

WEB PROGRAMLAMA DERSİ FINAL PROJESİ

Akıllı Kampüs Ekosistem Yönetim Platformu

Ders Adı: Web ve Mobil Programlama

Öğretim Üyesi: Dr. Öğretim Üyesi Mehmet Sevri

Dönem: Güz 2024-5-2025-6

Proje Türü: Grup Projesi (3-4 kişi)

Final Teslim: 28 Aralık 2025, 23:59

Sunum Günü: 29 Aralık 2025

⚠ ÖNEMLİ UYARI:

Bu proje aşamalı olarak geliştirilecektir. Her hafta Pazar günü **zorunlu ara teslimler** bulunmaktadır. Son güne bırakılan projeler değerlendirilmeyecektir. Sürekli geliştirme ve GitHub commit geçmişi değerlendirmeye dahildir.

İçindekiler

- Proje Tanımı ve Amaç
- Projenin Kapsamı ve Modüller
- Teknik Gereksinimler
- Proje Zaman Çizelgesi ve Ara Teslimler
- Part 1: Kimlik Doğrulama ve Kullanıcı Yönetimi
- Part 2: Akademik Yönetim ve GPS Yollama
- Part 3: Yemek Servisi, Etkinlik ve Çizelgeleme
- Part 4: Final Teslim ve Entegrasyon
- Dokümantasyon Gereksinimleri
- Değerlendirme Kriterleri
- Teslim Formatı ve Kurallar
- Akademik Dürüstlük

1. Proje Tanımı ve Amaç

1.1 Genel Bakış

Akıllı Kampüs Ekosistem Yönetim Platformu, bir üniversite kampüsünün günlük operasyonlarını dijitalleştirerek optimize eden kapsamlı bir web uygulaması geliştirilecektir. Bu proje, öğrencilerin gerçek dünya senaryolarına uygun, ölçeklenebilir ve modern web teknolojileri kullanarak profesyonel bir uygulama geliştirme deneyimi kazanmalarını amaçlamaktadır.

1.2 Proje Amaçları

Bu proje ile sizlerden beklenenler:

- Full-Stack Development:** Modern React frontend ve Node.js/Express backend geliştirme
- RESTful API Tasarımı:** Profesyonel API mimari ve dokümantasyonu
- Veritabanı Yönetimi:** İlişkisel veritabanı (PostgreSQL) tasarımcılık ve optimizasyonu
- Authentication & Authorization:** JWT tabanlı güvenli kimlik doğrulama sistemi
- GPS ve Geolocation:** Tarayıcı tabanlı GPS entegrasyonu ve coğrafi hesaplamalar
- Payment Integration:** Ödeme sistemi entegrasyonu (test ortamı)
- Real-time Features:** WebSocket ile gerçek zamanlı bildirimler ve veri akışı
- QR Code Integration:** QR kod oluşturma ve okuma
- File Upload & Storage:** Profil fotoğrafları ve döküman yükleme
- Responsive Design:** Mobil ve desktop uyumlu arayüz
- Testing:** Unit, integration ve E2E testleri
- Deployment:** Docker containerization ve cloud deployment
- Documentation:** Profesyonel teknik dokümantasyon hazırlama
- Team Collaboration:** Git/GitHub ile ekip çalışması ve versiyon kontrolü

1.3 Hedef Kullanıcılar

- Öğrenciler:** Ders kayıt, yoklama, not görüntüleme, yemek rezervasyonu
- Öğretim Üyeleri:** Yoklama açma, not girişi, ders programı görüntüleme
- İdari Personel:** Kullanıcı yönetimi, raporlama, sistem yapılandırması
- Yemekhane Personeli:** Yemek menüsü yönetimi, QR kod okutma
- Tesis Yöneticileri:** Derslik rezervasyonu, IoT sensör verilerini izleme

2. Projenin Kapsamı ve Modüller

2.1 Modüller Genel Bakış

Proje 12 ana modülden oluşmaktadır. Her modül birbirleriyle entegre çalışan bağımsız bir fonksiyonellik sağlar.

2.1.1 Authentication & User Management (Zorunlu)

Modern ve güvenli kullanıcı kimlik doğrulama sistemi.

Özellikler:

- Kullanıcı kaydı (öğrenci, öğretim üyesi, admin)
- Email doğrulama sistemi
- JWT tabanlı login/logout
- Refresh token mekanizması
- Şifre sıfırlama (forgot password)
- Profil yönetimi (CRUD)
- Profil fotoğrafı yükleme
- İsteğe bağlı: 2FA (Two-Factor Authentication)

Teknik Detaylar:

- Şifre hashleme: bcrypt (minimum 10 salt rounds)
 - JWT token expiry: Access token 15 dakika, Refresh token 7 gün
 - Email servis: NodeMailer + Gmail SMTP veya başka servis
 - Dosya upload: Multer middleware
 - Storage: AWS S3 , Google Cloud, Azure vb.
-

2.1.2 Academic Management (Zorunlu)

Akademik süreçlerin yönetimi.

Özellikler:

- Ders kataloğu (course catalog) görüntüleme
- Ders detayları (açıklama, kredi, ECTS, önkoşullar)
- Ders bölümü (section) yönetimi
- Derse kayıt olma (enrollment)
 - Önkoşul kontrolü (recursive prerequisite checking)
 - Çakışma kontrolü (schedule conflict detection)
 - Kapasite kontrolü (atomic increment)

- Dersi bırakma (drop course)
- Kayıtlı derslerim listesi
- Not görüntüleme (öğrenci)
- Not girişi (öğretim üyesi)
- Transkript, öğrenci belegesi görüntüleme ve PDF indirme
- Akademik takvim, kişisel ders takvimi, duyuru görüntüleme

Teknik Detaylar:

- Önkoşul kontrolü: Graph traversal algoritması (BFS/DFS)
 - Schedule conflict: Time overlap detection algorithm
 - Capacity: Database transaction ve row-level locking
 - PDF generation: PDFKit veya Puppeteer
 - Grade calculation: Otomatik harf notu hesaplama (4.0 scale)
-

2.1.3 GPS-Based Attendance System (Zorunlu)

Coğrafi konum tabanlı otomatik yoklama sistemi.

Özellikler:

- Yıklama oturumu açma (öğretim üyesi)
 - Derslik GPS koordinatları otomatik alınır
 - Geofencing radius (varsayılan 15m)
 - QR kod alternatif de olacak, 5 saniye de bir yenilecek, okudnuğunda konum doğrulama ile bilirke var yazacak (backup)
- Öğrenci yıkama verme
 - Tarayıcı GPS API ile konum alma
 - Sunucuda mesafe hesaplama (Haversine formula)
 - GPS spoofing tespiti (mock location detection)
- Yıkama durumu görüntüleme (öğrenci)
- Yıkama raporları (öğretim üyesi)
- Devamsızlık uyarıları (otomatik email/SMS)
- Mazeret bildirme ve onaylama

Teknik Detaylar:

- GPS API: Navigator.geolocation.getCurrentPosition()

- Doğruluk: high accuracy mode, timeout: 10s
- Mesafe hesaplama: Haversine formula

```
distance = 2 * R * asin(sqrt(sin²(Δlat/2) + cos(lat1) * cos(lat2) * sin²(Δlon/2)))
```

- Spoofing detection:
 - IP address validation (kampüs IP aralığı), öğrenci yoklamaya sadece kapüs ağına bağlı iken bağlanabilmelidir, gelen ip belirleme
 - Mock location flag kontrolü
 - Velocity check (önceki konumdan impossible travel)
 - Device sensor tutarlılığı
 - Fraud flagging: Şüpheli aktiviteler otomatik işaretlenir
-

2.1.4 Course Scheduling & Resource Management (Zorunlu)

Ders programını verilen kısıtlara göre otomatik oluşturma ve kaynak yönetimi.

Özellikler:

- Otomatik ders programı oluşturma (constraint satisfaction problem)
- Ders programı görüntüleme (öğrenci/öğretim üyesi)
- Derslik rezervasyonu
- Laboratuvar rezervasyonu
- Ekipman ödünç alma/iade sistemi
- Çakışma kontrolü
- Kaynak kullanım raporları

Teknik Detaylar:

- Scheduling algorithm: Constraint satisfaction problem (CSP)
 - Hard constraints: Çakışma olmamalı, kapasite yeterli olmalı
 - Soft constraints: Öğretim üyesi tercihleri, boşlukları minimize et
- Algorithm seçenekleri:
 - Backtracking with heuristics
 - Genetic algorithm (bonus)
 - Simulated annealing (bonus)

- iCal export: ICS format dosya oluşturma
 - Conflict detection: Time interval overlap
-

2.1.5 Meal Reservation System (Zorunlu)

Yemekhane rezervasyon ve ödeme sistemi.

Özellikler:

- Günlük menü görüntüleme
- Yemek rezervasyonu (öğle/akşam)
- Burslu/ücretli öğrenci ayrımı
- Rezervasyon iptali
- QR kod ile yemek kullanımı
- Sanal cüzdan (wallet) sistemi
- Para yükleme (ödeme entegrasyonu)
- İşlem geçmişi
- Beslenme takibi (kalori, makro besinler)

Teknik Detaylar:

- QR code: QRCode library ile unique ID oluşturma
 - Payment integration: Stripe test mode veya PayTR sandbox
 - Webhook: Payment confirmation callback
 - Quota check: Burslu öğrenciler günde max 2 öğün
 - Wallet transaction: ACID properties, double-entry bookkeeping
 - Nutritional data: JSON format menu items
-

2.1.6 Event Management (Zorunlu)

Kampüs etkinlikleri yönetimi.

Özellikler:

- Etkinlik listeleme ve filtreleme
- Etkinlik detayları
- Etkinliğe kayıt olma
- Kapasite yönetimi

- Bekleme listesi (waitlist)
- QR kod ile giriş kontrolü
- Etkinlik sonrası anket
- Email bildirimler

Teknik Detaylar:

- QR code generation: Unique registration code per user
 - Capacity management: Optimistic locking veya pessimistic locking
 - Waitlist: FIFO queue implementation
 - Survey: Dynamic form builder (JSON schema)
 - Email notifications: Event reminders (1 day before, 1 hour before)
-

2.1.7 Notification System (Zorunlu)

Çok kanallı bildirim sistemi.

