, онда біз қоршаған орта айнымалыларын config.js -де сақтаймыз. Біз бұл файлды қолданбадағы бірнеше орталардың конфигурацияларын реттеу үшін қолданамыз.

Services директориясы

Егер біз жобамызға redux қолдансақ, бұл папка іске қосылады. Оның ішінде атаулары бар 3 папка болады: күйлерді басқаруға арналған әрекеттер, редукторлар және тұрақты ішкі папкалар. Әрекеттер мен өңдеушілер барлық беттерде шақырылады, сондықтан беттердің атауына сәйкес әрекеттерді, өңдеушілерді және тұрақтыларды жасай аламыз.

Utils директориясы

Utils папкасы жобада жиі қолданылатын кейбір қайта пайдаланылатын функциялардан тұрады. Онда тек жалпы функциялар мен JS нысандары болуы керек, мысалы, ашылмалы параметрлер, тұрақты өрнек жағдайы, деректерді пішімдеу және т. б.

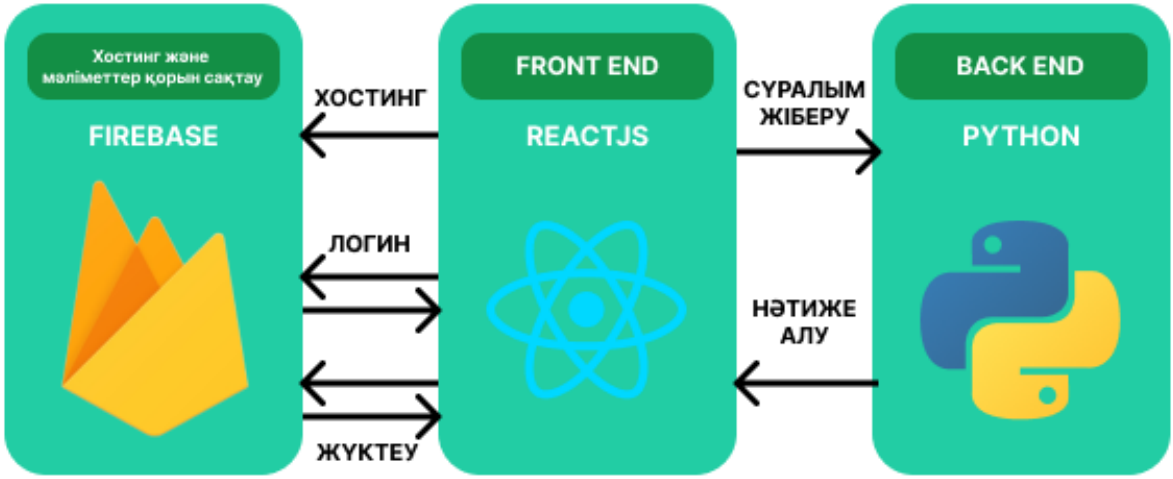
3.5 Деректерді сақтау қоры Firebase

Firebase – бастапқыда нақты уақыттағы дерекқорымен танымал әзірлеу платформасы, оның негізі ретінде деректерді синхрондау үшін оңтайландырылған, көбінесе пайдаланушы машиналары немесе смартфондары және бұлттағы орталықтандырылған сақтау үшін оңтайландырылған көп түйінді, кілт-мәнді дерекқор болып табылады. Ол деректерді қабылдау мен таратудың көп бөлігін өңдеу арқылы әзірлеушілердің өмірін жеңілдетуге арналған платформа. Бұл қолданба әзірлеушілерін нұсқаларды немесе деректерді басқарумен байланысты бағдарламалау жүктемелерінен босатады. Олар Firebase-ге жаңа биттерді жаза алады және деректер бүкіл жүйеде сәйкесінше болады. (19 – сурет)

Firebase негізінен бұлтта сақталған нұсқалармен пайдаланушылардың машиналарында сақталған ақпараттың жергілікті көшірмелері арасындағы өзгерістерді үнемі таратып, синхрондауы мүмкін болғандықтан бағаланады. Firebase аутентификацияны, синхронизацияны және сегрегацияны араластырудың көптеген қиындықтарын бірнеше нұсқаларды біріктіру және дұрыс биттердің бүкіл жүйеде бірдей болуын қамтамасыз ету арқылы жояды.

Firebase бір физикалық компьютермен шектелмейтін деректер қорының түрін алуға болады. Оның заманауи түрі деректер жиынын бөлу, олардың биттерінің көшірмелерін жасау немесе екеуі де арқылы бірнеше машиналар арасында жұмыс жүктемесін бөлуге мүмкіндік береді. Firebase пайдаланушылардың телефондарында немесе жұмыс үстелінде сақталған деректерді үлкен дерекқордың жергілікті нұсқалары ретінде қарастыру арқылы деректер орталығының үйлесімділігі үшін пайдаланылатын алгоритмдерді бүкіл желіге кеңейтеді. Негізінде телефон немесе ноутбук енді бұлттың бір бөлігі ретінде қарастыруға болады.

Firebase Cloud Messaging пайдаланушыларды аты немесе тақырыбы бойынша топтастыру арқылы хабарларды жіберу процесіне ұйымның қосымша деңгейін қосады. Баптандырудан кейін Firebase оқиға хабарландыруларын алдын ала анықталған топтарға немесе белгілі бір тақырыптарға жазылған пайдаланушыларға хабар ретінде жібере алады.



19 – сурет. Firebase деректер алмасу процессі

Такси көлік құралдарының бақылау жүйесіне шолу

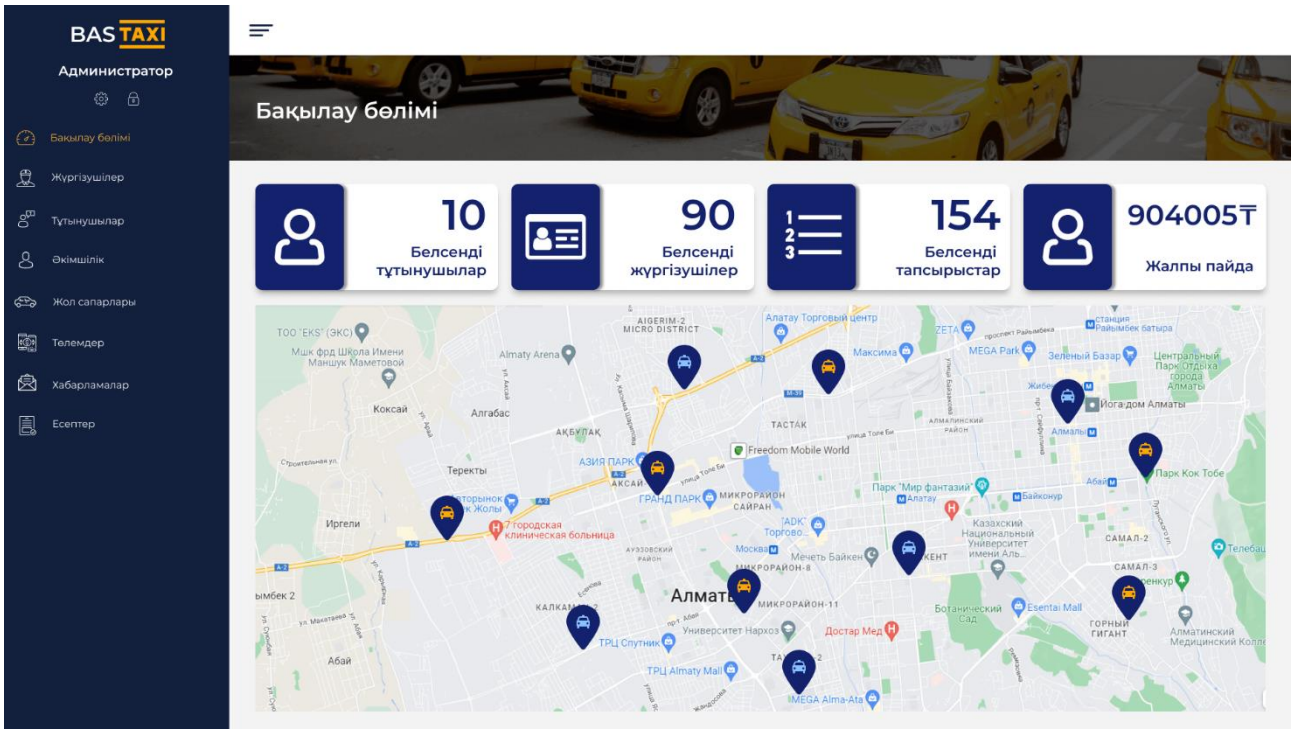
Диспетчерлік панель эксклюзивті кіруді ұсына отырып, такси жүргізушілерін, таксилерді, тұтынушыларды және диспетчерді басқарудың шексіз мүмкіндіктерін ұсынады. Оған қоса, ол жүру есептерін қадағалау арқылы прогресті бақылауға және әрекет ететін шешімдер қабылдауға көмектеседі.

