Yazılım Geliştirme-1 Proje-2
Dosyalama Ve Yedekleme Sistemi

Gizem Zor
Yazılım Mühendisliği
Kocaeli Üniversitesi
Kocaeli,Türkiye
gizemzor@gmail.com

***Özetçe*—Bu belge, Python ve tkinter GUI geliştirme modülü kullanılarak , kullanıcılara özel dosyalama ve yedekleme sistemi geliştirmiştir**

Anahtar Kelimeler — dosyalama,yedekleme,log,paylaşım.

I. GİRİŞ

Dosyalama ve yedekleme sistemi uygulaması ,Python proglamlama dili ve Python GUI modülü olan tkinter kullanılarak yapılmıştır.

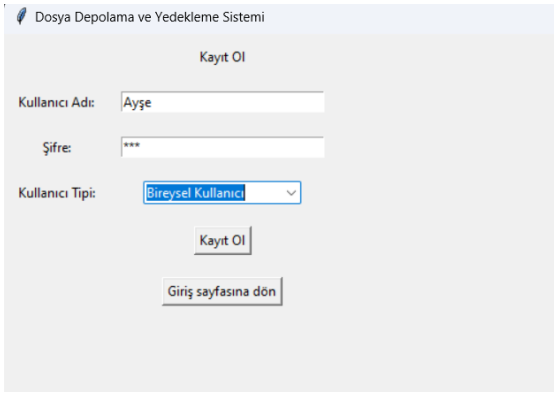
A. Arayüz

- Her profil sahibinin eşsiz bir kullanıcı adı almasına izin verilmelidir. Daha önceden alınmış bir kullanıcı adı ile profil oluşturulmak istenirse uyarı gösterilmelidir.

```
def kayıtOl(self):  (parameter) self: Self@File
    username = self.register_username.get()
    password = self.register_password.get()
    userType=self.user_type.get()

    if not username or not password:
        messagebox.showwarning("Uyarı", "Kullanıcı adı ve şifre boş bırakılamaz.")
        return

    try:
        with open("kayıt.txt", "r", encoding="utf-8") as file:
            for line in file:
                if f"Kullanıcı  Adı:{username}" in line:
                    messagebox.showwarning("Uyarı", "Bu kullanıcı adı kullanılıyor. Başka bir kullanıcı adı seçiniz.")
                    self.register_username.delete(0, tk.END)
                    return
    except FileNotFoundError:
        pass

    hashed_password = self.hash_password(password)
    with open("kayıt.txt", "a", encoding="utf-8") as file:
        file.write(f"Kullanıcı  Adı:{username}      Şifre:{hashed_password.decode('utf-8')}     Kullanıcı Tipi:{userType}\n")

    messagebox.showinfo("Başarılı", "Kayıt başarıyla tamamlandı.")
    self.show_login_frame()
```

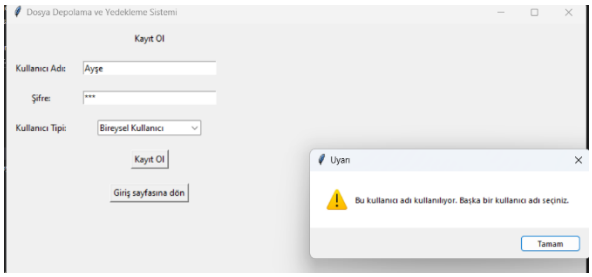
Şekil 1

Şekil 1 ‘de kayıtol metodum bulunmaktadır . Kullanıcı kaydı burada yapılır. Arayüzdeki girişlere göre kullanıcıya uyarı yazıları döndürülür.



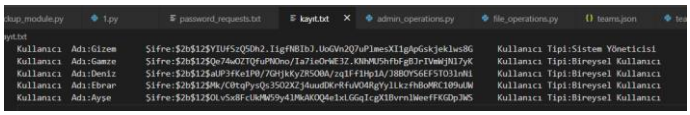
Şekil 2

Şekil 2’de kullanıcı kayıt arayüzü bulunmaktadır.Uygulamaya Ayşe kullanıcı adlı birinin kaydı yapılmaktadır.



Şekil 3

Şekil 3’de yine Ayşe kullanıcı adlı birinin kaydı yapılmak istenmektedir. Fakat o kullanıcı kaydı ,kullanıcıların kayıt edildiği ‘kayıt.txt ’ dosyasında bulunduğu için kullanıcı kaydı yapılamamıştır ve uyarı döndürülmüştür.



Şekil 4

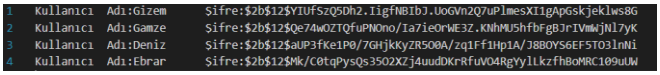
Şekil 4'de kaydı yapılan kullanıcılarımın bulunduğu kayıt.txt dosyasının içi görünmektedir. Kayıt dosyasında kullanıcı adı, hashlanmiş şifre yapısı ile şifre ve kullanıcı tipi başlıkları bulunmaktadır.