Özellikler:

- In-app notifications (bildirim kutusu)
- Email notifications
- Push notifications (web push - bonus)
- SMS notifications (isteğe bağlı)
- Bildirim tercihleri (kategori bazında aç/kapa)
- Okundu/okunmadı durumu
- Toplu işaretleme

Teknik Detaylar:

- Real-time: WebSocket (Socket.io)
 - Email: NodeMailer
 - Push: Web Push API (bonus)
 - SMS: Twilio API (bonus - test mode)
 - Queue system: Bull + Redis (bonus) veya simple cron job
 - Categories: academic, attendance, meal, event, payment, system
-

2.1.8 Analytics & Reporting (Gerekli)

Sistem geneli istatistikler ve raporlar.

Özellikler:

- Admin dashboard (istatistikler)
- Akademik performans raporları (GPA, geçme oranı)
- Yoklama analitiği (devamsızlık istatistikleri)
- Yemek kullanım raporları
- Kaynak kullanım raporları
- Export: Excel, PDF, CSV

Teknik Detaylar:

- Charts: Chart.js veya Recharts
 - Data aggregation: SQL GROUP BY, window functions
 - Export libraries: ExcelJS, PDFKit
 - Caching: Redis veya memory cache (optimization)
-

2.1.9 IoT Sensor Integration (İsteğe Bağlı - Bonus)

Akıllı kampüs için IoT sensör entegrasyonu.

Özellikler:

- Sensör verilerini görüntüleme (sıcaklık, nem, doluluk)
- Gerçek zamanlı veri akışı (WebSocket)
- Anomali tespiti ve uyarılar
- Enerji tüketimi takibi
- Grafikler ve görselleştirme

Teknik Detaylar:

- WebSocket: Real-time sensor data streaming
 - Time-series data: Efficient storage and querying
 - Mock sensors: Simulated sensor data for testing
 - Alerts: Threshold-based notification triggers
-

2.2 Modül Öncelik ve Zorunluluk

Modül	Öncelik	Zorunluluk	Part
Authentication & User Management	P0	Zorunlu	Part 1
Academic Management	P0	Zorunlu	Part 2
GPS-Based Attendance	P0	Zorunlu	Part 2
Course Scheduling	P1	Zorunlu	Part 3
Meal Reservation System	P1	Zorunlu	Part 3
Event Management	P1	Zorunlu	Part 3
Notification System	P1	Zorunlu	Part 4
Analytics & Reporting	P2	Gerekli	Part 4
IoT Sensor Integration	P3	Bonus	Part 4

Not: P0 (kritik), P1 (yüksek), P2 (orta), P3 (düşük)

3. Teknik Gereksinimler

3.1 Zorunlu Teknolojiler

Frontend (Kesinlikle React)

- React 18+ (Hooks kullanımı zorunlu)
- React Router v6 (client-side routing)
- State Management: Context API + useReducer VEYA Redux Toolkit
- HTTP Client: Axios
- Styling: Material-UI, Tailwind CSS, VEYA CSS Modules
- Form Handling: React Hook Form + Yup validation
- Charts: Chart.js, Recharts VEYA Victory
- QR Code: qrcode.react
- Maps: Leaflet VEYA Google Maps API

Backend

- Node.js 18+ LTS
- Express.js 4+
- PostgreSQL 14+ (zorunlu - SQLite kabul edilmez)
- ORM: Sequelize VEYA Prisma
- Authentication: JWT (jsonwebtoken)
- Password Hashing: bcrypt
- File Upload: Multer
- Email: NodeMailer
- Validation: Joi VEYA express-validator
- API Documentation: Swagger/OpenAPI

Development Tools

- Version Control: Git + GitHub (zorunlu)
- Package Manager: npm VEYA yarn
- Environment: dotenv
- Linting: ESLint
- Code Formatting: Prettier
- Testing: Jest + React Testing Library (backend & frontend)

Deployment & DevOps

- Containerization: Docker + Docker Compose (zorunlu)
- CI/CD: GitHub Actions (bonus)
- Database Migrations: Sequelize migrations VEYA Prisma migrate
- Logging: Winston VEYA Morgan

3.2 Veritabanı Gereksinimleri

Minimum Tablolar (30+):

```
users, students, faculty, admins, departments, courses,  
course_sections, enrollments, attendance_sessions,  
attendance_records, excuse_requests, classrooms, reservations,  
schedules, meal_menus, meal_reservations, wallets, transactions,  
events, event_registrations, notifications, notification_preferences,  
iot_sensors, sensor_data, audit_logs, password_resets,  
email_verifications, session_tokens, ve daha fazlası...
```

Veritabanı Tasarım Kuralları:

- Normalization: 3NF minimum

- Foreign keys: CASCADE ve RESTRICT uygun kullanımı
- Indexes: Performance için gerekli alanlara index
- Constraints: CHECK, UNIQUE, NOT NULL constraints
- JSONB: Flexible data için kullanılabilir (schedule, metadata)
- Soft delete: deleted_at timestamp pattern (bazı tablolarda)

3.3 API Gereksinimleri

RESTful API Standards:

- Base URL: /api/v1/
- HTTP Methods: GET, POST, PUT, PATCH, DELETE (uygun kullanım)
- Status Codes: 200, 201, 204, 400, 401, 403, 404, 409, 500
- Response Format: JSON (consistent structure)
- Error Handling: Standardized error responses
- Pagination: page, limit, sort parameters
- Filtering: Query parameters
- Rate Limiting: Express-rate-limit (bonus)

Minimum 60+ Endpoints: Tüm modüller için CRUD operations + özel endpoint'ler

3.4 Güvenlik Gereksinimleri

Zorunlu Güvenlik Önlemleri:

1. Authentication:

- JWT token-based auth
- Refresh token mechanism
- Token expiration handling
- Secure password storage (bcrypt)

2. Authorization:

- Role-based access control (RBAC)
- Middleware authentication guards
- Route protection (frontend & backend)

3. Input Validation:

- Backend validation (Joi/express-validator)

- Frontend validation (React Hook Form + Yup)
- SQL injection prevention (ORM parameterized queries)
- XSS prevention (input sanitization)

4. CORS:

- Proper CORS configuration
- Allowed origins whitelist

5. HTTPS:

- Development: self-signed certificate (optional)
- Production: Let's Encrypt SSL (bonus)

6. Environment Variables:

- Sensitive data in .env
- .env.example provided
- Never commit .env to Git

3.5 Testing Gereksinimleri

Zorunlu Testler:

- **Unit Tests:** Critical business logic (minimum 50+ tests)
- **Integration Tests:** API endpoints (minimum 30+ tests)
- **E2E Tests:** Critical user flows (minimum 5 scenarios - bonus)

Test Coverage:

- Backend: Minimum %85 code coverage
- Frontend: Minimum %75 code coverage

Test Araçları:

- Backend: Jest + Supertest
- Frontend: Jest + React Testing Library
- E2E: Cypress VEYA Playwright (bonus)

3.6 Performance Gereksinimleri

Hedef Metrikler:

- Page load time: < 3 saniye (initial load)
- API response time: < 500ms (average)
- Database query time: < 100ms (optimized queries)
- Concurrent users: 100+ kullanıcı desteği

Optimization Teknikler:

- Database indexing
- Query optimization (EXPLAIN ANALYZE)
- Lazy loading (React components)
- Code splitting (React.lazy)
- Image optimization
- Caching (Redis - bonus)
- CDN for static assets (bonus)

4. Proje Zaman Çizelgesi ve Ara Teslimler

4.1 Genel Zaman Çizelgesi

Hafta 1: 5 Aralık - 8 Aralık (4 gün)

└ Part 1: Authentication & User Management

Hafta 2: 9 Aralık - 15 Aralık (7 gün)

└ Part 2: Academic Management + GPS Attendance

Hafta 3: 16 Aralık - 22 Aralık (7 gün)

└ Part 3: Meal + Event + Scheduling

Hafta 4: 23 Aralık - 28 Aralık (6 gün)

└ Part 4: Final Integration + Analytics + Polish

Hafta 4+: 29 Aralık 2025

└ SUNUM GÜNÜ

4.2 Ara Teslim Tarihleri (Zorunlu)

Ara Teslim	Tarih	Saat	İçerik
Part 1	8 Aralık 2025 (Pazar)	23:59	Auth + User Management + Docs
Part 2	15 Aralık 2025 (Pazar)	23:59	Academic + Attendance + Docs
Part 3	22 Aralık 2025 (Pazar)	23:59	Meal + Event + Schedule + Docs
FINAL	28 Aralık 2025 (Cumartesi)	23:59	Complete Project + All Docs
SUNUM	29 Aralık 2025 (Pazar)	TBA	Live Demo + Q&A

4.3 Ara Teslim Kuralları

Önemli Notlar:

- Her ara teslim **ayrı ayrı değerlendirilecektir**
- Her ara teslimde **hem kod hem dokümantasyon olmalıdır**
- Geç teslimler **her gün için %10 puan kesilir** (max 3 gün)
- Son iki güne bırakılan projeler **düşük süreklilik puanı** alır
- GitHub commit history kontrol edilecektir
- Her part, bir önceki part'i **içermelidir** (incremental)

5. Part 1: Kimlik Doğrulama ve Kullanıcı Yönetimi

Teslim Tarihi: 8 Aralık 2025 (Pazar), 23:59

Süre: 4 gün

Ağırlık: %20

5.1 Part 1 Hedefleri

Bu aşamada projenin temelini oluşturacaksınız:

- Proje yapısını kurmak
- Veritabanı şemasını tasarlamak
- Authentication sistemi
- Kullanıcı yönetimi
- Temel frontend yapısı

5.2 Part 1 - Yapılacaklar Listesi

5.2.1 Backend Görevleri

Proje Kurulumu:

- Node.js projesi oluşturma (`npm init`)
- Gerekli paketleri yükleme (express, pg, sequelize, bcrypt, jwt, vs.)
- Folder structure oluşturma:

```
backend/
  └── src/
    ├── config/
    ├── controllers/
    ├── middleware/
    ├── models/
    ├── routes/
    ├── services/
    ├── utils/
    └── app.js
  └── tests/
  └── .env.example
  └── .gitignore
  └── package.json
  └── README.md
```

Veritabanı:

- PostgreSQL veritabanı oluşturma
- Sequelize/Prisma kurulumu ve konfigürasyonu
- User tablosu (id, email, password_hash, role, created_at, updated_at)
- Student tablosu (user_id, student_number, department_id, gpa, cgpa)
- Faculty tablosu (user_id, employee_number, title, department_id)
- Department tablosu (id, name, code, faculty)
- Migration dosyaları oluşturma
- Seed data (test users: 5 öğrenci, 2 öğretim üyesi, 1 admin)

Authentication Endpoints:

