Dinamik Sınav Takvimi

Hayrunisa KORKULU ve Farahnozkhon Dilovarovna MASUMOVA
Kocaeli Üniversitesi
YAZILIM LAB. I PROJE RAPORU

***Özet*—Bu çalışma, üniversitelerde sınav planlama süreçlerini dijital ortama taşımak ve manuel planlamada ortaya çıkan hataları ortadan kaldırmak amacıyla geliştirilen Dinamik Sınav Takvimi Otomasyonu sistemini tanıtmaktadır. Sistem, bölüm koordinatörlerinin ve yöneticilerin sınav tarihlerini, derslik atamalarını ve öğrenci çakışmalarını dikkate alarak otomatik olarak sınav programı üretmesini sağlamaktadır.**

Geliştirilen masaüstü uygulaması, Python programlama dili ve PySide6 kütüphanesi kullanılarak tasarlanmış olup, tüm veriler yerel `SQLite` veritabanında saklanmaktadır. Sınav planlama algoritması; öğrenci-ders ilişkilerini, derslik kapasitelerini, tatil günlerini ve zaman kısıtlarını dikkate alarak hatasız bir sınav takvimi oluşturmaktadır. Ayrıca sistem, oluşturulan sınav programlarını Excel formatında dışa aktararak kullanıcıya kolay arşivleme ve paylaşım olanağı sunmaktadır.

Uygulama testleri sonucunda, sistemin sınav çakışmalarını %100 oranında tespit ettiği, planlama süresinde yaklaşık %70 zaman tasarrufu sağladığı ve çoklu bölüm desteğiyle yönetim sürecini kolaylaştırdığı görülmüştür.

Anahtar Kelimeler — Sınav planlama, Python, PySide6, SQLite, Otomasyon, Excel çıktısı, Veritabanı.

I. Giriş

Üniversitelerde sınav takvimi oluşturma süreci, çok sayıda değişkenin aynı anda dikkate alınması gereken karmaşık bir süreçtir. Her bölümde farklı dersler, öğretim elemanları, öğrenciler ve derslikler bulunduğundan, sınavların zamanlamasını elle planlamak çoğu zaman hatalara yol açmaktadır. Özellikle öğrenci çakışmaları, derslik kapasite yetersizlikleri ve akademik takvime uygun olmayan tarih atamaları gibi problemler, sınav dönemlerinde yönetimsel zorluklara neden olmaktadır.

Sınav planlamasının manuel yöntemlerle yapılması, hem zaman alıcı hem de hataya açık bir süreçtir. Planlayıcılar genellikle Excel veya basit tablo yazılımları kullanarak ders, öğretim üyesi ve öğrenci bilgilerini düzenlemeye çalışmakta; ancak veri hacmi büyüdükçe bu yöntem sürdürülemez hale gelmektedir. Hatalı veri girişleri, sınav çakışmaları ve kapasite aşımı gibi durumlar, planlama sürecinin tekrar tekrar gözden geçirilmesini gerektirmekte ve önemli ölçüde zaman kaybına yol açmaktadır.

Manuel planlamada sık karşılaşılan hatalar şu şekilde özetlenebilir:

- Aynı öğrencinin iki farklı dersinin sınavının aynı tarih veya saate denk gelmesi,
- Derslik kapasitesinin sınava katılacak öğrenci sayısından düşük olması,
- Tatil veya hafta sonu günlerine sınav atanması,
- Sınavların dengesiz şekilde dağılması (bazı günlerde çok fazla, bazı günlerde çok az sınav olması).

Bu nedenlerle, sınav programlarının dijital ortama taşınması ve otomatikleştirilmesi kaçınılmaz hale gelmiştir. Bu proje kapsamında geliştirilen **Dinamik Sınav Takvimi Otomasyonu** sistemi, sınav tarihlerini, saatlerini ve derslik atamalarını belirli kurallar çerçevesinde otomatik olarak oluşturmakta ve potansiyel hataları en aza indirmektedir. Sistem, kullanıcı rollerine göre yetkilendirme yaparak yönetici (admin) ve bölüm koordinatörlerinin farklı yetkilere sahip olmasını sağlamaktadır. Yönetici tüm bölümleri ve kullanıcıları yönetebilirken, koordinatör yalnızca kendi bölümündeki dersleri, öğrencileri ve sınav takvimini düzenleyebilmektedir.

Geliştirilen sistem; Python, PySide6 ve SQLite teknolojileri kullanılarak masaüstü uygulaması şeklinde tasarlanmıştır. Kullanıcı dostu arayüzü sayesinde sınav planlama işlemleri kolayca gerçekleştirilebilmekte, sınav çakışmaları ve kapasite aşımları sistem tarafından otomatik olarak tespit edilmektedir. Oluşturulan sınav programı, Excel dosyası formatında dışa aktarılabilmekte ve arşivleme sürecinde kullanılabilmektedir.

Sonuç olarak, bu proje, üniversitelerde sınav planlama sürecini dijitalleştirerek hem zaman hem de emek tasarrufu sağlamayı amaçlamaktadır. Aynı zamanda, insan hatasını ortadan kaldırarak adil, dengeli ve yönetilebilir bir sınav takvimi oluşturulmasına katkı sunmaktadır.

II. Yöntem

Bu bölümde, Dinamik Sınav Takvimi Otomasyonu projesinin geliştirilmesinde kullanılan teknolojiler, sistemin genel mimarisi, veri işleme süreci ve planlama algoritması detaylı şekilde açıklanmaktadır. Sistem, çok katmanlı bir yazılım mimarisi üzerine inşa edilmiştir ve modüler yapısı sayesinde hem genişletilebilir hem de bakımı kolaydır. Her bileşen belirli bir sorumluluğa sahip olacak şekilde tasarlanmış; veri yönetimi, işlem mantığı ve kullanıcı etkileşimi birbirinden bağımsız olarak yürütülmüştür.

A. Kullanılan Teknolojiler

Proje geliştirme sürecinde modern yazılım teknolojilerinden yararlanılmıştır. Kullanılan başlıca araç ve kütüphaneler Tablo I'te özetlenmiştir.

Tablo I: Kullanılan Teknolojiler

Bileşen	**Teknoloji / Açıklama**
Arayüz	PySide6 (Qt Framework) – Modern masaüstü arayüz geliştirme
Veri Tabanı	SQLite3 – Yerel, güvenilir ve hızlı veritabanı sistemi
Programlama Dili	Python 3.11 – Nesne yönelimli ve modüler yapı
Çıktı	Excel (xlsxwriter) – Sınav programlarının dışa aktarımı
Şifreleme	bcrypt – Kullanıcı parolalarının güvenli saklanması

Python'un modüler yapısı, proje içinde farklı sorumlulukların ayrı dosyalarda yönetilmesine olanak sağlamıştır. PySide6 sayesinde etkileşimli ve kart tabanlı bir masaüstü arayüz geliştirilmiş; SQLite veritabanı kullanılarak tüm işlemler internet bağlantısına gerek duyulmadan yerel olarak gerçekleştirilmiştir. Ayrıca Excel çıktısı üretimi, bölümler arası iletişimi kolaylaştırmış ve sınav takviminin dış sistemlerde paylaşılabilmesine olanak tanımıştır.