- Her profil sahibinin profil parolası, bir şifreleme yöntemi ile sistemde saklanmalıdır. Her giriş denemesinde; giriş esnasında yazılan parola aynı şifreleme yöntemi ile dönüştürüldükten sonra sistemde saklanan haliyle aynı olup olmadığı kontrol edilmelidir. Aynı ise profile erişim sağlanmalı, farklıysa parolanın yanlış olduğu belirtilerek erişim sağlanmamalıdır.

```
10  import bcrypt
```

Şekil 5

Şekil 5' de Python' da bulunan parola karma algoritması olan bcrypt algoritması uygulamaya dahil edilmiştir. Böylece hem kullanıcıların girdiği şifrelerin karma bir şekilde txt ye yazılmasına hem de okunmasına olanak sağlar.



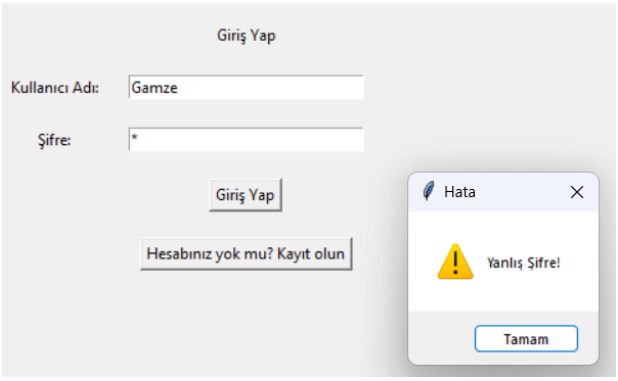
Şekil 6

Şekil 6'da kullanıcılara ait hashlenmiş(karıştırılmış) şifrelerin görüntüsü vardır.

```
def hash_password(self, password):
    salt = bcrypt.gensalt()
    hashed = bcrypt.hashpw(password.encode('utf-8'), salt)
    return hashed

def check_password(self, stored_hash, password):
    return bcrypt.checkpw(password.encode('utf-8'), stored_hash)
```

Şekil 7

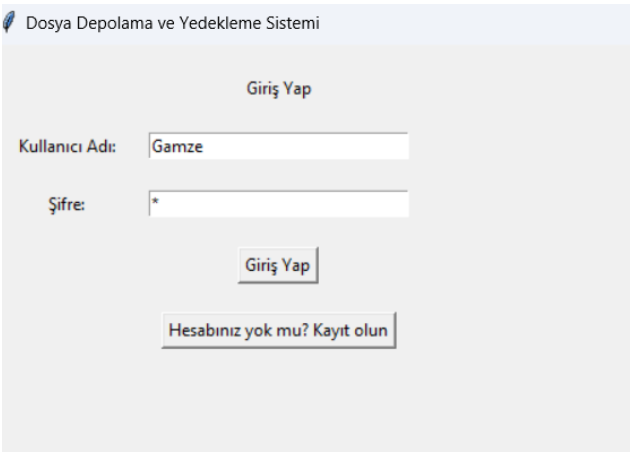
Şekil 7' de şifreleri hashleme kodu mevcur hash_password metodu ile şifre hashleniyor.check_password metodu ile de kullanıcı girişi yapılırken şifre kontrolü yapılıyor ve kullanıcının sisteme girişi sağlanıyor.



Şekil 8

Şekil 8 'de kullanıcın girdiği şifre yanlış ise arayüzde kullanıcıya bir mesaj döndürülüyor.

- Arayüzün tasarımı, rengi açısından herhangi bir kısıtlama yoktur.



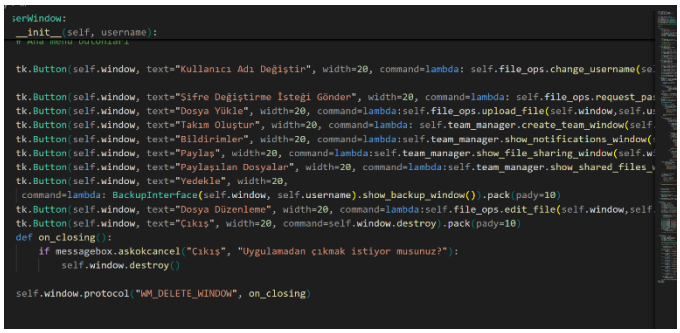
Şekil 9

Şekil 9'da arayüz tasarımı verilmiştir.Arayüzde bir kısıtlama olmadığı için tkinter modülünün default olarak verdiği tasarım kullanılmıştır.

B. Profiller

1. Bireysel Kullanıcılar

- Yapabileceği İşlemler: Kullanıcı adı belirleme/değiştirme, parola belirleme, parola değiştirme isteği gönderme, dosya yükleme, takım üyesi belirleme, paylaşma ve dosya düzenleme.



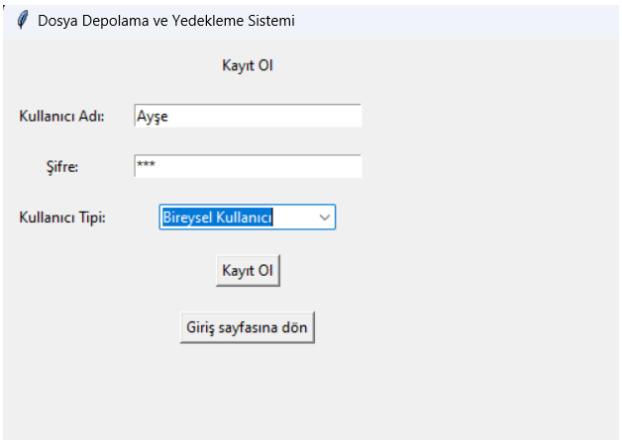
Şekil 10

Şekil 10’ da bireysel kullanıcı profilinin yapabileceği işlemler button özelliği kullanılarak tanımlanmıştır.