Бүкіл жүйенің қысқаша мазмұнын қарастыраты болсақ. Нақты уақыттағы картада таксилер мен белсенді жүргізушілерді қадағалауға болады. Ағымдағы сапарларды, хабарландыруларды және т.б. тексеруге мүмкіндік береді (19-сурет):

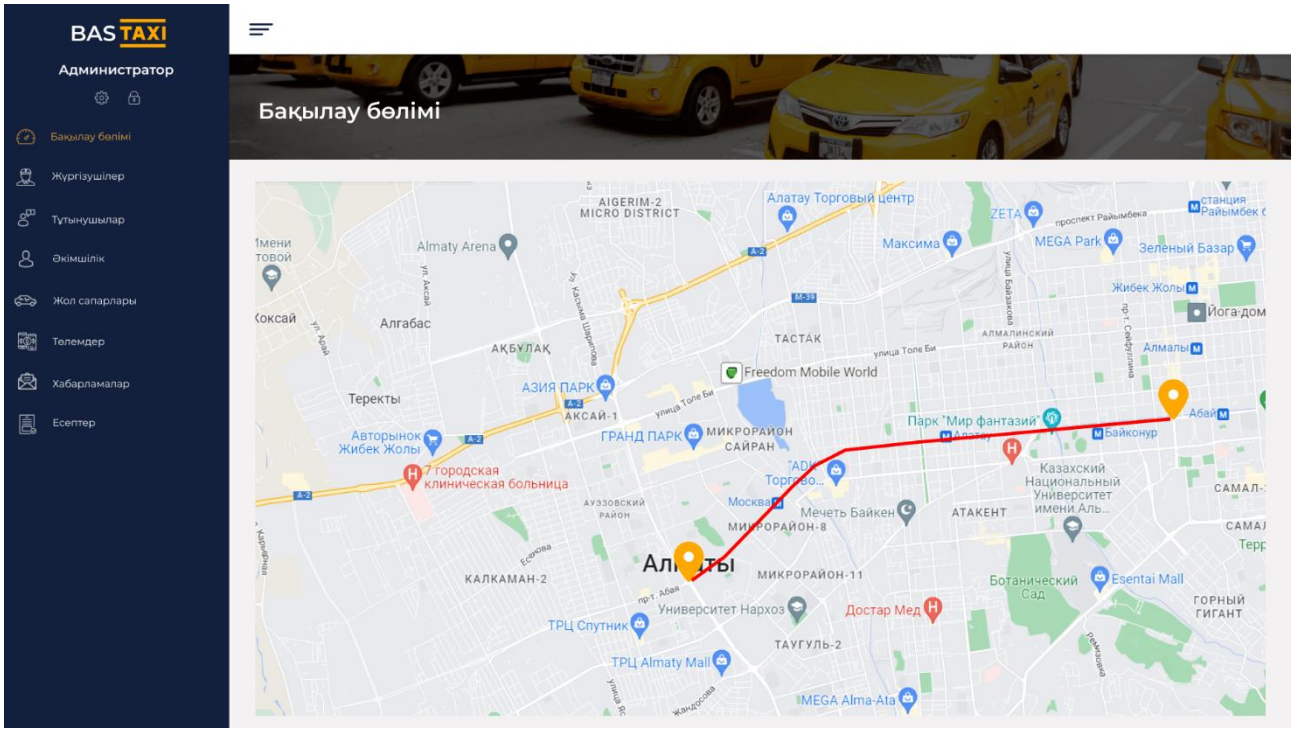
- Белсенді драйверлердің санын тексеруге болады
- Белсенді тұтынушылар саны
- Белсенді брондаулар
- Ағымдағы сапарлар
- Табылған жалпы пайда т.б.

Нақты уақыт режимінегі карта

Әкімшілік картаны нақты уақыт режимінде көре алады және таксиді Google картасында нақты уақыт режимінде бақылай алады. Түстер көмегімен бос немесе бос тұрған барлық таксилер туралы ақпарат алыңыз. Мұнда сары таксилер бос емес, ал көк таксилер бос. (20, 21 – суреттер)



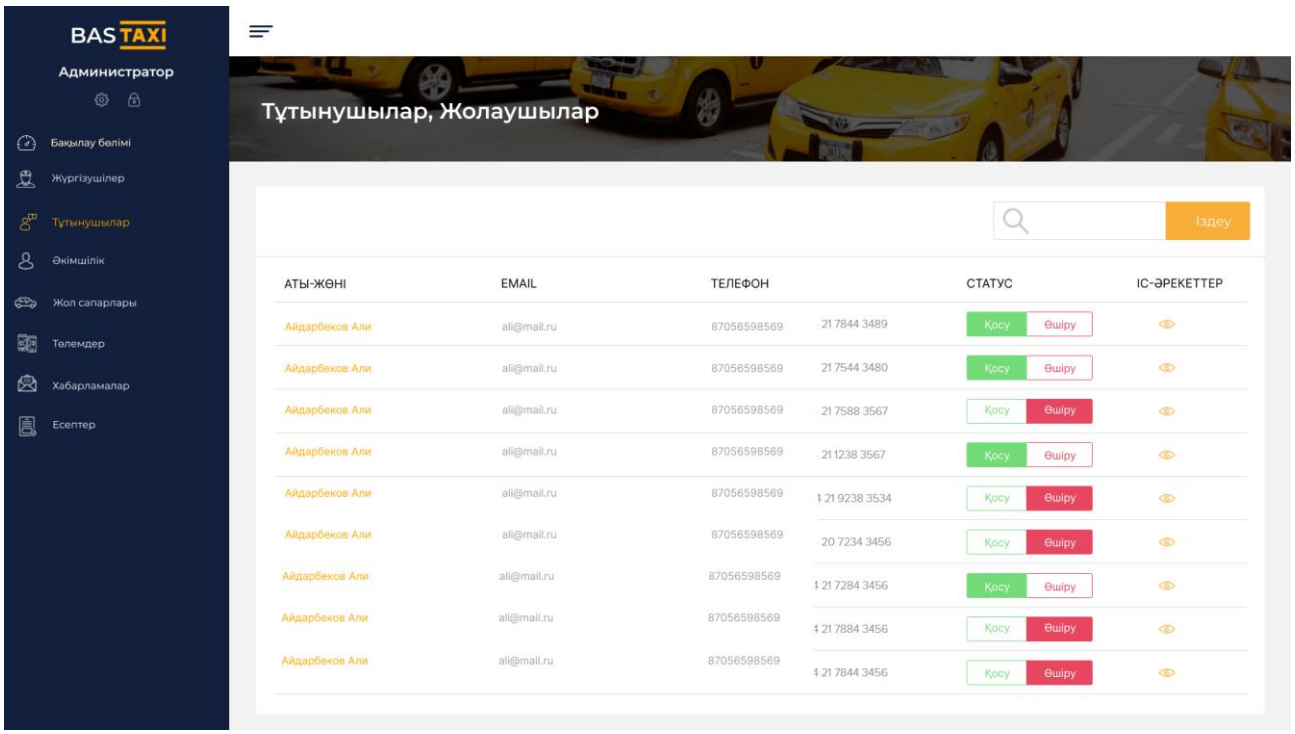
20 – сурет. Бақылау бөлімінің негізгі беті



21– сурет. Нақты уақыт режиміндегі бақылау картасы

Тұтынушыларды басқару

Барлық клиенттер туралы ақпаратты қарауға болады. Қажетті шараларды дер кезінде қабылдауға болады. Бұл бетте такси қолданушылары туралы барлық ақпараттар көрініп тұрады. Жүрілге сапарлар архиві және қанша ақша төлегені туралы мәліметтерді оңай бақылау мүмкіндік болады. (22 – сурет)

