|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ** |  |

Никола Драгомировић

**СИСТЕМ ПАМЕТНОГ НАДЗОРА БАЗИРАН НА ТЕХНОЛОГИЈИ ПРЕПОЗНАВАЊА ЛИЦА**

ДИПЛОМСКИ РАД

* Основне Академске Студије -

Нови Сад, 2024

|  |  |  |  |
| --- | --- | --- | --- |
| Редни број, **РБР**: | |  | |
| Идентификациони број, **ИБР**: | |  | |
| Тип документације, **ТД**: | | Монографска документација | |
| Тип записа, **ТЗ**: | | Текстуални штампани материјал | |
| Врста рада, **ВР**: | | Дипломски рад | |
| Аутор, **АУ**: | | Никола Драгомировић | |
| Ментор, **МН**: | | Др Милан Лукић, доцент | |
| Наслов рада, **НР**: | | Систем паметног надзора базиран на технологији препознавања лица | |
| Језик публикације, **ЈП**: | | Српски / латиница | |
| Језик извода, **ЈИ**: | | Српски | |
| Земља публиковања, **ЗП**: | | Република Србија | |
| Уже географско подручје, **УГП**: | | Војводина | |
| Година, **ГО**: | | 2024 | |
| Издавач, **ИЗ**: | | Ауторски репринт | |
| Место и адреса, **МА**: | | Нови Сад; трг Доситеја Обрадовића 6 | |
| Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога) | | 6/28/0/0/4/0/0 | |
| Научна област, **НО**: | | Електротехника и рачунарство | |
| Научна дисциплина, **НД**: | | Рачунарска техника | |
| Предметна одредница/Кqучне речи, **ПО**: | | Уграђени рачунарски системи | |
| **УДК** | |  | |
| Чува се, **ЧУ**: | | У библиотеци Факултета техничких наука, Нови Сад | |
| Важна напомена, **ВН**: | |  | |
| Извод, **ИЗ**: | | У овом раду представљенa је имплементација сигурносног система са функцијом препознавања лица и веб апликацијом. Систем се састоји од корисничког интерфејса изграђеног користећи React.js и TailwindCSS, интеграције са Telegram платформом у циљу лаког достављања обавештења, ESP32 микроконтролера са уграђеном камером и позадинскe логикe која опслужује све делове ситема изграђеном помоћу Python програмског језика и Flask оквира. | |
| Датум прихватања теме, **ДП**: | |  | |
| Датум одбране, **ДО**: | |  | |
| Чланови комисије, **КО**: | Председник: | Др Иван Мезеи |  |
|  | Члан: | Др Предраг Теодоровић | Потпис ментора |
|  | Члан, ментор: | Др Милан Лукић |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Accession number, **ANO**: | |  | |
| Identification number, **INO**: | |  | |
| Document type, **DT**: | | Monographic publication | |
| Type of record, **TR**: | | Textual printed material | |
| Contents code, **CC**: | | Batchelor Thesis | |
| Author, **AU**: | | Nikola Dragomirović | |
| Mentor, **MN**: | | Milan Lukić, Ph. D | |
| Title, **TI**: | | Smart surveillance system based on facial recognition technology | |
| Language of text, **LT**: | | Serbian | |
| Language of abstract, **LA**: | | Serbian | |
| Country of publication, **CP**: | | Republic of Serbia | |
| Locality of publication, **LP**: | | Vojvodina | |
| Publication year, **PY**: | | 2024 | |
| Publisher, **PB**: | | Author’s reprint | |
| Publication place, **PP**: | | Novi Sad, Dositeja Obradovica sq. 6 | |
| Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes) | | 6/28/0/0/4/0/0 | |
| Scientific field, **SF**: | | Electrical Engineering | |
| Scientific discipline, **SD**: | | Computer Engineering, Engineering of Computer Based Systems | |
| Subject/Key words, **S**/**KW**: | | Embedded Computer Systems | |
| **UC** | |  | |
| Holding data, **HD**: | | The Library of Faculty of Technical Sciences, Novi Sad, Serbia | |
| Note, **N**: | |  | |
| Abstract, **AB**: | | This project presents the implementation of a security system with face recognition capabilities and a web application. The system consists of a user interface built using React.js and TailwindCSS, a Telegram platform integration with the goal of easy notification delivery, ESP32 microcontroller with a built-in camera module and a backend that serves all other parts of the system built using Python with the Flask framework. | |
| Acscordcepted by the Scientific Board on, **ASB**: | |  | |
| Defended on, **DE**: | |  | |
| Defended Board, **DB**: | President: | Dr Ivan Mezei |  |
|  | Member: | Dr Predrag Teodorović | Menthor's sign |
|  | Member, Mentor: | Dr Milan Lukić |  |

Изјава о академској честитости

Студент: Никола Драгомировић

Број индекса: RA202/2020

Студент: основних студија

Аутор рада под називом: СИСТЕМ ПАМЕТНОГ НАДЗОРА БАЗИРАН НА ТЕХНОЛОГИЈИ ПРЕПОЗНАВАЊА ЛИЦА

Потписивањем изјављујем:

* да је рад искључиво резултат мог сопственог истраживачког рада;
* да сам рад и мишљења других аутора које сам користиo у овом раду назначиo или цитираo и да су наведени у списку литературе који је саставни део овог рада;
* да сам добиo све дозволе за коришћење ауторског дела које се у потпуности уносе у предати рад и да сам то јасно навеo;
* да сам свестан да је плагијат коришћење туђих радова у било ком облику (као цитата, парафраза, слика, табела, дијаграма, дизајна, планова, фотографија, филма, музике, формула, веб сајтова, компјутерских програма и сл.) без навођења аутора или представљање туђих ауторских дела као мојих, кажњиво по закону (Закон о ауторском и сродним правима, Службени гласник Републике Србије, бр. 104/2009, 99/2011, 119/2012), као и других закона и одговарајућих аката Универзитета у Новом Саду;
* да сам свестан да плагијат укључује и представљање, употребу и дистрибуирање рада предавача или других студената као сопствених;
* да сам свестан последица које код доказаног плагијата могу проузроковати на предати рад и мој статус;
* да је електронска верзија рада идентична штампаном примерку и да пристајем на његово објављивање под условима прописаним актима Универзитета.

Нови Сад, 15.10.2024. Потпис студента: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

САДРЖАЈ

[1. УВОД 7](#_Toc177826753)

[2. ТЕОРИЈСКЕ ОСНОВЕ 8](#_Toc177826754)

[2.1 Мрежни протоколи 8](#_Toc177826755)

[2.1.1 TCP/IP модел 8](#_Toc177826756)

[2.1.2 Transmission Control Protocol 8](#_Toc177826757)

[2.1.3 HTTP 9](#_Toc177826758)

[2.1.4 Рута 10](#_Toc177826759)

[2.1.5 WebSocket 11](#_Toc177826760)

[2.2 Технологије 11](#_Toc177826761)

[2.2.1 ESP32 11](#_Toc177826762)

[2.2.2 Python Flask 11](#_Toc177826763)

[2.2.3 Python face\_recognition 12](#_Toc177826764)

[2.2.4 MinIO 12](#_Toc177826765)

[2.2.5 React.js 13](#_Toc177826766)

[2.2.6 TailwindCSS 13](#_Toc177826767)

[2.2.7 MongoDB 13](#_Toc177826768)

[2.3 ПРОГРАМСКИ АЛАТИ 13](#_Toc177826769)

[2.3.1 GIT 13](#_Toc177826770)

[2.3.2 GitHub 14](#_Toc177826771)

[3. КОНЦЕПТ РЕШЕЊА 15](#_Toc177826772)

[3.1 Микроконтролерски уређај са камером 15](#_Toc177826773)

[3.2 Cloud 15](#_Toc177826774)

[3.2.1 Backend 16](#_Toc177826775)

[3.2.2 База Података 18](#_Toc177826776)

[3.2.3 Object Store 18](#_Toc177826777)

[3.3 Frontend 18](#_Toc177826778)

[3.3.1 Екран за пријаву и регистрацију 19](#_Toc177826779)

[3.3.2 Екран за преглед и управљање сликама 19](#_Toc177826780)

[3.3.3 Екран за предлед и управљање подешавањима 20](#_Toc177826781)

[3.4 Telegram 22](#_Toc177826782)

[4. ИМПЛЕМЕНТАЦИЈА 23](#_Toc177826783)

[4.1 Микроконтролерски уређај са камером 23](#_Toc177826784)

[4.2 Backend 23](#_Toc177826785)

[4.2.1 upload\_image() 23](#_Toc177826786)

[4.2.2 send\_telegram\_notification() 24](#_Toc177826787)

[4.2.3 face\_recognition.face\_encodings() 24](#_Toc177826788)

[4.2.4 face\_recognition.compare\_faces() 25](#_Toc177826789)

[4.3 Telegram 25](#_Toc177826790)

[4.4 Frontend 25](#_Toc177826791)

[4.4.1 Login 25](#_Toc177826792)

[4.4.2 Settings 26](#_Toc177826793)

[4.4.3 Feed 26](#_Toc177826794)

