Mobilna iOS aplikacija za evidenciju nazočnosti studenata pomoću RFID/NFC čitača iksica

Đurčević, Juraj

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni prijediplomski studij Računarstvo

MOBILNA IOS APLIKACIJA ZA EVIDENCIJU NAZOČNOSTI STUDENATA POMOĆU RFID/NFC ČITAČA IKSICA

Završni rad

Juraj Đurčević

Osijek, 2024

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Juraj Đurčević
Studij, smjer:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	R4640, 27.07.2021.
JMBAG:	0165091067
Mentor:	izv. prof. dr. sc. Ivan Aleksi
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Mobilna iOS aplikacija za evidenciju nazočnosti studenata pomoću RFID/NFC čitača iksica
Znanstvena grana završnog rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak završnog rada:	Temu rezervirao Juraj Đurčević. Potrebno je napraviti mobilnu iOS aplikaciju za evidenciju nazočnosti studenata u nastavi na temelju iksice. Aplikacija treba prikazivati sliku, ime, prezime, e-mail te ostale osnovne podatke o studentu. Potrebno je realizirati bazu podataka iz koje je moguće prikazati trenutni postotak nazočnosti. Potrebno je omogućiti izvoz baze u excel tablicu. Aplikaciju je potrebno testirati i dokumentirati rezultate testiranja.

Datum prijedloga ocjene završnog rada od strane mentora:	12.09.2024.
Prijedlog ocjene završnog rada od strane mentora:	Izvrstan (5)
Datum potvrde ocjene završnog rada od strane Odbora:	25.09.2024.
Ocjena završnog rada nakon obrane:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	26.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 26.09.2024.

Ime i prezime Pristupnika:	Juraj Đurčević
Studij:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R4640, 27.07.2021.
Turnitin podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Mobilna iOS aplikacija za evidenciju nazočnosti studenata pomoću RFID/NFC čitača iksica**

izrađen pod vodstvom mentora izv. prof. dr. sc. Ivan Aleksi

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD

Evidencija nazočnosti studenata na nastavi predstavlja ključni dio akademskog procesa jer omogućuje praćenje prisutnosti studenata na predavanjima. Tradicionalni pristupi evidentiranja, poput ručnog bilježenja prisutnosti, često su podložni pogreškama, vremenski su zahtjevni i neefikasni. S razvojem novih tehnologija, poput RFID (engl. *Radio Frequency Identification*) i NFC (engl. *Near Field Communication*) sustava, otvara se mogućnost automatizacije ovog procesa. Ova tehnologija omogućuje brže, preciznije i sigurnije bilježenje prisutnosti, čime se znatno smanjuje mogućnost pogrešaka te se optimizira vrijeme predviđeno za vođenje evidencije.

U ovom završnom radu obrađuje se problem implementacije sustava za automatsku evidenciju nazočnosti studenata korištenjem RFID/NFC tehnologije u kombinaciji s mobilnom iOS aplikacijom. Aplikacija prikazuje ključne podatke o studentima, kao što su slika, ime, prezime i e-mail adresa, te omogućuje pregled trenutnog postotka nazočnosti. Sustav uključuje bazu podataka, iz koje se podaci prikazuju u aplikaciji, s dodatnom mogućnošću izvoza tih podataka u Excel tablicu radi lakše analize.

Struktura ovog rada sastoji se od nekoliko poglavlja. U drugom poglavlju opisuje se korišteni hardver i softver, uključujući Dasduino ESP32 Connectplus, PN532 čitač i Mifare kartice, Xcode i Swift te korištenje Firebase baze podataka. Treće poglavlje prikazuje način spajanja sklopa i funkcionalnosti koda. Četvrto poglavlje bavi se izradom aplikacije i njenom implementacijom. U petom poglavlju navedeni su izazovi, poput ograničenja sustava na jedan termin po kolegiju te problemi s kompatibilnošću Dasduino ESP32 pločice s Firebase bibliotekama, što je rezultiralo ručnim unosom ID-eva. Na kraju, šesto poglavlje donosi zaključak i prijedloge za unapređenje sustava.

1.1. Zadatak završnog rada

Zadatak završnog rada je izrada mobilne iOS aplikacije za evidenciju nazočnosti studenata korištenjem RFID/NFC tehnologije. Potrebno je omogućiti prikaz osnovnih podataka o studentima, izračun postotka nazočnosti te izvoz podataka u Excel tablicu. Aplikaciju je potrebno testirati i dokumentirati rezultate.

2. PREGLED PODRUČJA ZA EVIDENCIJU NAZOČNOSTI

Sustavi za automatsku evidenciju nazočnosti, poput RFID i NFC tehnologija, postali su standard u mnogim obrazovnim ustanovama zbog njihove brzine i preciznosti. RFID omogućuje čitanje kartica na daljinu, dok je NFC, kao podskup RFID-a, specifičan po radu na kraćim udaljenostima i koristi se za autentifikaciju i evidenciju nazočnosti putem mobilnih uređaja. Primjeri aplikacija za evidenciju nazočnosti su "Easy Attendance" i "Jibble".

2.1. Easy Attendance

Easy Attendance je jednostavna aplikacija za evidenciju nazočnosti koja se oslanja isključivo na ručni unos podataka. Profesori ili administratori ručno bilježe prisutnost studenata tijekom predavanja, bez korištenja naprednih tehnologija kao što su NFC ili QR kodovi. Ovaj pristup je pogodan za manje grupe studenata i za ustanove koje traže jednostavno rješenje bez potrebe za dodatnim uređajima ili složenim softverskim integracijama. Slika 2.1. prikazuje početni zaslon aplikacije [1].

Slika 2.1 Početni zaslon Easy Attendance aplikacije

2.2. Jibble

Jibble je naprednija aplikacija za praćenje radnog vremena i nazočnosti koja koristi NFC tehnologiju za evidenciju. Sustav omogućuje studentima ili zaposlenicima da se prijave jednostavnim prislanjanjem svojih NFC kartica na uređaje integrirane s aplikacijom. Ova rješenja često uključuju oblak za centraliziranu pohranu podataka, omogućujući administratorima pristup statistikama o nazočnosti s bilo koje lokacije i u bilo kojem trenutku. Jibble je posebno popularan

u korporativnom sektoru, ali nalazi svoju primjenu i u obrazovnim ustanovama zbog jednostavne integracije s postojećom infrastrukturom [2].

3. KORIŠTENE TEHNOLOGIJE

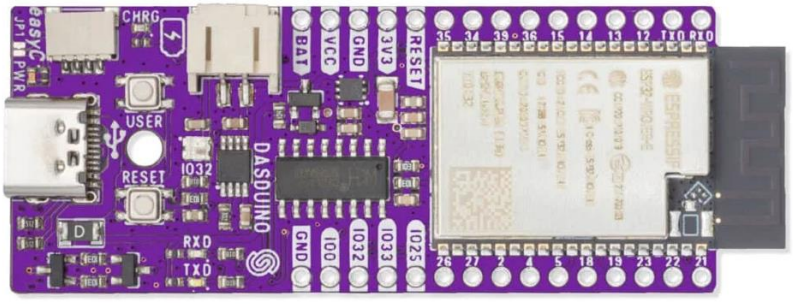
U ovom poglavlju opisuje se hardver i softver korišten u razvoju sustava za evidenciju nazočnosti studenata. Sustav je dizajniran za integraciju RFID/NFC tehnologije u praćenje nazočnosti studenata putem “iksica“, a obuhvaća odgovarajuće hardverske komponente i programske biblioteke potrebne za komunikaciju s bazom podataka i obradu podataka.