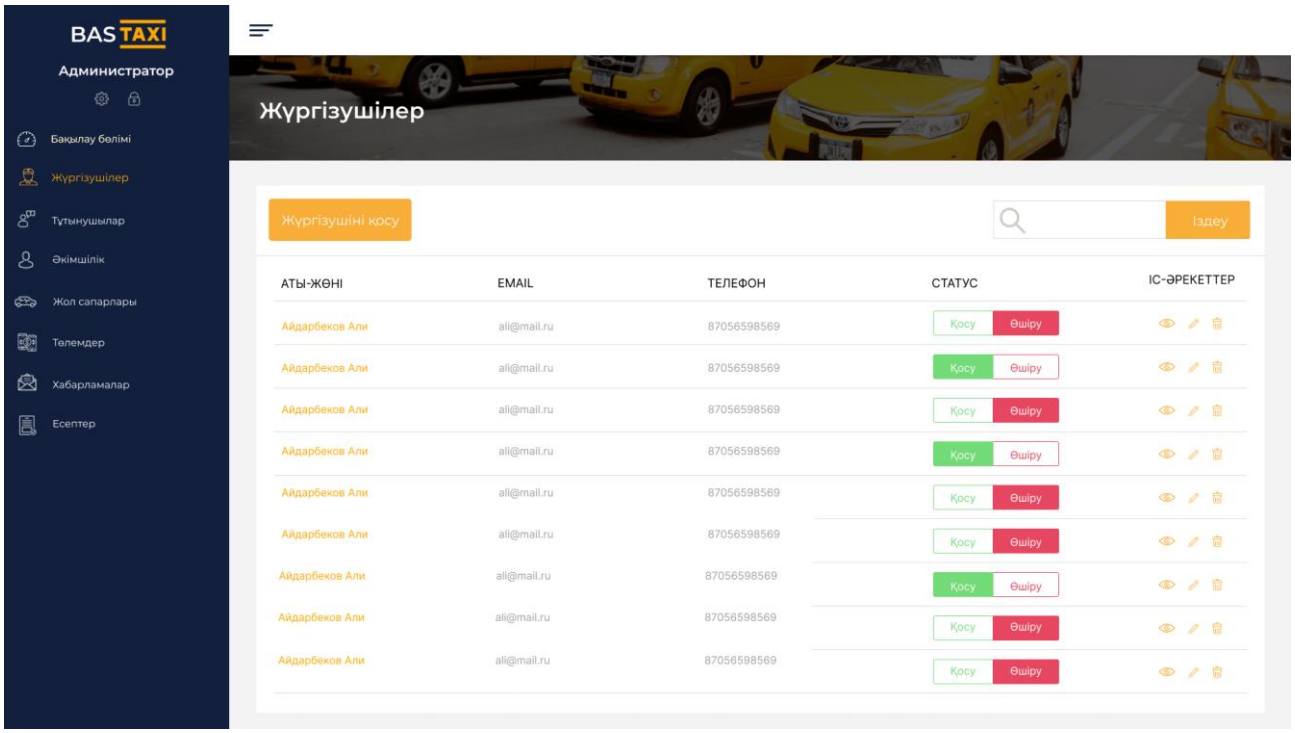


22– сурет. Тұтынушылар туралы ақпарат беті

Жүргізушілерді бақылау беті

Бұл бетті бақылаушы жүйеге жаңадан жүргізушілерді тіркей алады. Ережеге сәйкес шарттарды немесе аса қатты жол тәртібін бұзушы такси жүргізушілерін қосып немесе жүйеден шығарып жібере алады. Жүргізуші келесідегідей келісім шарттар мен ережелерді қатаң ұстануы тиіс (22-сурет):

- Барлық ережелер бойынша автомобильді дұрыс және кәсіби жүргізу, жолаушылардың өміріне, денсаулығына және мүлкіне зиян келтіруі мүмкін авариялық жағдайларды болдырмау;
- Автомобильді қауіпсіз пайдалану, шанақ пен салонды таза ұстау, көлік құралына күтім жасау, уақтылы жөндеу;
- Машинаның және оның ішіндегісінің материалдық сақталуын қамтамасыз ету: автомобильді қараусыз қалдырмау, аялдамаларда есіктер мен терезелерді бұғаттау, сигнализацияға қою, тек күзетілетін және заңды тұрақтарда ғана қою;
- Тікелей басшының бұйрықтарын орындау, оны жүргізушінің барлық проблемалары туралы хабардар ету. Соның ішінде басшыны жүргізуге кедергі келтіруі мүмкін (алкогольдік немесе есірткілік мас болу жағдайы, психотроптық, ұйықтататын дәрілерді және антидепрессивті препараттарды қабылдау және т. б.) өзінің хал-ахуалы туралы да хабардар ету қажет.;
- Жол парақтарын күнделікті жүргізу. (23 – сурет)



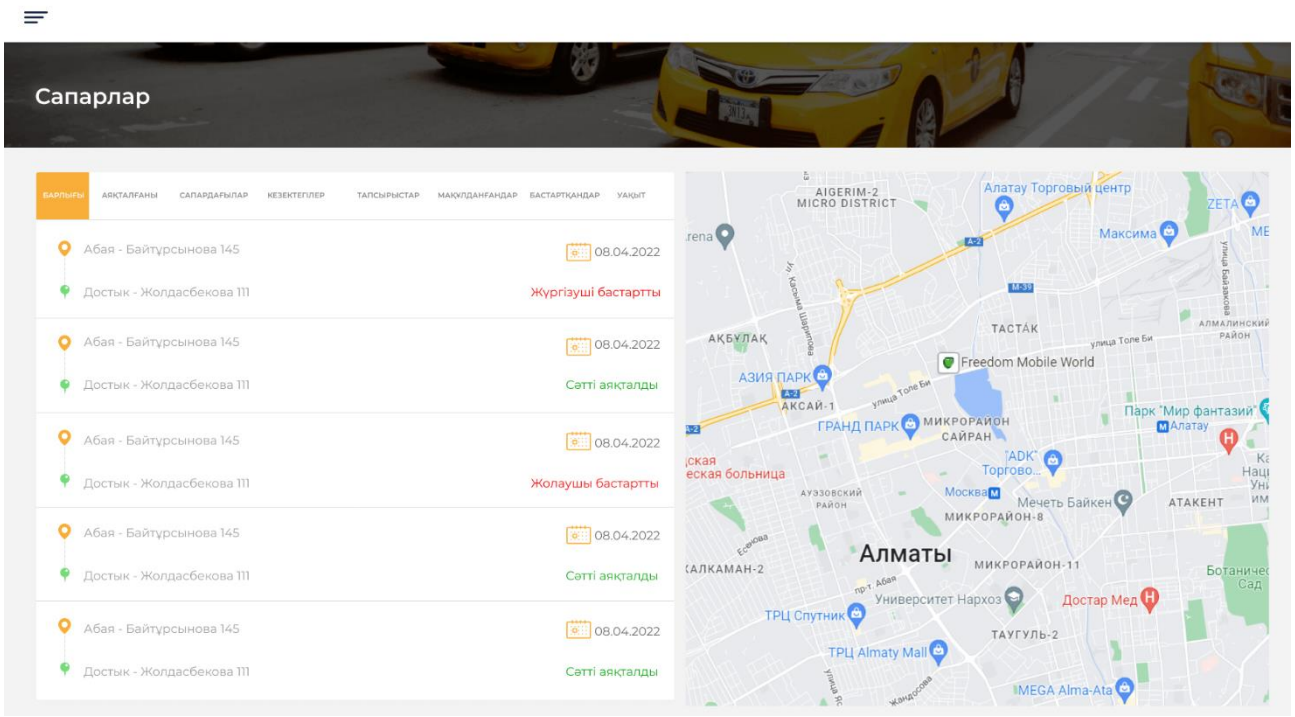
23– сурет. Жүргізушілерді бақылау беті

Сапарларды бақылау беті

Әкімшілік барлық орындалған, ағымдағы, сұралған, қабылданған және тоқтатылған сапарларды тексере алады. Нақты уақыттағы карта жағында жұмыс істейді. Сапарлар тарихы басқа да бірқатар жағдайларда пайдалы болады:

– көлікте ұмытылған затты қайтару үшін;
– жолаушыға қатысты өзін дұрыс ұстамаған такси жүргізушісіне шағым жасау үшін.

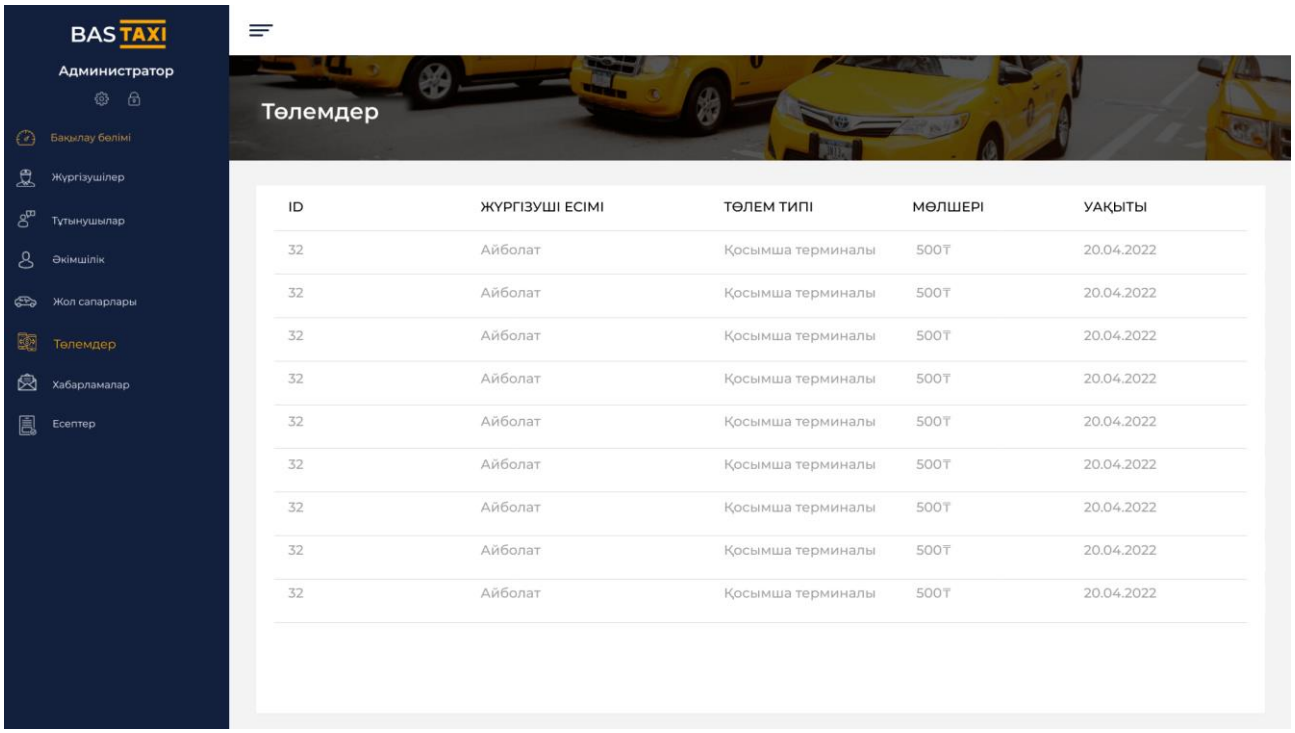
Жүргізушілерге тапсырыс тарихы клиентке салонда ұмытылған затты тез қайтаруға мүмкіндік береді. Бұрын табылған зат туралы мәліметтер таксопаркке немесе Қолдау қызметіне берілетін. Делдалдарды тарту арқылы ұмытылған нәрсені қайтару процедурасы уақытқа созылды. Тапсырыс тарихының арқасында такси жүргізушісі клиентпен тікелей байланыс орнатуға мүмкіндік алды, бұл екі жаққа да өте ыңғайлы. (24 – сурет)



