

# SchoolWeb Projesi Nedir?

SchoolWeb, bir eğitim kurumunun öğrenci, öğretmen ve ders yönetimini kolaylaştırmak amacıyla geliştirilmiş bir web tabanlı uygulamadır. Bu proje, bir eğitim yönetim sistemine ihtiyaç duyan okullar ve eğitim kurumlarına yönelik çözümler sunar. Kullanıcılar (Admin, Superadmin, User) rollerine göre yetkilendirilir ve öğrenci kayıt işlemleri, ders atamaları, öğretmen görev dağılımı gibi temel eğitim süreçleri yönetilebilir hale getirilir.

Proje, eğitim kurumlarının dijital dönüşümünü hızlandırmayı ve eğitim süreçlerini merkezi bir platformda toplamayı hedefler.

## SchoolWeb Infographic

1

### Project Overview

SchoolWeb, eğitim kurumları için tasarlanan bir yönetim platformudur. Öğrenci, öğretmen ve ders yönetimi gibi temel işlevlerin dijitalleştirilmesi amacıyla geliştirilmiştir.



2

### Relational Database Design

Uygulamanın kalbi olarak veri tabanı ilişkisel model kullanılarak tasarlandı. Öğrenci, öğretmen ve ders tabloları arasında güçlü ilişkiler kurularak veri bütünlüğü sağlandı.



3

### Role-Based Authorization

Kullanıcılar, rollerine göre farklı yetkilere sahiptir. Superadmin, admin ve user seviyelerindeki kullanıcılar için farklı erişim izinleri yapılandırıldı.



4

### API Integration

Veri tabanı işlemleri API üzerinden gerçekleştirildi. Web application ile API entegrasyonu sayesinde veri çekme ve gönderme işlemleri RESTful mimarisi kullanılarak sağlandı.

