ДОДАТОК А

Програмний код API для розроблюваної системи

Лістинг А.1 – Налаштування «Program.cs»

```
var builder = WebApplication.CreateBuilder(args);

builder.Configuration.AddJsonFile(
    "appDescription.json",
    optional: true,
    reloadOnChange: true);

var swaggerConfig = builder.Configuration.GetSection("SwaggerConfig").Get<SwaggerConfigModel>();

builder.Services.AddControllers().AddNewtonsoftJson();

builder.Services.AddEndpointsApiExplorer();

builder.Services.Configure<SmtpSettings>(builder.Configuration.GetSection("SmtpSettings"));
builder.Services.Configure<InvitationSettings>(builder.Configuration.GetSection("InvitationSettings"));
builder.Services.Configure<QrCodeInfo>(builder.Configuration.GetSection("QrCodeInfo"));

builder.Services.AddSwaggerGen(options =>
{
   options.SwaggerDoc(swaggerConfig.Version, new OpenApiInfo
   {
       Title = swaggerConfig.Title,
       Version = swaggerConfig.Version,
       Description = swaggerConfig.Description
   });

   var xmlFilename = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
   options.IncludeXmlComments(Path.Combine(AppContext.BaseDirectory, xmlFilename));

   options.AddSecurityDefinition(JwtBearerDefaults.AuthenticationScheme, new OpenApiSecurityScheme
   {
       Name = "Authorization",
       Description = "JWT Authorization using the Bearer scheme",
       Type = SecuritySchemeType.Http,
       In = ParameterLocation.Header,
       Scheme = JwtBearerDefaults.AuthenticationScheme,
       BearerFormat = "JWT"
   });

   options.AddSecurityRequirement(new OpenApiSecurityRequirement
   {
       {
           new OpenApiSecurityScheme
           {
               Reference = new OpenApiReference
               {
```

```
                        Type = ReferenceType.SecurityScheme,
                        Id = JwtBearerDefaults.AuthenticationScheme
                    }
                }, new string[] {}
            }
        });
    }).AddSwaggerGenNewtonsoftSupport();

    builder.Services.AddDbContext<WorkQrDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("WorkQRConnection")));

    builder.Services.ServicesExtensions();
    builder.Services.AddHostedService<QrCodeCleanupService>();
    builder.Services.AddHostedService<InvitationCleanupService>();
    builder.Services.AddApiAuthentication(builder.Configuration);

    builder.Services.AddAutoMapper(AppDomain.CurrentDomain.GetAssemblies());

    builder.Services.AddFluentValidationAutoValidation();
    builder.Services.AddValidatorsFromAssemblyContaining<EmployeeUpdateValidator>();

    builder.Services.AddHttpContextAccessor();


    var app = builder.Build();

    if (app.Environment.IsDevelopment())
    {
        app.UseSwagger();
        app.UseSwaggerUI(options =>
        {
            options.SwaggerEndpoint($"/swagger/{swaggerConfig.Version}/swagger.json",
swaggerConfig.Version);
        });
    }

    app.UseMiddleware<ExceptionHandlerMiddleware>();
    app.UseHttpsRedirection();

    app.UseCors("DefaultCorsPolicy");

    app.UseAuthentication();
    app.UseAuthorization();

    app.MapControllers();

    app.Run();
```

Лістинг А.2 – Сутність «EmployeeEntity»

```
    public class EmployeeEntity
    {
        public Guid Id { get; set; }
        public string FirstName { get; set; } = string.Empty;
        public string Surname { get; set; } = string.Empty;
        public string Email { get; set; } = string.Empty;
        public string PasswordHash { get; set; } = string.Empty;
```

```
    public ICollection<RoleEntity> Roles { get; set; } = [];

    public ICollection<VisitRecordEntity> VisitRecords { get; set; } = [];


    public Guid JobPositionId { get; set; }
    public JobPositionEntity? JobPosition { get; set; }

    public ICollection<QrCodeEntity> QrCodes { get; set; } = [];
}
```

Лістинг А.3 – Сутність «InvitationInfoEntity»

```
public class InvitationInfoEntity
{
    public Guid Id { get; set; }
    public string InvitationCode { get; set; } = string.Empty;
    public DateTime CreatedAt { get; set; }
    public DateTime? ExpiresAt { get; set; }
    public bool IsActive { get; set; }

    public Guid InvitedEmployeeId { get; set; }
    public InvitedEmployeesEntity InvitedEmployee { get; set; }
}
```

Лістинг А.4 – Сутність «InvitedEmployeesEntity»

```
public class InvitedEmployeesEntity
{
    public Guid Id { get; set; }
    public string FirstName { get; set; } = string.Empty;
    public string Surname { get; set; } = string.Empty;
    public string Email { get; set; } = string.Empty;

    public InvitationInfoEntity? InvitationInfo { get; set; }
    public ICollection<RoleEntity>? Roles { get; set; } = [];
    public JobPositionEntity? JobPosition { get; set; }
}
```

Лістинг А.5 – Сутність «JobPositionEntity»

```
public class JobPositionEntity
{
    public Guid Id { get; set; }
    public string Name { get; set; } = string.Empty;

    public ICollection<EmployeeEntity>? Employees { get; set; }
    public ICollection<InvitedEmployeesEntity>? InvitedEmployees { get; set; }
}
```

Лістинг А.6 – Сутність «QrCodeEntity»

```
public class QrCodeEntity
{
    public Guid Id { get; set; }
    public Guid Code { get; set; }
    public DateTime ExpiryDate { get; set; }
    public bool IsDynamic { get; set; }

    public Guid EmployeeId { get; set; }
    public EmployeeEntity Employee { get; set; }
}
```

Лістинг А.7 – Сутність «RoleEntity»

```
public class RoleEntity
{
    public int Id { get; set; }
    public string Name { get; set; } = string.Empty;

    public ICollection<EmployeeEntity>? Employees { get; set; }
    public ICollection<InvitedEmployeesEntity>? InvitedEmployees { get; set; }
}
```

Лістинг А.8 – Сутність «VisitRecordEntity»

```
public class VisitRecordEntity
{
    public Guid Id { get; set; }
    public DateTime RecordDate { get; set; }
    public DateTime? TimeIn { get; set; }
    public DateTime? TimeOut { get; set; }

    public Guid EmployeeId { get; set; }
    public EmployeeEntity Employee { get; set; }

}
```