[5. Закључак 27](#_Toc177826795)

[6. Литература 28](#_Toc177826796)

**СПИСАК СЛИКА**

[Слика 1.1 Успешно слање TCP пакета Слика 1.2 Ретрансмисија TCP пакета 9](#_Toc177299564)

[Слика 2.1 Екран за пријављивање 19](#_Toc177299565)

[Слика 3.1 Објава на главном екрану Слика 3.2 Чување лица 20](#_Toc177299566)

[Слика 4.1 Екран подешавања Слика 4.2 Приказ грешке 21](#_Toc177299567)

**СКРАЋЕНИЦЕ**

**IoT** - *Internet of things, Интернет Ствари*

**TCP** - *Transmission Control Protocol, Протокол Контроле Преноса*

**IP** - Internet Protokol, Интернет Протокол

**HTTP** - *HyperText Transfer Protocol, Језик за означавање веб страница*

**CSS** - *Cascading Style Sheets, Каскадни Стилски Образци*

**URL** - Uniform Resource Locator, Јединствени Локатор Ресурса

**URI** - Uniform Resource Identifier, Jединствени Идентификатор Ресурса

**URN** - Uniform Resource Name, Јединствено Име Ресурса

**Wi - Fi** - *Wireless-Fidelity, Бежична Мрежа*

**WSGI** - *Web Server Gateway Interface, Веб Сервер Пролазни Интерфејс*

**UI** - User Interface, Кориснички Интерфејс

**DOM** - *Document Object Model, Документ Објектни Модел*

**VCS** - *Version Control System, Систем за контролу верзија*

**GUI** - Graphical User Interface, Графичко Корисничко Окружење

**SQL** - Structured Query Language, Структуирани Упитни Језик

# УВОД

Задатак рада био је израда сигурносног IoT система са функцијом препознавања лица. Систем се састоји од ESP32 микроконтролера са пратећом камером, веб интерфејса за управљање и подешавање камера, сачуваних лица и телеграм интеграције и саме телеграм интеграције која доставља обавештења о лицима које камера детектује. Идеја овог система је да пружи корисницима могућност праћења активности на њиховим улазним вратима, дворишту итд. без потребе да константно проверавају ток камере. Корисници могу да сачувају познато лице на веб интерфејсу, затим сваки следећи пут када се та особа детектује на било којој од више камера које се могу повезати са једним профилом, биће препозната, и обавештења ће садржати њихово постављено име.

Прво ће бити установљене теоријске основе које су потребне ради разумевања рада, затим ће се говорити о идеји реализације и након тога о самој имплементацији и резултатима које је систем постигао.

# ТЕОРИЈСКЕ ОСНОВЕ

У оквиру овог поглавља говориће се о мрежним протоколима, технологијама и алатима коришћеним приликом израде система. Разумевање тих појмова биће кључно ради даљег разумевања самог рада и система.

## Мрежни протоколи

Мрежни протоколи су скуп правила и процедура које уређаји користе за комуникацију у мрежи. Они одређују начине на које се подаци преносе између уређаја, како се откривају грешке приликом истог, како и у ком формату су ти подаци представљени. У овом раду говориће се о 2 мрежна протокола: TCP (Transmission Control Protocol) и HTTP (HyperText Transfer Protocol)

### TCP/IP модел

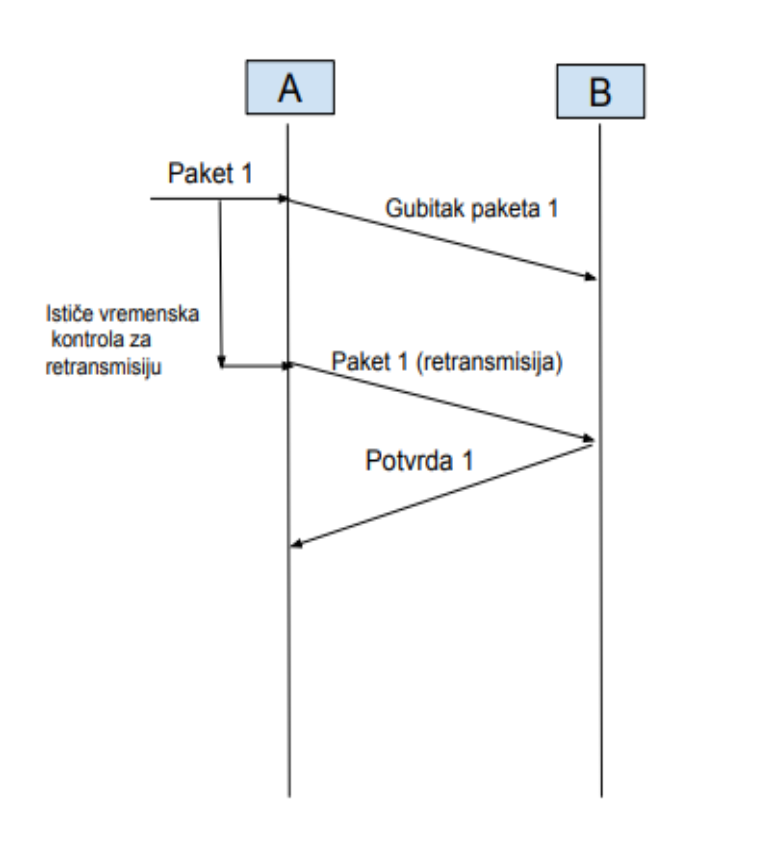
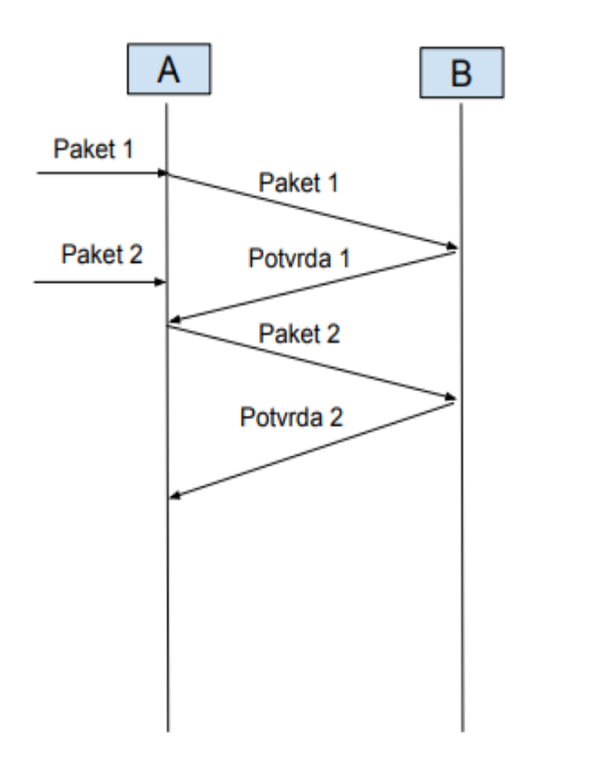
TCP/IP представља фамилију протокола за комуникацију различитих уређаја. Ти протоколи функционишу на различитим нивоима TCP/IP стека. Стек има 4 нивоа, и то су:

* Апликативни - обезбеђује спрегу између корисничких апликација са нижим слојевима. Спрега апликације и мреже.
* Транспортни - пакетизира информације које прима од апликативног нивоа и шаље их даље на ниже нивое, ради и у супротном смеру где пакет добијен од стране мрежног нивоа распакује и шаље информације апликативном. Такође је задужен и за контролу тока између два рачунара.
* Мрежни - задужен за усмеравање пакета кроз мрежу.
* Ниво канала - рукује физичким детаљима система, састоји се од нивоа везе података и физичког нивоа.

### Transmission Control Protocol

TCP [1] је протокол поузданог преноса. Он се налази у транспортном слоју TCP/IP стека и ради на врху IP-а који је део мрежног слоја. IP служи за испоруку датаграма без потребе да се успостави веза између изворишта и одредишта. Овај протокол је непоуздан јер не гарантује да ће пакети бити испоручени, а и ако буду, не гарантује да ће се налазити у редоследу у ком су послати. Измена редоследа проузрокована је различитим мрежним путањама којим пакети могу бити упућени ка одредишту.

Иако је IP непоуздан протокол, TCP је реализован тако да гарантује да ће пакети доспети до примаоца и то у редоследу у ком су послати. Да би то постигао прималац потврђује сваки пакет који прими, тако да пошиљалац зна да је пакет стигао на своје одредиште. Пошиљалац тек тада наставља са даљим слањем. На потврду пријема (енгл. Acknowledgement, ACK) пошиљалац чека одређени временски период, где у случају његовог истека, он поново шаље непотврђени пакет (енгл. Retransmission). На тај начин апликација може рачунати да ће мрежа обавити поуздану испоруку информације жељеном одредишту.



Слика 1.1 Успешно слање TCP пакета Слика 1.2 Ретрансмисија TCP пакета

### HTTP

HyperText Transport Protocol [2] служи за пренос HTML (енгл. HyperText Markup Language) страница од веб сервера до веб прегледача где се њихов садржај приказује корисницима. О HTML језику и HTML страницама ће се више говорити у наставку рада.

