مستندات سامانه سفارش غذا

پروژه پایانی درس برنامه‌نویسی پیشرفته

دانشکده: مهندسی کامپیوتر
استاد: دکتر رفیعی

تهیه‌کننده:
حمیدرضا کاظمی

شماره دانشجویی: ۴۰۱۴۷۲۰۱۶

تاریخ: بهمن‌ماه ۱۴۰۴

فهرست مطالب

۱ مقدمه و معرفی سیستم ۳
۱.۱ هدف پروژه ۳
۲.۱ ویژگی‌های اصلی سیستم ۳
۳.۱ اهداف آموزشی ۳

۲ معماری و طراحی سیستم ۴
۱.۲ معماری کلی سیستم ۴
۱.۱.۲ رابط گرافیکی ۴
۲.۱.۲ منطق کسب‌وکار ۴
۳.۱.۲ فایل‌های داده ۵
۲.۲ نمودار کلاس‌ها ۵
۱.۲.۲ کلاس‌های اصلی ۵
۲.۲.۲ رابطه‌های کلاس‌ها ۵
۳.۲ طراحی شیءگرا ۶
۱.۳.۲ کپسوله‌سازی (Encapsulation) ۶
۲.۳.۲ چندریختی (Polymorphism) ۶
۳.۳.۲ وراثت (Inheritance) ۷

۳ پیاده‌سازی جزئیات ۸
۱.۳ مدل‌های داده ۹
۱.۱.۳ کلاس Food ۹
۲.۱.۳ کلاس Order ۹
۲.۳ سیستم احراز هویت ۹
۱.۲.۳ ثبت‌نام مشتری ۱۰
۲.۲.۳ اعتبارسنجی ورودی‌ها ۱۰
۳.۳ مدیریت منوی غذا ۱۱
۱.۳.۳ الگوریتم کنترل موجودی ۱۱
۲.۳.۳ فیلتر غذا بر اساس تاریخ ۱۱
۳.۳.۳ مقایسه قیمت با رقبا ۱۲
۴.۳.۳ ثبت نظر ۱۳

۴ سیستم سفارش و پرداخت ۱۵
۱.۴ فرآیند سفارش ۱۵
۱.۱.۴ الگوریتم ثبت سفارش ۱۵
۲.۱.۴ پیاده‌سازی checkout ۱۵
۲.۴ سیستم پرداخت ۱۶
۱.۲.۴ شبیه‌سازی پرداخت آنلاین ۱۶
۲.۲.۴ لغو سفارش و بازگشت موجودی ۱۷

۵ سیستم امتیازات و گزارش‌ها ۱۸
۱.۵ سیستم وفاداری ۱۸
۱.۱.۵ محاسبه امتیازات ۱۸
۲.۵ گزارش‌های مدیریتی ۱۹
۱.۲.۵ گزارش فروش ۱۹
۲.۲.۵ نمودار فروش ۲۰

۶ پایگاه داده و ذخیره‌سازی ۲۱
۱.۶ ساختار فایل‌های CSV ۲۱
۱.۱.۶ جدول users.csv ۲۱
۲.۱.۶ جدول foods.csv ۲۱
۲.۶ کلاس Database ۲۱
۱.۲.۶ متدهای اصلی ۲۱

۷ رابط کاربری گرافیکی ۲۳
۱.۷ معماری GUI ۲۳
۱.۱.۷ کلاس اصلی برنامه ۲۳
۲.۱.۷ صفحه ورود ۲۳
۳.۱.۷ منوهای اصلی ۲۳

۸ تست‌های واحد ۲۵
۱.۸ ساختار تست‌ها ۲۵
۱.۱.۸ کلاس‌های تست ۲۵
۲.۱.۸ تست‌های Edge Case ۲۶
۲.۸ پوشش تست ۲۷
۱.۲.۸ آمار تست‌ها ۲۷

۹ نتیجه‌گیری و جمع‌بندی ۲۸
۱.۹ دستاوردهای پروژه ۲۸
۱.۱.۹ ویژگی‌های پیاده‌سازی شده ۲۸
۲.۱.۹ چالش‌های حل شده ۲۸
۲.۹ نتیجه‌گیری نهایی ۲۸

۱۰ ضمایم ۲۹
۱.۱۰ کدهای تکمیلی ۲۹
۱.۱.۱۰ کلاس Cart ۲۹
۲.۱.۱۰ کلاس DiscountCode ۳۰

فصل ۱

مقدمه و معرفی سیستم

۱.۱ هدف پروژه

سامانه سفارش غذا یک سیستم جامع و کاربردی برای مدیریت فرآیند سفارش غذا به صورت آنلاین است که با هدف پیاده‌سازی عملی مفاهیم پیشرفته برنامه‌نویسی شیءگرا و معماری نرم‌افزار طراحی شده است. این سیستم کلیه فرآیندهای یک پلتفرم سفارش غذای واقعی را شبیه‌سازی می‌کند و قابلیت توسعه و گسترش در محیط‌های عملیاتی را دارا می‌باشد.

۲.۱ ویژگی‌های اصلی سیستم

- طراحی کامل شیءگرا با معماری لایه‌ای
- سیستم احراز هویت امن با اعتبارسنجی جامع
- مدیریت کامل منوی غذا با کنترل موجودی و تاریخ
- سبد خرید پیشرفته با قابلیت پرداخت
- سیستم امتیازات وفاداری و کد تخفیف
- پنل مدیریت با گزارش‌های تحلیلی
- رابط کاربری گرافیکی با Tkinter
- پایگاه داده مبتنی بر CSV با استفاده از pandas

