Grand Prix Ticketing Experience

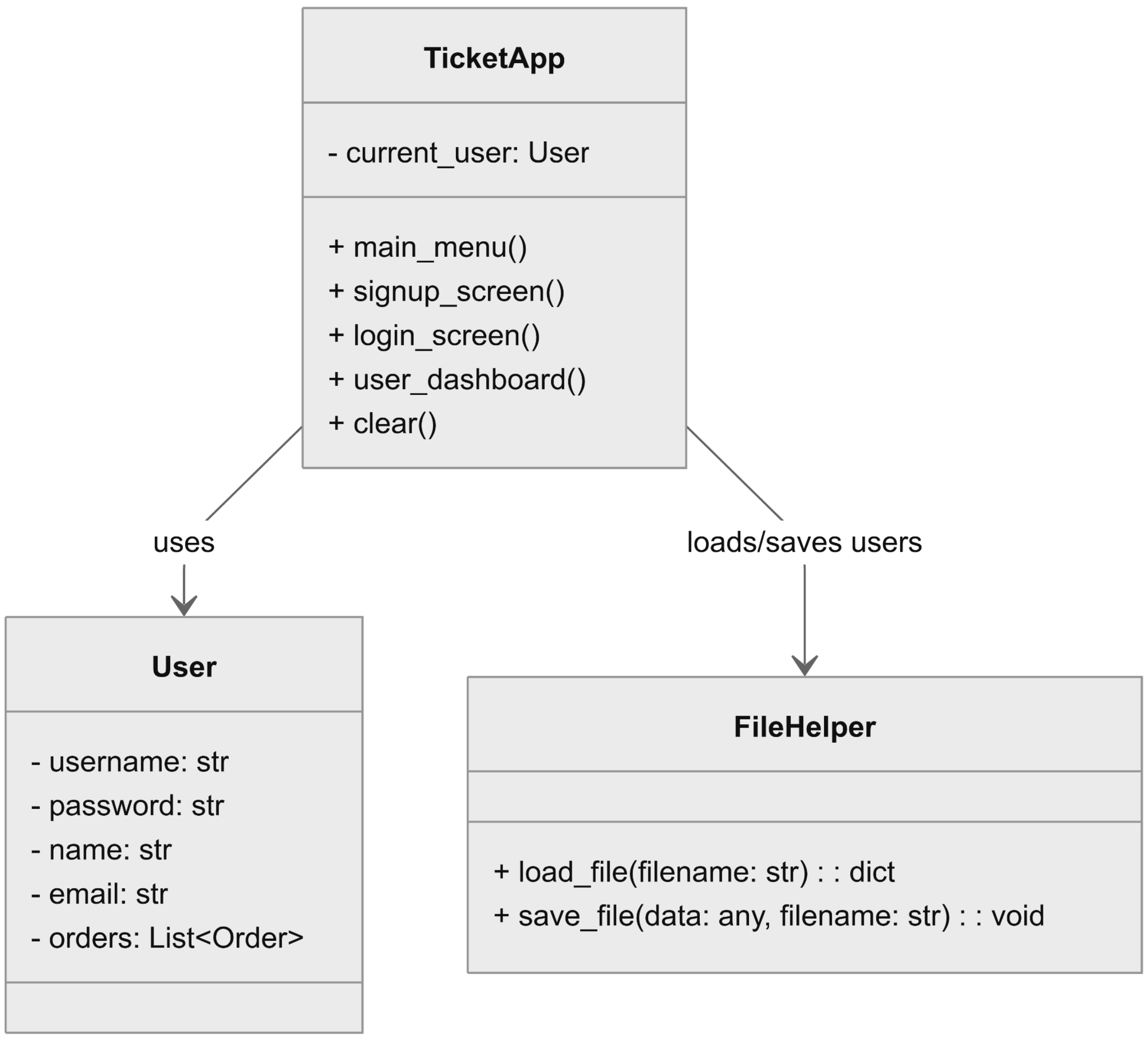
Mohammad Ismail 202105609

Nasser Lootah 202221929

ICS220 > 21383 Program. Fund.

May 13, 2025

UML Class Diagram:



User

- Attributes: username, password, name, email, and a list of orders
- Each User can create multiple Order objects. These are stored in the orders list.
- The class models a customer account and handles individual purchase history.

Order

- Attributes: ticket, date, payment_method, and race_date
- Each Order object stores details of a single ticket purchase made by a user.

- It references ticket type as a string, not a direct object, to keep storage simple.

Ticket

- Attributes: ticket_type, price, validity, and features
- Represents the available ticket options in the system.
- Tickets are predefined in the code and not editable by the admin.

