ДОДАТОК А

Програмний код вебдодатка

Лістинг А.1 – Основний файл запуску серверної частини (main.ts)

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { ValidationPipe } from '@nestjs/common';
import{ DocumentBuilder, SwaggerModule } from '@nestjs/swagger';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);

  // Enable CORS
  app.enableCors();

  // Global pipes
  app.useGlobalPipes(
    new ValidationPipe({
      whitelist: true,
      transform: true,
    }),
  );

  // Swagger setup
  const config = new DocumentBuilder()
    .setTitle('CRM API')
    .setDescription('The CRM API description')
    .setVersion('1.0')
    .addBearerAuth()
    .build();
  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup('api/docs', app, document);

  await app.listen(3000);
```

```
}
bootstrap();
```

Лістинг А.2 – Конфігурація головного модуля системи (app.module.ts)

```
import { Module } from '@nestjs/common';
import { DatabaseModule } from
'./modules/database/database.module';
import { AuthModule } from './modules/auth/auth.module';
import { UsersModule } from './modules/users/users.module';
import { EcommerceModule } from
'./modules/ecommerce/ecommerce.module';
import { AnalyticsModule } from
'./modules/analytics/analytics.module';
import { ProjectsModule } from
'./modules/projects/projects.module';
import { DealsModule } from './modules/deals/deals.module';
import { CustomersModule } from
'./modules/customers/customers.module';
import { OrganizationsModule } from
'./modules/organizations/organizations.module';
import { APP_FILTER } from '@nestjs/core';
import { DatabaseExceptionFilter } from './exeptionFilter';

@Module({
  imports: [
    DatabaseModule,
    AuthModule,
    UsersModule,
    EcommerceModule,
    AnalyticsModule,
    ProjectsModule,
    DealsModule,
    CustomersModule,
    OrganizationsModule,
  ],
```

```
  providers: [
    {
      provide: APP_FILTER,
      useClass: DatabaseExceptionFilter,
    },
  ],
})
export class AppModule {}
```

Лістинг А.3 – Модуль автентифікації (auth.module.ts)

```
import { Module, forwardRef } from '@nestjs/common';
import { JwtModule } from '@nestjs/jwt';
import { PassportModule } from '@nestjs/passport';
import { ConfigModule, ConfigService } from '@nestjs/config';
import { AuthService } from './auth.service';
import { AuthController } from './auth.controller';
import { JwtStrategy } from './strategies/jwt.strategy';
import { UsersModule } from '../users/users.module';
import { RolesGuard } from './guards/roles.guard';

@Module({
  imports: [
    forwardRef(() => UsersModule),
    PassportModule,
    JwtModule.registerAsync({
      imports: [ConfigModule],
      useFactory: async (configService: ConfigService) => ({
        secret: configService.get<string>('JWT_SECRET'),
        signOptions: {
          expiresIn: '1d',
        },
      }),
      inject: [ConfigService],
    }),
  ],
```

```
  providers: [AuthService, JwtStrategy, RolesGuard],
  controllers: [AuthController],
  exports: [AuthService],
})
export class AuthModule {}
```

Лістинг А.4 – Сервіс автентифікації (auth.service.ts)

```
import {
  Injectable,
  UnauthorizedException,
  ConflictException,
} from '@nestjs/common';
import { JwtService } from '@nestjs/jwt';
import { ConfigService } from '@nestjs/config';
import { UsersService } from '../users/users.service';
import { LoginDto } from './dto/login.dto';
import { RegisterDto } from './dto/register.dto';
import * as bcrypt from 'bcrypt';
import { UserRole } from '@prisma/client';

@Injectable()
export class AuthService {
  constructor(
    private usersService: UsersService,
    private jwtService: JwtService,
    private configService: ConfigService,
  ) {}

  async validateUser(email: string, password: string) {
    const user = await this.usersService.findByEmail(email);
    if (user && (await bcrypt.compare(password, user.password)))
{
      const { password, ...result } = user;
      return result;
    }
```