Лістинг А.9 – Обробник помилок «ExceptionHandlerMiddleware»

```
public class ExceptionHandlerMiddleware
{
    private const HttpStatusCode InternalServerError =
HttpStatusCode.InternalServerError;

    private readonly RequestDelegate _next;
    private readonly ILogger<ExceptionHandlerMiddleware> _logger;

    public ExceptionHandlerMiddleware(RequestDelegate next,
ILogger<ExceptionHandlerMiddleware> logger)
    {
```

```
            _next = next;
            _logger = logger;
        }

        public async Task InvokeAsync(HttpContext context)
        {
            try
            {
                await _next(context);
            }
            catch (Exception e)
            {
                await HandleExceptionAsync(e, context);
            }
        }

        private async Task HandleExceptionAsync(Exception exception, HttpContext
context)
        {
            _logger.LogError(exception.Message);

            context.Response.ContentType = MediaTypeNames.Application.Json;

            var errorModel = new ErrorModel();
            switch (exception)
            {
                case CustomException customException:
                    context.Response.StatusCode = (int)customException.StatusCode;
                    errorModel.ErrorMessage = customException.Message;
                    break;
                case ValidationException validationException:
                    context.Response.StatusCode = (int)HttpStatusCode.BadRequest;
                    errorModel.ErrorMessage = "Validation failed";
                    errorModel.Errors = validationException.Errors.Select(e => new
ValidationError
                    {
                        PropertyName = e.PropertyName,
                        ErrorMessage = e.ErrorMessage
                    }).ToList();
                    break;
                default:
                    context.Response.StatusCode = (int)InternalServerError;
                    errorModel.ErrorMessage = exception.Message;
                    break;
            }

            await context.Response.WriteAsJsonAsync(errorModel);
        }
    }
```

Лістинг А.10 – Батьківський клас для власних помилок «CustomException»

```
    public class CustomException : Exception
    {
        public HttpStatusCode StatusCode { get; set; }

        protected CustomException(HttpStatusCode statusCode, string errorMessage) :
base(errorMessage)
        {
            StatusCode = statusCode;
```

```
    }
}
```

Лістинг А.11 – Приклад налаштування «AutoMapper»

```
public class EmployeeMappingProfile: Profile
{
    public EmployeeMappingProfile()
    {
        CreateMap<EmployeeEntity, EmployeeDto>();
        CreateMap<EmployeeEntity, EmployeeUpdateDto>().ReverseMap();
        CreateMap<InvitedEmployeeDto, EmployeeEntity>();
        CreateMap<InvitedEmployeesEntity, EmployeeEntity>();
    }
}
```

Лістинг А.12 – Приклад використання «AutoMapper»

```
public async Task<ICollection<EmployeeDto>> GetListAsync()
    {
        var employees = await _dbContext.Employees
            .Include(emp => emp.JobPosition)
            .Include(emp => emp.Roles)
            .AsNoTracking()
            .ToListAsync();

        return _mapper.Map<ICollection<EmployeeDto>>(employees);
    }
```

Лістинг А.13 – Приклад налаштування валідації для «EmployeeLoginDto»

```
public class LoginValidator: AbstractValidator<EmployeeLoginDto>
{
    public LoginValidator()
    {
        RuleFor(employee => employee.Email)
            .IsRequired()
            .EmailAddress().WithMessage("Incorrect Email address");

        RuleFor(employee => employee.Password)
            .IsRequired()
            .Password();
    }
}
```

Лістинг А.14 – Інтерфейс «IemployeeService»

```
public interface IEmployeeService
{
    Task<ICollection<EmployeeDto>> GetListAsync();
```

```
        Task<EmployeeDto> GetByIdAsync(Guid employeeId);
        Task<EmployeeDto> GetByEmailAsync(string employeeEmail);
        Task CreateAsync(string invitationCode, EmployeeRegisterDto registerDto);

        Task<EmployeeDto>                  UpdateByIdAsync(Guid                  id,
JsonPatchDocument<EmployeeUpdateDto> employeeUpdatePatch);

        Task DeleteByIdAsync(Guid employeeId);

        Task   UpdatePasswordByIdAsync(Guid   employeeId,   UpdateEmployeePasswordDto
passwordDto);
    }
```

Лістинг А.15 – Клас «EmployeeService»

```
    public class EmployeeService: IEmployeeService
    {
        private readonly WorkQrDbContext _dbContext;
        private readonly IMapper _mapper;
        private readonly IInvitedEmployeeService _invitedEmployeeService;
        private readonly IPasswordAction _passwordAction;
        private readonly IValidator<EmployeeUpdateDto> _validator;
        private readonly ClaimsInfo? _currentEmployeeInfo;

        public EmployeeService(WorkQrDbContext dbContext,
            IMapper mapper,
            IInvitedEmployeeService invitedEmployeeService,
            IPasswordAction passwordAction,
            IValidator<EmployeeUpdateDto> validator,
            IHttpContextAccessor httpContextAccessor)
        {
            _dbContext = dbContext;
            _mapper = mapper;
            _invitedEmployeeService = invitedEmployeeService;
            _passwordAction = passwordAction;
            _validator = validator;
            _currentEmployeeInfo                                                    =
ParseInfoFromClaims(httpContextAccessor.HttpContext!);
        }

        public async Task<ICollection<EmployeeDto>> GetListAsync()
        {
            var employees = await _dbContext.Employees
                .Include(emp => emp.JobPosition)
                .Include(emp => emp.Roles)
                .AsNoTracking()
                .ToListAsync();

            return _mapper.Map<ICollection<EmployeeDto>>(employees);
        }

        public async Task<EmployeeDto> GetByIdAsync(Guid employeeId)
        {
            var employee = await _dbContext.Employees
                                .Include(emp => emp.JobPosition)
                                .Include(emp => emp.Roles)
                                .AsNoTracking()
                                .FirstOrDefaultAsync(e => e.Id == employeeId)
```

```
                            ?? throw new NotFoundException($"Employee with id:
{employeeId} was not found!");

                CheckPermissionAccess(employee.Id);

                return _mapper.Map<EmployeeDto>(employee);
            }

            public async Task<EmployeeDto> GetByEmailAsync(string employeeEmail)
            {
                var employee = await _dbContext.Employees
                                    .Include(emp => emp.JobPosition)
                                    .Include(emp => emp.Roles)
                                    .AsNoTracking()
                                    .FirstOrDefaultAsync(e => e.Email.ToLower() ==
employeeEmail.ToLower())
                            ?? throw new NotFoundException($"Employee with email:
{employeeEmail} was not found!");
                CheckPermissionAccess(employee.Id);

                return _mapper.Map<EmployeeDto>(employee);
            }

            public async Task CreateAsync(string invitationCode, EmployeeRegisterDto
registerDto)
            {
                var invitedEmployee = await GetByInvitationCodeAsync(invitationCode);

                var employee = _mapper.Map<EmployeeEntity>(invitedEmployee);

                employee.PasswordHash =
_passwordAction.HashPassword(registerDto.Password);

                await _dbContext.Employees.AddAsync(employee);
                await _dbContext.SaveChangesAsync();

                _dbContext.Remove(invitedEmployee);
                await _dbContext.SaveChangesAsync();
            }

            public async Task<EmployeeDto> UpdateByIdAsync(Guid id,
JsonPatchDocument<EmployeeUpdateDto> employeeUpdatePatch)
            {
                var employee = await _dbContext.Employees
                                    .Include(e => e.JobPosition)
                                    .Include(e => e.Roles)
                    .FirstOrDefaultAsync(emp => emp.Id == id)
                    ?? throw new NotFoundException($"Employee with id: {id} was not
found!");

                var employeeToPatch = _mapper.Map<EmployeeUpdateDto>(employee);
                employeeUpdatePatch.ApplyTo(employeeToPatch);

                if (!string.IsNullOrEmpty(employeeToPatch.Email) && employeeToPatch.Email
!= employee.Email)
                {
                    await ThrowIfEmailIsNotAvailableAsync(employeeToPatch.Email, id);
                }

                var validationResult = await _validator.ValidateAsync(employeeToPatch);
                if (!validationResult.IsValid)
                {
```

```
                throw new ValidationException(validationResult.Errors);
            }

            _mapper.Map(employeeToPatch, employee);
            await _dbContext.SaveChangesAsync();

            return _mapper.Map<EmployeeDto>(employee);
        }


        public async Task DeleteByIdAsync(Guid employeeId)
        {
            var employee = await _dbContext.Employees
                                .FirstOrDefaultAsync(e => e.Id == employeeId)
                            ?? throw new NotFoundException($"Employee with Id:
{employeeId} was not found!");

            _dbContext.Remove(employee);
            await _dbContext.SaveChangesAsync();
        }

        public async Task UpdatePasswordByIdAsync(Guid employeeId,
UpdateEmployeePasswordDto passwordDto)
        {
            var employee = await _dbContext.Employees
                                .FirstOrDefaultAsync(e => e.Id == employeeId)
                            ?? throw new NotFoundException($"Employee with Id:
{employeeId} was not found!");

            CheckPermissionAccess(employee.Id);

            await VerifyHashedPasswordAsync(passwordDto.CurrentPassword,
employee.PasswordHash);

            employee.PasswordHash = await Task.Run(() =>
_passwordAction.HashPassword(passwordDto.NewPassword));
            await _dbContext.SaveChangesAsync();

        }

        private async Task ThrowIfEmailIsNotAvailableAsync(string email, Guid id)
        {
            var isEmailExists = await _dbContext.Employees
                .AnyAsync(user => user.Email.ToLower() == email.ToLower() && user.Id
!= id);

            if (isEmailExists)
            {
                throw new BadRequestException("Employee with this email already
exists");
            }
        }

        private async Task VerifyHashedPasswordAsync(string password, string
hashedPassword)
        {
            var verificationPassword = await Task.Run(() => _passwordAction
                .VerifyPassword(password, hashedPassword));

            if (!verificationPassword)
                throw new BadRequestException("Current Password dont match");
```

```
        }

        private async Task<InvitedEmployeesEntity> GetByInvitationCodeAsync(string
invitationCode)
        {
            return await _dbContext.InvitedEmployees
                .Include(e => e.JobPosition)
                .Include(e => e.Roles)
                .Include(e => e.InvitationInfo)
                .FirstOrDefaultAsync(employee                                    =>
employee.InvitationInfo!.InvitationCode == invitationCode)
                ?? throw new NotFoundException(
                    $"Invited employee with this Invitation Code: {invitationCode} was
not found!");
        }

        private void CheckPermissionAccess(Guid id)
        {
            if (!(_currentEmployeeInfo.Role == Roles.Admin.ToString() || id ==
_currentEmployeeInfo.Id))
            {
                throw new ForbiddenException("You do not have permission!");
            }
        }
    }
```