- POST `/api/v1/auth/register` - Kullanıcı kaydı
 - Email validation
 - Password strength check (min 8 char, uppercase, number)
 - Hash password with bcrypt

- Create verification token
- Send verification email
- Return 201 Created

POST /api/v1/auth/verify-email - Email doğrulama

- Validate token
- Activate user account
- Return 200 OK

POST /api/v1/auth/login - Kullanıcı girişи

- Email/password validation
- Check if user exists and activated
- Verify password
- Generate JWT access token (15 min)
- Generate refresh token (7 days)
- Return tokens + user info (without password)

POST /api/v1/auth/refresh - Token yenileme

- Validate refresh token
- Generate new access token
- Return new access token

POST /api/v1/auth/logout - Çıkış yapma

- Invalidate refresh token
- Return 204 No Content

POST /api/v1/auth/forgot-password - Şifre sıfırlama isteği

- Generate reset token (24h expiry)
- Send reset email
- Return 200 OK

POST /api/v1/auth/reset-password - Şifre sıfırlama

- Validate reset token
- Hash new password
- Update user password
- Invalidate all sessions
- Return 200 OK

User Management Endpoints:

- GET /api/v1/users/me - Profil görüntüleme
 - Require authentication
 - Return user info (no password)
- PUT /api/v1/users/me - Profil güncelleme
 - Validate input (name, phone, etc.)
 - Update user record
 - Return updated user info
- POST /api/v1/users/me/profile-picture - Profil fotoğrafı yükleme
 - Multer middleware (max 5MB, jpg/png)
 - Save to uploads/ folder
 - Update user profile_picture_url
 - Return new URL
- GET /api/v1/users - Kullanıcı listesi (admin only)
 - Pagination (page, limit)
 - Filtering (role, department)
 - Search (name, email)
 - Return user list

Middleware:

- Authentication middleware (verify JWT)
- Authorization middleware (role-based)
- Error handling middleware
- Request validation middleware
- File upload middleware (Multer)

Utilities:

- JWT utilities (generate, verify)
- Email service (NodeMailer)
- Validation schemas (Joi)
- Logger (Winston - bonus)

Testing:

- Unit tests: Auth service (register, login, token generation)

- Unit tests: User service
- Integration tests: Auth endpoints (minimum 10 tests)
- Integration tests: User endpoints (minimum 5 tests)

5.2.2 Frontend Görevleri

Proje Kurulumu:

- React projesi oluşturma (`npx create-react-app` veya Vite)
- Gerekli paketleri yükleme (react-router-dom, axios, formik, yup, vs.)
- Folder structure:

```
frontend/
├── public/
└── src/
    ├── assets/
    ├── components/
    ├── context/
    ├── hooks/
    ├── pages/
    ├── services/
    ├── utils/
    ├── App.js
    └── index.js
├── .env.example
└── package.json
└── README.md
```

Pages:

- Login Page (`/login`)
 - Email input
 - Password input
 - "Remember me" checkbox (optional)
 - "Forgot password?" link
 - Form validation
 - Error messages
 - Loading state
 - Redirect to dashboard on success

Register Page (/register)

- o Full name, email, password, confirm password
- o User type selection (student/faculty)
- o Student number input (if student)
- o Department selection
- o Terms & conditions checkbox
- o Form validation
- o Success message (email verification required)

Email Verification Page (/verify-email/:token)

- o Verify token on mount
- o Show success/error message
- o Redirect to login after 3 seconds

Forgot Password Page (/forgot-password)

- o Email input
- o Send reset link button
- o Success message

Reset Password Page (/reset-password/:token)

- o New password input
- o Confirm password input
- o Form validation
- o Success message
- o Redirect to login

Dashboard Page (/dashboard)

- o Require authentication
- o Show welcome message with user name
- o Role-based navigation menu
- o Placeholder for upcoming features

Profile Page (/profile)

- o Display current user info
- o Edit form (name, phone, etc.)
- o Profile picture upload
- o Change password section

- o Save button
- o Success/error messages

Components:

- Navbar (with user menu, logout button)
- Sidebar (navigation menu)
- ProtectedRoute (auth guard)
- Loading spinner
- Alert/Toast notifications
- Form input components (TextInput, Select, etc.)

State Management:

- Auth Context (user state, login, logout functions)
- API service (axios instance, interceptors)
- Token management (localStorage or cookies)
- Auto token refresh

Routing:

- React Router setup
- Public routes (login, register, forgot password)
- Protected routes (dashboard, profile)
- Role-based routing (student/faculty/admin views)
- 404 page

Styling:

- Responsive design (mobile-first)
- Consistent theme (colors, typography)
- Material-UI / Tailwind CSS integration
- Form styling
- Button styles
- Card layouts

Testing:

- Component tests: Login form
- Component tests: Register form
- Integration tests: Auth flow (bonus)

5.2.3 DevOps & Deployment

- Docker Compose dosyası (backend + postgres)

```
version: '3.8'
services:
  postgres:
    image: postgres:14
    environment:
      POSTGRES_DB: campus_db
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: password
    ports:
      - "5432:5432"
  backend:
    build: ./backend
    ports:
      - "5000:5000"
    depends_on:
      - postgres
```

- .dockerignore dosyası
- README.md (setup instructions)

5.3 Part 1 - Teslim Edilecekler

Kod:

1. Backend repository (GitHub link)
 - o Tüm kaynak kodlar
 - o .env.example dosyası
 - o README.md (setup, run, test instructions)
 - o Migration ve seed dosyaları
2. Frontend repository (GitHub link)
 - o Tüm kaynak kodlar
 - o .env.example dosyası
 - o README.md
3. Docker Compose dosyası

- Tek komutla çalışır durumda (docker-compose up)

Dokümantasyon (Markdown formatında):

1. PROJECT_OVERVIEW.md

- Proje tanımı
- Teknoloji stack'i
- Proje yapısı
- Grup üyeleri ve görev dağılımı

2. API_DOCUMENTATION.md (Part 1 için)

- Authentication endpoints (8 adet)
- User management endpoints (4 adet)
- Request/response examples
- Error codes

3. DATABASE_SCHEMA.md

- ER diagram (draw.io, dbdiagram.io, veya manuel)
- Tablo açıklamaları (users, students, faculty, departments)
- İlişkiler (foreign keys)
- Indexes

4. USER_MANUAL_PART1.md

- Nasıl kayıt olunur?
- Nasıl giriş yapılır?
- Profil nasıl güncellenir?
- Ekran görüntülerি

5. TEST_REPORT_PART1.md

- Test coverage raporu
- Test sonuçları (passed/failed)
- Ekran görüntülerি

Demo Video: Projenizi anlatan bir video hazırlayıp youtube'a yükleneyeceksiniz public paylaşım olmayacak mehmetsevri@gmail.com ile paylaşılacak.

5-10 dakikalık video (YouTube unlisted link)

- Kayıt olma
- Email doğrulama (simüle edilebilir)
- Giriş yapma
- Profil görüntüleme ve güncelleme
- Profil fotoğrafı yükleme

5.4 Part 1 - Değerlendirme Kriterleri

Kriter	Puan	Açıklama
Backend Implementasyon	40	
- Database design	10	Normalization, relations, indexes
- Auth endpoints	15	Register, login, token refresh, password reset
- User management	10	Profile CRUD, file upload
- Code quality	5	Clean code, comments, error handling
Frontend Implementasyon	30	
- UI/UX design	10	Responsive, intuitive, consistent
- Forms & validation	10	Client-side validation, error messages
- State management	5	Context API, token handling
- Routing	5	Protected routes, redirects
Testing	10	
- Unit tests	5	Coverage, quality
- Integration tests	5	API endpoint tests
Dokümantasyon	15	
- API documentation	5	Complete, clear examples
- Database schema	5	ER diagram, descriptions
- User manual	5	Screenshots, step-by-step
DevOps & Deployment	5	
- Docker setup	3	Works with docker-compose up
- README quality	2	Clear instructions
TOPLAM	100	

Bonus Puanlar (+10):

- 2FA (Two-Factor Authentication) implementasyonu: +3
- Email template'leri (HTML emails): +2
- Password strength meter: +1

- Account lockout (brute-force protection): +2
 - User activity log: +2
-

6. Part 2: Akademik Yönetim ve GPS Yoklama

Teslim Tarihi: 15 Aralık 2025 (Pazar), 23:59

Süre: 7 gün

Ağırlık: %25

6.1 Part 2 Hedefleri

Bu aşamada projenin kalbi olan akademik işlemler ve GPS tabanlı yoklama sistemini geliştireceksiniz:

- Ders yönetimi ve kayıt sistemi
- GPS tabanlı yoklama
- Önkoşul ve çıkışma kontrolleri
- Not yönetimi
- Kompleks business logic

6.2 Part 2 - Yapılacaklar Listesi

6.2.1 Backend Görevleri

Veritabanı (Yeni Tablolar):

- courses** (id, code, name, description, credits, ects, syllabus_url, department_id)
- course_sections** (id, course_id, section_number, semester, year, instructor_id, capacity, enrolled_count, schedule_json)
- course_prerequisites** (course_id, prerequisite_course_id)
- enrollments** (id, student_id, section_id, status, enrollment_date, midterm_grade, final_grade, letter_grade, grade_point)
- attendance_sessions** (id, section_id, instructor_id, date, start_time, end_time, latitude, longitude, geofence_radius, qr_code, status)
- attendance_records** (id, session_id, student_id, check_in_time, latitude, longitude, distance_from_center, is_flagged, flag_reason)
- excuse_requests** (id, student_id, session_id, reason, document_url, status, reviewed_by, reviewed_at, notes)

- classrooms (id, building, room_number, capacity, features_json)
- Migration ve seed data

Academic Management Endpoints:

Courses:

- GET /api/v1/courses - Ders listesi (pagination, filtering, search)
- GET /api/v1/courses/:id - Ders detayları (prerequisites dahil)
- POST /api/v1/courses - Yeni ders oluşturma (admin only)
- PUT /api/v1/courses/:id - Ders güncelleme (admin only)
- DELETE /api/v1/courses/:id - Ders silme (soft delete, admin only)

Course Sections:

- GET /api/v1/sections - Section listesi (filtering by semester, instructor)
- GET /api/v1/sections/:id - Section detayları
- POST /api/v1/sections - Section oluşturma (admin)
- PUT /api/v1/sections/:id - Section güncelleme (admin)

Enrollments:

- POST /api/v1/enrollments - Derse kayıt olma (student)