```
    return null;
  }

  async login(loginDto: LoginDto) {
    const user = await this.validateUser(loginDto.email,
loginDto.password);
    if (!user) {
      throw new UnauthorizedException('Invalid credentials');
    }

    if (!user.isActive) {
      throw new UnauthorizedException('User is inactive');
    }

    await this.usersService.updateLastLogin(user.id);

    return this.generateTokens(user);
  }

  async register(registerDto: RegisterDto) {
    const existingUser = await
this.usersService.findByEmail(registerDto.email);
    if (existingUser) {
      throw new ConflictException('Email already exists');
    }

    const user = await this.usersService.create({
      ...registerDto,
      role: UserRole.USER,
    });

    return this.generateTokens(user);
  }

  private generateTokens(user: any) {
```

```
    const payload = { email: user.email, sub: user.id, role:
user.role };

    return {
      access_token: this.jwtService.sign(payload),
      user: {
        id: user.id,
        email: user.email,
        firstName: user.firstName,
        lastName: user.lastName,
        role: user.role,
      },
    };
  }

  async refreshToken(userId: string) {
    const user = await this.usersService.findOne(userId);
    return this.generateTokens(user);
  }

  async getProfile(userId: string) {
    const user = await this.usersService.findOne(userId);

    if (!user) {
      throw new UnauthorizedException('User not found');
    }

    const organization = await
this.usersService.findUserOrganization(userId);

    return {
      id: user.id,
      email: user.email,
      firstName: user.firstName,
      lastName: user.lastName,
```

```
      role: user.role,
      isActive: user.isActive,
      lastLoginAt: user.lastLoginAt,
      organization: {
        id: organization.id,
        name: organization.name,
        domain: organization.domain,
      },
      createdAt: user.createdAt,
      updatedAt: user.updatedAt,
    };
  }
}
```

Лістинг А.5 – Контролер автентифікації (auth.controller.ts)

```
import {
  Controller,
  Post,
  Body,
  HttpCode,
  HttpStatus,
  UseGuards,
  Get,
  Request,
} from '@nestjs/common';
import {
  ApiTags,
  ApiOperation,
  ApiResponse,
  ApiBearerAuth,
} from '@nestjs/swagger';
import { AuthService } from './auth.service';
import { LoginDto } from './dto/login.dto';
import { RegisterDto } from './dto/register.dto';
import { JwtAuthGuard } from './guards/jwt-auth.guard';
```

```
@ApiTags('auth')
@Controller('auth')
export class AuthController {
  constructor(private readonly authService: AuthService) {}

  @Post('login')
  @HttpCode(HttpStatus.OK)
  @ApiOperation({ summary: 'User login' })
  @ApiResponse({
    status: HttpStatus.OK,
    description: 'Successfully logged in',
  })
  @ApiResponse({
    status: HttpStatus.UNAUTHORIZED,
    description: 'Invalid credentials',
  })
  async login(@Body() loginDto: LoginDto) {
    return this.authService.login(loginDto);
  }

  @Post('register')
  @ApiOperation({ summary: 'Register new user' })
  @ApiResponse({
    status: HttpStatus.CREATED,
    description: 'User successfully registered',
  })
  @ApiResponse({
    status: HttpStatus.CONFLICT,
    description: 'Email already exists',
  })
  async register(@Body() registerDto: RegisterDto) {
    return this.authService.register(registerDto);
  }
```

```
  @Get('profile')
  @UseGuards(JwtAuthGuard)
  @ApiBearerAuth()
  @ApiOperation({ summary: 'Get current user profile' })
  @ApiResponse({
    status: HttpStatus.OK,
    description: 'Returns the current user profile',
  })
  async getProfile(@Request() req) {
    return this.authService.getProfile(req.user.id);
  }