3.1. Računalna oprema

Korištena računalna oprema:

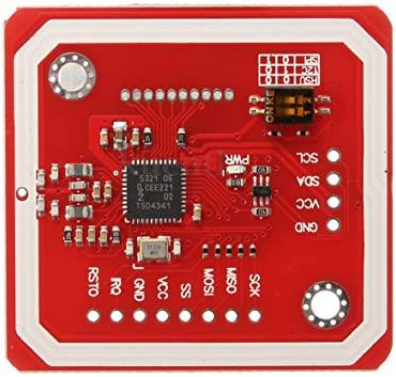
1. Dasduino ESP32 Connectplus
2. PN532 NFC RFID Module V4
3. EasyC adapter
4. Mifare kartice

Dasduino ESP32 Connectplus je središnji dio hardverskog sustava, odgovoran za obradu podataka i povezivanje s vanjskim uređajima. Ovaj mikrokontroler pruža napredne mogućnosti, uključujući podršku za Wi-Fi i Bluetooth, što omogućava brzu komunikaciju s bazom podataka u oblaku. Njegova fleksibilnost i podrška za razne senzore i periferije omogućava jednostavnu integraciju RFID/NFC čitača. U ovom sustavu ESP32 mikrokontroler upravlja procesom skeniranja iksica te obrađuje i šalje podatke o nazočnosti na Firebase bazu podataka [3].



Slika 3.1 Dasduino ESP32 Connectplus mikrokontroler

PN532 je RFID/NFC čitač koji se koristi za prepoznavanje i očitavanje podataka s Mifare kartica. Ovaj modul je poznat po svojoj pouzdanosti i brzini, te je široko korišten u sustavima za prepoznavanje korisnika putem beskontaktnih kartica. U ovom sustavu, PN532 modul se koristi za skeniranje iksica koje studenti koriste za identifikaciju prilikom ulaska u predavaonicu [4].



Slika 3.2 PN532 RFID/NFC čitač

EasyC adapter igra važnu ulogu u povezivanju hardverskih komponenti. Adapter omogućava stabilno povezivanje između Dasduino ESP32 Connectplus mikrokontrolera i PN532 NFC čitača. Povezivanje putem ovog funkcionira slično kao I2C protokol [5].

3.3 Dasduino EasyC adapter

Mifare kartice su beskontaktne pametne kartice koje studenti koriste za identifikaciju. Ove kartice pohranjuju osnovne podatke o studentima, poput njihovog ID broja, koji se koristi za evidenciju nazočnosti. Kada student prisloni svoju karticu na PN532 čitač, podaci se odmah očitavaju i šalju na obradu.

3.2. Programska oprema

3.2.1. Arduino IDE

Arduino IDE (engl. *Integrated Development Environment*) koristi se za pisanje, kompilaciju i prijenos koda na mikrokontrolere. Koristi programski jezik temeljen na C i C++, što omogućuje programerima da koriste dobro poznate sintakse i strukture ovih jezika. Ovaj programski jezik

omogućuje korisnicima jednostavan pristup funkcijama potrebnim za upravljanje hardverom, kao i širok spektar knjižnica koje proširuju funkcionalnost uređaja.

Jedan od ključnih elemenata Arduino IDE-a je podrška za serijsku komunikaciju putem UART protokola (engl. *Universal Asynchronous Receiver-Transmitter*), koji omogućuje prijenos podataka između mikrokontrolera i vanjskih uređaja. Korištenjem serijskog monitora unutar IDE-a, programeri mogu lako pratiti tok podataka između uređaja, što je vrlo korisno za dijagnostiku i testiranje.

Arduino mikrokontroleri dolaze s integriranim bootloaderom, koji pojednostavljuje proces prijenosa koda na uređaj. Bootloader omogućuje da se kod prenosi putem USB veze bez potrebe za vanjskim programatorom, olakšavajući cijeli proces razvoja. Ovo čini Arduino IDE pristupačnim i moćnim alatom za programiranje mikrokontrolera, s posebnim naglaskom na jednostavnost korištenja i fleksibilnost u radu s različitim hardverskim komponentama [6].

3.2.2. Xcode i Swift

Xcode je integrirano razvojno okruženje koje Apple nudi za razvoj aplikacija na svojim platformama, uključujući iOS. Xcode pruža sve potrebne alate za pisanje, testiranje i ispravljanje pogrešaka aplikacija, te integrira različite komponente kao što su Interface Builder za dizajn korisničkog sučelja i simulatore za testiranje aplikacija na različitim uređajima.

Swift je moderni programski jezik koji je Apple razvio za razvoj aplikacija na iOS, macOS, watchOS i tvOS platformama. Swift je poznat po svojoj sigurnosti, brzini i jednostavnosti korištenja, što ga čini idealnim izborom za razvoj mobilnih aplikacija. U ovom projektu, Swift se koristi za izradu funkcionalnosti aplikacije koja prikazuje podatke o studentima, prati nazočnost i omogućuje izvoz podataka u CSV (engl. *Comma Seperated Values*) tj. Excel formatu. Integracija Swift-a s Xcode-om omogućava brzi razvoj i iteraciju aplikacije, osiguravajući visokokvalitetno korisničko iskustvo [7].

3.2.3. Firebase

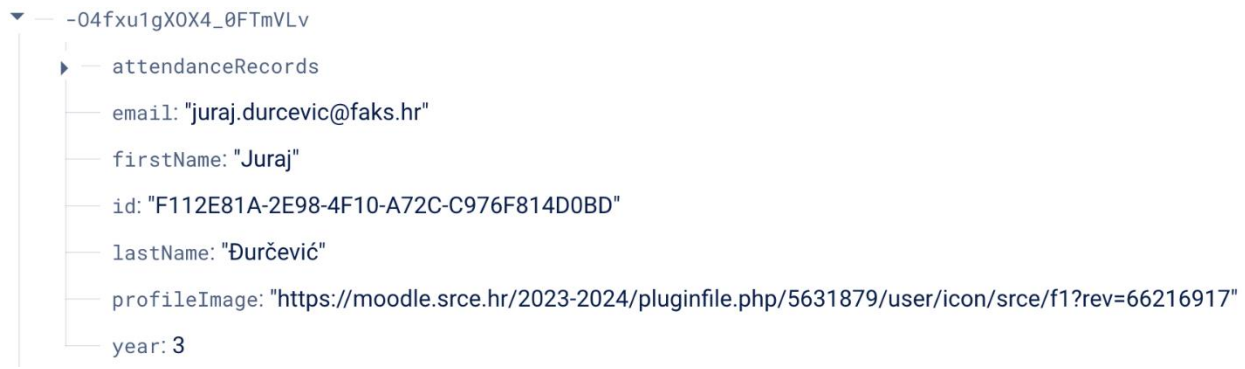
Firebase je platforma razvijena od strane Googlea, koja nudi niz alata i usluga za razvoj mobilnih i web aplikacija. U ovom projektu korištena je ključna značajka Firebase Realtime Database, koja

omogućuje pohranu i sinkronizaciju podataka u stvarnom vremenu između korisnika i aplikacije. Ova značajka osigurava da svi podaci budu trenutno ažurirani i vidljivi svim korisnicima bez potrebe za ručnim osvježavanjem.

Korištenjem Firebase-a, projekt omogućuje sigurno i učinkovito upravljanje podacima o nazočnosti studenata, pružajući brz pristup informacijama, jednostavnu analizu te mogućnost izvoza podataka u različite formate. Svi podaci aplikacije preuzimaju se iz Firebase baze podataka, a za razliku od tradicionalnih SQL baza, Firebase koristi JSON format za pohranu. Ovaj format omogućuje hijerarhijsko organiziranje podataka, gdje se svaki novi unos prikazuje kao novi "čvor" [8].

U čvoru koji se odnosi na kolegije, slika 3.5, pod šifrom svakog kolegija pohranjuju se podaci kao što su naziv kolegija, ID kolegija, ID-evi studenata koji pohađaju kolegij te datumi održavanja predavanja. U čvoru studenata, slika 3.4, pod svakim ID-om studenta nalaze se podaci poput imena, prezimena, emaila, godine studija, datuma prisustvovanja predavanjima i profilne slike.

Uz to, postoji i poseban čvor za prisustvo, slika 3.6, gdje se pomoću ESP32 uređaja šalju skenirani ID-ovi studenata. Ovi ID-ovi bilježe se u stvarnom vremenu kada studenti skeniraju svoje kartice, čime se automatski ažuriraju podaci o njihovom prisustvu na predavanju.



Slika 3.4 Struktura Student objekta na Firebase-u

Slika 3.5 Struktura Subject objekta na Firebase-u

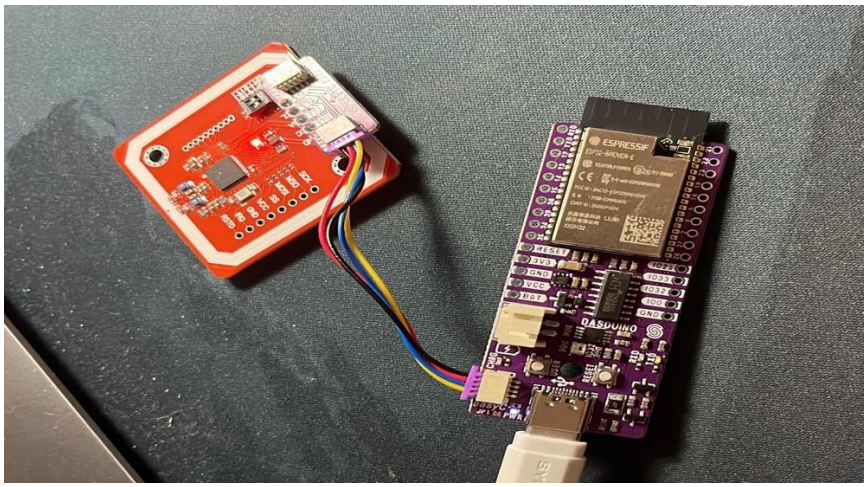
Slika 3.6 Struktura Attendance objekta na Firebase-u

4. INTEGRACIJA HARDVERSKIH KOMPONENTI

4.1. Spajanje komponenti

Za pravilno funkcioniranje sustava korišten je Dasduino ESP32 Connectplus mikrokontroler, koji je povezan s PN532 NFC čitačem koristeći EasyC adapter i odgovarajući EasyC kabel. Adapter EasyC, koji u suštini omogućava I2C komunikaciju, zalemljen je na odgovarajuće pinove na PN532 (GND, VCC, SCL i SDA), čime je omogućeno pravilno povezivanje i prijenos podataka između mikrokontrolera i čitača kartica. Na PN532 je odabran I2C način rada, a postavljen je pomoću switcheva koji se nalaze na samom čitaču.

Kompletan sklop, prikazan na slici 4.1, se povezuje putem USB kabela na računalo, što omogućuje napajanje mikrokontrolera i NFC čitača, kao i jednostavno učitavanje koda potrebnog za rad sustava. Tako konfiguriran sustav omogućuje precizno očitavanje MIFARE kartica, koje studenti koriste za evidenciju nazočnosti u nastavi.



Slika 4.1 Spoj Dasduino ESP32 Connectplus-a i PN532 čitača pomoću EasyC adaptera

4.2. Kod za komunikaciju s čitačem kartica i Firebase bazom podataka

Korišteni kod implementira sustav za čitanje NFC kartica pomoću ESP32 pločice i PN532 NFC/RFID modula te pohranjuje podatke o pročitanim karticama u Firebase Realtime bazu podataka. Nakon inicijalizacije Wi-Fi veze i uspješne autentifikacije s Firebaseom, sustav kontinuirano očitava NFC kartice. Kada se kartica prepozna, njen jedinstveni identifikator (UID) prikazuje se na serijskom monitoru i pohranjuje u Firebase. Osim toga, sustav periodično šalje vrijednost brojača u bazu podataka svake sekunde, omogućujući dodatno praćenje aktivnosti [11].

4.2.1. Uključivanje biblioteka i definicija

U 4.2 prikazu koda uključuju se knjižnice koje omogućuju rad svih ključnih komponenti:

- Wire.h omogućuje I2C komunikaciju, koja se koristi za povezivanje ESP32 s NFC/RFID modulom.
- Adafruit_PN532.h knjižnica koristi se za rad s PN532 NFC/RFID modulom, omogućujući čitanje i pisanje podataka s kartica [9].
- Arduino.h pruža osnovne funkcije za rad s ESP32 mikrokontrolerom, kao što su serijska komunikacija i rad s I/O pinovima.
- WiFi.h koristi se za povezivanje ESP32 na Wi-Fi mrežu.
- Firebase_ESP_Client.h omogućuje komunikaciju između ESP32 i Firebase Realtime baze podataka [10].
- Dodatno, TokenHelper.h i RTDBHelper.h pomažu u upravljanju autentifikacijom i ispisu informacija prilikom slanja podataka na Firebase.

Ove knjižnice zajedno omogućuju interakciju s NFC/RFID karticama, slanje podataka u realnom vremenu na Firebase te povezivanje na internet putem Wi-Fi mreže.

```
1   #include <Wire.h>
2   #include <Adafruit_PN532.h>
3   #include <Arduino.h>
4   #include <WiFi.h>
5   #include <Firebase_ESP_Client.h>
6
7   //Provide the token generation process info.
8   #include "addons/TokenHelper.h"
9   //Provide the RTDB payload printing info and other helper functions.
10  #include "addons/RTDBHelper.h"
```

Slika 4.2 Kod za uključivanje biblioteka

4.2.2. Definiranje Wi-Fi i Firebase podataka

U 4.3 isječku koda su definirani ključni parametri za Wi-Fi povezivanje i Firebase autentifikaciju.

- WIFI_SSID i WIFI_PASSWORD predstavljaju naziv i lozinku Wi-Fi mreže koja omogućuje ESP32-u pristup internetu.
- API_KEY je API ključ specifičan za Firebase projekt, koji omogućuje sigurno slanje podataka prema bazi.

- DATABASE_URL služi kao URL Firebase Realtime baze podataka, gdje će se pohranjivati podaci prikupljeni iz aplikacije.