- o Önkoşul kontrolü: Recursive prerequisite checking

```
// Pseudocode
function checkPrerequisites(courseId, studentId) {
    prerequisites = getPrerequisites(courseId);
    for (prereq of prerequisites) {
        if (!hasCompletedCourse(studentId, prereq)) {
            throw new Error("Prerequisite not met");
        }
        // Recursive check
        checkPrerequisites(prereq, studentId);
    }
}
```

- o Çakışma kontrolü: Time overlap detection

```

function hasScheduleConflict(studentSchedule, newSectionSchedule) {
    for (existingClass of studentSchedule) {
        if (timeOverlap(existingClass, newSectionSchedule)) {
            return true;
        }
    }
    return false;
}

```

- Kapasite kontrolü: Atomic update

```

UPDATE course_sections
SET enrolled_count = enrolled_count + 1
WHERE id = ? AND enrolled_count < capacity;
-- Check affected rows, if 0 then full

```

- Create enrollment record
- Send confirmation notification
- Return 201 Created

`DELETE /api/v1/enrollments/:id` - Dersi bırakma

- Check drop period (first 4 weeks)
- Update section capacity
- Update enrollment status
- Send notification

`GET /api/v1/enrollments/my-courses` - Kayıtlı derslerim (student)

`GET /api/v1/enrollments/students/:sectionId` - Dersin öğrenci listesi (faculty)

Grades:

`GET /api/v1/grades/my-grades` - Notlarım (student)

`GET /api/v1/grades/transcript` - Transkript JSON (student)

`GET /api/v1/grades/transcript/pdf` - Transkript PDF (student)

- Generate PDF with PDFKit or Puppeteer
- Include university letterhead

- List all courses, grades, CGPA

- Return PDF buffer

`POST /api/v1/grades` - Not girişi (faculty)

- Validate instructor teaches the section
- Update enrollment record (midterm, final, letter_grade)
- Calculate grade_point automatically
- Calculate GPA/CGPA
- Send notification to student

Attendance System Endpoints:

Sessions (Faculty):

`POST /api/v1/attendance/sessions` - Yoklama oturumu açma

- Get section info
- Get classroom GPS coordinates from database
- Set geofence_radius (default 15m)
- Generate unique QR code
- Set expiry (e.g., 30 minutes)
- Send push notification to enrolled students
- Return session details

`GET /api/v1/attendance/sessions/:id` - Oturum detayları

`PUT /api/v1/attendance/sessions/:id/close` - Oturumu kapatma

`GET /api/v1/attendance/sessions/my-sessions` - Benim oturumlarım (faculty)

`GET /api/v1/attendance/report/:sectionId` - Yoklama raporu (faculty)

Check-in (Student):

`POST /api/v1/attendance/sessions/:id/checkin` - Yoklama verme

- Request body: `{ latitude, longitude, accuracy }`
- Get session details (lat, lon, radius)
- **Calculate distance:** Haversine formula

- **GPS Spoofing Detection:**

- Validate distance \leq radius + accuracy buffer (e.g., radius + 5m)
- Check if already checked in
- Detect spoofing
- Record attendance (with flag if suspicious)
- If flagged, notify instructor
- Return success message

`GET /api/v1/attendance/my-attendance` - Yoklama durumum (student)

- List all courses with attendance stats
- Calculate attendance percentage
- Highlight warning ($>20\%$ absence) and critical ($>30\%$)

Excuse Requests:

`POST /api/v1/attendance/excuse-requests` - Mazeret bildirme (student)

- Upload document (Multer)
- Create excuse request
- Notify instructor

`GET /api/v1/attendance/excuse-requests` - Mazeret listesi (faculty)

`PUT /api/v1/attendance/excuse-requests/:id/approve` - Mazeret onaylama (faculty)

`PUT /api/v1/attendance/excuse-requests/:id/reject` - Mazeret reddetme (faculty)

Background Jobs:

Absence Warning Job (runs daily via cron)

- Calculate attendance rates for all students
- If $\geq 20\%$, send warning email + push
- If $\geq 30\%$, send critical warning email + SMS (optional) + push
- Notify advisor
- Log warnings

Business Logic Services:

- PrerequisiteService (recursive checking)
- ScheduleConflictService (time overlap)
- EnrollmentService (orchestrates checks)
- AttendanceService (distance calculation, spoofing detection)
- GradeCalculationService (GPA/CGPA)

Testing:

- Unit tests: PrerequisiteService (recursive scenarios)
- Unit tests: ScheduleConflictService
- Unit tests: Haversine calculation
- Unit tests: Spoofing detection
- Integration tests: Enrollment flow (15+ tests)
- Integration tests: Attendance flow (15+ tests)

6.2.2 Frontend Görevleri

Academic Management Pages:

- Course Catalog Page (/courses)**
 - List all courses (cards or table)
 - Search by code or name
 - Filter by department
 - Click to view details
- Course Detail Page (/courses/:id)**
 - Course info (code, name, credits, ECTS, description)
 - Prerequisites (with links to prerequisite courses)
 - Available sections (instructor, schedule, capacity)
 - "Enroll" button for each section
 - Enrollment modal/confirmation
- My Courses Page (/my-courses) - Student**
 - List enrolled courses
 - Show section info, instructor, schedule
 - "Drop" button (with confirmation)
 - Attendance percentage for each course
 - Warning/critical indicators

Grades Page (/grades) - Student

- o List courses with grades (midterm, final, letter)
- o Show GPA and CGPA
- o "Download Transcript" button (PDF)
- o Grade statistics (chart)

Gradebook Page (/gradebook/:sectionId) - Faculty

- o List enrolled students
- o Input fields for midterm, final, homework grades
- o Auto-calculate letter grade
- o "Save Grades" button
- o Bulk actions (export, send notifications)

GPS Attendance Pages:

Start Attendance Page (/attendance/start) - Faculty

- o Select section
- o Classroom auto-selected (GPS from database)
- o Geofence radius input (default 15m)
- o Session duration input
- o "Start Session" button
- o Display QR code (backup method)
- o Real-time attendance count

Give Attendance Page (/attendance/give/:sessionId) - Student

- o Show session info (course, time, location)
- o "Give Attendance" button
 - Request GPS permission
 - Show loading spinner (getting location...)
 - Display current location on mini map (Leaflet)
 - Show distance from classroom
 - Submit to backend
 - Show success/error message
- o Alternative: "Scan QR Code" button (bonus)

My Attendance Page (/my-attendance) - Student

- o List courses with attendance stats

- For each course:
 - Total sessions
 - Attended sessions
 - Excused absences
 - Attendance percentage
 - Status badge (OK/Warning/Critical)
- Button to "Request Excuse" for absences

Attendance Report Page (/attendance/report/:sectionId) - Faculty

- Student list with attendance counts
- Attendance percentage
- Flagged students (GPS spoofing suspects)
- Export to Excel button
- Filter by date range

Excuse Requests Page (/excuse-requests) - Faculty

- List pending excuse requests
- View student info, absence date, reason
- View uploaded document
- "Approve" / "Reject" buttons with notes

GPS & Maps Components:

- GPS permission handler
- Map component (Leaflet or Google Maps)
- Distance calculator (client-side preview)
- Location accuracy indicator

Charts & Visualizations:

- Attendance chart (line chart over time)
- Grade distribution chart (bar chart)
- GPA trend chart (student)

Testing:

- Component tests: Course list, enrollment form
- Component tests: Attendance button, GPS handler
- Integration tests: Enrollment flow (bonus)

6.3 Part 2 - Teslim Edilecekler

Kod:

1. Updated backend repository (GitHub link)

- o New models, controllers, routes
- o Business logic services
- o Tests

2. Updated frontend repository (GitHub link)

- o New pages and components
- o GPS integration
- o Maps integration

Dokümantasyon:

1. API_DOCUMENTATION_PART2.md

- o Academic endpoints (~15 adet)
- o Attendance endpoints (~12 adet)
- o Request/response examples
- o Algorithm explanations (Haversine, prerequisite checking)

2. DATABASE_SCHEMA_UPDATE.md

- o Yeni tablolar (courses, sections, enrollments, attendance_*, excuse_requests)
- o Updated ER diagram

3. GPS_IMPLEMENTATION_GUIDE.md

- o GPS kullanımı
- o Haversine formula açıklaması
- o Spoofing detection yöntemleri
- o Test senaryoları

4. USER_MANUAL_PART2.md

- o Derse kayıt nasıl olunur?
- o GPS ile yoklama nasıl verilir?
- o Mazeret nasıl bildirilir?

- Ekran görüntüleri
5. TEST_REPORT_PART2.md

- Test coverage
- Test sonuçları
- Known issues

Demo Video:

- 10-15 dakikalık video
- Ders listesi ve kayıt
- Önkoşul kontrolü (başarısız senaryo)
- Başarılı kayıt
- Öğretim üyesi yoklama açma
- Öğrenci GPS ile yoklama verme (başarılı)
- GPS spoofing senaryosu (reddedilme)
- Yoklama raporları

6.4 Part 2 - Değerlendirme Kriterleri

Kriter	Puan	Açıklama
Academic Management	35	
- Course management	10	CRUD, listing, filtering
- Enrollment logic	15	Prerequisite, conflict, capacity checks
- Grade management	10	Input, calculation, transcript PDF
GPS Attendance	40	
- Session management	10	Start, close, QR code
- GPS check-in	15	Location accuracy, Haversine, validation
- Spoofing detection	10	Multiple checks, flagging
- Excuse system	5	Submit, approve/reject
Frontend	15	
- Academic pages	7	UI/UX, forms, validation
- Attendance pages	8	GPS integration, maps, real-time
Testing	5	
- Business logic tests	3	Prerequisite, conflict, distance
- API tests	2	Enrollment, attendance flows
Dokümantasyon	5	
- API docs	2	Complete, clear
- GPS guide	2	Technical explanation
- User manual	1	Screenshots, steps
TOPLAM	100	

Bonus Puanlar (+15):

- QR code alternative (scan QR for attendance): +5
- Real-time attendance dashboard (WebSocket): +5
- Advanced spoofing detection (device sensors): +3
- Attendance analytics (trends, predictions): +2

7. Part 3: Yemek Servisi, Etkinlik ve Çizelgeleme

Teslim Tarihi: 22 Aralık 2025 (Pazar), 23:59

Süre: 7 gün

Ağırlık: %25

7.1 Part 3 Hedefleri

Bu aşamada kampüs yaşamını kolaylaştıran servisler:

- Yemek rezervasyon sistemi
- QR kod ile yemek kullanımı
- Ödeme entegrasyonu
- Etkinlik yönetimi
- Otomatik ders programı oluşturma