TicketApp

- Attribute: current_user for session tracking
- Methods handle all GUI views and user actions: login, signup, ticket purchase, admin panel
- Manages system logic and controls access to users, orders, and tickets

Relationships

- User has a one-to-many relationship with Order (composition)
- Order is associated with Ticket (by ticket type string, not object reference)
- TicketApp acts as the controller, handling GUI events, data saving, and class interaction

Assumptions

- Tickets are hardcoded for simplicity and to meet the requirement of disabling admin ticket creation
- Data is stored using pickle in separate binary files: users.pkl, orders.pkl, and discount.pkl
- No need for inheritance as all classes serve distinct roles without overlapping behavior
- Errors like invalid input or missing files are handled gracefully using basic exception handling

Full Code:

```
import tkinter as tk
from tkinter import messagebox
import pickle
import os

class User:
   def __init__(self, username, password, name, email):
       self.username = username
       self.password = password
       self.name = name
       self.email = email
       self.orders = []  # Placeholder for future order support


def load_file(filename):
   if os.path.exists(filename):
       with open(filename, 'rb') as f:
           return pickle.load(f)
   return {}

def save_file(data, filename):
   with open(filename, 'wb') as f:
       pickle.dump(data, f)

users = load_file('users.pkl')


class TicketApp:
   def __init__(self, root):
       self.root = root
       self.root.title("Grand Prix Booking - Part 1")
       self.root.geometry("600x400")
       self.current_user = None
       self.main_menu()

   def main_menu(self):
       self.clear()
       tk.Label(self.root, text="Welcome to Grand Prix Booking",
font=("Arial", 18)).pack(pady=20)
       tk.Button(self.root, text="Login", width=30,
command=self.login_screen).pack(pady=10)
       tk.Button(self.root, text="Sign Up", width=30,
command=self.signup_screen).pack(pady=10)

   def signup_screen(self):
       self.clear()
       tk.Label(self.root, text="Create New Account", font=("Arial",
14)).pack(pady=10)
       entries = {}
       for field in ["Username", "Password", "Name", "Email"]:
           tk.Label(self.root, text=field).pack()
           entry = tk.Entry(self.root, show="*" if field == "Password" else
"")
           entry.pack()
           entries[field.lower()] = entry

       def create_user():
           u = entries['username'].get()
           if u in users:
               messagebox.showerror("Error", "Username already exists")
               return
           users[u] = User(
               u,
               entries['password'].get(),
               entries['name'].get(),
               entries['email'].get()
           )
           save_file(users, 'users.pkl')
           messagebox.showinfo("Success", "Account created")
           self.main_menu()

       tk.Button(self.root, text="Create Account",
command=create_user).pack(pady=10)
       tk.Button(self.root, text="Back", command=self.main_menu).pack()

   def login_screen(self):
       self.clear()
       tk.Label(self.root, text="Login", font=("Arial", 14)).pack(pady=10)
       tk.Label(self.root, text="Username").pack()
       username_entry = tk.Entry(self.root)
       username_entry.pack()
       tk.Label(self.root, text="Password").pack()
       password_entry = tk.Entry(self.root, show="*")
       password_entry.pack()

       def login():
           u = username_entry.get()
           p = password_entry.get()
           if u in users and users[u].password == p:
               self.current_user = users[u]
               self.user_dashboard()
           else:
               messagebox.showerror("Error", "Invalid login")

       tk.Button(self.root, text="Login", command=login).pack(pady=10)
       tk.Button(self.root, text="Back", command=self.main_menu).pack()

   def user_dashboard(self):
       self.clear()
       tk.Label(self.root, text=f"Welcome {self.current_user.name}",
font=("Arial", 16)).pack(pady=20)
       tk.Button(self.root, text="Logout", width=30,
command=self.main_menu).pack(pady=10)

   def clear(self):
       for widget in self.root.winfo_children():
           widget.destroy()

root = tk.Tk()
app = TicketApp(root)
root.mainloop()
```

File Structure Explanation

The system uses the `pickle` library to store user account data persistently in binary format. At this stage, only one file is created and managed during program execution:

1. users.pkl

- Stores all registered user accounts.
- Each `User` object includes:
    - `username`, `password`, `name`, `email`, and a placeholder list for future ticket orders.

File Handling Logic

- When the application starts, `users.pkl` is loaded using `load_file()`.
- When a new user registers, the updated data is saved using `save_file()`.
- The system uses a dedicated file for users to keep account data isolated and easily manageable.

Assurance

- Before loading data, the program checks if the file exists using `os.path.exists()` to prevent file-related errors.
- If the file doesn't exist, an empty dictionary is initialized instead.
- This ensures smooth first-time usage without manual file setup.

Github repository link: