Техничка документација: Students Attendance Management System

1. Вовед

Student Attendance Management System (SAMS) претставува напредна, високо-софистицирана веб платформа наменета за целосна дигитална трансформација на процесот на евиденција на студентското присуство. Системот е архитектонски поставен врз најновата **ASP.NET Core MVC 9.0** рамка, имплементирајќи **Onion Architecture**. Овој архитектонски стил го поставува доменскиот модел во самиот центар на апликацијата, обезбедувајќи независност од надворешните слоеви, висока тестирабилност и лесно одржување на кодот.

SAMS е дизајниран да функционира како **автономен екосистем** кој драстично ја намалува потребата од мануелна администрација преку интелигентни алгоритми за управување со времето и просторот. За разлика од конвенционалните системи, SAMS интегрира проактивни механизми за безбедност и валидација, гарантирајќи дека секој запис за присуство е веродостоен и физички потврден.

2. Технички спецификации и работна околина

За реализација на SAMS е избран модерен и робустен технолошки стек кој гарантира високи перформанси, но пред сè – **интероперабилност**. Системот е дизајниран со претпоставка дека ќе биде дел од поголем универзитетски екосистем, а не изолирана апликација.

2.1 Технички спецификации

- **Backend Технологии:**
  - **Framework: .NET 9.0** (ASP.NET Core MVC)
  - **Архитектура: Onion Architecture**
- **Стратегија за База на Податоци):**
  - **ORM: Entity Framework Core 9.0.11**.
  - **Избор на база:** За потребите на развој и тестирање на функционалностите, апликацијата користи **SQLite**.
  - **Образложение:** Во реален, продукциски амбиент, секоја образовна институција веќе поседува свои централизирани бази на податоци со

регистри на студенти и професори (на пр. MS SQL Server, Oracle, PostgreSQL).
    - **Интеграција:** SAMS е намерно дизајниран да **не креира нови изолирани податоци**, туку да се **поврзе (mount)** врз постоечката инфраструктура на институцијата. SQLite базата во овој проект служи исклучиво за симулација на податочниот слој за време на фазата на тестирање, докажувајќи дека логиката на апликацијата функционира беспрекорно пред да се направи поврзувањето со реалниот систем.
- **Frontend Интерфејс:**
    - **Razor Views & Bootstrap**
    - **JQuery Validation Unobtrusive:** За брза клиентска валидација.
- **Клучни Библиотеки:**
    - QRCoder: Специјализирана библиотека за брзо генерирање на QR кодови во меморија (без непотребно зачувување слики на диск).

2.2 Системски Барања (System Requirements)

За успешно инсталирање и извршување на системот во оваа тест фаза:

- **Оперативен Систем:** Windows, Linux, или macOS.
- **Software Development Kit: .NET 9.0 SDK**.
- **Иднина:** При имплементација, потребен е само Connection String до постоечката база на институцијата.

3. Архитектура на проектот

SAMS е дизајниран следејќи ги принципите на **Onion Architecture.** Овој пристап е избран за да се обезбеди долготрајност на софтверот и лесно одржување. За разлика од традиционалните n-tier архитектури кои зависат од базата на податоци, во SAMS **Доменскиот модел (Domain Model)** е во центарот, а сите останати слоеви (сервиси, UI, база) зависат од него.

3.1 Доменски Слој (Domain Layer)

Овој слој нема никакви надворешни зависности. Тука се дефинирани ентитетите кои го моделираат реалниот академски процес.

- **Клучни Ентитети:** Student, Professor, Course, Session и Attendance.
- **Enumerations:** Тука се дефинирани фиксните типови како SemesterType (Winter/Summer) и SessionStatus (Active/Finished/Expired), кои обезбедуваат типизирана контрола на логиката.

3.2 Репозиториумски Слој (Repository / Persistence Layer)

Овој слој служи како мост помеѓу апликацијата и базата на податоци (SQLite/SQL Server).

- **Generic Repository Pattern:** Универзален механизам за CRUD операции, што го намалува повторувањето на кодот.
- **ApplicationDbContext:** Ја поврзува апликацијата со базата, управувајќи со релациите помеѓу табелите (на пр. кој професор кој предмет го предава).

3.3 Сервисен Слој (Service Layer - Business Logic)

Ова е слојот на бизнис логичатс на SAMS. Сите проверки се случуваат тука.

- **AttendanceService:** Го валидира секое скенирање на QR код.
- **SessionService:** Управува со тајмерите за часовите и валидацијата на IP адресите.
- **QrCodeService:** Генерира криптирани податоци за QR сликите.
- **ReportService:** Ги претвора податоците во статистички извештаи.

3.4 Презентациски Слој (Web / Presentation Layer)

Ова е надворешната обвивка со која директно комуницираат корисниците. Овој слој е одговорен за **визуелизација** на податоците и прифаќање на корисничките наредби.