ESP32 koristi ove vrijednosti za povezivanje na mrežu i pristup bazi podataka, no stvarni podaci su izostavljeni kako bi se osigurala sigurnost projekta.

```
14  #define WIFI_SSID "Naziv wifi-a"
15  #define WIFI_PASSWORD "Lozinka wifi-a"
16  #define API_KEY "Api ključ baze"
17  #define DATABASE_URL "Url baze"
```

Slika 4.3 Kod za definiranje Wi-Fi i Firebase podataka

4.2.3. Firebase konfiguracija i objekti

Prikaz 4.4 ilustrira dio koda gdje se deklariraju ključni objekti za rad s Firebaseom:

- FirebaseData fbdo upravlja podacima između ESP32 i Firebasea.
- FirebaseAuth auth omogućuje autentifikaciju korisnika.
- FirebaseConfig config pohranjuje postavke za spajanje na Firebase, uključujući API ključ i URL baze.
- Također, sendDataPrevMillis prati vrijeme između slanja podataka, dok count bilježi broj poslanih poruka. signupOK označava je li autentifikacija uspješna.

```
20  FirebaseData fbdo;
21  FirebaseAuth auth;
22  FirebaseConfig config;
23
24  unsigned long sendDataPrevMillis = 0;
25  int count = 0;
26  bool signupOK = false;
```

Slika 4.4 Kod za Firebase konfiguraciju i dodatni objekti

4.2.4. NFC definicije

U kodu 4.5 definiraju se postavke potrebne za komunikaciju s NFC čitačem:

- PN532_IRQ i PN532_RESET predstavljaju pinove na ESP32 pločici koji su povezani na NFC modul (pinovi 21 i 22).
- Adafruit_PN532 nfc inicijalizira objekt koji omogućuje rad s NFC modulom koristeći I2C protokol.

Ove definicije omogućuju uspostavljanje pravilne komunikacije između ESP32 i NFC čitača.

```
29    #define PN532_IRQ   (21)
30    #define PN532_RESET (22)
31    Adafruit_PN532 nfc(PN532_IRQ, PN532_RESET);
```

Slika 4.5 Kod gdje se definiraju NFC podaci

4.2.5. Setup funkcija

Kada se uređaj pokrene, izvršava se setup() funkcija, prikazana isječcima 4.6 i 4.7, koja pokreće glavne dijelove sustava.

- Na početku, koristi se Serial.begin(115200) za uspostavljanje serijske komunikacije pri brzini od 115200 bauda u sekundi, što omogućava ispis podataka u serijski monitor za dijagnostiku.
- Zatim slijedi proces povezivanja na Wi-Fi mrežu putem WiFi.begin(WIFI_SSID, WIFI_PASSWORD). Uređaj čeka dok ne uspostavi vezu, nakon čega ispisuje IP adresu na serijski monitor.
- Firebase se inicijalizira putem config.api_key i config.database_url, a u slučaju uspješne prijave, postavlja se zastavica signupOK na true.
- Na kraju, NFC čitač se inicijalizira i ispisuje verzija ugrađenog softvera ako je uspješno pronađen PN532 čip.

```
33  void setup() {
34    Serial.begin(115200);
35
36    // Wi-Fi povezivanje
37    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
38    Serial.print("Connecting to Wi-Fi");
39    while (WiFi.status() != WL_CONNECTED) {
40      Serial.print(".");
41      delay(300);
42    }
43    Serial.println();
44    Serial.print("Connected with IP: ");
45    Serial.println(WiFi.localIP());
46    Serial.println();
47
48    // Firebase inicijalizacija
49    config.api_key = API_KEY;
50    config.database_url = DATABASE_URL;
51
52    if (Firebase.signUp(&config, &auth, "", "")) {
53      Serial.println("Firebase Signup OK");
54      signupOK = true;
55    } else {
56      Serial.printf("Firebase Signup Error: %s\n", config.signer.signupError.message.c_str());
57    }
58
59    config.token_status_callback = tokenStatusCallback;
60    Firebase.begin(&config, &auth);
61    Firebase.reconnectWiFi(true);
```

Slika 4.6 Prva polovina koda setup funkcije programa

```
63    // NFC inicijalizacija
64    nfc.begin();
65    uint32_t versiondata = nfc.getFirmwareVersion();
66    if (!versiondata) {
67      Serial.println("Didn't find PN53x board");
68      while (1);
69    }
70    Serial.print("Found chip PN5");
71    Serial.println((versiondata >> 24) & 0xFF, HEX);
72    Serial.print("Firmware ver. ");
73    Serial.print((versiondata >> 16) & 0xFF, DEC);
74    Serial.print('.');
75    Serial.println((versiondata >> 8) & 0xFF, DEC);
76    Serial.println("Waiting for an ISO14443A Card ...");
77  }
```

Slika 4.7 Druga polovina koda setup funkcije programa

4.2.6. NFC čitanje u loop funkciji

Funkcija loop() neprekidno provjerava prisutnost NFC kartice. Kada se kartica detektira, očita se njen UID (jedinstveni identifikator), a njegova duljina i vrijednost ispisuju se na serijski monitor.

- nfc.readPassiveTargetID() očitava ID kartice koristeći ISO14443A protokol, koji je standard za Mifare kartice.
- Ako je očitavanje uspješno (success), prikazuje se UID i broj bajtova koje kartica koristi.

Na slici 4.8 možemo vidjeti prethodno opisani postupak.

```
79  void loop() {
80    // NFC čitanje
81    uint8_t success;
82    uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };
83    uint8_t uidLength;
84
85    success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength);
86
87    if (success) {
88      Serial.println("Found an ISO14443A card");
89      Serial.print("  UID Length: ");
90      Serial.print(uidLength, DEC);
91      Serial.println(" bytes");
92      Serial.print("  UID Value: ");
93      nfc.PrintHex(uid, uidLength);
94      Serial.println();
```

Slika 4.8 Kod za čitanje MIFARE kartice

4.2.7. Spremanje UID-a na Firebase

Ako je Firebase spreman i prijava je uspješna, UID očitane NFC kartice sprema se u Firebase bazu podataka, što je prikazano na slici 4.9.

- Firebase.ready() provjerava je li Firebase veza aktivna, dok signupOK osigurava da je prijava na Firebase uspješna.
- uidString konvertira UID iz bajtova u heksadecimalni string, koji se zatim pohranjuje u Firebase putem funkcije Firebase.RTDB.setString().
- Ako je slanje uspješno, ispisuje se poruka o uspjehu, a u slučaju greške prikazuje se razlog.

```
96      // Spremanje podataka u Firebase
97      if (Firebase.ready() && signupOK) {
98        String path = "nfc_cards/card_uid";
99        String uidString = "";
100       for (int i = 0; i < uidLength; i++) {
101         uidString += String(uid[i], HEX);
102       }
103
104       if (Firebase.RTDB.setString(&fbdo, path, uidString)) {
105         Serial.println("UID stored in Firebase!");
106       } else {
107         Serial.println("Failed to store UID in Firebase!");
108         Serial.println(fbdo.errorReason());
109       }
110     }
111   }
```

Slika 4.9 Kod za spremanje ID-a studenta na Firebase

4.2.8. Periodično slanje brojača na Firebase

Periodično slanje podataka implementirano je pomoću brojača, koji se svake sekunde šalje na Firebase, što je prikazano na slici 4.10.

- Funkcija millis() mjeri proteklo vrijeme od početka rada programa. Ako je prošlo više od 1 sekunde ili je ovo prvo slanje (sendDataPrevMillis == 0), podatak se šalje na Firebase.
- Firebase.RTDB.setInt() funkcija koristi se za slanje vrijednosti brojača na Firebase pod ključem "test/int".
- Uspjeh ili neuspjeh slanja prikazuje se na serijskom monitoru.

```
112   // Firebase periodičko slanje podataka
113   if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 1000 || sendDataPrevMillis == 0)) {
114     sendDataPrevMillis = millis();
115
116     // Slanje brojača na Firebase
117     if (Firebase.RTDB.setInt(&fbdo, "test/int", count)) {
118       Serial.println("PASSED");
119       Serial.println("PATH: " + fbdo.dataPath());
120       Serial.println("TYPE: " + fbdo.dataType());
121     } else {
122       Serial.println("FAILED");
123       Serial.println("REASON: " + fbdo.errorReason());
124     }
125     count++;
126   }
127 }
```

Slika 4.10 Kod za periodično slanje brojača na Firebase

5. RAZVOJ APLIKACIJE

5.1. Struktura podataka i vezane funkcije

U aplikaciji se koriste dvije osnovne klase za pohranu i upravljanje podacima: Student i Subject. Ove klase omogućuju upravljanje informacijama o studentima i kolegijima, kao i evidenciju nazočnosti za svakog studenta u svakom kolegiju.

