# ПРИЛОЖЕНИЕ В. ТЕКСТ ПРОГРАММЫ

АННОТАЦИЯ

В данном программном документе приведён текст программы для мобильного приложения «Upavlika».

В разделе «Текст программы» указано наименование приложения, область применения приложения, модули приложения в виде таблицы с указанием описания и размера каждого модуля, код программы.

1. ТЕКСТ ПРОГРАММЫ

1.1. Наименование программы

Полное наименование: «Программа приложения для сайта кондитерской «Upavlika».

1.2. Область применения программы

Программа «Upavlika» предназначена для личного использования и покупки/ другого взаимодействия с товарами и заказами товаров для пользователей.

1.3. Модули

Таблица 7 – Модули

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название модуля | Описание модуля | Размер модуля | Кол-во строк |
| Java файлы |  |  |  |  |
| 1 | RegisterActivity.java | Файл регистрации пользователя | 9.9 Кб | 229 |
| 2 | LoginActivity.java | Файл авторизации пользователя | 9.2 Кб | 205 |
| 3 | ProfileActivity.java | Файл профиля пользователя | 11 Кб | 304 |
| 4 | ProductDetailActivity.java | Файл деталей продукта | 5.2 Кб | 165 |
| 5 | ProductAdapter.java | Адаптер списка продуктов  Модель продукта | 3.1 Кб | 97 |
| 6 | Product.java | 6.4 Кб | 160 |
| 7 | OrderHistoryActivity.java | Файл истории заказов | 27 Кб | 748 |
| 8 | OrderConfirmationActivity.java | Файл подтверждения заказа | 5.9 Кб | 169 |
| 9 | MainActivity.java | Главная активность | 15 Кб | 355 |
| 10 | DeliveryTrackingActivity.java | Файл отслеживания доставки | 22 Кб | 578 |
| 11 | CartActivity.java | Файл корзины покупок | 35 Кб | 898 |
| 12 | AdminPanelActivity.java | Панель администратора | 2.8 Кб | 94 |
| 13 | ManageUsersFragment.java | Фрагмент управления пользователями | 10 Кб | 244 |
| 14 | ManageProductsFragment.java | Фрагмент управления продуктами | 18 Кб | 405 |
| 15 | DatabaseHelper.java | Помощник базы данных | 57 Кб | 1244 |
| 16 | ProductAdapter.java | Адаптер продуктов | 4.9 Кб | 151 |
| 17 | OrderItemAdapter.java | Адаптер элементов заказа | 2.8 Кб | 83 |
| 18 | OrderAdapter.java | Адаптер заказов | 8.0 Кб | 209 |
| 19 | CartAdapter.java | Адаптер корзины | 9.4 Кб | 231 |
| 20 | DeliveryTrackerService.java | Сервис отслеживания доставки | 20 Кб | 519 |
| 21 | DeliveryNotificationReceiver.java | Приемник уведомлений о доставке | 4.2 Кб | 106 |
| 22 | BootReceiver.java | Приемник загрузки системы | 5.2 Кб | 119 |
| 23 | UserManager.java | Менеджер пользователей | 5.5 Кб | 178 |
| 24 | OrderFileManager.java | Менеджер файлов заказов | 16 Кб | 443 |
| 25 | User.java | Модель пользователя | 2.6 Кб | 116 |
| 26 | UserRole.java | Модель роли пользователя | 0.4 Кб | 18 |
| 27 | UserProfile.java | Модель профиля пользователя | 2.1 Кб | 96 |
| 28 | Product.java | Модель продукта | 2.7 Кб | 121 |
| 29 | OrderStatus.java | Модель статуса заказа | 0.6 Кб | 29 |
| 30 | OrderItem.java | Модель элемента заказа | 1.8 Кб | 98 |
| 31 | Order.java | Модель заказа | 5.0 Кб | 136 |
| 32 | CartItem.java | Модель элемента корзины | 0.9 Кб | 47 |
| 33 | Cart.java | Модель корзины | 10 Кб | 303 |
| XML файлы (layout) |  |  |  |  |
| 34 | activity\_product\_detail.xml | Макет деталей продукта | 10 Кб | 212 |
| 35 | item\_order\_item.xml | Макет элемента заказа | 2.5 Кб | 64 |
| 36 | item\_order.xml | Макет заказа | 1.9 Кб | 56 |
| 37 | activity\_product\_details.xml | Макет деталей продукта | 12 Кб | 252 |
| 38 | dialog\_forgot\_password.xml | Диалог восстановления пароля | 1.7 Кб | 46 |
| 39 | activity\_login.xml | Макет авторизации | 4.5 Кб | 108 |
| 40 | dialog\_reset\_password.xml | Диалог сброса пароля | 2.3 Кб | 58 |
| 41 | item\_product.xml | Макет продукта | 3.4 Кб | 80 |
| 42 | item\_cart\_product.xml | Макет продукта в корзине | 4.8 Кб | 117 |
| 43 | fragment\_manage\_users.xml | Макет управления пользователями | 2.1 Кб | 50 |
| 44 | | | | |
| 45 | dialog\_save\_order.xml | Диалог сохранения заказа | 1.5 Кб | 43 |
| 46 | dialog\_edit\_product.xml | Диалог редактирования продукта | 4.9 Кб | 117 |
| 47 | dialog\_delivery\_rating.xml | Диалог оценки доставки | 1.2 Кб | 34 |
| 48 | dialog\_change\_role.xml | Диалог смены роли | 1.7 Кб | 49 |
| 49 | dialog\_add\_user.xml | Диалог добавления пользователя | 3.0 Кб | 80 |
| 50 | activity\_register.xml | Макет регистрации | 7.4 Кб | 179 |
| 51 | activity\_profile.xml | Макет профиля | 11 Кб | 232 |
| 52 | activity\_order\_history.xml | Макет истории заказов | 1.6 Кб | 39 |
| 53 | activity\_order\_confirmation.xml | Макет подтверждения заказа | 6.1 Кб | 149 |
| 54 | activity\_main.xml | Главный макет | 3.1 Кб | 79 |
| 55 | activity\_delivery\_tracking.xml | Макет отслеживания доставки | 7.1 Кб | 173 |
| 56 | activity\_cart.xml | Макет корзины | 3.5 Кб | 87 |
| 57 | activity\_admin\_panel.xml | Макет панели администратора | 1.4 Кб | 39 |
| XML файлы (values) |  |  |  |  |
| 58 | strings.xml | Строковые ресурсы | 3.4 Кб | 62 |
| 59 | themes.xml | Тема приложения | 0.3 Кб | 10 |
| 60 | colors.xml | Цветовые ресурсы | 0.5 Кб | 16 |
| XML файлы (menu) |  |  |  |  |
| 61 | main\_menu.xml | Главное меню | 1.2 Кб | 39 |
| XML файлы (drawable) |  |  |  |  |
| 62-84 | Различные иконки и фоны | Иконки и фоновые элементы | ~15 Кб | ~400 |

1.4. Код программы

package com.example.upavlika.adapter;  
import android.content.Context;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.ImageButton;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.upavlika.R;  
import com.example.upavlika.model.CartItem;  
import com.example.upavlika.model.Product;  
  
import java.math.BigDecimal;  
import java.text.NumberFormat;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Locale;  
  
public class CartAdapter extends RecyclerView.Adapter<CartAdapter.CartViewHolder> {  
   
 private Context context;  
 private List<CartItem> cartItems;  
 private final CartItemListener listener;  
   
 public interface CartItemListener {  
 void onQuantityChanged(long productId, int newQuantity);  
 void onItemRemoved(long productId);  
 }  
   
 public CartAdapter(Context context, List<CartItem> cartItems, CartItemListener listener) {  
 this.context = context;  
 this.cartItems = new ArrayList<>(cartItems);  
 this.listener = listener;  
 }  
   
 public void updateCartItems(List<CartItem> newItems) {  
 try {  
 this.cartItems = new ArrayList<>(newItems != null ? newItems : new ArrayList<>());  
 notifyDataSetChanged();  
 } catch (Exception e) {  
 Log.e("CartAdapter", "Ошибка при обновлении элементов корзины: " + e.getMessage());  
 this.cartItems = new ArrayList<>();  
 notifyDataSetChanged();  
 }  
 }  
   
 @NonNull  
 @Override  
 public CartViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 try {  
 View view = LayoutInflater.from(context)  
 .inflate(R.layout.item\_cart\_product, parent, false);  
 return new CartViewHolder(view);  
 } catch (Exception e) {  
 Log.e("CartAdapter", "Ошибка при создании ViewHolder: " + e.getMessage());  
   
 View fallbackView = new View(context);  
 fallbackView.setLayoutParams(new ViewGroup.LayoutParams(  
 ViewGroup.LayoutParams.MATCH\_PARENT,  
 ViewGroup.LayoutParams.WRAP\_CONTENT));  
 return new CartViewHolder(fallbackView);  
 }  
 }  
   
 @Override  
 public void onBindViewHolder(@NonNull CartViewHolder holder, int position) {  
 try {  
 if (cartItems == null || position < 0 || position >= cartItems.size()) {  
   
 Log.e("CartAdapter", "Некорректная позиция: " + position);  
 holder.cartProductNameTextView.setText("Ошибка: неверная позиция");  
 holder.cartItemPriceTextView.setText("0 ₽");  
 holder.cartQuantityTextView.setText("0");  
 holder.cartItemTotalTextView.setText("0 ₽");  
 holder.cartDecreaseButton.setEnabled(false);  
 holder.cartIncreaseButton.setEnabled(false);  
 return;  
 }  
  
 CartItem item = cartItems.get(position);  
 if (item == null || item.getProduct() == null) {  
   
 holder.cartProductNameTextView.setText("Ошибка: товар не найден");  
 holder.cartItemPriceTextView.setText("0 ₽");  
 holder.cartQuantityTextView.setText("0");  
 holder.cartItemTotalTextView.setText("0 ₽");  
 holder.cartDecreaseButton.setEnabled(false);  
 holder.cartIncreaseButton.setEnabled(false);  
 return;  
 }  
   
   
 Product product = item.getProduct();  
   
   
 try {  
 holder.cartProductImageView.setImageResource(product.getImageResourceId());  
 } catch (Exception e) {  
   
 holder.cartProductImageView.setImageResource(R.drawable.placeholder);  
 }  
   
   
 holder.cartProductNameTextView.setText(product.getName() != null ? product.getName() : "Неизвестный товар");  
   
   
 try {  
 double price = product.getPrice() != null ? product.getPrice().doubleValue() : 0.0;  
 holder.cartItemPriceTextView.setText(String.format("%s ₽", formatPrice(price)));  
 } catch (Exception e) {  
 holder.cartItemPriceTextView.setText("0 ₽");  
 }  
   
   
 holder.cartQuantityTextView.setText(String.valueOf(item.getQuantity()));  
   
   
 try {  
 holder.cartItemTotalTextView.setText(  
 String.format("x%d: %s ₽", item.getQuantity(), formatPrice(item.getTotal())));  
 } catch (Exception e) {  
 holder.cartItemTotalTextView.setText("x0: 0 ₽");  
 }  
   
   
 holder.cartDecreaseButton.setOnClickListener(v -> {  
 try {  
 int currentQuantity = item.getQuantity();  
 if (currentQuantity > 1) {  
 listener.onQuantityChanged(item.getProduct().getId(), currentQuantity - 1);  
 }  
 } catch (Exception e) {  
   
 Log.e("CartAdapter", "Ошибка уменьшения количества: " + e.getMessage());  
 }  
 });  
   
 holder.cartIncreaseButton.setOnClickListener(v -> {  
 try {  
 int currentQuantity = item.getQuantity();  
   
 if (currentQuantity < 10) {  
 listener.onQuantityChanged(item.getProduct().getId(), currentQuantity + 1);  
 }  
 } catch (Exception e) {  
   
 Log.e("CartAdapter", "Ошибка увеличения количества: " + e.getMessage());  
 }  
 });  
   
   
 holder.removeCartItemButton.setOnClickListener(v -> {  
 try {  
 listener.onItemRemoved(item.getProduct().getId());  
 } catch (Exception e) {  
   
 Log.e("CartAdapter", "Ошибка при удалении: " + e.getMessage());  
 try {  
 CartItem posItem = cartItems.get(holder.getAdapterPosition());  
 if (posItem != null && posItem.getProduct() != null) {  
 listener.onItemRemoved(posItem.getProduct().getId());  
 }  
 } catch (Exception ex) {  
   
 Log.e("CartAdapter", "Невозможно удалить товар: " + ex.getMessage());  
 Toast.makeText(context, "Не удалось удалить товар", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
 } catch (Exception e) {  
   
 Log.e("CartAdapter", "Ошибка при привязке данных: " + e.getMessage(), e);  
 if (holder != null) {  
 try {  
 holder.cartProductNameTextView.setText("Ошибка при загрузке товара");  
 holder.cartItemPriceTextView.setText("");  
 holder.cartQuantityTextView.setText("");  
 holder.cartItemTotalTextView.setText("");  
 holder.cartDecreaseButton.setEnabled(false);  
 holder.cartIncreaseButton.setEnabled(false);  
 } catch (Exception ex) {  
   
 }  
 }  
 }  
 }  
   
 @Override  
 public int getItemCount() {  
 return cartItems != null ? cartItems.size() : 0;  
 }  
   
   
 private String formatPrice(double price) {  
 return String.format("%.0f", price);  
 }  
   
   
 static class CartViewHolder extends RecyclerView.ViewHolder {  
 ImageView cartProductImageView;  
 TextView cartProductNameTextView;  
 TextView cartItemPriceTextView;  
 TextView cartQuantityTextView;  
 TextView cartItemTotalTextView;  
 ImageButton cartDecreaseButton;  
 ImageButton cartIncreaseButton;  
 ImageButton removeCartItemButton;  
   
 public CartViewHolder(@NonNull View itemView) {  
 super(itemView);  
 cartProductImageView = itemView.findViewById(R.id.cartProductImageView);  
 cartProductNameTextView = itemView.findViewById(R.id.cartProductNameTextView);  
 cartItemPriceTextView = itemView.findViewById(R.id.cartItemPriceTextView);  
 cartQuantityTextView = itemView.findViewById(R.id.cartQuantityTextView);  
 cartItemTotalTextView = itemView.findViewById(R.id.cartItemTotalTextView);  
 cartDecreaseButton = itemView.findViewById(R.id.cartDecreaseButton);  
 cartIncreaseButton = itemView.findViewById(R.id.cartIncreaseButton);  
 removeCartItemButton = itemView.findViewById(R.id.removeCartItemButton);  
 }  
 }  
}

package com.example.upavlika.adapter;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.upavlika.R;  
import com.example.upavlika.data.DatabaseHelper;  
import com.example.upavlika.model.Order;  
import com.example.upavlika.model.OrderItem;  
import com.example.upavlika.model.User;  
import com.example.upavlika.model.UserRole;  
import com.example.upavlika.util.UserManager;  
  
import java.text.SimpleDateFormat;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Locale;  
import java.util.Map;  
  
public class OrderAdapter extends RecyclerView.Adapter<OrderAdapter.OrderViewHolder> {  
 private List<Order> orders;  
 private OnOrderClickListener listener;  
 private Map<Long, Integer> userOrderNumbers;   
 private boolean isAdmin = false;  
 private Context context;  
 private Map<Long, String> userNamesCache = new HashMap<>();   
  
 public interface OnOrderClickListener {  
 void onOrderClick(Order order);  
 }  
  
 public OrderAdapter(List<Order> orders, OnOrderClickListener listener) {  
 this.orders = orders;  
 this.listener = listener;  
 this.userOrderNumbers = new HashMap<>();  
 }  
  
 @NonNull  
 @Override  
 public OrderViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 context = parent.getContext();  
 View view = LayoutInflater.from(context)  
 .inflate(R.layout.item\_order, parent, false);  
 return new OrderViewHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull OrderViewHolder holder, int position) {  
 Order order = orders.get(position);  
 holder.bind(order);  
 }  
  
 @Override  
 public int getItemCount() {  
 return orders.size();  
 }  
  
 public void updateOrders(List<Order> newOrders) {  
 this.orders = newOrders;  
   
   
 userOrderNumbers.clear();  
   
   
 for (Order order : newOrders) {  
 long userId = order.getUserId();  
 if (!userOrderNumbers.containsKey(userId)) {  
 userOrderNumbers.put(userId, 1);  
 } else {  
 userOrderNumbers.put(userId, userOrderNumbers.get(userId) + 1);  
 }  
 order.setOrderNumber("" + userOrderNumbers.get(userId));  
 }  
   
 notifyDataSetChanged();  
 }  
   
   
 public void setAdmin(boolean isAdmin) {  
 this.isAdmin = isAdmin;  
 }  
  
   
 private String getUserNameById(long userId) {  
   
 if (userNamesCache.containsKey(userId)) {  
 return userNamesCache.get(userId);  
 }  
  
   
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(context);  
 User user = dbHelper.getUserById(userId);  
 String userName = (user != null) ? user.getName() : "Неизвестный пользователь";  
   
   
 userNamesCache.put(userId, userName);  
   
 return userName;  
 }  
  
 class OrderViewHolder extends RecyclerView.ViewHolder {  
 private TextView textViewOrderId;  
 private TextView textViewOrderDate;  
 private TextView textViewOrderStatus;  
 private TextView textViewOrderTotal;  
 private TextView textViewOrderItems;  
  
 public OrderViewHolder(@NonNull View itemView) {  
 super(itemView);  
 textViewOrderId = itemView.findViewById(R.id.textViewOrderId);  
 textViewOrderDate = itemView.findViewById(R.id.textViewOrderDate);  
 textViewOrderStatus = itemView.findViewById(R.id.textViewOrderStatus);  
 textViewOrderTotal = itemView.findViewById(R.id.textViewOrderTotal);  
 textViewOrderItems = itemView.findViewById(R.id.textViewOrderItems);  
  
 itemView.setOnClickListener(v -> {  
 int position = getAdapterPosition();  
 if (position != RecyclerView.NO\_POSITION) {  
 listener.onOrderClick(orders.get(position));  
 }  
 });  
 }  
  
 public void bind(Order order) {  
 try {  
   
 if (isAdmin) {  
 String userName = getUserNameById(order.getUserId());  
 textViewOrderId.setText("Заказ #" + order.getOrderNumber() + " (ID: " + order.getId() + ", Клиент: " + userName + ")");  
 } else {  
 textViewOrderId.setText("Заказ #" + order.getOrderNumber());  
 }  
   
   
 String dateText;  
 try {  
 dateText = "Дата: " + order.getFormattedDate();  
 } catch (Exception e) {  
 dateText = "Дата: не указана";  
 }  
 textViewOrderDate.setText(dateText);  
   
   
 String statusDisplay;  
 try {  
 if (order.getOrderStatus() != null) {  
 statusDisplay = order.getOrderStatus().getDisplayName();  
 } else if (order.getStatus() != null && !order.getStatus().isEmpty()) {  
 statusDisplay = order.getStatus();  
 } else {  
 statusDisplay = "Неизвестный";  
 }  
 } catch (Exception e) {  
 statusDisplay = order.getStatus() != null ? order.getStatus() : "Неизвестный";  
 }  
 textViewOrderStatus.setText("Статус: " + statusDisplay);  
   
   
 String totalText;  
 try {  
 totalText = String.format("Итого: %.2f ₽", order.getTotalAmount());  
 } catch (Exception e) {  
 totalText = "Итого: не указано";  
 }  
 textViewOrderTotal.setText(totalText);  
   
   
 StringBuilder itemsText = new StringBuilder();  
 try {  
 List<OrderItem> items = order.getOrderItems();  
 if (items != null && !items.isEmpty()) {  
 for (OrderItem item : items) {  
 if (item != null) {  
 try {  
 String productName = item.getProductName() != null ? item.getProductName() : "Неизвестный товар";  
 int quantity = item.getQuantity();  
 double price = item.getPrice();  
 itemsText.append(String.format("%s x%d - %.2f ₽\n", productName, quantity, price));  
 } catch (Exception ex) {  
 itemsText.append("Ошибка при отображении товара\n");  
 }  
 }  
 }  
 } else {  
 itemsText.append("Информация о товарах недоступна");  
 }  
 } catch (Exception e) {  
 itemsText.append("Ошибка при загрузке товаров");  
 }  
 textViewOrderItems.setText(itemsText.toString());  
 } catch (Exception e) {  
   
 textViewOrderId.setText("Заказ");  
 textViewOrderDate.setText("Дата: не указана");  
 textViewOrderStatus.setText("Статус: ошибка");  
 textViewOrderTotal.setText("Итого: ошибка");  
 textViewOrderItems.setText("Ошибка при загрузке информации о заказе");  
 }  
 }  
 }  
}

package com.example.upavlika.adapter;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.upavlika.R;  
import com.example.upavlika.model.OrderItem;  
  
import java.text.NumberFormat;  
import java.util.List;  
import java.util.Locale;  
  
  
public class OrderItemAdapter extends RecyclerView.Adapter<OrderItemAdapter.OrderItemViewHolder> {  
  
 private final Context context;  
 private final List<OrderItem> orderItems;  
 private final NumberFormat currencyFormat;  
  
 public OrderItemAdapter(Context context, List<OrderItem> orderItems) {  
 this.context = context;  
 this.orderItems = orderItems;  
 this.currencyFormat = NumberFormat.getCurrencyInstance(new Locale("ru", "RU"));  
 }  
  
 @NonNull  
 @Override  
 public OrderItemViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.from(context).inflate(R.layout.item\_order\_item, parent, false);  
 return new OrderItemViewHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull OrderItemViewHolder holder, int position) {  
 OrderItem item = orderItems.get(position);  
 holder.bind(item);  
 }  
  
 @Override  
 public int getItemCount() {  
 return orderItems != null ? orderItems.size() : 0;  
 }  
  
 class OrderItemViewHolder extends RecyclerView.ViewHolder {  
 private final TextView productNameTextView;  
 private final TextView quantityTextView;  
 private final TextView priceTextView;  
 private final TextView totalTextView;  
  
 public OrderItemViewHolder(@NonNull View itemView) {  
 super(itemView);  
 productNameTextView = itemView.findViewById(R.id.product\_name\_text\_view);  
 quantityTextView = itemView.findViewById(R.id.quantity\_text\_view);  
 priceTextView = itemView.findViewById(R.id.price\_text\_view);  
 totalTextView = itemView.findViewById(R.id.total\_text\_view);  
 }  
  
 public void bind(OrderItem item) {  
 if (item != null) {  
   
 productNameTextView.setText(item.getProductName());  
   
   
 quantityTextView.setText(String.format(Locale.getDefault(), "x%d", item.getQuantity()));  
   
   
 priceTextView.setText(currencyFormat.format(item.getPrice()));  
   
   
 double totalPrice = item.getPrice() \* item.getQuantity();  
 totalTextView.setText(currencyFormat.format(totalPrice));  
 }  
 }  
 }  
}

package com.example.upavlika.adapter;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.upavlika.R;  
import com.example.upavlika.model.Product;  
  
import java.text.NumberFormat;  
import java.util.List;  
import java.util.Locale;  
  
public class ProductAdapter extends RecyclerView.Adapter<ProductAdapter.ProductViewHolder> {  
  
 private List<Product> productList;  
 private Context context;  
   
   
 public interface OnProductClickListener {  
 void onProductClick(Product product);  
 void onAddToCartClick(Product product);  
 }  
   
 private OnProductClickListener listener;  
  
 public ProductAdapter(Context context, List<Product> productList) {  
 this.context = context;  
 this.productList = productList;  
 }  
   
 public void setOnProductClickListener(OnProductClickListener listener) {  
 this.listener = listener;  
 }  
  
 @NonNull  
 @Override  
 public ProductViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.from(context).inflate(R.layout.item\_product, parent, false);  
 return new ProductViewHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull ProductViewHolder holder, int position) {  
 Product product = productList.get(position);  
   
   
 holder.productNameTextView.setText(product.getName());  
 holder.productDescriptionTextView.setText(product.getDescription());  
 holder.productPriceTextView.setText(formatPrice(product.getPrice().doubleValue()));  
   
   
 int imageResource = getImageResourceByCategory(product.getCategory());  
 if (imageResource != 0) {  
 holder.productImageView.setImageResource(imageResource);  
 } else {  
   
 holder.productImageView.setImageResource(product.getImageResourceId());  
 }  
   
   
 holder.itemView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (listener != null) {  
 listener.onProductClick(product);  
 }  
 }  
 });  
   
 holder.addToCartButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (listener != null) {  
 listener.onAddToCartClick(product);  
 }  
 }  
 });  
 }  
  
 @Override  
 public int getItemCount() {  
 return productList.size();  
 }  
   
   
 public void updateProducts(List<Product> newProducts) {  
 this.productList = newProducts;  
 notifyDataSetChanged();  
 }  
  
 class ProductViewHolder extends RecyclerView.ViewHolder {  
 private ImageView productImageView;  
 private TextView productNameTextView;  
 private TextView productPriceTextView;  
 private TextView productDescriptionTextView;  
 private Button addToCartButton;  
  
 public ProductViewHolder(@NonNull View itemView) {  
 super(itemView);  
   
 productImageView = itemView.findViewById(R.id.productImageView);  
 productNameTextView = itemView.findViewById(R.id.productNameTextView);  
 productPriceTextView = itemView.findViewById(R.id.productPriceTextView);  
 productDescriptionTextView = itemView.findViewById(R.id.productDescriptionTextView);  
 addToCartButton = itemView.findViewById(R.id.addToCartButton);  
 }  
 }  
  
 private String formatPrice(double price) {  
 NumberFormat formatter = NumberFormat.getCurrencyInstance(new Locale("ru", "RU"));  
 return formatter.format(price);  
 }  
   
   
 private int getImageResourceByCategory(String category) {  
 if (category == null) {  
 return 0;   
 }  
   
 String lowerCategory = category.toLowerCase();  
   
 if (lowerCategory.contains("торт")) {  
 return R.drawable.ic\_cake;  
 } else if (lowerCategory.contains("пирожн")) {  
 return R.drawable.ic\_pastry;  
 } else if (lowerCategory.contains("печен")) {  
 return R.drawable.ic\_cookie;  
 } else if (lowerCategory.contains("хлеб") || lowerCategory.contains("выпеч")) {  
 return R.drawable.ic\_bread;  
 } else if (lowerCategory.contains("пончик") || lowerCategory.contains("донат") || lowerCategory.contains("donut")) {  
 return R.drawable.ic\_donut;  
 } else if (lowerCategory.contains("кекс") || lowerCategory.contains("капкейк") || lowerCategory.contains("cupcake")) {  
 return R.drawable.ic\_cupcake;  
 }  
   
   
   
 return 0;  
 }  
}

package com.example.upavlika.data;  
  
import android.content.ContentValues;  
import android.content.Context;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;  
import android.util.Log;  
  
import com.example.upavlika.R;  
import com.example.upavlika.model.Product;  
import com.example.upavlika.model.User;  
import com.example.upavlika.model.UserRole;  
import com.example.upavlika.model.Order;  
import com.example.upavlika.model.OrderItem;  
import com.example.upavlika.model.UserProfile;  
  
import java.math.BigDecimal;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
import java.text.SimpleDateFormat;  
import java.text.ParseException;  
  
public class DatabaseHelper extends SQLiteOpenHelper {  
   
 public static final String DATABASE\_NAME = "upavlika.db";  
 private static final int DATABASE\_VERSION = 2;  
   
 public static final String TABLE\_USERS = "users";  
 public static final String COLUMN\_USER\_ID = "id";  
 public static final String COLUMN\_USER\_NAME = "name";  
 public static final String COLUMN\_USER\_EMAIL = "email";  
 public static final String COLUMN\_USER\_PASSWORD = "password";  
 public static final String COLUMN\_USER\_ROLE = "role";  
 public static final String COLUMN\_USER\_BALANCE = "balance";  
   
 public static final String TABLE\_PRODUCTS = "products";  
 public static final String COLUMN\_PRODUCT\_ID = "id";  
 public static final String COLUMN\_PRODUCT\_NAME = "name";  
 public static final String COLUMN\_PRODUCT\_DESCRIPTION = "description";  
 public static final String COLUMN\_PRODUCT\_PRICE = "price";  
 public static final String COLUMN\_PRODUCT\_STOCK = "stock\_quantity";  
 public static final String COLUMN\_PRODUCT\_IMAGE = "image\_url";  
 public static final String COLUMN\_PRODUCT\_CATEGORY = "category";  
   
 public static final String TABLE\_ORDERS = "orders";  
 public static final String COLUMN\_ORDER\_ID = "id";  
 public static final String COLUMN\_ORDER\_USER\_ID = "user\_id";  
 public static final String COLUMN\_ORDER\_TOTAL\_AMOUNT = "total\_amount";  
 public static final String COLUMN\_ORDER\_DATE = "order\_date";  
 public static final String COLUMN\_ORDER\_STATUS = "status";  
 public static final String COLUMN\_ORDER\_DELIVERY\_ADDRESS = "delivery\_address";  
 public static final String COLUMN\_ORDER\_CONTACT\_PHONE = "contact\_phone";  
 public static final String COLUMN\_ORDER\_PAID = "paid";  
 public static final String COLUMN\_ORDER\_CANCELLATION\_REASON = "cancellation\_reason";  
   
 public static final String TABLE\_ORDER\_ITEMS = "order\_items";  
 public static final String COLUMN\_ORDER\_ITEM\_ID = "id";  
 public static final String COLUMN\_ORDER\_ITEM\_ORDER\_ID = "order\_id";  
 public static final String COLUMN\_ORDER\_ITEM\_PRODUCT\_ID = "product\_id";  
 public static final String COLUMN\_ORDER\_ITEM\_QUANTITY = "quantity";  
 public static final String COLUMN\_ORDER\_ITEM\_PRICE = "price";  
 public static final String COLUMN\_ORDER\_ITEM\_PRODUCT\_NAME = "product\_name";  
   
 public static final String TABLE\_USER\_PROFILES = "user\_profiles";  
 public static final String COLUMN\_PROFILE\_USER\_ID = "user\_id";  
 public static final String COLUMN\_PROFILE\_PHONE = "phone";  
 public static final String COLUMN\_PROFILE\_ADDRESS = "address";  
 public static final String COLUMN\_PROFILE\_CITY = "city";  
 public static final String COLUMN\_PROFILE\_POSTAL\_CODE = "postal\_code";  
   
 private static DatabaseHelper instance;  
   
 private static final String SQL\_CREATE\_USERS\_TABLE =   
 "CREATE TABLE " + TABLE\_USERS + " (" +  
 COLUMN\_USER\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +  
 COLUMN\_USER\_NAME + " TEXT NOT NULL, " +  
 COLUMN\_USER\_EMAIL + " TEXT NOT NULL UNIQUE, " +  
 COLUMN\_USER\_PASSWORD + " TEXT NOT NULL, " +  
 COLUMN\_USER\_ROLE + " TEXT NOT NULL, " +  
 COLUMN\_USER\_BALANCE + " REAL NOT NULL DEFAULT 0.0)";  
   
 private static final String SQL\_CREATE\_PRODUCTS\_TABLE =   
 "CREATE TABLE " + TABLE\_PRODUCTS + " (" +  
 COLUMN\_PRODUCT\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +  
 COLUMN\_PRODUCT\_NAME + " TEXT NOT NULL, " +  
 COLUMN\_PRODUCT\_DESCRIPTION + " TEXT, " +  
 COLUMN\_PRODUCT\_PRICE + " REAL NOT NULL, " +  
 COLUMN\_PRODUCT\_STOCK + " INTEGER NOT NULL DEFAULT 0, " +  
 COLUMN\_PRODUCT\_IMAGE + " TEXT, " +  
 COLUMN\_PRODUCT\_CATEGORY + " TEXT)";  
   
 private static final String SQL\_CREATE\_ORDERS\_TABLE =   
 "CREATE TABLE " + TABLE\_ORDERS + " (" +  
 COLUMN\_ORDER\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +  
 COLUMN\_ORDER\_USER\_ID + " INTEGER NOT NULL, " +  
 COLUMN\_ORDER\_TOTAL\_AMOUNT + " REAL NOT NULL, " +  
 COLUMN\_ORDER\_DATE + " TEXT NOT NULL, " +  
 COLUMN\_ORDER\_STATUS + " TEXT NOT NULL, " +  
 COLUMN\_ORDER\_DELIVERY\_ADDRESS + " TEXT, " +  
 COLUMN\_ORDER\_CONTACT\_PHONE + " TEXT, " +  
 COLUMN\_ORDER\_PAID + " INTEGER DEFAULT 0, " +  
 COLUMN\_ORDER\_CANCELLATION\_REASON + " TEXT" +  
 ")";  
   
 private static final String SQL\_CREATE\_ORDER\_ITEMS\_TABLE =   
 "CREATE TABLE " + TABLE\_ORDER\_ITEMS + " (" +  
 COLUMN\_ORDER\_ITEM\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +  
 COLUMN\_ORDER\_ITEM\_ORDER\_ID + " INTEGER NOT NULL, " +  
 COLUMN\_ORDER\_ITEM\_PRODUCT\_ID + " INTEGER NOT NULL, " +  
 COLUMN\_ORDER\_ITEM\_QUANTITY + " INTEGER NOT NULL, " +  
 COLUMN\_ORDER\_ITEM\_PRICE + " REAL NOT NULL, " +  
 COLUMN\_ORDER\_ITEM\_PRODUCT\_NAME + " TEXT NOT NULL" +  
 ")";  
   
 public static synchronized DatabaseHelper getInstance(Context context) {  
 if (instance == null) {  
 instance = new DatabaseHelper(context.getApplicationContext());  
 }  
 return instance;  
 }  
   
 private DatabaseHelper(Context context) {  
 super(context, DATABASE\_NAME, null, DATABASE\_VERSION);  
 }  
   