7.2 Part 3 - Yapılacaklar Listesi

7.2.1 Backend Görevleri

Veritabanı (Yeni Tablolar):

- cafeterias** (id, name, location, capacity)
- meal_menus** (id, cafeteria_id, date, meal_type, items_json, nutrition_json, is_published)
- meal_reservations** (id, user_id, menu_id, cafeteria_id, meal_type, date, amount, qr_code, status, used_at)
- wallets** (id, user_id, balance, currency, is_active)
- transactions** (id, wallet_id, type, amount, balance_after, reference_type, reference_id, description)
- events** (id, title, description, category, date, start_time, end_time, location, capacity, registered_count, registration_deadline, is_paid, price, status)
- event_registrations** (id, event_id, user_id, registration_date, qr_code, checked_in, checked_in_at, custom_fields_json)
- schedules** (id, section_id, day_of_week, start_time, end_time, classroom_id)
- reservations** (id, classroom_id, user_id, date, start_time, end_time, purpose, status, approved_by)
- Migration ve seed data

Meal Service Endpoints:

Menus:

- GET /api/v1/meals/menus - Menü listesi (date filter)
- GET /api/v1/meals/menus/:id - Menü detayı
- POST /api/v1/meals/menus - Menü oluşturma (admin/cafeteria staff)
- PUT /api/v1/meals/menus/:id - Menü güncelleme
- DELETE /api/v1/meals/menus/:id - Menü silme

Reservations:

- POST /api/v1/meals/reservations - Yemek rezervasyonu

```
// Logic:  
1. Get user type (scholarship vs paid)  
2. If scholarship:  
    - Check daily quota (max 2 meals/day)  
    - If quota exceeded, return 400  
3. If paid:  
    - Check wallet balance  
    - If insufficient, return 400  
4. Create reservation with status='reserved'  
5. Generate unique QR code (e.g., UUID)  
6. If paid:  
    - Create pending transaction (debit)  
    - Don't deduct yet (will deduct on use)  
7. Send confirmation notification (email + push)  
8. Return reservation with QR code
```

- DELETE /api/v1/meals/reservations/:id - Rezervasyon iptali

- o Check if >= 2 hours before meal time
 - o If paid, refund to wallet
 - o Update reservation status='cancelled'
 - o Send notification

- GET /api/v1/meals/reservations/my-reservations - Rezervasyonlarım

- POST /api/v1/meals/reservations/:id/use - Yemek kullanımı (cafeteria staff)

- o Validate QR code
 - o Check if today's date matches
 - o Check if already used
 - o Mark as used (used_at = now, status='used')
 - o If paid, complete transaction (deduct from wallet)

- o Return success

Wallet:

GET /api/v1/wallet/balance - Bakiye sorgulama

POST /api/v1/wallet/topup - Para yükleme

- o Minimum amount check (e.g., 50 TRY)
- o Payment gateway integration (Stripe test mode / PayTR sandbox)
- o Create payment session
- o Return payment URL
- o Handle webhook callback:

```
POST /api/v1/wallet/topup/webhook
// Verify signature
// If payment successful:
//   - Create transaction (type='credit')
//   - Update wallet balance (atomic)
//   - Send confirmation email
```

GET /api/v1/wallet/transactions - İşlem geçmişi (pagination)

Event Management Endpoints:

Events:

GET /api/v1/events - Etkinlik listesi (filter by category, date)

GET /api/v1/events/:id - Etkinlik detayı

POST /api/v1/events - Etkinlik oluşturma (admin/event manager)

PUT /api/v1/events/:id - Etkinlik güncelleme

DELETE /api/v1/events/:id - Etkinlik silme

Registrations:

POST /api/v1/events/:id/register - Etkinliğe kayıt

- o Check capacity
- o If full, add to waitlist (bonus)

- o Generate QR code
- o Update registered_count (atomic)
- o Send confirmation email with QR code
- o Return registration

`DELETE /api/v1/events/:eventId/registrations/:regId` - Kayıt iptali

- o Update registered_count
- o If waitlist exists, notify next person
- o Send cancellation email

`GET /api/v1/events/:id/registrations` - Kayıtlı kullanıcılar (event manager)

`POST /api/v1/events/:eventId/registrations/:regId/checkin` - QR ile giriş

- o Validate QR code
- o Mark as checked_in
- o Update checked_in_at
- o Return user info

Course Scheduling Endpoints:

`POST /api/v1/scheduling/generate` - Otomatik program oluşturma (admin)

```

// Constraint Satisfaction Problem (CSP)

// Input:
- sections (course, instructor, capacity)
- classrooms (capacity, features)
- time_slots (Monday-Friday, 09:00-17:00)
- instructor_preferences (optional)

// Hard Constraints:
1. No instructor double-booking
2. No classroom double-booking
3. No student schedule conflict (based on enrollments)
4. Classroom capacity >= section capacity
5. Classroom features match course requirements

// Soft Constraints (optimize):
1. Respect instructor time preferences
2. Minimize gaps in student schedules
3. Distribute courses evenly across week
4. Prefer morning slots for required courses

// Algorithm: Backtracking with heuristics
en iyi ders planı heuristiklere dayalı backtracking yaklaşımları
ile oluşturulmalı ve tüm hard maksimum soft koşullar sağlanmalıdır.
koşullar dinamik değiştirebilmelidir

```

- Return generated schedule

- Store in database

GET /api/v1/scheduling/:scheduleId - Program görüntüleme

GET /api/v1/scheduling/my-schedule - Benim programım (öğrenci/öğretim üyesi)

- Return weekly schedule (JSON)

GET /api/v1/scheduling/my-schedule/ical - iCal export

- Generate .ics file
- Return file download

Classroom Reservations:

POST /api/v1/reservations - Derslik rezerve etme

- Check classroom availability
- Check user permissions
- Create reservation (status='pending' if needs approval)

- Notify admin for approval
- Return reservation

`GET /api/v1/reservations` - Rezervasyon listesi (filter by date, classroom, user)

`PUT /api/v1/reservations/:id/approve` - Rezervasyon onaylama (admin)

`PUT /api/v1/reservations/:id/reject` - Rezervasyon reddetme (admin)

Business Logic Services:

- PaymentService (Stripe/PayTR integration, webhook handling)
- QRCodeService (generation, validation)
- SchedulingService (CSP algorithm)
- NotificationService (email, push, SMS)

Testing:

- Unit tests: Payment service
- Unit tests: QR code generation/validation
- Unit tests: Scheduling algorithm (small example)
- Integration tests: Meal reservation flow (10+ tests)
- Integration tests: Event registration flow (8+ tests)
- Integration tests: Classroom reservation (5+ tests)

7.2.2 Frontend Görevleri

Meal Service Pages:

Menu Page (`/meals/menu`)

- Calendar view (select date)
- Show lunch and dinner menus
- Nutritional info (calories, protein, etc.)
- Vegan/vegetarian badges
- "Reserve" button for each meal
- Reservation modal (confirm details)

My Reservations Page (`/meals/reservations`)

- List upcoming and past reservations
- Display QR code for upcoming meals (full screen on click)

- o "Cancel" button (if >= 2 hours before)
- o Status badges (reserved, used, cancelled)

Wallet Page (/wallet)

- o Display current balance
- o "Add Money" button
- o Payment amount input
- o Payment method selection
- o Redirect to payment gateway
- o Transaction history table (with pagination)

QR Scanner Page (/meals/scan) - Cafeteria staff

- o QR code scanner (webcam or input field)
- o Validate QR code via API
- o Display user info and meal type
- o "Confirm Use" button
- o Success/error feedback

Event Management Pages:

Events Page (/events)

- o List upcoming events (cards)
- o Filter by category (conference, workshop, social, sports)
- o Search by title
- o Click to view details

Event Detail Page (/events/:id)

- o Event info (title, description, date, location, capacity)
- o Remaining spots
- o Registration deadline
- o Price (if paid)
- o "Register" button
- o Registration form (if custom fields required)

My Events Page (/my-events)

- o List registered events
- o Display QR code for each event

- "Cancel Registration" button
- Past events with check-in status

Event Check-in Page (/events/checkin) - Event manager

- QR scanner
- Validate registration
- Mark as checked in
- Display attendee count

Scheduling Pages:

My Schedule Page (/schedule)

- Weekly calendar view (FullCalendar or custom)
- Color-coded courses
- Show course code, instructor, room
- Click to view course details
- "Export to iCal" button

Generate Schedule Page (/admin/scheduling/generate) - Admin

- Input: semester, year
- Select sections to schedule
- "Generate Schedule" button (loading state)
- Display generated schedule alternatives
- Select and save one
- Preview before publishing

Classroom Reservations Page (/reservations)

- List available classrooms
- Filter by building, capacity
- Select classroom, date, time
- Enter purpose
- "Reserve" button
- Approval status (pending/approved/rejected)

Components:

- QR code display component (with full-screen modal)
- QR code scanner component (react-qr-reader)

- Calendar component (FullCalendar or custom)
- Payment form component
- Event card component

Testing:

- Component tests: Meal reservation form
- Component tests: QR scanner
- Component tests: Event registration

7.3 Part 3 - Teslim Edilecekler

Kod:

1. Updated backend repository
 - Meal, event, scheduling modules
 - Payment integration
 - QR code functionality
2. Updated frontend repository
 - Meal service pages
 - Event management pages
 - Scheduling pages
 - QR code scanner

Dokümantasyon:

1. API_DOCUMENTATION_PART3.md
 - Meal endpoints (~10 adet)
 - Event endpoints (~8 adet)
 - Scheduling endpoints (~6 adet)
 - Payment webhook
2. PAYMENT_INTEGRATION_GUIDE.md
 - Payment flow diagram
 - Stripe/PayTR setup
 - Webhook implementation
 - Test cards

3. SCHEDULING_ALGORITHM.md

- CSP problem definition
- Algorithm explanation
- Pseudocode
- Example solution

4. USER_MANUAL_PART3.md

- Yemek rezervasyonu nasıl yapılır?
- Para nasıl yüklenir?
- Etkinliğe nasıl kayıt olunur?
- QR kod nasıl kullanılır?
- Ekran görüntüleri

5. TEST_REPORT_PART3.md

Demo Video:

- 10-15 dakikalık video
- Menü görüntüleme ve rezervasyon
- Para yükleme (test mode)
- QR kod ile yemek kullanımı
- Etkinlik listesi ve kayıt
- QR kod ile etkinlik girişи
- Program görüntüleme