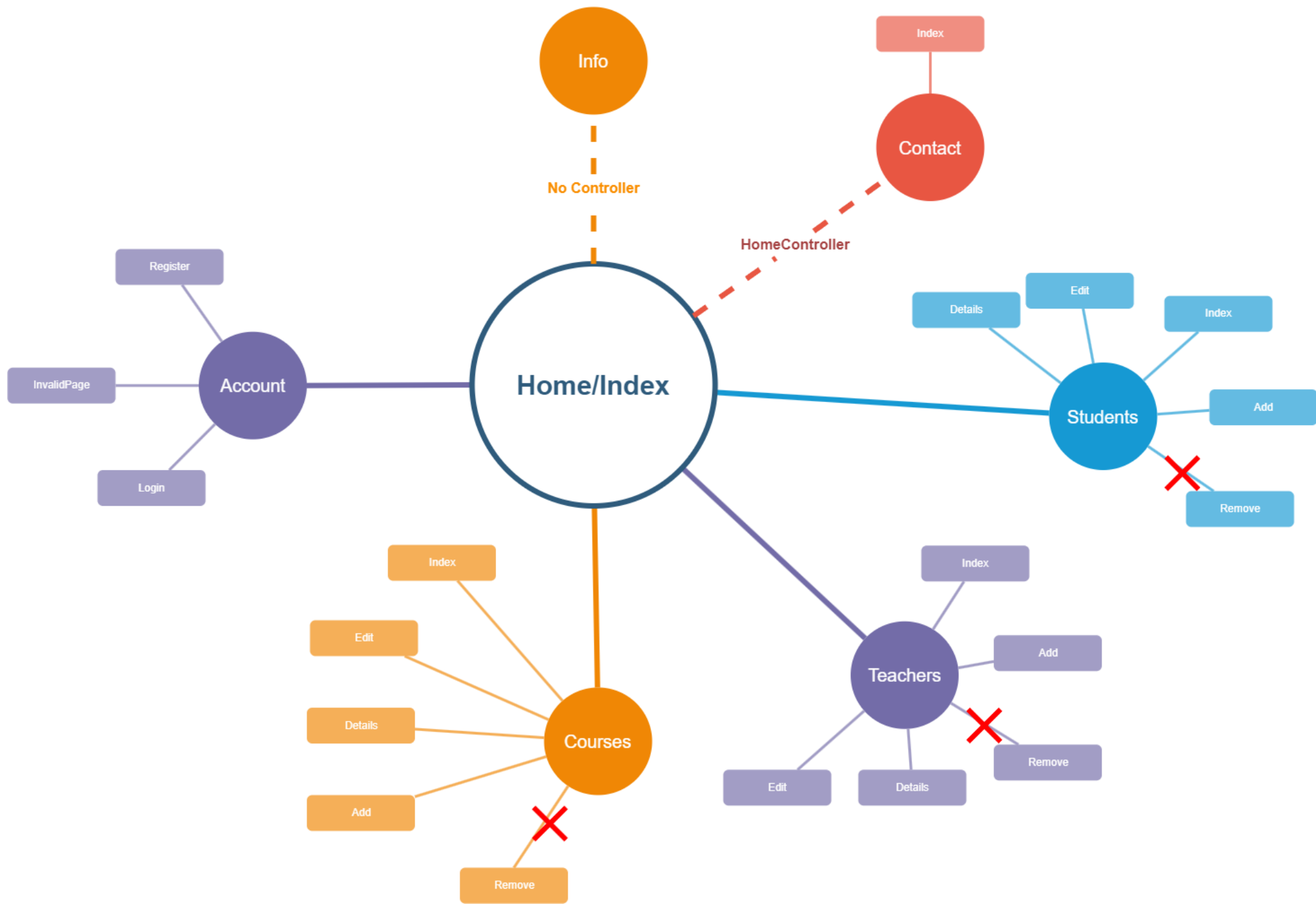


5

### UI and Performance Enhancements

UI geliştirmeleri ve Redis ile cache optimizasyonu yapıldı. Böylece sayfa yüklenme süreleri azaltıldı, veri çekme hızlandırıldı ve kullanıcı deneyimi iyileştirildi.





# Projenin Amaçları

- Eğitim kurumlarında öğrenci ve ders takibini dijitalleştirerek manuel işlemlerin minimize edilmesi.
- Öğretmen, öğrenci ve ders yönetimini tek bir platformda buluşturarak eğitim süreçlerinin daha verimli hale getirilmesi.
- Kullanıcıların yetkilendirilmesi ile güvenli ve kontrollü bir yönetim sağlanması.
- Kullanıcı dostu bir arayüz ile kolay kullanılabilirlik.

# Projenin Hedefleri

- Öğrenci, öğretmen ve ders yönetimi süreçlerinin web tabanlı bir platformda entegre edilmesi.
- Eğitim kurumlarının ihtiyaç duyduğu veri tabanı, yetkilendirme ve roller bazında kontrol mekanizmalarının geliştirilmesi.
- Performans odaklı bir sistem sunarak, hız ve güvenliği bir arada sağlamak.
- İleride eklenebilecek yeni modüller ile projenin genişletilebilir olmasını sağlamak (örneğin: sınav yönetimi, not sistemi vb.).

# SchoolWeb'in Çözdüğü Problemler

- Öğrenci, öğretmen ve ders kayıtlarının dijital ortamda tutulması.
- Roller bazında yetkilendirme ile kullanıcıların sadece kendi yetkileri dahilindeki işlemleri yapabilmesi.
- Ders ve öğretmen atamalarının otomatikleştirilmesi ve hataların minimize edilmesi.

# Proje Mimari Yapısı

**Genel Mimari Tasarım** SchoolWeb projesi, katmanlı mimari (Layered Architecture) ile tasarlanmış olup, API ve Web Application olarak iki farklı yapıdan oluşur.

- **Web Application:** Kullanıcı arayüzünün (UI) sağlandığı ve kullanıcıların sistemle etkileşimde bulunduğu kısımdır.
- **API Katmanı:** Veritabanı ile Web Application arasında veri alışverişini sağlar. CRUD işlemlerini gerçekleştiren RESTful API kullanılarak geliştirilmiştir.