HTTP ради на највишем, апликационом, нивоу и заснива се на принципу захтев/одговор. Прегледач, са којим корисник интерагује, шаље захтеве на сервер који му враћа одређени одговор. Комуникација у оквиру овог протокола је двосмерна, сервер преноси странице ка прегледачу, док прегледач серверу може да достави различите форме (нпр. Форма за регистрацију корисника). Прегледач има могућност да баферује странице које корисник потражује и онда, када корисник поново потражи исте странице, их достави доста брже.

Неки од HTTP захтева:

* GET – прибављање веб странице
* POST – допуњавање веб странице
* DELETE – уклањање веб странице
* HEAD – прибављање заглавља веб странице

HTTP такође доставља и поруке о грешкама, а у зависности од кода грешке разликујемо следеће:

* 1xx – информациони
* 2xx – успешно извршена операција
* 3xx – преусмерење
* 4xx – грешка од стране клијента
* 5xx – грешка од стране сервера

### URI - Рута

URI [3] или рута, служи као јединствена адреса за ресурсе на интернету, омогућавајући клијентима да захтевају одређене податке или услуге од сервера. URI се дели на две категорије: URL-ове и URN-ове. URL-ови су најчешћи облик URI-ја и описују не само ресурс, већ и метод за његово преузимање. Састоје се од више компоненти, укључујући протокол који се користи за приступ ресурсу (као што је HTTP), адресу сервера и одређену локацију ресурса или услуге на том серверу. URL-ови пружају мапу за проналажење ресурса и услуга на веб платформи и њихово преузимање преко одговарајућег протокола. Скоро сви URI који се данас користе су URL-ови, што их сврстава под шири HTTP оквир, јер дефинишу начин на који се ресурсима приступа путем веб протокола.

### WebSocket

WebSocket [4] омогућује истовремену (енгл. Full-duplex) комуникацију у оба смера (енгл. Bidirectional) која се одвија преко HTTP-а кроз један пролаз (енлг. Socket). Овде више не морају да се шаљу захтеви већ се комуникација одвија константно докле год је корисник повезан са WebSocket-ом, такође неће бити потребе да се корисник преусмерава на друге странице већ сва комуникација и приказ података може да се одвија на истој страници. Приликом отварања странице која користи WebSocket, остварује се конекција са сервером слањем одређених пакета, а са изласком са странице конекција се прекида. Сервер страна у овој апликацији реализована је користећи Python и Flask о којима ће више бити речено у наставку рада. Поред њих постоје и друге могућности за реализацију сервера: Node.js, PHP, Ruby…

## Технологије

### ESP32

ESP32 [5] је нискобуџетна и нископотрошна серија микроконтролера на чипу са Wi-Fi и Bluetooth могућностима и високо интегрисаном структуром коју покреће двојезгарни Tensilica Xtensa LX6 микропроцесор. Широко се користи у IoT апликацијама, кућној аутоматизацији, носивим уређајима и уграђеним системима.

Главне карактеристике:

* Двојезгарни Xtensa LX6 микропроцесор
* Уграђен Wi-Fi (802.11 b/g/n)
* Више GPIO pinova, UART, SPI, I2C i PWM за разноврсно повезивање периферија
* Режим уштеде енергије, погодан за батеријско напајанје
* Компатибилност са развојним окружењима као што су ЕSP-IDF, MicroPython и Arduino, који је коришћен за имплементацију овог система

### Python Flask

Flask [6] се издваја од других оквира пружањем пуне креативне контроле, избегавајући уобичајен проблем „борбе са оквиром” при коришћењу нестандардних решења. За разлику од других оквира који су ограничени на одређене алате или методе, Flask је изузетно флексибилан, подржава и релационе и NoSQL базе података, па чак и опције које су ручно израђене. Омогућава креирање компоненти без ограничења. Ова слобода долази из Flask-овог дизајна, који се ослања на робусно језгро са основним функционалностима за веб апликације, остављајући простор за екстензије од стране програмера. Заснован је на WSGI стандарду.

Kључне карактеристике:

* Минималистички приступ без сувишних слојева и компоненти
* Подршка за URL рутирање и обрађивање HTTP захтева
* Лако проширивање путем додатних Python библиотека, за функционалности као што су аутентикација, рад са базама података, и препознавање лица.

### Python face\_recognition

Библиотека face\_recognition у Python-у је једноставан и моћан алат за препознавање и детекцију лица. Kористи моделе дубоког учења, за ефикасно енкодовање лица и поређење.

Кључне функције библиотеке укључују:

* Идентификацију и локацију лица на сликама.
* Енкодирање лица у векторе од 128 димензија, омогућавајући поређење и усклађивање са познатим лицима.
* Библиотека нуди једноставне функције као што су face\_locations(), face\_encodings() и compare\_faces()

Библиотека зависи од dlib алата, који укључује високо оптимизоване алгоритме машинског учења и обраде слике, чинећи је брзом и прецизном за примене у реалном времену.

### MinIO

MinIO је високо перформантан, дистрибуирани систем за складиштење објеката дизајниран за управљање неструктурираним подацима попут фотографија, видео снимака. Компатибилан је са Amazon S3 услугом за складиштење у облаку, што га чини популарним избором за локална решења за складиштење у облаку.

### React.js

React [7] је JavaScript оквир осмишљен да поједностави развој комплексних корисничких интерфејса који рукују динамичким подацима. Појавио се како би решио ограничења традиционалног Model-View-Controller приступа, који се ослањао на двосмерно везивање података и рендеровање шаблона. Објављен 2013. године, React је увео иновативне концепте који су изазвали претходно утврђене најбоље праксе у развоју веб апликација.

Кључни напредци које је React донео, попут виртуелног DOM-а, JSX-а и једносмерног протока података, променио је начин на који програмери граде скалабилне и одрживе апликације.

### TailwindCSS

TailwindCSS [8] је познат по високом нивоу прилагођавања, дајући програмерима слободу да стилизују веб странице без ослањања на унапред дизајниране компоненте. Ова флексибилност омогућава изградњу јединствених, потпуно прилагођених интерфејса, уз спречавање да измене на једној страници утичу на друге. Убрзава развој уклањањем потребе за традиционалним CSS стилским листовима, смањујући величину датотека и оптерећење. Фокусирајући се на HTML и елиминишући непотребни CSS, омогућава креирање лаганих али моћних веб интерфејса.

### MongoDB

MongoDB [9] је нерелациона (NoSQL) база података заснована на документима, дизајнирана за складиштење великих количина података на флексибилан и скалабилан начин. Уместо традиционалних табела и редова као у релационим базама, MongoDB чува податке у JSON сличним документима, омогућавајући лако руковање комплексним и полуструктурираним подацима. Овај модел омогућава брзу интеграцију и прилагођавање променама у структури података.

## ПРОГРАМСКИ АЛАТИ

### GIT

Git [10] је алат који припада групи система за контролу верзија. Оно што Гит чини једним од најбољих VCS алата и што га разликује од других алата који се користе за исто то јесте начин на који Гит посматра податке са којима ради. Док други VCS системи информације које чувају посматрају као скуп датотека и промена које су направљене у њима током времена, Git их посматра као као низ снимака. Са Гитом, сваки пут када се сачува стање пројекта, он чува референце на тај снимак, уколико није било никаквих промена Гит неће поново складиштити исту датотеку. Осим што нуди опцију чувања и праћења тока нашиx промена, Гит пружа и разне функционалности за манипулацију са истим.

Још једна од предности Гита јесте што ради локално. Рецимо да програмер жели да погледа шта је то радио прошле недеље, Гит не мора да приступа неком серверу ради добављања тих података, он једноставно прочита податке директно из базе података која је на уређају. Чак и уколико програмер није у могућности да користи интернет, и даље може да користи скоро све функционалности Гита, све то ће се одвијати локално.

Гит разликује три стања у којима се датотеке могу наћи:

* измењене (енгл. Modified) - датотеке којима су се десиле било какве промене
* припремљене (енгл. Staged) - оне датотеке које су спремне за слање у базу података, да би се припремиле датотеке за слање користи се команда git add filename или уколико је потребно да се све датотеке припреме git add –А
* потврђене (енгл. Commited) - стање датотека шаљу се у базу података где Гит том стању додељује хеш на основу ког после може опет да му се приступи. Подаци се потврђују користећи git commit. Корисно је и оставити неки коментар што се може урадити са git commit –m ‘komentar’. Да би се видела историја свих потврда може се користити команда git log. Док се датотеке налазе у било ком од ових стања, над њима се могу обављати разне акције. Са Гитом се управља из терминала (или неке од GUI апликација), где се и све команде уносе.

### GitHub

GitHub је платформа за хостовање кода и управљање верзијама заснована на Git систему. Омогућава програмерима да прате промене у коду, креирају гране и враћају се на претходна стања пројекта. Једна од главних предности GitHub-а је могућност да корисници "пушују" код на удаљени репозиторијум, чиме омогућавају сигурно чување и приступ пројекту са било које локације. Ово је посебно корисно за рад на удаљеним пројектима или за рад са више уређаја. GitHub је због својих могућности постао стандардни алат за управљање кодом у професионалном и личном развоју софтвера.