7.4 Part 3 - Değerlendirme Kriterleri

Kriter	Puan	Açıklama
Meal Service	30	
- Menu management	5	CRUD, display
- Reservation system	10	Quota, wallet, QR generation
- QR code usage	8	Scanning, validation, marking
- Wallet & payment	7	Balance, top-up, webhook
Event Management	25	
- Event CRUD	10	Create, list, detail
- Registration system	10	Capacity, QR, notifications
- Check-in system	5	QR scanning, validation
Course Scheduling	25	
- Scheduling algorithm	15	CSP constraints, optimization
- Schedule display	5	Calendar view, export
- Classroom reservation	5	Availability check, approval
Frontend	12	
- Meal pages	5	UI/UX, QR display
- Event pages	4	Listing, registration
- Schedule pages	3	Calendar, export
Testing	5	
- Payment tests	2	Webhook, transaction
- QR code tests	2	Generation, validation
- Scheduling tests	1	Algorithm correctness
Dokümantasyon	3	
- API docs	1	Complete
- Payment guide	1	Clear instructions
- Algorithm explanation	1	Detailed

Kriter	Puan	Açıklama
TOPLAM	100	

Bonus Puanlar (+12):

- Waitlist system for events: +3
 - Advanced scheduling (genetic algorithm): +5
 - SMS notifications: +2
 - Web push notifications: +2
-

8. Part 4: Final Teslim ve Entegrasyon

Teslim Tarihi: 28 Aralık 2025 (Cumartesi), 23:59

Süre: 6 gün

Ağırlık: %30

8.1 Part 4 Hedefleri

Final aşamada:

- Tüm modülleri birleştirmek
- Analytics ve raporlama
- IoT sensör entegrasyonu (bonus)
- Notification sistemi
- Polish ve optimizasyon
- Comprehensive testing
- Final dokümantasyon
- Deployment

8.2 Part 4 - Yapılacaklar Listesi

8.2.1 Backend Görevleri

Analytics & Reporting:

- GET /api/v1/analytics/dashboard - Admin dashboard istatistikleri

```
{  
    "totalUsers": 1250,  
    "activeUsersToday": 456,  
    "totalCourses": 120,  
    "totalEnrollments": 3450,  
    "attendanceRate": 87.5,  
    "mealReservationsToday": 890,  
    "upcomingEvents": 12,  
    "systemHealth": "healthy"  
}
```

GET /api/v1/analytics/academic-performance - Akademik performans

- o Average GPA by department
- o Grade distribution (A, B, C, D, F percentages)
- o Pass/fail rates
- o Top performing students
- o At-risk students (low GPA)

GET /api/v1/analytics/attendance - Yoklama analitiği

- o Attendance rate by course
- o Attendance trends over time
- o Students with critical absence rates
- o Courses with low attendance

GET /api/v1/analytics/meal-usage - Yemek kullanım raporları

- o Daily meal counts
- o Cafeteria utilization
- o Peak hours
- o Revenue (for paid users)

GET /api/v1/analytics/events - Etkinlik raporları

- o Most popular events
- o Registration rates
- o Check-in rates
- o Category breakdown

GET /api/v1/analytics/export/:type - Rapor dışa aktarma

- o Types: academic, attendance, meal, event
- o Formats: Excel, PDF, CSV

- o Generate and return file

Notification System (Enhanced):

- GET /api/v1/notifications - Bildirim listesi (pagination)
 - o Filter by category, read/unread
 - o Sort by date
- PUT /api/v1/notifications/:id/read - Okundu işaretle
- PUT /api/v1/notifications/mark-all-read - Hepsini okundu işaretle
- DELETE /api/v1/notifications/:id - Bildirimi sil
- GET /api/v1/notifications/preferences - Bildirim tercihleri
- PUT /api/v1/notifications/preferences - Tercihleri güncelle

```
{
  "email": {
    "academic": true,
    "attendance": true,
    "meal": false,
    "event": true,
    "payment": true,
    "system": true
  },
  "push": {
    "academic": true,
    "attendance": true,
    "meal": true,
    "event": true,
    "payment": true,
    "system": false
  },
  "sms": {
    "attendance": true, // critical only
    "payment": false
  }
}
```

WebSocket Implementation:

- Socket.io server setup
- Authentication for WebSocket
- Real-time notification broadcasting

- Real-time attendance updates (for faculty dashboard)
- Real-time sensor data streaming (bonus)

IoT Sensor Integration (Bonus):

- sensor_data** table (id, sensor_id, timestamp, value, unit)
- GET /api/v1/sensors - Sensör listesi
- GET /api/v1/sensors/:id/data - Sensör verisi
 - Time range filter
 - Aggregation (avg, min, max per hour/day)
- WS /api/v1/sensors/:id/stream - Real-time veri akışı
 - WebSocket endpoint
 - Stream latest sensor readings
- Simulated sensors for demo (temperature, occupancy, energy)

Background Jobs & Automation:

- Daily absence warnings (cron)
- Event reminders (1 day before, 1 hour before)
- Meal reservation reminders
- Database backup (daily)
- Log cleanup (weekly)
- Analytics data aggregation (daily)

Security & Optimization:

- Rate limiting (express-rate-limit)
- Request logging (Morgan)
- Error logging (Winston)
- Input sanitization
- SQL injection prevention check
- CORS hardening
- Database query optimization (EXPLAIN ANALYZE)
- Index creation for slow queries
- Caching (Redis - optional)

Testing:

- Complete unit test coverage (>60%)
- Complete integration test coverage (>60%)

- E2E tests for critical flows (5+ scenarios - bonus)
- Load testing (bonus)

8.2.2 Frontend Görevleri

Analytics & Reporting Pages:

- Admin Dashboard (/admin/dashboard)
 - Key metrics cards (users, enrollments, attendance rate)
 - Charts (line charts for trends, bar charts for distributions)
 - Recent activity feed
 - System health status
- Academic Performance Page (/admin/analytics/academic)
 - GPA statistics by department
 - Grade distribution charts
 - Student performance table
 - Export button
- Attendance Analytics Page (/admin/analytics/attendance)
 - Attendance rate chart (by course)
 - Time series chart (attendance over time)
 - At-risk students table
 - Export button
- Meal Analytics Page (/admin/analytics/meal)
 - Daily usage chart
 - Revenue chart (if applicable)
 - Peak hours heatmap
 - Export button

Notification Center:

- Notification Bell (component in navbar)
 - Badge with unread count
 - Dropdown with recent notifications
 - Click to view all
- Notifications Page (/notifications)

- List all notifications
- Filter by category
- Filter by read/unread
- Mark as read/unread
- Delete individual notifications
- Mark all as read button

Notification Settings Page (/settings/notifications)

- Toggle switches for each category and channel
- Email, Push, SMS columns
- Save button

IoT Dashboard (Bonus):

IoT Dashboard Page (/admin/iot)

- List all sensors with current readings
- Real-time updates (WebSocket)
- Historical data charts
- Alerts/anomalies section

Polish & UX Improvements:

- Loading states (skeletons, spinners)
- Empty states (no data messages)
- Error boundaries (React error handling)
- Toast notifications (success, error, info)
- Confirmation dialogs (before delete/cancel actions)
- Form validation feedback
- Accessibility (a11y) improvements
 - ARIA labels
 - Keyboard navigation
 - Focus management
 - Screen reader support
- Dark mode (bonus)
- Internationalization (i18n) - Turkish/English (bonus)

Performance Optimization:

- Code splitting (React.lazy)

- Lazy loading images
- Memoization (React.memo, useMemo, useCallback)
- Debouncing (search inputs)
- Pagination (infinite scroll or traditional)
- Reduce bundle size (analyze with webpack-bundle-analyzer)

Testing:

- Component tests (>40% coverage)
- Integration tests (key user flows)
- E2E tests (Cypress/Playwright - bonus)

8.2.3 DevOps & Deployment

Docker:

- Multi-stage Dockerfile (backend)
- Dockerfile (frontend)
- Docker Compose (backend + frontend + postgres + redis)
- Environment configuration (.env files)
- Health checks

CI/CD (Bonus):

- GitHub Actions workflow
 - Run tests on push
 - Build Docker images
 - Deploy to cloud (Heroku, AWS, GCP, DigitalOcean)

Database:

- All migrations tested
- Seed data for demo
- Backup script

Monitoring & Logging (Bonus):

- Application logs (Winston)
- Error tracking (Sentry - bonus)
- Performance monitoring (New Relic - bonus)

8.3 Part 4 - Teslim Edilecekler

Kod:

1. Complete backend repository (GitHub link)
 - All modules implemented
 - Tests passing
 - Deployment ready
2. Complete frontend repository (GitHub link)
 - All pages implemented
 - Tests passing
 - Production build
3. Docker Compose setup
 - Single command to run entire application
 - README with instructions

Dokümantasyon (Tüm Dökümanlar):

1. README.md (Main project README)
 - Project overview
 - Tech stack
 - Features
 - Installation guide
 - Usage guide
 - Contributors
2. ARCHITECTURE.md
 - System architecture diagram
 - Technology choices
 - Design patterns
 - Data flow
3. API_DOCUMENTATION.md (Complete)
 - All endpoints (60+ endpoints)

- Authentication
- Error codes
- Examples

4. **DATABASE_SCHEMA.md** (Complete)

- Complete ER diagram
- All tables (30+ tables)
- Indexes
- Relationships

5. **DEPLOYMENT_GUIDE.md**

- Docker setup
- Environment variables
- Database migrations
- Production deployment steps
- Troubleshooting

6. **USER_MANUAL.md** (Complete)

- For students
- For faculty
- For admins
- Screenshots and tutorials

7. **DEVELOPER_GUIDE.md**

- Project structure
- Coding conventions
- How to contribute
- Testing guide

8. **TEST_REPORT.md** (Complete)

- Total test coverage
- Test results
- Known issues
- Performance benchmarks

9. **ANALYTICS_GUIDE.md**

- Available reports
- How to interpret data
- Export options

10. PROJECT_RETROSPECTIVE.md

- What went well
- Challenges faced
- Lessons learned
- Future improvements
- Team collaboration reflection

Demo Video (Final):

- 20-30 dakikalık comprehensive demo
- User registration to graduation (full journey)
- All major features
- Admin dashboards
- Mobile responsive demo

Live Deployment (Bonus):

- Deployed to cloud (Heroku, AWS, GCP, DigitalOcean)
- Public URL provided
- Demo credentials shared