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 db.execSQL("DROP TABLE IF EXISTS " + TABLE\_ORDER\_ITEMS);  
 db.execSQL("DROP TABLE IF EXISTS " + TABLE\_ORDERS);  
 db.execSQL("DROP TABLE IF EXISTS " + TABLE\_PRODUCTS);  
 db.execSQL("DROP TABLE IF EXISTS " + TABLE\_USERS);  
 db.execSQL("DROP TABLE IF EXISTS " + TABLE\_USER\_PROFILES);  
   
 db.execSQL(SQL\_CREATE\_USERS\_TABLE);  
 db.execSQL(SQL\_CREATE\_PRODUCTS\_TABLE);  
 db.execSQL(SQL\_CREATE\_ORDERS\_TABLE);  
 db.execSQL(SQL\_CREATE\_ORDER\_ITEMS\_TABLE);  
   
 db.execSQL("CREATE TABLE " + TABLE\_USER\_PROFILES + " (" +  
 COLUMN\_PROFILE\_USER\_ID + " INTEGER PRIMARY KEY, " +  
 COLUMN\_PROFILE\_PHONE + " TEXT, " +  
 COLUMN\_PROFILE\_ADDRESS + " TEXT, " +  
 COLUMN\_PROFILE\_CITY + " TEXT, " +  
 COLUMN\_PROFILE\_POSTAL\_CODE + " TEXT" +  
 ")");  
   
 insertInitialUsers(db);  
 insertInitialProducts(db);  
 }  
   
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 if (oldVersion < 2) {  
 try {  
 addProductIfNotExists(db, "Эклер с заварным кремом", "Классическое французское пирожное с нежным заварным кремом и шоколадной глазурью", 179.99, 25, "product\_pastry\_eclair", "Пирожные");  
 addProductIfNotExists(db, "Тирамису", "Итальянское пирожное с кофейной пропиткой, сыром маскарпоне и какао", 259.99, 15, "product\_pastry\_tiramisu", "Пирожные");  
 addProductIfNotExists(db, "Наполеон", "Традиционное многослойное пирожное с заварным кремом и хрустящими коржами", 229.99, 20, "product\_pastry\_napoleon", "Пирожные");  
   
 addProductIfNotExists(db, "Батон белый", "Классический белый хлеб из пшеничной муки высшего сорта", 89.99, 30, "product\_bread\_white", "Хлеб");  
 addProductIfNotExists(db, "Хлеб ржаной", "Традиционный ржаной хлеб с насыщенным вкусом и ароматом", 99.99, 25, "product\_bread\_rye", "Хлеб");  
 addProductIfNotExists(db, "Багет французский", "Хрустящий французский багет с мягким мякишем и хрустящей корочкой", 129.99, 15, "product\_bread\_baguette", "Хлеб");  
 } catch (Exception e) {  
 Log.e("DatabaseHelper", "Ошибка при добавлении новых товаров: " + e.getMessage());  
 }  
 }  
 }  
   
 public void addProductIfNotExists(SQLiteDatabase db, String name, String description,   
 double price, int stock, String imageUrl, String category) {  
 Cursor cursor = db.query(TABLE\_PRODUCTS, new String[]{COLUMN\_PRODUCT\_ID},  
 COLUMN\_PRODUCT\_NAME + "=? AND " + COLUMN\_PRODUCT\_CATEGORY + "=?",  
 new String[]{name, category}, null, null, null);  
   
 boolean exists = cursor != null && cursor.moveToFirst();  
 if (cursor != null) {  
 cursor.close();  
 }  
   
 if (!exists) {  
 ContentValues values = new ContentValues();  
 values.put(COLUMN\_PRODUCT\_NAME, name);  
 values.put(COLUMN\_PRODUCT\_DESCRIPTION, description);  
 values.put(COLUMN\_PRODUCT\_PRICE, price);  
 values.put(COLUMN\_PRODUCT\_STOCK, stock);  
 values.put(COLUMN\_PRODUCT\_IMAGE, imageUrl);  
 values.put(COLUMN\_PRODUCT\_CATEGORY, category);  
 db.insert(TABLE\_PRODUCTS, null, values);  
 Log.d("DatabaseHelper", "Добавлен новый товар: " + name + " (" + category + ")");  
 }  
 }  
   
 private void insertInitialUsers(SQLiteDatabase db) {  
 ContentValues adminValues = new ContentValues();  
 adminValues.put(COLUMN\_USER\_NAME, "Администратор");  
 adminValues.put(COLUMN\_USER\_EMAIL, "admin@example.com");  
 adminValues.put(COLUMN\_USER\_PASSWORD, "admin123");  
 adminValues.put(COLUMN\_USER\_ROLE, UserRole.АДМИН.name());  
 adminValues.put(COLUMN\_USER\_BALANCE, 10000.0);  
 db.insert(TABLE\_USERS, null, adminValues);  
   
 ContentValues customer1Values = new ContentValues();  
 customer1Values.put(COLUMN\_USER\_NAME, "Иван Покупатель");  
 customer1Values.put(COLUMN\_USER\_EMAIL, "customer@example.com");  
 customer1Values.put(COLUMN\_USER\_PASSWORD, "customer123");  
 customer1Values.put(COLUMN\_USER\_ROLE, UserRole.ПОЛЬЗОВАТЕЛЬ.name());  
 customer1Values.put(COLUMN\_USER\_BALANCE, 5000.0);  
 db.insert(TABLE\_USERS, null, customer1Values);  
   
 ContentValues customer2Values = new ContentValues();  
 customer2Values.put(COLUMN\_USER\_NAME, "Мария Покупатель");  
 customer2Values.put(COLUMN\_USER\_EMAIL, "maria@example.com");  
 customer2Values.put(COLUMN\_USER\_PASSWORD, "maria123");  
 customer2Values.put(COLUMN\_USER\_ROLE, UserRole.ПОЛЬЗОВАТЕЛЬ.name());  
 customer2Values.put(COLUMN\_USER\_BALANCE, 2000.0);  
 db.insert(TABLE\_USERS, null, customer2Values);  
 }  
   
 private void insertInitialProducts(SQLiteDatabase db) {  
 String[][] products = {  
 {"Шоколадный торт", "Нежный шоколадный бисквит с кремом из темного шоколада и вишневым наполнителем", "1299.99", "15", "product\_cake\_chocolate", "Торты"},  
 {"Ванильный торт", "Воздушный бисквит с ванильным кремом и свежими ягодами", "1199.99", "12", "product\_cake\_vanilla", "Торты"},  
 {"Медовый торт", "Классический медовый торт со сметанным кремом и грецкими орехами", "1099.99", "8", "product\_cake\_honey", "Торты"},  
 {"Шоколадное печенье", "Хрустящее печенье с кусочками шоколада", "349.99", "30", "product\_cookie\_chocolate", "Печенье"},  
 {"Овсяное печенье", "Полезное печенье из овсяных хлопьев с изюмом", "299.99", "25", "product\_cookie\_oatmeal", "Печенье"},  
 {"Миндальное печенье", "Нежное печенье с миндальной мукой и лимонной цедрой", "399.99", "20", "product\_cookie\_almond", "Печенье"},  
 {"Ванильные кексы", "Классические кексы с ванильным ароматом", "249.99", "40", "product\_cupcake\_vanilla", "Кексы"},  
 {"Шоколадные кексы", "Кексы с добавлением какао и шоколадной глазурью", "279.99", "35", "product\_cupcake\_chocolate", "Кексы"},  
 {"Ягодные кексы", "Кексы с добавлением свежих ягод и глазурью", "299.99", "30", "product\_cupcake\_berry", "Кексы"},  
 {"Пончики с глазурью", "Классические пончики с сахарной глазурью", "199.99", "45", "product\_donut\_glazed", "Пончики"},  
 {"Пончики с шоколадом", "Пончики с шоколадной глазурью и посыпкой", "229.99", "40", "product\_donut\_chocolate", "Пончики"},  
 {"Пончики с начинкой", "Пончики с клубничной начинкой и сахарной пудрой", "249.99", "35", "product\_donut\_filled", "Пончики"},  
 {"Эклер с заварным кремом", "Классическое французское пирожное с нежным заварным кремом и шоколадной глазурью", "179.99", "25", "product\_pastry\_eclair", "Пирожные"},  
 {"Тирамису", "Итальянское пирожное с кофейной пропиткой, сыром маскарпоне и какао", "259.99", "15", "product\_pastry\_tiramisu", "Пирожные"},  
 {"Наполеон", "Традиционное многослойное пирожное с заварным кремом и хрустящими коржами", "229.99", "20", "product\_pastry\_napoleon", "Пирожные"},  
 {"Батон белый", "Классический белый хлеб из пшеничной муки высшего сорта", "89.99", "30", "product\_bread\_white", "Хлеб"},  
 {"Хлеб ржаной", "Традиционный ржаной хлеб с насыщенным вкусом и ароматом", "99.99", "25", "product\_bread\_rye", "Хлеб"},  
 {"Багет французский", "Хрустящий французский багет с мягким мякишем и хрустящей корочкой", "129.99", "15", "product\_bread\_baguette", "Хлеб"}  
 };  
   
 for (String[] product : products) {  
 ContentValues values = new ContentValues();  
 values.put(COLUMN\_PRODUCT\_NAME, product[0]);  
 values.put(COLUMN\_PRODUCT\_DESCRIPTION, product[1]);  
 values.put(COLUMN\_PRODUCT\_PRICE, Double.parseDouble(product[2]));  
 values.put(COLUMN\_PRODUCT\_STOCK, Integer.parseInt(product[3]));  
 values.put(COLUMN\_PRODUCT\_IMAGE, product[4]);  
 values.put(COLUMN\_PRODUCT\_CATEGORY, product[5]);  
 db.insert(TABLE\_PRODUCTS, null, values);  
 }  
 }  
   
 public User getUserByEmailAndPassword(String email, String password) {  
 SQLiteDatabase db = this.getReadableDatabase();  
   
 String[] columns = {  
 COLUMN\_USER\_ID,  
 COLUMN\_USER\_NAME,  
 COLUMN\_USER\_EMAIL,  
 COLUMN\_USER\_PASSWORD,  
 COLUMN\_USER\_ROLE,  
 COLUMN\_USER\_BALANCE  
 };  
   
 String selection = COLUMN\_USER\_EMAIL + " = ? AND " + COLUMN\_USER\_PASSWORD + " = ?";  
 String[] selectionArgs = {email, password};  
   
 Cursor cursor = db.query(  
 TABLE\_USERS,  
 columns,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
   
 User user = null;  
   
 if (cursor != null && cursor.moveToFirst()) {  
 int id = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN\_USER\_ID));  
 String name = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_NAME));  
 String userEmail = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_EMAIL));  
 String userPassword = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_PASSWORD));  
 String roleString = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_ROLE));  
 double balance = cursor.getDouble(cursor.getColumnIndexOrThrow(COLUMN\_USER\_BALANCE));  
   
 UserRole role = UserRole.valueOf(roleString);  
 user = new User(id, name, userEmail, userPassword, role, balance);  
 cursor.close();  
 }  
   
 return user;  
 }  
   
 public User getUserByEmail(String email) {  
 SQLiteDatabase db = this.getReadableDatabase();  
   
 String[] columns = {  
 COLUMN\_USER\_ID,  
 COLUMN\_USER\_NAME,  
 COLUMN\_USER\_EMAIL,  
 COLUMN\_USER\_PASSWORD,  
 COLUMN\_USER\_ROLE,  
 COLUMN\_USER\_BALANCE  
 };  
   
 String selection = COLUMN\_USER\_EMAIL + " = ?";  
 String[] selectionArgs = {email};  
   
 Cursor cursor = db.query(  
 TABLE\_USERS,  
 columns,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
   
 User user = null;  
   
 if (cursor != null && cursor.moveToFirst()) {  
 int id = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN\_USER\_ID));  
 String name = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_NAME));  
 String userEmail = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_EMAIL));  
 String userPassword = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_PASSWORD));  
 String roleString = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_ROLE));  
 double balance = cursor.getDouble(cursor.getColumnIndexOrThrow(COLUMN\_USER\_BALANCE));  
   
 UserRole role = UserRole.valueOf(roleString);  
 user = new User(id, name, userEmail, userPassword, role, balance);  
 cursor.close();  
 }  
   
 return user;  
 }  
   
 public User getUserByName(String name) {  
 SQLiteDatabase db = this.getReadableDatabase();  
   
 String[] columns = {  
 COLUMN\_USER\_ID,  
 COLUMN\_USER\_NAME,  
 COLUMN\_USER\_EMAIL,  
 COLUMN\_USER\_PASSWORD,  
 COLUMN\_USER\_ROLE,  
 COLUMN\_USER\_BALANCE  
 };  
   
 String selection = COLUMN\_USER\_NAME + " = ?";  
 String[] selectionArgs = {name};  
   
 Cursor cursor = db.query(  
 TABLE\_USERS,  
 columns,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
   
 User user = null;  
   
 if (cursor != null && cursor.moveToFirst()) {  
 int id = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN\_USER\_ID));  
 String userName = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_NAME));  
 String userEmail = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_EMAIL));  
 String userPassword = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_PASSWORD));  
 String roleString = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_ROLE));  
 double balance = cursor.getDouble(cursor.getColumnIndexOrThrow(COLUMN\_USER\_BALANCE));  
   
 UserRole role = UserRole.valueOf(roleString);  
 user = new User(id, userName, userEmail, userPassword, role, balance);  
 cursor.close();  
 }  
   
 return user;  
 }  
   
 public User registerUser(String name, String email, String password, UserRole role) {  
 if (getUserByEmail(email) != null) {  
 return null;  
 }  
   
 SQLiteDatabase db = this.getWritableDatabase();  
   
 ContentValues values = new ContentValues();  
 values.put(COLUMN\_USER\_NAME, name);  
 values.put(COLUMN\_USER\_EMAIL, email);  
 values.put(COLUMN\_USER\_PASSWORD, password);  
 values.put(COLUMN\_USER\_ROLE, role.name());  
 values.put(COLUMN\_USER\_BALANCE, 0.0);  
   
 long id = db.insert(TABLE\_USERS, null, values);  
   
 if (id == -1) {  
 return null;  
 }  
   
 return new User((int)id, name, email, password, role, 0.0);  
 }  
   
 public List<User> getAllUsers() {  
 List<User> userList = new ArrayList<>();  
   
 SQLiteDatabase db = this.getReadableDatabase();  
 Cursor cursor = db.query(  
 TABLE\_USERS,  
 null,  
 null,  
 null,  
 null,  
 null,  
 null  
 );  
   
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 int id = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN\_USER\_ID));  
 String name = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_NAME));  
 String email = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_EMAIL));  
 String password = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_PASSWORD));  
 String roleString = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_ROLE));  
 double balance = cursor.getDouble(cursor.getColumnIndexOrThrow(COLUMN\_USER\_BALANCE));  
   
 UserRole role = UserRole.valueOf(roleString);  
 User user = new User(id, name, email, password, role, balance);  
 userList.add(user);  
 } while (cursor.moveToNext());  
   
 cursor.close();  
 }  
   
 return userList;  
 }  
   
 public boolean updateUser(User user) {  
 SQLiteDatabase db = this.getWritableDatabase();  
   
 ContentValues values = new ContentValues();  
 values.put(COLUMN\_USER\_NAME, user.getName());  
 values.put(COLUMN\_USER\_PASSWORD, user.getPassword());  
 values.put(COLUMN\_USER\_ROLE, user.getRole().name());  
 values.put(COLUMN\_USER\_BALANCE, user.getBalance());  
   
 String whereClause = COLUMN\_USER\_EMAIL + " = ?";  
 String[] whereArgs = {user.getEmail()};  
   
 int rowsAffected = db.update(TABLE\_USERS, values, whereClause, whereArgs);  
 return rowsAffected > 0;  
 }  
   
 public boolean updateUserBalance(long userId, double newBalance) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 ContentValues values = new ContentValues();  
 values.put(COLUMN\_USER\_BALANCE, newBalance);  
   
 int rowsAffected = db.update(  
 TABLE\_USERS,  
 values,  
 COLUMN\_USER\_ID + " = ?",  
 new String[]{String.valueOf(userId)}  
 );  
   
 return rowsAffected > 0;  
 }  
   
 public boolean deleteUser(String email) {  
 SQLiteDatabase db = this.getWritableDatabase();  
   
 String whereClause = COLUMN\_USER\_EMAIL + " = ?";  
 String[] whereArgs = {email};  
   
 int rowsAffected = db.delete(TABLE\_USERS, whereClause, whereArgs);  
 return rowsAffected > 0;  
 }  
   
 public User getUserById(long id) {  
 SQLiteDatabase db = this.getReadableDatabase();  
   
 String[] columns = {  
 COLUMN\_USER\_ID,  
 COLUMN\_USER\_NAME,  
 COLUMN\_USER\_EMAIL,  
 COLUMN\_USER\_PASSWORD,  
 COLUMN\_USER\_ROLE,  
 COLUMN\_USER\_BALANCE  
 };  
   
 String selection = COLUMN\_USER\_ID + " = ?";  
 String[] selectionArgs = {String.valueOf(id)};  
   
 Cursor cursor = db.query(  
 TABLE\_USERS,  
 columns,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
   
 User user = null;  
   
 if (cursor != null && cursor.moveToFirst()) {  
 int userId = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN\_USER\_ID));  
 String name = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_NAME));  
 String userEmail = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_EMAIL));  
 String userPassword = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_PASSWORD));  
 String roleString = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_USER\_ROLE));  
 double balance = cursor.getDouble(cursor.getColumnIndexOrThrow(COLUMN\_USER\_BALANCE));  
   
 UserRole role = UserRole.valueOf(roleString);  
 user = new User(userId, name, userEmail, userPassword, role, balance);  
 cursor.close();  
 }  
   
 return user;  
 }  
   
 public long addProduct(Product product) {  
 SQLiteDatabase db = this.getWritableDatabase();  
   
 ContentValues values = new ContentValues();  
 values.put(COLUMN\_PRODUCT\_NAME, product.getName());  
 values.put(COLUMN\_PRODUCT\_DESCRIPTION, product.getDescription());  
 values.put(COLUMN\_PRODUCT\_PRICE, product.getPrice().doubleValue());  
 values.put(COLUMN\_PRODUCT\_STOCK, product.getStockQuantity());  
 values.put(COLUMN\_PRODUCT\_IMAGE, product.getImageUrl());  
 values.put(COLUMN\_PRODUCT\_CATEGORY, product.getCategory());  
   
 return db.insert(TABLE\_PRODUCTS, null, values);  
 }  
   
 public Product getProductById(long id) {  
 SQLiteDatabase db = this.getReadableDatabase();  
   
 String[] columns = {  
 COLUMN\_PRODUCT\_ID,  
 COLUMN\_PRODUCT\_NAME,  
 COLUMN\_PRODUCT\_DESCRIPTION,  
 COLUMN\_PRODUCT\_PRICE,  
 COLUMN\_PRODUCT\_STOCK,  
 COLUMN\_PRODUCT\_IMAGE,  
 COLUMN\_PRODUCT\_CATEGORY  
 };  
   
 String selection = COLUMN\_PRODUCT\_ID + " = ?";  
 String[] selectionArgs = {String.valueOf(id)};  
   
 Cursor cursor = db.query(  
 TABLE\_PRODUCTS,  
 columns,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 null  
 );  
   
 Product product = null;  
   
 if (cursor != null && cursor.moveToFirst()) {  
 String name = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_NAME));  
 String description = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_DESCRIPTION));  
 double price = cursor.getDouble(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_PRICE));  
 int stock = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_STOCK));  
 String imageUrl = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_IMAGE));  
 String category = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_CATEGORY));  
   
 product = new Product(  
 id,   
 name,   
 description,   
 "Ингредиенты не указаны",  
 "Информация о питательности не указана",  
 price,   
 R.drawable.ic\_cake,  
 category  
 );  
 product.setStockQuantity(stock);  
 if (imageUrl != null) {  
 product.setImageUrl(imageUrl);  
 }  
 cursor.close();  
 }  
   
 return product;  
 }  
   
 public List<Product> getAllProducts() {  
 List<Product> products = new ArrayList<>();  
 SQLiteDatabase db = this.getReadableDatabase();  
   
 Cursor cursor = db.query(  
 TABLE\_PRODUCTS,  
 null,  
 null,  
 null,  
 null,  
 null,  
 COLUMN\_PRODUCT\_NAME + " ASC"  
 );  
   
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 long id = cursor.getLong(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_ID));  
 String name = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_NAME));  
 String description = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_DESCRIPTION));  
 double price = cursor.getDouble(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_PRICE));  
 int stock = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_STOCK));  
 String imageUrl = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_IMAGE));  
 String category = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_CATEGORY));  
   
 Product product = new Product(  
 id,   
 name,   
 description,   
 "Ингредиенты не указаны",  
 "Информация о питательности не указана",  
 price,   
 R.drawable.ic\_cake,  
 category  
 );  
 product.setStockQuantity(stock);  
 if (imageUrl != null) {  
 product.setImageUrl(imageUrl);  
 }  
 products.add(product);  
 } while (cursor.moveToNext());  
   
 cursor.close();  
 }  
   
 if (products.isEmpty()) {  
 SQLiteDatabase writeDb = this.getWritableDatabase();  
 insertInitialProducts(writeDb);  
   
 return getAllProducts();  
 }  
   
 return products;  
 }  
   
 public List<Product> getProductsByCategory(String category) {  
 List<Product> productList = new ArrayList<>();  
   
 SQLiteDatabase db = this.getReadableDatabase();  
   
 String selection = COLUMN\_PRODUCT\_CATEGORY + " = ?";  
 String[] selectionArgs = {category};  
   
 Cursor cursor = db.query(  
 TABLE\_PRODUCTS,  
 null,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 COLUMN\_PRODUCT\_NAME + " ASC"  
 );  
   
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 long id = cursor.getLong(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_ID));  
 String name = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_NAME));  
 String description = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_DESCRIPTION));  
 double price = cursor.getDouble(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_PRICE));  
 int stock = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_STOCK));  
 String imageUrl = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_IMAGE));  
 String productCategory = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_CATEGORY));  
   
 Product product = new Product(  
 id,   
 name,   
 description,   
 "Ингредиенты не указаны",  
 "Информация о питательности не указана",  
 price,   
 R.drawable.ic\_cake,  
 productCategory  
 );  
 product.setStockQuantity(stock);  
 if (imageUrl != null) {  
 product.setImageUrl(imageUrl);  
 }  
 productList.add(product);  
 } while (cursor.moveToNext());  
   
 cursor.close();  
 }  
   
 return productList;  
 }  
   
 public boolean updateProduct(Product product) {  
 SQLiteDatabase db = this.getWritableDatabase();  
   
 ContentValues values = new ContentValues();  
 values.put(COLUMN\_PRODUCT\_NAME, product.getName());  
 values.put(COLUMN\_PRODUCT\_DESCRIPTION, product.getDescription());  
 values.put(COLUMN\_PRODUCT\_PRICE, product.getPrice().doubleValue());  
 values.put(COLUMN\_PRODUCT\_STOCK, product.getStockQuantity());  
 values.put(COLUMN\_PRODUCT\_IMAGE, product.getImageUrl());  
 values.put(COLUMN\_PRODUCT\_CATEGORY, product.getCategory());  
   
 String whereClause = COLUMN\_PRODUCT\_ID + " = ?";  
 String[] whereArgs = {String.valueOf(product.getId())};  
   
 int rowsAffected = db.update(TABLE\_PRODUCTS, values, whereClause, whereArgs);  
 return rowsAffected > 0;  
 }  
   
 public boolean deleteProduct(long id) {  
 SQLiteDatabase db = this.getWritableDatabase();  
   
 String whereClause = COLUMN\_PRODUCT\_ID + " = ?";  
 String[] whereArgs = {String.valueOf(id)};  
   
 int rowsAffected = db.delete(TABLE\_PRODUCTS, whereClause, whereArgs);  
 return rowsAffected > 0;  
 }  
   
 public List<String> getAllCategories() {  
 List<String> categories = new ArrayList<>();  
   
 SQLiteDatabase db = this.getReadableDatabase();  
 Cursor cursor = db.query(  
 true,  
 TABLE\_PRODUCTS,  
 new String[]{COLUMN\_PRODUCT\_CATEGORY},  
 null,  
 null,  
 COLUMN\_PRODUCT\_CATEGORY,  
 null,  
 COLUMN\_PRODUCT\_CATEGORY + " ASC",  
 null  
 );  
   
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 String category = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_CATEGORY));  
 categories.add(category);  
 } while (cursor.moveToNext());  
   
 cursor.close();  
 }  
   
 return categories;  
 }  
   
 public List<Product> searchProducts(String query) {  
 List<Product> results = new ArrayList<>();  
   
 SQLiteDatabase db = this.getReadableDatabase();  
   
 String selection = COLUMN\_PRODUCT\_NAME + " LIKE ? OR " + COLUMN\_PRODUCT\_DESCRIPTION + " LIKE ?";  
 String[] selectionArgs = {"%" + query + "%", "%" + query + "%"};  
   
 Cursor cursor = db.query(  
 TABLE\_PRODUCTS,  
 null,  
 selection,  
 selectionArgs,  
 null,  
 null,  
 COLUMN\_PRODUCT\_NAME + " ASC"  
 );  
   
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 long id = cursor.getLong(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_ID));  
 String name = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_NAME));  
 String description = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_DESCRIPTION));  
 double price = cursor.getDouble(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_PRICE));  
 int stock = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_STOCK));  
 String imageUrl = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_IMAGE));  
 String category = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_PRODUCT\_CATEGORY));  
   
 Product product = new Product(  
 id,   
 name,   
 description,   
 "Ингредиенты не указаны",  
 "Информация о питательности не указана",  
 price,   
 R.drawable.ic\_cake,  
 category  
 );  
 product.setStockQuantity(stock);  
 if (imageUrl != null) {  
 product.setImageUrl(imageUrl);  
 }  
 results.add(product);  
 } while (cursor.moveToNext());  
   
 cursor.close();  
 }  
   
 return results;  
 }  
  
 public long createOrder(Order order, ArrayList<OrderItem> orderItems) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 long orderId = -1;  
   
 Log.d("DatabaseHelper", "Начало транзакции для создания заказа");  
   
 db.beginTransaction();  
 try {  
 ContentValues orderValues = new ContentValues();  
 orderValues.put(COLUMN\_ORDER\_USER\_ID, order.getUserId());  
 orderValues.put(COLUMN\_ORDER\_TOTAL\_AMOUNT, order.getTotalAmount());  
 orderValues.put(COLUMN\_ORDER\_DATE, order.getOrderDate());  
 orderValues.put(COLUMN\_ORDER\_STATUS, order.getStatus());  
 orderValues.put(COLUMN\_ORDER\_PAID, order.isPaid() ? 1 : 0);  
   
 Log.d("DatabaseHelper", "Вставка заказа: " + orderValues);  
   
 orderId = db.insert(TABLE\_ORDERS, null, orderValues);  
   
 if (orderId == -1) {  
 Log.e("DatabaseHelper", "Ошибка при вставке заказа");  
 throw new Exception("Не удалось создать заказ");  
 }  
   
 Log.d("DatabaseHelper", "Заказ успешно создан с ID: " + orderId);  
   
 for (OrderItem item : orderItems) {  
 ContentValues itemValues = new ContentValues();  
 itemValues.put(COLUMN\_ORDER\_ITEM\_ORDER\_ID, orderId);  
 itemValues.put(COLUMN\_ORDER\_ITEM\_PRODUCT\_ID, item.getProductId());  
 itemValues.put(COLUMN\_ORDER\_ITEM\_QUANTITY, item.getQuantity());  
 itemValues.put(COLUMN\_ORDER\_ITEM\_PRICE, item.getPrice());  
 itemValues.put(COLUMN\_ORDER\_ITEM\_PRODUCT\_NAME, item.getProductName());  
   
 Log.d("DatabaseHelper", "Вставка товара заказа: " + itemValues);  
   
 long itemId = db.insert(TABLE\_ORDER\_ITEMS, null, itemValues);  
   
 if (itemId == -1) {  
 Log.e("DatabaseHelper", "Ошибка при вставке товара заказа: " + item.getProductName());  
 throw new Exception("Не удалось добавить товар " + item.getProductName() + " в заказ");  
 }  
   
 Log.d("DatabaseHelper", "Товар заказа успешно добавлен с ID: " + itemId);  
 }  
   
 db.setTransactionSuccessful();  
 Log.d("DatabaseHelper", "Транзакция успешно завершена");  
 } catch (Exception e) {  
 Log.e("DatabaseHelper", "Ошибка в транзакции: " + e.getMessage(), e);  
 orderId = -1;  
 } finally {  
 db.endTransaction();  
 Log.d("DatabaseHelper", "Транзакция завершена");  
 }  
   
 return orderId;  
 }  
   
 public Order getOrder(long orderId) {  
 SQLiteDatabase db = this.getReadableDatabase();  
   