# КОНЦЕПТ РЕШЕЊА

## Микроконтролерски уређај са камером

У IoT системима "ивица" се односи на обраду података на месту или у близини извора њиховог генерисања, уместо да се подаци шаљу у централизовани облак или дата центар за обраду. Обрада на ивици смањује кашњење, побољшава време одзива и смањује количину података који се морају пренети преко мрежа.

Ивица овог система су ESP32 микроконтролери, са уграђеним камерама. Ц код, са помоћним библиотекама, који се извршава на микроконтролеру, користи камеру тако што узима један фрејм сваких 5 секунди. Како би се смањио број позива ка серверу, микроконтролер прво на ухваћеном фрејму проверава да ли постоји присуство људских лица, пре слања фрејма ка серверу, користећи уграђене могућности ESP32 микроконтролера за препознавање лица. Када би се сваки ухваћен фрејм слао ка серверу, непотребно би се користили ресурси обрадом сваке слике. Такође, када се не би користио овај приступ, корисник би добијао обавештења о активности сваких 5 секунди, што би претворило овај систем у лошију верзију сигурносне камере, уместо ефикаснијег решења за преглед активности, који не захтева од корисника да мора да приступа камерама директно.

Микроконтролер је једноставан, постоје функције за:

* Детекцију људских лица
* Слање API POST позива ка серверу са сликом и осталим информацијама (Датум, Време, Идентификатор камере)
* Преузимање тренутног фрејма камере, захваљујући званичној библиотеци ESP32 микроконтролера

## Cloud

У IoT системима, „облак“ се односи на употребу удаљених сервера за чување, управљање и обраду података прикупљених са ивице. Cloud технологија пружа скалабилно складиштење и рачунарску снагу, омогућавајући системима да анализирају велике количине података, управљају уређајима и покрећу апликације без ослањања на локалне ресурсе.

Овај централизовани приступ пружа већу флексибилност и контролу, подржава напредну анализу података и омогућава лаку интеграцију услуга и ажурирања на више уређаја, што га чини идеалним за велике инсталације.

Облак овог система се састоји од бекенда, који обрађује позиве, и складишта објеката, које чува податке.

### Backend

Бекенд овог система, написан у Python програмском језику, помоћу Flask оквира за изградњу веб сервиса, састоји се од:

* Посреднчког кода – декоратора
* Помоћних функција
* Рута за постављање слике, коју користи уређај
* Руте за управљање камерама
* Руте за управљање сликама као корисник
* Руте за управљање сачуваним лицима
* Руте за управљање подешавањима корисника
* Руте за пријављивање и регистровање корисника

#### Руте за пријављивање и регистровање корисника

Ове руте опслужују основно регистровање, пријављивање и контролу сесије корисника.

Када се корисник пријави, добије насумично генерисан кључ сесије, који служи као јединствени идентификатор за сваку од појединачних сесија корисника са различитих претраживача или уређаја. Ово омогућава кориснику да се одјави из појединачне сесије, без утицаја на друге, као и да спречи претње систему у виду малициозног коришћења API преко API клијената.

#### Руте за управљањем подешавањима корисника

Ове руте опслужују добављање сачуваних лица, корисникових додатих камера, као и подешавање телеграм интеграције.

#### Руте за управљање сачуваним лицима

Ове руте опслужују чување и брисање лица. Чување врши провере већ постојећег постављеног надимка и енокодованог лица, такође проверава постојање лица на слици, у случају да корисник жели да сачува лице са слике на којој оно није довољно дефинисано како би се у будућности могло коректно препознати.

Ако су провере испуњене, математички енкодовано лице се чува под надимком који је корисник изабрао.

#### Руте за управљање сликама као корисник

Ове руте обслужују добављање тренутних слика као и брисање слика које корисник не сматра више потребним.

#### Руте за управљање камерама

Ове руте облужују додавање, промену имена и брисање појединачних камера. Приликом додавања нових камера помоћу њиховог јединственог идентификатора, постоје провере за:

* Регистровање камере која припада неком другом кориснику
* Постојање идентификатора камере у систему
* Регистровања камере која већ припада кориснику

#### Рута за постављање слике

Ова рута је најбитнија у самом систему, јер опслужује најбитнију функцију, препознавање лица.

Када нека од камера детектује присуство лица на ухваћеном фрејму, она шаље захтев на ову руту, која затим проверава присуство слике у захтеву. Ако слика постоји у захтеву, врши се провера идентификатора камере, њеног постојања у систему, као и припадности те камере неком кориснику. Ако камера припада кориснику, лица са слике из захтева се енкодују, и пореде се са сачуваним енкодованим лицима корисника коме камера припада.

Ако је лице препознато, слика се не чува, већ се само шаље као обавештење путем Telegram апликације, које наглашава о ком сачуваном лицу је реч. Ако лице није препознато, слика се чува, како би корисник могао управљати њоме преко апликације, такође се шаље обавештење кориснику да је детектовано лице на камери, али и да није сачувано/непознато је.

#### Посреднчки код – декоратори

Dекоратор додаје провере аутентикације рути. Обухвата функцију руте и прво проверава да ли су корисничко име и токен сесије присутни у подацима из форме. Ако недостаје неки од њих, враћа грешку. Затим, проверава валидност сесијског токена. Ако је валидан, чува username у глобалном контексту, и наставља са извршавањем оригиналне функције руте.

#### Помоћне функције

Постоје помоћне функције, коришћене ради читљивости кода и сегментације.

Помоћне функције су:

* Функција за генерисање насумичног сесијског токена
* Функција за проверу постојања сесијског токена
* Функција за добављање објеката свих камера које припадају кориснику
* Функција за слање обавештења корисницима користећи јавни Telegram API

### База Података

Структура базе података је веома једноставна, постоји само једна колекција 'accounts' у којој сваки документ представља појединачног корисника.

Поља које поседује сваки корисник су:

* \_id - Насумично генерисан ID – string
* username - Корисничко име – string

password - Хешована и Посољена Корисничка Шифра – string

* session\_tokens - Низ токена сесије - array[string]
* chat\_id - Идентификатор Telegram разговора – string

cameras - Низ објеката камера које припадају корисику, објекат камере се састоји од имена и јединственог идентификатора камере - array[object]

* faces - Низ објеката енкодованих и сачуваних лица корисника, објекат се састоји од имена и математичке матрице која представља лице - array[object]

### Object Store

Како би се велики број слика чувао, као и обезбедила сигурност приказа слика само одговарајућим корисницима, потребно је брзо и сигурно решење попут складишта објеката, систем користи 'MinIO', пројекат отворене лиценце који пружа еквивалентан API амазоновом S3 решењу, али омогућава подацима да буду сервирани са системског сервера, уместо са амазонових AWS сервера. Ово обезбеђује систему пуну контролу над подацима, као и могућност сигурног приказа слика на апликацији, тако што само корисник коме камера припада може декриптовати садржај њених слика.

Свака камера у систему има своју 'канту', унутар које се чувају постављене слике у хронолошком редоследу.

## Frontend

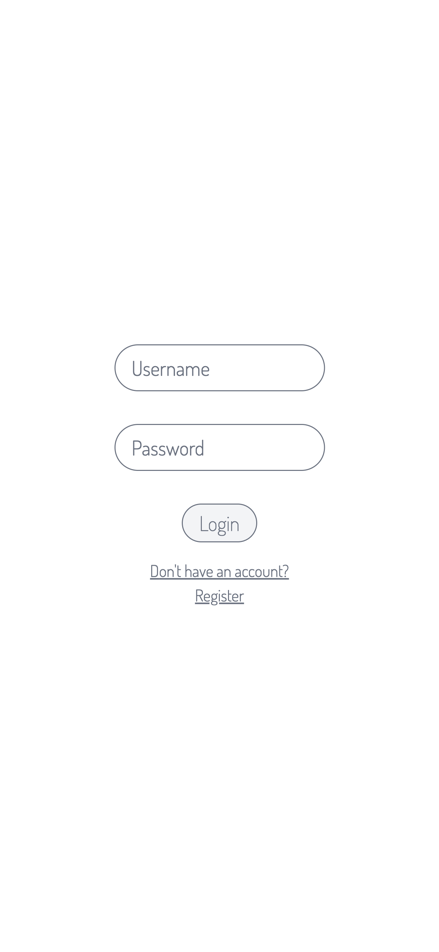
У апликацијама, фронтенд је слој са којим корисници директно комуницирају, одговоран за приказивање свих визуелних и интерактивних елемената. Његов задатак је да обезбеди добро корисничко искуство кроз функционалност, интуитивну навигацију и дизајн.