5.1.1. Student klasa

Kod u 5.1 predstavlja klasu Student koja sadrži osnovne podatke o studentu, uključujući jedinstveni ID studenta, ime, prezime, godinu studija, e-mail adresu, URL slike profila te niz datuma koji predstavljaju evidenciju nazočnosti. Svaki student može imati više zapisa o nazočnosti, a ti zapisi se pohranjuju kao datumi. Klasu Student čine i dodatne funkcionalnosti, kao što je addAttendanceRecord, koja omogućuje dodavanje novog datuma nazočnosti.

Klasa također implementira protokole Identifiable i Codable. Implementacija Identifiable znači da svaki student ima jedinstveni id, što omogućava jednostavno razlikovanje između objekata studenata, dok implementacija Codable omogućava jednostavnu serijalizaciju i deserijalizaciju objekata, odnosno pretvaranje objekta u JSON format i obratno. To je korisno prilikom slanja ili primanja podataka iz baze podataka ili API-ja.

```
3  class Student: Identifiable, Codable {
4      var id: String = UUID().uuidString      // Jedinstveni identifikator studenta (može biti broj iksice)
5      var firstName: String    // Ime studenta
6      var lastName: String     // Prezime studenta
7      var year: Int            // Godina studija studenta
8      var email: String        // E-mail adresa studenta
9      var profileImage: URL  // Slika profila studenta (pohranjena kao Data)
10     var attendanceRecords: [Date] // Evidencija nazočnosti (datumi kada je student bio prisutan)
```

Slika 5.1 Kod klase Student

5.1.2. Kolegij klasa

Kod u 5.2 predstavlja klasu Subject koja pohranjuje informacije o kolegiju, uključujući jedinstveni ID kolegija, naziv kolegija, popis studenata koji pohađaju kolegij te niz datuma održanih predavanja. Funkcija addAttendanceRecord omogućuje dodavanje novog datuma predavanja, dok metoda stripTime uklanja vremenske komponente kako bi se datumi ispravno uspoređivali i

pohranjivali. Kao i klasa Student, implementira protokole Identifiable i Codable, što olakšava rad s jedinstvenim objektima i njihovo pohranjivanje.

```
3  class Subject: Identifiable, Codable {
4      var id: String = UUID().uuidString
5      var name: String
6      var students: [String]
7      var attendanceDates: [Date]
```

Slika 5.2 Kod klase Subject

5.1.3. Klase za rad s Firebaseom

Klasa StudentData služi za upravljanje podacima o studentima u aplikaciji. Omogućuje dodavanje, ažuriranje i dohvaćanje studenata iz baze podataka, kao i provjeru je li student već registriran. Glavni atribut ove klase je:

- students: lista svih studenata tj. objekata Student preuzetih iz baze podataka.

Važne funkcije:

- registerStudent(student: Student) – Dodaje novog studenta u bazu podataka i pohranjuje njegove podatke koristeći POST zahtjev.

- updateStudent(student: Student) – Ažurira podatke o postojećem studentu u bazi podataka. Pronađe odgovarajućeg studenta prema ID-u i ažurira njegove podatke.

- fetchStudents() – Dohvaća sve studente iz baze podataka koristeći GET zahtjev i pohranjuje ih u listu students.

- isStudentRegistered(studentIdentifier: String) – Provjerava je li student s određenim ID-om već registriran u aplikaciji.

Slična klasa SubjectData postoji za upravljanje kolegijima, s istim funkcijama kao za studente.

5.1.4. Funkcija za ažuriranje

Primjer 5.3 predstavlja funkciju updateStudent koja služi za ažuriranje podataka o studentu u bazi podataka. Proces započinje dohvaćanjem svih studenata iz baze koristeći asinkroni zahtjev, nakon čega se dobiveni podaci dekodiraju u obliku rječnika, gdje su ključevi jedinstveni identifikatori studenata. Zatim se iterira kroz sve studente kako bi se pronašao onaj koji ima isti ID kao i onaj koji treba biti ažuriran. Nakon pronalaska, podaci studenta se ažuriraju, kodiraju u JSON format i šalju putem PUT zahtjeva prema bazi podataka na odgovarajući URL. Nakon što se podaci uspješno ažuriraju, funkcija ispisuje odgovor servera kako bi se potvrdilo da je operacija prošla ispravno.

```
45  func updateStudent(student: Student) async
46  {
47          do {
48              // Fetch all users
49              let (data, _) = try await URLSession.shared.data(from: students_url)
50
51              // Decode users data
52              let decoder = JSONDecoder()
53              let decodedStudents = try decoder.decode([String: Student].self, from: data)
54
55              // Iterate through users to find and update matching user
56              for (key, existingStudent) in decodedStudents {
57                  if existingStudent.id == student.id {
58                      // Update user if names match
59                      let encoder = JSONEncoder()
60                      encoder.dateEncodingStrategy = .iso8601
61                      let json = try encoder.encode(student)
62
63                      // Construct URL with key appended to users_url
64                      let urlWithKey = URL(string: "\(studentPut_url)/\(key).json")!
65
66                      print(urlWithKey)
67
68                      var request = URLRequest(url: urlWithKey)
69                      request.httpMethod = "PUT"
70                      request.httpBody = json
71
72                      let (_, response) = try await URLSession.shared.data(for: request)
73                      print(response)
74                      // Assuming only one user can have the same name,
75                      // break the loop if a match is found
76                      break
77                  }
78              }
79          } catch let error {
80              print(error)
81          }
82  }
```

Slika 5.3 Kod funkcije updateStudent

5.1.5. Funkcija za dohvaćanje

Funkcija fetchStudents, prikazana u isječku 5.4, koristi se za dohvaćanje svih studenata iz baze podataka. Proces započinje asinkronim HTTP zahtjevom prema bazi podataka na zadanoj URL adresi. Nakon uspješnog dohvaćanja podataka, oni se dekodiraju pomoću JSONDecoder objekta, koji pretvara JSON u oblik studenta uz prilagodbu datuma na ISO8601 format. Dobiveni podaci se pohranjuju u lokalnu varijablu students, koja sadrži sve dekodirane studente. Za dodatnu

provjeru, funkcija ispisuje ključeve i vrijednosti svakog studenta. U slučaju greške tijekom dohvaćanja ili dekodiranja podataka, greška se ispisuje.

```
85      func fetchStudents() async
86      {
87          do {
88              let (data, _) = try await URLSession.shared.data(from: students_url)
89
90              let decoder = JSONDecoder()
91              decoder.dateDecodingStrategy = .iso8601
92
93              let decoded_users = try decoder.decode([String: Student].self, from: data)
94              students = [Student](decoded_users.values)
95
96              for (key, value) in decoded_users {
97                          print("Key: \(key), Value: \(value)")
98                  }
99
100         } catch let error {
101             print(error)
102         }
103     }
```

Slika 5.4 Kod funkcije fetchStudents

5.2. Pokretanje aplikacije

Kod 5.5 prikazuje strukturu glavnog prikaza aplikacije korištenjem SwiftUI okvira. Aplikacija koristi TabView, koji omogućava navigaciju kroz različite dijelove aplikacije, svaku s odgovarajućom ikonom i naslovom. Unutar TabView, imamo tri osnovne stavke:

- SettingsView (prikazana ikonom zupčanika) omogućava korisnicima pristup postavkama aplikacije.
- MainView (ikona povećala) služi za pregled glavnih funkcionalnosti aplikacije, poput pregleda podataka o studentima i njihovim postocima nazočnosti.
- AddLectureView (ikona kalendara s plusom) omogućava dodavanje novih predavanja unutar aplikacije.