Şekil 11

Şekil 11 ‘de bireysel kullanıcının yapabileceği işlemler tanımlanmıştır .Kullanıcı sisteme giriş yaptıktan sonra bu butonları kullanarak gerekli işlemi yapılabilir.

- Bireysel kullanıcılar, kullanıcı adı ve parola belirleyerek profil oluşturabilmelidir.



Şekil 12

Şekil 12 ‘ de kullanıcı bireysel kullanıcı olarak kaydını yaparak kendini oluşturuyor.

- İhtiyaç duyduklarında kullanıcı adını değiştirebilmeli ve sistem yöneticisine parola değiştirme isteği gönderebilmelidir. Sistem yöneticisi onay verdikten sonra kullanıcı parolasını değiştirebilmelidir. Onay verilmediği durumda arayüz üzerinde uyarı gösterilmelidir.

a-Kullanıcı Adı Değiştirme

Kullanıcı adı değiştirme için kullandığım change_username metodu file operations.py dosyamda bulunmaktadır.

```
def change_username(self, current_window, current_username, open_
# Yeni pencere oluştur
    start_time=datetime.now()
    self.change_window = Toplevel(current_window)
    self.change_window.title("Kullanıcı Adı Değiştir")
    self.change_window.geometry('400x200')

    Label(self.change_window, text="Yeni Kullanıcı Adı:", font=('
    username_entry = Entry(self.change_window, width=30)
    username_entry.pack(pady=10)

    def update_files_and_contents(old_username, new_username):
        try:
            # 1. Önce dosya içeriklerini güncelle
            files_to_update = {
                'txt': [
                    'password_requests.txt',

                    'kayıt.txt'
                ],
                'json': [
                    'notifications.json',
                    'teams.json'
                ]
            }
```

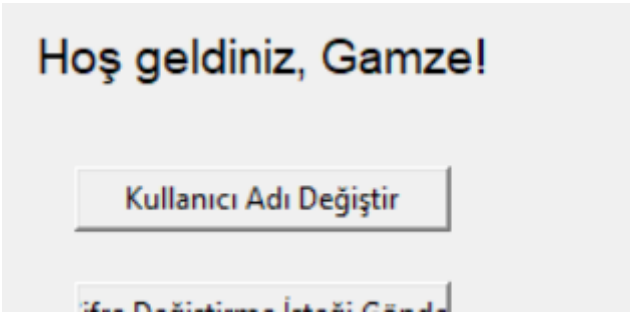
Şekil 13

Şekil 13'de change_username metodum bulunmaktadır.Kullanıcı adını değiştir butonuna basdığında bu metot çalışacaktır . update_files_and_contents metodunda ise şu işlem yapılmaktadır. Kullanıcı adı değiştirildiğinde eski adının kullanıldığı bütün txt ve json dosyaları yeni adı ile güncellenecektir.

```
def save_new_username():
    end_time=datetime.now()
    new_username = username_entry.get().strip()

    if not new_username:
        messagebox.showwarning("Uyarı", "Kullanıcı adı boş olamaz!")
        return

    try:
        with open("kayıt.txt", "r", encoding="utf-8") as file:
            content = file.read()
            if f"Kullanıcı  Adı:{new_username}" in content:
                messagebox.showwarning("Uyarı", "Bu kullanıcı adı kullanılıyor. Başka bir kullanıc
                username_entry.delete(0, tk.END)
                return

        if update_files_and_contents(current_username, new_username):
```

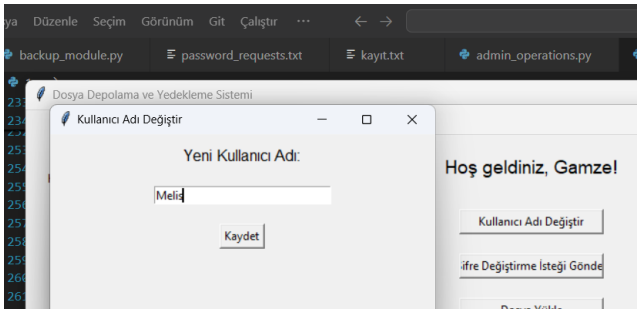
Şekil 14

Şekil 14'de değiştirilen kullanıcı adının txt dosyasın kaydedildiği save_username metodu var.



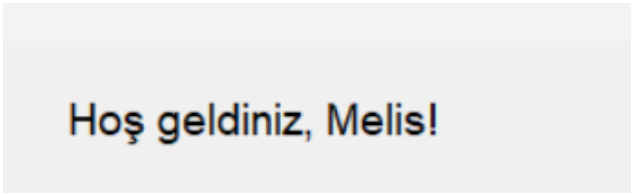
Şekil 15

Şekil 15'de kullanıcı adı değiştirmeden önce arayüzde kullanıcının adı mevcuttur.



Şekil 16

Şekil 16'da kullanıcı yeni adını girmektedir.