Фронтенд овог система састоји се od:

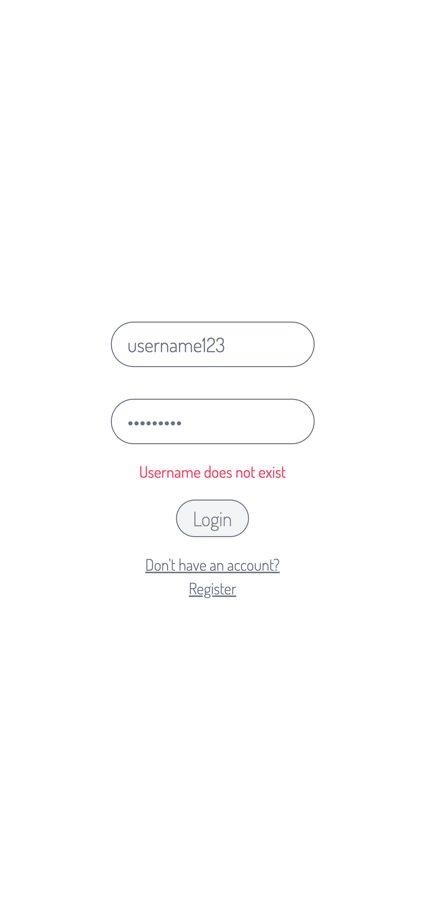
* Екрана за пријаву и регистрацију
* Екрана за преглед и управљање сликама
* Екрана за предлед и управљање подешавањима

### Екран за пријаву и регистрацију

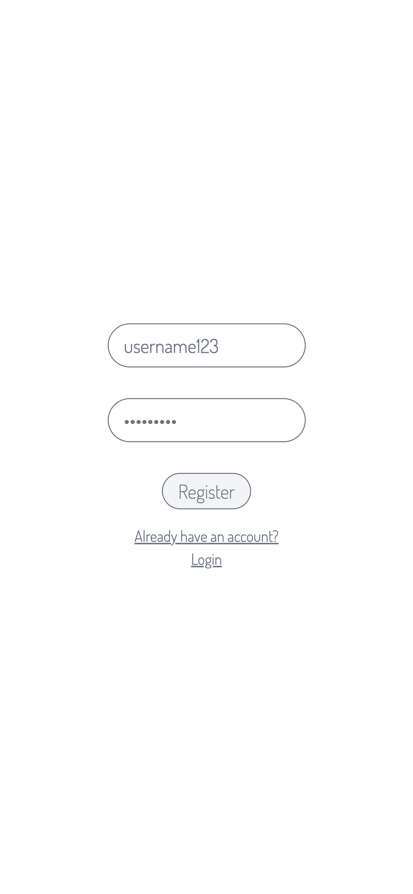
Екран ѕа пријаву и регистрацију је врло једноставан, садржи поља за унос корисничког имена и шифре, као и кондиционе поруке у случају да је дошло до грешке при пријави/регистрацији.



Слика 2.1 Екран за пријављивање



Слика 2.2 Порука грешке



Слика 2.3 Екран за регистрацију

### Екран за преглед и управљање сликама

Ово је главни екран апликације, на горњем делу екрана налази се једноставна навигациона трака, са једним дугметом, које садржи корисничко име.

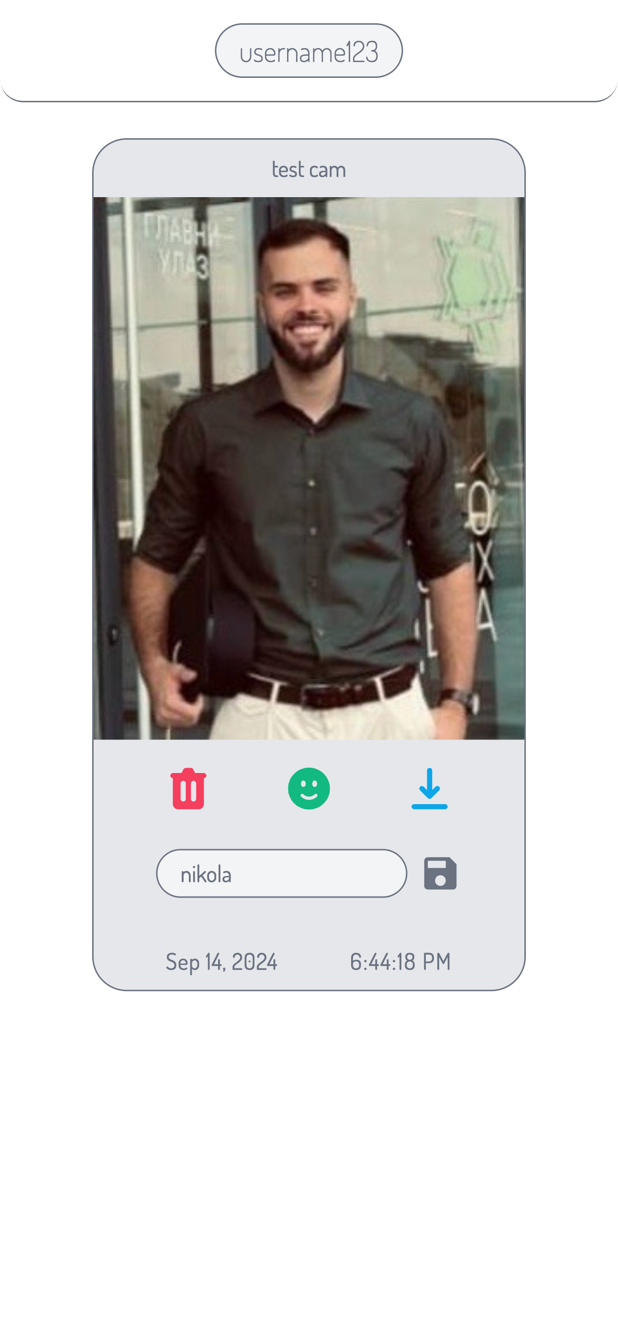
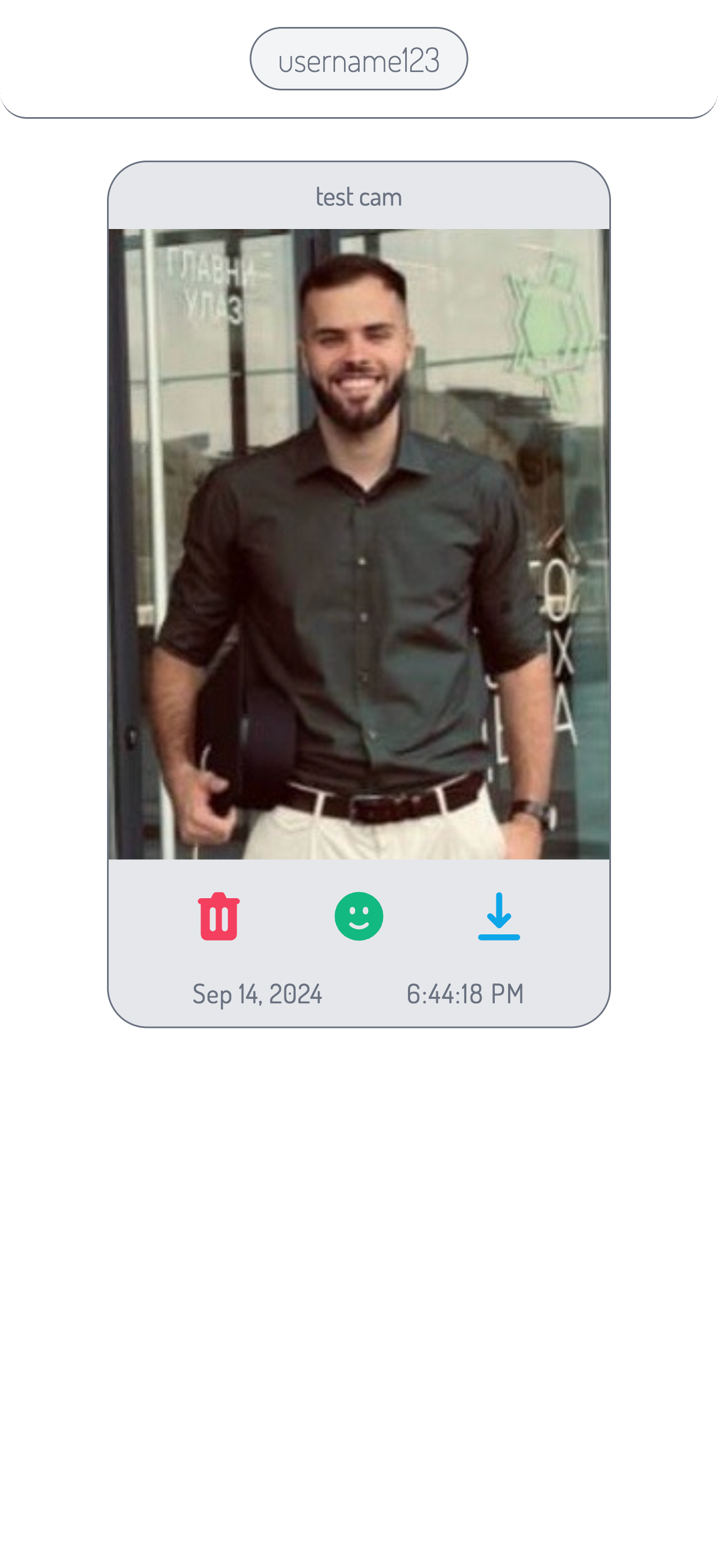
Кликом на дугме, отвара се падајући мени са три опције:

* Кликом на дугме, отвара се падајући мени са три опције:
* Settings, навигација на страницу подешавања
* Logout, одјављивање сесије корисника

Главни део екрана структуиран је као ток вести друштвених мрежа, ако нема нових слика које би корисник требао прегледати, постоји одговарајућа порука која обавештава корисника да је цео ток прегледан.