24 – сурет. Сапарларды бақылау беті

Төлемдер транзакцияларының беті

Жүргізушілердің жеке белгілі бір уақытта тапқан жалпы пайдаларының соммасы төлемдер бетінен қарауға және бақылауға болады. Аналитикалық сараптамаларды жүргізу үшін такси жүргізушілерінің күндік, айлық және жылдық жалақылары есептелініп отыру қажет. Тоқталымда тұрған автокөлік иелеріне такси кәсіпорынан оларға жалақы берілмейді, яғни айтқанда жүргізушілер күндік жоспарды орындап ақшасын және қосымша бонустарын алады. Бұл бетте барлық транзакциялар қай уақытта кімнің есепшотына түскені туралы ақпарат көрсетіледі. (25 – сурет)



25 – сурет. Төлемдер транзакцияларының беті

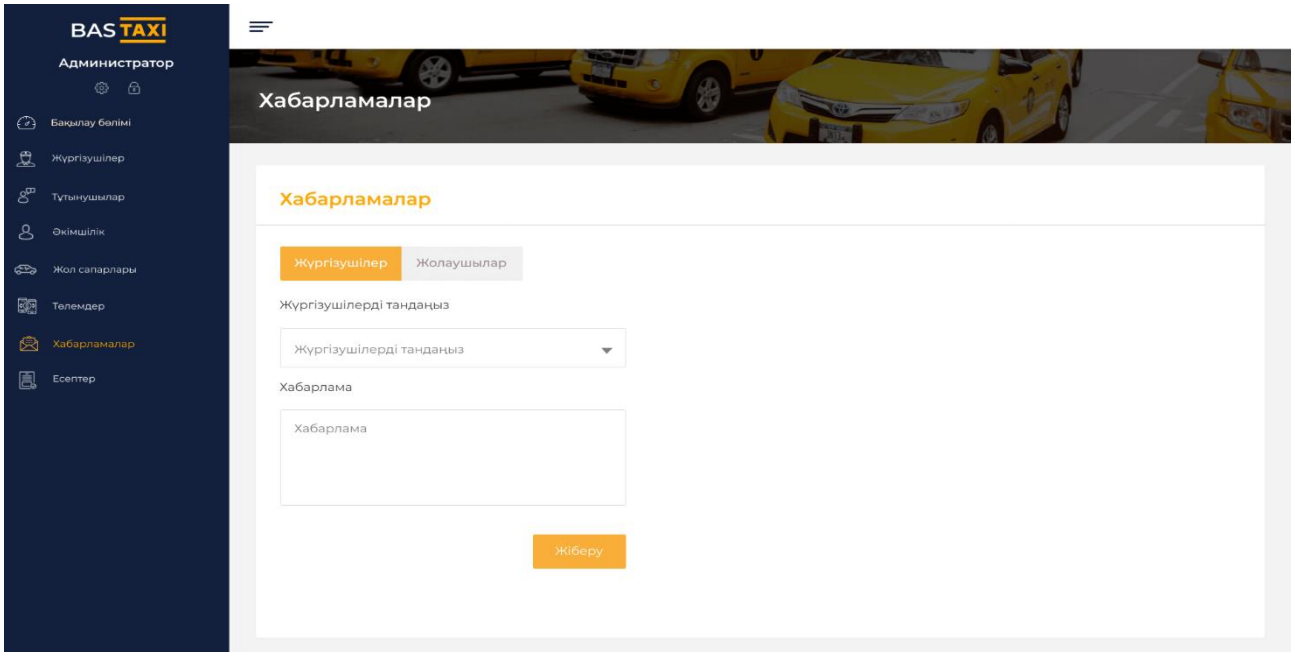
Хабарлама жіберу беті

Телефонмен сөйлескеннен гөрі хабарлама жазу ыңғайлы болатын көптеген жағдайлар бар. Қазіргі смартфондарда қоңырау шалу-бұл байланыс тәсілдерінің бірі және енді маңызды емес. Күн сайын біз мессенджерлерде ондаған жұмыс және жеке мәселелерді шешеміз — такси жүргізушісімен қарым-қатынас жасау көптеген адамдарға мәтіндік түрде де ыңғайлы болар еді.

Чатпен байланыс жасау үшін жүргізуші тапсырысты қабылдап, жолаушыға қарай бастаған сәтте қол жетімді болады. Пайдаланушы жүргізушіге хабарлама жібере алады, сонымен қатар жиналыс орнын нақтылау үшін геолокациямен бөлісе алады. Жауап ретінде жүргізуші жолаушының хабарын алғанын растау немесе өзінің хабарын жіберу арқылы ұнай алады.

Жолдан шықпау үшін жүргізуші хабарлама термейді, бірақ оны жүргізуші қосымшасында — таксометрде жазады. Ал жолаушы хабарламаны мәтіндік түрде алады.

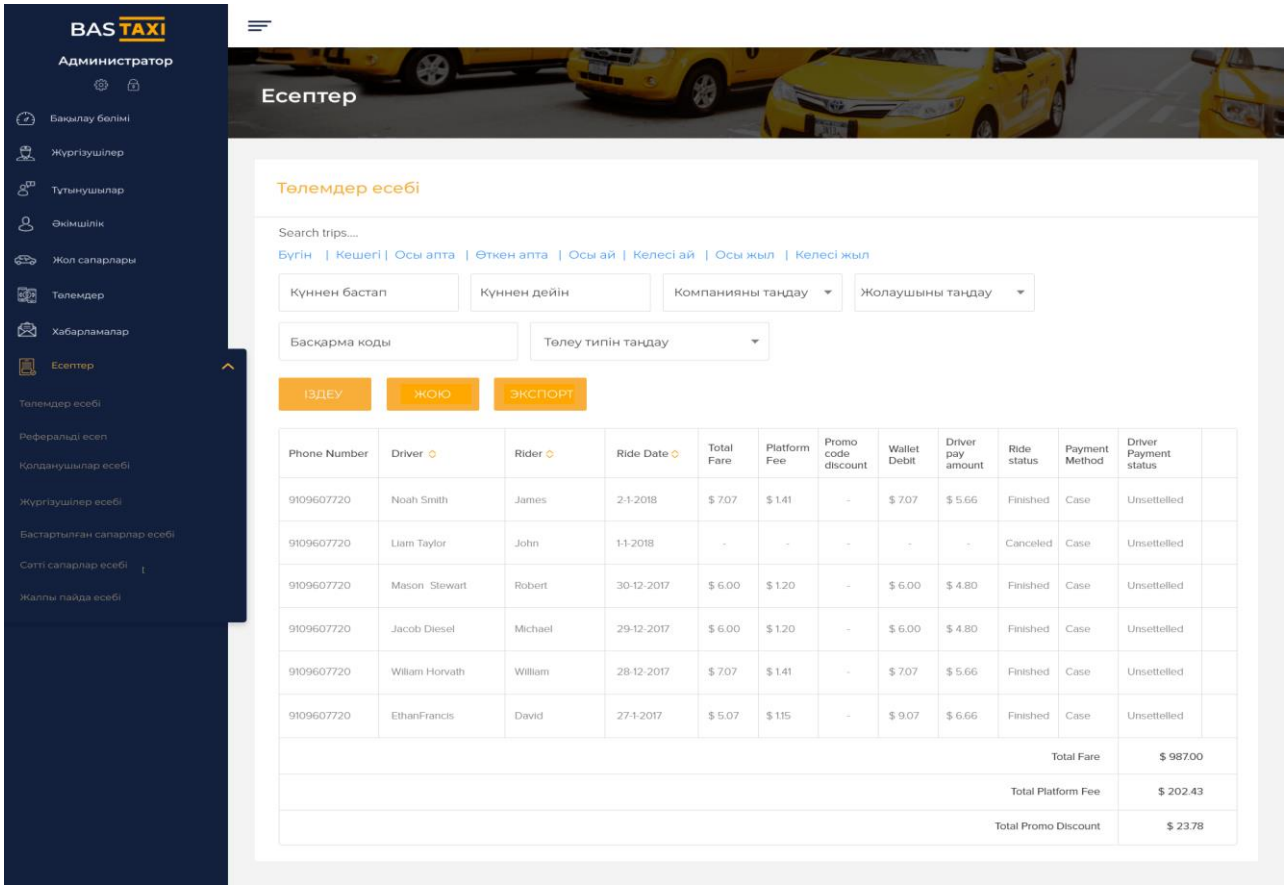
Дауыстың мәтінге айналуы үшін біз Speechkit — Яндекстің сөйлеу технологияларының кешенін қолдандық, оған сөйлеуді синтездеу және тану, сөйлеудегі семантикалық екпіндерді таңдау және қосымшаны дауыстық басқару кіреді. Біздің чаттарымызда технологиялар сапардың екі қатысушысына да көмектеседі-хабарламаны жүргізуші ғана емес, жолаушы да, егер ол оған ыңғайлы болса, жаза алады. (26 – сурет)