Лістинг А.16 – Реєстрація створений сервісів

```
    public static void ServicesExtensions(this IServiceCollection services)
        {
            services.AddTransient<IPasswordAction, PasswordAction>();
            services.AddScoped<IInvitedEmployeeService, InvitedEmployeeService>();
            services.AddScoped<IJobPositionService, JobPositionService>();
            services.AddScoped<IRoleService, RoleService>();
            services.AddScoped<IInvitationService, InvitationService>();
            services.AddScoped<IEmployeeService, EmployeeService>();
            services.AddScoped<IAuthService, AuthService>();
            services.AddScoped<IJwtProvider, JwtProvider>();
            services.AddScoped<IQrCodeService, QrCodeService>();
            services.AddScoped<IVisitRecordService, VisitRecordService>();
        }
```

Лістинг А.17 – Метод «AddApiAuthentication»

```
    public static void AddApiAuthentication(this IServiceCollection services,
            IConfiguration configuration)
        {

            var                              jwtOptions                              =
configuration.GetSection("JwtOptions").Get<JwtOptions>();

            services.Configure<JwtOptions>(configuration.GetSection("JwtOptions"));

            services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
                .AddJwtBearer(JwtBearerDefaults.AuthenticationScheme, options =>
```

```
                {
                    options.TokenValidationParameters = new()
                    {
                        ValidateAudience = false,
                        ValidateIssuer = false,
                        ValidateLifetime = true,
                        ValidateIssuerSigningKey = true,
                        IssuerSigningKey                    =                    new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtOptions.SecretKey))
                    };

                });

            services.AddAuthorization();
            services.AddCors(options =>
            {
                options.AddPolicy("DefaultCorsPolicy", builder =>
                {
                    builder.AllowAnyOrigin()
                        .AllowAnyMethod()
                        .AllowAnyHeader();
                });
            });
        }
```

Лістинг А.18 – Клас «WorkQrDbContext»

```
    public class WorkQrDbContext : DbContext
    {
        public DbSet<EmployeeEntity> Employees { get; set; }
        public DbSet<InvitedEmployeesEntity> InvitedEmployees { get; set; }
        public DbSet<JobPositionEntity> JobPositions { get; set; }
        public DbSet<RoleEntity> Roles { get; set; }
        public DbSet<VisitRecordEntity> VisitRecords { get; set; }
        public DbSet<InvitationInfoEntity> InvitationInfo { get; set; }

        public DbSet<QrCodeEntity> QrCodes { get; set; }

        public    WorkQrDbContext(DbContextOptions<WorkQrDbContext>    options)    :
base(options) {}

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

modelBuilder.ApplyConfigurationsFromAssembly(typeof(WorkQrDbContext).Assembly);
            modelBuilder.SeedRoles();
            modelBuilder.SeedJobPosition();
        }
    }
```

ДОДАТОК Б

Програмний код та розмітка веб-інтерфейсу

Лістинг Б.1 – Розмітка файлу «App.razor»

```
<CascadingAuthenticationState>
    <Router AppAssembly="@typeof(App).Assembly">
        <Found Context="routeData">
            <AuthorizeRouteView                                RouteData="@routeData"
DefaultLayout="@typeof(MainLayout)">
                <NotAuthorized>
                    @if (!context.User.Identity.IsAuthenticated)
                    {
                        <RedirectToWelcomePage/>
                    }
                    else
                    {
                        <AuthorizationError/>
                    }
                </NotAuthorized>
            </AuthorizeRouteView>
        </Found>
        <NotFound>
            <PageTitle>Not found</PageTitle>
            <NotFound/>
        </NotFound>
    </Router>
</CascadingAuthenticationState>
```