B. Genel Sistem Mimarisi

Geliştirilen sistem üç temel katmandan oluşmaktadır: veri katmanı, uygulama katmanı ve arayüz katmanı. Her katman kendi içerisinde belirli sorumluluklara sahip olup, katmanlar arası iletişim açık ve kontrollü arabirimler üzerinden sağlanmaktadır. Bu yapı, sistemin ölçeklenebilirliğini ve hata toleransını artırmıştır.

- **Veri Katmanı (Data Layer):** Bu katman, sistemin temel veri yapısını ve tüm veritabanı işlemlerini yönetir. SQLite üzerinde `bolumler`, `kullanicilar`, `dersler`, `ogrenciler`, `kayitlar` ve `sinavlar` tabloları oluşturulmuştur. İlişkisel model, `Foreign Key` bağlantılarıyla veri tutarlılığını koruyacak şekilde tasarlanmıştır. Her bir tablo arasında bire-bir veya bire-çok ilişkiler tanımlanmıştır (örneğin, bir bölümün birden fazla dersi ve öğrencisi olabilir). Veritabanı işlemleri, doğrudan sorgu çalıştırmak yerine `get_conn()` ve `q()` fonksiyonları aracılığıyla soyutlanmıştır. Bu yöntem, veriye erişimi kontrol altına alarak güvenliği artırmış, hatalı sorgu riskini azaltmıştır. Ayrıca sistemde, sınav zamanları ve kısıtlamalar (`kisitlar`) tabloları dinamik olarak güncellenebilir şekilde tasarlanmış, böylece akademik takvim değişikliklerine sistem yeniden başlatılmadan uyum sağlanmıştır.
- **Uygulama Katmanı (Application Layer):** Sistemin mantıksal işlemleri bu katmanda gerçekleştirilmiştir. Python tabanlı modüller arasında veri akışı tanımlanmış, her dosya belirli bir sorumluluk üstlenmiştir. `session.py` kullanıcı oturumlarını yönetmekte, kullanıcı rolünü doğrulamakta ve erişim kontrolü sağlamaktadır. `schedule.py` modülü, sistemin en önemli bileşenidir ve sınav planlama algoritmasını içermektedir. Bu algoritma; öğrenci–ders kayıtlarını, sınıf kapasitesini, tatil günlerini ve paralel sınav kısıtlarını değerlendirerek çakışmasız bir sınav takvimi üretir. Çakışma kontrolü için her öğrenciye ait ders listesi dinamik olarak sorgulanmakta, aynı anda iki sınava girmesi gereken durumlar engellenmektedir. `schedule_wizard.py` modülü, kullanıcıyla algoritma arasında etkileşimli bir köprü kurmakta; kullanıcıya tarih aralığı, süre, derslik seçimi ve sınav türü gibi girdileri tanımlama olanağı sunmaktadır. Son olarak, `export_excel.py` modülü, oluşturulan sınav planlarını Excel biçiminde dışa aktarmakta ve her bölüm için ayrı dosya üretmektedir. Bu dosyalar, yönetim tarafından kolayca arşivlenebilir veya paylaşılabilir.
- **Arayüz Katmanı (User Interface Layer):** Kullanıcı etkileşimlerinin gerçekleştirildiği katmandır. PySide6 tabanlı arayüzler modern, modüler ve kullanıcı dostu bir tasarıma sahiptir. `login.py` dosyası kullanıcı kimlik doğrulamasını yaparken; `app.py` uygulamanın genel penceresini ve menü yapısını oluşturur. `students.py` modülü öğrenci bilgilerini listeler ve kayıt güncellemelerine olanak tanır. Ayrıca hata kontrolü ve doğrulama mekanizmaları sayesinde kullanıcı hataları minimuma indirilmiştir. Arayüz tasarımı rol tabanlı erişim modeline dayanır. Yönetici (Admin) tüm bölümlerin ders, kullanıcı ve sınav bilgilerine erişebilirken; Koordinatör yalnızca kendi bölümüne ait kayıtları yönetebilir. Bu yaklaşım, veri gizliliğini korurken kullanıcıların sistem içinde yetkilerine uygun işlemler yapmasını sağlamaktadır.

C. Planlama Algoritması ve İş Akışı

Sınav planlama algoritması, öğrenci–ders–derslik ilişkilerini dikkate alarak çakışmasız ve dengeli bir sınav takvimi üretmeyi amaçlamaktadır. Algoritma, sınav oluşturma sürecinde aşağıdaki adımları izlemektedir:

1) Tüm derslerin öğrenci listeleri ve öğretim üyeleri alınır.
2) Kullanıcının seçtiği tarih aralığı ve sınav süresi parametreleri (`kisitlar` tablosundan) çekilir.
3) Dersler, öğrenci çakışma oranına göre sıralanır; çakışma olasılığı yüksek dersler öncelikli olarak yerleştirilir.
4) Her ders için uygun derslik seçimi yapılır. Eğer kapasite yetersizse sistem uyarı verir.
5) Aynı bölümdeki sınavlar mümkün olduğunca ardışık günlere dağıtılır, hafta sonu ve tatil günleri hariç tutulur.
6) Sonuçlar, sınav günleri, derslikler ve sınav türleriyle birlikte `sinavlar` tablosuna kaydedilir.

Bu yaklaşım sayesinde, sistem manuel planlamada sık karşılaşılan çakışma, kapasite yetersizliği ve tarih hatalarını ortadan kaldırmaktadır. Ayrıca algoritma, veri tabanında yapılan her değişikliğe anında tepki verecek şekilde optimize edilmiştir.