Уз слике на току, које су распоређене хоризонтално као објаве, стоје и информације о камери са које потичу и датума. Такође, испод сваке од слика, стоји трака са контролама, која садржи контроле за:

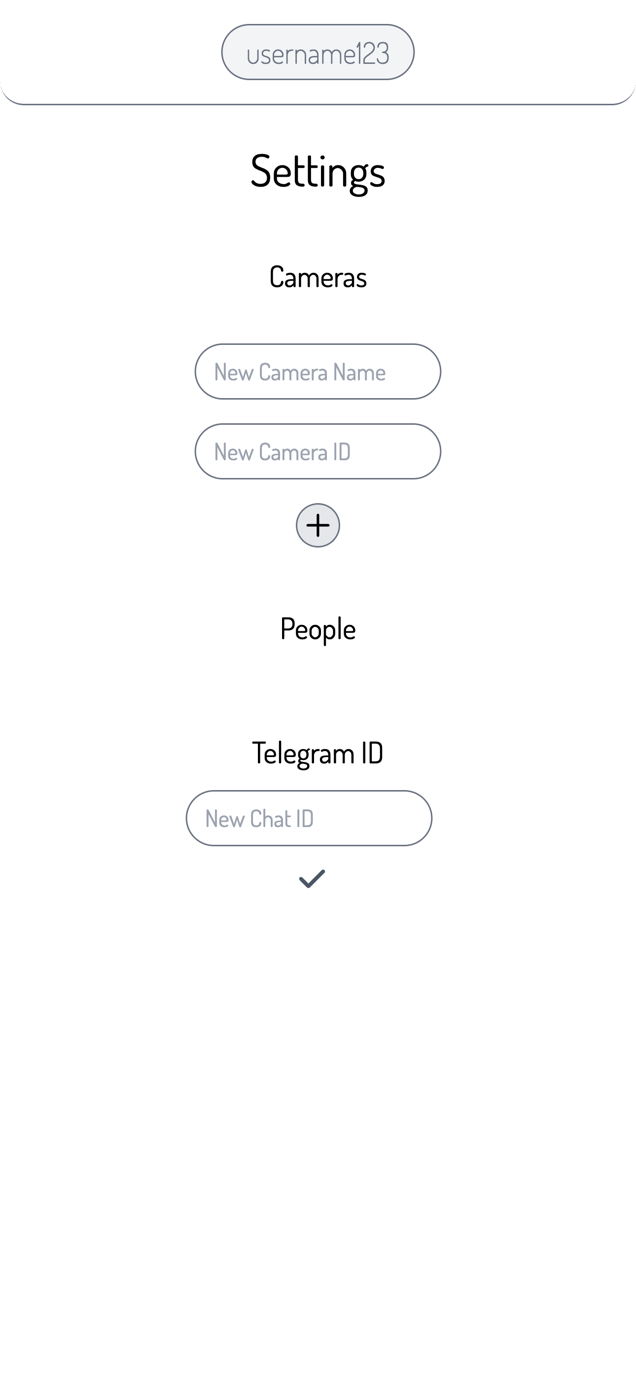
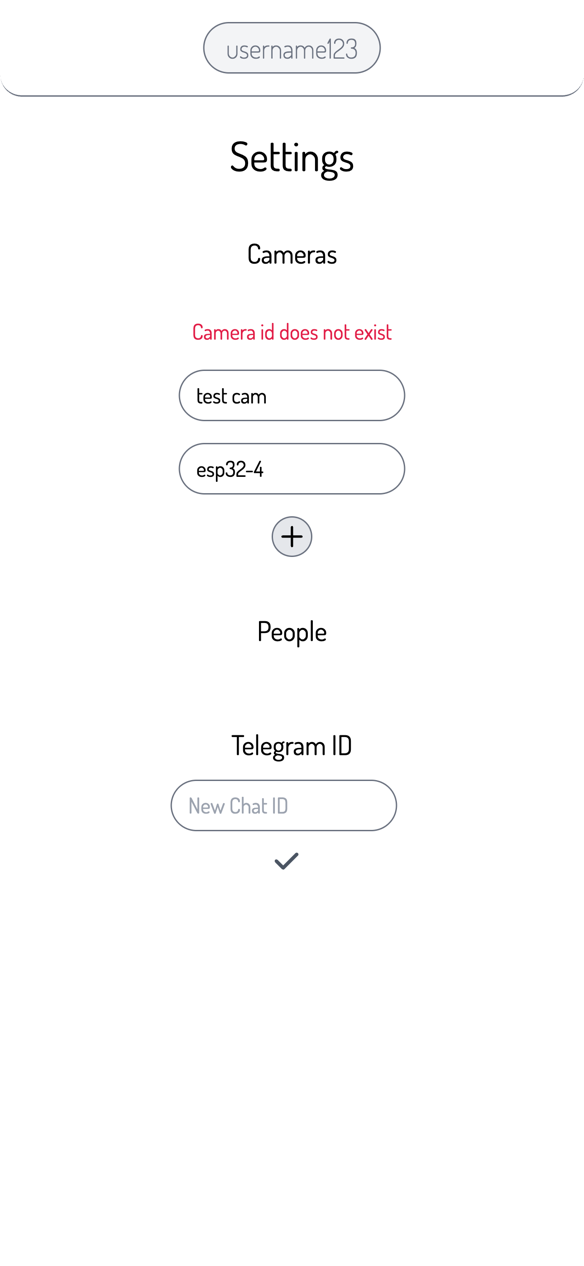
* Брисање слике
* Чување лица на слици
* Преузимање слике



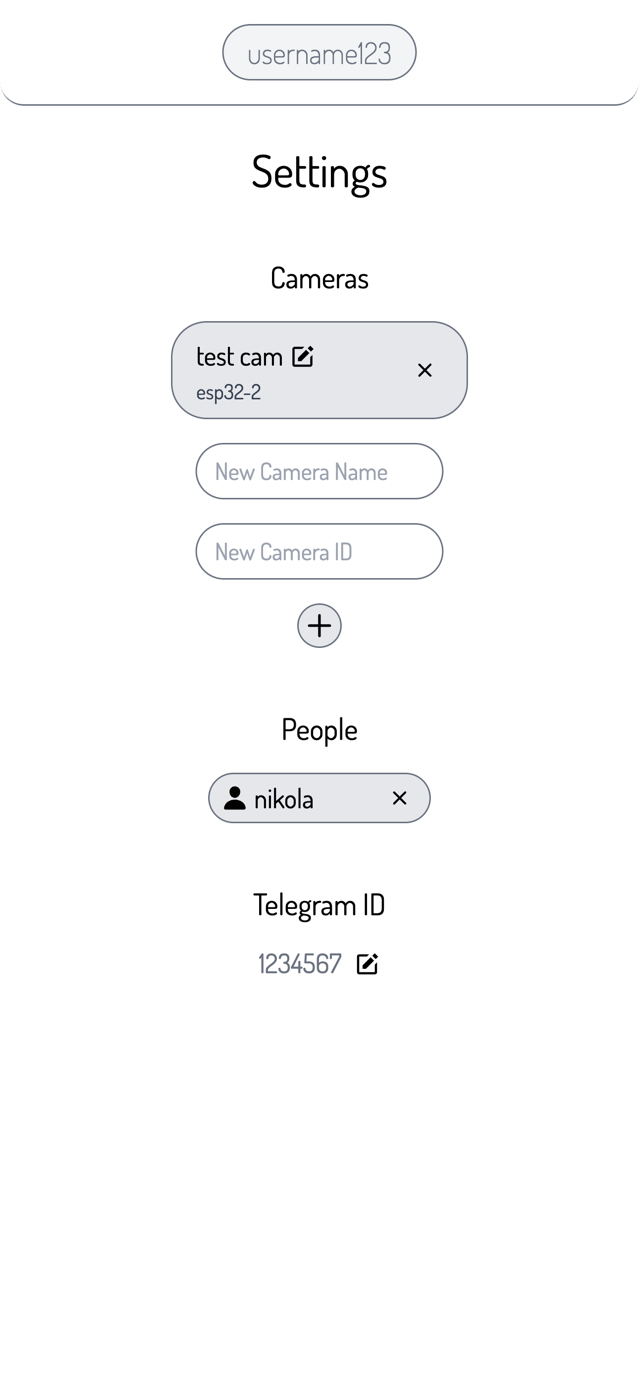
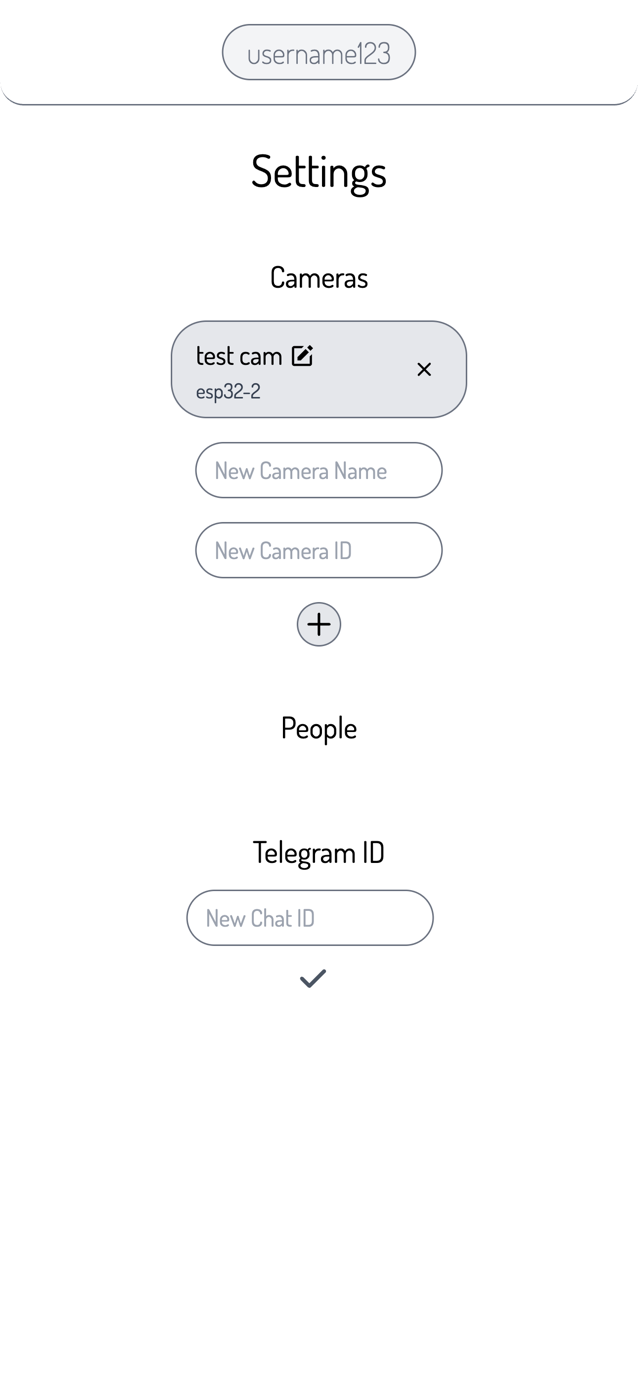
Слика 3.1 Објава на главном екрану Слика 3.2 Чување лица