## Mimari Yapıda Kullanılan Teknolojiler

- **Backend:** ASP.NET MVC
- **Frontend:** HTML, CSS, JavaScript, Bootstrap
- **Veritabanı:** MSSQL, Entity Framework
- **API:** RESTful API
- **Cache:** Redis
- **Diğer:** SMTP ile mail yönetimi, SignalR ile websocket denemesi.

# Kullanıcı Roller ve Yetkilendirme

SchoolWeb projesinde üç ana rol bulunmaktadır:

- **Superadmin:** Sistemde en geniş yetkiye sahip olan roldür. Öğrenci, öğretmen ve ders ekleme, silme, güncelleme işlemleri gerçekleştirebilir.
- **Admin:** Yetkisi user'dan fazla, superadmin'den azdır.
- **User:** Öğrenci bilgilerini görüntüleyebilir, ders kayıtlarını kontrol edebilir.

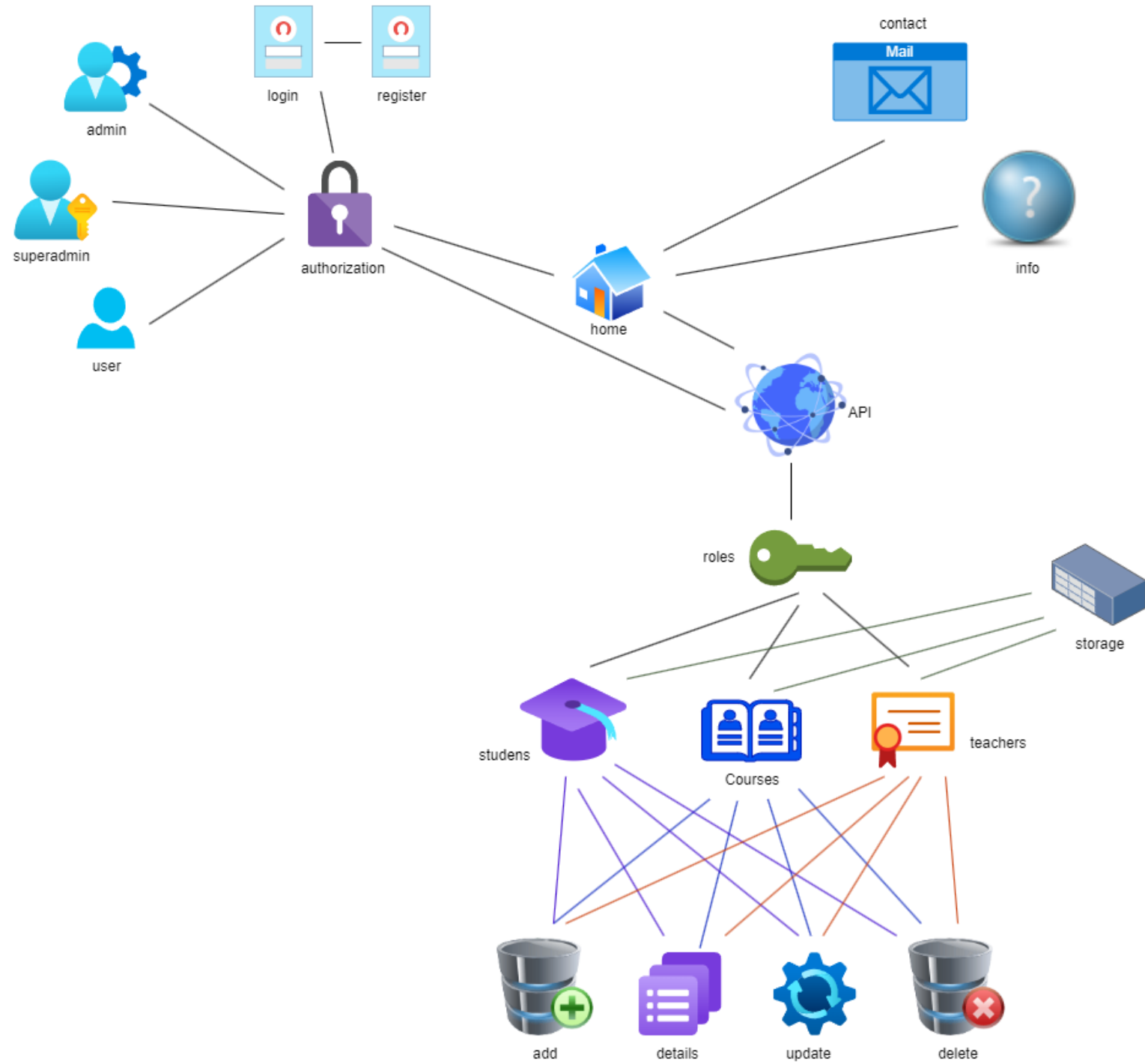
Yetkilendirme işlemleri, kullanıcı rolleriyle entegre edilen **Authorize** mekanizması ile sağlanmıştır. Her kullanıcı, sadece yetkisi dahilinde işlemler gerçekleştirebilir.