Лістинг Б.2 – Клас «VisitRecordsService»

```
public class VisitRecordsService: IVisitRecordsService
{
    private readonly HttpClient _httpClient;
    private readonly IHttpClientFactory _httpClientFactory;
    private readonly IAuthService _authService;

    public VisitRecordsService(HttpClient httpClient,
        IHttpClientFactory httpClientFactory,
        IAuthService authService)
    {
        _httpClient = httpClient;
        _httpClientFactory = httpClientFactory;
        _authService = authService;
    }
    private HttpClient HttpAuthorizedClient()
    {
        return _httpClientFactory.CreateClient("AuthorizedClient");
    }


    public async Task<ICollection<VisitRecordDto>?> GetListAsync()
    {
```

```
            var                 response                 =                 await
HttpAuthorizedClient().GetAsync("api/visitRecords/getList");
            if (!response.IsSuccessStatusCode)
            {
                if   (response.StatusCode   ==   HttpStatusCode.Unauthorized)   await
_authService.LogoutAsync();
                else return null;
            }

            return                                                              await
response.Content.ReadFromJsonAsync<ICollection<VisitRecordDto>>();
        }

        public  async  Task<ICollection<VisitRecordDto>?>  GetListForEmployeeAsync(Guid
employeeId)
        {
            var                 response                 =                 await
HttpAuthorizedClient().GetAsync($"api/visitRecords/getList/employee/{employeeId}");
            if (!response.IsSuccessStatusCode)
            {
                if   (response.StatusCode   ==   HttpStatusCode.Unauthorized)   await
_authService.LogoutAsync();
                else return null;
            }

            return                                                              await
response.Content.ReadFromJsonAsync<ICollection<VisitRecordDto>>();
        }

        public async Task<VisitRecordDto?> GetByIdAsync(Guid visitRecordId)
        {
            var                 response                 =                 await
HttpAuthorizedClient().GetAsync($"api/visitRecords/{visitRecordId}");
            if (!response.IsSuccessStatusCode)
            {
                if   (response.StatusCode   ==   HttpStatusCode.Unauthorized)   await
_authService.LogoutAsync();
                else return null;
            }

            return await response.Content.ReadFromJsonAsync<VisitRecordDto>();
        }
    }
```

ДОДАТОК В

Лістинг MATLAB коду

Лістинг В.1 – Скрипт для побудови діаграми Найквіста

```
numerator = [6 4];
denominator = [7.5 3 0];
sys = tf(numerator, denominator);

figure;
nyquist(sys);
title('Nyquist Diagram');
grid on;
```

Лістинг В.2 – Скрипт для побудови графіка Михайлова

```
coeffs = [7.5 3 4];

w = 0:0.01:10;
Pjw = polyval(coeffs, 1j*w);

figure;
plot(real(Pjw), imag(Pjw));
xlabel('Real Part');
ylabel('Imaginary Part');
title('Mikhailov Plot');
grid on;
```

ДОДАТОК Г

Демонстраційний матеріал у вигляді презентації

1/16

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра КІТАР

КВАЛІФІКАЦІЙНА РОБОТА

На тему: Розробка системи автоматизації для контролю доступу та обліку робочого часу співробітників на основі QR-кодів

Виконав:
студент групи АКТАКІТ-20-1
Стасенко Ярослав Олександрович

Керівник:
професор кафедри КІТАР
Сезонова Ірина Костянтинівна

Мета та завдання роботи

Мета роботи - підвищення безпеки контролю доступу співробітників, а також точного обліку їх робочого часу шляхом впровадження автоматизованої системи на основі QR-кодів.

Об'єкт розробки - процес контролю доступу та обліку робочого часу співробітників на основі QR-кодів системи автоматизації.

Предмет розробки - програмні та апаратні засоби, алгоритми, що забезпечують функціональність системи контролю доступу та обліку робочого часу на основі QR-кодів.

- Розробка серверної складової для впровадження мережевого шаблону системи
- Впровадження зручного інтерфейсу для взаємодії із системою, включаючи облік робочого часу
- Контроль доступу з підвищеною безпекою для підприємства та ідентифікації користувачів

3/16

Контроль доступу

Види систем контролю доступу

Автономна (локальна)

Мережева

4/16

Облік робочого часу

5/16

Вибір програмних засобів

C#

ASP.NET Core (Web Api)

Blazor (WASM)

WPF

MSSQL Server (Express)