  @Post('refresh')
  @UseGuards(JwtAuthGuard)
  @ApiBearerAuth()
  @ApiOperation({ summary: 'Refresh access token' })
  @ApiResponse({
    status: HttpStatus.OK,
    description: 'Returns new access token',
  })
  async refresh(@Request() req) {
    return this.authService.refreshToken(req.user.id);
  }
}
```

Лістинг А.6 – Схема бази даних Prisma (schema.prisma)

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}
```

```
// Common enums
enum UserRole {
  ADMIN
  HR_MANAGER
  CONTENT_MANAGER
  WEBSITE_MANAGER
  STORE_MANAGER
  MARKETING_MANAGER
  ANALYST
  MANAGER
  USER
}

enum OrderStatus {
  PENDING
  CONFIRMED
  PROCESSING
  SHIPPED
  DELIVERED
  CANCELLED
  REFUNDED
}

model Organization {
  id              String          @id @default(cuid())
  name            String
  domain          String?         @unique
  settings        Json?
  createdAt       DateTime        @default(now())
  updatedAt       DateTime        @updatedAt

  users           User[]
  customers       Customer[]
  projects        Project[]
  products        Product[]
```

```
  analyticsEvents Analytics[]

  @@map("organizations")
}

model User {
  id              String          @id @default(cuid())
  email           String          @unique
  password        String
  firstName       String
  lastName        String
  role            UserRole        @default(USER)
  isActive        Boolean         @default(true)
  lastLoginAt     DateTime?
  createdAt       DateTime        @default(now())
  updatedAt       DateTime        @updatedAt
  organizationId  String

  organization    Organization    @relation(fields:
[organizationId], references: [id])
  assignedTasks   Task[]
  assignedDeals   Deal[]
  createdProjects Project[]       @relation("ProjectCreator")
  projectMembers  ProjectMember[]
  analyticsEvents Analytics[]

  @@map("users")
}

model Customer {
  id              String    @id @default(cuid())
  email           String?
  firstName       String
  lastName        String
  company         String?
```

```
  phone           String?
  address         Json?
  tags            String[]
  type            CustomerType @default(INDIVIDUAL)
  status          CustomerStatus @default(ACTIVE)
  customFields    Json?
  createdAt       DateTime  @default(now())
  updatedAt       DateTime  @updatedAt
  organizationId  String

  organization    Organization @relation(fields:
[organizationId], references: [id])
  deals           Deal[]
  orders          Order[]
  contacts        Contact[]

  @@map("customers")
}

model Deal {
  id                 String      @id @default(cuid())
  title             String
  description       String?
  amount            Decimal     @default(0)
  currency          String      @default("USD")
  stage             DealStage   @default(NEW)
  probability       Int         @default(0)
  expectedCloseDate DateTime?
  actualCloseDate   DateTime?
  customerId        String
  assignedUserId    String
  createdAt         DateTime    @default(now())
  updatedAt         DateTime    @updatedAt

  customer          Customer    @relation(fields: [customerId],
```

```
references: [id])
  assignedUser     User        @relation(fields:
[assignedUserId], references: [id])

  @@map("deals")
}

model Project {
  id              String    @id @default(cuid())
  name            String
  description     String?
  status          ProjectStatus @default(PLANNING)
  startDate       DateTime
  endDate         DateTime?
  budget          Decimal?
  createdAt       DateTime  @default(now())
  updatedAt       DateTime  @updatedAt
  organizationId  String
  creatorId       String

  organization    Organization   @relation(fields:
[organizationId], references: [id])
  creator         User          @relation("ProjectCreator",
fields: [creatorId], references: [id])
  tasks           Task[]
  members         ProjectMember[]

  @@map("projects")
}

enum CustomerType {
  INDIVIDUAL
  BUSINESS
}
```

```
enum CustomerStatus {
  ACTIVE
  INACTIVE
  BLOCKED
}

enum DealStage {
  NEW
  CONTACTED
  QUALIFIED
  PROPOSAL
  NEGOTIATION
  CLOSED_WON
  CLOSED_LOST
}

enum ProjectStatus {
  PLANNING
  ACTIVE
  ON_HOLD
  COMPLETED
  CANCELLED
}
```

Лістинг А.7 – Головний компонент React додатку (App.tsx)

```
import { BrowserRouter, Routes, Route, Navigate } from 'react-
router-dom'
import { MainLayout } from './components/layout/MainLayout'
import { HomePage } from './pages/HomePage'
import { LoginPage } from './pages/LoginPage'
import { RegisterPage } from './pages/RegisterPage'
import { DashboardPage } from './pages/DashboardPage'
import { CustomersPage } from './pages/customers/CustomersPage'
import { CustomerForm } from './pages/customers/CustomerForm'
import { PrivateRoute } from './components/common/PrivateRoute'
```

```
import { OrganizationsPage } from
'./pages/organizations/OrganizationsPage'
import { ProductsPage } from './pages/products/ProductsPage'
import { UsersPage } from './pages/users/UsersPage'
import { ProjectsPage } from './pages/projects/ProjectsPage'
import { ProjectForm } from './pages/projects/ProjectForm'
import { DealsPage } from './pages/deals/DealsPage'
import { DealForm } from './pages/deals/DealForm'
import { OrdersPage } from './pages/orders/OrdersPage'
import { OrderForm } from './pages/orders/OrderForm'

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="login" element={<LoginPage />} />
        <Route path="register" element={<RegisterPage />} />

        <Route path="/" element={<MainLayout />}>
          <Route index element={<HomePage />} />

          <Route element={<PrivateRoute />}>
            <Route path="dashboard" element={<DashboardPage />}
/>
            <Route path="customers" element={<CustomersPage />}
/>
            <Route path="customers/new" element={<CustomerForm
/>} />
            <Route path="customers/:id/edit"
element={<CustomerForm />} />
            <Route path="organizations"
element={<OrganizationsPage />} />
            <Route path="ecommerce/products"
element={<ProductsPage />} />
            <Route path="profile" element={<UsersPage />} />
```

```
            <Route path="projects" element={<ProjectsPage />} />
            <Route path="projects/new" element={<ProjectForm />}
/>
            <Route path="projects/:id/edit"
element={<ProjectForm />} />
            <Route path="deals" element={<DealsPage />} />
            <Route path="deals/new" element={<DealForm />} />
            <Route path="deals/:id/edit" element={<DealForm />}
/>
            <Route path="orders" element={<OrdersPage />} />
            <Route path="orders/new" element={<OrderForm />} />
          </Route>

          <Route path="*" element={<Navigate to="/" replace />}
/>
        </Route>
      </Routes>
    </BrowserRouter>
  )
}

export default App
```