# Teknik Detaylar ve Kullanılan Teknolojiler

## Kullanılan Teknolojiler

- **.NET Framework ve ASP.NET MVC:** Projenin ana omurgasını oluşturur. MVC yapısı ile veri, kontrol ve görselleştirme katmanları arasında net bir ayrım sağlanır.
- **Entity Framework:** Veritabanı işlemlerinde kullanılan ORM aracıdır. Veritabanı ile uygulama arasında veri aktarımı sağlar.
- **HTML, CSS, JavaScript ve Bootstrap:** Kullanıcı arayüzünün (UI) tasarımında kullanılmıştır.
- **Redis Cache:** Öğrenci, öğretmen ve ders bilgilerini cache'de tutarak API performansını artırır.
- **SMTP (Gmail):** Kullanıcılar tarafından gönderilen iletişim maillerini SMTP protokolü üzerinden yönlendirme sağlar.



# API Yapısı ve İşleyişi

## API Tasarımı

- SchoolWeb API, RESTful prensipler doğrultusunda tasarlanmıştır.
- Öğrenci, öğretmen ve ders işlemlerini gerçekleştiren endpointler bulunur.
- CRUD (Create, Read, Update, Delete) işlemleri API üzerinden yönetilir.
- **Soft Delete** mantığı, öğretmen ve ders silme işlemlerinde kullanılmıştır.

## API – Web App Entegrasyonu

- Web Application, API'dan gelen verilerle çalışır. AJAX ve JavaScript kullanılarak veri, sayfa yenilenmeden kullanıcıya sunulur(dropdowns).

# Başarılar ve Karşılaşılan Zorluklar

- **Veritabanı Yönetimi:** Öğrenci, öğretmen ve ders bilgileri başarılı bir şekilde ilişkilendirilerek yönetilmiştir.
- **UI Geliştirmeleri:** Bootstrap kullanılarak modern ve kullanıcı dostu bir arayüz oluşturulmuştur.
- **Cache Yönetimi:** Redis ile cache kullanımı API performansını artırmıştır.

## Karşılaşılan Zorluklar

- **SignalR Entegrasyonu:** WebSocket ile veri aktarımı denemeleri yapılmış fakat uygulamaya başarılı bir şekilde eklenememiştir.
- **Responsive Tasarım:** Bazı sayfalarda responsive tasarım doğru çalışmıyor.

# Proje Sonuçları ve Değerlendirme

## Başarıyla Tamamlanan Özellikler

- Öğrenci, öğretmen ve ders yönetimi sisteminin başarılı bir şekilde uygulanması.
- Veri tabanı performansının artırılması ve cache kullanımı.

## Eksiklikler

- SignalR ile sayfa yenilenmeden veri çekme özelliğinin projeye eklenememesi.
- UI performansının bazı durumlarda iyileştirmeye ihtiyaç duyması.