Апаратні засоби

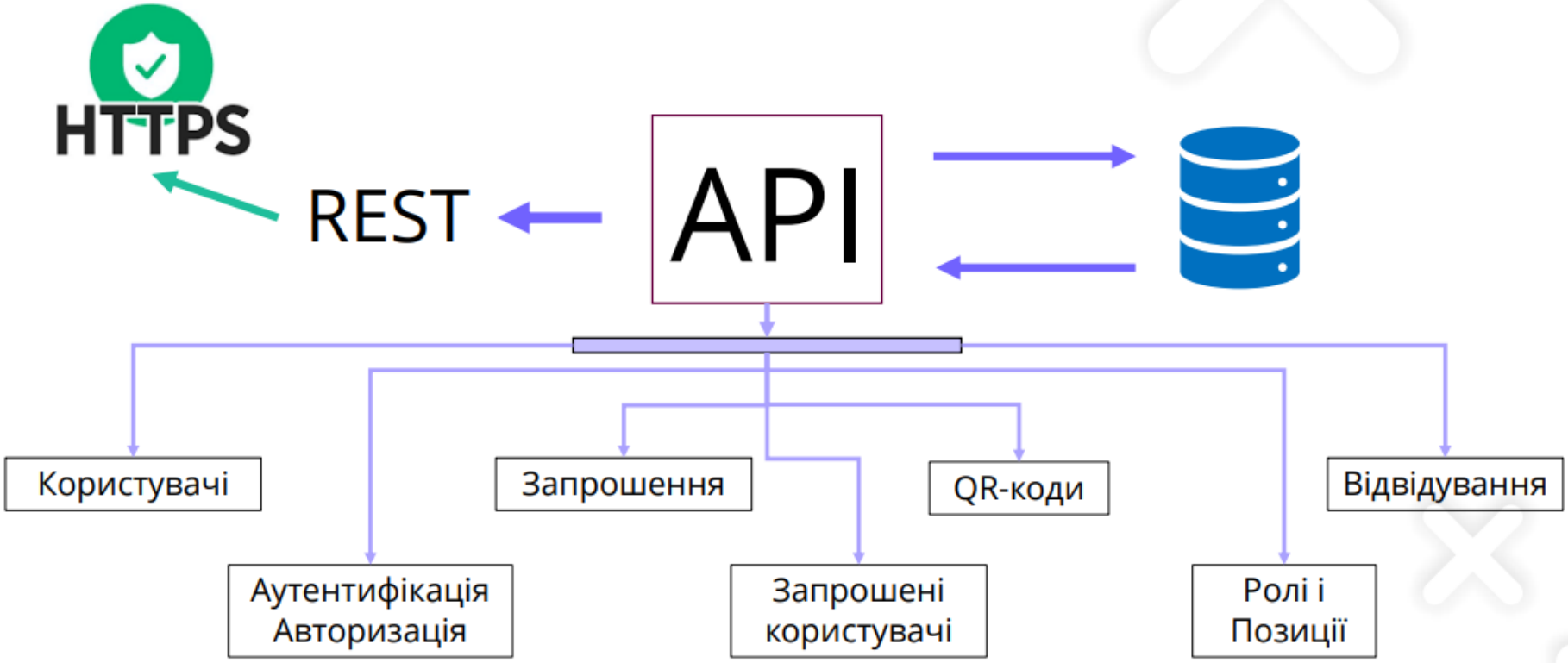
Турнікет TS1000 Pro

Сканер QR-кодів
QR500

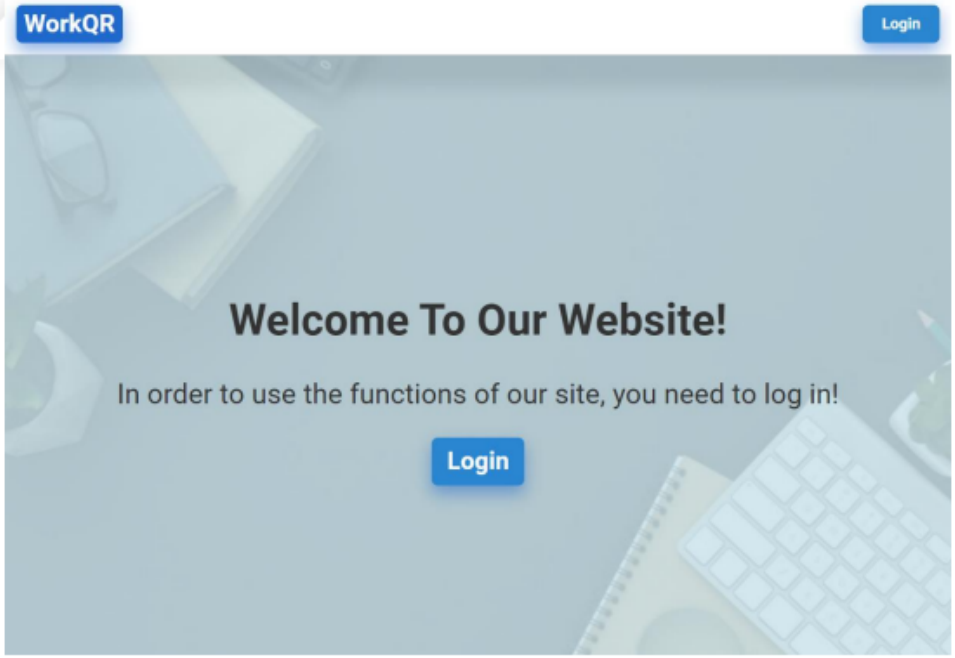
Два інфрачервоні
датчики E18-D80NK

7/16

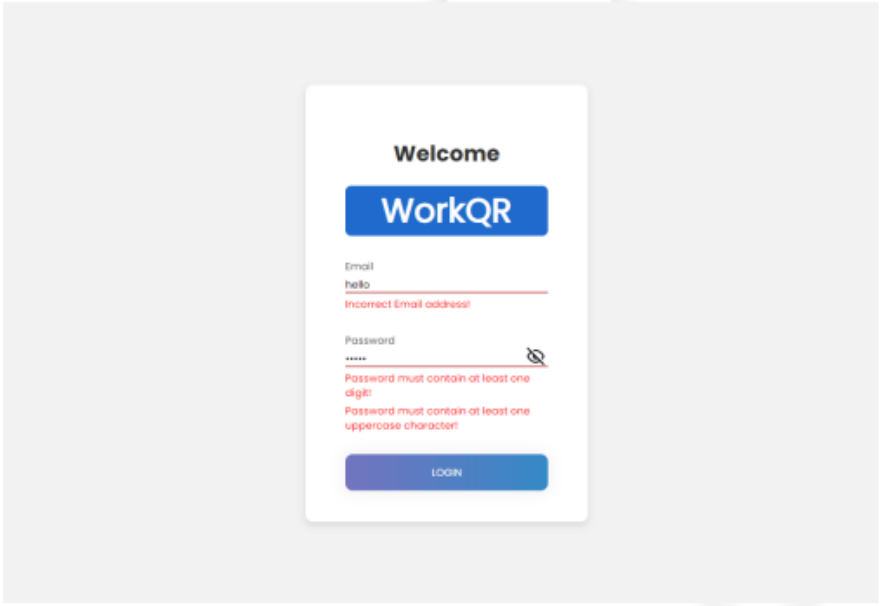
Розробка серверної частини системи



Розробка веб-інтерфейсу



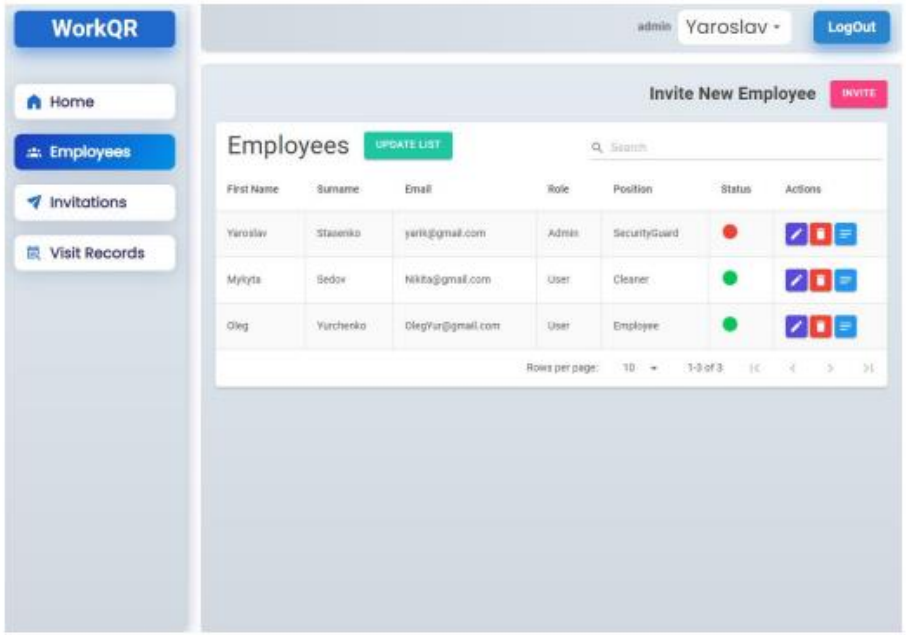
Привітальна сторінка



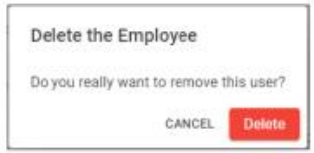