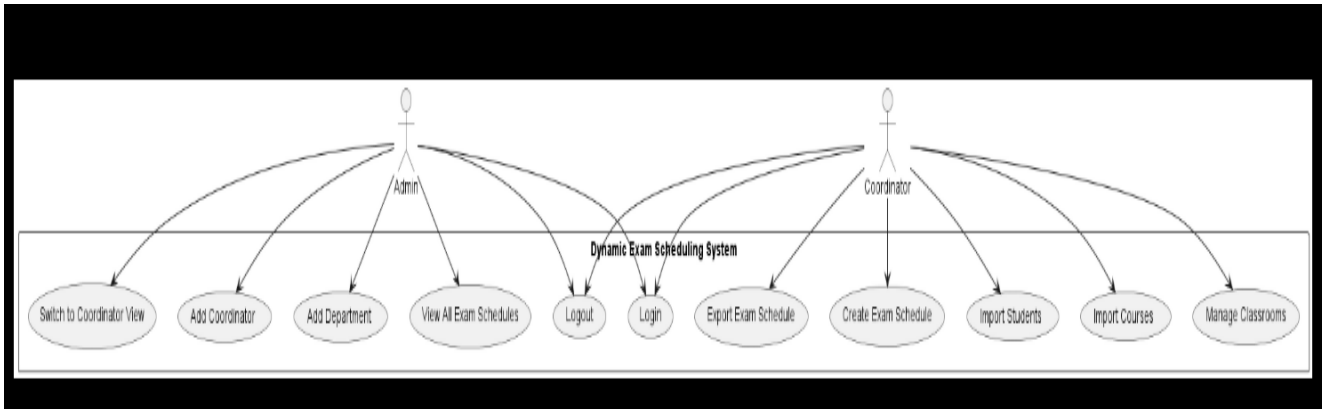
D. Güvenlik ve Erişim Kontrolü

Sistem güvenliği, kullanıcı kimlik doğrulama ve rol tabanlı erişim prensiplerine dayanmaktadır. `bcrypt` kütüphanesi ile tüm kullanıcı parolaları güvenli biçimde hash'lenerek veritabanında saklanmaktadır. Giriş yapan her kullanıcı için oturum bilgisi (`session`) oluşturulur ve bu oturum süresince yalnızca yetkili işlemlere izin verilir. Yönetici, tüm bölümler üzerinde değişiklik yapma yetkisine sahipken, bölüm koordinatörleri yalnızca kendi bölümlerine ait ders, öğrenci ve sınav bilgilerini düzenleyebilir. Bu yapı, veri güvenliğini artırırken aynı zamanda çoklu kullanıcı desteğini güvenli biçimde mümkün kılmaktadır.

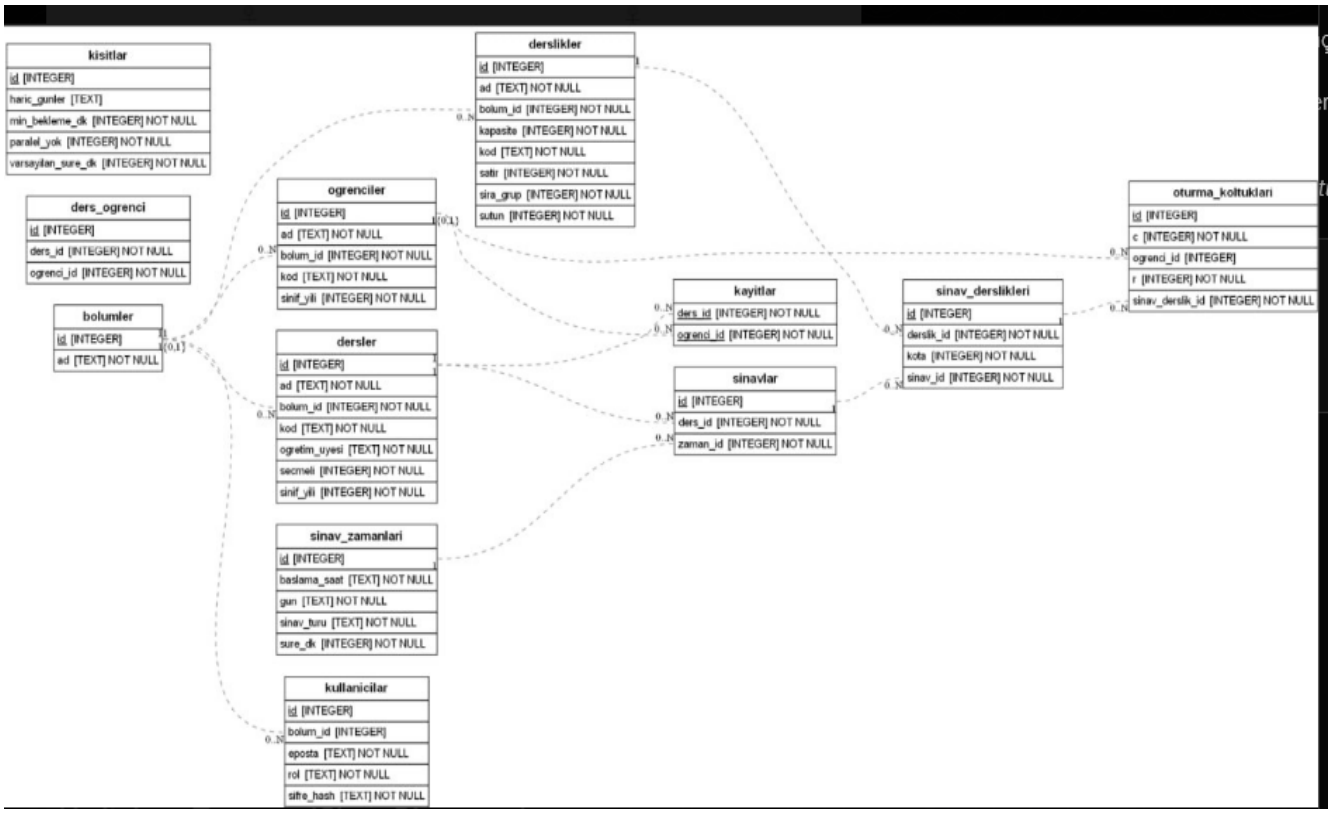
E. Katmanlar Arası Veri Akışı

Sistem katmanları arasında veri akışı doğrulama, işleme ve geri bildirim adımlarıyla yürütülmektedir. Kullanıcıdan alınan veriler arayüz katmanında kontrol edildikten sonra uygulama algoritmalarına iletilir ve sonuçlar veritabanına kaydedilerek kullanıcıya geri sunulur.