Şekil 17

Şekil'17'de kullanıcın yeni adının ekrana yazdırılmış hali mevcuttur.

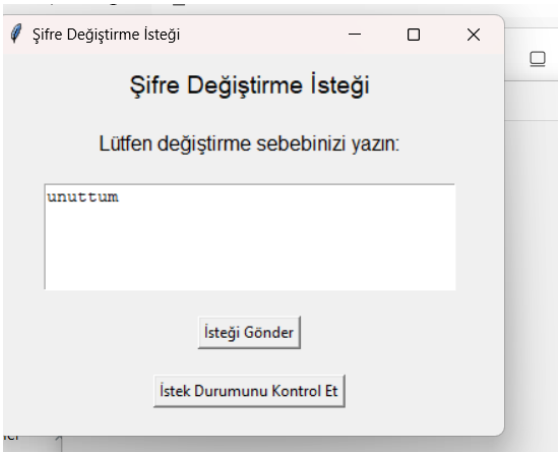
b-Şifre Değiştirme İsteği

Şifre değiştime isteği göndermek için yazılan kodum file_operations.py dosyasında bulunuyor.

```
class FileOperations:

    def request_password_change(self, current_window, username):
        start_time=datetime.now()
        self.request_window = Toplevel(current_window)
        self.request_window.title("Şifre Değiştirme İsteği")
        self.request_window.geometry('400x300')

        Label(self.request_window, text="Şifre Değiştirme İsteği", font=('Arial', 14)).pack(pady=10)
        Label(self.request_window, text="Lütfen değiştirme sebebinizi yazın:", font=('Arial', 12)).pack(pady=10)

        reason_text = Text(self.request_window, height=5, width=40)
        reason_text.pack(pady=10)

        def send_request():
            end_time=datetime.now()
            reason = reason_text.get("1.0", "end-1c").strip()
            if not reason:
                messagebox.showwarning("Uyarı", "Lütfen bir sebep belirtin!")
                return
```

Şekil 18

Şekil 18'de şifre değiştirme isteği için yazılmış metod mevcuttur.



Şekil 19

Şekil 19 şifre değiştirme isteği arayüzü vardır.

```
password_requests.txt
Kullanıcı:Melis
Sebep:Unuttum
Durum:Beklemede
Durum: Onaylandı
---
Sebep:Unuttum
Durum:Beklemede
 Tarih:2024-12-30 20:06:14
---
Kullanıcı:Deniz
Sebep:Unuttum
Durum:Beklemede
 Tarih:2024-12-30 20:06:43
```

Şekil 20

Şekil 20'de password.request.txt de gönderilen şifre değiştirme istekleri bulunmaktadır.henüz onaylanmaynaların beklmede ,onaylanan istekler ise onaylandı olarak görülüyor.

- Bireysel kullanıcılar, diğer kullanıcılardan bir veya daha fazlasını takım üyesi olarak belirleyebilir. Takım üyeliği, kullanıcılar arasında dosya paylaşımı ve ortak dosya düzenlemeye olanak sağlayan soyut bir kavramdır.

Projede takım oluşturma kodları teams_management.py dosyasında bulunmaktadır.

```
ss TeamManagement:

def create_team_window(self, current_window, username,open_main
    start_time=datetime.now()
    team_window = tk.Toplevel(current_window)
    team_window.title("Takım Oluştur")
    team_window.geometry('500x400')

    # Mevcut kullanıcıları yükle
    users = self.load_users()
    team_members = []

    # Kullanıcı listesi
    tk.Label(team_window, text="Kullanıcılar").pack(pady=10)
    listbox = tk.Listbox(team_window, selectmode=tk.MULTIPLE, w
    for user in users:
        if user != username:  # Kendisi hariç
            listbox.insert(tk.END, user)
    listbox.pack(pady=10)
```

Şekil 21

Şekil 21'de takım oluşturmak için yazılan create_team_window metodu bulunmaktadır.

Şekil 22

Takım oluşturmak isteyen kullanıcı arayüzdeki takım oluştur butonuna basarak takımını oluşturabilir. Ekranda görünen kullanıcılar kayıt.txt dosyasından kullanıcıların adları alınmaktadır.

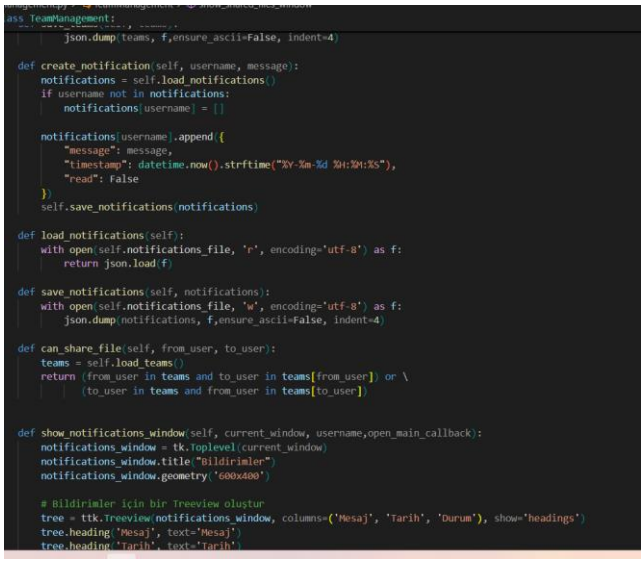
```
teams.json > ...
{
    "Melis ": [
        "Gizem",
        "Deniz",
        "Ebrar"
    ],
    "Gizem": [
        "Melis "
    ],
    "Deniz": [
        "Melis "
    ],
    "Ebrar": [
        "Melis "
    ]
}
```

Şekil 23

Şekil 23'de teams.json dosyamın ekran görüntüsü bulunmkatadır. Teams.json dosyasında oluşturulmuş takımlar mevcuttur.