Сторінка входу в систему

9/16

Розробка веб-інтерфейсу



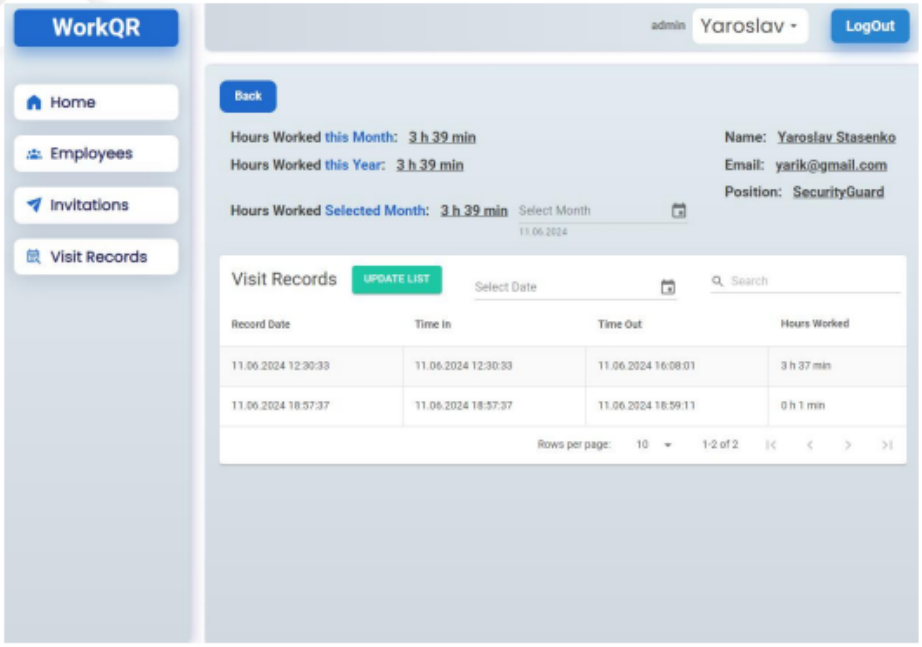
Сторінка «Employees»



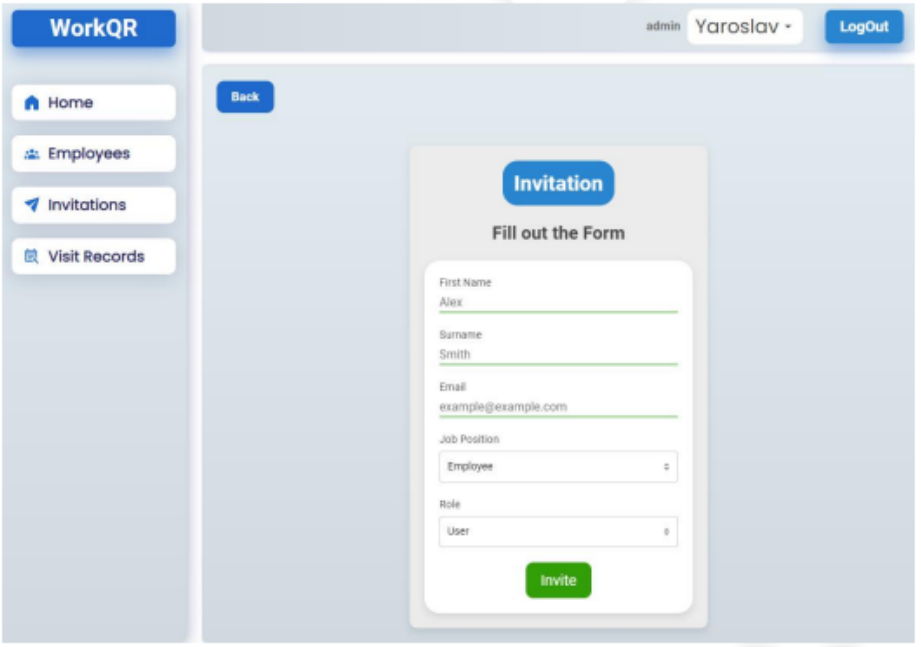
Діалогові вікна сторінки «Employees»

10/16

Розробка веб-інтерфейсу



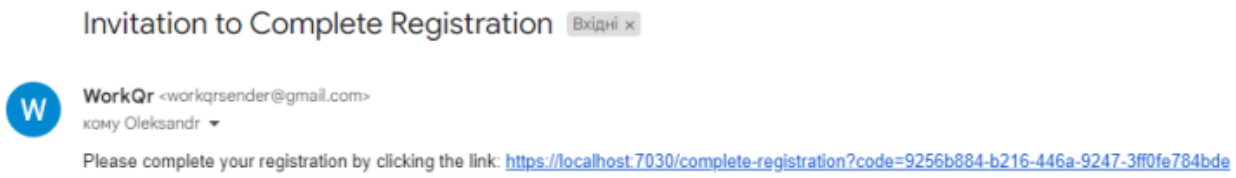
Сторінка із інформацією про працівника



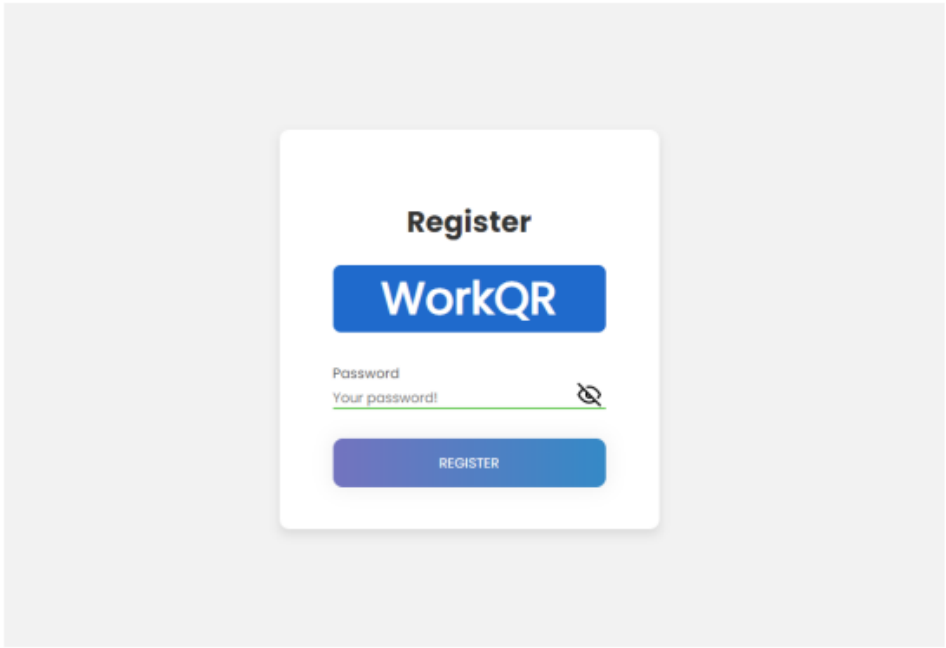
Сторінка із формою для запрошення нового користувача