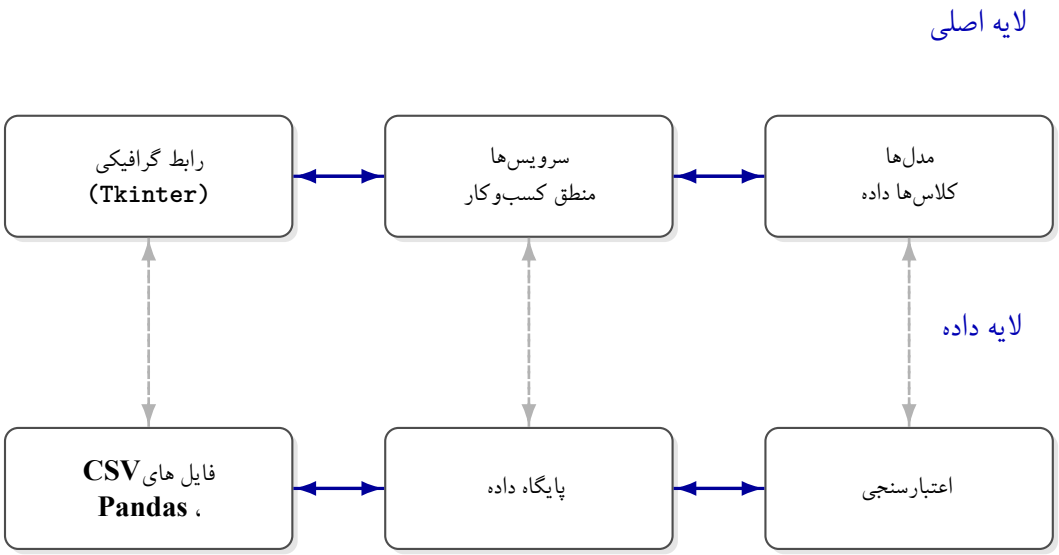
۳.۱ اهداف آموزشی

۱. پیاده‌سازی اصول طراحی شیءگرا
۲. پیاده‌سازی اعتبارسنجی و امنیت
۳. کار با ساختار داده
۴. توسعه رابط کاربری گرافیکی
۵. نوشتن تست‌های واحد جامع

فصل ۲

معماری و طراحی سیستم

۱.۲ معماری کلی سیستم



شکل ۱.۲: نمودار معماری سیستم

۱.۱.۲ رابط گرافیکی

رابط کاربری گرافیکی با استفاده از Tkinter است که امکان تعامل کاربر با سیستم را فراهم می‌کند.

۲.۱.۲ منطق کسب‌وکار

شامل سرویس‌های مختلف که منطق کسب‌وکار را پیاده‌سازی می‌کنند:

- **Model**: تعریف کلاس‌های داده و ساختار اصلی سیستم
- **AuthManager**: مدیریت احراز هویت
- **FoodService**: مدیریت غذاها
- **OrderService**: مدیریت سفارشات

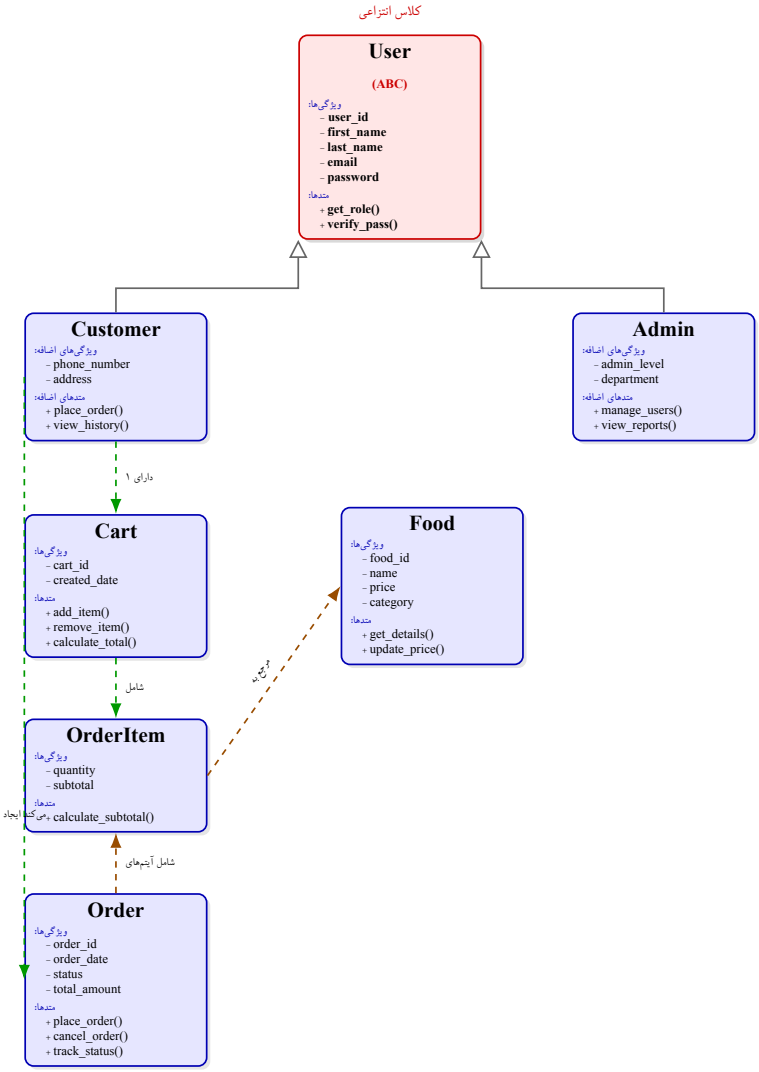
- **CustomerService**: خدمات مشتری
- **AdminService**: خدمات ادمین

۳.۱.۲ فایل‌های داده

مدیریت ذخیره‌سازی و بازیابی داده‌ها با استفاده از کلاس Database و فایل‌های CSV.