8.4 Part 4 - Değerlendirme Kriterleri

Kriter	Puan	Açıklama
Analytics & Reporting	20	
- Dashboard metrics	8	Data accuracy, charts, real-time
- Reports	8	Academic, attendance, meal, event
- Export functionality	4	Excel, PDF, CSV
Notification System	15	
- In-app notifications	5	List, read/unread, real-time
- Email notifications	5	Templates, delivery
- Notification preferences	5	User control, respect settings
Integration & Polish	25	
- Module integration	10	All parts work together seamlessly
- UI/UX polish	10	Loading states, error handling, consistency
- Performance	5	Load times, optimization
Testing	15	
- Backend coverage	8	Unit + integration (>60%)
- Frontend coverage	5	Component + integration (>40%)
- E2E tests	2	Critical flows (bonus)
Deployment	10	
- Docker setup	5	Works with one command
- Deployment guide	3	Clear instructions
- Live deployment	2	Bonus
Documentation	10	
- Completeness	5	All required docs present
- Quality	3	Clear, detailed, formatted
- Project retrospective	2	Reflective, insightful
Code Quality	5	

Kriter	Puan	Açıklama
- Clean code	2	Readable, maintainable
- Best practices	2	Security, performance, conventions
- Git history	1	Meaningful commits, branching
TOPLAM	100	

Bonus Puanlar (+20):

- WebSocket real-time features: +5
 - IoT sensor integration: +5
 - CI/CD pipeline: +3
 - Live cloud deployment: +3
 - Dark mode: +2
 - Internationalization: +2, türkçe olsaç zaten ingilizce olmak üzere en az bir dil daha
-

9. Dokümantasyon Gereksinimleri

9.1 Genel Dokümantasyon Kuralları

Format:

- Tüm dökümanlar **Markdown (.md)** formatında olmalıdır
- README.md her repository'de bulunmalıdır
- Tüm dökümanlar `docs/` klasöründe organize edilmelidir

İçerik Standartları:

- Başlıklar net ve hiyerarşik (#, ##, ###)
- Code blocks uygun syntax highlighting ile (`javascript`, `sql`)
- Tablolar düzenli formatlanmalı
- Screenshots ve diagrams kullanılmalı
- Örnekler eksiksiz ve çalışır durumda olmalı
- Referanslar ve linkler açık olmalı

Görsel Materyal:

- ER diagram (draw.io, dbdiagram.io, Lucidchart)
- System architecture diagram
- User flow diagrams
- Screenshots (en az 20 adet tüm proje için)

9.2 Zorunlu Dökümanlar

Part 1 için:

1. PROJECT_OVERVIEW.md
2. API_DOCUMENTATION.md (Part 1)
3. DATABASE_SCHEMA.md (initial)
4. USER_MANUAL_PART1.md
5. TEST_REPORT_PART1.md

Part 2 için: 6. API_DOCUMENTATION_PART2.md 7. DATABASE_SCHEMA_UPDATE.md 8. GPS_IMPLEMENTATION_GUIDE.md 9. USER_MANUAL_PART2.md 10. TEST_REPORT_PART2.md

Part 3 için: 11. API_DOCUMENTATION_PART3.md 12. PAYMENT_INTEGRATION_GUIDE.md 13. SCHEDULING_ALGORITHM.md 14. USER_MANUAL_PART3.md 15. TEST_REPORT_PART3.md

Part 4 için: 16. README.md (main) 17. ARCHITECTURE.md 18. API_DOCUMENTATION.md (complete, consolidated) 19. DATABASE_SCHEMA.md (complete, consolidated) 20. DEPLOYMENT_GUIDE.md 21. USER_MANUAL.md (complete) 22. DEVELOPER_GUIDE.md 23. TEST_REPORT.md (complete) 24. ANALYTICS_GUIDE.md 25. PROJECT_RETROSPECTIVE.md

Toplam: Minimum 25 Markdown dökümanı

9.3 API Dokümantasyon Formatı

Her endpoint için:

Endpoint Name

URL: `POST /api/v1/resource`

Authentication: Required (JWT)

Authorization: Student, Faculty

Description: Brief description of what this endpoint does.

Request Headers:

Header	Type	Required	Description
Authorization	String	Yes	Bearer {token}

Request Body:

Field	Type	Required	Description
name	String	Yes	User's full name
email	String	Yes	Valid email address

Example Request:

```
```json
{
 "name": "Ahmet Yilmaz",
 "email": "ahmet@example.com"
}
```

### Success Response (201 Created):

```
{
 "success": true,
 "data": {
 "id": "123e4567-e89b-12d3-a456-426614174000",
 "name": "Ahmet Yilmaz",
 "email": "ahmet@example.com",
 "createdAt": "2025-12-05T10:30:00Z"
 }
}
```

### Error Responses:

#### 400 Bad Request:

```
{
 "success": false,
 "error": {
 "code": "VALIDATION_ERROR",
 "message": "Invalid email format"
 }
}
```

401 Unauthorized:

```
{
 "success": false,
 "error": {
 "code": "UNAUTHORIZED",
 "message": "Invalid or expired token"
 }
}
```

---

## ## 10. Değerlendirme Kriterleri

### ### 10.1 Genel Değerlendirme Dağılımı

Bileşen	Toplam Puan	Açıklama
**Part 1**	20 puan	Auth + User Management
**Part 2**	25 puan	Academic + GPS Attendance
**Part 3**	25 puan	Meal + Event + Scheduling
**Part 4**	30 puan	Analytics + Integration + Final
**GENEL TOPLAM**	**100 puan**	

\*\*Bonus:\*\* Maximum +50 puan (toplam 150 üzerinden)

### ### 10.2 Değerlendirme Boyutları

Her part için değerlendirme kriterleri:

#### #### 10.2.1 Teknik Implementasyon (40-50%)

\*\*Backend:\*\*

- API endpoint'lerin doğru çalışması
- Business logic doğruluğu
- Error handling kalitesi
- Security best practices
- Database design ve query optimization
- Code organization ve clean code

\*\*Frontend:\*\*

- UI/UX kalitesi
- Responsive design
- State management
- Form validation
- Error handling
- Performance optimization

#### #### 10.2.2 Dokümantasyon (10-15%)

- Eksiksizlik (tüm gerekli dökümanlar mevcut)
- Kalite (açık, detaylı, örnekli)
- Format (düzgün markdown, görsellerle zenginleştirilmiş)
- Güncellilik (kod ile uyumlu)

#### #### 10.2.3 Testing (5-15%)

- Test coverage (unit + integration)
- Test quality (meaningful tests)
- Edge cases covered

- E2E tests (bonus)

#### #### 10.2.4 DevOps & Deployment (5-10%)

- Docker setup çalışır durumda
- README instructions açık
- Environment configuration
- Migration scripts
- Live deployment (bonus)

#### #### 10.2.5 Code Quality & Best Practices (5-10%)

- Clean code principles
- Consistent naming conventions
- Comments (where needed)
- No code smells
- Git commit quality
- Branching strategy

#### #### 10.2.6 Innovation & Extras (0-15%, bonus)

- Extra features beyond requirements
- Creative solutions
- Advanced algorithms
- Outstanding UI/UX
- Exceptional performance

### ### 10.3 Sürekliklik ve İlerleme Değerlendirmesi

#### \*\*GitHub Commit History (10 bonus puan):\*\*

- Düzenli commit'ler (her gün en az 2-3 commit)
- Meaningful commit messages
- İyi branching strategy
- Code review evidence (pull requests)

#### \*\*Ara Teslim Performansı:\*\*

- Her ara teslim zamanında teslim edilirse: +2 puan
- Geç teslimlerde her gün için: -10% (max 3 gün)

#### \*\*Proje Süreci:\*\*

- Projeyi son 2 güne bırakmamak önemli
- Commit history'de sürekli geliştirme görülmeli
- Last-minute rush belirtileri: Puan kesintisi

### ### 10.4 Ekip Çalışması Değerlendirmesi

#### \*\*Bireysel Katkı:\*\*

- Her öğrencinin GitHub contribution'ları incelenecuk
- Eşit dağılmamış iş yükü: Bireysel puan düzenlenmesi
- Peer evaluation (opsiyonel)

- \*\*Ekip Organizasyonu:\*\***
- Task distribution evidence
  - Communication (Slack, Discord, meetings)
  - Documentation of responsibilities

### 10.5 Sunum Değerlendirmesi (29 Aralık)

**Sunum Formatı (15-20 dakika + Q&A):\*\***

**İçerik (60%):\*\***

- [ ] Proje tanıtımı (2 min)
- [ ] Teknik mimari açıklaması (3 min)
- [ ] Live demo - tüm modüller (10-12 min)
- [ ] Challenges & solutions (2 min)
- [ ] Q&A (5 min)

**Sunum Becerileri (20%):\*\***

- Akıcı anlatım
- Zaman yönetimi
- Visual aids (slides) kalitesi
- Ekip koordinasyonu

**Demo Kalitesi (20%):\*\***

- Çalışır durumda gösterim
- Tüm major features
- Error handling
- Edge cases

**Sunum Puanı:** Toplam 20 puan (nihai nota eklenir)

---

## 11. Teslim Formatı ve Kurallar

### 11.1 GitHub Repository Yapısı

**Backend Repository:\*\***

```
smart-campus-backend/ ├── .github/ | └── workflows/ # CI/CD (bonus) └── docs/ # Tüm dökümanlar | ├── API_DOCUMENTATION.md | ├── DATABASE_SCHEMA.md | ├── DEPLOYMENT_GUIDE.md | └── ... └── src/ | ├── config/ | ├── controllers/ | ├── middleware/ | ├── models/ | ├── routes/ | ├── services/ | ├── utils/ | └── app.js └── tests/ | ├── unit/ | └── integration/ └── migrations/ └── seeders/ └── .env.example └── .gitignore └── Dockerfile └── docker-compose.yml └── package.json └── README.md
```

**Frontend Repository:\*\***