### Екран за предлед и управљање подешавањима

На овом екрану, корисник може да управља додатим камерама тако што ће им променити име, или их уклонити. Такође може додати нову камеру. Постоји преглед свих сачуваних лица са могућношћу њиховог брисања.

Последња ставка је поље за унос идентификатора разговора, који се преузима из разговора са ботом апликације на Telegram платформи. 

Слика 4.1 Екран подешавања Слика 4.2 Приказ грешке



Слика 4.3 Додата камера Слика 4.4 Потпун Екран Подешавања

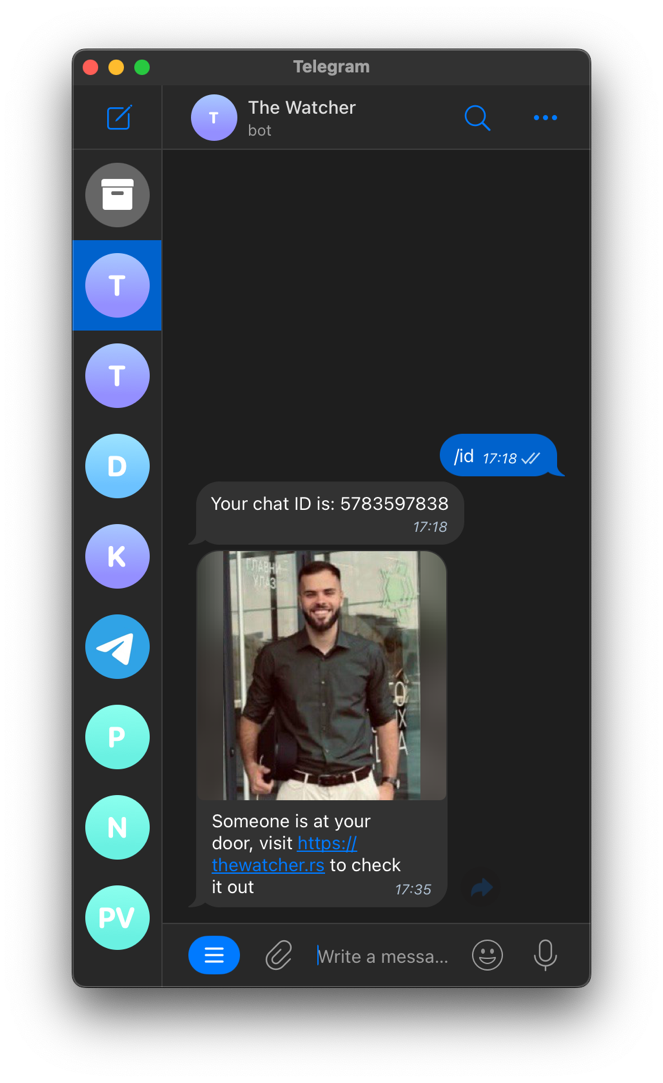
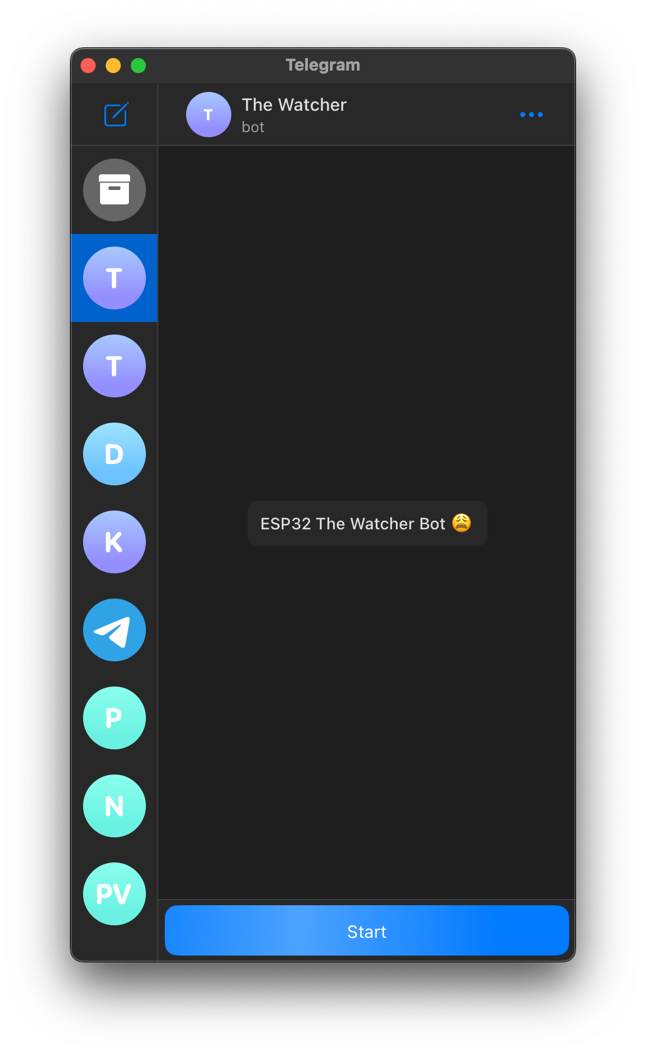
## Telegram

Телеграм је платформа за размену порука заснована на облаку, позната по брзини, сигурности и приватности корисника. Подржава функције као што су шифроване поруке, групе, канали и дељење медија.

BotFather је званични бот на Телеграму који омогућава корисницима да креирају и управљају својим ботoвима. Он нуди команде за постављање бота, укључујући генерисање јединственог API токена, управљање детаљима бота и омогућавање његових функција.

Телеграм Bot API је интерфејс заснован на HTTP протоколу који омогућава остваривање програмске интеракције са корисницима Телеграма. Подржава слање и примање порука.

Системски бот, коме корисници могу да приступе било када на телеграм апликацији, служи као канал обавештења. Слањем /id команде бот вам враћа поруку која садржи ваш јединствени идентификатор разговора, који када се унесе у подешавања апликације омогућава да примате обавештења о детектованим лицима преко Telegram платформе.



Слика 5.1 Приказ иницијалног разговора Слика 5.2 Приказ обавештења

# ИМПЛЕМЕНТАЦИЈА

## Микроконтролерски уређај са камером

Код микроконтролера написан је у Ц++ програмском језику, користећи Arduino проширења која нуде додатне функције и олакшања у писању кода за уграђене системе.