۲.۲ نمودار کلاس‌ها

۱.۲.۲ کلاس‌های اصلی



شکل ۲.۲: نمودار کلاس‌های سیستم سفارش غذا

۲.۲.۲ رابطه‌های کلاس‌ها

- وراثت: Customer و Admin از User ارث‌بری می‌کنند

- ترکیب: Cart شامل لیستی از OrderItem است
- ارتباط: Order با OrderItem و Food ارتباط دارد

۳.۲ طراحی شیءگرا

۱.۳.۲ کپسوله‌سازی (Encapsulation)

```
class User(ABC):
    def __init__(self, user_id: str, first_name: str,
                 last_name: str, email: str, password: str):
        self._user_id = user_id  # Private attribute
        self.first_name = first_name
        self.last_name = last_name
        self.email = email
        self._password = password  # Private attribute

    @property
    def user_id(self) -> str:
        return self._user_id

    @property
    def password(self) -> str:
        return self._password
```

۲.۳.۲ چندریختی (Polymorphism)

```
class User(ABC):
    @abstractmethod
    def get_role(self) -> str:
        pass

class Customer(User):
    def get_role(self) -> str:
        return "Customer"

class Admin(User):
    def get_role(self) -> str:
        return "Admin"
```

۳.۳.۲ وراثت (**Inheritance**)

```python
class User(ABC):
    # Base class implementation
    pass

class Customer(User):
    # Inherits from User, adds customer-specific features
    def __init__(self, user_id: str, first_name: str,
                 last_name: str, email: str, password: str,
                 phone: str, national_code: str):
        super().__init__(user_id, first_name, last_name, email, password)
        self.phone = phone
        self.national_code = national_code
```

فصل ۳

پیاده‌سازی جزئیات

```
├─ __pycache__
├─ .gitignore
├─ admin_service.py
├─ price_comparison.py
├─ auth.py
├─ customer_service.py
├─ database.py
├─ threaded_scraper.py
├─ food_service.py
├─ restaurant_scrapers.py
├─ gui_app.py
├─ model.py
├─ scraper.py
├─ order_service.py
├─ orders.csv
├─ README.md
└─ reviews.csv
```

۱.۳ مدل‌های داده

کلاس‌های مدل داده با استفاده از dataclass پیاده‌سازی شده‌اند که خوانایی و نگهداری کد را افزایش می‌دهد. کلاس Food شامل ویژگی‌هایی مانند موجودی، تاریخ‌های دسترسی و قیمت‌ها بوده و کلاس Order کلیه اطلاعات مربوط به سفارش را مدیریت می‌کند.

۱.۱.۳ کلاس Food

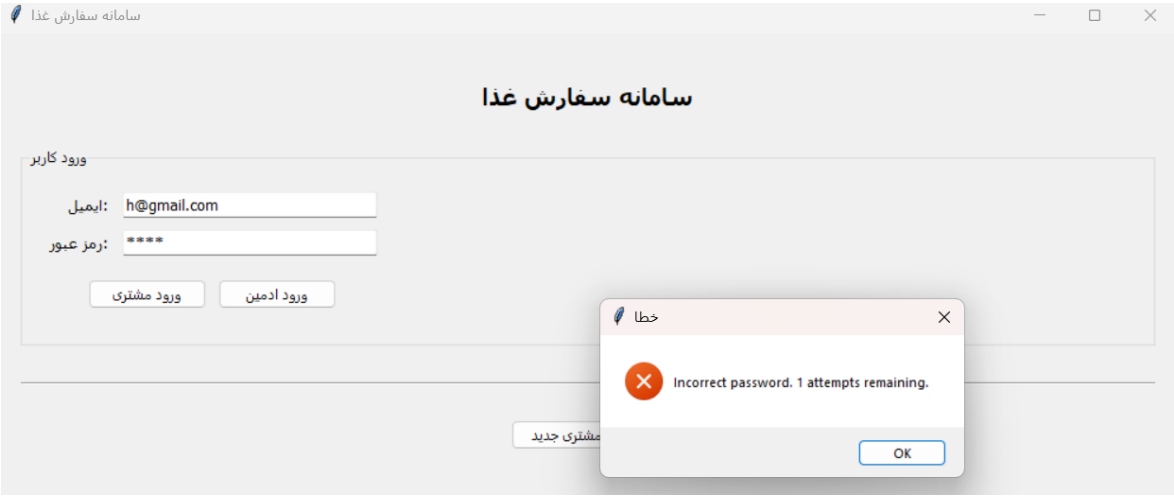
```
@dataclass
class Food:
    food_id: str
    restaurant_id: str
    name: str
    category: str
    selling_price: float
    cost_price: float
    ingredients: str
    description: str
    stock: int
    available_dates: List[date] = field(default_factory=list)

    def is_available(self, requested_quantity: int = 1) -> bool:
        return self.stock >= requested_quantity
```

۲.۱.۳ کلاس Order

```
class Order:
    STATUS_PENDING = "Pending"
    STATUS_PAID = "Paid"
    STATUS_SENT = "Sent"
    STATUS_CANCELLED = "Cancelled"

    def __init__(self, restaurant_id: str, order_id: str,
                 customer_id: str, items: List[OrderItem],
                 delivery_date: date, payment_method: str):
        self.restaurant_id = restaurant_id
        self.order_id = order_id
        self.customer_id = customer_id
        self.items = items
        self.order_date = datetime.now()
        self.delivery_date = delivery_date
        self.status = self.STATUS_PENDING
        self.payment_method = payment_method
        self._total_amount = sum(item.total_price for item in items)
        self.discount_amount: float = 0.0
```

۲.۳ سیستم احراز هویت

سیستم احراز هویت با قابلیت‌های ثبت‌نام، ورود و اعتبارسنجی جامع پیاده‌سازی شده است. اعتبارسنجی ورودی‌ها شامل بررسی قالب ایمیل، شماره تلفن، کد ملی و پیچیدگی رمز عبور می‌باشد. همچنین سیستم از مکانیزم قفل کردن حساب پس از تلاش‌های ناموفق متعدد پشتیبانی می‌کند.