```
smart-campus-frontend/ └── docs/ # User manual, guides └── public/ └── src/ | └──
assets/ | └── components/ | └── context/ | └── hooks/ | └── pages/ | └── services/ |
└── utils/ | └── App.js | └── index.js └── tests/ └── .env.example └── .gitignore └──
Dockerfile └── package.json └── README.md
```

### ### 11.2 Teslim Checklist

\*\*Her Ara Teslim İçin:\*\*

- [ ] GitHub repository link (public veya private with access)
- [ ] README.md güncel ve detaylı
- [ ] .env.example dosyası eksiksiz
- [ ] Docker setup çalışır durumda
- [ ] Tests çalışır durumda (npm test)
- [ ] Linting hatası yok (npm run lint)
- [ ] Build başarılı (npm run build)
- [ ] Dökümanlar eksiksiz (ilgili part için)
- [ ] Demo video linki (YouTube/Drive)

\*\*Final Teslim İçin (Part 4):\*\*

- [ ] Yukarıdaki tüm maddeler
- [ ] 25 döküman tamamlanmış
- [ ] Tüm testler passing
- [ ] Test coverage raporları (coverage/)
- [ ] Live deployment URL (bonus)
- [ ] Sunum slides (PDF)

### ### 11.3 Teslim Yöntemi

\*\*Platform:\*\* LMS (öğretim yönetim sistemi) + GitHub

\*\*LMS'ye Yüklenenekler:\*\*

1. \*\*submission\_partX.txt\*\* dosyası (X = 1, 2, 3, 4):

Group Name: [Grup Adınız] Members:

- [Öğrenci 1 - Numara]
- [Öğrenci 2 - Numara]
- [Öğrenci 3 - Numara]
- [Öğrenci 4 - Numara]

GitHub Repositories:

- Backend: [URL]

- Frontend: [URL]

Docker Compose:

- Instructions: See README.md
- Command: docker-compose up

Demo Video: [YouTube/Drive URL]

Live Deployment (if any): [URL]

Notes: [Özel notlar varsa]

- demo\_video\_partX.mp4 (veya link)
- docs.zip (tüm dökümanlar ZIP'lenmiş)

## 11.4 Geç Teslim Politikası

- 0-24 saat geç: -10%
- 24-48 saat geç: -20%
- 48-72 saat geç: -30%
- 72+ saat geç: Kabul edilmez (0 puan)

İstisnalar:

- Sağlık raporu (resmi belgeli)
- Teknik sorunlar (kanıt gereklili, GitHub/LMS downtime)

## 12. Akademik Dürüstlük

### 12.1 Kabul Edilebilir Kaynaklar

Izin Verilen:

- Resmi dokümantasyon (React, Node.js, PostgreSQL, vs.)
- StackOverflow, GitHub discussions (genel sorular için)
- Tutorial'lar (temel kavramlar için)
- ChatGPT/AI araçları (sadece yardımcı olarak, anlayarak)

- Open-source libraries (lisansa uygun)
- Kendi önceki projeleriniz

**İzin Verilmeyen:**

- Başka grupların kodu (tamamını veya büyük bölümlerini)
- GitHub'da bulunan benzer projeleri kopyalamak
- Code generation tools kullanıp anlamadan yapıştırmak
- Freelancer'lardan kod satın almak
- Arkadaşlarınızın kodunu kopyalamak

## 12.2 Benzerlik Kontrolü

- Tüm kod submission'ları **plagiarism detection tool** ile kontrol edilecek
- GitHub public projeleri ile karşılaştırma yapılacak
- Gruplar arası kod benzerliği kontrol edilecek

**Tespit Edilirse:**

- İlk ihlal: Uyarı + tüm proje ödevinden 0 puan
- İkinci ihlal: Tüm projeden 0 puan + disiplin işlemi

## 12.3 AI Kullanımı Rehberi

**Doğru Kullanım:**

- "How to implement JWT authentication in Express?"
- AI açıklar, siz anlayıp kendi kodunuzu yazarsınız
- "What's wrong with this Haversine formula?"
- AI debug yapar, siz düzeltirsiniz

**Yanlış Kullanım:**

- "Write complete authentication system for me"
- "Generate entire GPS attendance module code"

**Genel Prensip:** AI'yi **öğretmen** olarak kullanın, **kod yazılıcı** olarak değil. Yazdığınız her satır kodu anlayın ve açıklayabilecek durumda olun.

## 12.4 Kaynak Belirtme

Eğer bir kod snippet'i veya algoritma alıyorsanız:

```
// Source: https://stackoverflow.com/questions/...
// Adapted from: [Author name] - [URL]
function haversineDistance(lat1, lon1, lat2, lon2) {
 // implementation
}
```

## 13. Ekstra Notlar ve İpuçları

### 13.1 Önerilen Çalışma Akışı

Günlük Rutin:

#### 1. Stand-up meeting (15 dakika)

- Dün ne yaptım?
- Bugün ne yapacağım?

#### 2. Kod yazma

- Feature development
- Bug fixing
- Testing

#### 3. Code review (30 dakika)

- Birbirinizin kodunu inceleyin
- Pull request approval

#### 4. Git commit (günde en az 2-3 commit)

- Meaningful commit messages
- Atomic commits

Haftalık:

- Sprint planning (Pazartesi)
- Sprint review (Pazar - ara teslim)
- Retrospective (ne iyi gitti, ne geliştirilebilir?)

## 13.2 Yaygın Hatalar ve Nasıl Önlenir

### Hata 1: Son güne bırakma

- "28 Aralık'ta başlarız"
- İlk günden başla, her gün ilerleme kaydet

### Hata 2: Planlamadan kod yazmaya başlamak

- "Hemen kod yazalım"
- Önce database schema, sonra API design, sonra implementation

### Hata 3: Test yazmamak

- "Sonra yazarız"
- Her feature ile birlikte test yaz (TDD - bonus)

### Hata 4: Dokümantasyon geciktirmek

- "Proje bitince yazarız"
- Her feature ile birlikte dokümante et

### Hata 5: Merge conflict'lerden kaçınmamak

- Herkes main branch'de çalışın
- Feature branches + pull requests

## 13.3 Zaman Yönetimi Önerileri

### Part 1 (4 gün):

- Gün 1: Database + Backend setup
- Gün 2: Auth endpoints
- Gün 3: Frontend (login, register, profile)
- Gün 4: Testing + Docs + Polish

### Part 2 (7 gün):

- Gün 1-2: Academic management backend
- Gün 3-4: GPS attendance backend (en kritik!)
- Gün 5-6: Frontend pages
- Gün 7: Testing + Docs + Polish

### **Part 3 (7 gün):**

- Gün 1-2: Meal service (backend + payment)
- Gün 3-4: Event management
- Gün 5-6: Scheduling (algorithm - zor!)
- Gün 7: Testing + Docs + Polish

### **Part 4 (6 gün):**

- Gün 1-2: Analytics + Notifications
- Gün 3-4: Integration + Bug fixes
- Gün 5: Final docs + Deployment
- Gün 6: Sunum hazırlığı + Rehearsal

## **13.4 Yardım Alma**

### **İlk Kaynak:**

1. Resmi dokümantasyon
2. Google / StackOverflow
3. ChatGPT (anlayarak kullan)
4. Grup arkadaşların

### **Office Hours:**

- [Gün ve Saat] - Instructor office hours
- Email: [Email adresi]
- LMS Forum: Genel sorular için

### **Soru Sorarken:**

- Spesifik ol ("Kod çalışmıyor" yerine "JWT token expire olmuyor, kod: ...")
- Hata mesajlarını paylaş
- Denediğin çözümleri açıkla
- Minimal reproducible example ver

## **13.5 Motivasyon ve Ekip Çalışması**

### **Ekip Dinamikleri:**

- Herkes eşit katkı sağlamalı

- Sorumlulukları net belirleyin
- Düzenli iletişim (daily standups)
- Conflict'leri erken çözün
- Birbirinizi destekleyin

#### Tükenmişlik Önleme:

- Düzenli molalar (Pomodoro: 25 min çalış, 5 min mola)
- Uyku düzeninizi bozmayın
- Günlük hedefler belirleyin (overwhelming olmasın)
- Başarıları kutlayın (küçük de olsa)

**Son Söz:** Gerçek dünya deneyimi kazanacaksınız. Yıllar sonra portfolio'nuzda gösterebileceğiniz bir proje olacak. **Başarılılar!** 

---

## 14. Sık Sorulan Sorular (FAQ)

\*

**Q: React zorunlu mu? Vue kullanabilir miyiz?** A: Evet, React zorunlu. Framework standardizasyonu için.

**Q: PostgreSQL yerine MongoDB kullanabilir miyiz?** A: Hayır, ilişkisel veritabanı (PostgreSQL) zorunlu.

**Q: Payment integration için gerçek API key gerekliliği?** A: Hayır, test/sandbox mode yeterli (Stripe test mode, PayTR sandbox).

**Q: GPS için gerçek cihaz gerekliliği?** A: Evet, tarayıcı GPS API'si gerçek cihazda test edilmeli. Desktop'ta simüle edilebilir (Chrome DevTools).

**Q: Dokümantasyonda İngilizce mi Türkçe mi kullanmalıyız?** A: Türkçe, ancak teknik terimler İngilizce olmalı.

**Q: Ara teslim yapmayı unuttuk, ne olur?** A: O part'tan 0 puan alırsınız. Mutlaka ara teslimleri yapın!

**Q: Live deployment zorunlu mu?** A: Evet, not açıklanan kadar sürekli açık kalacak test edilebilecek yapılan kısımlar. Docker ile local deployment da zorunlu.

Q: ChatGPT vb. ile kod yapsak sorun olur mu? A: Yardımcı olarak kullanabilirsiniz ama kodu anlamalısınız. Kör kopya yasak.

Q: Bonus feature'ları yapmak zorunda mıyız? A: Hayır, ama extra puan ister ve vakit varsa yapın.

## 15. Değerlendirme Rubric Özeti

Part	Deadline	Ağırlık	Ana Özellikler	Bonus Maks
Part 1	8 Ara Pazar 23:59	20%	Auth + User Mgmt	+10
Part 2	15 Ara Pazar 23:59	25%	Academic + GPS Attendance	+15
Part 3	22 Ara Pazar 23:59	25%	Meal + Event + Schedule	+12
Part 4	28 Ara Cmt 23:59	30%	Analytics + Integration	+20
Sunum	29 Ara Pazar	+20	Live Demo + Q&A	-
TOPLAM		100%	+ Sunum 20	+57 bonus

Maximum Proje Puanı: 100 + 20 (sunum) + 57 (bonus) = 177 puan Passing Score: 60/10

👉 Başarılar dileriz! Bu projeyi tamamladığınızda, yapay zeka ile dahi olsa :) gerçek dünya senaryolarında modern web uygulaması geliştirebilecek deneyime sahip olacaksınız.

### 📅 Önemli Tarihler:

- 5 Aralık: Proje başlangıç
- 8 Aralık: Part 1 teslim
- 15 Aralık: Part 2 teslim
- 22 Aralık: Part 3 teslim
- 28 Aralık: Final teslim
- 29 Aralık: Sunumlar

Good luck and happy coding! 💻🚀

\*Dr. Öğretim Mehmet Sevri