- Belirleme durumunda, takım üyesi belirlenen kullanıcının profiline konuyla ilgili bildirim iletilmelidir.

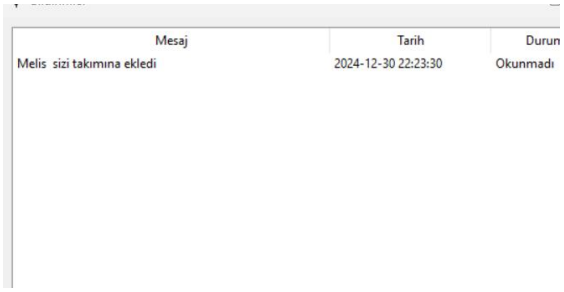
Kullanıcıya ait bildirimler ,bildirimler butonuna basıldığında görüntülenmektedir.Bildirim gösterme metotları teams_management.py dosyasında bulunmaktadır.



```
ass TeamManagement:
        json.dump(teams, f,ensure_ascii=False, indent=4)

    def create_notification(self, username, message):
        notifications = self.load_notifications()
        if username not in notifications:
            notifications[username] = []

        notifications[username].append({
            "message": message,
            "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            "read": False
        })
        self.save_notifications(notifications)

    def load_notifications(self):
        with open(self.notifications_file, 'r', encoding='utf-8') as f:
            return json.load(f)

    def save_notifications(self, notifications):
        with open(self.notifications_file, 'w', encoding='utf-8') as f:
            json.dump(notifications, f,ensure_ascii=False, indent=4)

    def can_share_file(self, from_user, to_user):
        teams = self.load_teams()
        return (from_user in teams and to_user in teams[from_user]) or \
               (to_user in teams and from_user in teams[to_user])


    def show_notifications_window(self, current_window, username,open_main_callback):
        notifications_window = tk.Toplevel(current_window)
        notifications_window.title("Bildirimler")
        notifications_window.geometry('600x400')

        # Bildirimler için bir Treeview oluştur
        tree = ttk.Treeview(notifications_window, columns=('Mesaj', 'Tarih', 'Durum'), show='headings')
        tree.heading('Mesaj', text='Mesaj')
        tree.heading('Tarih', text='Tarih')
```

Şekil 24

Şekil 24'de bildirim olışturma bildirimleri kaydetme ve bildirimleri arayüzdeki göstermek için verilen kodlar vardır.



Şekil 25

Şekil 25'de kullanıcı bildirimler butonuna bastığında kendine gelen bildirimleri gösterir

```
"Melis": [
    {
        "message": "Melis kullanıcısının şifre değiştirme isteği onaylandı",
        "timestamp": "2024-12-30 22:14:34",
        "read": false
    }
],
"Gizem": [
    {
        "message": "Melis  sizi takımına ekledi",
        "timestamp": "2024-12-30 22:23:30",
        "read": false
    }
],
"Deniz": [
    {
        "message": "Melis  sizi takımına ekledi",
        "timestamp": "2024-12-30 22:23:30",
        "read": false
    }
],
"Ebrar": [
    {
        "message": "Melis  sizi takımına ekledi",
        "timestamp": "2024-12-30 22:23:30",
        "read": false
    }
```

Şekil 26

Şekil 26'da bütün bildirimlerin bulunduğu notifications.json dosyasının ekran görüntüsü vardır.

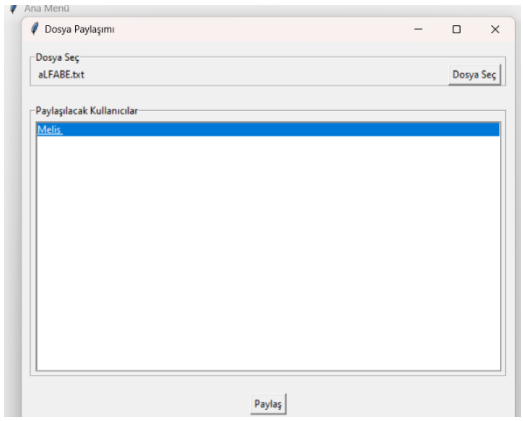
- Bireysel kullanıcılar, takım üyesi belirledikten sonra takım üyesi oldukları kullanıcılar ile dosya paylaşımında bulunabilir.