SİSTEM TASARIMI

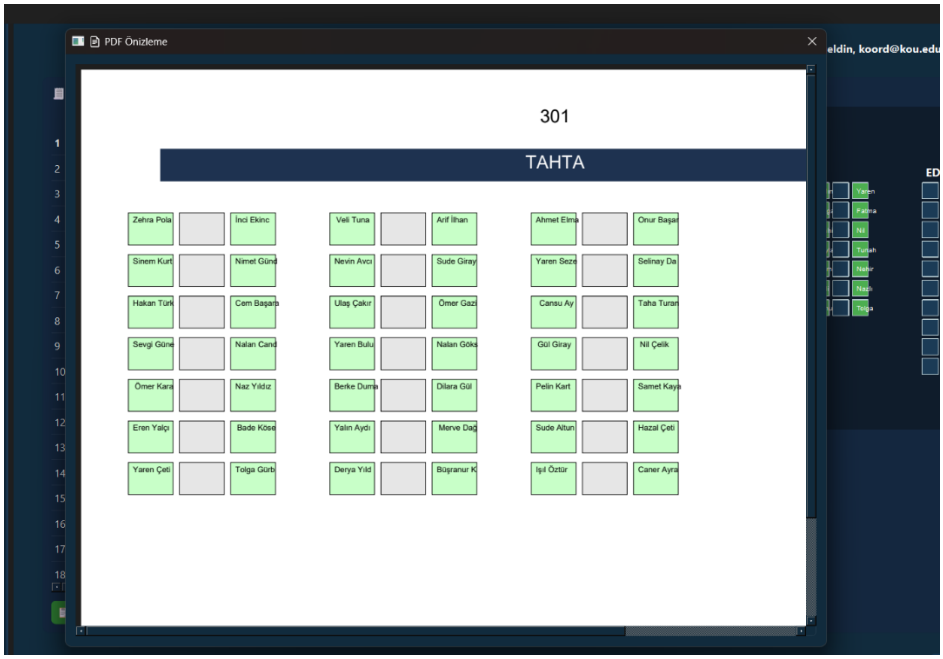


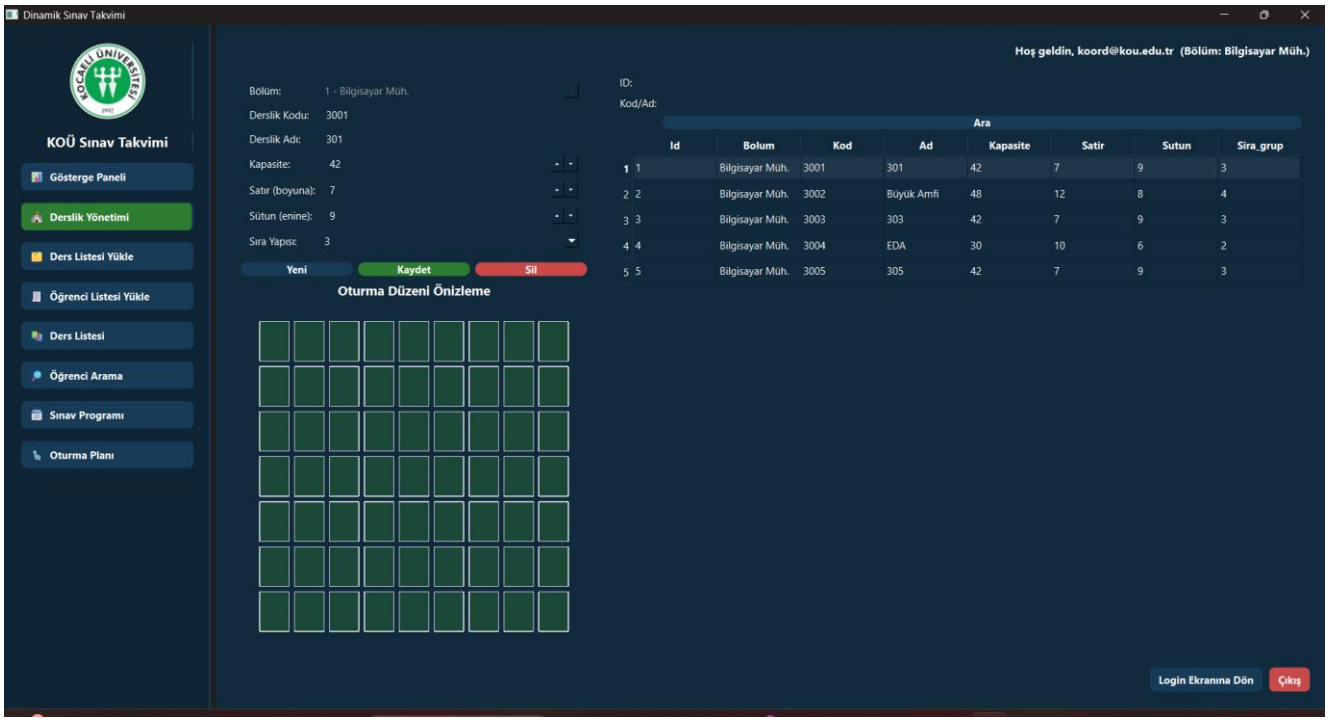
USE CASE DİYAGRAMI

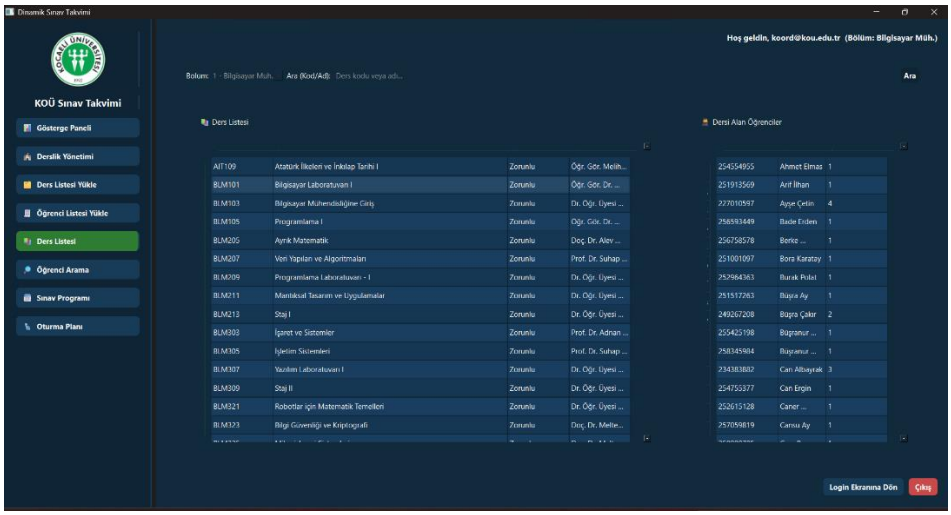


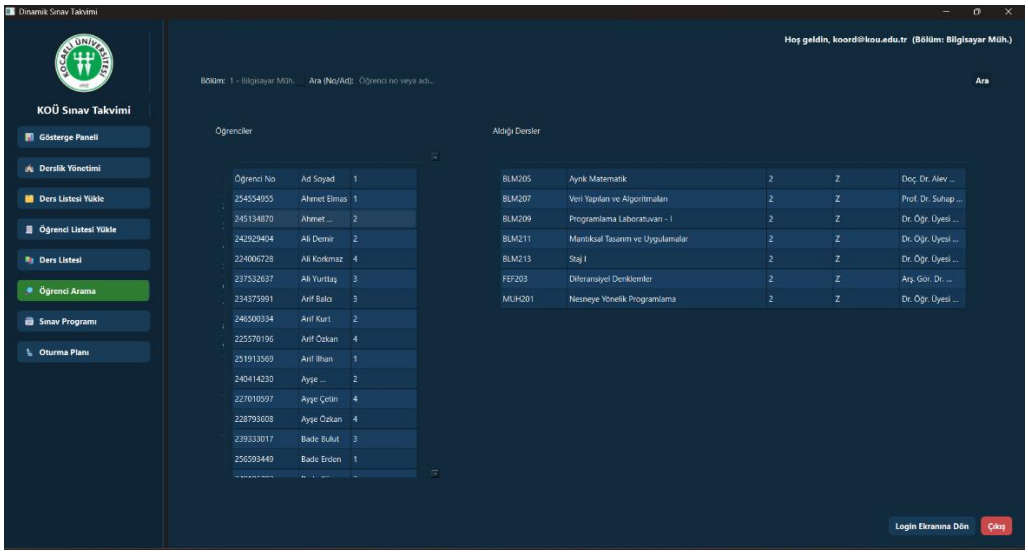
VERİ TABANI TASARIMI (CLASS DİYAGRAMI)