Svaka od ovih stavki je stilizirana s bijelom pozadinom na alatnoj traci. TabView također postavlja accentColor na plavu boju kako bi se osiguralo vizualno konzistentno iskustvo kroz cijelu aplikaciju. Osim toga, koriste se environmentObject za dijeljenje podataka o studentima, kolegijima, nazočnosti i temama kroz različite prikaze aplikacije. Prilikom pokretanja aplikacije, pozivaju se metode fetchStudents i fetchSubjects kako bi se dohvatili svi potrebni podaci iz baze.

```
27          WindowGroup {
28              TabView(selection: $selectedTab) {
29                  SettingsView()
30                      .tabItem {
31                          Label("", systemImage: "gearshape.fill")
32                      }
33                      .tag(1)
34                      .toolbarBackground(Color.white)
35                      .toolbarBackground(.visible, for: .tabBar)
36                  MainView()
37                      .tabItem {
38                          Label("", systemImage: "magnifyingglass.circle")
39                      }
40                      .tag(0)
41                      .toolbarBackground(Color.white)
42                      .toolbarBackground(.visible, for: .tabBar)
43                  AddLectureView()
44                      .tabItem {
45                          Label("", systemImage: "calendar.badge.plus")
46                      }
47                      .tag(2)
48                      .toolbarBackground(Color.white)
49                      .toolbarBackground(.visible, for: .tabBar)
50              }  // Scale down the entire TabView
51              .padding(.bottom, -65)
52              .accentColor(.blue)
53              .environmentObject(studentData)
54              .environmentObject(subjectData)
55              .environmentObject(attendanceData)
56              .environmentObject(theme)
57              .task {
58                  await studentData.fetchStudents()
59                  await subjectData.fetchSubjects()
60              }
61          }
```

Slika 5.5 Kod TabView pogleda

5.3. Glavni pogled

MainView je glavna komponenta aplikacije koja omogućuje prikaz i obradu podataka o nazočnosti studenata. U sučelju korisnici mogu putem izbornika odabrati godinu i predmet, prikazati popis studenata, te izračunati i izvesti postotke nazočnosti studenata. Elementi su raspoređeni putem VStack i HStack struktura za organizaciju sadržaja. Glavne funkcionalnosti uključuju izračun postotka nazočnosti, filtriranje studenata i izvoz podataka u CSV format.

Slika 5.6 MainView pogled

Slika 5.7 MainView pogled s prikazom nazočnosti za kolegij Matematiku 1

5.3.1. Funkcija za izračun postotka nazočnosti

Funkcija calculateAttendancePercentage iz isječka 5.8 izračunava postotak nazočnosti studenta na odabranom predmetu. Prvo izračunava broj ukupnih predavanja, a zatim filtrira predavanja na kojima je student bio prisutan. Ako nije bilo predavanja, vraća 0 %. Na kraju vraća postotak nazočnosti kao omjer prisutnih predavanja u odnosu na ukupni broj.

```swift
func calculateAttendancePercentage(for student: Student, in subject: Subject) ->
    Double {
    let totalClasses = subject.attendanceDates.count

    // If there are no classes scheduled, return 0%
    if totalClasses == 0 {
        return 0.0
    }

    // Count attended classes
    let attendedClasses = subject.attendanceDates.filter { date in
        student.attendanceRecords.contains(where: { $0 == date })
    }.count

    return (Double(attendedClasses) / Double(totalClasses)) * 100.0
}
```

Slika 5.8 Kod funkcije calculateAttendancePercentage

5.3.2. Funkcija za izvoz podataka u CSV

U prikazu 5.9 vidimo funkciju exportToCSV koja generira CSV datoteku s podacima o studentima. Podaci uključuju ID, ime, prezime i postotak nazočnosti. CSV string se formatira i zapisuje u privremenu direktoriju, a zatim se pomoću funkcije shareCSV korisniku omogućuje dijeljenje datoteke putem aplikacija poput e-pošte ili AirDropa. Spremljenu datoteku je moguće otvoriti s aplikacijom Microsoft Excel.

```swift
func exportToCSV() {
    // Generate CSV string
    var csvString = "ID,First Name,Last Name,Attendance Percentage\n"
    for student in studentExports {
        let formattedPercentage = String(format: "%.2f%%",
            student.attendancePercentage) // Format the percentage with two decimals
            and add a %
        csvString +=
            "\(student.id),\(student.firstName),\(student
            .lastName),\(formattedPercentage)\n"
    }

    // Create CSVDocument and trigger file export
    let csvDocument = CSVDocument(data: csvString)
    let fileURL =
        FileManager.default.temporaryDirectory.appendingPathComponent("attendance.csv")

    do {
        try csvString.write(to: fileURL, atomically: true, encoding: .utf8)
        shareCSV(url: fileURL)
    } catch {
        print("Error writing CSV file: \(error)")
    }
}
```

Slika 5.9 Kod funkcije exportToCSV

5.3.3. Filtriranje i prikaz studenata

Dio koda 5.10 prikazuje studente koji su upisani na odabrani predmet. Prvo filtrira studente prema predmetu, a zatim ih sortira prema prezimenu. Prikazuje se ime, prezime, postotak nazočnosti i

profilna slika svakog studenta. Ako predmet nije odabran, korisniku je prikazana poruka da treba odabrati predmet kako bi vidio podatke o nazočnosti.

```
142                 if !selectedSubject.isEmpty, let subject = subjectData.subjects.first(where: { $0.name == selectedSubject }) {
143                     List(studentData.students
144                         .filter { subject.students.contains($0.id) }  // Filter students by subject
145                         .sorted(by: { $0.lastName < $1.lastName })) { student in
146
147                             let attendancePercentage = calculateAttendancePercentage(for: student, in: subject)
148
149                             HStack {
150                                 AsyncImage(url: student.profileImage) { image in
151                                     image.resizable()
152                                         .scaledToFit()
153                                         .frame(width: 50, height: 50)
154                                         .clipShape(Circle())
155                                 } placeholder: {
156                                     // Placeholder while the image is loading
157                                     Circle()
158                                         .fill(Color.gray)
159                                         .frame(width: 50, height: 50)
160                                 }
161                                 Spacer()
162                                 VStack {
163                                     Text("\(student.firstName)")
164                                         .foregroundStyle(theme.isDark ? .white : .black)
165                                     Text("\(student.lastName)")
166                                         .foregroundStyle(theme.isDark ? .white : .black)
167                                 }
168                                 Spacer()
169                                 Text("\(attendancePercentage, specifier: "%.2f")%")
170                                     .foregroundColor(attendancePercentage < 70 ? .red : .green)
171                             }
```

Slika 5.10 Kod za filtriranje i prikaz studenata u MainView pogledu

5.4. Dodavanje predavanja

5.4.1. Pogled za prijavu administratora

Pogled za autentifikaciju u AddLectureView je jednostavan sustav za prijavu koji se aktivira čim korisnik otvori ovaj pogled. Korisnik mora unijeti svoje korisničko ime i lozinku kako bi dobio pristup glavnim funkcionalnostima dodavanja predavanja i upravljanja prisutnošću studenata. Autentifikacija je realizirana pomoću dva osnovna elementa: TextField za unos korisničkog imena i SecureField za unos lozinke. Nakon unosa podataka, korisnik pritišće gumb "Login". Ako su korisničko ime i lozinka ispravni korisnik se uspješno autentificira i dobiva pristup glavnim funkcionalnostima. U suprotnom, aplikacija prikazuje upozorenje u obliku Alert prozora s porukom "Authentication Failed", te obavještava korisnika da su uneseni podaci netočni.