Dosya paylaşım kodları teams_management.py dosyasında mevcuttur.

```
def show_file_sharing_window(self, current_window, username,open_main_callback):
    start_time=datetime.now()
    share_window = tk.Toplevel(current_window)
    share_window.title("Dosya Paylaşımı")
    share_window.geometry('600x500')

    # Takım üyelerini yükle
    teams = self.load_teams()
    team_members = teams.get(username, [])

    # Dosya seçme bölümü
    file_frame = tk.LabelFrame(share_window, text="Dosya Seç")
    file_frame.pack(pady=10, padx=10, fill=tk.X)

    selected_file = tk.StringVar()
    file_label = tk.Label(file_frame, textvariable=selected_file)
    file_label.pack(side=tk.LEFT, padx=5)

    def select_file():
        filename = filedialog.askopenfilename()
        if filename:
            selected_file.set(os.path.basename(filename))
            share_button['state'] = 'normal'

    tk.Button(file_frame, text="Dosya Seç", command=select_file).pack(side=tk.RIGHT,

    # Kullanıcı listesi
    users_frame = tk.LabelFrame(share_window, text="Paylaşılacak Kullanıcılar")
    users_frame.pack(pady=10, padx=10, fill=tk.BOTH, expand=True)
```

Şekil 27

Şekil 27'de dosya paylaşımı için yazılmış kodumun birazı mevcutur.



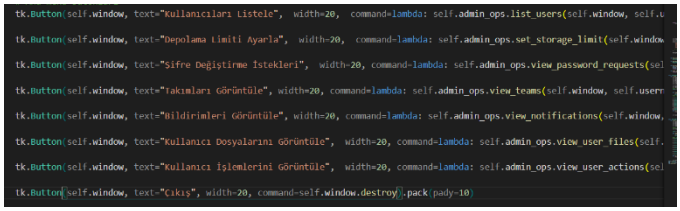
Şekil 28

Şekil 28'de dosya paylaşım arayüzü mevcuttur

2. Sistem Yöneticileri

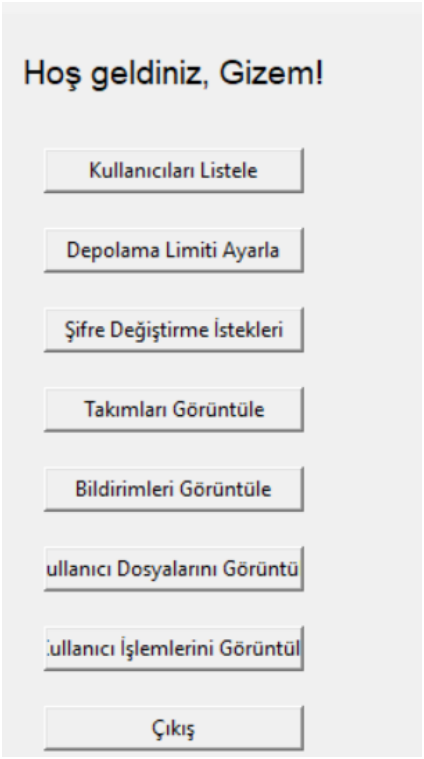
   Sistem yöneticileri için yazılmış kodlar admin_management.py dosyasında .

- Yapabileceği İşlemler: Kullanıcı profillerini yönetme, depolama limitlerini kontrol etme.



Şekil 29

Şekil 29 ‘da yönetici işlemlerinin yapılabilmesi için tasarlanmış butonlar mevcut.

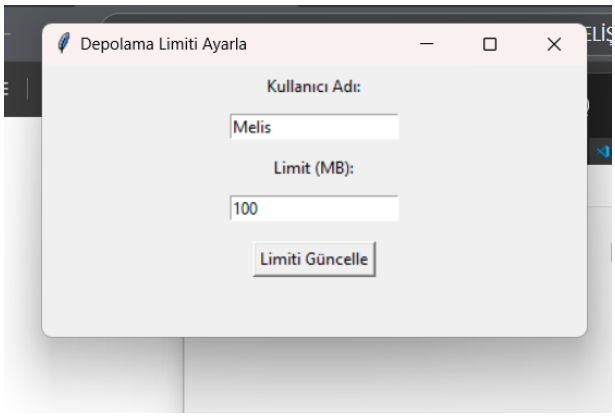


Şekil 30

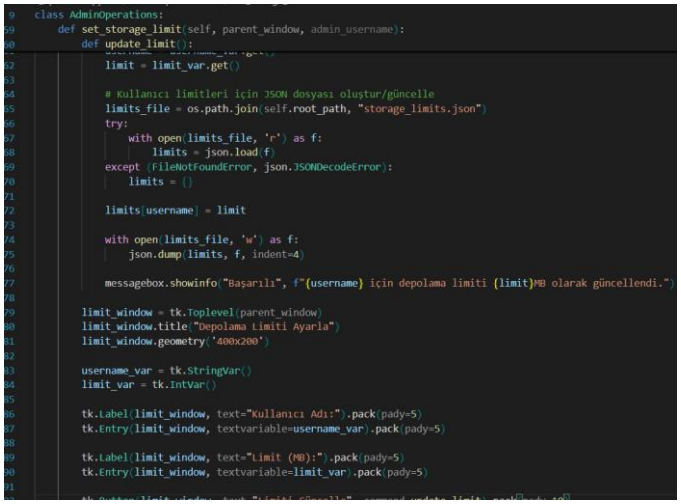
Şekil 30’da butonların arayüzdeki hali mevcuttur.

- Kullanıcıların depolama limitlerini arayüzdeki sistem yöneticisi profili üzerinden belirleyebilmelidir.