- **Controllers (Контролери):** Ова се „сообраќајците“ на апликацијата. Тие ги примаат HTTP барањата (на пр. GET /Professor/Dashboard) и одлучуваат кој сервис треба да се повика.
    - Пример: SessionController, AccountController, ProfessorController.
- **Views (Погледи / HTML Страници):** Ова е корисничкиот интерфејс (UI).
    - **Што претставуваат:** Тоа се .cshtml датотеки кои содржат **HTML код** комбиниран со C# (Razor синтакса).
    - **Функција:** Кога корисникот ќе побара некоја страница, серверот го зема соодветниот View, ги пополнува податоците (на пр. име на студент, листа на предмети) и го испраќа назад како **чист HTML документ** кој прелистувачот го прикажува.
    - **Стилизација:** Сите Views се стилизирани со **Bootstrap**, што ги прави страниците (копчиња, табели, менија) модерни и прилагодливи за мобилни телефони.

4. Функционална спецификација на системот

Оваа секција ги дефинира клучните модули и деловните правила кои го прават SAMS интелигентен и автономен систем.

4.1 Хиерархиска Администрација и Улоги

Системот функционира врз основа на строга поделба на привилегиите за да се спречи хаос во распоредот.

- **Администратор (Super-User):**
  - Има ексклузивно право да го дефинира „скелетот“ на семестарот.
  - Креира нови предмети, доделува професори и ги брише старите курсеви.
  - Го поставува почетниот датум на семестарот
- **Професор:**
  - Не може да менува глобални поставки, туку само управува со своите доделени предмети.
  - Има автономија да управува со тековниот час (сесија), но мора да се придржува до глобалниот распоред.

4.2 Паметен Распоред и Детекција на Конфликти

SAMS користи алгоритам за валидација на времето и просторот за да спречи преклопување на наставата.

- **Логика на закажување:** Кога професорот или администраторот креира распоред за предметот, системот проверува во базата дали тој термин (ден и време) е веќе зафатен од друг предмет во истата просторија или за истиот професор.
- **Систем за Нотификации:** Доколку се детектира поклопување (на пр. Професорот се обидува да закаже час во Понеделник 10:00, а тој термин е зафатен), системот веднаш враќа **грешка/нотификација** и не дозволува креирање на сесијата додека не се избере слободен термин.
- **Автоматска промена на семестар:** Системот препознава дали тековниот датум спаѓа во Зимски или Летен семестар и соодветно го прилагодува приказот на предметите.

4.3 Автоматизирана Временска Проекција

Оваа функција драстично ја намалува мануелната работа на професорите.

- **Принцип „Еден клик“:** Професорот треба да го дефинира само **првиот час** во семестарот.

- **Проекција:** Откако ќе се стартува првиот час, системот автоматски ги генерира и визуелизира датумите за сите наредни 14-15 недели од семестарот. Ова овозможува лесна навигација низ неделите без потреба рачно да се внесува секој датум посебно.

4.4 Безбедност: Динамични QR Кодови и IP Заштита

За да се гарантира физичкото присуство, имплементирани се два слоја на заштита:

1. **Динамичен QR Код:**
    - QR кодот што се прикажува на проекторот не е статична слика. Тој содржи криптиран токен кој е валиден само ограничено време (на пр. 10 секунди).
    - Откако ќе истече времето или сесијата ќе се затвори, кодот станува неупотреблив, спречувајќи студентите да го сликаат и испратат на колеги кои не се присутни.
2. **IP Whitelisting (Мрежна Бариера):**
    - Системот дозволува конфигурација на дозволени IP адреси.
    - Професорот може да подеси најавата да биде можна **само** ако студентот е поврзан на универзитетската Wi-Fi мрежа. Обид за најава од домашна мрежа или мобилен интернет (4G/5G) ќе биде одбиен, дури и ако QR кодот е валиден.

4.5 Мониторинг во Реално Време (Live Dashboard)

- Кога сесијата е активна, професорот гледа табела која се ажурира во живо.
- Достапни се опции за **продолжување на времето** (ако има технички проблеми) или **предвремено затворање** на сесијата.

5. Упатство за користење

5.1 За Професори

Чекор 1: Најава и Почетен Екран

- Внесете го вашето корисничко име и лозинка.
- На почетниот екран веднаш ќе го видите предметот што е на ред според распоредот.

Чекор 2: Стартување на Час (Live Session)

- Кликнете на предметот и изберете **"Start Session"**.

- На екранот ќе се појави голем QR код. Овој екран треба да биде видлив за сите студенти (преку проектор).

Чекор 3: Следење и Затворање

- Следете ја листата на присутни на вашиот екран во реално време.
- Кога сметате дека сите присутни се пријавиле, кликнете на копчето **"Close Session"**. Со ова, QR кодот станува невалиден и списокот се заклучува.

Чекор 4: Извештаи

- Во секое време можете да пристапите до менито "Reports" за да видите статистика за присуството (оптимизирано за преглед од мобилен телефон).

5.2 За Студенти

Процесот за студентите е дизајниран да биде брз и безбеден:

Чекор 1: Скенирање

- Скенирајте го QR кодот од проекторот со камерата на вашиот телефон.

Чекор 2: Најава (Автентикација)

- По скенирањето, телефонот ќе ве однесе на страницата за најава (доколку веќе не сте најавени).
- Внесете го вашето корисничко име и лозинка.