# Projenin Geleceği ve Geliştirme Alanları

## Potansiyel Geliştirmeler

- **SignalR Entegrasyonu:** Gerçek zamanlı veri güncellemeleri için yeniden planlanabilir.
- **Performans Optimizasyonu:** Gelişen veri tabanı büyüklüğüyle birlikte performans iyileştirmelerine ihtiyaç duyulabilir(İndeksleme).
- **Ek Modüller:** Sınav, not yönetimi gibi ek modüller eklenerek proje genişletilebilir.

# SchoolWeb Timeline

## Week1

- .NET Framework dinamikleri üzerine çalışmalar
- Web sitesi tasarımı ve .NET yapısındaki Controller, UI dosyaları, dosya hiyerarşisi
- Örnek bir Mobile Store web sitesi üzerinde çalışmalar
- ADO.NET ve ASP.NET kullanılarak CRUD (Create, Read, Update, Delete) Projesi

## Week2

- SchoolWeb Uygulaması Proje Başlangıcı
- SchoolWeb İlişkisel Veri Tabanı Tasarımı
- SchoolWeb Veri Tabanı Modelinin Koda Aktarımı
- SchoolWeb Authorize İşlemleri Üzerinde Çalışmalar ve Başarısızlıklar
- SchoolWeb Authorize Çözümü ve Veri Tabanı Değişikliği

## Week3

- SchoolWeb DB Yapılandırması ve Veri Çekme
- SchoolWeb Ekleme, Okuma, Detay ve Silme Operasyonları
- SchoolWeb Global.asax Yapısı ve Güncelleme Methodları
- SchoolWeb Update ve Conflict Çözümleri
- SchoolWeb Kullanıcı Rol Yetkileri ve UI Geliştirmeleri

## Week4

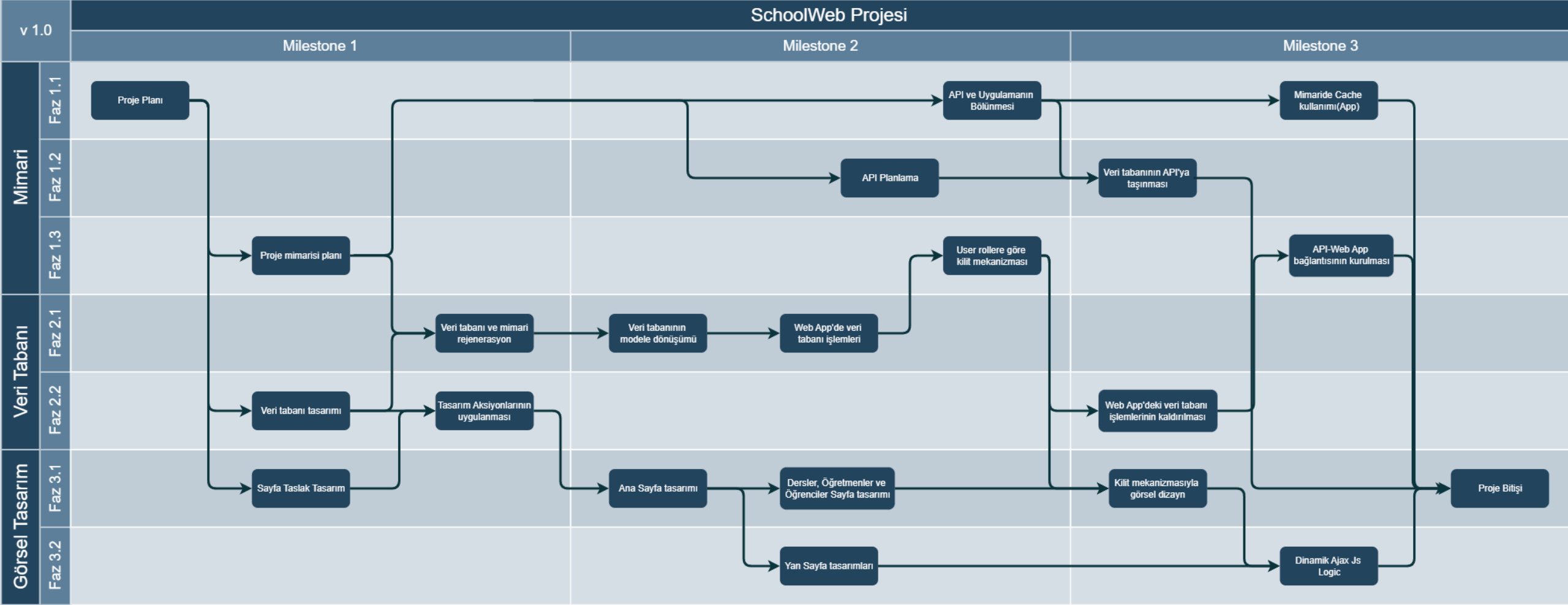
- SchoolWeb Authorization ve Son UI Güncellemeleri
- SchoolWeb API Planlama ve Çalışmaları
- SchoolWeb API Modellemesi ve Method Eklemesi
- SchoolWeb API – Web Application Entegrasyonu