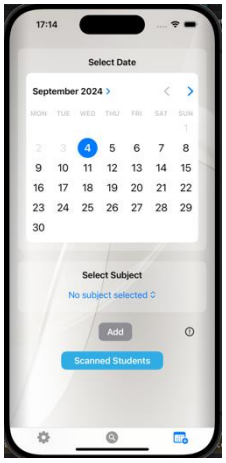
Slika 5.11 Admin Login pogled za pristup AddLectureView pogledu

5.4.2. Pogled za dodavanje

AddLectureView je ključan pogled unutar aplikacije koji profesorima omogućava dodavanje predavanja i praćenje prisutnosti studenata. U ovom pogledu, korisnik prvo mora proći kroz autentifikaciju pomoću jednostavnog login sustava, gdje unosi korisničko ime i lozinku. Nakon uspješne prijave, otvara se glavni dio pogleda s nekoliko važnih funkcionalnosti.

Prvi dio prikazuje komponentu DatePicker, koja omogućava profesorima odabir datuma za novo predavanje. Datum mora biti budući, čime se sprječava kreiranje predavanja u prošlosti. Nakon toga, korisnik bira predmet putem Picker izbornika, gdje su svi dostupni predmeti prikazani iz baze podataka. Tek nakon odabira datuma i predmeta, profesor može dodati predavanje pritiskom na gumb "Add".

Ako korisnik pokuša dodati predavanje bez odabira predmeta, gumb za dodavanje je onemogućen, a ako odabrani datum već postoji, pojavljuje se upozorenje.



Slika 5.12 AddLectureView pogled

Kada korisnik pritisne gumb, prvo se provjerava je li taj datum već zauzet za drugi predmet. Ako je datum već zauzet, postavlja se varijabla showingAlert na true kako bi se prikazala obavijest o konfliktu termina.

Ako datum nije zauzet, u asinkronom zadatku dohvaća se posljednja evidencija nazočnosti. Zatim se dodaje novi termin u evidenciju kolegija i. ažuriraju se podaci o studentima koji su nazočili predavanju. Kod ovih funckionalnosti je prikazan u isječku 5.13.

```
64                  Button(action: {
65                      let adjustedDate = Calendar.current.date(byAdding: .day, value: 1, to: selectedDate) ?? selectedDate
66
67                      let dateExists = subjectData.subjects.contains { subject in
68                          subject.attendanceDates.contains { existingDate in
69                              Calendar.current.isDate(existingDate, inSameDayAs: adjustedDate)
70                          }
71                      }
72
73                      if dateExists {
74                          showingAlert = true
75                      } else if !selectedSubject.isEmpty {
76                          Task {
77                              await attendanceData.fetchAttendance()
78                              await attendanceData.clearDatabase()
79
80                              if let lastAttendance = attendanceData.attendances.last {
81                                  lastAttendance.attendanceDate = adjustedDate
82                                  lastAttendance.subjectName = selectedSubject
83
84                                  for index in subjectData.subjects.indices {
85                                      if subjectData.subjects[index].name == selectedSubject {
86                                          subjectData.subjects[index].addAttendanceDate(adjustedDate)
87                                          await subjectData.updateSubject(subject: subjectData.subjects[index])
88                                          break
89                                      }
90                                  }
91
92                                  for studentIndex in studentData.students.indices {
93                                      if lastAttendance.IDs.contains(studentData.students[studentIndex].id) {
94                                          studentData.students[studentIndex].addAttendanceRecord(adjustedDate)
95                                          await studentData.updateUser(student: studentData.students[studentIndex])
96                                      }
97                                  }
98                              }
99                          }
100                     }
```

Slika 5.13 Kod gumba za dodavanje predavanja

5.4.3. Pogled za pregled prisutnih studenata

Dodatno, postoji i pogled za prikaz skeniranih studenata putem NFC-a. Nakon skeniranja, profesoru je omogućeno da putem gumba "Scanned Students" vidi listu studenata koji su skenirani. Prikazuju se njihova imena, prezimena i profilne slike.

Slika 5.14 ScannedStudentsView pogled

5.5. Pogled podešavanja

Pogled SettingsView prikazuje korisniku popis opcija unutar aplikacije koje omogućuju pristup dodatnim funkcijama i informacijama. U ovom pogledu koristi se NavigationView kako bi se omogućilo jednostavno kretanje između različitih dijelova aplikacije. Na popisu se nalaze stavke kao što su About, Accessibility, Patch Notes te Subject Lectures, koje vode do detaljnijih pregleda svakog od tih dijelova.

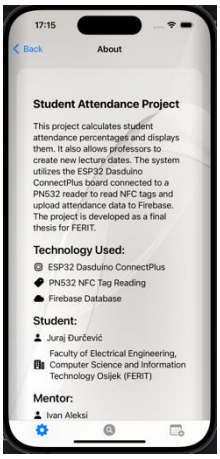
Korisnik također može uključiti ili isključiti određene opcije pomoću ugrađenog prekidača, poput opcije za upravljanje načinom prikaza. Ako je prekidač uključen vizualni aspekt aplikacije postaje tamne teme, a inače je svijetla tema.

Slika 5.15 SettingsView pogled

5.5.1. Ostali pogledi

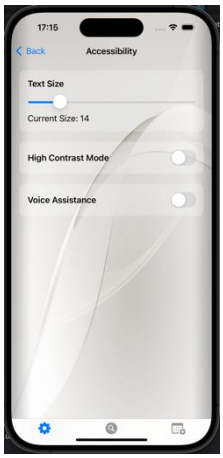
U pogledu SettingsView nalaze se sljedeći podpogledi:

- About: Ovaj podpogled prikazuje osnovne informacije o projektu, uključujući opis aplikacije, korištenu tehnologiju i podatke o autoru i mentoru.



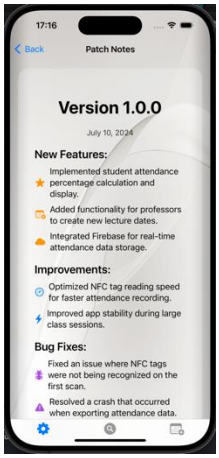
Slika 5.16 AboutView pogled

- Accessibility: Iako je ovaj podpogled vizualno dizajniran, trenutno ne sadrži implementirane funkcionalnosti. Služi kao prikaz mogućnosti za buduće prilagodbe pristupačnosti, poput podešavanja veličine teksta, kontrasta i glasovne pomoći.



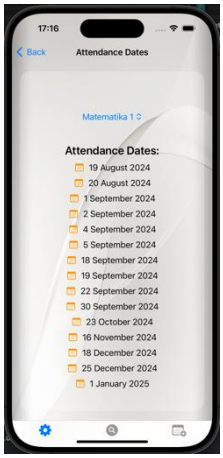
Slika 5.17 AccessibilityView pogled

- Patch Notes: Prikazuje bilješke o promjenama i nadogradnjama unutar aplikacije, uključujući nove funkcionalnosti, poboljšanja i ispravke grešaka.