 String[] columns = {  
 COLUMN\_ORDER\_ID,  
 COLUMN\_ORDER\_USER\_ID,  
 COLUMN\_ORDER\_DATE,  
 COLUMN\_ORDER\_STATUS,  
 COLUMN\_ORDER\_TOTAL\_AMOUNT,  
 COLUMN\_ORDER\_PAID,  
 COLUMN\_ORDER\_CANCELLATION\_REASON  
 };  
   
 String selection = COLUMN\_ORDER\_ID + " = ?";  
 String[] selectionArgs = {String.valueOf(orderId)};  
   
 Cursor cursor = db.query(TABLE\_ORDERS, columns, selection, selectionArgs, null, null, null);  
   
 Order order = null;  
   
 if (cursor != null && cursor.moveToFirst()) {  
 long orderUserId = cursor.getLong(cursor.getColumnIndex(COLUMN\_ORDER\_USER\_ID));  
 String orderDate = cursor.getString(cursor.getColumnIndex(COLUMN\_ORDER\_DATE));  
 String status = cursor.getString(cursor.getColumnIndex(COLUMN\_ORDER\_STATUS));  
 double totalAmount = cursor.getDouble(cursor.getColumnIndex(COLUMN\_ORDER\_TOTAL\_AMOUNT));  
 boolean isPaid = cursor.getInt(cursor.getColumnIndex(COLUMN\_ORDER\_PAID)) == 1;  
   
 String cancellationReason = "";  
 int reasonIndex = cursor.getColumnIndex(COLUMN\_ORDER\_CANCELLATION\_REASON);  
 if (reasonIndex != -1 && !cursor.isNull(reasonIndex)) {  
 cancellationReason = cursor.getString(reasonIndex);  
 }  
   
 List<OrderItem> items = getOrderItems(orderId);  
   
 order = new Order(orderId, orderUserId, orderDate, status, totalAmount, "ORD-" + orderId, items, isPaid);  
 order.setCancellationReason(cancellationReason);  
 }  
   
 if (cursor != null) {  
 cursor.close();  
 }  
   
 return order;  
 }  
   
 public List<Order> getUserOrders(long userId) {  
 SQLiteDatabase db = this.getReadableDatabase();  
 List<Order> orders = new ArrayList<>();  
   
 String[] columns = {  
 COLUMN\_ORDER\_ID,  
 COLUMN\_ORDER\_USER\_ID,  
 COLUMN\_ORDER\_DATE,  
 COLUMN\_ORDER\_STATUS,  
 COLUMN\_ORDER\_TOTAL\_AMOUNT,  
 COLUMN\_ORDER\_PAID  
 };  
   
 String selection = COLUMN\_ORDER\_USER\_ID + " = ?";  
 String[] selectionArgs = {String.valueOf(userId)};  
 String orderBy = COLUMN\_ORDER\_DATE + " DESC";  
   
 Cursor cursor = db.query(TABLE\_ORDERS, columns, selection, selectionArgs, null, null, orderBy);  
   
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 long orderId = cursor.getLong(cursor.getColumnIndex(COLUMN\_ORDER\_ID));  
 String orderDate = cursor.getString(cursor.getColumnIndex(COLUMN\_ORDER\_DATE));  
 String status = cursor.getString(cursor.getColumnIndex(COLUMN\_ORDER\_STATUS));  
 double totalAmount = cursor.getDouble(cursor.getColumnIndex(COLUMN\_ORDER\_TOTAL\_AMOUNT));  
 boolean isPaid = cursor.getInt(cursor.getColumnIndex(COLUMN\_ORDER\_PAID)) == 1;  
   
 List<OrderItem> items = getOrderItems(orderId);  
   
 Order order = new Order(orderId, userId, orderDate, status, totalAmount, "ORD-" + orderId, items, isPaid);  
 orders.add(order);  
 } while (cursor.moveToNext());  
 }  
   
 if (cursor != null) {  
 cursor.close();  
 }  
   
 return orders;  
 }  
   
 public List<Order> getAllOrders() {  
 SQLiteDatabase db = this.getReadableDatabase();  
 List<Order> orders = new ArrayList<>();  
   
 String[] columns = {  
 COLUMN\_ORDER\_ID,  
 COLUMN\_ORDER\_USER\_ID,  
 COLUMN\_ORDER\_DATE,  
 COLUMN\_ORDER\_STATUS,  
 COLUMN\_ORDER\_TOTAL\_AMOUNT,  
 COLUMN\_ORDER\_PAID  
 };  
   
 String orderBy = COLUMN\_ORDER\_DATE + " DESC";  
   
 Cursor cursor = db.query(TABLE\_ORDERS, columns, null, null, null, null, orderBy);  
   
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 long orderId = cursor.getLong(cursor.getColumnIndex(COLUMN\_ORDER\_ID));  
 long userId = cursor.getLong(cursor.getColumnIndex(COLUMN\_ORDER\_USER\_ID));  
 String orderDate = cursor.getString(cursor.getColumnIndex(COLUMN\_ORDER\_DATE));  
 String status = cursor.getString(cursor.getColumnIndex(COLUMN\_ORDER\_STATUS));  
 double totalAmount = cursor.getDouble(cursor.getColumnIndex(COLUMN\_ORDER\_TOTAL\_AMOUNT));  
 boolean isPaid = cursor.getInt(cursor.getColumnIndex(COLUMN\_ORDER\_PAID)) == 1;  
   
 List<OrderItem> items = getOrderItems(orderId);  
   
 Order order = new Order(orderId, userId, orderDate, status, totalAmount, "ORD-" + orderId, items, isPaid);  
 orders.add(order);  
 } while (cursor.moveToNext());  
 }  
   
 if (cursor != null) {  
 cursor.close();  
 }  
   
 return orders;  
 }  
   
 private List<OrderItem> getOrderItems(long orderId) {  
 List<OrderItem> items = new ArrayList<>();  
 SQLiteDatabase db = this.getReadableDatabase();  
   
 String[] columns = {  
 COLUMN\_ORDER\_ITEM\_ID,  
 COLUMN\_ORDER\_ITEM\_ORDER\_ID,  
 COLUMN\_ORDER\_ITEM\_PRODUCT\_ID,  
 COLUMN\_ORDER\_ITEM\_QUANTITY,  
 COLUMN\_ORDER\_ITEM\_PRICE,  
 COLUMN\_ORDER\_ITEM\_PRODUCT\_NAME  
 };  
   
 String selection = COLUMN\_ORDER\_ITEM\_ORDER\_ID + " = ?";  
 String[] selectionArgs = {String.valueOf(orderId)};  
   
 Cursor cursor = db.query(TABLE\_ORDER\_ITEMS, columns, selection, selectionArgs, null, null, null);  
   
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 long id = cursor.getLong(cursor.getColumnIndex(COLUMN\_ORDER\_ITEM\_ID));  
 long itemOrderId = cursor.getLong(cursor.getColumnIndex(COLUMN\_ORDER\_ITEM\_ORDER\_ID));  
 long productId = cursor.getLong(cursor.getColumnIndex(COLUMN\_ORDER\_ITEM\_PRODUCT\_ID));  
 int quantity = cursor.getInt(cursor.getColumnIndex(COLUMN\_ORDER\_ITEM\_QUANTITY));  
 double price = cursor.getDouble(cursor.getColumnIndex(COLUMN\_ORDER\_ITEM\_PRICE));  
 String productName = cursor.getString(cursor.getColumnIndex(COLUMN\_ORDER\_ITEM\_PRODUCT\_NAME));  
   
 OrderItem item = new OrderItem(id, itemOrderId, productId, quantity, price, productName);  
 items.add(item);  
 } while (cursor.moveToNext());  
 }  
   
 if (cursor != null) {  
 cursor.close();  
 }  
   
 return items;  
 }  
   
 public boolean updateOrderStatus(long orderId, String newStatus) {  
 try {  
 if (orderId <= 0) {  
 Log.e("DatabaseHelper", "Некорректный ID заказа: " + orderId);  
 return false;  
 }  
   
 if (newStatus == null || newStatus.isEmpty()) {  
 Log.e("DatabaseHelper", "Пустой статус заказа");  
 return false;  
 }  
   
 SQLiteDatabase db = this.getWritableDatabase();  
 ContentValues values = new ContentValues();  
 values.put(COLUMN\_ORDER\_STATUS, newStatus);  
   
 String[] columns = {COLUMN\_ORDER\_ID};  
 String selection = COLUMN\_ORDER\_ID + " = ?";  
 String[] selectionArgs = {String.valueOf(orderId)};  
 Cursor cursor = db.query(TABLE\_ORDERS, columns, selection, selectionArgs, null, null, null);  
   
 boolean exists = cursor != null && cursor.moveToFirst();  
   
 if (cursor != null) {  
 cursor.close();  
 }  
   
 if (!exists) {  
 Log.e("DatabaseHelper", "Заказ с ID " + orderId + " не найден");  
 return false;  
 }  
   
 int rowsAffected = db.update(TABLE\_ORDERS, values,  
 COLUMN\_ORDER\_ID + "=?", new String[]{String.valueOf(orderId)});  
   
 boolean result = rowsAffected > 0;  
   
 if (result) {  
 Log.d("DatabaseHelper", "Статус заказа " + orderId + " обновлен на " + newStatus);  
 } else {  
 Log.e("DatabaseHelper", "Не удалось обновить статус заказа " + orderId);  
 }  
   
 return result;  
 } catch (Exception e) {  
 Log.e("DatabaseHelper", "Ошибка при обновлении статуса заказа: " + e.getMessage(), e);  
 return false;  
 }  
 }  
  
 public boolean saveUserProfile(long userId, String phone, String address, String city, String postalCode) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 ContentValues values = new ContentValues();  
 values.put(COLUMN\_PROFILE\_PHONE, phone);  
 values.put(COLUMN\_PROFILE\_ADDRESS, address);  
 values.put(COLUMN\_PROFILE\_CITY, city);  
 values.put(COLUMN\_PROFILE\_POSTAL\_CODE, postalCode);  
  
 Cursor cursor = db.query(TABLE\_USER\_PROFILES, null,  
 COLUMN\_PROFILE\_USER\_ID + "=?", new String[]{String.valueOf(userId)},  
 null, null, null);  
  
 if (cursor.moveToFirst()) {  
 int rowsAffected = db.update(TABLE\_USER\_PROFILES, values,  
 COLUMN\_PROFILE\_USER\_ID + "=?", new String[]{String.valueOf(userId)});  
 cursor.close();  
 return rowsAffected > 0;  
 } else {  
 values.put(COLUMN\_PROFILE\_USER\_ID, userId);  
 long result = db.insert(TABLE\_USER\_PROFILES, null, values);  
 cursor.close();  
 return result != -1;  
 }  
 }  
  
 public UserProfile getUserProfile(long userId) {  
 SQLiteDatabase db = this.getReadableDatabase();  
   
 User user = getUserById(userId);  
 if (user == null) {  
 return null;  
 }  
  
 UserProfile profile = new UserProfile(user.getName(), user.getEmail());  
   
 Cursor cursor = db.query(TABLE\_USER\_PROFILES, null,  
 COLUMN\_PROFILE\_USER\_ID + "=?", new String[]{String.valueOf(userId)},  
 null, null, null);  
  
 if (cursor != null && cursor.moveToFirst()) {  
 int phoneColumnIndex = cursor.getColumnIndex(COLUMN\_PROFILE\_PHONE);  
 int addressColumnIndex = cursor.getColumnIndex(COLUMN\_PROFILE\_ADDRESS);  
 int cityColumnIndex = cursor.getColumnIndex(COLUMN\_PROFILE\_CITY);  
 int postalCodeColumnIndex = cursor.getColumnIndex(COLUMN\_PROFILE\_POSTAL\_CODE);  
   
 String phone = phoneColumnIndex >= 0 ? cursor.getString(phoneColumnIndex) : "";  
 String address = addressColumnIndex >= 0 ? cursor.getString(addressColumnIndex) : "";  
 String city = cityColumnIndex >= 0 ? cursor.getString(cityColumnIndex) : "";  
 String postalCode = postalCodeColumnIndex >= 0 ? cursor.getString(postalCodeColumnIndex) : "";  
   
 profile.setPhone(phone != null ? phone : "");  
 profile.setAddress(address != null ? address : "");  
 profile.setCity(city != null ? city : "");  
 profile.setPostalCode(postalCode != null ? postalCode : "");  
 }  
   
 if (cursor != null) {  
 cursor.close();  
 }  
   
 return profile;  
 }  
  
 public boolean deleteUserProfile(long userId) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 int rowsAffected = db.delete(TABLE\_USER\_PROFILES,  
 COLUMN\_PROFILE\_USER\_ID + "=?", new String[]{String.valueOf(userId)});  
 return rowsAffected > 0;  
 }  
  
 public boolean updateOrderPaymentStatus(long orderId, boolean isPaid) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 ContentValues values = new ContentValues();  
 values.put(COLUMN\_ORDER\_PAID, isPaid ? 1 : 0);  
   
 int rowsAffected = db.update(TABLE\_ORDERS, values,  
 COLUMN\_ORDER\_ID + "=?", new String[]{String.valueOf(orderId)});  
   
 return rowsAffected > 0;  
 }  
  
 public boolean updateOrderStatusWithReason(long orderId, String newStatus, String cancellationReason) {  
 try {  
 if (orderId <= 0) {  
 Log.e("DatabaseHelper", "Некорректный ID заказа: " + orderId);  
 return false;  
 }  
   
 if (newStatus == null || newStatus.isEmpty()) {  
 Log.e("DatabaseHelper", "Пустой статус заказа");  
 return false;  
 }  
   
 SQLiteDatabase db = this.getWritableDatabase();  
 String[] columns = {COLUMN\_ORDER\_ID};  
 String selection = COLUMN\_ORDER\_ID + " = ?";  
 String[] selectionArgs = {String.valueOf(orderId)};  
 Cursor cursor = db.query(TABLE\_ORDERS, columns, selection, selectionArgs, null, null, null);  
   
 boolean exists = cursor != null && cursor.moveToFirst();  
   
 if (cursor != null) {  
 cursor.close();  
 }  
   
 if (!exists) {  
 Log.e("DatabaseHelper", "Заказ с ID " + orderId + " не найден");  
 return false;  
 }  
   
 ContentValues values = new ContentValues();  
 values.put(COLUMN\_ORDER\_STATUS, newStatus);  
   
 if ("Отменен".equals(newStatus)) {  
 if (cancellationReason == null || cancellationReason.isEmpty()) {  
 Log.w("DatabaseHelper", "Пустая причина отмены для заказа " + orderId);  
 values.put(COLUMN\_ORDER\_CANCELLATION\_REASON, "Причина не указана");  
 } else {  
 values.put(COLUMN\_ORDER\_CANCELLATION\_REASON, cancellationReason);  
 }  
 }  
   
 int rowsAffected = db.update(TABLE\_ORDERS, values,  
 COLUMN\_ORDER\_ID + "=?", new String[]{String.valueOf(orderId)});  
   
 boolean result = rowsAffected > 0;  
   
 if (result) {  
 Log.d("DatabaseHelper", "Статус заказа " + orderId + " обновлен на " + newStatus +   
 ("Отменен".equals(newStatus) ? " с причиной: " + cancellationReason : ""));  
 } else {  
 Log.e("DatabaseHelper", "Не удалось обновить статус заказа " + orderId);  
 }  
   
 return result;  
 } catch (Exception e) {  
 Log.e("DatabaseHelper", "Ошибка при обновлении статуса заказа с причиной: " + e.getMessage(), e);  
 return false;  
 }  
 }  
   
 public String getOrderCancellationReason(long orderId) {  
 SQLiteDatabase db = this.getReadableDatabase();  
 String reason = "";  
   
 String[] columns = { COLUMN\_ORDER\_CANCELLATION\_REASON };  
 String selection = COLUMN\_ORDER\_ID + " = ?";  
 String[] selectionArgs = { String.valueOf(orderId) };  
   
 Cursor cursor = db.query(TABLE\_ORDERS, columns, selection, selectionArgs, null, null, null);  
   
 if (cursor != null && cursor.moveToFirst()) {  
 int columnIndex = cursor.getColumnIndex(COLUMN\_ORDER\_CANCELLATION\_REASON);  
 if (columnIndex != -1 && !cursor.isNull(columnIndex)) {  
 reason = cursor.getString(columnIndex);  
 }  
 cursor.close();  
 }  
   
 return reason;  
 }  
}

package com.example.upavlika.fragments;  
  
import android.app.AlertDialog;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ArrayAdapter;  
import android.widget.LinearLayout;  
import android.widget.ListView;  
import android.widget.Spinner;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.fragment.app.Fragment;  
  
import com.example.upavlika.R;  
import com.example.upavlika.data.DatabaseHelper;  
import com.example.upavlika.model.Product;  
import com.google.android.material.floatingactionbutton.FloatingActionButton;  
import com.google.android.material.textfield.TextInputEditText;  
  
import java.math.BigDecimal;  
import java.util.ArrayList;  
import java.util.List;  
  
public class ManageProductsFragment extends Fragment {  
  
 private DatabaseHelper dbHelper;  
 private ListView productsListView;  
 private TextView emptyTextView;  
 private LinearLayout containerLayout;  
 private FloatingActionButton addProductFab;  
 private List<Product> productList;  
 private ArrayAdapter<Product> productAdapter;  
  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 View rootView = inflater.inflate(R.layout.fragment\_manage\_products, container, false);  
  
   
 dbHelper = DatabaseHelper.getInstance(requireContext());  
  
   
 productsListView = rootView.findViewById(R.id.productsListView);  
 emptyTextView = rootView.findViewById(R.id.emptyTextView);  
 containerLayout = rootView.findViewById(R.id.containerLayout);  
 addProductFab = rootView.findViewById(R.id.addProductFab);  
  
   
 loadProducts();  
  
   
 productsListView.setOnItemClickListener((parent, view, position, id) -> {  
 Product selectedProduct = productList.get(position);  
 showProductActionsDialog(selectedProduct);  
 });  
  
   
 addProductFab.setOnClickListener(v -> showAddProductDialog());  
  
 return rootView;  
 }  
  
 private void loadProducts() {  
   
 emptyTextView.setVisibility(View.VISIBLE);  
 emptyTextView.setText("Загрузка...");  
  
   
 new Thread(() -> {  
 productList = dbHelper.getAllProducts();  
   
   
 for (Product p : productList) {  
 Log.d("ManageProductsFragment", "Loaded product: " + p.getId() + " - " + p.getName());  
 }  
  
   
 requireActivity().runOnUiThread(() -> {  
 if (productList.isEmpty()) {  
 productsListView.setVisibility(View.GONE);  
 emptyTextView.setVisibility(View.VISIBLE);  
 emptyTextView.setText("Список товаров пуст");  
 } else {  
 productsListView.setVisibility(View.VISIBLE);  
 emptyTextView.setVisibility(View.GONE);  
  
   
 productAdapter = new ArrayAdapter<Product>(requireContext(), android.R.layout.simple\_list\_item\_1, productList) {  
 @NonNull  
 @Override  
 public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {  
 View view = super.getView(position, convertView, parent);  
 TextView textView = view.findViewById(android.R.id.text1);  
 Product product = getItem(position);  
 textView.setText(product.getName() + " - " + product.getPrice() + " ₽ (" + product.getCategory() + ")");  
 return view;  
 }  
 };  
 productsListView.setAdapter(productAdapter);  
 }  
 });  
 }).start();  
 }  
  
 private void showProductActionsDialog(Product product) {  
 String[] options = {"Редактировать товар", "Удалить товар"};  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(requireContext());  
 builder.setTitle(product.getName())  
 .setItems(options, (dialog, which) -> {  
 switch (which) {  
 case 0:  
 showEditProductDialog(product);  
 break;  
 case 1:  
 showDeleteProductDialog(product);  
 break;  
 }  
 })  
 .setNegativeButton("Отмена", null)  
 .show();  
 }  
  
 private void showEditProductDialog(Product product) {  
 View dialogView = LayoutInflater.from(requireContext()).inflate(R.layout.dialog\_edit\_product, null);  
   
 TextInputEditText nameEditText = dialogView.findViewById(R.id.nameEditText);  
 TextInputEditText descriptionEditText = dialogView.findViewById(R.id.descriptionEditText);  
 TextInputEditText priceEditText = dialogView.findViewById(R.id.priceEditText);  
 TextInputEditText stockEditText = dialogView.findViewById(R.id.stockEditText);  
   
   
 Spinner categorySpinner = dialogView.findViewById(R.id.categorySpinner);  
   
 TextInputEditText ingredientsEditText = dialogView.findViewById(R.id.ingredientsEditText);  
 TextInputEditText nutritionEditText = dialogView.findViewById(R.id.nutritionEditText);  
  
   
 nameEditText.setText(product.getName());  
 descriptionEditText.setText(product.getDescription());  
 priceEditText.setText(product.getPrice().toString());  
 stockEditText.setText(String.valueOf(product.getStockQuantity()));  
 ingredientsEditText.setText(product.getIngredients());  
 nutritionEditText.setText(product.getNutritionInfo());  
  
   
 setupCategorySpinner(categorySpinner, product.getCategory());  
  
 new AlertDialog.Builder(requireContext())  
 .setTitle("Редактировать товар")  
 .setView(dialogView)  
 .setPositiveButton("Сохранить", (dialog, which) -> {  
   
 String name = nameEditText.getText().toString().trim();  
 String description = descriptionEditText.getText().toString().trim();  
 String priceString = priceEditText.getText().toString().trim();  
 String stockString = stockEditText.getText().toString().trim();  
 String category = categorySpinner.getSelectedItem().toString();  
 String ingredients = ingredientsEditText.getText().toString().trim();  
 String nutrition = nutritionEditText.getText().toString().trim();  
  
   
 if (name.isEmpty() || priceString.isEmpty()) {  
 Toast.makeText(requireContext(), "Необходимо заполнить обязательные поля", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
  
 try {  
 double price = Double.parseDouble(priceString);  
 int stock = stockString.isEmpty() ? 0 : Integer.parseInt(stockString);  
  
   
 Product updatedProduct = new Product(  
 product.getId(),  
 name,  
 description,  
 ingredients,  
 nutrition,  
 price,  
 R.drawable.ic\_cake,   
 category  
 );  
 updatedProduct.setStockQuantity(stock);  
  
   
 new Thread(() -> {  
 boolean success = dbHelper.updateProduct(updatedProduct);  
 requireActivity().runOnUiThread(() -> {  
 if (success) {  
 Toast.makeText(requireContext(), "Товар обновлен", Toast.LENGTH\_SHORT).show();  
 loadProducts();   
 } else {  
 Toast.makeText(requireContext(), "Ошибка при обновлении товара", Toast.LENGTH\_SHORT).show();  
 }  
 });  
 }).start();  
 } catch (NumberFormatException e) {  
 Toast.makeText(requireContext(), "Неверный формат цены или количества", Toast.LENGTH\_SHORT).show();  
 }  
 })  
 .setNegativeButton("Отмена", null)  
 .show();  
 }  
  
 private void showDeleteProductDialog(Product product) {  
 new AlertDialog.Builder(requireContext())  
 .setTitle("Удаление товара")  
 .setMessage("Вы уверены, что хотите удалить товар \"" + product.getName() + "\"?")  
 .setPositiveButton("Удалить", (dialog, which) -> {  
   
 new Thread(() -> {  
 boolean success = dbHelper.deleteProduct(product.getId());  
 requireActivity().runOnUiThread(() -> {  
 if (success) {  
 Toast.makeText(requireContext(), "Товар удален", Toast.LENGTH\_SHORT).show();  
 loadProducts();   
 } else {  
 Toast.makeText(requireContext(), "Ошибка при удалении товара", Toast.LENGTH\_SHORT).show();  
 }  
 });  
 }).start();  
 })  
 .setNegativeButton("Отмена", null)  
 .show();  
 }  
  
 private void showAddProductDialog() {  
 View dialogView = LayoutInflater.from(requireContext()).inflate(R.layout.dialog\_edit\_product, null);  
   
 TextInputEditText nameEditText = dialogView.findViewById(R.id.nameEditText);  
 TextInputEditText descriptionEditText = dialogView.findViewById(R.id.descriptionEditText);  
 TextInputEditText priceEditText = dialogView.findViewById(R.id.priceEditText);  
 TextInputEditText stockEditText = dialogView.findViewById(R.id.stockEditText);  
   
   
 Spinner categorySpinner = dialogView.findViewById(R.id.categorySpinner);  
   
 TextInputEditText ingredientsEditText = dialogView.findViewById(R.id.ingredientsEditText);  
 TextInputEditText nutritionEditText = dialogView.findViewById(R.id.nutritionEditText);  
  
   
 setupCategorySpinner(categorySpinner, null);  
  
 new AlertDialog.Builder(requireContext())  
 .setTitle("Добавить товар")  
 .setView(dialogView)  
 .setPositiveButton("Добавить", (dialog, which) -> {  
   
 String name = nameEditText.getText().toString().trim();  
 String description = descriptionEditText.getText().toString().trim();  
 String priceString = priceEditText.getText().toString().trim();  
 String stockString = stockEditText.getText().toString().trim();  
 String category = categorySpinner.getSelectedItem().toString();  
 String ingredients = ingredientsEditText.getText().toString().trim();  
 String nutrition = nutritionEditText.getText().toString().trim();  
  
   
 if (name.isEmpty() || priceString.isEmpty()) {  
 Toast.makeText(requireContext(), "Необходимо заполнить обязательные поля", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
  
 try {  
 double price = Double.parseDouble(priceString);  
 int stock = stockString.isEmpty() ? 0 : Integer.parseInt(stockString);  
  
   
 Product newProduct = new Product(  
 0,   
 name,  
 description,  
 ingredients.isEmpty() ? "Ингредиенты не указаны" : ingredients,  
 nutrition.isEmpty() ? "Информация о питательности не указана" : nutrition,  
 price,  
 R.drawable.ic\_cake,   
 category  
 );  
 newProduct.setStockQuantity(stock);  
  
   
 Log.d("ManageProductsFragment", "Adding product: " + newProduct.getName());  
  
   
 new Thread(() -> {  
 long productId = dbHelper.addProduct(newProduct);  
 Log.d("ManageProductsFragment", "Product added with ID: " + productId);  
   
 if (productId != -1) {  
 requireActivity().runOnUiThread(() -> {  
 Toast.makeText(requireContext(), "Товар добавлен", Toast.LENGTH\_SHORT).show();  
 loadProducts();   
 });  
 } else {  
 requireActivity().runOnUiThread(() -> {  
 Toast.makeText(requireContext(), "Ошибка при добавлении товара", Toast.LENGTH\_SHORT).show();  
 });  
 }  
 }).start();  
 } catch (NumberFormatException e) {  
 Toast.makeText(requireContext(), "Неверный формат цены или количества", Toast.LENGTH\_SHORT).show();  
 }  
 })  
 .setNegativeButton("Отмена", null)  
 .show();  
 }  
  
   
 private void setupCategorySpinner(Spinner spinner, String selectedCategory) {  
   
 new Thread(() -> {  
 List<String> categories = dbHelper.getAllCategories();  
   
   
   