Kullanıcı limitleri belirleme kodları admin_operitions.py dosyaında



Şekil 31



```
class AdminOperations:
    def set_storage_limit(self, parent_window, admin_username):
        def update_limit():
            limit = limit_var.get()

            # Kullanıcı limitleri için JSON dosyası oluştur/güncelle
            limits_file = os.path.join(self.root_path, "storage_limits.json")
            try:
                with open(limits_file, 'r') as f:
                    limits = json.load(f)
            except (FileNotFoundError, json.JSONDecodeError):
                limits = {}

            limits[username] = limit

            with open(limits_file, 'w') as f:
                json.dump(limits, f, indent=4)

            messagebox.showinfo("Başarılı", f"{username} için depolama limiti {limit}MB olarak güncellendi.")

        limit_window = tk.Toplevel(parent_window)
        limit_window.title("Depolama Limiti Ayarla")
        limit_window.geometry("400x200")

        username_var = tk.StringVar()
        limit_var = tk.IntVar()

        tk.Label(limit_window, text="Kullanıcı Adı:").pack(pady=5)
        tk.Entry(limit_window, textvariable=username_var).pack(pady=5)

        tk.Label(limit_window, text="Limit (MB):").pack(pady=5)
        tk.Entry(limit_window, textvariable=limit_var).pack(pady=5)
```

Şekil 32

Şekil 32’de set_storage_limit kodu yani limit güncelleme kodu vardır.Arayüzde limit güüncelle butonuna basınca işlemler yapılıyor.

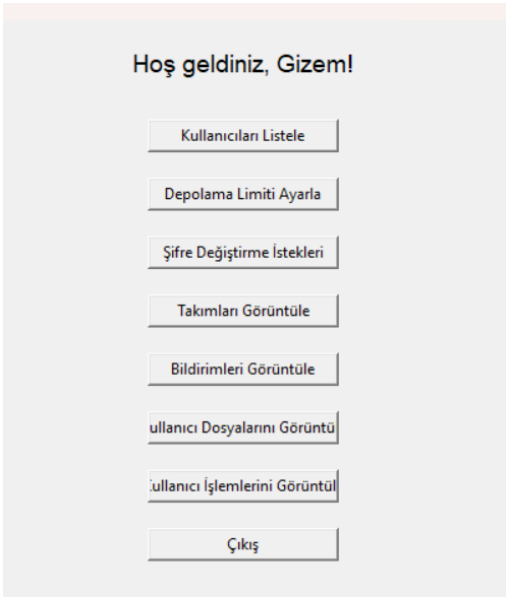
- Herhangi bir kullanıcıdan talep geldiğinde parola değiştirme isteğine onay verebilmelidir.

Şekil 33

```python
def view_password_requests(self, parent_window, admin_username):
    """Parola değiştirme isteklerini görüntüler"""
    try:
        with open(self.password_requests_file, 'r', encoding='utf-8') as file:
            lines = file.readlines()  # Dosyadaki tüm satırları oku

        requests = []  # İstekleri saklamak için bir liste
        current_request = {}  # Her bir isteği geçici olarak saklayacağımız sözlük

        for line in lines:
            line = line.strip()  # Satırdaki boşlukları temizle
            if line == "---":  # Bu, isteklerin arasındaki ayırıcı
                if current_request:
                    requests.append(current_request)  # İstek bitince listeye ekle
                current_request = {}  # Yeni istek için boş bir sözlük oluştur
            elif line.startswith("Kullanıcı:"):
                current_request['username'] = line.replace("Kullanıcı:", "").strip()  # Kull
            elif line.startswith("Sebep:"):
                current_request['reason'] = line.replace("Sebep:", "").strip()  # Sebep
            elif line.startswith("Tarih:"):
                current_request['date'] = line.replace("Tarih:", "").strip()  # Tarih
            elif line.startswith("Durum:"):
                current_request['status'] = line.replace("Durum:", "").strip()  # Durum

        if not requests:
            messagebox.showinfo("Bilgi", "Bekleyen şifre değiştirme isteği bulunmamaktadır."
            return

        # Yeni pencere oluştur
```

Şekil 34

- Sistemdeki herhangi bir kullanıcının dokümanlarına, paylaşımlarına, parolanın şifrelenmiş haline, log dosyalarına erişim sağlayabilmelidir.



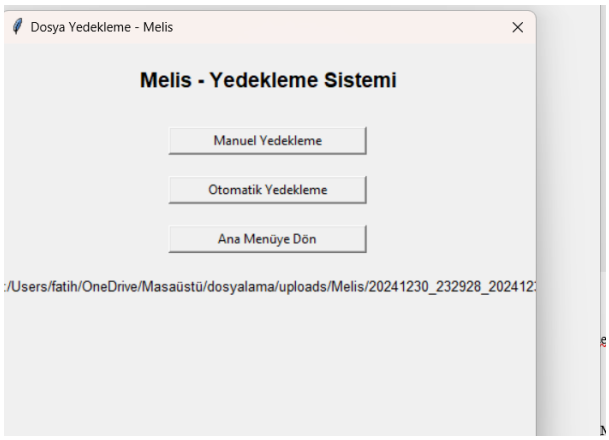
Şekil 35

Kullanıcı bütün bu işlemleri arayüzdee verilen butonlar sayesinde yapabilir.