11/16

Розробка веб-інтерфейсу



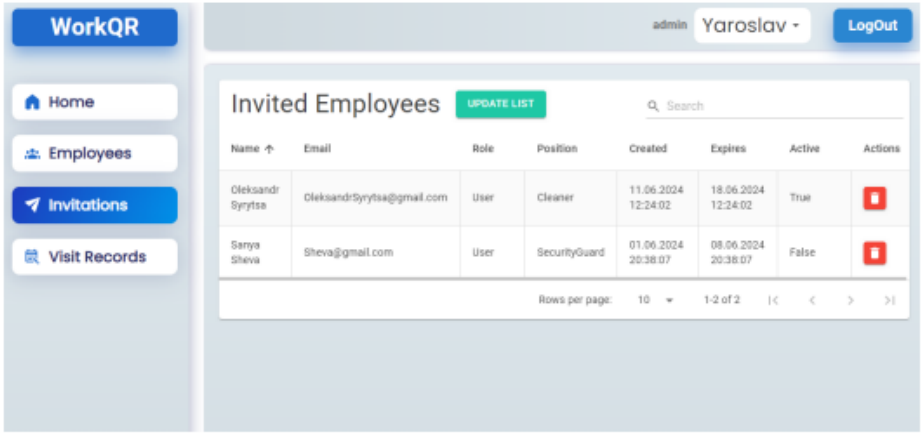
Приклад повідомлення на пошті для нового користувача



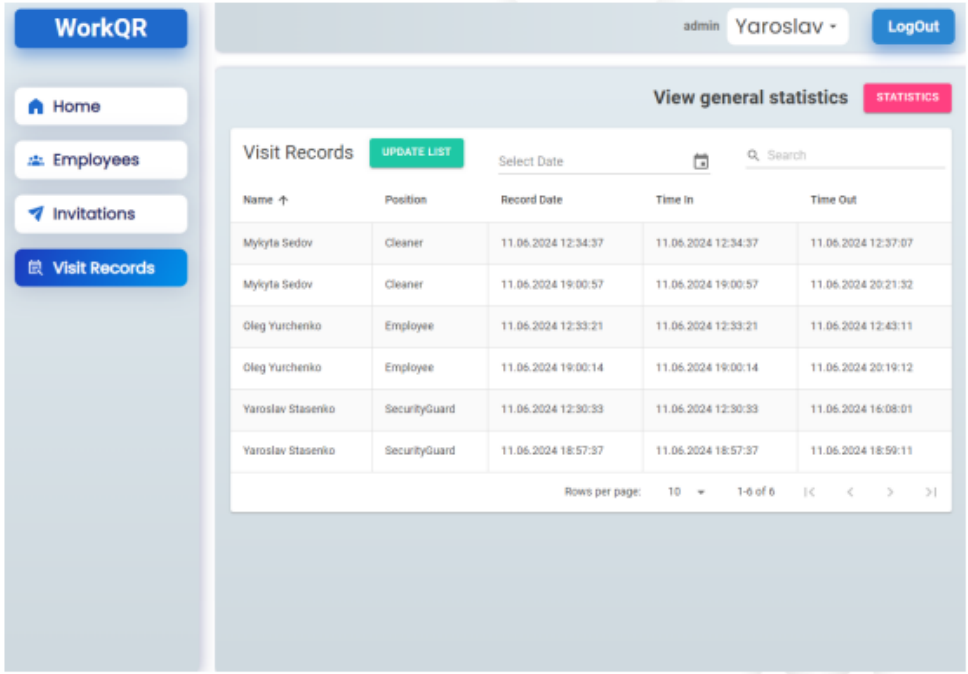
Сторінка із формою для запрошення нового користувача

12/16

Розробка веб-інтерфейсу



Сторінка із запрошеними користувачами

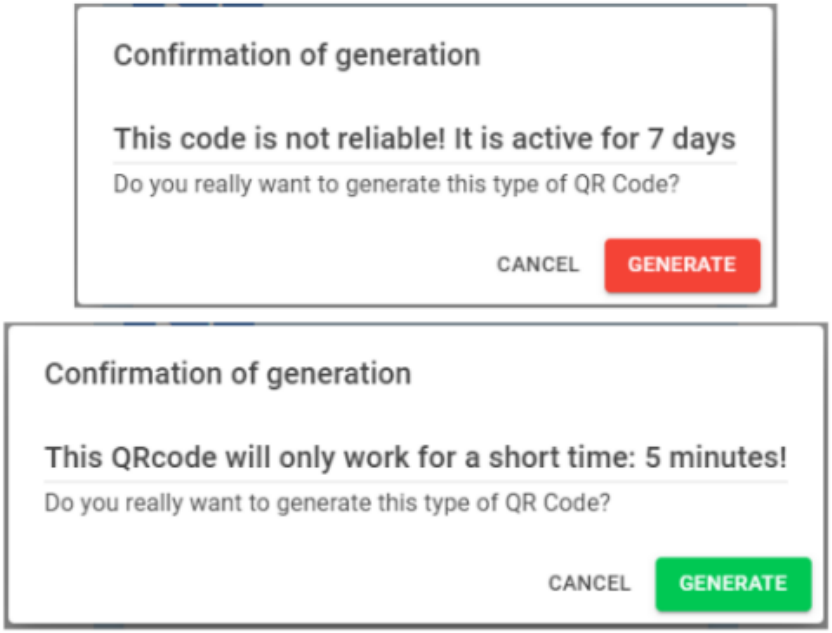


Сторінка відвідувань користувачів

13/16

Розробка веб-інтерфейсу

Сторінка із генерацією QR-кодів



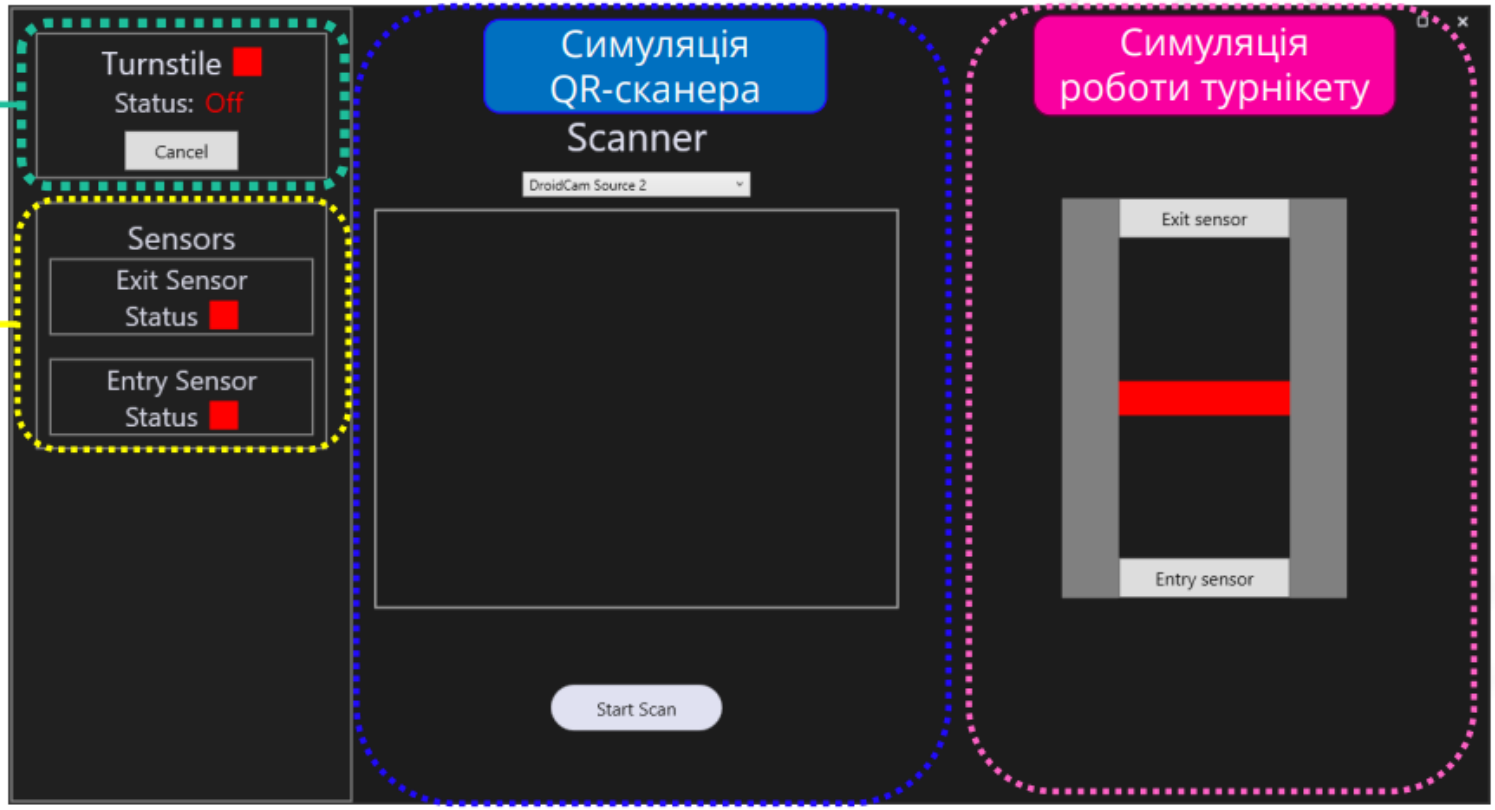
Діалогові вікна при генерації певного типу QR-коду