26 – сурет. Хабарлама жіберу беті

Есептеме алу беті

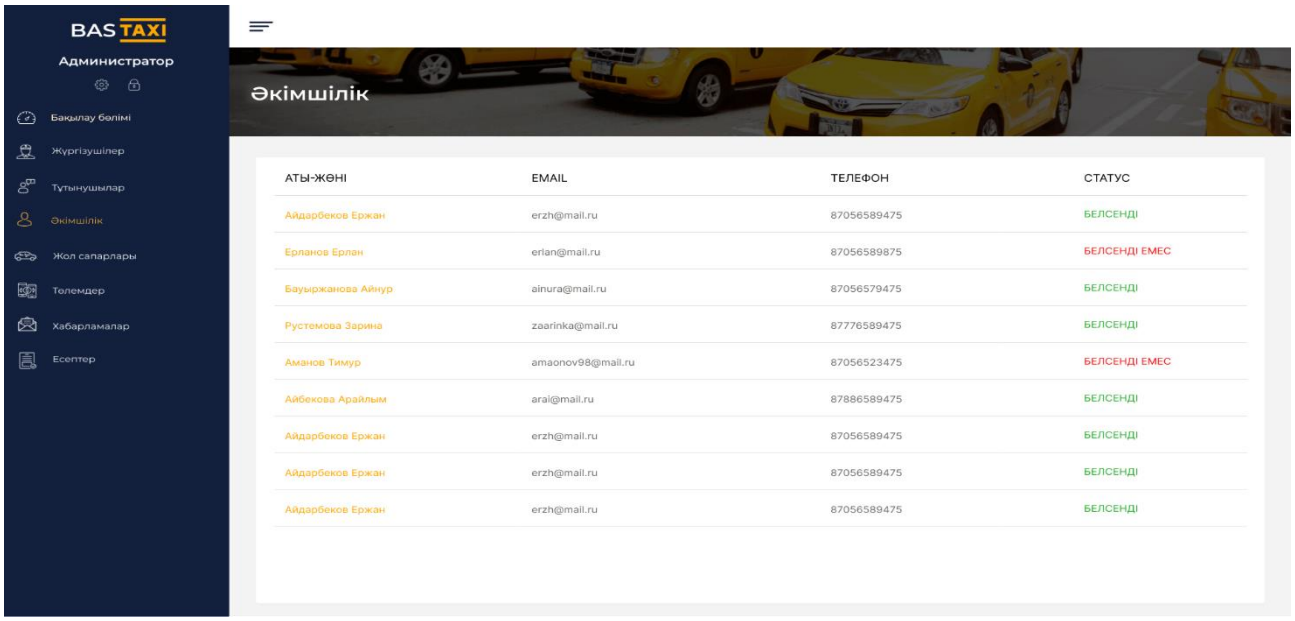
Есептеме алу бетінде төлемдер туралы сәтті және сәтсіз аяқталған сапарлар туралы ақпарат алып, сондағы жалпы пайданы есептеуге мүмкіндік бар. Белгілі бір такси жүргізушінің сапар барысында тиімді маршрутпен қаншалықты дұрыс жеткізгендігі және сол сапарда жанармайды қаншалықты үнемдеп құртқаны туралы ақпараттарды алып, pdf файл ретінде жүктеп алуға болады. (27 – сурет)



27 – сурет. Есептеме алу беті

Әкімшілк(администратор)

Әкімшілікті басқару бетінде жүйеге тіркелген бақылаушылардың тізімі шығып тұрады. Олардың әрқайсысының онлайн режимдегі белсенділігі көрініп тұрады. Әрбір администратор жүйедегі такси жүргізушілерін бақылай алады, яғни ақпараттарды өзгерте алады, енгізеді және өшіре алады. (28 – сурет)



28 – сурет. Әкімшілік бет

Бөлімге қорытынды

Қорытындылай келе үшінші бөлімде жасалынған такси көліктерін бақылау жүйесінің технологиялары мен интерфейстеріне тоқталып өтілді. Такси көліктерін бақылау үшін жүйелі түрдегі қосымша қажет болады. Яғни көлік жүргізушісінің арнайы интерфейсі және жасалынған іс әрекеттерді бақылайтын бақылау интерфейсі болады. Бақылаушы интерфейс түрлі платформада жасалынады(мобильді және вэб), бұл өз кезегінде такси көлік құралдарын әртүрлі құрылғыда кез келген уақытта тексеруге мүмкіндік береді. Google Map API интерфейстері құрылғыдан локацияны табу үшін қол жетімді. Көлік құралдарын бақылау жүйесі Google Api арқылы орналасқан жерді анықтауға мүмкіндік беретін бағдарламалық жасақтамамен біріктірілген. Құрылғы орналасқан жерді ендік және бойлық тұрғысынан анықтайды. Орналасқан жерді анықтау модулі пайдаланушының арнайы енгізуін қажет етпестен GPS қолданатын көлікті автоматты түрде орының бақылай алады. Алынған ендік пен бойлық GIS (ғаламдық ақпараттық жүйе) арқылы белгілі бір жердің координатасы арқылы анықтай алады. Осы орналасқан жерді анықтап алғаннан кейін такси жүргізушінің бастапқы мекенжайдан тапсырыс берілген мекенжайға бара жатқан траектироясын картадан көре аламыз. Firebase – бастапқыда нақты уақыттағы дерекқорымен танымал әзірлеу платформасы, оның негізі ретінде деректерді синхрондау үшін оңтайландырылған, көбінесе пайдаланушы машиналары немесе смартфондары және бұлттағы орталықтандырылған сақтау үшін оңтайландырылған көп түйінді, кілт-мәнді дерекқор болып табылады. Ол деректерді қабылдау мен таратудың көп бөлігін өңдеу арқылы әзірлеушілердің өмірін жеңілдетуге арналған платформа. Бұл қолданба әзірлеушілерін нұсқаларды немесе деректерді басқарумен байланысты бағдарламалау жүктемелерінен босатады. Олар Firebase-ге жаңа биттерді жаза алады және деректер бүкіл жүйеде сәйкесінше болады.

Қорытынды

Қазақстанда 18 миллионнан астам адам тұрады, оның 10 миллионнан астамы қала тұрғындары. Урбанизацияның жоғары дәрежесі қалалық және қала маңындағы қоғамдық көлік саласының ел өміріндегі маңыздылығын анықтайды. Қозғалыс қажеттіліктерін сезіне отырып, ел тұрғындары барлық тұрғындарды ескере отырып, бір тұрғын жылына таксимен 50-100-ден астам сапар жасайды. Экономикалық тұрғыдан белсенді халық үшін, әсіресе ірі қалалардың тұрғындары үшін такси көлігін пайдалану қарқындылығы әлдеқайда жоғары, сондықтан такси саласы тек әлеуметтік аспектіде ғана емес, сонымен бірге елдің жалпы ішкі өнімі көлемінің қалыптасуына зор үлесін қосады.