У функцији setup(), микроконтролер иницијализује камеру са одређеним подешавањима, као што су величина слике и квалитет. Повезује се на WiFi мрежу користећи SSID и лозинку и покреће камеру. Ако иницијализација камере не успе, ESP32 ће се поново покренути.

У функцији loop(), микроконтролер проверава да ли је прошло довољно времена од последњег снимања и слања слике. Затим снима слику, проверава да ли је детектовано лице коришћењем алгоритма за детекцију лица, и ако је лице пронађено, шаље слику серверу путем HTTP-а.

Функција faceDetected() обрађује снимљену слику и покреће алгоритам за детекцију како би идентификовала лица. Ако је једно или више лица детектовано, функција враћа true.

На крају, функција sendPhoto() управља слањем слике серверу путем HTTP POST захтева. Форматира слику као део multipart form-data захтева и шаље је на сервер. Ако веза са сервером не успе, исписује поруку о грешци.

## Backend

Поред основних CRUD (Create, Read, Update, Delete) функција за управљање подацима у бази података, бекенд овог система користи напредне функције обраде слика и руковања подацима.

### upload\_image()

Ова функција се користи за обраду слика које долазе са камера, анализу тих слика за препознавање лица, и евентуално чување слика ако лице није препознато. Процес иде кроз неколико корака:

1. Прво, функција користи Flask-ову request.files да извади слику коју је корисник послао путем HTTP POST захтева. Ако слика није пронађена, враћа се грешка са поруком „Image not found“.
2. Датотека слике је у бајтовима, па функција користи библиотеку Image из PIL-а (Python Imaging Library) како би декодирала те бајтове и претворила их у формат који може да се обрађује.
3. Након тога, функција користи библиотеку face\_recognition како би пронашла лица на слици. Енкодовање лица извршава се користећи face\_recognition.face\_encodings() – енкодовано лице је низ бројева који представљају карактеристике лица. Ако ниједно лице није пронађено на слици, функција враћа грешку.
4. Када су енкодована лица извучена са слике, следећи корак је упоредити их са лицима која су већ сачувана у корисниковом профилу. Функција преузима енкодовања познатих лица из базе података и користи face\_recognition.compare\_faces() да провери да ли се лице са слике поклапа са неким већ сачуваним лицем.
   * Ако постоји поклапање, функција шаље обавештење кориснику путем Телеграма користећи send\_telegram\_notification(). Слика се не чува јер лице већ постоји у бази.
   * Ако ниједно лице са слике није препознато, функција чува слику у MinIO складишту користећи API за рад са објектима. Слика се шаље као фајл у bucket специфичан за ту камеру. Након чувања, кориснику се шаље обавештење да неко (непознато лице) стоји испред камере. Следећи корак укључује отпремање слике у MinIO објекат. Функција користи MinIO API како би сачувала слику у одговарајући bucket базиран на camera\_id. Назив фајла је генерисан користећи тренутни timestamp како би био јединствен.

Функција је пажљиво дизајнирана да ради брзо, чак и са сликама високе резолуције, и да буде поуздана у раду са стварним системима.

### send\_telegram\_notification()

Ова функција омогућава слање нотификација путем Телеграма када се догоде одређени догађаји, као што је препознавање или откривање непознатог лица. Telegram bot API се користи за ово, где се HTTP POST захтев шаље директно Telegram серверима. Функција узима следеће параметре:

* chat\_id: Идентификатор Telegram корисника коме ће обавештење бити послато.
* text: Текстуална порука која ће бити приказана у обавештењу.
* image\_bytes: Бајтови слике која ће бити послата уз поруку. Ако слика није прослеђена, само текст ће бити приказан у обавештењу.

### face\_recognition.face\_encodings()

Ово је кључна функција из библиотеке face\_recognition, која генерише енкодовања за лица на слици. Функција анализира фацијалне карактеристике (као што су раздаљина између очију, облик носа и уста) и креира низ бројева који је јединствен за сваку особу. Енкодовања затим служе као "отисак прста" који се користи за поређење лица.

### face\_recognition.compare\_faces()

Ова функција упоређује енкодована лица са слике са сачуваним лицима из базе података. Ако се поклапање пронађе, функција враћа вредност True, што значи да је лице препознато. Ако не, враћа False, и тада систем зна да лице није препознато, па може да предузме даље кораке (као што је чување слике непознатог лица и слање обавештења).

## Telegram

Код бота дефинише се користећи библиотеку python-telegram-bot. Токен бота се преузима из променљиве окружења, а боту се додељује корисничко име. Главна функција, get\_chat\_id, ослушкује команду /id од корисника. Када се активира, преузима ID разговора из долазне поруке и враћа га кориснику као одговор.

## Frontend

Коришћењем React.js оквира омогућено је сегментисање кода у посебне компоненте, апликација овог система састоји се од 6 модула:

* App - Проверава креденцијале у локалном складишту претраживача и усмерава корисника на апликацију или страницу за пријаву/регистрацију.
* Login
* Options - Приказује корисничко име и мени са опцијама за "Home", "Settings", и "Logout". Кликом на корисничко име отвара се или затвара подмени.
* Page – Приказује главну страницу
* Settings
* Feed

### Login

Ова компонента управља формом за пријаву и регистрацију са обрадом грешака. Форма шаље POST захтев на /login или /register руту сервера, зависно од режима, и у случају успеха чува корисничко име и сесијски токен у локално складиште претраживача.

Обрада грешака се врши кроз мапу порука грешака, који повезује HTTP статус кодове са порукама попут "Корисничко име већ постоји" или "Погрешна лозинка". Ове грешке се приказују испод форме када је то потребно. Компонента такође омогућава кориснику да се пребацује између режима пријаве и регистрације.

### Settings

Интерфејс приказује листу регистрованих камера и лица, са опцијама за уређивање или брисање камера и лица, као и за додавање нових. Такође укључује функционалност за управљање идентификатором Telegram разговора, омогућавајући корисницима да виде или уреде свој тренутни ID. Користи се условно рендеровање за модове уређивања, а пружени су одговарајући симболи за побољшање корисничког искуства при додавању, брисању или ажурирању ставки.

### Feed

Компонента прво проверава да ли постоје кадрови за приказивање. Ако нема кадрова и учитавање је завршено, приказује поруку која указује да нема ништа за приказ. Ако су кадрови доступни, приказује их у распореду тока новости. Сваки кадар укључује слику, име камере и опције за брисање кадра, улазак у режим чувања лица и преузимање слике. Када корисник кликне на икону чувања лица, активира се режим уређивања у коме могу унети име за лица и сачувати га. Грешке се приказују када операција чувања не успе, ако име већ постоји или је то лице већ сачувано. Такође, компонента форматира и приказује датум и време повезане са сваком сликом.

# Закључак

Циљ овог дипломског рада била је имплементација система за паметан надзор, фокус је усмерен ка изради потпуног система, који се може користити лако и приступачно. Корисници система имају више начина за интеракцију са њим, било то преко Telegram платформе или веб интерфејса који је приступачан на рачунарима као и мобилним уређајима. Систем је стабилан, изграђен помоћу модерних технологија и парадигми. Пружа корисницима мир када је у питању надзор и не захтева од њих константну интеракцију и проверавање. Постоји пуно простора за унапређивање, поготово у виду корисничког интерфејса, као и самог кода бекенда и микроконтролера. Аутентикација би могла да користи новије стандарде попут JWT (Json Web Token) уместо токена сесије, структура базе података би могла бити робуснија, задржавајући још битних информација о корисницима и њиховим преференцама. Кориснички интерфејс би могао бити приступачнији, са већим фокусом на UX (user experience). Поред свега овога, за сврху дипломског рада, овај пројекат је потпун, и садржи све потребне функционалности какве би систем ове врсте требао да поседује.

# Литература

1. Др Илија Башичевић, Др Мирослав Поповић, Др Владимир Ковачевић: Основе рачунарских мрежа 1, Универзитет у Новом Саду, Факултет Техничких Наука, 2017
2. Douglas E. Comer: Internetworking With TCP/IP Vol I: Principles, Protocols, and Architecture Sixth Edition, Pearson, 2013
3. Gourley, David, and Brian Totty. HTTP: the definitive guide. " O'Reilly Media, Inc.", 2002.
4. Andrew Lombardi: WebSocket, lightweight client-server communications, O’Reilly, 2015
5. Maier, Alexander, Andrew Sharp, and Yuriy Vagapov. "Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things." 2017 Internet Technologies and Applications (ITA). IEEE, 2017.
6. Grinberg, Miguel. Flask web development. " O'Reilly Media, Inc.", 2018.
7. Gackenheimer, Cory. Introduction to React. Apress, 2015.
8. Al Salmi, Hatim. "Comparative CSS frameworks." Multi-Knowledge Electronic Comprehensive Journal For Education And Science Publications (MECSJ) 66 (2023).
9. Chauhan, Anjali. "A review on various aspects of MongoDB databases." International Journal of Engineering Research & Technology (IJERT) 8.05 (201`): 90-92.
10. Scott Chacon, Ben Straub: Pro Git, Apress, 2024