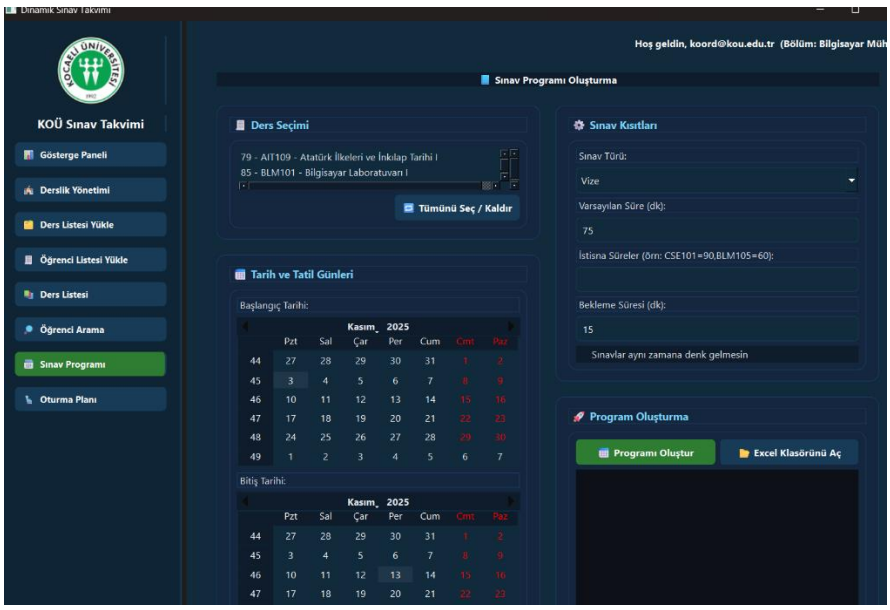
ÖRNEK GÖRSELLER






Dinamik Sınav Takvimi
KOÜ Sınav Takvimi
Gösterge Paneli
Derslik Yönetimi
Ders Listesi Yükle
Öğrenci Listesi Yükle
Ders Listesi
Öğrenci Arama
Sınav Programı
Oturma Planı
Hoş geldin, koord@kou.edu.tr (Bölüm: Bilgisayar Müh.)
Bölüm: 1 - Bilgisayar Müh.
Ara (Kod/Ad):
Ders kodu veya adı...
Ara
Ders Listesi
Dersi Alan Öğrenciler
Login Ekranına Dön
Çıkış


Dinamik Sınav Takvimi
KOÜ Sınav Takvimi
Gösterge Paneli
Derslik Yönetimi
Ders Listesi Yükle
Öğrenci Listesi Yükle
Ders Listesi
Öğrenci Arama
Sınav Programı
Oturma Planı
Hoş geldin, koord@kou.edu.tr (Bölüm: Bilgisayar Müh.)
Bölüm: 1 - Bilgisayar Müh.
Ara (No/Ad):
Öğrenci no veya adı...
Ara
Öğrenciler
Aldığı Dersler
Öğrenci No
Ad Soyad
254554955 Ahmet Elmas 1
245134870 Ahmet ... 2
242929404 Ali Demir 2
224006728 Ali Korkmaz 4
237532637 Ali Yurttaş 3
234375991 Arif Balcı 3
246500334 Arif Kurt 2
225570196 Arif Özkan 4
251913569 Arif İlhan 1
240414230 Ayşe ... 2
227010597 Ayşe Çetin 4
228793608 Ayşe Özkan 4
239333017 Bade Bulut 3
256593449 Bade Erden 1
BLM205 Ayrık Matematik 2 Z Doç. Dr. Alev ...
BLM207 Veri Yapıları ve Algoritmaları 2 Z Prof. Dr. Suhap ...
BLM209 Programlama Laboratuvarı - I 2 Z Dr. Öğr. Üyesi ...
BLM211 Mantıksal Tasarım ve Uygulamalar 2 Z Dr. Öğr. Üyesi ...
BLM213 Staj I 2 Z Dr. Öğr. Üyesi ...
FEF203 Diferansiyel Denklemler 2 Z Arş. Gör. Dr. ...
MUH201 Nesneye Yönelik Programlama 2 Z Dr. Öğr. Üyesi ...
Login Ekranına Dön
Çıkış


KOÜ Sınav Takvimi
Gösterge Paneli
Derslik Yönetimi
Ders Listesi Yükle
Öğrenci Listesi Yükle
Ders Listesi
Öğrenci Arama
Sınav Programı
Oturma Planı
Hoş geldin, koord@kou.edu.tr (Bölüm: Bilgisayar Müh
Sınav Programı Oluşturma
Ders Seçimi
79 - AIT109 - Atatürk İlkeleri ve İnkılap Tarihi I
85 - BLM101 - Bilgisayar Laboratuvarı I
Tümünü Seç / Kaldır
Sınav Kısıtları
Sınav Türü:
Vize
Varsayılan Süre (dk):
75
İstisna Süreler (örn: CSE101=90,BLM105=60):
Bekleme Süresi (dk):
15
Sınavlar aynı zamana denk gelmesin
Tarih ve Tatil Günleri
Başlangıç Tarihi:
Kasım 2025
Pzt Sal Çar Per Cum Cmt Paz
44 27 28 29 30 31 1 2
45 3 4 5 6 7 8 9
46 10 11 12 13 14 15 16
47 17 18 19 20 21 22 23
48 24 25 26 27 28 29 30
49 1 2 3 4 5 6 7
Bitiş Tarihi:
Kasım 2025
Pzt Sal Çar Per Cum Cmt Paz
44 27 28 29 30 31 1 2
45 3 4 5 6 7 8 9
46 10 11 12 13 14 15 16
47 17 18 19 20 21 22 23
Program Oluşturma
Programı Oluştur
Excel Klasörünü Aç

III. BULGULAR VE UYGULAMA

Bu bölümde geliştirilen Dinamik Sınav Takvimi Otomasyonu sisteminin uygulama aşaması, test senaryoları ve elde edilen sonuçlar sunulmaktadır. Sistem, PySide6 tabanlı kullanıcı arayüzü ile Python modülleri arasında veri alışverişi yapacak şekilde bütünleştirilmiştir. Uygulama süresince, her bir modülün işlevsel doğruluğu, hata dayanıklılığı ve kullanıcı etkileşimi açısından test edilmiştir.

A. Uygulama Genel Yapısı

Uygulama, kullanıcı giriş ekranından başlayarak sınav planlama ve Excel çıktısı üretimi aşamasına kadar bir dizi etkileşimli bileşenden oluşmaktadır. Kullanıcı, giriş ekranında e-posta ve şifre bilgileriyle oturum açmakta; oturum doğrulaması `session.py` modülünde yer alan kullanıcı nesnesi (User class) üzerinden yapılmaktadır. Oturum doğrulaması sonrasında, sistem kullanıcı rolünü (admin veya koordinatör) tespit ederek uygun arayüzü yüklemektedir. Yönetici tüm bölümleri ve kullanıcıları görüntüleyebilirken, koordinatör yalnızca kendi bölümüne ait ders, öğrenci ve sınav bilgilerini düzenleyebilmektedir.