Жүйеде қолданылатын алгоритмдерді қорытындалай келе, зерттеу барысында ұсынылған алгоритмдер Android құрылғысында жасалынған такси қызметіндегі көлік құралдарын бақылау жүйесіне оң әсерін тигізді. Жүргізушілер мен жолаушылардың қауіісіздігінен бастап , қысқа уақыт ішінде межелеген жерге жету және сапар барысынды жанармайды үнемдеу секілді процесстер автоматты түрде орындалатын болғандықтан такси саласындағы кәсіпке қомақты пайда табуға ықпал етеді. Бұл жүйеде таксидің орналасқан локациясын және белгілі бір тапсырыста жүргендігін нақты уақыт режимінде бақылауға болады. Сонымен қатар, жүйе бақылаушысы жүйедегі процесстер туралы статистикалық мәліметер арқылы есеп алып, алдағы уақытта такси қосымшасын дамыту үшін өзгерістер енгізе алады, такси көліктерінің нақты уақыт режимінде жағдайын бақылап, жолдағы орын алған жол апатын жедел түрде шешуге көмектеседі.

Қазақстанның барлық аймақтарында қоғамдық көліктің диспетчерлік жүйесін енгізу мүлдем барлық машиналар мен жабдықтарды бір уақытта бақылауда ұстап, оны күтіп ұстауға кететін шығындарды айтарлықтай төмендетуге мүмкіндік береді. Жабдықты жеке мақсаттарға пайдалану, маршруттарды өзгерту, тоқтап қалу, компанияның көлігімен немқұрайлы қарау, жанар-жағармай материалдарын мақсатсыз пайдалану (жанармай бақылау датчиктерін қолдану) жағдайлары алдын алады.

ҚЫСҚАРТЫЛҒАН ЖӘНЕ ТҮСІНІКТЕМЕЛІК СӨЗДЕР

Қысқыртылған сөздер	Түсініктемесі
GPS	Global Positioning System — жаһандық позициялау жүйесі — аралықты, уақытты және орналасу нүктесін анықтауға арналған навигацияның жерсеріктік жүйесі.
GSM	Groupe Spécial Mobile – мобильді байланыс үшін Ғаламдық жүйені білдіреді.
GIS	Ғаламдық ақпараттық жүйе
AMQP	Жүйе хабарламалармен алмасу жолын анықтайтын ашық хаттама.
RabbitMQ	Бұл хабар алмасу платформасы.
API	Application Programming Interface – бағдарламалық бағдарламалау интерфейсі – жолдардың (сыныптар, процедуралар, функциялар, құрылымдар немесе тұрақтылар жиыны) сипаттамасы. Компьютер бағдарламасы басқа бағдарламамен әрекеттесе алады.
Java NDK	Native Development Kit— C, C++ және Jтілдерінде жазылған кодпен жұмыс істеуге көмектесетін құралдар жиынтығы.
Android Manifest.xml	қолданба туралы толық ақпарат береді. Әрбір Android қолданбасында бұл файлды көру керек, себебі ол Android қолданбасын жасау үшін қажет.
React JS	Пайдаланушы интерфейстерін құруға арналған декларативті, тиімді және икемді JavaScript кітапханасы.
Java	Интернеттегі шолуды қызықты әрі ыңғайлы ететін қолданбаларды әзірлеу үшін қолданылатын технология.
Firebase	Пайдаланушыларға сақталған ақпаратты сақтауға және алуға мүмкіндік беретін бұлтқа негізделген дерекқор

ҚОЛДАНЫЛҒАН ӘДЕБИЕТТЕР

1. Taxi fuel consumption and emissions estimation model based on the reconstruction of driving trajectory / Jiancheng Weng, Quan Liang, Zhihong Chen // Sagepub Journals – 2017. – С. 8-11
2. Efficient Shortest Path Query Processing in Dynamic Road Networks/ Mengxuan Zhang // Control Theory and Control Engineering. – 2021. – С. 43-68.
3. Novel online routing algorithms for smart people-parcel taxi sharing services / Son Nguyen Van, Anton Dries, Behrouz Babaki // ETRI Journal Wiley. – 2022. – С. 1-12.
4. Taxi Operational Status Real Time Monitoring System based on seat sensing / Zhongqing Huang // International Conference on Intelligent Systems Research and Mechatronics Engineering (ISRME 2015), 2015. – С. 1-4.
5. Online monitoring of local taxi travel momentum and congestion effects using projections of taxi GPS-based vector field / Joseph Y.J Chow // Journal of Geographical Systems – 2017. – С. 6-18
6. Taxi Dispatch with Real-Time Sensing Data in Metropolitan Areas — a Receding Horizon Control Approach / John A. Stankovic, George J. Pappas, Sirajum Munir // Miscellaneous – 2015. – С. 4-10
7. Optimization Model of Taxi Fleet Size Based on GPS Tracking Data / Zhenzhou Yuan, Dongye Sun // MDPI Articles – 2019. – С. 10-19
8. A Framework of Vehicular Security and Demand Service Prediction Based on Data Analysis Integrated with Blockchain Approach / Zeinab Shahbazi // MDPI Articles – 2021. – С. 5-15
9. Building maintainable web applications using React / David Johansson // Linköping University | Department of Computer and Information Science – 2020. – С. 41-75
10. Android Application Development / Rick Rogers, John Lombardo, Zigurd Mednieks, and Blake Meike // E-books from O'Reilly Media– 2021. – С. 5-15
11. React in Action / Mark Tielens Thomas // ISBN 9781617293856 – 2018. – С. 100-168
12. Design and Implementation of Taxi Management System / Yueming Peng // WHIOCE Original Research Article – 2018. – С. 5-10
13. Optimizing Efficiency of Taxi Systems: Scaling-up and Handling Arbitrary Constraints / Jiarui Gan // MDPI Articles – 2019. – С. 1-9
14. Methodology of introducing fleet management system / Kristijan Rogic, Branislav Sutic, Goran Kolaric // Intelligent Transport System Journal – 2008.– С. 1-6
15. Design and implementation of taxi dispatching and monitoring system based on GPS / https://www.roadragon.com/news/design-and-implementation-of-taxi-dispatching-and-monitoring-system-based-on-gps // Sep / 4 / 2020
16. Android Programming: The Big Nerd Ranch Guide (4th Edition) / Kristin Marsicano, Brian Gardner, Bill Phillips, Chris Stewart // Big Nerd Ranch – 2020. – С. 50-84
17. The Busy Coder's Guide to Advanced Android Development 2nd edition / Mr. Mark L Murphy // CommonWare Edition – 2018. – С 75-106

18. Mastering Firebase for Android Development: Build real-time, scalable, and cloud-enabled Android apps with Firebase / Ashok Kumar S // Packt Publishing – 2018. – С 84-251

19. Dynamic Taxi-sharing Service Using Intelligent Transportation System Technologies / Chi-Chung Tao // Dept. of Transportation Management, Tamkang University – 2007

20. Android Design Patterns: Interaction Design Solutions for Developers 1st Edition / Greg Nudelman // Wiley; 1st edition – 2015. – С 74-98

21. Водители такси, выполняющие работу через онлайн-платформы: каковы правовые последствия «уберизации» труда? / Н. Л. Лютов, И. В. Войтковская // Трудовые отношения и социальное обеспечение – 2020

22. Система идентификации водителя транспортного средства на основе биометрических данных / Е. С. Козин , А. В. Базанов // Интеллект. Инновации. Инвестиции – 2020

23. Қазақстан Республикасында қоғамдық көлікті диспетчерлендіру туралы / DAMU RG Зерттеу тобы ЖШС // Алматы қ. Қазан 2018.

24. Жолаушыларды тасымалдау технологиясы және оны ұйымдастыру / Атькен Ержанов // «Автомобиль көлігі» профилі бойынша оқу-әдістемелік бірлестігі – 2020

25. Разработка на JavaScript. Построение кроссплатформенных приложений с помощью GraphQL, React, React Native и Electron / Скотт Адам Д. //O’Relly – 2021

26. Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker / Frank Zammetti // Apress Publishing – 2020

27. Taxi as a Part of Public Transport / Jorgen Aarhaug // GIZ.SUTP – 2016

28. Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design 3rd Edition / Hernandez Michael J // Addison-Wesley Professional – 2021

29. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems 1st Edition / Martin Kleppmann // O'Reilly Media – 2017

30. Database Internals: A Deep Dive into How Distributed Data Systems Work 1st Edition / Alex Petrov // O'Reilly Media - 2017