14/16

Симуляція апаратних компонентів

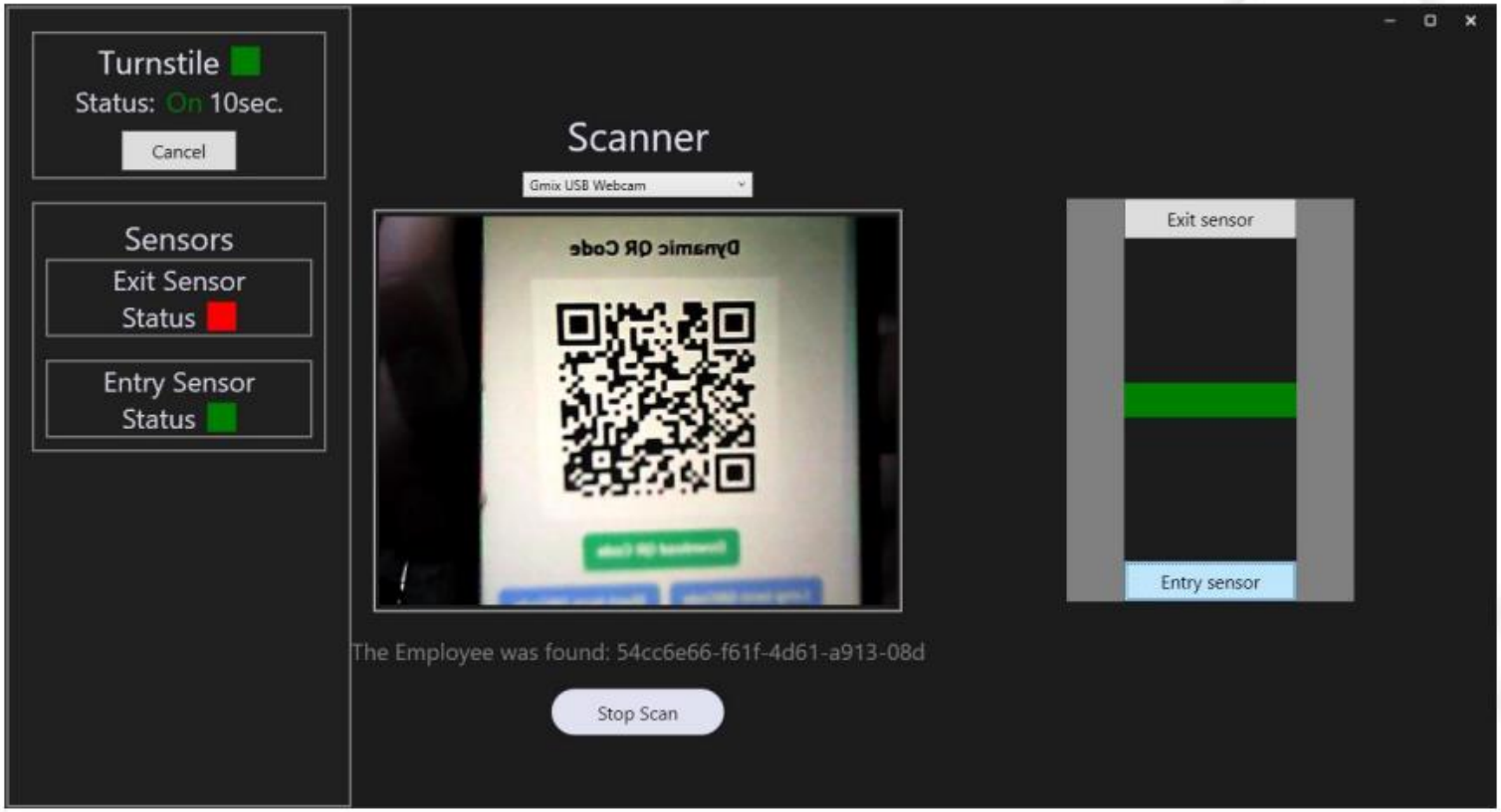
Стан активності турнікету

Стан датчиків



15/16

Процес симуляції апаратних компонентів



16/16

ВИСНОВКИ

- Було проведено аналіз існуючих систем контролю доступу та обліку робочого часу
- Було підібрано та використано сучасні програмні засоби та апаратні компоненти
- Спроектовано та реалізовано серверну частину системи
- Розроблено веб-інтерфейс для взаємодії із системою
- Проведено симуляцію апаратних компонентів системи
- В результаті було отримано розроблену систему автоматизації для контролю доступу та обліку робочого часу співробітників на основі QR-кодів

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Д

Відомість кваліфікаційної роботи бакалавра

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. 505900.013 ВД	Пояснювальна записка	А4, 83 с.
		Додаткові матеріали	
2		Демонстраційний матеріал у	А4, 16 с.
		вигляді презентації	

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК. 505900.013 ВД			
Розроб.		Стасенко Я. О.		15.06.2024	Розробка системи автоматизації для контролю доступу та обліку робочого часу співробітників на основі QR-кодів	Літера	Аркуш	Аркушів
Перевір.		Сезонова І. К.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Стародубцев М.Г.						
Затв.		Невлюдов І.Ш.						