## Week5

- SchoolWeb API - Web App Student Section
- Dinamik Öğretmen ve Ders Atama Problemi
- Cache Entegrasyonu ve Redis Kullanımı
- SMTP ile Mail Yönetimi ve İletişim Formu Entegrasyonu
- SignalR ile WebSocket Denemesi ve Kod Bakımı

## Week6

- SchoolWeb Proje Sunum ve Dokümantasyon





-- Roles Tablosu

```
CREATE TABLE Roles (  
  RoleId INT PRIMARY KEY,  
  RoleName NVARCHAR(50) NOT NULL  
);
```

-- Users Tablosu

```
CREATE TABLE Users (  
  UserId INT PRIMARY KEY IDENTITY(1,1),  
  Username NVARCHAR(50) NOT NULL UNIQUE,  
  Password NVARCHAR(100) NOT NULL,  
  RoleId INT NOT NULL FOREIGN KEY REFERENCES Roles(RoleId)  
);
```

-- Students Tablosu

```
CREATE TABLE Students (  
  StudentId INT PRIMARY KEY IDENTITY(1,1),  
  StudentName NVARCHAR(100) NOT NULL,  
  IsActive BIT DEFAULT 1,  
  IsDeleted BIT DEFAULT 0,  
  CreatedDate DATETIME DEFAULT GETDATE(),  
  UpdatedDate DATETIME NULL,  
  DeletedDate DATETIME NULL  
);
```

-- Courses Tablosu

```
CREATE TABLE Courses (  
  CourseId INT PRIMARY KEY IDENTITY(1,1),  
  CourseName NVARCHAR(100) NOT NULL,  
  IsActive BIT DEFAULT 1,  
  IsDeleted BIT DEFAULT 0,  
  CreatedDate DATETIME DEFAULT GETDATE(),  
  UpdatedDate DATETIME NULL,  
  DeletedDate DATETIME NULL  
);
```

-- Teachers Tablosu

```
CREATE TABLE Teachers (  
  TeacherId INT PRIMARY KEY IDENTITY(1,1),  
  TeacherName NVARCHAR(100) NOT NULL,  
  CourseId INT NOT NULL FOREIGN KEY REFERENCES Courses(CourseId),  
  IsActive BIT DEFAULT 1,  
  IsDeleted BIT DEFAULT 0,  
  CreatedDate DATETIME DEFAULT GETDATE(),  
  UpdatedDate DATETIME NULL,  
  DeletedDate DATETIME NULL  
);
```