 List<String> normalizedCategories = new ArrayList<>();  
 for (String category : categories) {  
 String normalized = category.trim().toLowerCase();  
 boolean isDuplicate = false;  
   
   
 for (String existing : normalizedCategories) {  
 if (existing.equals(normalized)) {  
 isDuplicate = true;  
 break;  
 }  
 }  
   
 if (!isDuplicate) {  
   
 normalizedCategories.add(normalized);  
 }  
 }  
   
   
 List<String> standardCategories = new ArrayList<>();  
 standardCategories.add("Торты");  
 standardCategories.add("Пирожные");  
 standardCategories.add("Печенье");  
 standardCategories.add("Хлеб");  
 standardCategories.add("Выпечка");  
 standardCategories.add("Другое");  
   
   
 final List<String> finalCategories = new ArrayList<>();  
 if (normalizedCategories.isEmpty()) {  
 finalCategories.addAll(standardCategories);  
 } else {  
   
 for (String normalized : normalizedCategories) {  
 boolean found = false;  
 for (String standard : standardCategories) {  
 if (standard.toLowerCase().equals(normalized)) {  
 finalCategories.add(standard);  
 found = true;  
 break;  
 }  
 }  
 if (!found) {  
   
 finalCategories.add(normalized.substring(0, 1).toUpperCase() + normalized.substring(1));  
 }  
 }  
 }  
   
   
 final ArrayAdapter<String> adapter = new ArrayAdapter<>(  
 requireContext(), android.R.layout.simple\_spinner\_item, finalCategories);  
 adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);  
   
 requireActivity().runOnUiThread(() -> {  
 spinner.setAdapter(adapter);  
   
   
 if (selectedCategory != null && !selectedCategory.isEmpty()) {  
   
 final String normalizedSelected = selectedCategory.trim().toLowerCase();  
   
   
 int position = -1;  
 for (int i = 0; i < finalCategories.size(); i++) {  
 if (finalCategories.get(i).toLowerCase().equals(normalizedSelected)) {  
 position = i;  
 break;  
 }  
 }  
   
 final int finalPosition = position;  
 if (finalPosition >= 0) {  
 spinner.setSelection(finalPosition);  
 } else {  
 spinner.setSelection(0);   
 }  
 }  
 });  
 }).start();  
 }  
}

package com.example.upavlika.fragments;  
  
import android.app.AlertDialog;  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.AdapterView;  
import android.widget.ArrayAdapter;  
import android.widget.LinearLayout;  
import android.widget.ListView;  
import android.widget.RadioGroup;  
import android.widget.RadioButton;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.fragment.app.Fragment;  
  
import com.example.upavlika.R;  
import com.example.upavlika.data.DatabaseHelper;  
import com.example.upavlika.model.User;  
import com.example.upavlika.model.UserRole;  
import com.google.android.material.floatingactionbutton.FloatingActionButton;  
  
import java.util.List;  
  
public class ManageUsersFragment extends Fragment {  
  
 private DatabaseHelper dbHelper;  
 private ListView usersListView;  
 private TextView emptyTextView;  
 private LinearLayout containerLayout;  
 private FloatingActionButton addUserFab;  
 private List<User> userList;  
 private ArrayAdapter<User> userAdapter;  
  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 View rootView = inflater.inflate(R.layout.fragment\_manage\_users, container, false);  
  
   
 dbHelper = DatabaseHelper.getInstance(requireContext());  
  
   
 usersListView = rootView.findViewById(R.id.usersListView);  
 emptyTextView = rootView.findViewById(R.id.emptyTextView);  
 containerLayout = rootView.findViewById(R.id.containerLayout);  
 addUserFab = rootView.findViewById(R.id.addUserFab);  
  
   
 loadUsers();  
  
   
 usersListView.setOnItemClickListener((parent, view, position, id) -> {  
 User selectedUser = userList.get(position);  
 showUserActionsDialog(selectedUser);  
 });  
  
   
 addUserFab.setOnClickListener(v -> showAddUserDialog());  
  
 return rootView;  
 }  
  
 private void loadUsers() {  
   
 emptyTextView.setVisibility(View.VISIBLE);  
 emptyTextView.setText("Загрузка...");  
  
   
 new Thread(() -> {  
 userList = dbHelper.getAllUsers();  
  
   
 requireActivity().runOnUiThread(() -> {  
 if (userList.isEmpty()) {  
 usersListView.setVisibility(View.GONE);  
 emptyTextView.setVisibility(View.VISIBLE);  
 emptyTextView.setText("Список пользователей пуст");  
 } else {  
 usersListView.setVisibility(View.VISIBLE);  
 emptyTextView.setVisibility(View.GONE);  
  
   
 userAdapter = new ArrayAdapter<User>(requireContext(), android.R.layout.simple\_list\_item\_1, userList) {  
 @NonNull  
 @Override  
 public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {  
 View view = super.getView(position, convertView, parent);  
 TextView textView = view.findViewById(android.R.id.text1);  
 User user = getItem(position);  
 textView.setText(user.getName() + " (" + user.getEmail() + ") - " + user.getRole());  
 return view;  
 }  
 };  
 usersListView.setAdapter(userAdapter);  
 }  
 });  
 }).start();  
 }  
  
 private void showUserActionsDialog(User user) {  
 String[] options = {"Изменить роль", "Удалить пользователя"};  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(requireContext());  
 builder.setTitle(user.getName())  
 .setItems(options, (dialog, which) -> {  
 switch (which) {  
 case 0:  
 showChangeRoleDialog(user);  
 break;  
 case 1:  
 showDeleteUserDialog(user);  
 break;  
 }  
 })  
 .setNegativeButton("Отмена", null)  
 .show();  
 }  
  
 private void showChangeRoleDialog(User user) {  
   
 View dialogView = LayoutInflater.from(requireContext()).inflate(R.layout.dialog\_change\_role, null);  
 TextView titleTextView = dialogView.findViewById(R.id.titleTextView);  
 RadioGroup roleRadioGroup = dialogView.findViewById(R.id.roleRadioGroup);  
 RadioButton customerRadioButton = dialogView.findViewById(R.id.customerRadioButton);  
 RadioButton adminRadioButton = dialogView.findViewById(R.id.adminRadioButton);  
  
   
 titleTextView.setText("Изменить роль для " + user.getName());  
  
   
 if (user.getRole().equals(UserRole.АДМИН)) {  
 adminRadioButton.setChecked(true);  
 } else {  
 customerRadioButton.setChecked(true);  
 }  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(requireContext());  
 builder.setView(dialogView)  
 .setPositiveButton("Сохранить", (dialog, which) -> {  
 UserRole newRole = adminRadioButton.isChecked() ?   
 UserRole.АДМИН : UserRole.ПОЛЬЗОВАТЕЛЬ;  
  
   
 if (user.getRole().equals(newRole)) {  
 return;  
 }  
  
   
 User updatedUser = new User(  
 user.getId(),  
 user.getName(),  
 user.getEmail(),  
 user.getPassword(),  
 newRole  
 );  
  
   
 new Thread(() -> {  
 boolean success = dbHelper.updateUser(updatedUser);  
 requireActivity().runOnUiThread(() -> {  
 if (success) {  
 Toast.makeText(requireContext(), "Роль пользователя изменена", Toast.LENGTH\_SHORT).show();  
 loadUsers();   
 } else {  
 Toast.makeText(requireContext(), "Ошибка при изменении роли", Toast.LENGTH\_SHORT).show();  
 }  
 });  
 }).start();  
 })  
 .setNegativeButton("Отмена", null)  
 .show();  
 }  
  
 private void showDeleteUserDialog(User user) {  
 new AlertDialog.Builder(requireContext())  
 .setTitle("Удаление пользователя")  
 .setMessage("Вы уверены, что хотите удалить пользователя " + user.getName() + "?")  
 .setPositiveButton("Удалить", (dialog, which) -> {  
   
 new Thread(() -> {  
 boolean success = dbHelper.deleteUser(user.getEmail());  
 requireActivity().runOnUiThread(() -> {  
 if (success) {  
 Toast.makeText(requireContext(), "Пользователь удален", Toast.LENGTH\_SHORT).show();  
 loadUsers();   
 } else {  
 Toast.makeText(requireContext(), "Ошибка при удалении пользователя", Toast.LENGTH\_SHORT).show();  
 }  
 });  
 }).start();  
 })  
 .setNegativeButton("Отмена", null)  
 .show();  
 }  
  
 private void showAddUserDialog() {  
   
 View dialogView = LayoutInflater.from(requireContext()).inflate(R.layout.dialog\_add\_user, null);  
 TextView nameEditText = dialogView.findViewById(R.id.nameEditText);  
 TextView emailEditText = dialogView.findViewById(R.id.emailEditText);  
 TextView passwordEditText = dialogView.findViewById(R.id.passwordEditText);  
 RadioGroup roleRadioGroup = dialogView.findViewById(R.id.roleRadioGroup);  
 RadioButton adminRadioButton = dialogView.findViewById(R.id.adminRadioButton);  
  
 new AlertDialog.Builder(requireContext())  
 .setTitle("Добавить пользователя")  
 .setView(dialogView)  
 .setPositiveButton("Добавить", (dialog, which) -> {  
 String name = nameEditText.getText().toString().trim();  
 String email = emailEditText.getText().toString().trim();  
 String password = passwordEditText.getText().toString().trim();  
 UserRole role = adminRadioButton.isChecked() ?   
 UserRole.АДМИН : UserRole.ПОЛЬЗОВАТЕЛЬ;  
  
   
 if (name.isEmpty() || email.isEmpty() || password.isEmpty()) {  
 Toast.makeText(requireContext(), "Все поля должны быть заполнены", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
  
   
 new Thread(() -> {  
 User newUser = dbHelper.registerUser(name, email, password, role);  
 requireActivity().runOnUiThread(() -> {  
 if (newUser != null) {  
 Toast.makeText(requireContext(), "Пользователь добавлен", Toast.LENGTH\_SHORT).show();  
 loadUsers();   
 } else {  
 Toast.makeText(requireContext(), "Ошибка при добавлении пользователя. Возможно, пользователь с таким email уже существует.", Toast.LENGTH\_SHORT).show();  
 }  
 });  
 }).start();  
 })  
 .setNegativeButton("Отмена", null)  
 .show();  
 }  
}

package com.example.upavlika.model;  
  
import android.content.Context;  
import android.util.Log;  
import java.math.BigDecimal;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
import com.example.upavlika.data.DatabaseHelper;  
import com.example.upavlika.util.UserManager;  
  
public class Cart {  
 private static Cart instance;  
 private Map<Long, CartItem> items;  
 private BigDecimal totalPrice;  
 private Context context;  
 private DatabaseHelper dbHelper;  
  
 private Cart() {  
 items = new HashMap<>();  
 totalPrice = BigDecimal.ZERO;  
 }  
  
 public static synchronized Cart getInstance() {  
 if (instance == null) {  
 instance = new Cart();  
 }  
 return instance;  
 }  
  
   
 public void initialize(Context context) {  
 this.context = context.getApplicationContext();  
 this.dbHelper = DatabaseHelper.getInstance(context);  
 }  
  
   
 public static void resetInstance() {  
 instance = null;  
 }  
  
 public List<CartItem> getItems() {  
 return new ArrayList<>(items.values());  
 }  
  
 public int getItemCount() {  
 return items.size();  
 }  
  
 public BigDecimal getTotalPrice() {  
 return totalPrice;  
 }  
  
   
 public void addProduct(Product product, int quantity) {  
 try {  
 if (product == null) {  
 Log.e("Cart", "Попытка добавить null-продукт в корзину");  
 return;  
 }  
   
 Long productId = product.getId();  
   
 if (items.containsKey(productId)) {  
   
 CartItem item = items.get(productId);  
 item.setQuantity(item.getQuantity() + quantity);  
 } else {  
   
 CartItem newItem = new CartItem(product, quantity);  
 items.put(productId, newItem);  
 }  
   
 updateTotalPrice();  
 } catch (Exception e) {  
 Log.e("Cart", "Ошибка при добавлении товара в корзину: " + e.getMessage());  
 }  
 }  
  
   
 public void removeProduct(long productId) {  
 try {  
 if (items.containsKey(productId)) {  
 items.remove(productId);  
 updateTotalPrice();  
 }  
 } catch (Exception e) {  
 Log.e("Cart", "Ошибка при удалении товара из корзины: " + e.getMessage());  
 }  
 }  
  
   
 public void updateQuantity(long productId, int newQuantity) {  
 try {  
 if (items.containsKey(productId) && newQuantity > 0) {  
 CartItem item = items.get(productId);  
 item.setQuantity(newQuantity);  
 updateTotalPrice();  
 }  
 } catch (Exception e) {  
 Log.e("Cart", "Ошибка при обновлении количества товара: " + e.getMessage());  
 }  
 }  
  
   
 public void clear() {  
 try {  
 items.clear();  
 totalPrice = BigDecimal.ZERO;  
 } catch (Exception e) {  
 Log.e("Cart", "Ошибка при очистке корзины: " + e.getMessage());  
 }  
 }  
  
   
 private void updateTotalPrice() {  
 try {  
 totalPrice = BigDecimal.ZERO;  
 for (CartItem item : items.values()) {  
 if (item != null && item.getProduct() != null && item.getProduct().getPrice() != null) {  
   
 totalPrice = totalPrice.add(BigDecimal.valueOf(item.getTotal()));  
 }  
 }  
 } catch (Exception e) {  
 Log.e("Cart", "Ошибка при обновлении общей стоимости: " + e.getMessage());  
 }  
 }  
   
   
 public long createOrder(boolean isPaid) {  
 if (context == null) {  
 Log.e("Cart", "Cart not initialized. Call initialize(Context) first.");  
 throw new IllegalStateException("Cart not initialized. Call initialize(Context) first.");  
 }  
   
 if (items == null || items.isEmpty()) {  
 Log.e("Cart", "Attempt to create order with empty cart");  
 return -1;  
 }  
   
 User currentUser = UserManager.getInstance(context).getCurrentUser();  
 if (currentUser == null) {  
 Log.e("Cart", "No authenticated user found");  
 return -1;  
 }  
   
 try {  
   
 double totalAmount = totalPrice.doubleValue();  
 String orderDate = new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss", java.util.Locale.getDefault())  
 .format(new java.util.Date());  
 String orderStatus = isPaid ? "Оплачен" : "Ожидает оплаты";  
 String orderNumber = "ORD-" + System.currentTimeMillis() % 10000;  
   
   
 ArrayList<OrderItem> orderItems = new ArrayList<>();  
 for (CartItem cartItem : items.values()) {  
 if (cartItem == null || cartItem.getProduct() == null) {  
 Log.e("Cart", "Invalid cart item encountered");  
 continue;  
 }  
   
 Product product = cartItem.getProduct();  
 OrderItem orderItem = new OrderItem(  
 0,   
 0,   
 product.getId(),  
 cartItem.getQuantity(),  
 product.getPrice().doubleValue(),   
 product.getName()  
 );  
 orderItems.add(orderItem);  
 }  
   
 if (orderItems.isEmpty()) {  
 Log.e("Cart", "No valid items in cart to create order");  
 return -1;  
 }  
   
   
 Order order = new Order(  
 0,   
 currentUser.getId(),  
 orderDate,  
 orderStatus,  
 totalAmount,  
 orderNumber,  
 orderItems,  
 isPaid  
 );  
   
   
 try {  
 if (dbHelper != null) {  
 com.example.upavlika.model.UserProfile profile = dbHelper.getUserProfile(currentUser.getId());  
 if (profile != null && profile.getAddress() != null && !profile.getAddress().isEmpty()) {  
   
 StringBuilder fullAddress = new StringBuilder();  
   
 if (profile.getCity() != null && !profile.getCity().isEmpty()) {  
 fullAddress.append(profile.getCity());  
 }  
   
 if (profile.getAddress() != null && !profile.getAddress().isEmpty()) {  
 if (fullAddress.length() > 0) {  
 fullAddress.append(", ");  
 }  
 fullAddress.append(profile.getAddress());  
 }  
   
 if (profile.getPostalCode() != null && !profile.getPostalCode().isEmpty()) {  
 if (fullAddress.length() > 0) {  
 fullAddress.append(", ");  
 }  
 fullAddress.append(profile.getPostalCode());  
 }  
   
 order.setDeliveryAddress(fullAddress.toString());  
 }  
 }  
 } catch (Exception e) {  
 Log.e("Cart", "Error while setting delivery address: " + e.getMessage());  
   
 }  
   
 long orderId = dbHelper.createOrder(order, orderItems);  
 if (orderId != -1) {  
 clear();   
 Log.d("Cart", "Order created successfully with ID: " + orderId);  
 } else {  
 Log.e("Cart", "Failed to create order in database");  
 }  
   
 return orderId;  
 } catch (Exception e) {  
 Log.e("Cart", "Error creating order: " + e.getMessage(), e);  
 return -1;  
 }  
 }  
   
   
 public long payWithBalance(long userId) {  
 if (context == null) {  
 Log.e("Cart", "Cart not initialized. Call initialize(Context) first.");  
 throw new IllegalStateException("Cart not initialized. Call initialize(Context) first.");  
 }  
   
 if (dbHelper == null) {  
 Log.e("Cart", "Database helper is null");  
 return -1;  
 }  
   
 try {  
 double total = totalPrice.doubleValue();  
 User user = dbHelper.getUserById(userId);  
   
 if (user == null) {  
 Log.e("Cart", "User not found with ID: " + userId);  
 return -1;  
 }  
   
   
 if (!user.canAfford(total)) {  
 Log.e("Cart", "Insufficient funds. Required: " + total + ", Available: " + user.getBalance());  
 return -1;  
 }  
   
   
 user.pay(total);  
 boolean balanceUpdated = dbHelper.updateUserBalance(userId, user.getBalance());  
   
 if (!balanceUpdated) {  
 Log.e("Cart", "Failed to update user balance");  
 return -1;  
 }  
   
   
 long orderId = createOrder(true);  
 if (orderId != -1) {  
 Log.d("Cart", "Order successfully paid with balance. Order ID: " + orderId);  
 }  
 return orderId;  
 } catch (Exception e) {  
 Log.e("Cart", "Error during payment with balance: " + e.getMessage(), e);  
 return -1;  
 }  
 }  
  
   
 public boolean isEmpty() {  
 return items == null || items.isEmpty();  
 }  
   
   
 public void removeItem(long productId) {  
 removeProduct(productId);  
 }  
}

package com.example.upavlika.model;  
  
import java.math.BigDecimal;  
  
  
public class CartItem {  
 private Product product;  
 private int quantity;  
  
   
 public CartItem(Product product, int quantity) {  
 this.product = product;  
 this.quantity = quantity;  
 }  
  
   
 public Product getProduct() {  
 return product;  
 }  
  
   
 public void setProduct(Product product) {  
 this.product = product;  
 }  
  
   
 public int getQuantity() {  
 return quantity;  
 }  
  
   
 public void setQuantity(int quantity) {  
 this.quantity = quantity;  
 }  
  
   
 public double getTotal() {  
 return product.getPrice().multiply(BigDecimal.valueOf(quantity)).doubleValue();  
 }  
   
   
 public BigDecimal getItemTotal() {  
 return product.getPrice().multiply(BigDecimal.valueOf(quantity));  
 }  
}

package com.example.upavlika.model;  
  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.List;  
  
  
public class Order implements Serializable {  
 private long id;  
 private long userId;  
 private String orderDate;  
 private String status;  
 private double totalAmount;  
 private String orderNumber;  
 private List<OrderItem> orderItems;  
 private String filePath;   
 private boolean isPaid;  
 private String deliveryAddress = "";   
 private String cancellationReason = "";   
  
   
 public Order() {  
 this.orderItems = new ArrayList<>();   
 this.isPaid = false;   
 }  
  
   
 public Order(long userId, String orderDate, String status) {  
 this.userId = userId;  
 this.orderDate = orderDate;  
 this.status = status;  
 this.orderItems = new ArrayList<>();   
 this.isPaid = false;   
 }  
  
   
 public Order(long id, long userId, String orderDate, String status, double totalAmount, String orderNumber, List<OrderItem> orderItems) {  
 this.id = id;  
 this.userId = userId;  
 this.orderDate = orderDate;  
 this.status = status;  
 this.totalAmount = totalAmount;  
 this.orderNumber = orderNumber;  
 this.orderItems = orderItems != null ? orderItems : new ArrayList<>();  
 this.isPaid = false;   
 }  
   
   
 public Order(long id, long userId, String orderDate, String status, double totalAmount, String orderNumber, List<OrderItem> orderItems, boolean isPaid) {  
 this.id = id;  
 this.userId = userId;  
 this.orderDate = orderDate;  
 this.status = status;  
 this.totalAmount = totalAmount;  
 this.orderNumber = orderNumber;  
 this.orderItems = orderItems != null ? orderItems : new ArrayList<>();  
 this.isPaid = isPaid;  
 }  
  
   
 public void addItem(OrderItem item) {  
 if (orderItems == null) {  
 orderItems = new ArrayList<>();  
 }  
 orderItems.add(item);  
   
 totalAmount += (item.getPrice() \* item.getQuantity());  
 }  
  
   
 public long getId() { return id; }  
 public void setId(long id) { this.id = id; }  
 public long getUserId() { return userId; }  
 public void setUserId(long userId) { this.userId = userId; }  
 public String getOrderDate() { return orderDate; }  
 public void setOrderDate(String orderDate) { this.orderDate = orderDate; }  
 public String getStatus() { return status; }  
 public void setStatus(String status) { this.status = status; }  
 public double getTotalAmount() { return totalAmount; }  
 public void setTotalAmount(double totalAmount) { this.totalAmount = totalAmount; }  
 public String getOrderNumber() { return orderNumber; }  
 public void setOrderNumber(String orderNumber) { this.orderNumber = orderNumber; }  
 public List<OrderItem> getOrderItems() { return orderItems; }  
 public void setOrderItems(List<OrderItem> orderItems) { this.orderItems = orderItems; }  
 public String getFilePath() { return filePath; }  
 public void setFilePath(String filePath) { this.filePath = filePath; }  
 public boolean isPaid() { return isPaid; }  
 public void setPaid(boolean paid) { this.isPaid = paid; }  
   
   
 public List<OrderItem> getItems() { return orderItems; }  
 public void setItems(List<OrderItem> items) { this.orderItems = items; }  
   
   
 public String getFormattedDate() { return orderDate; }  
   
 public OrderStatus getOrderStatus() {   
 if (status == null || status.isEmpty()) {  
 return OrderStatus.PROCESSING;  
 }  
   
   
 for (OrderStatus os : OrderStatus.values()) {  
 if (os.name().equalsIgnoreCase(status) ||   
 os.getDisplayName().equalsIgnoreCase(status)) {  
 return os;  
 }  
 }  
   
   
 if (status.toLowerCase().contains("обработ")) {  
 return OrderStatus.PROCESSING;  
 } else if (status.toLowerCase().contains("подтв")) {  
 return OrderStatus.PAID;  
 } else if (status.toLowerCase().contains("отправ")) {  
 return OrderStatus.SHIPPED;  
 } else if (status.toLowerCase().contains("достав")) {  
 return OrderStatus.DELIVERED;  
 } else if (status.toLowerCase().contains("заверш") || status.toLowerCase().contains("выполн")) {  
 return OrderStatus.COMPLETED;  
 } else if (status.toLowerCase().contains("отмен")) {  
 return OrderStatus.CANCELLED;  
 }  
   
   
 return OrderStatus.PROCESSING;  
 }  
   
 public String getFormattedTotal() { return String.format("%.2f", totalAmount); }  
 public String getDeliveryAddress() { return deliveryAddress; }  
 public void setDeliveryAddress(String address) { this.deliveryAddress = address; }  
 public String getCancellationReason() { return cancellationReason; }  
 public void setCancellationReason(String reason) { this.cancellationReason = reason; }  
}

package com.example.upavlika.model;  
  
  
public class OrderItem {  
 private long id;  
 private long orderId;  
 private long productId;  
 private int quantity;  
 private double price;  
 private String productName;  
   
   
 public OrderItem() {  
 this.id = 0;  
 this.orderId = 0;  
 this.productId = 0;  
 this.quantity = 0;  
 this.price = 0.0;  
 this.productName = "";  
 }  
  
   
 public OrderItem(long id, long orderId, long productId, int quantity, double price, String productName) {  
 this.id = id;  
 this.orderId = orderId;  
 this.productId = productId;  
 this.quantity = quantity;  
 this.price = price;  
 this.productName = productName;  
 }  
  
   
 public long getId() {  
 return id;  
 }  
  
   
 public void setId(long id) {  
 this.id = id;  
 }  
  
   
 public long getOrderId() {  
 return orderId;  
 }  
  
   
 public void setOrderId(long orderId) {  
 this.orderId = orderId;  
 }  
  
   
 public long getProductId() {  
 return productId;  
 }  
  
   
 public void setProductId(long productId) {  
 this.productId = productId;  
 }  
  
   
 public int getQuantity() {  
 return quantity;  
 }  
  
   
 public void setQuantity(int quantity) {  
 this.quantity = quantity;  
 }  
  
   
 public double getPrice() {  
 return price;  
 }  
  
   
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
   
 public String getProductName() {  
 return productName;  
 }  
  
   
 public void setProductName(String productName) {  
 this.productName = productName;  
 }  
  
   
 public double getTotal() {  
 return price \* quantity;  
 }  
}

package com.example.upavlika.model;  
  
  
public enum OrderStatus {  
 WAITING\_PAYMENT("Ожидает оплаты"),  
 PAID("Оплачен"),  
 PROCESSING("Обработка"),  
 SHIPPED("Отправлен"),  
 DELIVERED("Доставлен"),  
 COMPLETED("Получен"),  
 CANCELLED("Отменен");  
   
 private final String displayName;  
   
 OrderStatus(String displayName) {  
 this.displayName = displayName;  
 }  
   
 public String getDisplayName() {  
 return displayName;  
 }  
   
 @Override  
 public String toString() {  
 return displayName;  
 }  
}

package com.example.upavlika.model;  
  
import java.math.BigDecimal;  
  
public class Product {  
 private long id;  
 private String name;  
 private String description;  
 private String ingredients;  
 private String nutritionInfo;  
 private BigDecimal price;  
 private int stockQuantity;  
 private String imageUrl;  
 private String category;  
 private int imageResourceId;  
  
 public Product(long id, String name, String description, String ingredients,   
 String nutritionInfo, double price, int imageResourceId, String category) {  
 this.id = id;  
 this.name = name;  
 this.description = description;  
 this.ingredients = ingredients;  
 this.nutritionInfo = nutritionInfo;  
 this.price = BigDecimal.valueOf(price);  
 this.imageResourceId = imageResourceId;  
 this.category = category;  
 }  
  
 public long getId() {  
 return id;  
 }  
  
 public void setId(long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 public String getIngredients() {  
 return ingredients;  
 }  
  
 public void setIngredients(String ingredients) {  
 this.ingredients = ingredients;  
 }  
  
 public String getNutritionInfo() {  
 return nutritionInfo;  
 }  
  
 public void setNutritionInfo(String nutritionInfo) {  
 this.nutritionInfo = nutritionInfo;  
 }  
  
 public BigDecimal getPrice() {  
 return price;  
 }  
  
 public void setPrice(BigDecimal price) {  
 this.price = price;  
 }  
  
 public int getStockQuantity() {  
 return stockQuantity;  
 }  
  
 public void setStockQuantity(int stockQuantity) {  
 this.stockQuantity = stockQuantity;  
 }  
  
 public String getImageUrl() {  
 return imageUrl;  
 }  
  
 public void setImageUrl(String imageUrl) {  
 this.imageUrl = imageUrl;  
 }  
  
 public String getCategory() {  
 return category;  
 }  
  
 public void setCategory(String category) {  
 this.category = category;  
 }  
  
 public int getImageResourceId() {  
 return imageResourceId;  
 }  
  
 public void setImageResourceId(int imageResourceId) {  
 this.imageResourceId = imageResourceId;  
 }  
  
 @Override  
 public String toString() {  
 return "Product{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", price=" + price +  
 ", stockQuantity=" + stockQuantity +  
 ", category='" + category + '\'' +  
 '}';  
 }  
}

package com.example.upavlika.model;  
  
import java.io.Serializable;  
  
public class User implements Serializable {  
 private long id;  
 private String name;  
 private String email;  
 private String password;  
 private UserRole role;  
 private double balance;   
  
 public User() {  
 }  
  
 public User(String name, String email, String password, UserRole role) {  
 this.name = name;  
 this.email = email;  
 this.password = password;  
 this.role = role;  
 this.balance = 0.0;   
 }  
  
 public User(long id, String name, String email, String password, UserRole role) {  
 this.id = id;  
 this.name = name;  
 this.email = email;  
 this.password = password;  
 this.role = role;  
 this.balance = 0.0;   
 }  
   
 public User(long id, String name, String email, String password, UserRole role, double balance) {  
 this.id = id;  
 this.name = name;  
 this.email = email;  
 this.password = password;  
 this.role = role;  
 this.balance = balance;  
 }  
  
 public long getId() {  
 return id;  
 }  
  
 public void setId(long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public UserRole getRole() {  
 return role;  
 }  
   
 public double getBalance() {  
 return balance;  
 }  
   
 public void setBalance(double balance) {  
 this.balance = balance;  
 }  
   
 public boolean canAfford(double amount) {  
 return this.balance >= amount;  
 }  
   
 public boolean deductBalance(double amount) {  
 if (canAfford(amount)) {  
 this.balance -= amount;  
 return true;  
 }  
 return false;  
 }  
   
 public void addToBalance(double amount) {  
 if (amount > 0) {  
 this.balance += amount;  
 }  
 }  
  
 public boolean pay(double amount) {  
 if (canAfford(amount)) {  
 this.balance -= amount;  
 return true;  
 }  
 return false;  
 }  
  
 @Override  
 public String toString() {  
 return "User{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", email='" + email + '\'' +  
 ", role=" + role +  
 ", balance=" + balance +  
 '}';  
 }  
}

package com.example.upavlika.model;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class UserProfile {  
 private String name;  
 private String email;  
 private String phone;  
 private String address;  
 private String city;  
 private String postalCode;  
 private List<Long> orderIds;  
  
 public UserProfile(String name, String email) {  
 this.name = name;  
 this.email = email;  
 this.phone = "";  
 this.address = "";  
 this.city = "";  
 this.postalCode = "";  
 this.orderIds = new ArrayList<>();  
 }  
  
 public UserProfile(String name, String email, String phone, String address, String city, String postalCode) {  
 this.name = name;  
 this.email = email;  
 this.phone = phone;  
 this.address = address;  
 this.city = city;  
 this.postalCode = postalCode;  
 this.orderIds = new ArrayList<>();  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getPhone() {  
 return phone;  
 }  
  
 public void setPhone(String phone) {  
 this.phone = phone;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public String getCity() {  
 return city;  
 }  
  
 public void setCity(String city) {  
 this.city = city;  
 }  
  
 public String getPostalCode() {  
 return postalCode;  
 }  
  
 public void setPostalCode(String postalCode) {  
 this.postalCode = postalCode;  
 }  
  
 public List<Long> getOrderIds() {  
 return orderIds;  
 }  
  
 public void addOrder(long orderId) {  
 this.orderIds.add(orderId);  
 }  
  
 public boolean hasCompletedProfile() {  
 return !phone.isEmpty() && !address.isEmpty() && !city.isEmpty() && !postalCode.isEmpty();  
 }  
}

package com.example.upavlika.model;  
  
public enum UserRole {  
 АДМИН("Администратор"),  
 ПОЛЬЗОВАТЕЛЬ("Покупатель");  
  
 private final String displayName;  
  
 UserRole(String displayName) {  
 this.displayName = displayName;  
 }  
  
 public String getDisplayName() {  
 return displayName;  
 }  
}

package com.example.upavlika.service;  
  
import android.content.BroadcastReceiver;  
import android.content.Context;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.os.Build;  
import android.util.Log;  
  
import java.util.Map;  
  
  
public class BootReceiver extends BroadcastReceiver {  
 private static final String TAG = "BootReceiver";  
  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 if (intent.getAction() == null) return;  
   
 if (Intent.ACTION\_BOOT\_COMPLETED.equals(intent.getAction()) ||  
 "android.intent.action.QUICKBOOT\_POWERON".equals(intent.getAction())) {  
   
 Log.d(TAG, "Устройство загружено, проверяем активные доставки");  
   
   
 SharedPreferences prefs = context.getSharedPreferences("delivery\_prefs", Context.MODE\_PRIVATE);  
 Map<String, ?> allPrefs = prefs.getAll();  
   
 if (allPrefs.isEmpty()) {  
 Log.d(TAG, "Нет активных доставок для восстановления");  
 return;  
 }  
   
   
 for (String key : allPrefs.keySet()) {  
 if (key.startsWith("order\_") && key.endsWith("\_start\_time")) {  
 try {  
   
 String orderId = key.substring(6, key.lastIndexOf("\_start\_time"));  
 long orderIdLong = Long.parseLong(orderId);  
   
   
 long startTime = prefs.getLong(key, 0);  
 long totalTime = prefs.getLong("order\_" + orderId + "\_total\_time", 0);  
   
 if (startTime > 0 && totalTime > 0) {  
   
 long elapsed = System.currentTimeMillis() - startTime;  
   
   
 if (elapsed < totalTime) {  
   
 long remainingTime = totalTime - elapsed;  
   
   
 restoreDeliveryTracking(context, orderIdLong, remainingTime, startTime);  
   
 Log.d(TAG, "Восстановлено отслеживание для заказа " + orderId +   
 ", осталось " + (remainingTime / 1000) + " секунд");  
 } else {  
   
   
 markDeliveryAsCompleted(context, orderIdLong);  
   
   
 prefs.edit()  
 .remove(key)  
 .remove("order\_" + orderId + "\_total\_time")  
 .apply();  
   
 Log.d(TAG, "Доставка для заказа " + orderId +   
 " завершилась во время выключения устройства");  
 }  
 }  
 } catch (Exception e) {  
 Log.e(TAG, "Ошибка при обработке информации о доставке: " + e.getMessage(), e);  
 }  
 }  
 }  
 }  
 }  
   
   
 private void restoreDeliveryTracking(Context context, long orderId, long remainingTime, long startTime) {  
 try {  
   
 Intent serviceIntent = new Intent(context, DeliveryTrackerService.class);  
 serviceIntent.setAction(DeliveryTrackerService.ACTION\_START\_TRACKING);  
 serviceIntent.putExtra(DeliveryTrackerService.EXTRA\_ORDER\_ID, orderId);  
 serviceIntent.putExtra(DeliveryTrackerService.EXTRA\_TOTAL\_TIME, remainingTime);  
 serviceIntent.putExtra(DeliveryTrackerService.EXTRA\_START\_TIME, startTime);  
   