Лістинг А.8 – Точка входу React додатку (main.tsx)

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import { Provider } from 'react-redux'
import { ThemeProvider } from '@mui/material/styles'
import CssBaseline from '@mui/material/CssBaseline'
import { theme } from './theme/theme'
import { store } from './app/store'
import App from './App'
import './index.css'
import { AdapterDateFns } from '@mui/x-date-
pickers/AdapterDateFns'
```

```
import { LocalizationProvider } from '@mui/x-date-pickers'

createRoot(document.getElementById('root')!).render(
  <StrictMode>
    <LocalizationProvider dateAdapter={AdapterDateFns}>
    <Provider store={store}>
      <ThemeProvider theme={theme}>
        <CssBaseline />
        <App />
      </ThemeProvider>
    </Provider>
    </LocalizationProvider>
  </StrictMode>,
)
```

Лістинг А.9 – Конфігурація залежностей backend (package.json)

```
{
  "name": "crm",
  "version": "0.0.1",
  "description": "CRM System Backend",
  "scripts": {
    "build": "nest build",
    "start": "nest start",
    "start:dev": "nest start --watch",
    "start:prod": "node dist/main",
    "lint": "eslint \"{src,apps,libs,test}/**/*.ts\" --fix",
    "test": "jest",
    "seed": "ts-node prisma/seeds/seed.ts"
  },
  "dependencies": {
    "@nestjs/common": "^10.0.0",
    "@nestjs/core": "^10.0.0",
    "@nestjs/platform-express": "^10.0.0",
    "@nestjs/swagger": "^11.2.0",
    "@nestjs/config": "^3.1.1",
```

```
    "@nestjs/jwt": "^10.2.0",
    "@nestjs/passport": "^10.0.3",
    "@prisma/client": "^5.22.0",
    "class-transformer": "^0.5.1",
    "class-validator": "^0.14.0",
    "bcrypt": "^5.1.1",
    "passport": "^0.7.0",
    "passport-jwt": "^4.0.1",
    "reflect-metadata": "^0.2.0",
    "rxjs": "^7.8.1"
  },
  "devDependencies": {
    "@nestjs/cli": "^10.0.0",
    "@nestjs/testing": "^10.0.0",
    "@types/express": "^4.17.17",
    "@types/jest": "^29.5.2",
    "@types/node": "^20.3.1",
    "prisma": "^5.7.1",
    "typescript": "^5.1.3"
  }
}
```

Лістинг А.10 – Конфігурація залежностей frontend (package.json)

```
{
  "name": "frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "tsc -b && vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
```

```
    "@ant-design/icons": "^6.0.0",
    "@emotion/react": "^11.14.0",
    "@emotion/styled": "^11.14.0",
    "@mui/icons-material": "^7.1.0",
    "@mui/material": "^7.1.0",
    "@mui/x-date-pickers": "^8.4.0",
    "@reduxjs/toolkit": "^2.8.2",
    "antd": "^5.25.3",
    "date-fns": "^4.1.0",
    "react": "^19.1.0",
    "react-dom": "^19.1.0",
    "react-redux": "^9.2.0",
    "react-router-dom": "^7.6.1",
    "recharts": "^2.15.3"
  },
  "devDependencies": {
    "@types/react": "^19.1.2",
    "@types/react-dom": "^19.1.2",
    "@vitejs/plugin-react": "^4.4.1",
    "eslint": "^9.25.0",
    "typescript": "~5.8.3",
    "vite": "^6.3.5"
  }
}
```

ДОДАТОК Б

Демонстраційний матеріал

Формат	Позиція.	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кільк.	Примітки
			Текстові документи		
А4	1	ГЮІК ХХХХХХ.028 ПЗ	Пояснювальна записка		83 с.
			Інші документи		
	2		Демонстраційний матеріал		с.

					ГЮІК ХХХХХХ.028 ПЗ			
Змін.	Арк	Номер докум.	Підпис	Дата				
Розроб.		Махов Д.Р.			Розробка автоматизованої системи управління контентом для малого бізнеса	Літера	Арку	Аркуш
Перевір.		Ромашов Ю.В.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Демська Н. П.						
Затв.		Невлюдов І.Ш.						