C. Dosya Yedekleme ve Senkronizasyon Modülü

- Yedekleme

  Projede bulunan yedekleme kodları bakup_module.py dosyası içindedir.



Şekil 36

```python
backup_module.py > ...
import os
import shutil
import threading
import time
from datetime import datetime
import tkinter as tk
from tkinter import filedialog, messagebox
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
from system_logger import LogType, StatusCode, SystemLogger

# Yedekleme işlemlerini yapan thread fonksiyonu
def backup_thread(file_path, handler, progress_callback):
    try:
        handler.backup_file(file_path)
        progress_callback("Başarıyla yedeklendi: " + file_path)
    except Exception as e:
        progress_callback(f"Hata oluştu: {str(e)}")

class BackupHandler(FileSystemEventHandler):
    def __init__(self, username, source_dir, backup_dir, progress_callback=None):
        self.username = username
```

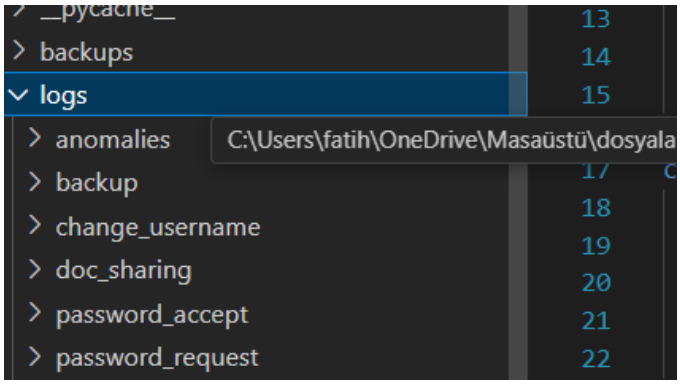
Şekil 37

- Raporlama

  Projedeki log dosyaarı system_logger.py dosyasında yazılmıştır.Bütün log dosyaları logs klasörünün üçinde mevcuttur.

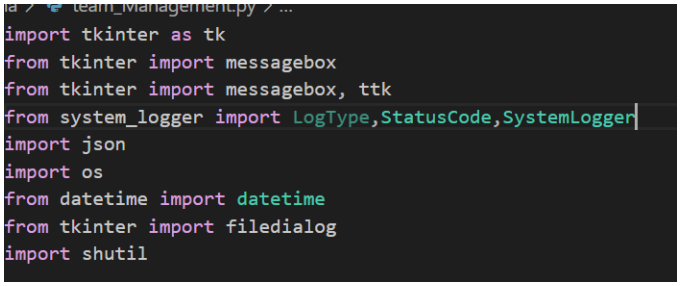
```python
system_logger.py > SystemLogger > log_doc_sharing
from enum import Enum
import os
import json
from datetime import datetime
import logging

class LogType(Enum):
    TEAM_MANAGEMENT = "TEAM_MANAGEMENT"  # Takım üyesi belirleme
    DOC_SHARING = "DOC_SHARING"      # Doküman paylaşımı
    PASSWORD_REQUEST = "PASSWORD_REQUEST" # Parola değiştirme talebi
    PASSWORD_ACCEPT = "PASSWORD_ACCEPT" # Parola değiştirme onayı
    PROFILE_ACCESS = "PROFILE_ACCESS"   # Profile giriş
    CHANGE_USERNAME = "CHANGE_USERNAME"
    BACKUP = "BACKUP"           # Yedekleme
    ANOMALY = "ANOMALY"         # Anormal durum

class StatusCode(Enum):
    SUCCESS = "SUCCESS"          # Başarılı
    FAILED = "FAILED"           # Başarısız
    PENDING = "PENDİNG"          # Beklemede
    APPROVED = "APPROVED"         # Onaylandı
    REJECTED = "REJECTED"         # Reddedildi

class SystemLogger:
    def __init__(self, root_path):
```

Şekil 38



Şekil 39

II. MATERYAL VE YÖNTEM



Şekil 40

Projede json modülü kullnarak json dosyalarının kullanımı sağlanmıştır.

Datetime modülü ile tarih ve saat işlemleri yapılmıştır.

Os modülü ile dosya işlemleri yapılmıştır.

Tkinter modülü ile arayüz tasarımı yapılmıştır

Logging modülü ile log işlemlerinin takibi yapılmıştır.

Shutil modülü ile dosya ve dizin işlemleri yapılmıştır.

III. SONUÇLAR VE TARTIŞMA

Proje adımları gerçekleştirilerek dosya yedekleme ve paylaşım projesi yapılmıştır.
Uygulamada kullanıcı tiplerine göre özellikler olup hangi kuıllanıcının hangi işlemi yapacağı kontroller ile sağlanmıştır.
Yapılan işlemler projenin giriş bölümünde ekran görüntüleri ile detaylıca anlatılmıştır.

4.KAYNAKLAR

1.https://youtube.com/playlist?list=PLSmHiN0iazy_qX_6Tmecj4tTOefqh2-m2&si=LQvdLY5VLMwarcdR

2.https://youtube.com/playlist?list=PLWctyKyPphPiul3WbHkniANLqSheBVP3O&si=6rWxYMA8IlOTgl_v