   
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {  
 context.startForegroundService(serviceIntent);  
 } else {  
 context.startService(serviceIntent);  
 }  
 } catch (Exception e) {  
 Log.e(TAG, "Ошибка при восстановлении отслеживания доставки: " + e.getMessage(), e);  
 }  
 }  
   
   
 private void markDeliveryAsCompleted(Context context, long orderId) {  
 try {  
   
 Intent notificationIntent = new Intent(context, DeliveryNotificationReceiver.class);  
 notificationIntent.putExtra(DeliveryTrackerService.EXTRA\_ORDER\_ID, orderId);  
   
   
 context.sendBroadcast(notificationIntent);  
 } catch (Exception e) {  
 Log.e(TAG, "Ошибка при обновлении статуса заказа: " + e.getMessage(), e);  
 }  
 }  
}

package com.example.upavlika.service;  
  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.app.PendingIntent;  
import android.content.BroadcastReceiver;  
import android.content.Context;  
import android.content.Intent;  
import android.os.Build;  
import android.util.Log;  
  
import androidx.core.app.NotificationCompat;  
  
import com.example.upavlika.OrderHistoryActivity;  
import com.example.upavlika.R;  
import com.example.upavlika.data.DatabaseHelper;  
import com.example.upavlika.model.Order;  
  
  
public class DeliveryNotificationReceiver extends BroadcastReceiver {  
   
 private static final String TAG = "DeliveryNotifReceiver";  
 private static final String CHANNEL\_ID = "delivery\_notification\_channel";  
 private static final int NOTIFICATION\_ID = 2000;  
   
 @Override  
 public void onReceive(Context context, Intent intent) {  
 Log.d(TAG, "Получено уведомление о доставке");  
   
   
 long orderId = intent.getLongExtra(DeliveryTrackerService.EXTRA\_ORDER\_ID, -1);  
 if (orderId == -1) {  
 Log.e(TAG, "Неверный ID заказа");  
 return;  
 }  
   
   
 createNotificationChannel(context);  
   
   
 updateOrderStatusAndNotify(context, orderId);  
 }  
   
 private void createNotificationChannel(Context context) {  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {  
 CharSequence name = "Уведомления о доставке";  
 String description = "Канал для уведомлений о доставке заказов";  
 int importance = NotificationManager.IMPORTANCE\_HIGH;  
 NotificationChannel channel = new NotificationChannel(CHANNEL\_ID, name, importance);  
 channel.setDescription(description);  
   
 NotificationManager notificationManager =   
 context.getSystemService(NotificationManager.class);  
   
 if (notificationManager != null) {  
 notificationManager.createNotificationChannel(channel);  
 }  
 }  
 }  
   
 private void updateOrderStatusAndNotify(Context context, long orderId) {  
 try {  
   
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(context);  
 dbHelper.updateOrderStatus(orderId, "Доставлен");  
   
   
 Order order = dbHelper.getOrder(orderId);  
 if (order == null) {  
 Log.e(TAG, "Не удалось найти заказ с ID " + orderId);  
 return;  
 }  
   
   
 Intent notificationIntent = new Intent(context, OrderHistoryActivity.class);  
 PendingIntent pendingIntent = PendingIntent.getActivity(  
 context,   
 0,   
 notificationIntent,   
 PendingIntent.FLAG\_UPDATE\_CURRENT | PendingIntent.FLAG\_IMMUTABLE);  
   
   
 NotificationCompat.Builder builder = new NotificationCompat.Builder(context, CHANNEL\_ID)  
 .setContentTitle("Заказ доставлен!")  
 .setContentText("Ваш заказ №" + order.getOrderNumber() + " успешно доставлен!")  
 .setSmallIcon(android.R.drawable.ic\_dialog\_info)  
 .setContentIntent(pendingIntent)  
 .setPriority(NotificationCompat.PRIORITY\_HIGH)  
 .setAutoCancel(true);  
   
   
 NotificationManager notificationManager =   
 (NotificationManager) context.getSystemService(Context.NOTIFICATION\_SERVICE);  
   
 if (notificationManager != null) {  
 notificationManager.notify(NOTIFICATION\_ID, builder.build());  
 }  
   
 Log.d(TAG, "Отправлено уведомление о доставке заказа " + orderId);  
 } catch (Exception e) {  
 Log.e(TAG, "Ошибка при обновлении статуса заказа и отправке уведомления: " + e.getMessage());  
 }  
 }  
}

package com.example.upavlika.service;  
  
import android.app.AlarmManager;  
import android.app.Notification;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.app.PendingIntent;  
import android.app.Service;  
import android.content.Context;  
import android.content.Intent;  
import android.os.Build;  
import android.os.Handler;  
import android.os.IBinder;  
import android.os.Looper;  
import android.os.SystemClock;  
import android.util.Log;  
  
import androidx.annotation.Nullable;  
import androidx.core.app.NotificationCompat;  
  
import com.example.upavlika.OrderHistoryActivity;  
import com.example.upavlika.R;  
import com.example.upavlika.data.DatabaseHelper;  
import com.example.upavlika.model.Order;  
  
import java.util.Timer;  
import java.util.TimerTask;  
  
  
public class DeliveryTrackerService extends Service {  
  
 private static final String TAG = "DeliveryTrackerService";  
 private static final String CHANNEL\_ID = "delivery\_tracking\_channel";  
 private static final int NOTIFICATION\_ID = 1001;  
 private static final int DELIVERY\_NOTIFICATION\_ID = 1002;  
 private static final int SHIPPED\_NOTIFICATION\_ID = 1003;  
   
   
 public static final String EXTRA\_ORDER\_ID = "order\_id";  
 public static final String EXTRA\_TOTAL\_TIME = "total\_time";  
 public static final String EXTRA\_START\_TIME = "start\_time";   
 public static final String ACTION\_START\_TRACKING = "com.example.upavlika.START\_TRACKING";  
 public static final String ACTION\_STOP\_TRACKING = "com.example.upavlika.STOP\_TRACKING";  
 public static final String ACTION\_CHECK\_STATUS = "com.example.upavlika.CHECK\_STATUS";   
   
   
 private static final long DEFAULT\_DELIVERY\_TIME = 60 \* 1000;  
   
 private Timer timer;  
 private long orderId;  
 private long totalTime;  
 private long startTime;  
 private boolean isDelivered = false;  
 private DatabaseHelper dbHelper;  
  
 @Override  
 public void onCreate() {  
 super.onCreate();  
 Log.d(TAG, "Сервис создан");  
 dbHelper = DatabaseHelper.getInstance(this);  
   
   
 createNotificationChannel();  
 }  
  
 @Override  
 public int onStartCommand(Intent intent, int flags, int startId) {  
 if (intent == null) {  
 Log.e(TAG, "Intent is null, stopping service");  
 stopSelf();  
 return START\_NOT\_STICKY;  
 }  
   
 String action = intent.getAction();  
   
   
   
 Notification notification = createTrackingNotification("Подготовка отслеживания доставки...");  
 startForeground(NOTIFICATION\_ID, notification);  
   
   
 if (action == null) {  
   
 handleStartCommand(intent);  
 return START\_STICKY;  
 }  
   
 Log.d(TAG, "Получено действие: " + action);  
   
 if (ACTION\_START\_TRACKING.equals(action)) {  
 handleStartCommand(intent);  
 } else if (ACTION\_STOP\_TRACKING.equals(action)) {  
   
 stopTracking();  
 } else if (ACTION\_CHECK\_STATUS.equals(action)) {  
   
 checkOrderStatus();  
 }  
   
 return START\_STICKY;  
 }  
   
   
 private void handleStartCommand(Intent intent) {  
 try {  
   
 orderId = intent.getLongExtra(EXTRA\_ORDER\_ID, -1);  
 totalTime = intent.getLongExtra(EXTRA\_TOTAL\_TIME, DEFAULT\_DELIVERY\_TIME);  
   
 if (orderId == -1) {  
 Log.e(TAG, "Некорректный ID заказа, останавливаем сервис");  
 stopSelf();  
 return;  
 }  
   
   
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(this);  
 if (dbHelper == null) {  
 Log.e(TAG, "Не удалось инициализировать базу данных");  
 stopSelf();  
 return;  
 }  
   
 Order order = dbHelper.getOrder(orderId);  
 if (order == null) {  
 Log.e(TAG, "Заказ не найден");  
 stopSelf();  
 return;  
 }  
   
   
 if (!"Отправлен".equals(order.getStatus()) && !"В пути".equals(order.getStatus()) && !"Обработан".equals(order.getStatus())) {  
 Log.w(TAG, "Предупреждение: отслеживание доставки для заказа со статусом '" + order.getStatus() + "'");  
 }  
   
   
 startTime = System.currentTimeMillis();  
   
   
 Notification updatedNotification = createTrackingNotification(  
 "Отслеживание заказа #" + order.getOrderNumber());  
 NotificationManager notificationManager =   
 (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);  
 if (notificationManager != null) {  
 notificationManager.notify(NOTIFICATION\_ID, updatedNotification);  
 }  
   
   
 startDeliveryTimer();  
   
   
 scheduleDeliveryEvent();  
   
   
 getSharedPreferences("delivery\_prefs", MODE\_PRIVATE)  
 .edit()  
 .putLong("order\_" + orderId + "\_start\_time", startTime)  
 .putLong("order\_" + orderId + "\_total\_time", totalTime)  
 .apply();  
   
 Log.d(TAG, "Отслеживание запущено для заказа " + orderId);  
 } catch (Exception e) {  
 Log.e(TAG, "Ошибка при обработке intent: " + e.getMessage(), e);  
 stopSelf();  
 }  
 }  
   
 private void startDeliveryTimer() {  
   
 if (startTime == 0) {  
 startTime = System.currentTimeMillis();  
 }  
   
 timer = new Timer();  
 timer.scheduleAtFixedRate(new TimerTask() {  
 @Override  
 public void run() {  
 long elapsed = System.currentTimeMillis() - startTime;  
 long remaining = totalTime - elapsed;  
   
 if (remaining <= 0) {  
   
 if (!isDelivered) {  
 isDelivered = true;  
 markOrderAsDelivered();  
 }  
   
 timer.cancel();  
 } else {  
   
 updateTrackingNotification(remaining);  
 }  
 }  
 }, 0, 1000);   
 }  
   
 private void scheduleDeliveryEvent() {  
   
 Intent deliveryIntent = new Intent(this, DeliveryNotificationReceiver.class);  
 deliveryIntent.putExtra(EXTRA\_ORDER\_ID, orderId);  
   
 PendingIntent pendingIntent = PendingIntent.getBroadcast(  
 this,   
 (int) orderId,   
 deliveryIntent,   
 PendingIntent.FLAG\_UPDATE\_CURRENT | PendingIntent.FLAG\_IMMUTABLE);  
   
   
 AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM\_SERVICE);  
   
   
 if (alarmManager != null) {  
   
 long deliveryTimeMillis = startTime + totalTime;  
   
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {  
 alarmManager.setExactAndAllowWhileIdle(  
 AlarmManager.RTC\_WAKEUP,  
 deliveryTimeMillis,  
 pendingIntent);  
 } else {  
 alarmManager.setExact(  
 AlarmManager.RTC\_WAKEUP,  
 deliveryTimeMillis,  
 pendingIntent);  
 }  
   
   
 scheduleStatusCheck();  
   
 Log.d(TAG, "Запланировано уведомление о доставке через " + (totalTime / 1000) + " секунд");  
 }  
 }  
   
   
 private void scheduleStatusCheck() {  
   
 Intent checkIntent = new Intent(this, DeliveryTrackerService.class);  
 checkIntent.setAction(ACTION\_CHECK\_STATUS);  
 checkIntent.putExtra(EXTRA\_ORDER\_ID, orderId);  
   
 PendingIntent pendingIntent = PendingIntent.getService(  
 this,  
 (int) (orderId + 1000),   
 checkIntent,  
 PendingIntent.FLAG\_UPDATE\_CURRENT | PendingIntent.FLAG\_IMMUTABLE);  
   
   
 AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM\_SERVICE);  
 if (alarmManager != null) {  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {  
 alarmManager.setRepeating(  
 AlarmManager.RTC\_WAKEUP,  
 System.currentTimeMillis() + 15000,   
 15000,   
 pendingIntent);  
 } else {  
 alarmManager.setRepeating(  
 AlarmManager.RTC\_WAKEUP,  
 System.currentTimeMillis() + 15000,  
 15000,  
 pendingIntent);  
 }  
 }  
 }  
   
   
 private void checkOrderStatus() {  
 try {  
 if (dbHelper == null) {  
 dbHelper = DatabaseHelper.getInstance(this);  
 }  
   
 if (dbHelper != null && orderId > 0) {  
 Order order = dbHelper.getOrder(orderId);  
 if (order != null) {  
 String status = order.getStatus();  
   
   
 if ("Доставлен".equals(status) || "Получен".equals(status)) {  
 if (!isDelivered) {  
 isDelivered = true;  
   
 if ("Доставлен".equals(status)) {  
 sendDeliveredNotification();  
 }  
 }  
 }  
   
   
 if ("Получен".equals(status)) {  
 stopSelf();  
 }  
 }  
 }  
 } catch (Exception e) {  
 Log.e(TAG, "Ошибка при проверке статуса заказа: " + e.getMessage(), e);  
 }  
 }  
   
 private void markOrderAsDelivered() {  
 try {  
   
 Order order = dbHelper.getOrder(orderId);  
 if (order == null) {  
 Log.e(TAG, "Заказ " + orderId + " не найден в базе данных");  
 return;  
 }  
   
   
 String currentStatus = order.getStatus();  
 if ("Доставлен".equals(currentStatus) || "Получен".equals(currentStatus)) {  
 Log.d(TAG, "Заказ " + orderId + " уже имеет статус '" + currentStatus + "', обновление не требуется");  
 return;  
 }  
   
   
 sendDeliveredNotification();  
   
   
 boolean updated = dbHelper.updateOrderStatus(orderId, "Доставлен");  
   
 if (updated) {  
 Log.d(TAG, "Заказ " + orderId + " отмечен как доставленный");  
   
   
 getSharedPreferences("delivery\_prefs", MODE\_PRIVATE)  
 .edit()  
 .remove("order\_" + orderId + "\_start\_time")  
 .remove("order\_" + orderId + "\_total\_time")  
 .apply();  
 } else {  
 Log.e(TAG, "Не удалось обновить статус заказа " + orderId);  
   
   
 new Handler(Looper.getMainLooper()).postDelayed(() -> {  
 boolean retryUpdate = dbHelper.updateOrderStatus(orderId, "Доставлен");  
 if (retryUpdate) {  
 Log.d(TAG, "Повторная попытка обновления статуса заказа " + orderId + " успешна");  
 } else {  
 Log.e(TAG, "Повторная попытка обновления статуса заказа " + orderId + " не удалась");  
 }  
 }, 5000);  
 }  
 } catch (Exception e) {  
 Log.e(TAG, "Ошибка при изменении статуса заказа: " + e.getMessage(), e);  
 }  
 }  
  
 private void stopTracking() {  
 if (timer != null) {  
 timer.cancel();  
 timer = null;  
 }  
   
   
 try {  
   
 Intent deliveryIntent = new Intent(this, DeliveryNotificationReceiver.class);  
 PendingIntent pendingIntent = PendingIntent.getBroadcast(  
 this,   
 (int) orderId,   
 deliveryIntent,   
 PendingIntent.FLAG\_UPDATE\_CURRENT | PendingIntent.FLAG\_IMMUTABLE);  
   
 AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM\_SERVICE);  
 if (alarmManager != null) {  
 alarmManager.cancel(pendingIntent);  
 }  
   
   
 Intent checkIntent = new Intent(this, DeliveryTrackerService.class);  
 checkIntent.setAction(ACTION\_CHECK\_STATUS);  
 PendingIntent checkPendingIntent = PendingIntent.getService(  
 this,  
 (int) (orderId + 1000),  
 checkIntent,  
 PendingIntent.FLAG\_UPDATE\_CURRENT | PendingIntent.FLAG\_IMMUTABLE);  
   
 if (alarmManager != null) {  
 alarmManager.cancel(checkPendingIntent);  
 }  
   
   
 getSharedPreferences("delivery\_prefs", MODE\_PRIVATE)  
 .edit()  
 .remove("order\_" + orderId + "\_start\_time")  
 .remove("order\_" + orderId + "\_total\_time")  
 .apply();  
   
 } catch (Exception e) {  
 Log.e(TAG, "Ошибка при отмене запланированных событий: " + e.getMessage(), e);  
 }  
   
   
 stopSelf();  
 }  
   
 private void createNotificationChannel() {  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {  
 CharSequence name = "Отслеживание доставки";  
 String description = "Канал для отслеживания доставки заказов";  
 int importance = NotificationManager.IMPORTANCE\_LOW;   
 NotificationChannel channel = new NotificationChannel(CHANNEL\_ID, name, importance);  
 channel.setDescription(description);  
   
 NotificationManager notificationManager = getSystemService(NotificationManager.class);  
 if (notificationManager != null) {  
 notificationManager.createNotificationChannel(channel);  
 }  
 }  
 }  
   
 private Notification createTrackingNotification(String message) {  
 Intent notificationIntent = new Intent(this, OrderHistoryActivity.class);  
 PendingIntent pendingIntent;  
   
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {  
 pendingIntent = PendingIntent.getActivity(this, 0, notificationIntent,   
 PendingIntent.FLAG\_IMMUTABLE | PendingIntent.FLAG\_UPDATE\_CURRENT);  
 } else {  
 pendingIntent = PendingIntent.getActivity(this, 0, notificationIntent,   
 PendingIntent.FLAG\_UPDATE\_CURRENT);  
 }  
   
 NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL\_ID)  
 .setContentTitle("Доставка заказа")  
 .setContentText(message)  
 .setSmallIcon(R.drawable.ic\_launcher\_foreground)  
 .setPriority(NotificationCompat.PRIORITY\_HIGH)  
 .setContentIntent(pendingIntent)  
 .setOngoing(true);  
   
 return builder.build();  
 }  
   
 private void updateTrackingNotification(long remainingMillis) {  
 long remainingSeconds = remainingMillis / 1000;  
 long minutes = remainingSeconds / 60;  
 long seconds = remainingSeconds % 60;  
   
 String message = String.format("Ожидаемое время доставки: %02d:%02d", minutes, seconds);  
   
 NotificationManager notificationManager =   
 (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);  
   
 if (notificationManager != null) {  
 Notification notification = createTrackingNotification(message);  
 notificationManager.notify(NOTIFICATION\_ID, notification);  
 }  
 }  
   
 private void sendDeliveredNotification() {  
 try {  
   
 Order order = dbHelper.getOrder(orderId);  
 if (order == null) {  
 Log.e(TAG, "Не удалось найти заказ с ID " + orderId);  
 return;  
 }  
   
   
 Intent notificationIntent = new Intent(this, OrderHistoryActivity.class);  
 notificationIntent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP);  
   
 PendingIntent pendingIntent;  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.S) {  
 pendingIntent = PendingIntent.getActivity(  
 this, 0, notificationIntent,   
 PendingIntent.FLAG\_UPDATE\_CURRENT | PendingIntent.FLAG\_IMMUTABLE);  
 } else {  
 pendingIntent = PendingIntent.getActivity(  
 this, 0, notificationIntent,   
 PendingIntent.FLAG\_UPDATE\_CURRENT);  
 }  
   
   
 NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL\_ID)  
 .setContentTitle("Заказ доставлен!")  
 .setContentText("Ваш заказ №" + order.getOrderNumber() + " успешно доставлен!")  
 .setSmallIcon(android.R.drawable.ic\_dialog\_info)  
 .setContentIntent(pendingIntent)  
 .setPriority(NotificationCompat.PRIORITY\_HIGH)  
 .setAutoCancel(true);  
   
   
 NotificationManager notificationManager =   
 (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);  
   
 if (notificationManager != null) {  
 notificationManager.notify(DELIVERY\_NOTIFICATION\_ID, builder.build());  
 }  
   
 Log.d(TAG, "Отправлено уведомление о доставке заказа " + orderId);  
 } catch (Exception e) {  
 Log.e(TAG, "Ошибка при отправке уведомления о доставке: " + e.getMessage(), e);  
 }  
 }  
  
 @Nullable  
 @Override  
 public IBinder onBind(Intent intent) {  
 return null;  
 }  
   
 @Override  
 public void onDestroy() {  
 super.onDestroy();  
   
 if (timer != null) {  
 timer.cancel();  
 timer = null;  
 }  
   
 Log.d(TAG, "Сервис уничтожен");  
 }  
}

package com.example.upavlika;  
  
import android.os.Bundle;  
import android.view.MenuItem;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
import androidx.fragment.app.Fragment;  
import androidx.fragment.app.FragmentManager;  
import androidx.lifecycle.Lifecycle;  
import androidx.viewpager2.adapter.FragmentStateAdapter;  
import androidx.viewpager2.widget.ViewPager2;  
  
import com.example.upavlika.fragments.ManageProductsFragment;  
import com.example.upavlika.fragments.ManageUsersFragment;  
import com.google.android.material.tabs.TabLayout;  
import com.google.android.material.tabs.TabLayoutMediator;  
  
public class AdminPanelActivity extends AppCompatActivity {  
  
 private TabLayout tabLayout;  
 private ViewPager2 viewPager;  
 private Toolbar toolbar;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_admin\_panel);  
  
   
 toolbar = findViewById(R.id.toolbar);  
 tabLayout = findViewById(R.id.tabLayout);  
 viewPager = findViewById(R.id.viewPager);  
  
   
 setSupportActionBar(toolbar);  
 if (getSupportActionBar() != null) {  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setTitle("Панель администратора");  
 }  
  
   
 AdminPagerAdapter pagerAdapter = new AdminPagerAdapter(getSupportFragmentManager(), getLifecycle());  
 viewPager.setAdapter(pagerAdapter);  
  
   
 new TabLayoutMediator(tabLayout, viewPager, (tab, position) -> {  
 switch (position) {  
 case 0:  
 tab.setText("Товары");  
 break;  
 case 1:  
 tab.setText("Пользователи");  
 break;  
 }  
 }).attach();  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
 if (item.getItemId() == android.R.id.home) {  
 onBackPressed();  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
  
   
 private static class AdminPagerAdapter extends FragmentStateAdapter {  
  
 public AdminPagerAdapter(@NonNull FragmentManager fragmentManager, @NonNull Lifecycle lifecycle) {  
 super(fragmentManager, lifecycle);  
 }  
  
 @NonNull  
 @Override  
 public Fragment createFragment(int position) {  
   
 if (position == 0) {  
 return new ManageProductsFragment();  
 } else {  
 return new ManageUsersFragment();  
 }  
 }  
  
 @Override  
 public int getItemCount() {  
 return 2;   
 }  
 }  
}

package com.example.upavlika;  
  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.os.Build;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.upavlika.model.Cart;  
import com.example.upavlika.model.CartItem;  
import com.example.upavlika.adapter.CartAdapter;  
import com.example.upavlika.model.Order;  
import com.example.upavlika.model.OrderItem;  
import com.example.upavlika.model.Product;  
import com.example.upavlika.model.User;  
import com.example.upavlika.util.UserManager;  
import com.example.upavlika.data.DatabaseHelper;  
import com.example.upavlika.util.OrderFileManager;  
import android.Manifest;  
import android.content.pm.PackageManager;  
import androidx.core.app.ActivityCompat;  
import androidx.core.content.ContextCompat;  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AlertDialog;  
  
import java.math.BigDecimal;  
import java.text.NumberFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
import java.util.Locale;  
import java.io.File;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import android.media.MediaScannerConnection;  
import android.os.Environment;  
import android.os.AsyncTask;  
import java.text.SimpleDateFormat;  
import android.app.ProgressDialog;  
import android.os.Handler;  
import android.provider.Settings;  
import android.net.Uri;  
import android.content.ContentResolver;  
import android.content.ContentValues;  
import android.provider.MediaStore;  
import java.io.OutputStream;  
import android.widget.ProgressBar;  
import android.view.MenuItem;  
import android.os.Looper;  
import android.app.Notification;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.app.PendingIntent;  
import androidx.core.app.NotificationCompat;  
import androidx.core.app.NotificationManagerCompat;  
import android.widget.Toast;  
import android.util.Log;  
  
public class CartActivity extends AppCompatActivity implements CartAdapter.CartItemListener {  
  
 private RecyclerView recyclerView;  
 private TextView totalPriceTextView;  
 private TextView emptyCartTextView;  
 private Button checkoutButton;  
 private Cart cart;  
 private CartAdapter adapter;  
   
   
 private static final double DELIVERY\_FEE = 299.0;   
 private static final double FREE\_DELIVERY\_THRESHOLD = 3000.0;   
 private static final int STORAGE\_PERMISSION\_CODE = 101;  
 private static final int WRITE\_PERMISSION\_CODE = 102;  
 private ProgressDialog progressDialog;  
 private UserManager userManager;  
 private DatabaseHelper dbHelper;  
 private List<CartItem> cartItems;  
 private RecyclerView cartRecyclerView;  
 private ProgressBar progressBar;  
 private User currentUser;  
 private double totalPrice = 0.0;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 try {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_cart);  
   
   
 userManager = UserManager.getInstance(this);  
 dbHelper = DatabaseHelper.getInstance(this);  
   
   
 long receiptOrderId = getIntent().getLongExtra("DOWNLOAD\_RECEIPT\_FOR\_ORDER\_ID", -1);  
 if (receiptOrderId != -1) {  
   
 downloadReceiptForOrder(receiptOrderId);  
 return;   
 }  
   
   
 try {  
 cart = Cart.getInstance();  
 if (cart != null) {  
   
 cart.initialize(this);  
 } else {  
 finish();  
 return;  
 }  
 } catch (Exception e) {  
 finish();  
 return;  
 }  
   
   
 try {  
 currentUser = userManager.getCurrentUser();  
 if (currentUser == null) {  
 startActivity(new Intent(this, LoginActivity.class));  
 finish();  
 return;  
 }  
 } catch (Exception e) {  
 finish();  
 return;  
 }  
   
   
 try {  
 Toolbar toolbar = findViewById(R.id.cartToolbar);  
 setSupportActionBar(toolbar);  
 if (getSupportActionBar() != null) {  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setDisplayShowHomeEnabled(true);  
 getSupportActionBar().setTitle("Корзина");  
 }  
 } catch (Exception e) {  
   
 }  
   
   
 try {  
 initViews();  
 } catch (Exception e) {  
 finish();  
 return;  
 }  
   
   
 try {  
 setupRecyclerView();  
 } catch (Exception e) {  
   
 }  
   
   
 try {  
 updateUI();  
 } catch (Exception e) {  
   
 }  
   
   
 try {  
 checkoutButton.setOnClickListener(v -> processCheckout());  
 } catch (Exception e) {  
   
 }  
   
   
 try {  
 progressDialog = new ProgressDialog(this);  
 progressDialog.setMessage("Обработка заказа...");  
 progressDialog.setCancelable(false);  
 } catch (Exception e) {  
   
 }  
   
 } catch (Exception e) {  
 finish();  
 }  
 }  
   
 private void initViews() {  
 recyclerView = findViewById(R.id.cart\_recycler\_view);  
 totalPriceTextView = findViewById(R.id.total\_price\_text\_view);  
 emptyCartTextView = findViewById(R.id.empty\_cart\_text\_view);  
 checkoutButton = findViewById(R.id.checkout\_button);  
 }  
   
 private void setupRecyclerView() {  
 if (recyclerView == null || cart == null) {  
 return;  
 }  
   
 cartItems = cart.getItems();  
 adapter = new CartAdapter(this, cartItems, this);  
 recyclerView.setLayoutManager(new LinearLayoutManager(this));  
 recyclerView.setAdapter(adapter);  
 }  
   
 private void updateUI() {  
 if (totalPriceTextView == null || emptyCartTextView == null ||   
 checkoutButton == null || cart == null) {  
 return;  
 }  
   
 if (cart.isEmpty()) {  
 emptyCartTextView.setVisibility(View.VISIBLE);  
 recyclerView.setVisibility(View.GONE);  
 totalPriceTextView.setVisibility(View.GONE);  
 checkoutButton.setVisibility(View.GONE);  
 } else {  
 emptyCartTextView.setVisibility(View.GONE);  
 recyclerView.setVisibility(View.VISIBLE);  
 totalPriceTextView.setVisibility(View.VISIBLE);  
 checkoutButton.setVisibility(View.VISIBLE);  
   
 try {  
 totalPrice = cart.getTotalPrice().doubleValue();  
   
   
 NumberFormat currencyFormat = NumberFormat.getCurrencyInstance(new Locale("ru", "RU"));  
 totalPriceTextView.setText("Итого: " + currencyFormat.format(totalPrice));  
 } catch (Exception e) {  
   
 }  
 }  
 }  
   
 private boolean checkAndRequestStoragePermission() {  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.R) {  
   
 if (!Environment.isExternalStorageManager()) {  
 try {  
 Intent intent = new Intent(Settings.ACTION\_MANAGE\_APP\_ALL\_FILES\_ACCESS\_PERMISSION);  
 Uri uri = Uri.fromParts("package", getPackageName(), null);  
 intent.setData(uri);  
 startActivity(intent);  
 } catch (Exception e) {  
   
 showManualStoragePermissionDialog();  
 }  
 return false;  
 }  
 return true;  
 } else {  
 if (ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE\_EXTERNAL\_STORAGE)   
 != PackageManager.PERMISSION\_GRANTED) {  
 requestStoragePermission();  
 return false;  
 }  
 return true;  
 }  
 }  
   
 private void requestStoragePermission() {  
 ActivityCompat.requestPermissions(this,  
 new String[]{Manifest.permission.WRITE\_EXTERNAL\_STORAGE,  
 Manifest.permission.READ\_EXTERNAL\_STORAGE},  
 STORAGE\_PERMISSION\_CODE);  
 }  
   
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
   
 if (requestCode == STORAGE\_PERMISSION\_CODE) {  
 if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION\_GRANTED) {  
   
 processCheckoutAfterPermissionCheck();  
 } else {  
   
 Toast.makeText(this, "Без разрешения невозможно сохранить заказ", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 }  
   
 private void processCheckout() {  
 try {  
   
 if (cart.isEmpty()) {  
 return;  
 }  
   
   
 if (checkAndRequestStoragePermission()) {  
 processCheckoutAfterPermissionCheck();  
 }  
 } catch (Exception e) {  
   
 }  
 }  
   
 private void processCheckoutAfterPermissionCheck() {  
   
 try {  
   
 if (cart == null || cart.isEmpty()) {  
 Toast.makeText(this, "Корзина пуста", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
   
   
 if (dbHelper == null) {  
 dbHelper = DatabaseHelper.getInstance(this);  
 }  
   
 User currentUserInfo = dbHelper.getUserById(currentUser.getId());  
 if (currentUserInfo == null) {  
 Toast.makeText(this, "Ошибка получения информации о пользователе", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
   
   
 double orderTotal = cart.getTotalPrice().doubleValue();  
   
   
 if (currentUserInfo.getBalance() < orderTotal) {  
 Toast.makeText(this,   
 "Недостаточно средств на балансе. Требуется: " + orderTotal + " ₽, доступно: " +   
 currentUserInfo.getBalance() + " ₽",   
 Toast.LENGTH\_LONG).show();  
 return;  
 }  
   
   
 double newBalance = currentUserInfo.getBalance() - orderTotal;  
 boolean updated = dbHelper.updateUserBalance(currentUserInfo.getId(), newBalance);  
   
 if (!updated) {  
 Toast.makeText(this, "Ошибка списания средств с баланса", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
   
   
 currentUser.setBalance(newBalance);  
 Log.d("CartActivity", "Баланс пользователя обновлен: " + newBalance);  
   
   
 new CreateOrderTask().execute();  
 } catch (Exception e) {  
 Log.e("CartActivity", "Ошибка при проверке баланса: " + e.getMessage());  
 Toast.makeText(this, "Произошла ошибка при обработке заказа", Toast.LENGTH\_SHORT).show();  
 }  
 }  
   
 private class CreateOrderTask extends AsyncTask<Void, Void, OrderResult> {  
 private boolean isPaid = false;  
   
 public CreateOrderTask() {  
 this.isPaid = true;   
 }  
   
 public CreateOrderTask(boolean isPaid) {  
 this.isPaid = true;   
 }  
   
 @Override  
 protected void onPreExecute() {  
 super.onPreExecute();  
 if (progressDialog != null) {  
 progressDialog.show();  
 }  
 }  
   
 @Override  
 protected OrderResult doInBackground(Void... voids) {  
 OrderResult result = new OrderResult();  
   
 try {  
   
 List<CartItem> items = cart.getItems();  
 if (items == null || items.isEmpty()) {  
 result.error = "Корзина пуста";  
 return result;  
 }  
   
   
 Order order = new Order(currentUser.getId(),   
 new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.getDefault()).format(new Date()),   
 "Обработка");  
   
   
 order.setPaid(true);  
   
   
 order.setOrderNumber(generateOrderNumber());  
   
   
 ArrayList<OrderItem> orderItems = new ArrayList<>();  
   
   
 BigDecimal totalAmount = BigDecimal.ZERO;  
 for (CartItem cartItem : items) {  
 Product product = cartItem.getProduct();  
   
   
 OrderItem orderItem = new OrderItem();  
 orderItem.setProductId(product.getId());  
 orderItem.setProductName(product.getName());  
 orderItem.setQuantity(cartItem.getQuantity());  
 orderItem.setPrice(product.getPrice().doubleValue());  
   
   
 orderItems.add(orderItem);  
   
   
 totalAmount = totalAmount.add(  
 new BigDecimal(product.getPrice().doubleValue())  
 .multiply(BigDecimal.valueOf(cartItem.getQuantity())));  
 }  
   
   
 order.setTotalAmount(totalAmount.doubleValue());  
   
   
 long orderId = dbHelper.createOrder(order, orderItems);  
 if (orderId == -1) {  
 result.error = "Ошибка при создании заказа";  
 return result;  
 }  
   
   
 result.success = true;  
 result.orderId = orderId;  
 result.order = order;  
 result.orderItems = orderItems;  
   
   
 cart.clear();  
   
   
 result = saveOrderToTextFile(result);  
   
 } catch (Exception e) {  
 result.error = "Непредвиденная ошибка: " + e.getMessage();  
 }  
   
 return result;  
 }  
   
 @Override  
 protected void onPostExecute(OrderResult result) {  
 super.onPostExecute(result);  
   
   
 if (progressDialog != null && progressDialog.isShowing()) {  
 progressDialog.dismiss();  
 }  
   
   
 if (result.success) {  
   
 sendOrderStatusNotification(  
 result.order.getOrderNumber(),  
 "Заказ создан",  
 "Заказ №" + result.order.getOrderNumber() + " успешно создан и обрабатывается"  
 );  
   
   
 updateUI();  
   
   
   
 saveOrderForStatusTracking(result.orderId, result.order.getOrderNumber());  
   
   
 startProcessingOrder(result.orderId, result.order.getOrderNumber());  
   
   
 if (result.fileSaved) {  
 Toast.makeText(CartActivity.this,   
 "Информация о заказе сохранена в Загрузки",   
 Toast.LENGTH\_LONG).show();  
 }  
 } else {  
   
 String errorMessage = result.error != null ? result.error : "Неизвестная ошибка";  
 Toast.makeText(CartActivity.this, errorMessage, Toast.LENGTH\_LONG).show();  
 }  
 }  
 }  
   
 private OrderResult saveOrderToTextFile(OrderResult result) {  
 try {  
   
 OrderFileManager fileManager = new OrderFileManager(this);  
   
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.Q) {  
   
 ContentResolver resolver = getContentResolver();  
 ContentValues contentValues = new ContentValues();  
 String fileName = "Order\_" + result.order.getOrderNumber() + "\_" + System.currentTimeMillis() + ".txt";  
 contentValues.put(MediaStore.MediaColumns.DISPLAY\_NAME, fileName);  
 contentValues.put(MediaStore.MediaColumns.MIME\_TYPE, "text/plain");  
 contentValues.put(MediaStore.MediaColumns.RELATIVE\_PATH, Environment.DIRECTORY\_DOWNLOADS);  
   
 Uri uri = resolver.insert(MediaStore.Downloads.EXTERNAL\_CONTENT\_URI, contentValues);  
   
 if (uri != null) {  
 try (OutputStream outputStream = resolver.openOutputStream(uri)) {  
 if (outputStream != null) {  
   
 StringBuilder content = new StringBuilder();  
 content.append("=== ЗАКАЗ #").append(result.order.getOrderNumber()).append(" ===\n\n");  
 content.append("Дата: ").append(result.order.getOrderDate()).append("\n");  
 content.append("Статус: ").append(result.order.getStatus()).append("\n");  
 content.append("Пользователь ID: ").append(String.valueOf(result.order.getUserId())).append("\n");  
 content.append("Статус оплаты: ").append(result.order.isPaid() ? "Оплачен" : "Не оплачен").append("\n\n");  
   
   
 content.append("ТОВАРЫ:\n");  
 content.append("----------------------------------------\n");  
 content.append(String.format("%-30s %8s %10s %10s\n", "Товар", "Кол-во", "Цена", "Сумма"));  
 content.append("----------------------------------------\n");  
   
 double total = 0.0;  
 for (OrderItem item : result.orderItems) {  
 double itemTotal = item.getQuantity() \* item.getPrice();  
 total += itemTotal;  
   
 content.append(String.format("%-30s %8d %10.2f %10.2f\n",   
 item.getProductName(),   
 item.getQuantity(),   
 item.getPrice(),   
 itemTotal));  
 }  
   
 content.append("----------------------------------------\n");  
 content.append(String.format("%49s %10.2f\n", "ИТОГО:", total));  
   
   
 outputStream.write(content.toString().getBytes());  
 result.fileSaved = true;  
 }  
 } catch (IOException e) {  
 Log.e("CartActivity", "Error writing to file: " + e.getMessage());  
 }  
 }  
 } else {  
   
 File downloadsDir = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY\_DOWNLOADS);  
 if (!downloadsDir.exists()) {  
 downloadsDir.mkdirs();  
 }  
   
 String fileName = "Order\_" + result.order.getOrderNumber() + "\_" + System.currentTimeMillis() + ".txt";  
 File orderFile = new File(downloadsDir, fileName);  
   
 result.fileSaved = fileManager.saveOrderToFile(result.order, orderFile);  
 if (result.fileSaved) {  
   
 MediaScannerConnection.scanFile(  
 CartActivity.this,  
 new String[]{orderFile.getAbsolutePath()},  
 new String[]{"text/plain"},  
 null  
 );  
 }  
 }  
   
 } catch (Exception e) {  
 Log.e("CartActivity", "Error saving order file: " + e.getMessage());  
 }  
   
 return result;  
 }  
   
 private static class OrderResult {  
 boolean success = false;  
 long orderId = -1;  
 boolean fileSaved = false;  
 boolean needSystemFileSave = false;   
 String error = null;  
 Order order = null;  
 ArrayList<OrderItem> orderItems = null;  
 }  
   
 private String generateOrderNumber() {  
 SimpleDateFormat dateFormat = new SimpleDateFormat("yyyyMMddHHmmss", Locale.getDefault());  
 return "ORD-" + dateFormat.format(new Date());  
 }  
   
 @Override  
 protected void onResume() {  
 super.onResume();  
   
   
 if (getIntent().getBooleanExtra("CONTINUE\_CHECKOUT", false)) {  
   
 new Handler().post(this::processCheckoutAfterPermissionCheck);  
   
 getIntent().removeExtra("CONTINUE\_CHECKOUT");  
 }  
 }  
   
 private void showManualStoragePermissionDialog() {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Требуется разрешение")  
 .setMessage("Для сохранения заказа необходимо разрешение на доступ к файлам. Пожалуйста, перейдите в настройки приложения, затем в 'Разрешения' и включите 'Хранилище'.")  
 .setPositiveButton("Перейти в настройки", (dialog, which) -> {  
 Intent intent = new Intent(Settings.ACTION\_APPLICATION\_DETAILS\_SETTINGS);  
 Uri uri = Uri.fromParts("package", getPackageName(), null);  
 intent.setData(uri);  
 startActivity(intent);  
 })  
 .setNegativeButton("Отмена", (dialog, which) -> {  
 dialog.dismiss();  
 })  
 .setCancelable(false)  
 .show();  
 }  
   
 @Override  
 public boolean onSupportNavigateUp() {  
 onBackPressed();  
 return true;  
 }  
   
 @Override  
 public void onQuantityChanged(long productId, int newQuantity) {  
 cart.updateQuantity(productId, newQuantity);  
 updateUI();  
 adapter.notifyDataSetChanged();  
 }  
   
 @Override  
 public void onItemRemoved(long productId) {  
 cart.removeItem(productId);  
 updateUI();  
 adapter.notifyDataSetChanged();  
 }  
   
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
 if (progressDialog != null && progressDialog.isShowing()) {  
 progressDialog.dismiss();  
 }  
 }  
   
 @Override  
 public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
 if (item.getItemId() == android.R.id.home) {  
 onBackPressed();  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
   
 private void sendOrderStatusNotification(String orderNumber, String title, String message) {  
 try {  
   
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {  
 NotificationChannel channel = new NotificationChannel(  
 "order\_status\_channel",  
 "Статус заказа",  
 NotificationManager.IMPORTANCE\_DEFAULT  
 );  
 channel.setDescription("Уведомления о статусе заказов");  
   
 NotificationManager notificationManager = getSystemService(NotificationManager.class);  
 notificationManager.createNotificationChannel(channel);  
 }  
   
   
 Intent intent = new Intent(this, OrderHistoryActivity.class);  
 intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);  
   
 PendingIntent pendingIntent;  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.S) {  
 pendingIntent = PendingIntent.getActivity(this, 0, intent, PendingIntent.FLAG\_IMMUTABLE);  
 } else {  
 pendingIntent = PendingIntent.getActivity(this, 0, intent, PendingIntent.FLAG\_UPDATE\_CURRENT);  
 }  
   
   
 NotificationCompat.Builder builder = new NotificationCompat.Builder(this, "order\_status\_channel")  
 .setSmallIcon(R.drawable.ic\_launcher\_foreground)  
 .setContentTitle(title)  
 .setContentText(message)  
 .setPriority(NotificationCompat.PRIORITY\_DEFAULT)  
 .setContentIntent(pendingIntent)  
 .setAutoCancel(true);  
   
   
 NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);  
   
   
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.TIRAMISU) {  
 if (ContextCompat.checkSelfPermission(this, Manifest.permission.POST\_NOTIFICATIONS)   
 == PackageManager.PERMISSION\_GRANTED) {  
 notificationManager.notify(orderNumber.hashCode(), builder.build());  
 }  
 } else {  
 notificationManager.notify(orderNumber.hashCode(), builder.build());  
 }  
   
 } catch (Exception e) {  
   
 }  
 }  
  
   
 private void downloadReceiptForOrder(long orderId) {  
   
 ProgressDialog progressDialog = new ProgressDialog(this);  
 progressDialog.setMessage("Подготовка чека...");  
 progressDialog.setCancelable(false);  
 progressDialog.show();  
   
   
 new Thread(() -> {  
 boolean success = false;  
 String errorMessage = null;  
   
 try {  
   
 Order order = dbHelper.getOrder(orderId);  
   
 if (order != null) {  
   
 OrderResult result = new OrderResult();  
 result.order = order;  
 result.orderId = order.getId();  
 result.orderItems = new ArrayList<>(order.getOrderItems());  
 result.success = true;  
   
   
 result = saveOrderToTextFile(result);  
 success = result.fileSaved;  
 } else {  
 errorMessage = "Заказ не найден";  
 }  
 } catch (Exception e) {  
 errorMessage = "Ошибка: " + e.getMessage();  
 }  
   
   
 final boolean finalSuccess = success;  
 final String finalErrorMessage = errorMessage;  
   
   
 runOnUiThread(() -> {  
   
 progressDialog.dismiss();  
   
 if (finalSuccess) {  
 Toast.makeText(CartActivity.this,   
 "Чек сохранен в Загрузки",   
 Toast.LENGTH\_LONG).show();  
 } else {  
 Toast.makeText(CartActivity.this,   
 finalErrorMessage != null ? finalErrorMessage : "Не удалось сохранить чек",   
 Toast.LENGTH\_LONG).show();  
 }  
   
   
 Intent intent = new Intent(CartActivity.this, OrderHistoryActivity.class);  
 intent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP);  
 startActivity(intent);  
 finish();  
 });  
 }).start();  
 }  
  
   
 private void saveOrderForStatusTracking(long orderId, String orderNumber) {  
 try {  
 SharedPreferences prefs = getSharedPreferences("OrderTracking", MODE\_PRIVATE);  
 SharedPreferences.Editor editor = prefs.edit();  
 editor.putLong("last\_order\_id", orderId);  
 editor.putString("last\_order\_number", orderNumber);  
 editor.putLong("order\_created\_time", System.currentTimeMillis());  
 editor.putString("current\_status", "Обработка");  
 editor.apply();  
 } catch (Exception e) {  
 Log.e("CartActivity", "Ошибка сохранения данных заказа: " + e.getMessage());  
 }  
 }  
  
   
 private void startProcessingOrder(final long orderId, final String orderNumber) {  
 try {  
   
 showTrackDeliveryDialog(orderId, orderNumber);  
   
   
 new Handler(Looper.getMainLooper()).postDelayed(() -> {  
 changeOrderStatus(orderId, orderNumber, "Доставка");  
 }, 8000);   
 } catch (Exception e) {  
 Log.e("CartActivity", "Ошибка запуска обработки заказа: " + e.getMessage());  
 }  
 }  
  
   
 private void changeOrderStatus(final long orderId, final String orderNumber, final String newStatus) {  
 try {  
   
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(this);  
 if (dbHelper == null) {  
 Log.e("CartActivity", "Нет доступа к базе данных");  
 return;  
 }  
   
   
 Order order = dbHelper.getOrder(orderId);  
 if (order == null) {  
 Log.e("CartActivity", "Заказ не найден: " + orderId);  
 return;  
 }  
   
   
 String currentStatus = order.getStatus();  
 Log.d("CartActivity", "Текущий статус заказа " + orderNumber + ": " + currentStatus);  
   
   
 boolean canChangeStatus = false;  
   
 if ("Доставка".equals(newStatus) && "Обработка".equals(currentStatus)) {  
   
 canChangeStatus = true;  
 } else if ("Доставлен".equals(newStatus) && "Доставка".equals(currentStatus)) {  
   
 canChangeStatus = true;  
 }  
   
 if (canChangeStatus) {  
   
 boolean success = dbHelper.updateOrderStatus(orderId, newStatus);  
   
 if (success) {  
   
 SharedPreferences prefs = getSharedPreferences("OrderTracking", MODE\_PRIVATE);  
 SharedPreferences.Editor editor = prefs.edit();  
 editor.putString("current\_status", newStatus);  
 editor.apply();  
   
   
 String title, message;  
 if ("Доставка".equals(newStatus)) {  
 title = "Статус заказа изменен";  
 message = "Заказ №" + orderNumber + " передан в доставку";  
   
   
 new Handler(Looper.getMainLooper()).postDelayed(() -> {  
 changeOrderStatus(orderId, orderNumber, "Доставлен");  
 }, 20000);  
 } else {  
 title = "Заказ доставлен";  
 message = "Заказ №" + orderNumber + " успешно доставлен!";  
 }  
   
 sendOrderStatusNotification(orderNumber, title, message);  
 Log.d("CartActivity", "Статус заказа " + orderNumber + " изменен на: " + newStatus);  
 } else {  
 Log.e("CartActivity", "Не удалось обновить статус заказа " + orderNumber);  
 }  
 } else {  
 Log.d("CartActivity", "Невозможно изменить статус с " + currentStatus + " на " + newStatus);  
 }  
 } catch (Exception e) {  
 Log.e("CartActivity", "Ошибка при смене статуса заказа: " + e.getMessage());  
 }  
 }  
  
   
 private void showTrackDeliveryDialog(long orderId, String orderNumber) {  
 try {  
 new AlertDialog.Builder(this)  
 .setTitle("Отслеживание доставки")  
 .setMessage("Хотите отследить доставку вашего заказа?")  
 .setPositiveButton("Да, отследить", (dialog, which) -> {  
   
 Intent intent = new Intent(this, DeliveryTrackingActivity.class);  
 intent.putExtra("orderId", orderId);  
 intent.putExtra("orderNumber", orderNumber);  
 startActivity(intent);  
 })  
 .setNegativeButton("Нет, позже", null)  
 .setCancelable(false)   
 .show();  
 } catch (Exception e) {  
 Log.e("CartActivity", "Ошибка показа диалога отслеживания: " + e.getMessage());  
 }  
 }  
}

package com.example.upavlika;  
  
import android.Manifest;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.app.PendingIntent;  
import android.content.Context;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.os.Build;  
import android.os.Bundle;  
import android.os.CountDownTimer;  
import android.util.Log;  
import android.view.View;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.RatingBar;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
import androidx.core.app.ActivityCompat;  
import androidx.core.app.NotificationCompat;  
import androidx.core.content.ContextCompat;  
import androidx.core.app.NotificationManagerCompat;  
  
import com.example.upavlika.data.DatabaseHelper;  
import com.example.upavlika.model.Order;  
import com.example.upavlika.service.DeliveryTrackerService;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.Locale;  
import java.util.Random;  
  
public class DeliveryTrackingActivity extends AppCompatActivity {  
  
 private static final String CHANNEL\_ID = "delivery\_notification\_channel";  
 private static final int NOTIFICATION\_ID = 1;  
   
   
 private static final int LOCATION\_PERMISSION\_REQUEST\_CODE = 1001;  
   
 private TextView textViewOrderNumber;  
 private TextView textViewDriverName;  
 private TextView textViewVehicleInfo;  
 private TextView textViewETA;  
 private TextView textViewRating;  
 private Button buttonChat;  
 private ImageView imageViewDriver;  
 private ImageView imageViewVehicle;  
 private Order currentOrder;  
 private CountDownTimer deliveryTimer;  
 private boolean deliveryCompleted = false;  
 private android.widget.ProgressBar progressBar;  
   
   
 private final String[] driverNames = {"Алексей", "Михаил", "Иван", "Дмитрий", "Сергей", "Андрей"};  
 private final String[] driverSurnames = {"Иванов", "Петров", "Сидоров", "Смирнов", "Соколов", "Морозов"};  
 private final String[] vehicleModels = {"Volkswagen Polo", "Hyundai Solaris", "Kia Rio", "Skoda Rapid", "Toyota Camry", "Ford Focus"};  
 private final String[] vehicleColors = {"белый", "черный", "серебристый", "синий", "красный", "серый"};  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_delivery\_tracking);  
   
   
 initViews();  
   
   
 long orderId = getIntent().getLongExtra("orderId", -1);  
 String orderNumber = getIntent().getStringExtra("orderNumber");  
   
   
 Toolbar toolbar = findViewById(R.id.toolbar);  
 if (toolbar != null) {  
 setSupportActionBar(toolbar);  
 if (getSupportActionBar() != null) {  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setDisplayShowHomeEnabled(true);  
 getSupportActionBar().setTitle("Отслеживание доставки");  
 }  
 }  
   
   
 createNotificationChannel();  
   
   
 if (loadOrderDetails(orderId, orderNumber)) {  
   
 generateDeliveryInfo();  
   
   
   
 startDeliveryTrackerService();  
 }  
 }  
  
 private void createNotificationChannel() {  
 try {  
   
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {  
 CharSequence name = "Доставка заказов";  
 String description = "Уведомления о статусе доставки заказов";  
 int importance = NotificationManager.IMPORTANCE\_HIGH;  
 NotificationChannel channel = new NotificationChannel(CHANNEL\_ID, name, importance);  
 channel.setDescription(description);  
   
   
 NotificationManager notificationManager = getSystemService(NotificationManager.class);  
 if (notificationManager != null) {  
 notificationManager.createNotificationChannel(channel);  
 }  
 }  
 } catch (Exception e) {  
   
 }  
 }  
  
 private void initViews() {  
 try {  
 textViewOrderNumber = findViewById(R.id.textViewOrderNumber);  
 textViewDriverName = findViewById(R.id.textViewDriverName);  
 textViewVehicleInfo = findViewById(R.id.textViewVehicleInfo);  
 textViewETA = findViewById(R.id.textViewETA);  
 textViewRating = findViewById(R.id.textViewRating);  
 buttonChat = findViewById(R.id.buttonChat);  
 imageViewDriver = findViewById(R.id.imageViewDriver);  
 imageViewVehicle = findViewById(R.id.imageViewVehicle);  
 progressBar = findViewById(R.id.progressBar);  
   
 if (progressBar != null) {  
 progressBar.setVisibility(View.GONE);  
 }  
   
 if (buttonChat != null) {  
 buttonChat.setOnClickListener(v -> openChatWithDriver());  
 }  
 } catch (Exception e) {  
   
 }  
 }  
  
 private boolean loadOrderDetails(long orderId, String orderNumber) {  
 try {  
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(this);  
 if (dbHelper == null) {  
 finish();  
 return false;  
 }  
   
 if (orderId != -1) {  
 currentOrder = dbHelper.getOrder(orderId);  
 } else {  
   
 finish();  
 return false;  
 }  
  
 if (currentOrder == null) {  
 finish();  
 return false;  
 }  
  
   
 if (textViewOrderNumber != null && currentOrder.getOrderNumber() != null) {  
 textViewOrderNumber.setText("Заказ № " + currentOrder.getOrderNumber());  
 }  
   
   
 String status = currentOrder.getStatus();  
   
   
 if ("Доставлен".equals(status)) {  
 deliveryCompleted = true;  
 if (textViewETA != null) {  
 textViewETA.setText("Заказ доставлен!");  
 }  
   
 if (deliveryTimer != null) {  
 deliveryTimer.cancel();  
 }  
 } else if (!"Отправлен".equals(status)) {  
   
 if (textViewETA != null) {  
 textViewETA.setText("Заказ " + status + ". Доставка еще не начата.");  
 }  
 return true;  
 }  
   
 return true;  
  
 } catch (Exception e) {  
 finish();  
 return false;  
 }  
 }  
  
 private void generateDeliveryInfo() {  
 try {  
   
 if (textViewDriverName == null || textViewVehicleInfo == null || textViewRating == null) {  
 return;  
 }  
   
   
 Random random = new Random();  
 String driverName = driverNames[random.nextInt(driverNames.length)] + " " +  
 driverSurnames[random.nextInt(driverSurnames.length)];  
 String vehicle = vehicleModels[random.nextInt(vehicleModels.length)] + ", " +  
 vehicleColors[random.nextInt(vehicleColors.length)];  
 String carNumber = generateCarNumber();  
 float rating = 4.0f + random.nextFloat();   
 if (rating > 5.0f) rating = 5.0f;  
  
   
 textViewDriverName.setText("Курьер: " + driverName);  
 textViewVehicleInfo.setText("Автомобиль: " + vehicle + "\nНомер: " + carNumber);  
 textViewRating.setText(String.format(Locale.getDefault(), "Рейтинг: %.1f ★", rating));  
  
 } catch (Exception e) {  
   
 }  
 }  
  
   
 private String generateCarNumber() {  
 Random random = new Random();  
 char[] letters = {'А', 'В', 'Е', 'К', 'М', 'Н', 'О', 'Р', 'С', 'Т', 'У', 'Х'};  
   
 StringBuilder sb = new StringBuilder();  
 sb.append(letters[random.nextInt(letters.length)]);  
 sb.append(String.format(Locale.getDefault(), "%03d", random.nextInt(1000)));  
 sb.append(letters[random.nextInt(letters.length)]);  
 sb.append(letters[random.nextInt(letters.length)]);  
   
   
 int region = random.nextInt(199) + 1;  
 sb.append(String.format(Locale.getDefault(), "%02d", region));  
   
   
 sb.append(" RUS");  
   
 return sb.toString();  
 }  
  
 private void startDeliveryTrackerService() {  
 try {  
   
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.TIRAMISU &&  
 ContextCompat.checkSelfPermission(this, Manifest.permission.POST\_NOTIFICATIONS) != PackageManager.PERMISSION\_GRANTED) {  
   
 ActivityCompat.requestPermissions(this,   
 new String[]{Manifest.permission.POST\_NOTIFICATIONS},   
 1003);  
 return;  
 }  
   
   
 if (!checkLocationPermissions()) {  
 return;  
 }  
   
   
 Intent intent = new Intent(this, DeliveryTrackerService.class);  
 intent.setAction(DeliveryTrackerService.ACTION\_START\_TRACKING);  
 intent.putExtra("orderId", currentOrder.getId());  
 intent.putExtra("orderNumber", currentOrder.getOrderNumber());  
   
   
 String currentStatus = currentOrder.getStatus();  
 long deliveryTime;  
   
 if ("Обработка".equals(currentStatus)) {  
   
 deliveryTime = 28000;  
 } else if ("Доставка".equals(currentStatus)) {  
   
 deliveryTime = 20000;  
 } else {  
   
 deliveryTime = 30000;  
 }  
   
 intent.putExtra("total\_time", deliveryTime);  
   
   
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {  
 startForegroundService(intent);  
 } else {  
 startService(intent);  
 }  
   
   
 startDeliveryCountdown(deliveryTime);  
   
   
 if (progressBar != null) {  
 progressBar.setVisibility(View.VISIBLE);  
 progressBar.setIndeterminate(false);  
 progressBar.setMax(100);  
 progressBar.setProgress(0);  
 }  
   
   
 Toast.makeText(this, "Отслеживание доставки запущено", Toast.LENGTH\_SHORT).show();  
 } catch (Exception e) {  
   
 Log.e("DeliveryTracking", "Ошибка запуска сервиса: " + e.getMessage());  
 Toast.makeText(this,   
 "Ошибка при запуске отслеживания: " + e.getMessage(),   
 Toast.LENGTH\_LONG).show();  
 }  
 }  
  
 private boolean checkLocationPermissions() {  
 if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_FINE\_LOCATION)  
 != PackageManager.PERMISSION\_GRANTED) {  
   
   
 if (ActivityCompat.shouldShowRequestPermissionRationale(this,  
 Manifest.permission.ACCESS\_FINE\_LOCATION)) {  
   
 new AlertDialog.Builder(this)  
 .setTitle("Необходимо разрешение на местоположение")  
 .setMessage("Для отслеживания доставки необходим доступ к местоположению.")  
 .setPositiveButton("ОК", (dialog, which) -> {  
   
 ActivityCompat.requestPermissions(DeliveryTrackingActivity.this,  
 new String[]{Manifest.permission.ACCESS\_FINE\_LOCATION},  
 LOCATION\_PERMISSION\_REQUEST\_CODE);  
 })  
 .setNegativeButton("Отмена", (dialog, which) -> {  
   
 dialog.dismiss();  
 })  
 .create()  
 .show();  
 } else {  
   
 ActivityCompat.requestPermissions(this,  
 new String[]{Manifest.permission.ACCESS\_FINE\_LOCATION},  
 LOCATION\_PERMISSION\_REQUEST\_CODE);  
 }  
 return false;  
 }  
 return true;  
 }  
  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (requestCode == LOCATION\_PERMISSION\_REQUEST\_CODE) {  
   
 if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION\_GRANTED) {  
   
 startDeliveryTrackerService();  
 } else {  
   
 new AlertDialog.Builder(this)  
 .setTitle("Доступ ограничен")  
 .setMessage("Без разрешения на доступ к местоположению функция отслеживания будет недоступна.")  
 .setPositiveButton("ОК", null)  
 .create()  
 .show();  
 }  
 }  
 }  
  
 private void startDeliveryCountdown(long totalMillis) {  
 try {  
   
 deliveryTimer = new CountDownTimer(totalMillis, 1000) {  
 @Override  
 public void onTick(long millisUntilFinished) {  
 try {  
 if (textViewETA != null) {  
   
 int minutes = (int) (millisUntilFinished / 1000) / 60;  
 int seconds = (int) (millisUntilFinished / 1000) % 60;  
 String timeFormatted = String.format(Locale.getDefault(),   
 "%02d:%02d", minutes, seconds);  
   
   
 textViewETA.setText("Примерное время доставки: " + timeFormatted);  
   
   
 if (progressBar != null) {  
 int progress = (int) (100 - ((double) millisUntilFinished / totalMillis \* 100));  
 progressBar.setProgress(progress);  
 }  
   
   
 checkAndUpdateOrderStatus(millisUntilFinished);  
 }  
 } catch (Exception e) {  
   
 }  
 }  
   
 @Override  
 public void onFinish() {  
 try {  
   
 deliveryCompleted = true;  
   
   
 if (textViewETA != null) {  
 textViewETA.setText("Заказ доставлен!");  
 }  
   
   
 if (progressBar != null) {  
 progressBar.setProgress(100);  
 }  
   
   
 checkAndShowRating();  
   
   
 stopDeliveryTrackerService();  
 } catch (Exception e) {  
   
 }  
 }  
 }.start();  
 } catch (Exception e) {  
   
 }  
 }  
  
   
 private void checkAndUpdateOrderStatus(long millisUntilFinished) {  
 try {  
 if (currentOrder == null) return;  
   
   
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(this);  
 if (dbHelper != null) {  
 Order updatedOrder = dbHelper.getOrder(currentOrder.getId());  
 if (updatedOrder != null) {  
   
 currentOrder = updatedOrder;  
   
   
 Log.d("DeliveryTracking", "Текущий статус заказа: " + currentOrder.getStatus());  
 }  
 }  
 } catch (Exception e) {  
 Log.e("DeliveryTracking", "Ошибка при проверке статуса: " + e.getMessage());  
 }  
 }  
   
   
 private void checkAndShowRating() {  
 try {  
   
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(this);  
 if (dbHelper != null) {  
 Order updatedOrder = dbHelper.getOrder(currentOrder.getId());  
 if (updatedOrder != null && "Доставлен".equals(updatedOrder.getStatus())) {  
   
 showRatingDialog();  
 } else {  
   
 Toast.makeText(this,   
 "Заказ еще в процессе доставки. Статус: " +   
 (updatedOrder != null ? updatedOrder.getStatus() : "неизвестен"),   
 Toast.LENGTH\_LONG).show();  
 }  
 }  
 } catch (Exception e) {  
 Log.e("DeliveryTracking", "Ошибка при проверке статуса: " + e.getMessage());  
 }  
 }  
  
 private void stopDeliveryTrackerService() {  
 try {  
 Intent serviceIntent = new Intent(this, DeliveryTrackerService.class);  
 stopService(serviceIntent);  
 } catch (Exception e) {  
   
 }  
 }  
  
 private void sendDeliveryNotification(String title, String message) {  
 try {  
   
 Intent intent = new Intent(this, DeliveryTrackingActivity.class);  
 intent.putExtra("orderId", currentOrder.getId());  
 intent.putExtra("orderNumber", currentOrder.getOrderNumber());  
 intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);  
   
   
 PendingIntent pendingIntent;  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {  
 pendingIntent = PendingIntent.getActivity(this, 0, intent, PendingIntent.FLAG\_IMMUTABLE | PendingIntent.FLAG\_UPDATE\_CURRENT);  
 } else {  
 pendingIntent = PendingIntent.getActivity(this, 0, intent, PendingIntent.FLAG\_UPDATE\_CURRENT);  
 }  
   
   
 NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL\_ID)  
 .setSmallIcon(R.drawable.ic\_launcher\_foreground)  
 .setContentTitle(title)  
 .setContentText(message)  
 .setPriority(NotificationCompat.PRIORITY\_HIGH)  
 .setContentIntent(pendingIntent)  
 .setAutoCancel(true);  
   
   
 NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);  
 if (ActivityCompat.checkSelfPermission(this, Manifest.permission.POST\_NOTIFICATIONS) == PackageManager.PERMISSION\_GRANTED) {  
 notificationManager.notify(NOTIFICATION\_ID, builder.build());  
 }  
 } catch (Exception e) {  
   
 }  
 }  
  
 private void openChatWithDriver() {  
   
 new AlertDialog.Builder(this)  
 .setTitle("Функция недоступна")  
 .setMessage("Чат с водителем временно недоступен.")  
 .setPositiveButton("ОК", null)  
 .show();  
 }  
  
 private void showRatingDialog() {  
 try {  
 View dialogView = getLayoutInflater().inflate(R.layout.dialog\_delivery\_rating, null);  
 RatingBar ratingBar = dialogView.findViewById(R.id.ratingBar);  
   
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Оцените доставку");  
 builder.setView(dialogView);  
 builder.setPositiveButton("Отправить", (dialog, which) -> {  
   
 float rating = ratingBar.getRating();  
   
   
 Toast.makeText(DeliveryTrackingActivity.this,   
 "Спасибо за вашу оценку: " + rating,   
 Toast.LENGTH\_SHORT).show();  
   
   
   
 });  
 builder.setNegativeButton("Пропустить", null);  
 builder.setCancelable(true);  
 builder.show();  
 } catch (Exception e) {  
   
 }  
 }  
  
 @Override  
 public boolean onSupportNavigateUp() {  
 onBackPressed();  
 return true;  
 }  
  
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
   
 if (deliveryTimer != null) {  
 deliveryTimer.cancel();  
 }  
 }  
}

package com.example.upavlika;  
  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
  
import com.example.upavlika.model.User;  
import com.example.upavlika.util.UserManager;  
import com.google.android.material.textfield.TextInputEditText;  
import com.google.android.material.textfield.TextInputLayout;  
  
public class LoginActivity extends AppCompatActivity {  
  
 private TextInputEditText emailEditText;  
 private TextInputEditText passwordEditText;  
 private Button loginButton;  
 private TextView registerTextView;  
 private TextView forgotPasswordTextView;  
 private UserManager userManager;  
 private boolean returnToCart = false;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 try {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_login);  
  
   
 if (getIntent().hasExtra("RETURN\_TO\_CART")) {  
 returnToCart = getIntent().getBooleanExtra("RETURN\_TO\_CART", false);  
 }  
  
   
 userManager = UserManager.getInstance(this);  
  
   
 emailEditText = findViewById(R.id.emailEditText);  
 passwordEditText = findViewById(R.id.passwordEditText);  
 loginButton = findViewById(R.id.loginButton);  
 registerTextView = findViewById(R.id.registerTextView);  
 forgotPasswordTextView = findViewById(R.id.forgotPasswordTextView);  
  
   
 loginButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 try {  
 String email = emailEditText.getText() != null ? emailEditText.getText().toString().trim() : "";  
 String password = passwordEditText.getText() != null ? passwordEditText.getText().toString().trim() : "";  
  
   
 if (email.isEmpty() || password.isEmpty()) {  
 Toast.makeText(LoginActivity.this, "Заполните все поля", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
  
   
 User user = userManager.authenticateUser(email, password);  
   
 if (user != null) {  
   
 saveUserSession(user);  
   
 if (returnToCart) {  
   
 Intent intent = new Intent(LoginActivity.this, CartActivity.class);  
 startActivity(intent);  
 } else {  
   
 Intent intent = new Intent(LoginActivity.this, MainActivity.class);  
 startActivity(intent);  
 }  
 finish();  
 } else {  
 Toast.makeText(LoginActivity.this, "Неверный email или пароль", Toast.LENGTH\_SHORT).show();  
 }  
 } catch (Exception e) {  
 Log.e("LoginActivity", "Ошибка при авторизации: " + e.getMessage());  
 Toast.makeText(LoginActivity.this, "Ошибка при авторизации", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
  
   
 registerTextView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(LoginActivity.this, RegisterActivity.class);  
   
 if (returnToCart) {  
 intent.putExtra("RETURN\_TO\_CART", true);  
 }  
 startActivity(intent);  
 }  
 });  
   
 forgotPasswordTextView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 showForgotPasswordDialog();  
 }  
 });  
 } catch (Exception e) {  
 Log.e("LoginActivity", "Ошибка при инициализации: " + e.getMessage());  
 Toast.makeText(this, "Произошла ошибка при запуске приложения", Toast.LENGTH\_LONG).show();  
 finish();  
 }  
 }  
   
 private void showForgotPasswordDialog() {  
 // Шаг 1: запрос имени пользователя для восстановления пароля  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 View dialogView = getLayoutInflater().inflate(R.layout.dialog\_forgot\_password, null);  
 builder.setView(dialogView);  
   
 final TextInputEditText usernameInputET = dialogView.findViewById(R.id.usernameRecoveryEditText);  
 Button sendButton = dialogView.findViewById(R.id.sendButton);  
   
 final AlertDialog dialog = builder.create();  
 dialog.show();  
   
 sendButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String username = usernameInputET.getText() != null ? usernameInputET.getText().toString().trim() : "";  
   
 if (username.isEmpty()) {  
 Toast.makeText(LoginActivity.this, "Введите имя пользователя", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
   
 // Проверяем, существует ли пользователь с таким именем  
 if (userManager.isNameExists(username)) {  
 dialog.dismiss();  
 // Переходим к диалогу сброса пароля  
 showResetPasswordDialog(username);  
 } else {  
 Toast.makeText(LoginActivity.this, "Пользователь с таким именем не найден", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
 }  
   
 private void showResetPasswordDialog(final String username) {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 View dialogView = getLayoutInflater().inflate(R.layout.dialog\_reset\_password, null);  
 builder.setView(dialogView);  
   
 final TextInputEditText newPasswordET = dialogView.findViewById(R.id.newPasswordEditText);  
 final TextInputEditText confirmPasswordET = dialogView.findViewById(R.id.confirmNewPasswordEditText);  
 Button resetPasswordButton = dialogView.findViewById(R.id.resetPasswordButton);  
   
 final AlertDialog dialog = builder.create();  
 dialog.show();  
   
 resetPasswordButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String newPassword = newPasswordET.getText() != null ? newPasswordET.getText().toString().trim() : "";  
 String confirmPassword = confirmPasswordET.getText() != null ? confirmPasswordET.getText().toString().trim() : "";  
   
 if (newPassword.isEmpty() || confirmPassword.isEmpty()) {  
 Toast.makeText(LoginActivity.this, "Заполните все поля", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
   
 if (!newPassword.equals(confirmPassword)) {  
 Toast.makeText(LoginActivity.this, "Пароли не совпадают", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
   
 // Обновляем пароль пользователя  
 boolean success = userManager.resetPasswordByName(username, newPassword);  
 if (success) {  
 dialog.dismiss();  
 Toast.makeText(LoginActivity.this, "Пароль успешно изменен", Toast.LENGTH\_SHORT).show();  
 } else {  
 Toast.makeText(LoginActivity.this, "Ошибка при изменении пароля", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
 }  
   
 private void saveUserSession(User user) {  
 SharedPreferences sharedPreferences = getSharedPreferences("UserPrefs", MODE\_PRIVATE);  
 SharedPreferences.Editor editor = sharedPreferences.edit();  
   
 editor.putLong("userId", user.getId());  
 editor.putString("userName", user.getName());  
 editor.putString("userEmail", user.getEmail());  
 editor.putString("userRole", user.getRole().name());  
   
 editor.apply();  
 }  
}

package com.example.upavlika;  
  
import android.content.Intent;  
import android.database.sqlite.SQLiteDatabase;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.ProgressBar;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
import androidx.recyclerview.widget.GridLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.upavlika.data.DatabaseHelper;  
import com.example.upavlika.model.Cart;  
import com.example.upavlika.model.Product;  
import com.example.upavlika.model.User;  
import com.example.upavlika.model.UserRole;  
import com.example.upavlika.util.UserManager;  
import com.google.android.material.floatingactionbutton.FloatingActionButton;  
import com.google.android.material.tabs.TabLayout;  
  
import java.text.NumberFormat;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
import java.util.Locale;  
import java.io.File;  
  
public class MainActivity extends AppCompatActivity implements com.example.upavlika.adapter.ProductAdapter.OnProductClickListener {  
   
 private UserManager userManager;  
 private DatabaseHelper dbHelper;  
 private RecyclerView productsRecyclerView;  
 private com.example.upavlika.adapter.ProductAdapter productAdapter;  
 private TextView emptyView;  
 private ProgressBar progressBar;  
 private FloatingActionButton cartFab;  
 private User currentUser;  
 private Cart cart;  
 private List<Product> allProducts;  
 private TabLayout categoriesTabLayout;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
   
 try {  
 userManager = UserManager.getInstance(this);  
 dbHelper = DatabaseHelper.getInstance(this);  
 cart = Cart.getInstance();  
 cart.initialize(this);  
   
 Toolbar toolbar = findViewById(R.id.toolbar);  
 setSupportActionBar(toolbar);  
   
 productsRecyclerView = findViewById(R.id.productsRecyclerView);  
 emptyView = findViewById(R.id.emptyView);  
 progressBar = findViewById(R.id.progressBar);  
 cartFab = findViewById(R.id.cartFab);  
 categoriesTabLayout = findViewById(R.id.categoriesTabLayout);  
   
 currentUser = userManager.getCurrentUser();  
   
 if (currentUser == null) {  
 Intent intent = new Intent(this, LoginActivity.class);  
 startActivity(intent);  
 finish();  
 return;  
 }  
   
 productsRecyclerView.setLayoutManager(new GridLayoutManager(this, 2));  
   
 checkAndEnsureAllCategories();  
   
 loadProducts();  
   
 setupCategoryTabs();  
   
 cartFab.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(MainActivity.this, CartActivity.class);  
 startActivity(intent);  
 }  
 });  
 } catch (Exception e) {  
 Log.e("MainActivity", "Ошибка при инициализации: " + e.getMessage(), e);  
 Toast.makeText(this, "Произошла ошибка при загрузке. Попробуйте позже.", Toast.LENGTH\_LONG).show();  
 finish();  
 }  
 }  
  
 private void setupCategoryTabs() {  
 categoriesTabLayout.addTab(categoriesTabLayout.newTab().setText(R.string.all\_products));  
   
 categoriesTabLayout.addTab(categoriesTabLayout.newTab().setText(R.string.cakes));  
 categoriesTabLayout.addTab(categoriesTabLayout.newTab().setText(R.string.pastries));  
 categoriesTabLayout.addTab(categoriesTabLayout.newTab().setText(R.string.cookies));  
 categoriesTabLayout.addTab(categoriesTabLayout.newTab().setText(R.string.bread));  
  
 categoriesTabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {  
 @Override  
 public void onTabSelected(TabLayout.Tab tab) {  
 filterProductsByCategory(tab.getPosition());  
 }  
  
 @Override  
 public void onTabUnselected(TabLayout.Tab tab) {  
 }  
  
 @Override  
 public void onTabReselected(TabLayout.Tab tab) {  
 filterProductsByCategory(tab.getPosition());  
 }  
 });  
 }  
  
 private void filterProductsByCategory(int tabPosition) {  
 List<Product> filteredProducts = new ArrayList<>();  
  
 if (allProducts == null || allProducts.isEmpty()) {  
 return;  
 }  
  
 switch (tabPosition) {  
 case 0:  
 filteredProducts.addAll(allProducts);  
 break;  
 case 1:  
 for (Product product : allProducts) {  
 if (product.getCategory().toLowerCase().equals("торты") ||   
 product.getCategory().toLowerCase().equals("торт")) {  
 filteredProducts.add(product);  
 }  
 }  
 break;  
 case 2:  
 for (Product product : allProducts) {  
 if (product.getCategory().toLowerCase().equals("пирожные") ||   
 product.getCategory().toLowerCase().equals("пирожное")) {  
 filteredProducts.add(product);  
 }  
 }  
 break;  
 case 3:  
 for (Product product : allProducts) {  
 if (product.getCategory().toLowerCase().equals("печенье")) {  
 filteredProducts.add(product);  
 }  
 }  
 break;  
 case 4:  
 for (Product product : allProducts) {  
 if (product.getCategory().toLowerCase().equals("хлеб") ||   
 product.getCategory().toLowerCase().equals("выпечка")) {  
 filteredProducts.add(product);  
 }  
 }  
 break;  
 }  
  
 productAdapter.updateProducts(filteredProducts);  
 }  
   
 private void loadProducts() {  
 progressBar.setVisibility(View.VISIBLE);  
   
 new Thread(new Runnable() {  
 @Override  
 public void run() {  
 final List<Product> productList = dbHelper.getAllProducts();  
   
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 progressBar.setVisibility(View.GONE);  
   
 if (productList.isEmpty()) {  
 emptyView.setVisibility(View.VISIBLE);  
 productsRecyclerView.setVisibility(View.GONE);  
 } else {  
 emptyView.setVisibility(View.GONE);  
 productsRecyclerView.setVisibility(View.VISIBLE);  
   
 allProducts = productList;  
   
 if (productAdapter == null) {  
 productAdapter = new com.example.upavlika.adapter.ProductAdapter(MainActivity.this, productList);  
 productAdapter.setOnProductClickListener(MainActivity.this);  
 productsRecyclerView.setAdapter(productAdapter);  
 } else {  
 productAdapter.updateProducts(productList);  
 }  
 }  
 }  
 });  
 }  
 }).start();  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.main\_menu, menu);  
   
 menu.findItem(R.id.action\_admin\_panel).setVisible(  
 currentUser != null && currentUser.getRole() == UserRole.АДМИН);  
   
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
 if (item.getItemId() == R.id.action\_cart) {  
 Intent intent = new Intent(this, CartActivity.class);  
 intent.putExtra("user", currentUser);  
 startActivity(intent);  
 return true;  
 } else if (item.getItemId() == R.id.action\_profile) {  
 Intent intent = new Intent(this, ProfileActivity.class);  
 intent.putExtra("user", currentUser);  
 startActivity(intent);  
 return true;  
 } else if (item.getItemId() == R.id.action\_order\_history) {  
 Intent intent = new Intent(this, OrderHistoryActivity.class);  
 intent.putExtra("user", currentUser);  
 startActivity(intent);  
 return true;  
 } else if (item.getItemId() == R.id.action\_admin\_panel && currentUser.getRole() == UserRole.АДМИН) {  
 Intent intent = new Intent(this, AdminPanelActivity.class);  
 intent.putExtra("user", currentUser);  
 startActivity(intent);  
 return true;  
 } else if (item.getItemId() == R.id.action\_debug\_reinstall\_db && true) {  
 reinstallDatabase();  
 return true;  
 } else if (item.getItemId() == R.id.action\_logout) {  
 currentUser = null;  
 Intent intent = new Intent(this, LoginActivity.class);  
 intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);  
 startActivity(intent);  
 finish();  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
  
 @Override  
 public void onProductClick(Product product) {  
 Intent intent = new Intent(this, ProductDetailActivity.class);  
 intent.putExtra(ProductDetailActivity.EXTRA\_PRODUCT\_ID, product.getId());  
 startActivity(intent);  
 }  
  
 @Override  
 public void onAddToCartClick(Product product) {  
 cart.addProduct(product, 1);  
 Toast.makeText(this, product.getName() + " добавлен в корзину", Toast.LENGTH\_SHORT).show();  
 }  
  
 private void checkAndEnsureAllCategories() {  
 new Thread(new Runnable() {  
 @Override  
 public void run() {  
 List<String> allCategories = dbHelper.getAllCategories();  
   
 List<String> requiredCategories = Arrays.asList("Торты", "Пирожные", "Печенье", "Хлеб", "Кексы", "Пончики");  
   
 for (String category : requiredCategories) {  
 if (!allCategories.contains(category)) {  
 addDemoProductsForCategory(category);  
 Log.d("MainActivity", "Добавлены демо-товары для категории: " + category);  
 } else {  
 List<Product> productsInCategory = dbHelper.getProductsByCategory(category);  
 if (productsInCategory.isEmpty()) {  
 addDemoProductsForCategory(category);  
 Log.d("MainActivity", "Добавлены демо-товары для пустой категории: " + category);  
 }  
 }  
 }  
   
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 loadProducts();  
 }  
 });  
 }  
 }).start();  
 }  
   
 private void addDemoProductsForCategory(String category) {  
 SQLiteDatabase db = dbHelper.getWritableDatabase();  
   
 switch (category) {  
 case "Пирожные":  
 dbHelper.addProductIfNotExists(db, "Эклер с заварным кремом", "Классическое французское пирожное с нежным заварным кремом и шоколадной глазурью", 179.99, 25, "product\_pastry\_eclair", "Пирожные");  
 dbHelper.addProductIfNotExists(db, "Тирамису", "Итальянское пирожное с кофейной пропиткой, сыром маскарпоне и какао", 259.99, 15, "product\_pastry\_tiramisu", "Пирожные");  
 dbHelper.addProductIfNotExists(db, "Наполеон", "Традиционное многослойное пирожное с заварным кремом и хрустящими коржами", 229.99, 20, "product\_pastry\_napoleon", "Пирожные");  
 break;  
   
 case "Хлеб":  
 dbHelper.addProductIfNotExists(db, "Батон белый", "Классический белый хлеб из пшеничной муки высшего сорта", 89.99, 30, "product\_bread\_white", "Хлеб");  
 dbHelper.addProductIfNotExists(db, "Хлеб ржаной", "Традиционный ржаной хлеб с насыщенным вкусом и ароматом", 99.99, 25, "product\_bread\_rye", "Хлеб");  
 dbHelper.addProductIfNotExists(db, "Багет французский", "Хрустящий французский багет с мягким мякишем и хрустящей корочкой", 129.99, 15, "product\_bread\_baguette", "Хлеб");  
 break;  
   
   
 }  
 }  
  
 private void reinstallDatabase() {  
 new Thread(() -> {  
 try {  
 File dbFile = getDatabasePath(DatabaseHelper.DATABASE\_NAME);  
 if (dbFile.exists()) {  
 boolean deleted = dbFile.delete();  
 Log.d("MainActivity", "Database deleted: " + deleted);  
 }  
   
 File walFile = new File(dbFile.getPath() + "-wal");  
 if (walFile.exists()) {  
 walFile.delete();  
 }  
   
 File shmFile = new File(dbFile.getPath() + "-shm");  
 if (shmFile.exists()) {  
 shmFile.delete();  
 }  
   
 runOnUiThread(() -> {  
 Toast.makeText(this, "База данных переустановлена", Toast.LENGTH\_SHORT).show();  
   
 Intent intent = getIntent();  
 finish();  
 startActivity(intent);  
 });  
 } catch (Exception e) {  
 Log.e("MainActivity", "Error reinstalling database", e);  
 runOnUiThread(() -> {  
 Toast.makeText(this, "Ошибка при переустановке базы данных", Toast.LENGTH\_SHORT).show();  
 });  
 }  
 }).start();  
 }  
}

package com.example.upavlika;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.upavlika.adapter.OrderItemAdapter;  
import com.example.upavlika.data.DatabaseHelper;  
import com.example.upavlika.model.Order;  
import com.example.upavlika.model.OrderItem;  
  
import java.text.NumberFormat;  
import java.util.List;  
import java.util.Locale;  
  
public class OrderConfirmationActivity extends AppCompatActivity {  
  
 private DatabaseHelper dbHelper;  
 private Order order;  
 private long orderId;  
 private TextView orderIdTextView;  
 private TextView orderStatusTextView;  
 private TextView orderDateTextView;  
 private TextView orderTotalTextView;  
 private TextView paymentStatusTextView;  
 private Button trackDeliveryButton;  
 private Button downloadReceiptButton;  
 private Button returnToShopButton;  
 private RecyclerView orderItemsRecyclerView;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_order\_confirmation);  
  
   
 dbHelper = DatabaseHelper.getInstance(this);  
  
   
 Toolbar toolbar = findViewById(R.id.toolbar);  
 setSupportActionBar(toolbar);  
 if (getSupportActionBar() != null) {  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setDisplayShowHomeEnabled(true);  
 getSupportActionBar().setTitle("Подтверждение заказа");  
 }  
  
   
 initViews();  
  
   
 orderId = getIntent().getLongExtra("orderId", -1);  
 if (orderId != -1) {  
   
 loadOrderDetails();  
 } else {  
 finish();   
 }  
 }  
  
 private void initViews() {  
 orderIdTextView = findViewById(R.id.order\_id\_text\_view);  
 orderStatusTextView = findViewById(R.id.order\_status\_text\_view);  
 orderDateTextView = findViewById(R.id.order\_date\_text\_view);  
 orderTotalTextView = findViewById(R.id.order\_total\_text\_view);  
 paymentStatusTextView = findViewById(R.id.payment\_status\_text\_view);  
 trackDeliveryButton = findViewById(R.id.track\_delivery\_button);  
 downloadReceiptButton = findViewById(R.id.download\_receipt\_button);  
 returnToShopButton = findViewById(R.id.return\_to\_shop\_button);  
 orderItemsRecyclerView = findViewById(R.id.order\_items\_recycler\_view);  
  
   
 trackDeliveryButton.setOnClickListener(v -> openDeliveryTracking());  
 downloadReceiptButton.setOnClickListener(v -> downloadOrderReceipt());  
 returnToShopButton.setOnClickListener(v -> returnToMainActivity());  
 }  
  
 private void loadOrderDetails() {  
   
 order = dbHelper.getOrder(orderId);  
   
 if (order != null) {  
   
 List<OrderItem> orderItems = order.getOrderItems();  
   
   
 orderIdTextView.setText("Заказ №" + order.getOrderNumber());  
 orderStatusTextView.setText("Статус: " + order.getStatus());  
 orderDateTextView.setText("Дата: " + order.getFormattedDate());  
   
   
 NumberFormat currencyFormat = NumberFormat.getCurrencyInstance(new Locale("ru", "RU"));  
 orderTotalTextView.setText("Сумма: " + currencyFormat.format(order.getTotalAmount()));  
   
   
 paymentStatusTextView.setText(order.isPaid() ? "Оплачен" : "Не оплачен");  
 paymentStatusTextView.setTextColor(order.isPaid() ?   
 getResources().getColor(R.color.colorSuccess) :   
 getResources().getColor(R.color.colorWarning));  
   
   
 setupRecyclerView(orderItems);  
   
   
 updateTrackButtonVisibility();  
 }  
 }  
   
 private void setupRecyclerView(List<OrderItem> items) {  
 if (orderItemsRecyclerView != null && items != null) {  
 OrderItemAdapter adapter = new OrderItemAdapter(this, items);  
 orderItemsRecyclerView.setLayoutManager(new LinearLayoutManager(this));  
 orderItemsRecyclerView.setAdapter(adapter);  
 }  
 }  
   
 private void updateTrackButtonVisibility() {  
 if (order != null) {  
 String status = order.getStatus();  
   
 if ("Отправлен".equals(status) || "В пути".equals(status)) {  
 trackDeliveryButton.setVisibility(View.VISIBLE);  
 } else {  
 trackDeliveryButton.setVisibility(View.GONE);  
 }  
 }  
 }  
   
 private void openDeliveryTracking() {  
 if (order != null) {  
 Intent intent = new Intent(this, DeliveryTrackingActivity.class);  
 intent.putExtra("orderId", order.getId());  
 intent.putExtra("orderNumber", order.getOrderNumber());  
 startActivity(intent);  
 }  
 }  
   
 private void downloadOrderReceipt() {  
   
 if (order != null) {  
   
 Intent intent = new Intent(this, CartActivity.class);  
 intent.putExtra("DOWNLOAD\_RECEIPT\_FOR\_ORDER\_ID", order.getId());  
 startActivity(intent);  
 finish();  
 }  
 }  
   
 private void returnToMainActivity() {  
 Intent intent = new Intent(this, MainActivity.class);  
 intent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP);  
 startActivity(intent);  
 finish();  
 }  
  
 @Override  
 public boolean onSupportNavigateUp() {  
 onBackPressed();  
 return true;  
 }  
}

package com.example.upavlika;  
  
import android.Manifest;  
import android.app.AlertDialog;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.net.Uri;  
import android.os.Build;  
import android.os.Bundle;  
import android.os.Environment;  
import android.os.Handler;  
import android.os.Looper;  
import android.provider.Settings;  
import android.text.Editable;  
import android.text.TextWatcher;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
import androidx.core.app.ActivityCompat;  
import androidx.core.content.ContextCompat;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.upavlika.adapter.OrderAdapter;  
import com.example.upavlika.data.DatabaseHelper;  
import com.example.upavlika.model.Order;  
import com.example.upavlika.model.OrderItem;  
import com.example.upavlika.model.User;  
import com.example.upavlika.model.UserRole;  
import com.example.upavlika.util.OrderFileManager;  
import com.example.upavlika.util.UserManager;  
  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Date;  
import java.util.List;  
import java.util.Locale;  
import java.text.SimpleDateFormat;  
  
public class OrderHistoryActivity extends AppCompatActivity implements OrderAdapter.OnOrderClickListener {  
 private RecyclerView recyclerView;  
 private TextView textViewNoOrders;  
 private OrderAdapter adapter;  
 private OrderFileManager orderFileManager;  
 private static final int STORAGE\_PERMISSION\_CODE = 102;  
 private User currentUser;  
 private static final int REQUEST\_CODE\_SAVE\_ORDER = 1001;  
 private Order currentOrderToSave;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 try {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_order\_history);  
  
   
 Toolbar toolbar = findViewById(R.id.toolbar);  
 setSupportActionBar(toolbar);  
 if (getSupportActionBar() != null) {  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setTitle("История заказов");  
 }  
  
   
 recyclerView = findViewById(R.id.recyclerViewOrders);  
 textViewNoOrders = findViewById(R.id.textViewNoOrders);  
  
   
 UserManager userManager = UserManager.getInstance(this);  
 currentUser = userManager.getCurrentUser();  
 if (currentUser == null) {  
 Intent intent = new Intent(this, LoginActivity.class);  
 startActivity(intent);  
 finish();  
 return;  
 }  
  
   
 recyclerView.setLayoutManager(new LinearLayoutManager(this));  
 adapter = new OrderAdapter(new ArrayList<>(), this);  
 recyclerView.setAdapter(adapter);  
  
   
 orderFileManager = new OrderFileManager(this);  
   
   
 loadOrdersFromDatabase();  
   
   
 checkStoragePermissions();  
   
 } catch (Exception e) {  
   
 }  
 }  
  
 private void checkStoragePermissions() {  
 try {  
   
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.UPSIDE\_DOWN\_CAKE) {   
   
 if (ContextCompat.checkSelfPermission(this, Manifest.permission.READ\_MEDIA\_IMAGES) != PackageManager.PERMISSION\_GRANTED) {  
 ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.READ\_MEDIA\_IMAGES}, STORAGE\_PERMISSION\_CODE);  
 return;  
 }  
 } else if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.R) {   
   
 if (!Environment.isExternalStorageManager()) {  
 Intent intent = new Intent(Settings.ACTION\_MANAGE\_APP\_ALL\_FILES\_ACCESS\_PERMISSION);  
 Uri uri = Uri.fromParts("package", getPackageName(), null);  
 intent.setData(uri);  
 startActivity(intent);  
 return;  
 }  
 } else if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.Q) {   
   
 if (ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE\_EXTERNAL\_STORAGE) != PackageManager.PERMISSION\_GRANTED) {  
 ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.WRITE\_EXTERNAL\_STORAGE}, STORAGE\_PERMISSION\_CODE);  
 return;  
 }  
 } else {  
   
 if (ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE\_EXTERNAL\_STORAGE) != PackageManager.PERMISSION\_GRANTED) {  
 ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.WRITE\_EXTERNAL\_STORAGE}, STORAGE\_PERMISSION\_CODE);  
 return;  
 }  
 }  
   
   
 loadOrders();  
 } catch (Exception e) {  
   
 loadOrdersFromDatabase();  
 }  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.main\_menu, menu);  
   
 try {  
   
 MenuItem orderHistoryItem = menu.findItem(R.id.action\_order\_history);  
 if (orderHistoryItem != null) {  
 orderHistoryItem.setVisible(false);  
 }  
   
   
 MenuItem adminPanelItem = menu.findItem(R.id.action\_admin\_panel);  
 if (adminPanelItem != null && currentUser != null) {  
 adminPanelItem.setVisible(currentUser.getRole() == UserRole.АДМИН);  
 }  
 } catch (Exception e) {  
   
 }  
   
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 try {  
 int id = item.getItemId();  
 if (id == android.R.id.home) {  
 onBackPressed();  
 return true;  
 } else if (id == R.id.action\_cart) {  
   
 try {  
 Intent cartIntent = new Intent(this, CartActivity.class);  
 startActivity(cartIntent);  
 } catch (Exception e) {  
   
 }  
 return true;  
 } else if (id == R.id.action\_profile) {  
 Intent intent = new Intent(this, ProfileActivity.class);  
 startActivity(intent);  
 return true;  
 } else if (id == R.id.action\_admin\_panel && currentUser.getRole() == UserRole.АДМИН) {  
 Intent intent = new Intent(this, AdminPanelActivity.class);  
 startActivity(intent);  
 return true;  
 } else if (id == R.id.action\_logout) {  
 UserManager.getInstance(this).logout();  
 Intent intent = new Intent(this, LoginActivity.class);  
 intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);  
 startActivity(intent);  
 return true;  
 }  
 } catch (Exception e) {  
   
 }  
 return super.onOptionsItemSelected(item);  
 }  
  
   
 private void loadOrdersFromDatabase() {  
 List<Order> dbOrders = new ArrayList<>();  
 List<Order> validOrders = new ArrayList<>();  
   
 try {  
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(this);  
 long userId = UserManager.getInstance(this).getCurrentUserId();  
   
   
 if (currentUser.getRole() == UserRole.АДМИН) {  
 dbOrders = dbHelper.getAllOrders();  
 } else if (userId != -1) {  
 dbOrders = dbHelper.getUserOrders(userId);  
 }  
   
 if (dbOrders != null && !dbOrders.isEmpty()) {  
   
 for (Order order : dbOrders) {  
   
 if (order.getOrderNumber() != null && !order.getOrderNumber().isEmpty() &&  
 order.getTotalAmount() > 0) {  
 validOrders.add(order);  
 }  
 }  
   