Чекор 3: Системска Валидација и Порака

Веднаш по најавата, системот прави проверки (дали кодот е активен, дали сте на Eduroam мрежа) и ви прикажува една од следниве пораки:

- **SUCCESS:** "Successfully checked in." (Успешно евидентирано присуство).
- **ERROR - EXPIRED:** "QR Code has expired." (Сесијата е веќе затворена од професорот).
- **ERROR - NETWORK:** "Access Denied. Please connect to **Eduroam**." (Мора да сте поврзани на универзитетската Wi-Fi мрежа).

6. Клучни конфигурации

За да се овозможи максимална безбедност и контрола, клучните параметри на системот се дефинирани директно во кодот. Оваа секција им служи на developers да ги лоцираат и променат овие поставки според потребите на мрежната инфраструктура.

6.1 Времетраење на сесијата (Session Duration)

Стандардната должина на еден час (прозорец за најава) е дефинирана во ентитетот на сесијата.

- **Локација:** *AttendanceStudents.Domain/Entities/Session.cs*
- **Променлива:** `LoginWindowSeconds`
- **Опис:** Тука се менува колку секунди ќе трае сесијата пред автоматски да истече.

```
// Пример: 300 секунди (5 минути). Променете го овој број за
подолги/пократки часови.
public int LoginWindowSeconds { get; set; } = 300;
```

6.2 Ограничување на пристап по IP адреса (IP Whitelisting)

Заштитата која спречува најава од дома е имплементирана во сервисниот слој.

- **Локација:** `AttendanceStudents.Service/Implementations/AttendanceService.cs`
- **Метод:** `IsAllowedForHomeTest`
- **Опис:** Оваа функција проверува дали IP адресата на студентот припаѓа на дозволениот опсег (CIDR). Доколку сакате системот да работи само на **Eduroam**, тука треба да се внесат јавните IP опсези на универзитетската мрежа.

```
private static bool IsAllowedForHomeTest(IPAddress ip)
{
    // Дозволи локален пристап (localhost)
    if (IPAddress.IsLoopback(ip)) return true;

    // Ограничување на мрежата (на пр. само уреди поврзани на
192.168.1.x)
    return IsInCidr(ip, "192.168.1.0/25");
}
```

6.3 Конфигурација на QR URL и База (Base URL)

Генерирањето на линкот што се крие во QR кодот се прави во сервисот за сесии.

- **Локација:** `AttendanceStudents.Service/Implementations/SessionService.cs`
- **Променлива:** `joinUrl`
- **Опис:** Тука се дефинира адресата на серверот. За продукција, треба да се одкоментира динамичкиот дел, а за тестирање во локална мрежа се користи фиксна IP адреса.

```
// ЗА ТЕСТИРАЊЕ (Фиксна IP на хост машината):
var joinUrl =
$"http://192.168.1.146:5035/Attendance/Join?sessionId={session.Id}&code
={rawCode}";


// ЗА ПРОДУКЦИЈА (Динамички URL):
// var joinUrl =
$"{baseUrl}/Attendance/Join?sessionId={session.Id}&code={rawCode}";
```

6.4 Времетраење и ротација на QR кодот

Безбедноста на QR кодот зависи од тоа колку често тој се менува (освежува).

- **Локација:** `AttendanceStudents.Service/Implementations/SessionService.cs`
- **Метод:** `GetLiveInfo`
- **Опис:** Оваа линија одредува колку секунди важи еден генериран QR код пред да стане невалиден.

```
// Кодот е валиден 50 секунди. Намалете го за поголема безбедност.
session.CodeValidUntil = now.AddSeconds(50);
```

6.5 Освежување на професорскиот екран (Frontend Refresh)

Брзината со која се ажурира листата на присутни кај професорот.

- **Локација:** `AttendanceStudents.Web/Views/Session/Details.cshtml`
- **Скрипта:** `setInterval`
- **Опис:** Оваа JavaScript функција го контролира интервалот на повикување на серверот за нови податоци.

```
// Освежување на QR кодот и бројчаникот на секои 50000ms (50 секунди)
setInterval(refresh, 50000);
```

7. Идни планови за развој

Иако моменталната верзија на SAMS е целосно функционална, архитектурата е поставена така што овозможува лесна надградба со нови модули. Клучни приоритети за следната верзија се:

8. **Автоматизација на Распоредот:** Елиминирање на рачното додавање на часот и денот за предметите. Системот ќе се надгради да повлекува податоци директно од централниот административен софтвер на факултетот, со што распоредот ќе се пополнува автоматски уште пред почетокот на семестарот.
9. **Напредна Биометриска Сигурност (FaceID):** Воведување на дополнителен слој на заштита каде студентот ќе мора да помине низ биометриска верификација (препознавање на лице или отпечаток) на својот телефон пред да добие дозвола да го скенира QR кодот. Ова ќе ја елиминира можноста еден студент да носи телефони на други колеги.
10. **Нативна Мобилна Апликација:** Развој на Android и iOS апликација која ќе овозможи *Push Notifications* потсетници пред час.
11. **Интеграција со LMS:** Поврзување со Moodle или Teams за автоматски пренос на податоците за присуство.