-- StudentCourses Tablosu


```
CREATE TABLE StudentCourses (  
  StudentCourseId INT PRIMARY KEY IDENTITY(1,1),  
  StudentId INT,  
  CourseId INT,  
  TeacherId INT,  
  CreatedDate DATETIME DEFAULT GETDATE(),  
  UpdatedDate DATETIME NULL,  
  DeletedDate DATETIME NULL,  
  IsActive BIT DEFAULT 1,  
  IsDeleted BIT DEFAULT 0,  
  CONSTRAINT FK_StudentCourses_Student FOREIGN KEY (StudentId) REFERENCES  
  Students(StudentId),  
  CONSTRAINT FK_StudentCourses_Course FOREIGN KEY (CourseId) REFERENCES Courses(CourseId),  
  CONSTRAINT FK_StudentCourses_Teacher FOREIGN KEY (TeacherId) REFERENCES  
  Teachers(TeacherId),  
  CONSTRAINT UQ_StudentCourse UNIQUE (StudentId, CourseId) -- Unique Constraint  
);
```


```
-- veri ekleme .....
INSERT INTO Roles (RoleId, RoleName)
VALUES
(0, 'SuperAdmin'), -- RoleId 0
(1, 'Admin'), -- RoleId 1
(2, 'User'); -- RoleId 2
INSERT INTO Users (Username, Password, RoleId)
VALUES
('Ahmet', 'ahmet123', 0), -- SuperAdmin
('Mehmet', 'memed123', 1), -- Admin
('Haydar', 'haydaririri', 2); -- User
INSERT INTO Students (StudentName, CreatedDate, IsActive)
VALUES
('Ali Veli', GETDATE(), 1),
('Ayşe Fatma', GETDATE(), 1),
('Mehmet Can', GETDATE(), 1);
```


```
INSERT INTO Courses (CourseName, CreatedDate, IsActive)
VALUES
('Matematik', GETDATE(), 1),
('Fizik', GETDATE(), 1),
('Kimya', GETDATE(), 1);
INSERT INTO Teachers (TeacherName, CourseId, CreatedDate, IsActive)
VALUES
('Ahmet Öğretmen', 1, GETDATE(), 1), -- Matematik
('Selin Öğretmen', 2, GETDATE(), 1), -- Fizik
('Ayşe Öğretmen', 2, GETDATE(), 1), -- Fizik bu hoca da
('Ayhan Öğretmen', 3, GETDATE(), 1); -- Kimya
INSERT INTO StudentCourses (StudentId, CourseId, TeacherId, CreatedDate, IsActive)
VALUES
(1, 1, 1, GETDATE(), 1), -- Ali Veli -> Matematik -> Ahmet Öğretmen
(1, 2, 2, GETDATE(), 1), -- Ali Veli -> Fizik -> Selin Öğretmen
(2, 2, 2, GETDATE(), 1), -- Ayşe Fatma -> Fizik -> Selin Öğretmen
(2, 3, 4, GETDATE(), 1), -- Ayşe Fatma -> Kimya -> Ayhan Öğretmen
(2, 1, 1, GETDATE(), 1), -- Ayşe Fatma -> Matematik -> Ahmet Öğretmen
(3, 1, 1, GETDATE(), 1), -- Mehmet Can -> Matematik -> Ahmet Öğretmen
(3, 2, 3, GETDATE(), 1); -- Mehmet Can -> Fizik -> Ayşe Öğretmen
-- hata verecek çünkü öğrenci aynı dersten 2 kez seçemez
INSERT INTO StudentCourses (StudentId, CourseId, TeacherId, CreatedDate, IsActive)
VALUES
(2, 2, 3, GETDATE(), 1); -- Ayşe Fatma -> Fizik -> Ayşe Öğretmen
```


Users	
	UserId
	Username
	Password
	RoleId

Roles	
	RoleId
	RoleName

StudentCourses	
	StudentCourseId
	StudentId
	CourseId
	TeacherId
	CreatedDate
	UpdatedDate
	DeletedDate
	IsActive
	IsDeleted

Students	
	StudentId
	StudentName
	CreatedDate
	UpdatedDate
	DeletedDate
	IsActive
	IsDeleted

Teachers	
	TeacherId
	TeacherName
	CourseId
	CreatedDate
	UpdatedDate
	DeletedDate
	IsActive
	IsDeleted

Courses	
	CourseId
	CourseName
	CreatedDate
	UpdatedDate
	DeletedDate
	IsActive
	IsDeleted