ҚОСЫМША А

Такси қызметіне арналған көлік құралдарын бақылау жүйесінің бағдарламалық жасақтама листинг коды

```
import React, { useState } from 'react';
import axios from 'axios';
import { Formik } from 'formik';
import {
  Breadcrumb, Button, Card, Form
} from 'react-bootstrap';
import { Link, Navigate } from 'react-router-dom';

function SignUp (props) {
  const [isSubmitted, setSubmitted] = useState(false);

  const onSubmit = async (values, actions) => {
    const url = `${process.env.REACT_APP_BASE_URL}/api/sign_up/`;
    const formData = new FormData();
    formData.append('username', values.username);
    formData.append('first_name', values.firstName);
    formData.append('last_name', values.lastName);
    formData.append('password1', values.password);
    formData.append('password2', values.password);
    formData.append('group', values.group);
    formData.append('photo', values.photo);
    try {
      await axios.post(url, formData);
      setSubmitted(true);
    } catch (response) {
      const data = response.response.data;
      for (const value in data) {
        actions.setFieldError(value, data[value].join(' '));
      }
    }
  };

  if (props.isLoggedIn) {
    return <Navigate to='/' />;
  }

  if (isSubmitted) {
    return <Navigate to='/log-in' />;
  }
```

```
 return (
   <>
     <Breadcrumb>
       <Breadcrumb.Item href='/#/'>Home</Breadcrumb.Item>
       <Breadcrumb.Item active>Sign up</Breadcrumb.Item>
     </Breadcrumb>
     <Card className='mb-3'>
       <Card.Header>Sign up</Card.Header>
       <Card.Body>
         <Formik
           initialValues={{
             username: '',
             firstName: '',
             lastName: '',
             password: '',
             group: 'rider',
             photo: []
           }}
           onSubmit={onSubmit}
         >
           {({
             errors,
             handleChange,
             handleSubmit,
             isSubmitting,
             setFieldValue,
             values
           }) => (
             <Form noValidate onSubmit={handleSubmit}>
               <Form.Group className='mb-3' controlId='username'>
                 <Form.Label>Username:</Form.Label>
                 <Form.Control
                   className={'username' in errors ? 'is-invalid' : ''}
                   name='username'
                   onChange={handleChange}
                   required
                   values={values.username}
                 />
                 {
                   'username' in errors && (
                     <Form.Control.Feedback
type='invalid'>{errors.username}</Form.Control.Feedback>
                   )
                 }
```

```
            </Form.Group>
            <Form.Group className='mb-3' controlId='firstName'>
              <Form.Label>First name:</Form.Label>
              <Form.Control
                className={'firstName' in errors ? 'is-invalid' : ''}
                name='firstName'
                onChange={handleChange}
                required
                values={values.firstName}
              />
              {
                'firstName' in errors && (
                  <Form.Control.Feedback
type='invalid'>{errors.firstName}</Form.Control.Feedback>
                )
              }
            </Form.Group>
            <Form.Group className='mb-3' controlId='lastName'>
              <Form.Label>Last name:</Form.Label>
              <Form.Control
                className={'lastName' in errors ? 'is-invalid' : ''}
                name='lastName'
                onChange={handleChange}
                required
                values={values.lastName}
              />
              {
                'lastName' in errors && (
                  <Form.Control.Feedback
type='invalid'>{errors.lastName}</Form.Control.Feedback>
                )
              }
            </Form.Group>
            <Form.Group className='mb-3' controlId='password'>
              <Form.Label>Password:</Form.Label>
              <Form.Control
                className={'password1' in errors ? 'is-invalid' : ''}
                name='password'
                onChange={handleChange}
                required
                type='password'
                value={values.password}
              />
              {
                'password1' in errors && (
```

```
                <Form.Control.Feedback
type='invalid'>{errors.password1}</Form.Control.Feedback>
              )
             }
            </Form.Group>
            <Form.Group className='mb-3' controlId='group'>
             <Form.Label>Group:</Form.Label>
             <Form.Select
              className={'group' in errors ? 'is-invalid' : ''}
              name='group'
              onChange={handleChange}
              required
              value={values.group}
             >
              <option value='rider'>Rider</option>
              <option value='driver'>Driver</option>
             </Form.Select>
             {
              'group' in errors && (
               <Form.Control.Feedback
type='invalid'>{errors.group}</Form.Control.Feedback>
              )
             }
            </Form.Group>
            <Form.Group className='mb-3' controlId='photo'>
             <Form.Label>Photo:</Form.Label>
             <Form.Control
              className={'photo' in errors ? 'is-invalid' : ''}
              name='photo'
              onChange={event => {
               setFieldValue('photo', event.currentTarget.files[0]);
              }}
              required
              type='file'
             />
             {
              'photo' in errors && (
               <Form.Control.Feedback
type='invalid'>{errors.photo}</Form.Control.Feedback>
              )
             }
            </Form.Group>
            <div className='d-grid mb-3'>
             <Button
              disabled={isSubmitting}
```

```
                type='submit'
                variant='primary'
              >Sign up
              </Button>
            </div>
          </Form>
        )}
      </Formik>
      <Card.Text className='text-center'>
        Already have an account? <Link to='/log-in'>Log in!</Link>
      </Card.Text>
    </Card.Body>
  </Card>
 </>
 );
}

export default SignUp;
```

```
import React, { useState } from 'react';
import {
  DirectionsRenderer,
  DirectionsService,
  GoogleMap,
  LoadScript,
  Marker
} from '@react-google-maps/api';

function Map ({ dropOffAddress, lat, lng, pickUpAddress, zoom }) {
  const [response, setResponse] = useState(null);

  const hasTwoAddresses = (
    pickUpAddress !== '' &&
    dropOffAddress !== ''
  );

  const directionsCallback = (response) => {
    if (response !== null && response.status === 'OK') {
      setResponse(response);
    }
  };
```

```
return (
  <LoadScript
    googleMapsApiKey={process.env.REACT_APP_GOOGLE_MAPS_KEY}
  >
    <GoogleMap
      center={{
        lat,
        lng
      }}
      mapContainerStyle={{
        width: '100%',
        height: '300px',
        marginBottom: '10px'
      }}
      zoom={zoom}
    >
      {
        hasTwoAddresses && (
          <DirectionsService
            options={{
              origin: pickUpAddress,
              destination: dropOffAddress,
              travelMode: 'DRIVING'
            }}
            callback={directionsCallback}
          />
        )
      }
      {
        hasTwoAddresses && response !== null && (
          <DirectionsRenderer
            options={{
              directions: response
            }}
          />
        )
      }
      {
        !hasTwoAddresses && (
          <Marker label='A' position={{ lat, lng }} />
        )
      }
    </GoogleMap>
  </LoadScript>
);
```

```
}

export default Map;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.FragmentActivity;

import android.Manifest;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Location;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.TextView;

import com.firebase.geofire.GeoFire;
import com.firebase.geofire.GeoLocation;
import com.firebase.geofire.GeoQuery;
import com.firebase.geofire.GeoQueryEventListener;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptor;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
```

```
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.squareup.picasso.Picasso;

import java.util.HashMap;
import java.util.List;

import de.hdodenhof.circleimageview.CircleImageView;

public class CustomersMapActivity extends FragmentActivity implements
OnMapReadyCallback,
        GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
        com.google.android.gms.location.LocationListener

{

    private GoogleMap mMap;
    GoogleApiClient googleApiClient;
    Location lasLocation;
    LocationRequest locationRequest;
    Marker driverMarker, PickUpMarker;
    GeoQuery geoQuery;

    private Button  settingsButton;
    private  Button callTaxiButton;
    private String customerID;
    private  LatLng CustomerPosition;
    private FirebaseAuth mAuth;
    private FirebaseUser currentUser;
    private DatabaseReference CustomerDatabaseRef;
    private  DatabaseReference DriversAvailableRef;
    private  int  radius = 1;
    private  Boolean driverFound = false,requestType = false;
    private  String  driverFoundID;
    private DatabaseReference DriversRef;
    private DatabaseReference DriversLocationRef;

    private ValueEventListener DriverLocationRefListener;
    private ImageView callDriver;

    private TextView txtName, txtPhone, txtCarName;
    private CircleImageView driverPhoto;
    private RelativeLayout relativeLayout;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customers_map);


        settingsButton = (Button)findViewById(R.id.customer_settings_button);
        callTaxiButton = (Button)findViewById(R.id.customer_order_button);
        callDriver= findViewById(R.id.call_to_driver);

        mAuth = FirebaseAuth.getInstance();

        currentUser = mAuth.getCurrentUser();
        customerID = FirebaseAuth.getInstance().getCurrentUser().getUid();
        CustomerDatabaseRef =
FirebaseDatabase.getInstance().getReference().child("Customers Requests");
        DriversAvailableRef =
FirebaseDatabase.getInstance().getReference().child("Driver Available");
        DriversLocationRef =
FirebaseDatabase.getInstance().getReference().child("Driver Working");

        txtName = (TextView)findViewById(R.id.driver_name);
        txtPhone = (TextView)findViewById(R.id.driver_phone_number);
        txtCarName = (TextView)findViewById(R.id.driver_car);
        driverPhoto = (CircleImageView)findViewById(R.id.driver_photo);
        relativeLayout = findViewById(R.id.rel1);

        relativeLayout.setVisibility(View.INVISIBLE);



        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);

        settingsButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(CustomersMapActivity.this,
SettingsActivity.class);
                intent.putExtra("type","Customers");
                startActivity((intent));
```

```
        }
    });


    callTaxiButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v)
        {
            if (requestType) {
                requestType = false;

                if(driverFound != null)
                {
                    DriversRef = FirebaseDatabase.getInstance().getReference()

.child("Users").child("Drivers").child(driverFoundID).child("CustomerRideID");

                    DriversRef.removeValue();
                    driverFoundID = null;

                }

                driverFound =false;
                radius = 1;
                GeoFire geoFire = new GeoFire(CustomerDatabaseRef);
                geoFire.removeLocation(customerID);

                if(PickUpMarker !=null)
                {
                    PickUpMarker.remove();
                }
                if(driverMarker !=null)
                {
                    driverMarker.remove();
                }


                callTaxiButton.setText("Вызвать такси");

            }
            else {
                requestType = true;
```

```
                GeoFire geoFire = new GeoFire(CustomerDatabaseRef);
                geoFire.setLocation(customerID, new
GeoLocation(lasLocation.getLatitude(), lasLocation.getLongitude()));

                CustomerPosition = new LatLng(lasLocation.getLatitude(),
lasLocation.getLongitude());
                PickUpMarker = mMap.addMarker(new
MarkerOptions().position(CustomerPosition).title("Я
здесь").icon(BitmapDescriptorFactory.fromResource(R.drawable.humancst)));

                callTaxiButton.setText("Поиск водителя...");
                getNearbyDrivers();
            }
        }
    });
}



@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    buildGoogleApiClient();
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {


        return;
    }
    mMap.setMyLocationEnabled(true);


}

@Override
public void onConnected(@Nullable Bundle bundle) {
    locationRequest = new LocationRequest();
    locationRequest.setInterval(100000);
    locationRequest.setFastestInterval(1000);
    locationRequest.setPriority(locationRequest.PRIORITY_HIGH_ACCURACY);
```

```
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            return;
        }
        LocationServices.FusedLocationApi.requestLocationUpdates(googleApiClient,
locationRequest, this);

    }

    @Override
    public void onConnectionSuspended(int i) {

    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

    }

    @Override
    public void onLocationChanged(Location location) {
        lasLocation = location;

        LatLng latLng  = new LatLng(location.getLatitude(), location.getLongitude());
        mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mMap.animateCamera(CameraUpdateFactory.zoomTo(15));

    }
    protected synchronized void buildGoogleApiClient()
    {
        googleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();

        googleApiClient.connect();
    }

    @Override
    protected void onStop() {
```

```
        super.onStop();

    }

    private void LogoutCustomer() {
        Intent welcomeIntent =  new
Intent(CustomersMapActivity.this,WelcomeActivity1.class);
        startActivity(welcomeIntent);
        finish();
    }

    private void getNearbyDrivers()
    {
        GeoFire geoFire = new GeoFire(DriversAvailableRef);
        GeoQuery geoQuery = geoFire.queryAtLocation(new
GeoLocation(CustomerPosition.latitude,CustomerPosition.longitude),radius);
        geoQuery.removeAllListeners();

        geoQuery.addGeoQueryEventListener(new GeoQueryEventListener() {
            @Override
            public void onKeyEntered(String key, GeoLocation location) {
                if(!driverFound && requestType)
                {
                    driverFound = true;
                    final String driverFoundID = key;

                    DriversRef =
FirebaseDatabase.getInstance().getReference().child("Users").child("Drivers").child(
driverFoundID);
                    HashMap driverMap = new HashMap();
                    driverMap.put("CustomerRideID",customerID);
                    DriversRef.updateChildren(driverMap);


                    GetDriverLocation();
                }
            }

            @Override
            public void onKeyExited(String key) {

            }

            @Override
            public void onKeyMoved(String key, GeoLocation location) {
```

```
        }

        @Override
        public void onGeoQueryReady() {
           if(!driverFound)
           {
              radius = radius + 1;
              getNearbyDrivers();
           }

        }

        @Override
        public void onGeoQueryError(DatabaseError error) {

        }
     });
  }

   private void GetDriverLocation() {
      DriverLocationRefListener =
DriversLocationRef.child(driverFoundID).child("l").
           addValueEventListener(new ValueEventListener() {
              @Override
              public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                 if(dataSnapshot.exists() && requestType)
                 {
                    List<Object> driverLocationMap = (List<Object>)
dataSnapshot.getValue();
                    double locationLat = 0;
                    double locationLng = 0;

                    callTaxiButton.setText("Водитель найден");




                    if(driverLocationMap.get(0) !=null)
                    {
                       locationLat =
Double.parseDouble(driverLocationMap.get(0).toString());
                    }
                    if(driverLocationMap.get(0) !=null)
                    {
```

```
                    locationLng =
Double.parseDouble(driverLocationMap.get(1).toString());
                }
                LatLng DriverLatLng = new LatLng(locationLat,locationLng);

                if(driverMarker !=null)
                {
                    driverMarker.remove();
                }
                Location location1 = new Location("");
                location1.setLatitude((CustomerPosition.latitude));
                location1.setLongitude((CustomerPosition.longitude));

                Location location2 = new Location("");
                location2.setLatitude((DriverLatLng.latitude));
                location2.setLongitude((DriverLatLng.longitude));

                float Distance = location1.distanceTo(location2);
                if(Distance>100)
                {
                    callTaxiButton.setText("Ваше такси подъезжает");
                }
                else {
                    callTaxiButton.setText("Расстояние до такси" +
String.valueOf(Distance));
                }


                driverMarker = mMap.addMarker(new
MarkerOptions().position(DriverLatLng).title("Ваш такси
тут").icon(BitmapDescriptorFactory.fromResource(R.drawable.taxi)));
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }
        });
    }
    private void  getDriverInformation()
    {
        relativeLayout.setVisibility(View.VISIBLE);
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference()
                .child("Users").child("Drivers").child(driverFoundID);
```

```
        reference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if(dataSnapshot.exists() && dataSnapshot.getChildrenCount()>0)
                {
                    String name = dataSnapshot.child("name").getValue().toString();
                    final String phone = dataSnapshot.child("phone").getValue().toString();
                    String carname = dataSnapshot.child("carname").getValue().toString();

                    txtName.setText(name);
                    txtPhone.setText(phone);
                    txtCarName.setText(name);

                    callDriver.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            int permissionCheck =
ContextCompat.checkSelfPermission(CustomersMapActivity.this,Manifest.permissio
n.CALL_PHONE);

                            if(permissionCheck
!=PackageManager.PERMISSION_GRANTED){
                                ActivityCompat.requestPermissions(
                                        CustomersMapActivity.this,new String[]{
                                                Manifest.permission.CALL_PHONE
                                        }, 123
                                );
                            }
                            else{
                                Intent intent = new Intent(Intent.ACTION_CALL,
Uri.parse("tel:"+ phone));
                                startActivity(intent);

                            }
                        }
                    });


                    if (dataSnapshot.hasChild("image")) {
                        String image = dataSnapshot.child("image").getValue().toString();
                        Picasso.get().load(image).into(driverPhoto);
                    }
                }
```

```
        }


        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
      });
   }

}
```