شکل ۱.۳: خطای ورود کاربر درصورت اشتباه وارد کردن تا ۳ بار

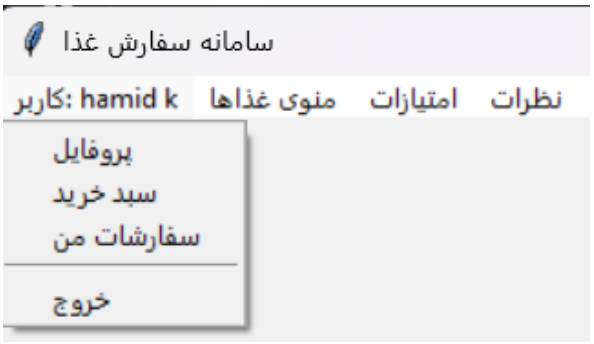
۱.۲.۳ ثبت‌نام مشتری

```python
def register_customer(self, first_name: str, last_name: str,
                      email: str, phone: str, national_code: str,
                      password: str, confirm_password: str,
                      address: str = "") -> tuple[bool, str]:
    # Validation checks
    if not self._validate_email(email):
        return False, "Invalid email format"

    if not self._validate_phone(phone):
        return False, "Invalid phone number"

    if password != confirm_password:
        return False, "Passwords do not match"

    # Create and save customer
    user_id = str(uuid.uuid4())
    new_customer = Customer(user_id, first_name, last_name,
                            email, password, phone,
                            national_code, address, 0)
    self.db.save_user(new_customer)

    return True, "Registration successful"
```

۲.۲.۳ اعتبارسنجی ورودی‌ها

ورودی	قالب صحیح	توضیح
ایمیل	name@domain.com	فرمت استاندارد ایمیل
تلفن	09xxxxxxxxx	۱۱ رقم، شروع با ۰۹
کد ملی	xxxxxxxxxx	۱۰ رقم عددی
رمز عبور	حداقل ۸ کاراکتر	حروف بزرگ، کوچک، عدد، کاراکتر ویژه

جدول ۱.۳: جدول اعتبارسنجی ورودی‌ها



شکل ۲.۳: امکانات منوی کاربری

۳.۳ مدیریت منوی غذا

منوی غذا با قابلیت‌های پیشرفته‌ای مانند فیلتر بر اساس تاریخ، کنترل موجودی زمان واقعی و جستجوی پیشرفته پیاده‌سازی شده است. الگوریتم کنترل موجودی از overselling جلوگیری کرده و موجودی را به صورت اتمی به‌روزرسانی می‌کند.

۱.۳.۳ الگوریتم کنترل موجودی

```
def add_to_cart(self, cart: Cart, food_id: str, quantity: int):
    # Check if quantity is positive
    if quantity <= 0:
        raise ValueError("Quantity must be positive")

    # Get food from database
    food_obj = self.get_food_by_id(food_id)
    if not food_obj:
        raise ValueError("Food not found")

    # Calculate current quantity in cart
    current_in_cart = 0
    for item in cart.items:
        if item.food.food_id == food_id:
            current_in_cart = item.quantity
            break

    # Validate stock availability
    if food_obj.stock < (current_in_cart + quantity):
        raise ValueError(f"Insufficient stock. Current: {food_obj.stock}"
    )

    # Add to cart
    cart.add_item(food_obj, quantity)
```

۲.۳.۳ فیلتر غذا بر اساس تاریخ

```
def get_menu_for_date(self, selected_date: date) -> List[Food]:
    df = self.db.load_foods()
    foods_list = []

    for _, row in df.iterrows():
        # Check if food is available on selected date
```

شکل ۳.۳: پنل گزارش‌گیری قیمت رقبا

```
        if selected_date in row['available_dates']:
            foods_list.append(self._parse_food_from_row(row))

    return foods_list
```

۳.۳.۳ مقایسه قیمت با رقبا

در این قسمت می‌توانید از طریق snappfood قیمت های سایر رستوران ها را بیابید که برای بررسی و قیمت‌گذاری مفید خواهد بود:

```
class BaseRestaurantScraper(ABC):
    """creating base class"""

    def __init__(self, name: str, base_url: str):
        self.name = name
        self.base_url = base_url
        self.headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36'
        }

    @abstractmethod
    def scrape_menu(self) -> List[Dict]:
        """for using in the GUI menu"""
        pass

    def fetch_page(self, url: str) -> str:
        """getting webpage"""
        try:
            response = requests.get(url, headers=self.headers, timeout=10)
            response.raise_for_status()
            return response.text
        except Exception as e:
            print(f"Error fetching {url}: {e}")
            return ""

    def save_to_csv(self, data: List[Dict], filename: str):
        "save to csv"
        df = pd.DataFrame(data)
        df.to_csv(filename, index=False, encoding='utf-8-sig')
```

شکل ۴.۳: ثبت نظر برای پنل مشتری

در بررسی هر item از مفهوم multithreading استفاده شده تا به سرعت آیتم ها استخراج شوند.

```
class ThreadedRestaurantScraper:
    def __init__(
        self,
        scraper_class: Type,
        max_workers: int = 3,
        output_file: Optional[str] = None,
        verbose: bool = True
    ):
        """
        Initialize threaded scraper

        Args:
            scraper_class: Scraper class to instantiate (e.g., SnappFoodScraper)
            max_workers: Maximum number of concurrent threads (default: 3)
            output_file: Optional CSV filename to save results
            verbose: Whether to print detailed logs (default: True)
        """
        self.scraper_class = scraper_class
        self.max_workers = max_workers
        self.output_file = output_file
        self.verbose = verbose
        self.results = []
        self.errors = []
```

۴.۳.۳ ثبت نظر

همچنین قابلیت ثبت نظر مشتری برای سفارشات قبلی وجود دارد.

```
def submit_review(self, customer_id: str, order_id: str, rating: int, comment: str):
        """Submit a review for a completed order"""
        order_row = self.db.get_order_by_id(order_id)
        if order_row is None:
            raise ValueError("Order not found")

        if order_row['customer_id'] != customer_id:
            raise ValueError("You can only review your own orders")

        # Validate rating range
        if rating < 1 or rating > 5:
            raise ValueError("Rating must be between 1 and 5")

        review = Review(
            review_id=str(uuid.uuid4()),
            customer_id=customer_id,
            order_id=order_id,
            rating=rating,
```

```
            comment=comment
        )
        self.db.save_review(review)

        points_for_review = 10 # points for each review
        self.db.add_loyalty_points(customer_id, points_for_review)
```

فصل ۴

سیستم سفارش و پرداخت

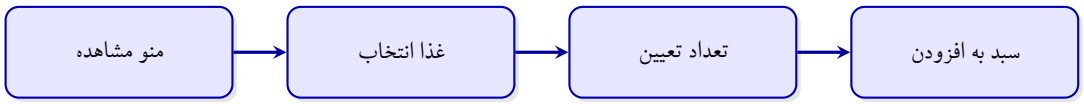
۱.۴ فرآیند سفارش

۱.۱.۴ الگوریتم ثبت سفارش

۲.۱.۴ پیاده‌سازی **checkout**

```python
def checkout(self, cart: Cart, customer_id: str,
             delivery_date: date, payment_method: str,
             discount_code_str: Optional[str] = None) -> Order:
    if not cart.items:
        raise ValueError("Cart is empty")

    # Validate all foods from same restaurant
    restaurant_ids = {item.food.restaurant_id for item in cart.items}
    if len(restaurant_ids) > 1:
        raise ValueError("All foods must be from same restaurant")

    restaurant_id = list(restaurant_ids)[0]
    order_id = str(uuid.uuid4())

    # Create order
    new_order = Order(
        restaurant_id=restaurant_id,
        order_id=order_id,
        customer_id=customer_id,
        items=cart.items,
        delivery_date=delivery_date,
        payment_method=payment_method
    )

    # Apply discount if provided
    if discount_code_str:
        discount_data = self.db.find_discount_code(discount_code_str)
        if discount_data and discount_data.is_valid():
            new_order.apply_discount(discount_data)
            self.db.mark_discount_code_used(discount_code_str)

    # Reduce food stock
    for item in cart.items:
        self.db.update_food_stock(
            item.food.food_id,
```

شکل ۱.۴: پنل ادمین



شکل ۲.۴: فرآیند سفارش غذا

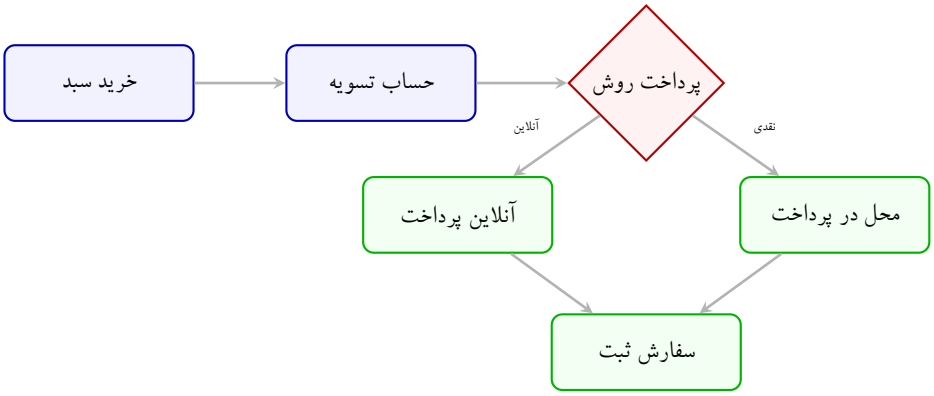
```
            item.food.stock - item.quantity
        )

    # Save order
    self.db.save_order(new_order)
    self.db.save_order_items(order_id, cart.items)

    # Clear cart and add loyalty points
    cart.clear()
    self.customer_service.add_purchase_points(
        customer_id,
        new_order.total_amount
    )

    return new_order
```

۲.۴ سیستم پرداخت

۱.۲.۴ شبیه‌سازی پرداخت آنلاین

```
def process_payment(self, order_id: str) -> bool:
    # Update order status
    self.db.update_order_status(order_id, Order.STATUS_PAID)

    # Get order details for loyalty points
    orders_df = pd.read_csv(self.db.orders_file)
    order_row = orders_df[orders_df['order_id'] == order_id]

    if not order_row.empty:
        order_data = order_row.iloc[0]
        customer_id = order_data['customer_id']
        total_amount = float(order_data['total_amount'])

        # Add loyalty points (1 point per 1000 units)
        points = int(total_amount // 1000)
        if points > 0:
```

```
            self.db.add_loyalty_points(customer_id, points)

    return True  # Payment successful
```



شکل ۳.۴: فرآیند پرداخت و ثبت سفارش

۲.۲.۴ لغو سفارش و بازگشت موجودی

```
def cancel_order(self, order_id: str):
    # Get order items
    items_df = self.db.get_order_items(order_id)

    if items_df.empty:
        raise ValueError("Order not found")

    # Restore food stock
    for _, row in items_df.iterrows():
        food_id = str(row["food_id"])
        quantity = int(row["quantity"])

        foods_df = self.db.load_foods()
        stock_row = foods_df[foods_df["food_id"] == food_id]

        if not stock_row.empty:
            current_stock = int(stock_row.iloc[0]["stock"])
            new_stock = current_stock + quantity
            self.db.update_food_stock(food_id, new_stock)

    # Update order status
    self.db.update_order_status(order_id, Order.STATUS_CANCELLED)
```

فصل ۵

سیستم امتیازات و گزارش‌ها

۱.۵ سیستم وفاداری

سیستم وفاداری به مشتریان امکان کسب امتیاز بر اساس میزان خرید را می‌دهد. هر ۱۰۰۰ تومان خرید معادل ۱ امتیاز محاسبه شده و کاربران می‌توانند امتیازات خود را به کد تخفیف ۱۰٪ تبدیل کنند.

۱.۱.۵ محاسبه امتیازات

$$\text{امتیازات} = \left\lfloor \frac{\text{مبلغ کل سفارش}}{۱۰۰۰} \right\rfloor \tag{۱.۵}$$

```python
def add_purchase_points(self, customer_id: str, total_amount: float):
    """Add loyalty points after purchase"""
    points = int(total_amount // 1000)
    if points > 0:
        self.db.add_loyalty_points(customer_id, points)

def generate_discount_code(self, customer_id: str, points_cost: int) -> DiscountCode:
    """Convert loyalty points to discount code"""
    if points_cost < 100:
        raise ValueError("Minimum 100 points required")

    # Deduct points
    self.db.deduct_loyalty_points(customer_id, points_cost)

    # Generate discount code
    code = f"LO-{uuid.uuid4().hex[:6].upper()}"
    discount = DiscountCode(
```

شکل ۱.۵: قابلیت محاسبه امتیاز در پنل مشتری

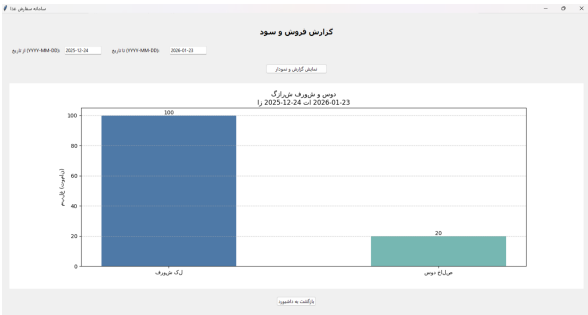
```
        code=code,
        discount_percentage=10.0,
        expiry_date=datetime.now() + timedelta(days=30)
    )

    self.db.save_discount_code(discount)
    return discount
```

۲.۵ گزارش‌های مدیریتی

۱.۲.۵ گزارش فروش

```
def get_sales_report(self, start_date: date, end_date: date) -> dict:
    orders_df = pd.read_csv(self.db.orders_file)
    items_df = pd.read_csv(self.db.order_items_file)
    foods_df = self.db.load_foods()

    # Filter orders by date range
    orders_df['order_date_parsed'] = pd.to_datetime(orders_df['order_date
    '])
    mask = (orders_df['order_date_parsed'].dt.date >= start_date) & \
           (orders_df['order_date_parsed'].dt.date <= end_date)
    filtered_orders = orders_df[mask]

    total_sales = 0.0
    total_profit = 0.0

    # Calculate sales and profit
    for _, order_row in filtered_orders.iterrows():
        order_id = order_row['order_id']
        order_items = items_df[items_df['order_id'] == order_id]

        for _, item_row in order_items.iterrows():
            food_id = str(item_row['food_id'])
            quantity = item_row['quantity']
            unit_price = float(item_row['unit_price'])

            total_sales += (unit_price * quantity)

            # Find cost price
            food_row = foods_df[foods_df['food_id'].astype(str) ==
    food_id]
            if not food_row.empty:
                cost_price = float(food_row.iloc[0]['cost_price'])
                total_profit += (unit_price - cost_price) * quantity

    return {
        "start_date": start_date.strftime("%Y-%m-%d"),
        "end_date": end_date.strftime("%Y-%m-%d"),
        "order_count": len(filtered_orders),
        "total_sales": total_sales,
        "total_profit": total_profit
    }
```

شکل ۲.۵: گزارش قیمت سایر رقبا



شکل ۳.۵: نمودار ترسیم شده برای محاسبه سود و میزان فروش

۲.۲.۵ نمودار فروش

```python
def plot_sales_chart(self, start_date, end_date):
    """Plot sales and profit chart"""
    report_data = self.get_sales_report(start_date, end_date)

    labels = ['Total Sales', 'Net Profit']
    values = [report_data['total_sales'], report_data['total_profit']]

    plt.figure(figsize=(8, 5))
    bars = plt.bar(labels, values, color=['skyblue', 'lightgreen'])

    plt.title(f'Sales Report: {start_date} to {end_date}')
    plt.ylabel('Amount (Toman)')

    # Add value labels on bars
    for bar in bars:
        yval = bar.get_height()
        plt.text(bar.get_x() + bar.get_width()/2, yval,
                 f'{int(yval):,}', ha='center', va='bottom')

    plt.tight_layout()
    plt.show()
```

فصل ۶

پایگاه داده و ذخیره‌سازی

۱.۶ ساختار فایل‌های CSV

۱.۱.۶ جدول users.csv

```
user_id,role,first_name,last_name,email,password,phone,national_code,
address,personnel_id,loyalty_points,failed_attempts,is_locked
e4b2fdbf-941a-4f7a-a955-a83c703b21df,Admin   ,    ,,
admin@restaurant.com,Admin@1234,09123456789,1234567890,,
1001,0,0,false
```

۲.۱.۶ جدول foods.csv

```
food_id,restaurant_id,name,category,selling_price,cost_price,
ingredients,description,stock,available_dates,
food_001,restaurant_001   ,        ,,50000,30000,
             ,          ,20,
["2024-12-20","2024-12-21"],pizza.jpg
```

۲.۶ کلاس Database

۱.۲.۶ متدهای اصلی

```python
class Database:
    def __init__(self):
        self.users_file = "users.csv"
        self.foods_file = "foods.csv"
        self.orders_file = "orders.csv"
        self.order_items_file = "order_items.csv"
        self.reviews_file = "reviews.csv"
        self.discount_codes_file = "discount_codes.csv"
        self._init_files()  # Initialize CSV files

    def _init_files(self):
        """Create CSV files with proper columns if they don't exist"""
        if not os.path.exists(self.users_file):
            cols = ['user_id', 'role', 'first_name', 'last_name',
                    'email', 'password', 'phone', 'national_code',
```

```
                    'address', 'personnel_id', 'loyalty_points',
                    'failed_attempts', 'is_locked']
            pd.DataFrame(columns=cols).to_csv(self.users_file, index=
    False)

        # Similar initialization for other files...

    def save_user(self, user: User):
        """Save a user to CSV file"""
        df = self.load_users()

        # Check for duplicate email
        if not df[df['email'] == user.email].empty:
            raise ValueError("Email already exists")

        # Prepare user data
        user_data = {
            'user_id': user.user_id,
            'role': user.get_role(),
            'first_name': user.first_name,
            'last_name': user.last_name,
            'email': user.email,
            'password': user.password,
            'phone': getattr(user, 'phone', None),
            'national_code': getattr(user, 'national_code', ""),
            'address': getattr(user, 'address', None),
            'personnel_id': getattr(user, 'personnel_id', None),
            'loyalty_points': getattr(user, 'loyalty_points', 0),
            'failed_attempts': 0,
            'is_locked': False
        }

        # Append and save
        pd.concat([df, pd.DataFrame([user_data])],
                  ignore_index=True).to_csv(self.users_file, index=False)
```

فصل ۷

رابط کاربری گرافیکی

۱.۷ معماری GUI

رابط کاربری گرافیکی با استفاده از Tkinter پیاده‌سازی شده . رابط کاربری کاملاً فارسی‌زبان بوده و برای کاربران ایرانی بهینه‌سازی شده است.

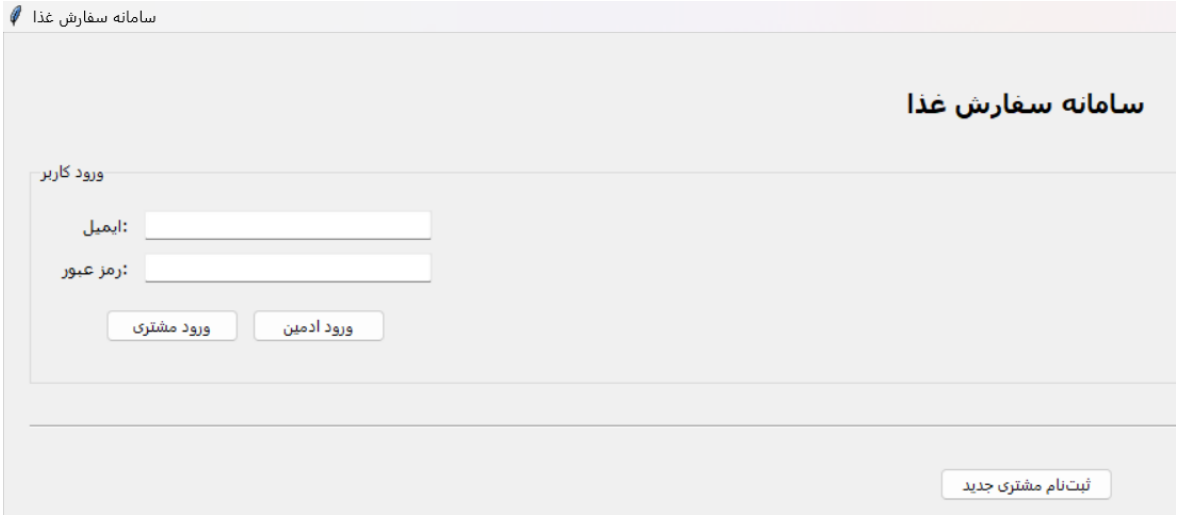
۱.۱.۷ کلاس اصلی برنامه

```
class FoodDeliveryApp:
    def __init__(self, root):
        self.root = root
        self.root.title("         ")
        self.root.geometry("1000x700")

        # Services initialization
        self.auth = AuthManager()
        self.food_service = FoodService()
        self.order_service = OrderService()
        self.customer_service = CustomerService()
        self.admin_service = AdminService()
        self.db = Database()

        # User state
        self.current_user = None
        self.user_role = None
        self.cart = Cart()
        self.selected_date = date.today()

        # Start with login page
        self.create_login_page()
```

۲.۱.۷ صفحه ورود

۳.۱.۷ منوهای اصلی

```
def create_customer_dashboard(self):
    # Create menu bar
    menubar = tk.Menu(self.root)
    self.root.config(menu=menubar)
```



شکل ۱.۷: صفحه ورود سیستم

```
    # User menu
    user_menu = tk.Menu(menubar, tearoff=0)
    menubar.add_cascade(label=f"   : {self.current_user.full_name}",
                        menu=user_menu)
    user_menu.add_command(label="    ", command=self.show_profile)
    user_menu.add_command(label="     ", command=self.show_cart)
    user_menu.add_command(label="      ",
                          command=self.show_order_history)
    user_menu.add_separator()
    user_menu.add_command(label="  ", command=self.logout)

    # Food menu
    food_menu = tk.Menu(menubar, tearoff=0)
    menubar.add_cascade(label="      ", menu=food_menu)
    food_menu.add_command(label="          ",
                          command=self.show_today_menu)
    food_menu.add_command(label="      ",
                          command=self.show_search_food)

    # Points menu
    points_menu = tk.Menu(menubar, tearoff=0)
    menubar.add_cascade(label="    ", menu=points_menu)
    points_menu.add_command(label="      ",
                            command=self.show_loyalty_points)
    points_menu.add_command(label="          ",
                            command=self.convert_points)
```

فصل ۸

تست‌های واحد

۱.۸ ساختار تست‌ها

۱.۱.۸ کلاس‌های تست

```python
import unittest
from auth import AuthManager
from food_service import FoodService
from order_service import OrderService
from customer_service import CustomerService
from admin_service import AdminService

class TestAuthManager(unittest.TestCase):
    """              """

    def setUp(self):
        """Setup before each test"""
        self._cleanup_test_files()
        self.auth = AuthManager()

    def tearDown(self):
        """Cleanup after each test"""
        self._cleanup_test_files()

    def _cleanup_test_files(self):
        """Remove test CSV files"""
        files = ['users.csv', 'foods.csv', 'orders.csv',
                'order_items.csv', 'reviews.csv', 'discount_codes.csv']
        for f in files:
            if os.path.exists(f):
                os.remove(f)

    def test_register_customer_success(self):
        """            """
        success, msg = self.auth.register_customer(
            first_name="  ",
            last_name="   ",
            email="ali@example.com",
            phone="09123456789",
            national_code="1234567890",
            password="Test@1234",
            confirm_password="Test@1234"
```

```
        )
        self.assertTrue(success)
        self.assertIn("successfully", msg.lower())

    def test_register_duplicate_email(self):
        """                    """
        self.auth.register_customer(
            "  ", "   ", "test@example.com", "09123456789",
            "1234567890", "Test@1234", "Test@1234"
        )
        success, msg = self.auth.register_customer(
            "  ", "   ", "test@example.com", "09987654321",
            "0987654321", "Test@1234", "Test@1234"
        )
        self.assertFalse(success)
        self.assertIn("email already exists", msg.lower())
```

۲.۱.۸ تست‌های Edge Case

```
def test_add_to_cart_zero_stock(self):
    """                      """
    cart = Cart()
    with self.assertRaises(ValueError) as context:
        self.food_service.add_to_cart(cart, "test-food-zero", 1)
    self.assertIn("stock", str(context.exception).lower())

def test_checkout_empty_cart(self):
    """                    """
    cart = Cart()
    with self.assertRaises(ValueError) as context:
        self.order_service.checkout(
            cart=cart,
            customer_id=self.customer.user_id,
            delivery_date=date.today() + timedelta(days=1),
            payment_method=Order.PAYMENT_ONLINE
        )
    self.assertIn("empty", str(context.exception).lower())

def test_account_lock_after_three_attempts(self):
    """                    3        """
    self.auth.register_customer(
        "  ", "   ", "ali@example.com", "09123456789",
        "1234567890", "Test@1234", "Test@1234"
    )

    # Three failed attempts
    self.auth.login_user("ali@example.com", "wrong1")
    self.auth.login_user("ali@example.com", "wrong2")
    self.auth.login_user("ali@example.com", "wrong3")

    # Account should be locked
    success, msg, user = self.auth.login_user(
        "ali@example.com", "Test@1234"
    )
    self.assertFalse(success)
    self.assertIn("locked", msg.lower())
```

```
    self.assertIsNone(user)
```

۲.۸ پوشش تست

۱.۲.۸ آمار تست‌ها

ماژول	تعداد تست	پوشش
احراز هویت	۸ تست	٪۹۵
مدیریت غذا	۶ تست	٪۹۰
سفارش‌ها	۴ تست	٪۸۸
مشتری	۳ تست	٪۸۵
ادمین	۳ تست	٪۸۰
مدل‌ها	۳ تست	٪۹۲
جمع	۲۷ تست	٪۸۸

جدول ۱.۸: پوشش تست‌های واحد

فصل ۹

نتیجه‌گیری و جمع‌بندی

۱.۹ دستاوردهای پروژه

۱.۱.۹ ویژگی‌های پیاده‌سازی شده

۱. طراحی کامل شیءگرا با رعایت اصول

۲. سیستم احراز هویت امن با اعتبارسنجی جامع

۳. مدیریت کامل چرخه سفارش از انتخاب تا تحویل

۴. سیستم گزارش‌گیری و تحلیل فروش

۵. رابط کاربری گرافیکی کاربرپسند

۶. پایگاه داده پایدار با مدیریت خطا

۷. تست‌های واحد جامع

۲.۱.۹ چالش‌های حل شده

- مدیریت همزمانی در به‌روزرسانی موجودی
- اعتبارسنجی تاریخ‌های سفارش
- پیاده‌سازی تراکنش‌های اتمی در فایل‌های CSV
- طراحی رابط کاربری فارسی‌زبان
- مدیریت استثناها و خطاها

۲.۹ نتیجه‌گیری نهایی

پروژه سامانه سفارش غذا به عنوان پروژه پایانی درس برنامه‌نویسی پیشرفته، موفق به پیاده‌سازی کلیه مفاهیم و اصول تدریس شده در درس گردیده است. سیستم طراحی شده از معماری مناسبی برخوردار بوده و تمامی الزامات فنی و کاربردی را برآورده می‌سازد.

این پروژه نه تنها به عنوان یک سیستم عملیاتی قابل استفاده است، بلکه به عنوان نمونه‌ای از طراحی و پیاده‌سازی حرفه‌ای نرم‌افزار می‌تواند مورد استفاده آموزشی قرار گیرد.

فصل ۱۰

ضمایم

۱.۱۰ کدهای تکمیلی

۱.۱.۱۰ کلاس Cart

```python
class Cart:
    def __init__(self):
        self.items: List[OrderItem] = []

    def add_item(self, food: Food, quantity: int):
        """Add item to cart with quantity validation"""
        if food.is_available(quantity):
            # Update quantity if item already in cart
            for item in self.items:
                if item.food.food_id == food.food_id:
                    if food.is_available(item.quantity + quantity):
                        item.quantity += quantity
                        return
                    else:
                        raise ValueError("Insufficient stock")

            # Add new item
            self.items.append(OrderItem(food, quantity))
        else:
            raise ValueError("Insufficient stock")

    def remove_item(self, food_id: str):
        """Remove item from cart"""
        self.items = [item for item in self.items
                      if item.food.food_id != food_id]

    def clear(self):
        """Clear all items from cart"""
        self.items = []

    def get_total(self) -> float:
        """Calculate total price"""
        return sum(item.total_price for item in self.items)
```

۲.۱.۱۰ کلاس DiscountCode

```python
@dataclass
class DiscountCode:
    """Discount code for customers"""
    code: str
    discount_percentage: float
    expiry_date: datetime
    is_used: bool = False
    customer_id: Optional[str] = None

    def is_valid(self) -> bool:
        """Check if discount code is still valid"""
        return not self.is_used and datetime.now() < self.expiry_date
```