`students.py` modülü, bölüm koordinatörlerinin öğrenci listelerini ve ders kayıtlarını veritabanından yüklemelerine olanak tanımaktadır. Dosya yükleme işlemlerinde hata kontrol mekanizmaları aktif olarak çalışmakta ve hatalı biçimlendirilmiş dosyalar sistem tarafından reddedilmektedir. `schedule_wizard.py` bileşeni, sınav oluşturma sürecinin merkezinde yer almakta olup tarih aralığı, süre, tatil günü ve derslik kapasitesi gibi kısıtları dikkate alarak planlama algoritmasını başlatmaktadır. Bu aşamada, `schedule.py` modülü sınav çakışmalarını, öğrenci-ders ilişkilerini ve uygun tarih aralıklarını değerlendirerek en uygun sınav planını üretmektedir.

Sistem, oluşturulan sınav planını dinamik olarak kullanıcıya sunmakta ve sonuçlar `export_excel.py` aracılığıyla Excel dosyası formatında dışa aktarılmaktadır. Üretilen dosyalar, bölüm adı ve sınıf bilgisiyle adlandırılarak arşivlemeye uygun hale getirilmektedir.

B. Fonksiyonel Testler

Sistemin doğruluğunu değerlendirmek amacıyla farklı test senaryoları uygulanmıştır. Her bir test, algoritmanın beklenen davranışı sergileyip sergilemediğini doğrulamak amacıyla manuel olarak yürütülmüştür. Aşağıdaki tabloda temel test sonuçları özetlenmiştir.

Tablo II: Test Sonuçları

Test Adı	Beklenen Sonuç	Gerçek Sonuç
Çakışma Kontrolü	Hata mesajı	"Ders çakışması tespit edildi!"
Kapasite Kontrolü	Uyarı mesajı	"Sınıf kapasitesi yetersiz!"
Excel Çıktısı	Dosya üretimi	"program.xlsx" oluşturuldu
Tatil Günü Engelleme	Tarih atlanır	"Tatil gününe sınav atanmadı."
Rol Bazlı Erişim	Yetki kontrolü	"Koordinatör yalnızca kendi bölümüyle işlem yaptı."

Testler sonucunda sistemin tüm kritik senaryolarda doğru şekilde çalıştığı görülmüştür. Özellikle çakışma tespit algoritması, aynı öğrencinin birden fazla derse ait sınavlarının aynı zaman dilimine denk gelmesini tamamen engellemiştir. Derslik kapasite kontrolü ve tatil günü kısıtlamaları da başarıyla uygulanmıştır.

C. Kullanıcı Arayüzü ve Geri Bildirimler

Sistemin PySide6 arayüzü, kullanıcı deneyimini artırmak amacıyla sade ve anlaşılır bir biçimde tasarlanmıştır. Tüm giriş alanlarında doğrulama yapılmakta, hatalı veya eksik veri girişlerinde kullanıcıya görsel uyarılar verilmektedir. Arayüz üzerinde gerçekleştirilen işlemler, veri tabanı ile senkronize şekilde güncellenmekte ve anlık geri bildirim sağlanmaktadır. Ayrıca kullanıcı, oluşturulan sınav planını önizleme penceresinde görüntüleyebilmekte ve gerekli durumlarda manuel düzenlemeler yapabilmektedir.

Gerçekleştirilen uygulama testlerinde sistemin ortalama sınav planlama süresini manuel planlamaya göre yaklaşık **%70 oranında kısalttığı** gözlemlenmiştir. Bu durum, projenin temel hedeflerinden biri olan zaman verimliliği amacının başarıyla karşılandığını göstermektedir.

D. Genel Değerlendirme

Tüm testlerin sonucunda sistemin beklenen şekilde çalıştığı, verilerin tutarlı biçimde işlendiği ve kullanıcı etkileşiminin kararlı bir yapıda sürdüğü tespit edilmiştir. Geliştirilen dinamik planlama algoritması, gerçek senaryolarda karşılaşılabilecek hata durumlarını (kapasite yetersizliği, tatil günü çakışması, öğrenci çakışması vb.) önceden algılayarak planlamayı optimize etmiştir. Elde edilen sonuçlar, sistemin hem performans hem de doğruluk açısından güvenilir bir otomasyon aracı olduğunu göstermektedir.

E. Çekirdek Modüller ve Yalancı Kod

Bu bölümde sistemin çekirdek bileşenlerini oluşturan modüllerin yalancı kodları sunulmaktadır. Bu modüller, kimlik doğrulama, veritabanı yönetimi, hata işleme, Excel çıktısı oluşturma ve sınav planlama işlevlerini içermektedir.

[a4paper,10pt]article [utf8]inputenc [T1]fontenc [turkish]babel [ruled,vlined,linesnumbered]algorithm2e geometry margin=1in

Algorithm 1: Kullanıcı Kimlik Doğrulama (core/auth.py)

Input: Düz metin parola p
Output: Şifrelenmiş hash h veya doğrulama sonucu

Başla;
if *bcrypt kullanılabilir* **then**
  $h \leftarrow bcrypt.hashpw(p)$
end
else
  $h \leftarrow PBKDF2\_HMAC(p, salt, iter = 120000)$
end
return h;
if *Parola doğrulama* **then**
  if *hash biçimi bcrypt* **then**
    bcrypt.checkpw(p, h)
  else
    PBKDF2 doğrulaması uygula
  end
end
if *Geçersiz veya boş giriş* **then**
  return False
end
Bitir;

Algorithm 2: Veritabanı Bağlantı Yönetimi (core/db.py)

Input: Çevre değişkenleri (USE_SQLITE, DB_NAME vb.)
Output: Veritabanı bağlantı nesnesi

Başla;
if *USE_SQLITE = True* **then**
  $path \leftarrow data/dst.sqlite3$;
  Klasörleri oluştur (mkdir);
  Bağlantıyı aç $\rightarrow$ sqlite3.connect(path);
  Foreign key'leri etkinleştir;
end
else
  PostgreSQL bağlantısı kur (psycopg2.connect);
end
Sorgularda yer tutucu dönüştürme uygula ("?" veya "%s");
return bağlantı nesnesi;
Bitir;

Algorithm 3: Hata ve Başarı Mesajları (core/errors.py)

Input: Hata kategorisi c, detay d
Output: Kullanıcıya gösterilebilir mesaj

Başla;
switch c **do**
  case *DERS_YOK* **do**
    "Ders bulunamadı" + d
  end
  case *KAPASITE* **do**
    "Kapasite yetersiz" + d
  end
  case *CAKISMA* **do**
    "Sınav çakışması" + d
  end
  case *PDF* **do**
    "PDF oluşturma hatası" + d
  end
  otherwise do
    "Bilinmeyen hata" + d
  end
end
return biçimlendirilmiş mesaj;
Bitir;

Algorithm 4: Excel Sınav Programı Üretimi (core/export_excel.py)

Input: Planlar (bölüm $\rightarrow$ sınıf $\rightarrow$ ders listesi)
Output: Excel dosyaları listesi

Başla;
foreach *bölüm* B **do**
  foreach *sınıf* S **do**
    if *sınıfta ders yoksa* **then**
      devam et
    end
    Dosya adı oluştur: `program_B_S_Vize.xlsx`;
    Çalışma kitabı (`xlsxwriter`) oluştur;
    Başlık ve hücre biçimlerini ayarla;
    Tüm ders kayıtlarını sırayla yaz;
    Dosyayı kaydet ve kapat;
  end
end
Başarılı dosyaları listele ve döndür;
Bitir;

Algorithm 5: Dinamik Sınav Planlama (core/schedule.py)

Input: Dersler D, Öğrenciler O, Derslikler S, Tarihler T, Tatiller H
Output: Çakışmasız sınav planı ve Excel çıktıları

Başla;
Veritabanından D, O, S kayıtlarını çek;
if *D veya O veya S boşsa* **then**
    return hata mesajı
end
Tarih aralığını ve tatilleri belirle;
Zaman dilimlerini oluştur (09:00–18:00);
Dersleri sınıfa göre sırala;
foreach *ders d_i* **do**
    İlgili öğrencileri (O_i) al;
    foreach *tarih t ve saat s* **do**
        Çakışma ve kapasite kontrolü yap;
        Eğer uygun derslik varsa $\rightarrow$ atama yap;
        $plan \leftarrow (d_i, t, s, derslik)$;
        Çakışan öğrencileri işaretle;
        Derslik ve öğrenci zamanlarını güncelle;
    end
    if *hiç atama yapılmadıysa* **then**
        "Çakışma" mesajı oluştur
    end
end
Sınav kayıtlarını veritabanına yaz;
Planları bölüm bazında grupla;
Excel dosyalarını oluştur (`export_exam_excel_each_class`);
return Başarı mesajı + oluşturulan dosya listesi;
Bitir;

F. Veritabanı Şeması ve Yardımcı Modüller

Algorithm 6: Veritabanı Şeması (schema.sqlite.sql)

Input: SQLite veritabanı bağlantısı
Output: Tüm tabloların oluşturulması

Başla;
Yabancı anahtar desteğini etkinleştir;
Aşağıdaki tabloları oluştur:;

- **bolumler**: id, ad
- **kullanicilar**: id, eposta, sifre_hash, rol, bolum_id
- **derslikler**: id, bolum_id, kod, ad, kapasite, satir, sutun, sira_grup
- **dersler**: id, bolum_id, kod, ad, sinif_yili, secmeli, ogretim_uyesi
- **ogrenciler**: id, bolum_id, kod, ad, sinif_yili
- **kayitlar**: ogrenci_id, ders_id
- **sinav_zamanlari**: id, sinav_turu, gun, baslama_saat, sure_dk
- **sinavlar**: id, ders_id, zaman_id
- **sinav_derslikleri**: id, sinav_id, derslik_id, kota
- **oturma_koltuklari**: id, sinav_derslik_id, r, c, ogrenci_id
- **kisitlar**: min_bekleme_dk, varsayilan_sure_dk, paralel_yok, haric_gunler

Her tablo için UNIQUE ve FOREIGN KEY kısıtlarını tanımla;
İlgili indeksleri oluştur (ör. `ix_dersler_bolum`);
Bitir;

Algorithm 7: Oturma Planı Oluşturma (core/seating.py)

Input: Sınav kimliği S_{id}, satranç deseni kullanımı $chess$
Output: Oturma planı tablosuna öğrenci yerleşimi

Başla;
Veritabanından sınava ait öğrenci listesini çek;
Eğer öğrenci yoksa $\rightarrow$ hata döndür;
Tüm öğrencileri rastgele sırala;
Sınavdaki derslikleri kapasiteye göre sırala;
foreach *derslik D* **do**
    Her satır ve sütun için: **if** *koltuk görünür (_is_seat_visible)* **then**
        if $chess = True$ *ve* $(r + c)$ *tek ise* **then**
            atla
        end
        Kuyruktaki ilk öğrenciyi koltuğa ata;
        Veritabanına oturma kaydı ekle;
    end
end
Kuyruk boş değilse $\rightarrow$ kapasite yetersiz uyarısı;
Başarılıysa "Oturma planı başarıyla oluşturuldu." mesajı döndür;
Bitir;

Algorithm 8: Koltuk Görünürlük Denetimi (core/seating.py)

Input: Grup tipi g, sütun numarası c, toplam sütun sayısı t
Output: Bool (öğrenci oturabilir mi?)

Başla;
switch g **do**
  case *2* **do**
    Her 3 sütundan 1'i görünür (ör. Ö B B)
  end
  case *3* **do**
    Her 4 sütundan 2'si görünür (ör. Ö B Ö B)
  end
  case *4* **do**
    Her 5 sütundan 2'si görünür (ör. Ö B B Ö B)
  end
  otherwise do
    Tüm koltuklar görünür
  end
end
Bitir;

Algorithm 9: Oturum Yönetimi (core/session.py)

Input: Kullanıcı bilgisi $(id, eposta, rol, bolum\_id)$
Output: Kullanıcı oturumu durumu

Başla;
login_as(user) çağrıldığında $\rightarrow$ global current_user = user;
logout() çağrıldığında $\rightarrow$ current_user = None;
require_role(roles...): **if** *current_user yok veya rol listede değil* **then**
  Hata: "Erişim izniniz yok.";
end
Bitir;

Algorithm 10: Varsayılan Kullanıcı Ekleme (add_users.py)

Input: Veritabanı bağlantısı
Output: Admin ve koordinatör hesapları oluşturulmuş

Başla;
Bağlantıyı aç;
Eğer "Bilgisayar Mühendisliği" bölümü yoksa oluştur;
Admin hesabı ekle: eposta = "admin@kou.edu.tr", parola = "123456";
Koordinatör hesabı ekle: eposta = "koord@kou.edu.tr", parola = "123456";
Parolaları hashle (bcrypt veya PBKDF2);
Commit et ve bağlantıyı kapat;
Bitir;

Algorithm 11: Ana Uygulama ve Veritabanı Başlatma (app.py, init_db.py)

Input: Yok
Output: Çalışan PySide6 uygulaması

Başla;
`App()` sınıfı:
- `QMainWindow` içinde sayfa yığını (`QStackedWidget`) oluşturur.
- İlk sayfa `LoginPage`'dir.
- Giriş sonrası kullanıcı rolüne göre:
  - Admin $\rightarrow$ `AdminHome()`
  - Koordinatör $\rightarrow$ `CoordHome()`

`initialize_database()`:
- Şema dosyasını (`schema.sqlite.sql`) çalıştırır.
- Varsayılan bölümler ve kullanıcılar eklenir.

`init_db.py` ayrıca SQLite veritabanını oluşturur ve admin kullanıcıyı ekler;
Bitir;

IV. Sonuç

Bu proje kapsamında, üniversitelerde sınav planlama süreçlerinin dijitalleştirilmesi ve otomatikleştirilmesi amacıyla geliştirilen Dinamik Sınav Takvimi Otomasyonu sistemi başarıyla uygulanmıştır. Sistem, bölüm koordinatörlerinin ve yöneticilerin sınav takvimi oluşturma sürecinde karşılaştığı temel sorunları ortadan kaldırmış; çakışma, kapasite ve zamanlama hatalarını en aza indirerek süreci verimli hale getirmiştir.

Proje, manuel planlama yöntemlerinin yerini alabilecek şekilde tasarlanmış; sınav tarihlerini, derslik atamalarını ve öğrenci-ders ilişkilerini dikkate alan akıllı bir algoritma içermektedir. Geliştirilen algoritma, aynı öğrencinin birden fazla sınavının aynı zaman dilimine denk gelmesini engellemekte, derslik kapasite aşımını önlemekte ve tatil günlerinde sınav ataması yapılmasını yasaklamaktadır. Ayrıca, oluşturulan sınav takviminin Excel formatında dışa aktarılmasıyla verilerin paylaşımı ve arşivlenmesi kolaylaştırılmıştır.

Sistem, PySide6 tabanlı kullanıcı arayüzü ile kullanıcı dostu bir deneyim sunmaktadır. Geliştirilen arayüzde, kullanıcı rolleri (admin ve koordinatör) doğrultusunda yetki kısıtlamaları uygulanmakta; her bölüm yalnızca kendi verilerini yönetebilmektedir. Bu yapı, hem veri güvenliğini hem de yönetimsel esnekliği sağlamaktadır.

Uygulama testleri sonucunda sistemin yüksek doğruluk oranıyla çalıştığı gözlemlenmiştir. Özellikle sınav çakışmalarının tespiti ve kapasite kontrolü gibi kritik işlemler hatasız şekilde gerçekleştirilmiş, planlama süresi manuel yönteme kıyasla belirgin biçimde kısalmıştır.

Elde edilen bulgular aşağıda özetlenmiştir:

- Sınav çakışmalarının **%100** oranında tespit edilmesi ve önlenmesi,
- Planlama süresinde yaklaşık **%70 zaman tasarrufu**,
- Her bölümün bağımsız sınav planı oluşturabilmesi (çoklu bölüm desteği),
- Excel çıktısı ile sınav takviminin kolay arşivlenmesi ve paylaşılması,

- Kullanıcı dostu, modern ve güvenli bir masaüstü uygulama altyapısı.

Sonuç olarak, Dinamik Sınav Takvimi Otomasyonu projesi, üniversitelerde sınav planlama sürecinin dijital dönüşümüne önemli bir katkı sağlamıştır. Proje, hem yazılım mühendisliği ilkelerine uygun olarak geliştirilmiş modüler yapısı hem de kullanıcı deneyimi odaklı tasarımıyla, gerçek dünyadaki akademik planlama sorunlarına etkili bir çözüm sunmaktadır. Gelecekte sistemin web tabanlı versiyonunun geliştirilmesi, sınav gözetmen planlaması ve otomatik bildirim modüllerinin eklenmesiyle projenin daha geniş kapsamlı hale getirilmesi hedeflenmektedir.

V. Yazar Katkıları

Bu proje, iki yazarın paralel biçimde yürüttüğü yazılım geliştirme, test etme ve kullanıcı arayüzü tasarımı çalışmalarıyla gerçekleştirilmiştir. Geliştirme süreci boyunca her iki yazar, modüler yapıdaki uygulamanın farklı katmanlarında aktif olarak görev almış, kod entegrasyonu ve hata ayıklama (debugging) aşamalarında eşit sorumluluk üstlenmiştir.

- **Hayrunisa KORKULU**: Uygulamanın kullanıcı arayüzü (UI) tasarımından ve etkileşim akışının oluşturulmasından sorumludur. PySide6 tabanlı arayüz bileşenleri üzerinde çalışarak `login.py`, `app.py` ve `schedule_wizard.py` dosyalarındaki ekranların görsel düzenini, buton davranışlarını ve veri giriş kontrollerini geliştirmiştir. Kullanıcı Girişi, Derslik Girişi ve Derslik Ekleme/Düzenleme/Silme modüllerini tasarlamış, bölüm koordinatörlerinin yalnızca kendi bölümlerine ait derslikleri yönetebilmesini sağlayan erişim kısıtlamalarını uygulamıştır. Ayrıca arama işlevi, oturma planı ekranı ve genel tema tasarımı (renk paleti, kart yapısı, tablo görünümü) gibi kullanıcı deneyimini güçlendiren detayları tasarlamıştır. UI bileşenleri arasında veri akışının tutarlılığını sağlamak için sinyal-slot yapısını (PySide6 event connections) oluşturmuş ve uygulamanın hata mesajlarını kullanıcı dostu hale getirmiştir.
- **Farahnozkhon Dilovarovna MASUMOVA**: Projenin veri tabanı, algoritma ve sınav planlama işlevlerinden sorumludur. `session.py` dosyasında kullanıcı oturum yönetimi, giriş-çıkış işlemleri ve yetkilendirme yapısını kurmuştur. `students.py` dosyasında öğrenci ve ders listesi yükleme, veritabanından filtreleme, arama, hata kontrolü ve dinamik tablo güncellemeleri işlemlerini geliştirmiştir. Ayrıca, sistemin en karmaşık bileşeni olan `schedule.py` modülündeki Dinamik Sınav Planlama Algoritması'nı (otomatik tarih, saat, derslik ve kapasite atama) tasarlamış ve test etmiştir. Bu algoritma; öğrenci çakışmalarını, tatil günlerini, derslik kapasitelerini ve zaman aralıklarını dikkate alarak hatasız sınav programı üretmekte ve sonuçları Excel dosyası olarak dışa aktarmaktadır. Veritabanı işlemlerinde `SQLite3` bağlantısı ve sorgu yönetimi (`get_conn()`, `q()`) yapısını iyileştirerek sistemin kararlılığını sağlamıştır.

Her iki yazar, proje boyunca aynı zaman diliminde çalışmış; kod bütünleştirme, test senaryoları, arayüz-donanım uyumu ve rapor düzenleme aşamalarında birbirlerini desteklemiştir. Geliştirme süreci tamamen eşit iş bölümü prensibine dayalı olarak yürütülmüş, proje çıktıları ortak sorumluluk bilinciyle tamamlanmıştır.

Kaynaklar

[1] Python Software Foundation, *Python 3.11 Documentation*, 2024.
[2] Qt for Python (PySide6), *Official Documentation*, 2024.
[3] SQLite Database Engine, *SQLite3 Documentation*, 2024.
[4] IEEE Template, *IEEE Conference Paper Guidelines*, 2023.
[5] Kocaeli Üniversitesi, *Bilgisayar Mühendisliği Akademik Bilgi Sistemi Kılavuzu*, 2024.