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/SandyBrown"
   tools:context=".CustomersMapActivity">

   <fragment
      android:id="@+id/map"
      android:name="com.google.android.gms.maps.SupportMapFragment"
      android:layout_marginTop="50dp"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      />

   <RelativeLayout
      android:id="@+id/rel1"
      android:background="#0f1e7e"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_margin="10dp"
      android:layout_above="@+id/customer_order_button">

      <de.hdodenhof.circleimageview.CircleImageView
         android:id="@+id/driver_photo"
         android:layout_width="80dp"
         android:layout_height="80dp"
         android:src="@drawable/qwertyuio"
         app:tint="@color/white"
         android:layout_marginTop="15dp"
```

```
    />


<TextView
    android:id="@+id/driver_name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Имя водителя"
    android:layout_toEndOf="@+id/driver_photo"
    android:textColor="@color/white"
    android:layout_marginTop="10dp"
    android:layout_marginLeft="15dp"
    android:textSize="16sp"
    android:layout_toRightOf="@+id/driver_photo" />

<TextView
    android:id="@+id/driver_phone_number"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toEndOf="@+id/driver_photo"
    android:text="Номер телефона"
    android:layout_below="@+id/driver_name"
    android:textColor="@color/white"
    android:layout_marginTop="10dp"
    android:layout_marginLeft="15dp"
    android:textSize="16sp"
    android:layout_toRightOf="@+id/driver_name" />
<TextView
    android:id="@+id/driver_car"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toEndOf="@+id/driver_photo"
    android:text="Марка автомобиля"
    android:layout_below="@+id/driver_phone_number"
    android:textColor="@color/white"
    android:layout_marginTop="10dp"
    android:layout_marginLeft="15dp"
    android:textSize="16sp"
    android:layout_toRightOf="@+id/driver_photo" />
<ImageView
    android:id="@+id/call_to_driver"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
```

```
            android:src="@drawable/telephone"
            android:layout_centerVertical="true"
            android:layout_marginRight="10dp"
            app:tint="@android:color/white" />



    </RelativeLayout>

    <Button
        android:id="@+id/customer_settings_button"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:text="Профиль"
        android:background="#085bfd"
        android:textColor="@color/white"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true" />



    <Button
        android:id="@+id/customer_order_button"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:background="#085bfd"
        android:textColor="@color/white"
        android:text="Вызвать такси"
        android:layout_alignParentBottom="true"/>

</RelativeLayout>
```