Slika 5.18 PatchNotesView pogled

- Subject Lectures: Omogućuje korisniku pregled predavanja unutar odabranih kolegija, s popisom datuma održavanja predavanja.



Slika 5.19 SubjectLecturesView pogled

6. PROBLEMI I IZAZOVI U RADU

Tijekom razvoja aplikacije za praćenje nazočnosti studenata, identificirano je nekoliko tehničkih ograničenja koja utječu na funkcionalnost i upotrebljivost sustava. Jedan od ključnih problema je nemogućnost održavanja predavanja različitih kolegija u istom terminu. Aplikacija trenutno dopušta samo jedan kolegij po određenom terminu, što znači da, ako je za određeni datum već zabilježeno predavanje za jedan kolegij, nije moguće dodati predavanje za drugi kolegij na isti datum. Ovo ograničenje predstavlja izazov u situacijama kada se više predavanja odvija istovremeno ili kada je potrebno zabilježiti nazočnost studenata iz različitih kolegija u isto vrijeme.

Drugi značajan problem je tehnička nekompatibilnost DASDUINO ESP32 ConnectPlus pločice s bibliotekom za prijenos podataka na Firebase bazu podataka. Iako je DASDUINO ESP32 uspješno integriran s PN532 NFC čitačem i omogućava očitavanje MIFARE kartica, podaci s očitanih kartica, poput jedinstvenog ID-a, ne mogu se automatski poslati u Firebase bazu podataka zbog nekompatibilnosti biblioteka. Zbog ovog ograničenja, podaci su morali biti ručno uneseni u bazu kako bi se simulirao rad aplikacije i omogućila daljnja testiranja. Ovaj problem značajno utječe na automatizaciju procesa i pouzdanost sustava.

Za buduća poboljšanja aplikacije, preporučuje se korištenje alternativnih ESP32 pločica koje su potpuno kompatibilne s bibliotekama potrebnim za integraciju s Firebase-om. Time bi se omogućio automatski prijenos podataka s NFC čitača u bazu podataka, što bi sustav učinilo učinkovitijim i pouzdanijim u stvarnim uvjetima korištenja. Također bi trebalo razmotriti implementaciju opcije koja omogućuje bilježenje nazočnosti za više kolegija istovremeno, čime bi se povećala fleksibilnost i prilagodljivost sustava.

7. ZAKLJUČAK

Zaključak završnog rada pruža osvrt na postavljene ciljeve i ostvarene rezultate vezane za razvoj mobilne iOS aplikacije za evidenciju nazočnosti studenata s pomoću RFID/NFC čitača iksica. Primarni cilj rada bio je razviti funkcionalni sustav koji omogućava praćenje i analizu nazočnosti studenata na predavanjima. Uspješno je implementirana aplikacija koja prikazuje osnovne podatke o studentima, kao što su ime, prezime, slika i e-mail, te izračunava postotak njihove nazočnosti. Također je omogućeno pohranjivanje i upravljanje podacima putem Firebase baze podataka, uz mogućnost izvoza podataka u Excel.

Jedna od prednosti ovog rješenja je intuitivno korisničko sučelje koje omogućuje lako upravljanje predavanjima i evidencijom nazočnosti. Međutim, tijekom razvoja uočena su i određena ograničenja. Na primjer, u trenutnoj implementaciji nije moguće organizirati predavanja različitih kolegija u istom terminu, što smanjuje fleksibilnost aplikacije. Nadalje, zbog nekompatibilnosti Dasduino ESP32 ConnectPlus pločice s Firebase bibliotekom, skeniranje MIFARE kartica radi, ali podaci se ne mogu automatski poslati na bazu podataka, što je simulirano ručnim unosom ID-ova.

Za budući razvoj, preporučuje se rješavanje problema s hardverom korištenjem kompatibilnijih ESP32 pločica, kao i proširenje funkcionalnosti za rad s više kolegija istovremeno. To bi omogućilo širu primjenu ovog sustava na drugim obrazovnim institucijama, gdje je potrebna pouzdana evidencija nazočnosti studenata.

LITERATURA

[1] “Easy Attendance App” [online], dostupno na: https://easyattendance.app/ [Pristupljeno: 3.9.2024.].

[2] “Jibble App” [online], dostupno na: https://www.jibble.io/ [Pristupljeno: 3.9.2024.].

[3] “Dasduino ESP32 Connectplus” [online], dostupno na: https://soldered.com/product/dasduino-connectplus/ [Pristupljeno: 3.9.2024.].

[4] “PN532 NFC/RFID Controller” [online], dostupno na: https://soldered.com/product/pn532-nfc-and-rfid-controller-2x-tag/ [Pristupljeno: 3.9.2024.].

[5] “EasyC adapter” [online], dostupno na: https://soldered.com/product/easyc-adapter/ [Pristupljeno: 3.9.2024.].

[6] “What is Arduino” [online], dostupno na: https://www.arduino.cc/en/Guide/Introduction [Pristupljeno: 4.9.2024.].

[7] “Apple Swift Documentation“ [online], dostupno na: https://developer.apple.com/documentation/swift/ [Pristupljeno: 4.9.2024.].

[8] “Firebase Documentation” [online], dostupno na: https://firebase.google.com/docs [Pristupljeno: 4.9.2024.].

[9] “Adafruit PN532 Arduino biblioteka” [online], dostupno na: https://github.com/adafruit/Adafruit-PN532/blob/master/Adafruit_PN532.h [Pristupljeno: 4.9.2024.].

[10] “Firebase ESP Client Arduino biblioteka” [online], dostupno na: https://github.com/mobizt/Firebase-ESP-Client [Pristupljeno: 4.9.2024.].

[11] “Arduino Documentation” [online], dostupno na: https://docs.arduino.cc/programming/ [Pristupljeno: 4.9.2024.].

SAŽETAK

Ovaj završni rad bavi se razvojem mobilne iOS aplikacije za evidenciju nazočnosti studenata s pomoću RFID/NFC čitača iksica. Glavni problem koji se rješava je digitalizacija evidencije nazočnosti, pri čemu je korišten Dasduino ESP32 ConnectPlus mikrokontroler u kombinaciji s PN532 NFC čitačem za očitavanje MIFARE kartica studenata. Podaci o nazočnosti pohranjuju se u Firebase bazu podataka, a aplikacija omogućuje pregled osnovnih informacija o studentima te izračun postotka njihove nazočnosti na predavanjima. Implementirano je i izvoz podataka u Excel tablicu. Glavni postignuti rezultat je funkcionalna aplikacija koja olakšava profesorima evidenciju i analizu nazočnosti studenata. Neka ograničenja sustava uključuju nemogućnost održavanja više predavanja istovremeno te tehničke poteškoće s pločicom Dasduino ESP32, koje je moguće unaprijediti u budućim verzijama.

Ključne riječi: aplikacija, evidencija, iOS, NFC

ABSTRACT

Title: Mobile iOS application for student attendance records using RFID/NFC reader

This final thesis focuses on the development of a mobile iOS application for student attendance tracking using RFID/NFC card readers. The main problem addressed is the digitization of attendance records, with the system utilizing the Dasduino ESP32 ConnectPlus microcontroller in combination with a PN532 NFC reader to scan student MIFARE cards. Attendance data is stored in the Firebase database, and the application provides an overview of basic student information along with the calculation of their attendance percentage. Data export to Excel spreadsheets is also implemented. The main result achieved is a functional application that simplifies the process of attendance tracking and analysis for professors. Some limitations include the inability to manage multiple courses simultaneously and technical challenges with the Dasduino ESP32 board, which could be improved in future versions.

Keywords: application, attendance, iOS, NFC