   
 Collections.sort(validOrders, (o1, o2) -> Long.compare(o2.getId(), o1.getId()));  
   
 if (!validOrders.isEmpty()) {  
 adapter.updateOrders(validOrders);  
 textViewNoOrders.setVisibility(View.GONE);  
 recyclerView.setVisibility(View.VISIBLE);  
 } else {  
 if (currentUser.getRole() == UserRole.АДМИН) {  
 textViewNoOrders.setText("Заказы не найдены");  
 } else {  
 textViewNoOrders.setText("У вас пока нет заказов");  
 }  
 textViewNoOrders.setVisibility(View.VISIBLE);  
 recyclerView.setVisibility(View.GONE);  
 }  
 } else {  
 if (currentUser.getRole() == UserRole.АДМИН) {  
 textViewNoOrders.setText("Заказы не найдены");  
 } else {  
 textViewNoOrders.setText("У вас пока нет заказов");  
 }  
 textViewNoOrders.setVisibility(View.VISIBLE);  
 recyclerView.setVisibility(View.GONE);  
 }  
 } catch (Exception e) {  
 textViewNoOrders.setText("Ошибка при загрузке заказов");  
 textViewNoOrders.setVisibility(View.VISIBLE);  
 recyclerView.setVisibility(View.GONE);  
 }  
 }  
  
   
 private void loadOrders() {  
   
 List<Order> allOrders = new ArrayList<>();  
   
   
 long userId = UserManager.getInstance(this).getCurrentUserId();  
 boolean isAdmin = (currentUser != null && currentUser.getRole() == UserRole.АДМИН);  
   
   
 adapter.setAdmin(isAdmin);  
   
 try {  
   
 List<Order> fileOrders = orderFileManager.getAllOrders();  
 if (fileOrders != null && !fileOrders.isEmpty()) {  
   
 if (isAdmin) {  
   
 for (Order order : fileOrders) {  
   
 if (order.getOrderNumber() != null && !order.getOrderNumber().isEmpty() &&  
 order.getTotalAmount() > 0) {  
 allOrders.add(order);  
 }  
 }  
 } else {  
 for (Order order : fileOrders) {  
   
 if (order.getUserId() == userId &&   
 order.getOrderNumber() != null && !order.getOrderNumber().isEmpty() &&  
 order.getTotalAmount() > 0) {  
 allOrders.add(order);  
 }  
 }  
 }  
 }  
 } catch (Exception e) {  
   
 }  
   
 try {  
   
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(this);  
   
   
 List<Order> dbOrders;  
 if (isAdmin) {  
 dbOrders = dbHelper.getAllOrders();  
 } else {  
 dbOrders = dbHelper.getUserOrders(userId);  
 }  
   
 if (dbOrders != null && !dbOrders.isEmpty()) {  
   
 for (Order dbOrder : dbOrders) {  
   
 if (dbOrder.getOrderNumber() == null || dbOrder.getOrderNumber().isEmpty()) {  
 continue;   
 }  
   
   
 boolean exists = false;  
 for (Order existingOrder : allOrders) {  
 if (existingOrder.getId() == dbOrder.getId()) {  
 exists = true;  
 break;  
 }  
 }  
 if (!exists) {  
 allOrders.add(dbOrder);  
 }  
 }  
 }  
 } catch (Exception e) {  
   
 }  
   
   
 if (!allOrders.isEmpty()) {  
 Collections.sort(allOrders, (o1, o2) -> Long.compare(o2.getId(), o1.getId()));  
 }  
   
   
 if (!allOrders.isEmpty()) {  
 adapter.updateOrders(allOrders);  
 textViewNoOrders.setVisibility(View.GONE);  
 recyclerView.setVisibility(View.VISIBLE);  
 } else {  
 textViewNoOrders.setText(isAdmin ? "Заказы не найдены" : "У вас пока нет заказов");  
 textViewNoOrders.setVisibility(View.VISIBLE);  
 recyclerView.setVisibility(View.GONE);  
 }  
 }  
  
 @Override  
 public void onOrderClick(Order order) {  
   
 showOrderDetailsDialog(order);  
 }  
  
   
 private void showOrderDetailsDialog(Order order) {  
 try {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
   
   
 boolean isAdmin = (currentUser != null && currentUser.getRole() == UserRole.АДМИН);  
 String userName = "";  
   
 if (isAdmin) {  
   
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(this);  
 User orderUser = dbHelper.getUserById(order.getUserId());  
 if (orderUser != null) {  
 userName = orderUser.getName();  
 } else {  
 userName = "Неизвестный пользователь";  
 }  
 }  
   
   
 StringBuilder content = new StringBuilder();  
 content.append("Заказ №: ").append(order.getOrderNumber());  
   
 if (isAdmin) {  
 content.append(" (ID: ").append(order.getId()).append(")");  
 content.append("\nКлиент: ").append(userName);  
 }  
   
 content.append("\nДата: ").append(order.getFormattedDate());  
   
   
 String statusDisplay = "Неизвестный";  
 try {  
 if (order.getOrderStatus() != null) {  
 statusDisplay = order.getOrderStatus().getDisplayName();  
 } else if (order.getStatus() != null && !order.getStatus().isEmpty()) {  
 statusDisplay = order.getStatus();  
 }  
 } catch (Exception e) {  
   
 }  
 content.append("\nСтатус: ").append(statusDisplay);  
   
   
 String address = order.getDeliveryAddress();  
 if (address != null && !address.isEmpty()) {  
 content.append("\nАдрес доставки: ").append(address);  
 }  
   
 content.append("\n\nТовары:");  
   
 List<OrderItem> items = order.getOrderItems();  
 if (items != null && !items.isEmpty()) {  
 for (OrderItem item : items) {  
 if (item != null) {  
 content.append("\n- ").append(item.getProductName())  
 .append(" x").append(item.getQuantity())  
 .append(", ").append(String.format("%.2f", item.getPrice())).append(" ₽");  
 }  
 }  
 } else {  
 content.append("\n(Информация о товарах недоступна)");  
 }  
   
 content.append("\n\nСумма заказа: ").append(order.getFormattedTotal()).append(" ₽");  
   
 builder.setTitle("Детали заказа")  
 .setMessage(content.toString())  
 .setPositiveButton("Закрыть", null);  
   
   
 boolean hasAction = false;   
   
   
 if (isAdmin) {  
 builder.setNeutralButton("Изменить статус", (dialog, which) -> {  
 showChangeStatusDialog(order);  
 });  
 hasAction = true;  
 }  
   
   
 if (!order.isPaid() && "Ожидает оплаты".equals(statusDisplay)) {  
 builder.setNegativeButton("Оплатить", (dialog, which) -> {  
 payOrder(order);  
 });  
 hasAction = true;  
 }   
   
 else if ("Отправлен".equals(statusDisplay)) {  
 builder.setNegativeButton("Отследить доставку", (dialog, which) -> {  
 openDeliveryTracking(order);  
 });  
 hasAction = true;  
 }  
   
   
 else if ("Отменен".equals(statusDisplay) && !isAdmin) {  
 String cancellationReason = order.getCancellationReason();  
 if (cancellationReason != null && !cancellationReason.isEmpty()) {  
 builder.setNegativeButton("Причина отмены", (dialog, which) -> {  
 showCancellationReasonInfo(order);  
 });  
 hasAction = true;  
 }  
 }  
   
 builder.create().show();  
 } catch (Exception e) {  
   
 }  
 }  
  
   
 private void showChangeStatusDialog(Order order) {  
 try {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Изменить статус заказа");  
   
   
 final String[] statuses = {  
 "Ожидает оплаты", "Оплачен", "Обработка",   
 "Отправлен", "Доставлен", "Отменен"  
 };  
   
 builder.setItems(statuses, (dialog, which) -> {  
 String newStatus = statuses[which];  
   
   
 if ("Отменен".equals(newStatus)) {  
 showCancellationReasonDialog(order);  
 } else {  
 updateOrderStatus(order, newStatus);  
 }  
 });  
   
 builder.setNegativeButton("Отмена", null);  
 builder.create().show();  
 } catch (Exception e) {  
   
 }  
 }  
  
   
 private void showCancellationReasonDialog(Order order) {  
 try {  
   
 final android.widget.EditText inputReason = new android.widget.EditText(this);  
 inputReason.setHint("Введите причину отмены заказа");  
 inputReason.setSingleLine(false);  
 inputReason.setLines(3);  
   
   
 int paddingPx = (int) (16 \* getResources().getDisplayMetrics().density);  
 inputReason.setPadding(paddingPx, paddingPx, paddingPx, paddingPx);  
   
   
 new AlertDialog.Builder(this)  
 .setTitle("Причина отмены заказа")  
 .setMessage("Укажите причину отмены заказа. Эта информация будет доступна клиенту.")  
 .setView(inputReason)  
 .setPositiveButton("Сохранить", (dialog, which) -> {  
 String reason = inputReason.getText().toString().trim();  
 if (reason.isEmpty()) {  
 return;  
 }  
   
   
 updateOrderStatusWithReason(order, "Отменен", reason);  
 })  
 .setNegativeButton("Отмена", null)  
 .show();  
 } catch (Exception e) {  
   
 }  
 }  
   
   
 private void updateOrderStatusWithReason(Order order, String newStatus, String reason) {  
 try {  
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(this);  
 boolean updated = dbHelper.updateOrderStatusWithReason(order.getId(), newStatus, reason);  
   
 if (updated) {  
   
 loadOrders();  
 }  
 } catch (Exception e) {  
   
 }  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
   
   
 if (checkStoragePermission()) {  
 loadOrders();  
 } else {  
 loadOrdersFromDatabase();  
 }  
 }  
   
   
 private boolean checkStoragePermission() {  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.R) {  
 return Environment.isExternalStorageManager();  
 } else {  
 return ContextCompat.checkSelfPermission(this,   
 Manifest.permission.WRITE\_EXTERNAL\_STORAGE) == PackageManager.PERMISSION\_GRANTED;  
 }  
 }  
  
 @Override  
 public boolean onSupportNavigateUp() {  
 onBackPressed();  
 return true;  
 }  
  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (requestCode == STORAGE\_PERMISSION\_CODE) {  
 if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION\_GRANTED) {  
 loadOrders();  
 } else {  
 loadOrdersFromDatabase();  
 }  
 }  
 }  
  
   
 private void openDeliveryTracking(Order order) {  
 try {  
 Intent intent = new Intent(this, DeliveryTrackingActivity.class);  
 intent.putExtra("orderId", order.getId());  
 intent.putExtra("orderNumber", order.getOrderNumber());  
 startActivity(intent);  
 } catch (Exception e) {  
   
 }  
 }  
  
   
 private void showCancellationReasonInfo(Order order) {  
 try {  
 String reason = order.getCancellationReason();  
 if (reason == null || reason.isEmpty()) {  
 reason = "Причина отмены не указана";  
 }  
   
 new AlertDialog.Builder(this)  
 .setTitle("Причина отмены заказа")  
 .setMessage(reason)  
 .setPositiveButton("Закрыть", null)  
 .show();  
 } catch (Exception e) {  
   
 }  
 }  
  
   
 private void payOrder(Order order) {  
 try {  
   
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(this);  
 User user = dbHelper.getUserById(currentUser.getId());  
   
 if (user == null) {  
 return;  
 }  
   
 if (!user.canAfford(order.getTotalAmount())) {  
 return;  
 }  
   
   
 new AlertDialog.Builder(this)  
 .setTitle("Подтверждение оплаты")  
 .setMessage("С вашего баланса будет списано " + order.getFormattedTotal() + " ₽. Продолжить?")  
 .setPositiveButton("Оплатить", (dialog, which) -> {  
   
 double newBalance = user.getBalance() - order.getTotalAmount();  
 boolean updated = dbHelper.updateUserBalance(user.getId(), newBalance);  
   
 if (updated) {  
   
 order.setPaid(true);  
 dbHelper.updateOrderPaymentStatus(order.getId(), true);  
 dbHelper.updateOrderStatus(order.getId(), "Оплачен");  
   
   
 scheduleOrderShipping(order);  
   
   
 loadOrders();  
 }  
 })  
 .setNegativeButton("Отмена", null)  
 .show();  
 } catch (Exception e) {  
   
 }  
 }  
   
   
 private void scheduleOrderShipping(Order order) {  
   
 AlertDialog progressDialog = new AlertDialog.Builder(this)  
 .setTitle("Подготовка к отправке")  
 .setMessage("Ваш заказ готовится к отправке. Пожалуйста, подождите...")  
 .setCancelable(false)  
 .create();  
 progressDialog.show();  
   
   
 new Handler(Looper.getMainLooper()).postDelayed(() -> {  
   
 progressDialog.dismiss();  
   
   
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(this);  
 if (dbHelper != null) {  
 boolean updated = dbHelper.updateOrderStatus(order.getId(), "Отправлен");  
   
 if (updated) {  
   
 new AlertDialog.Builder(this)  
 .setTitle("Заказ отправлен")  
 .setMessage("Ваш заказ №" + order.getOrderNumber() + " успешно отправлен! Вы можете отслеживать его статус в истории заказов.")  
 .setPositiveButton("Отследить доставку", (dialog, which) -> {  
   
 openDeliveryTracking(order);  
 })  
 .setNegativeButton("ОК", (dialog, which) -> {  
   
 loadOrders();  
 })  
 .setCancelable(false)  
 .show();  
 } else {  
 loadOrders();  
 }  
 } else {  
 loadOrders();  
 }  
 }, 5000);   
 }  
  
   
 private void updateOrderStatus(Order order, String newStatus) {  
 try {  
 DatabaseHelper dbHelper = DatabaseHelper.getInstance(this);  
 boolean updated = dbHelper.updateOrderStatus(order.getId(), newStatus);  
   
 if (updated) {  
   
 loadOrders();  
 }  
 } catch (Exception e) {  
   
 }  
 }  
  
   
 private void saveOrderToTextFile(Order order) {  
 try {  
   
 if (!checkStoragePermission()) {  
 checkStoragePermissions();  
 return;  
 }  
  
   
 if (orderFileManager != null) {  
 orderFileManager.saveOrderToTextFile(order);  
 }  
 } catch (Exception e) {  
   
 }  
 }  
}

package com.example.upavlika;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Product {  
 private final int id;  
 private final String name;  
 private final String description;  
 private final String ingredients;  
 private final String nutritionInfo;  
 private final double price;  
 private final int imageResourceId;  
 private final String category;   
  
 public Product(int id, String name, String description, String ingredients,   
 String nutritionInfo, double price, int imageResourceId, String category) {  
 this.id = id;  
 this.name = name;  
 this.description = description;  
 this.ingredients = ingredients;  
 this.nutritionInfo = nutritionInfo;  
 this.price = price;  
 this.imageResourceId = imageResourceId;  
 this.category = category;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public String getIngredients() {  
 return ingredients;  
 }  
  
 public String getNutritionInfo() {  
 return nutritionInfo;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public int getImageResourceId() {  
 return imageResourceId;  
 }  
  
 public String getCategory() {  
 return category;  
 }  
  
   
 public static List<Product> getSampleProducts() {  
 List<Product> products = new ArrayList<>();  
   
   
 products.add(new Product(  
 1,  
 "Шоколадный торт",  
 "Нежный шоколадный торт с кремовой начинкой и шоколадной глазурью. Идеально подходит для праздников и особых случаев.",  
 "Мука пшеничная, яйца, сахар, масло сливочное, какао, шоколад, сливки, ваниль",  
 "Калории: 350 ккал на 100г\nБелки: 5г\nЖиры: 18г\nУглеводы: 42г",  
 1299.0,  
 R.drawable.ic\_cake,  
 "торты"  
 ));  
   
 products.add(new Product(  
 2,  
 "Медовик",  
 "Классический многослойный торт с медовыми коржами и нежным сметанным кремом.",  
 "Мука пшеничная, мед, яйца, сахар, сметана, масло сливочное",  
 "Калории: 320 ккал на 100г\nБелки: 4г\nЖиры: 15г\nУглеводы: 45г",  
 1199.0,  
 R.drawable.ic\_cake,  
 "торты"  
 ));  
   
 products.add(new Product(  
 3,  
 "Наполеон",  
 "Торт из множества слоев тонкого слоеного теста с заварным кремом.",  
 "Мука пшеничная, масло сливочное, яйца, молоко, сахар, ваниль",  
 "Калории: 380 ккал на 100г\nБелки: 4.5г\nЖиры: 23г\nУглеводы: 38г",  
 1350.0,  
 R.drawable.ic\_cake,  
 "торты"  
 ));  
   
   
 products.add(new Product(  
 4,  
 "Эклеры (набор)",  
 "Ассорти из 6 эклеров с разными начинками: шоколадной, ванильной и кофейной.",  
 "Мука пшеничная, яйца, сливки, масло сливочное, сахар, шоколад, кофе, ваниль",  
 "Калории: 300 ккал на 100г\nБелки: 5г\nЖиры: 17г\nУглеводы: 32г",  
 899.0,  
 R.drawable.ic\_cake,  
 "пирожные"  
 ));  
   
 products.add(new Product(  
 5,  
 "Макаронс (набор)",  
 "Набор из 8 французских миндальных пирожных с различными вкусами.",  
 "Миндальная мука, сахарная пудра, яичные белки, сливки, ароматизаторы",  
 "Калории: 270 ккал на 100г\nБелки: 6г\nЖиры: 15г\nУглеводы: 28г",  
 950.0,  
 R.drawable.ic\_cake,  
 "пирожные"  
 ));  
   
   
 products.add(new Product(  
 6,  
 "Шоколадное печенье",  
 "Рассыпчатое печенье с кусочками шоколада.",  
 "Мука пшеничная, масло сливочное, сахар, яйца, шоколад, ваниль",  
 "Калории: 450 ккал на 100г\nБелки: 4г\nЖиры: 22г\nУглеводы: 55г",  
 450.0,  
 R.drawable.ic\_cake,  
 "печенье"  
 ));  
   
   
 products.add(new Product(  
 7,  
 "Фруктовый тарт",  
 "Песочная основа с заварным кремом и свежими фруктами.",  
 "Мука пшеничная, масло сливочное, сахар, яйца, молоко, свежие фрукты",  
 "Калории: 270 ккал на 100г\nБелки: 3.5г\nЖиры: 14г\nУглеводы: 33г",  
 750.0,  
 R.drawable.ic\_cake,  
 "выпечка"  
 ));  
   
 products.add(new Product(  
 8,  
 "Багет французский",  
 "Традиционный французский багет с хрустящей корочкой и мягким мякишем.",  
 "Мука пшеничная, вода, соль, дрожжи",  
 "Калории: 240 ккал на 100г\nБелки: 8г\nЖиры: 2г\nУглеводы: 48г",  
 180.0,  
 R.drawable.ic\_cake,  
 "хлеб"  
 ));  
   
 return products;  
 }  
}

package com.example.upavlika;  
  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import java.util.List;  
  
public class ProductAdapter extends RecyclerView.Adapter<ProductAdapter.ProductViewHolder> {  
  
 private List<Product> products;  
 private final OnProductClickListener listener;  
  
   
 public interface OnProductClickListener {  
 void onProductClick(Product product);  
 void onAddToCartClick(Product product);  
 }  
  
 public ProductAdapter(List<Product> products, OnProductClickListener listener) {  
 this.products = products;  
 this.listener = listener;  
 }  
  
 @NonNull  
 @Override  
 public ProductViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.from(parent.getContext())  
 .inflate(R.layout.item\_product, parent, false);  
 return new ProductViewHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull ProductViewHolder holder, int position) {  
 Product product = products.get(position);  
   
 holder.productImageView.setImageResource(product.getImageResourceId());  
 holder.productNameTextView.setText(product.getName());  
 holder.productPriceTextView.setText(String.format("%s ₽", formatPrice(product.getPrice())));  
 holder.productDescriptionTextView.setText(product.getDescription());  
   
   
 holder.itemView.setOnClickListener(v -> {  
 if (listener != null) {  
 listener.onProductClick(product);  
 }  
 });  
   
 holder.addToCartButton.setOnClickListener(v -> {  
 if (listener != null) {  
 listener.onAddToCartClick(product);  
 }  
 });  
 }  
  
 @Override  
 public int getItemCount() {  
 return products.size();  
 }  
   
   
 public void updateProducts(List<Product> newProducts) {  
 products = newProducts;  
 notifyDataSetChanged();  
 }  
   
   
 private String formatPrice(double price) {  
 return String.format("%.0f", price);  
 }  
  
   
 static class ProductViewHolder extends RecyclerView.ViewHolder {  
 ImageView productImageView;  
 TextView productNameTextView;  
 TextView productPriceTextView;  
 TextView productDescriptionTextView;  
 Button addToCartButton;  
  
 public ProductViewHolder(@NonNull View itemView) {  
 super(itemView);  
 productImageView = itemView.findViewById(R.id.productImageView);  
 productNameTextView = itemView.findViewById(R.id.productNameTextView);  
 productPriceTextView = itemView.findViewById(R.id.productPriceTextView);  
 productDescriptionTextView = itemView.findViewById(R.id.productDescriptionTextView);  
 addToCartButton = itemView.findViewById(R.id.addToCartButton);  
 }  
 }  
}

package com.example.upavlika;  
  
import android.os.Bundle;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
  
import com.example.upavlika.data.DatabaseHelper;  
import com.example.upavlika.model.Cart;  
import com.example.upavlika.model.Product;  
  
import java.text.NumberFormat;  
import java.util.Locale;  
  
public class ProductDetailActivity extends AppCompatActivity {  
  
 public static final String EXTRA\_PRODUCT\_ID = "com.example.upavlika.EXTRA\_PRODUCT\_ID";  
  
 private ImageView productDetailImageView;  
 private TextView productDetailNameTextView;  
 private TextView productDetailPriceTextView;  
 private TextView ingredientsTextView;  
 private TextView productDetailDescriptionTextView;  
 private TextView nutritionTextView;  
 private TextView quantityTextView;  
 private Button decreaseButton;  
 private Button increaseButton;  
 private Button addToCartDetailButton;  
   
 private int currentQuantity = 1;  
 private Cart cart;  
 private DatabaseHelper dbHelper;  
 private Product product;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_product\_detail);  
   
   
 cart = Cart.getInstance();  
 dbHelper = DatabaseHelper.getInstance(this);  
   
   
 long productId = getIntent().getLongExtra(EXTRA\_PRODUCT\_ID, -1);  
 if (productId == -1) {  
 Toast.makeText(this, "Ошибка: товар не найден", Toast.LENGTH\_SHORT).show();  
 finish();  
 return;  
 }  
   
   
 product = dbHelper.getProductById(productId);  
 if (product == null) {  
 Toast.makeText(this, "Ошибка: товар не найден", Toast.LENGTH\_SHORT).show();  
 finish();  
 return;  
 }  
   
   
 Toolbar toolbar = findViewById(R.id.toolbar);  
 if (toolbar != null) {  
 setSupportActionBar(toolbar);  
 if (getSupportActionBar() != null) {  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setDisplayShowHomeEnabled(true);  
 getSupportActionBar().setTitle(product.getName());  
 }  
 }  
   
   
 initViews();  
   
   
 displayProductDetails();  
   
   
 setupQuantityButtons();  
   
   
 addToCartDetailButton.setOnClickListener(view -> addToCart());  
 }  
   
 private void initViews() {  
 productDetailImageView = findViewById(R.id.productDetailImageView);  
 productDetailNameTextView = findViewById(R.id.productDetailNameTextView);  
 productDetailPriceTextView = findViewById(R.id.productDetailPriceTextView);  
 ingredientsTextView = findViewById(R.id.ingredientsTextView);  
 productDetailDescriptionTextView = findViewById(R.id.productDetailDescriptionTextView);  
 nutritionTextView = findViewById(R.id.nutritionTextView);  
 quantityTextView = findViewById(R.id.quantityTextView);  
 decreaseButton = findViewById(R.id.decreaseButton);  
 increaseButton = findViewById(R.id.increaseButton);  
 addToCartDetailButton = findViewById(R.id.addToCartDetailButton);  
 }  
   
 private void displayProductDetails() {  
   
 productDetailImageView.setImageResource(product.getImageResourceId());  
   
   
 productDetailNameTextView.setText(product.getName());  
   
   
 NumberFormat formatter = NumberFormat.getCurrencyInstance(new Locale("ru", "RU"));  
 productDetailPriceTextView.setText(formatter.format(product.getPrice()));  
   
 ingredientsTextView.setText(product.getIngredients());  
 productDetailDescriptionTextView.setText(product.getDescription());  
 nutritionTextView.setText(product.getNutritionInfo());  
 }  
   
 private void setupQuantityButtons() {  
   
 quantityTextView.setText(String.valueOf(currentQuantity));  
   
   
 decreaseButton.setOnClickListener(view -> {  
 if (currentQuantity > 1) {  
 currentQuantity--;  
 quantityTextView.setText(String.valueOf(currentQuantity));  
 }  
 });  
   
   
 increaseButton.setOnClickListener(view -> {  
   
 if (currentQuantity < 10) {  
 currentQuantity++;  
 quantityTextView.setText(String.valueOf(currentQuantity));  
 }  
 });  
 }  
   
 private void addToCart() {  
 if (product != null && currentQuantity > 0) {  
   
 cart.addProduct(product, currentQuantity);  
   
   
 String message = String.format(  
 "%s (x%d) добавлен в корзину",  
 product.getName(),   
 currentQuantity  
 );  
   
 Toast.makeText(this, message, Toast.LENGTH\_SHORT).show();  
   
   
 onBackPressed();  
 }  
 }  
   
 @Override  
 public boolean onSupportNavigateUp() {  
 onBackPressed();  
 return true;  
 }  
}

package com.example.upavlika;  
  
import android.app.AlertDialog;  
import android.content.Intent;  
import android.os.Bundle;  
import android.text.InputType;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ProgressBar;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
  
import com.example.upavlika.data.DatabaseHelper;  
import com.example.upavlika.model.User;  
import com.example.upavlika.model.UserProfile;  
import com.example.upavlika.util.UserManager;  
import com.google.android.material.textfield.TextInputEditText;  
  
public class ProfileActivity extends AppCompatActivity {  
  
 private UserManager userManager;  
 private DatabaseHelper dbHelper;  
 private User currentUser;  
 private UserProfile userProfile;  
  
 private TextView userNameTextView;  
 private TextView userEmailTextView;  
 private TextView balanceTextView;  
 private TextInputEditText phoneEditText;  
 private TextInputEditText addressEditText;  
 private TextInputEditText cityEditText;  
 private TextInputEditText postalCodeEditText;  
 private Button saveProfileButton;  
 private Button ordersHistoryButton;  
 private Button topUpBalanceButton;  
 private ProgressBar progressBar;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_profile);  
  
   
 userManager = UserManager.getInstance(this);  
 dbHelper = DatabaseHelper.getInstance(this);  
 currentUser = userManager.getCurrentUser();  
  
 if (currentUser == null) {  
 Toast.makeText(this, "Необходимо войти в систему", Toast.LENGTH\_SHORT).show();  
 startActivity(new Intent(this, LoginActivity.class));  
 finish();  
 return;  
 }  
  
   
 Toolbar toolbar = findViewById(R.id.profileToolbar);  
 setSupportActionBar(toolbar);  
 if (getSupportActionBar() != null) {  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setDisplayShowHomeEnabled(true);  
 getSupportActionBar().setTitle("Профиль");  
 }  
  
   
 userNameTextView = findViewById(R.id.userNameTextView);  
 userEmailTextView = findViewById(R.id.userEmailTextView);  
 balanceTextView = findViewById(R.id.balanceTextView);  
 phoneEditText = findViewById(R.id.phoneEditText);  
 addressEditText = findViewById(R.id.addressEditText);  
 cityEditText = findViewById(R.id.cityEditText);  
 postalCodeEditText = findViewById(R.id.postalCodeEditText);  
 saveProfileButton = findViewById(R.id.saveProfileButton);  
 ordersHistoryButton = findViewById(R.id.ordersHistoryButton);  
 topUpBalanceButton = findViewById(R.id.topUpBalanceButton);  
 progressBar = findViewById(R.id.progressBar);  
  
   
 loadProfileDataFromDB();  
  
   
 saveProfileButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 saveProfileButton.setEnabled(false);  
 progressBar.setVisibility(View.VISIBLE);  
 saveProfileDataToDB();  
 }  
 });  
  
 ordersHistoryButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
   
 if (userManager.getCurrentUserId() == -1) {  
 Toast.makeText(ProfileActivity.this, "Ошибка сессии. Пожалуйста, войдите снова.", Toast.LENGTH\_SHORT).show();  
   
 startActivity(new Intent(ProfileActivity.this, LoginActivity.class));  
 finish();  
 return;  
 }  
 Intent intent = new Intent(ProfileActivity.this, OrderHistoryActivity.class);  
 startActivity(intent);  
 }  
 });  
   
 topUpBalanceButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 showTopUpBalanceDialog();  
 }  
 });  
 }  
  
   
 private void loadProfileDataFromDB() {  
 progressBar.setVisibility(View.VISIBLE);  
  
   
 userNameTextView.setText(currentUser.getName());  
 userEmailTextView.setText(currentUser.getEmail());  
   
   
 updateBalanceText();  
  
   
 new Thread(new Runnable() {  
 @Override  
 public void run() {  
 final UserProfile profile = dbHelper.getUserProfile(currentUser.getId());  
   
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 progressBar.setVisibility(View.GONE);  
   
 if (profile != null) {  
 userProfile = profile;  
   
   
 phoneEditText.setText(profile.getPhone() != null ? profile.getPhone() : "");  
 addressEditText.setText(profile.getAddress() != null ? profile.getAddress() : "");  
 cityEditText.setText(profile.getCity() != null ? profile.getCity() : "");  
 postalCodeEditText.setText(profile.getPostalCode() != null ? profile.getPostalCode() : "");  
 } else {  
   
 userProfile = new UserProfile(currentUser.getName(), currentUser.getEmail());  
 }  
 }  
 });  
 }  
 }).start();  
 }  
   
   
 private void updateBalanceText() {  
   
 currentUser = userManager.getCurrentUser();   
 if (currentUser != null) {  
   
 String formattedBalance = String.format("%.2f ₽", currentUser.getBalance());  
 balanceTextView.setText(formattedBalance);  
 }  
 }  
   
   
 private void showTopUpBalanceDialog() {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Пополнение баланса");  
   
   
 final EditText input = new EditText(this);  
 input.setInputType(InputType.TYPE\_CLASS\_NUMBER | InputType.TYPE\_NUMBER\_FLAG\_DECIMAL);  
 input.setHint("Введите сумму");  
 builder.setView(input);  
   
   
 builder.setPositiveButton("Пополнить", (dialog, which) -> {  
 String amountString = input.getText().toString().trim();  
 if (amountString.isEmpty()) {  
 Toast.makeText(ProfileActivity.this, "Введите сумму", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
   
 try {  
 double amount = Double.parseDouble(amountString);  
 if (amount <= 0) {  
 Toast.makeText(ProfileActivity.this, "Сумма должна быть больше нуля", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
   
   
 topUpBalance(amount);  
 } catch (NumberFormatException e) {  
 Toast.makeText(ProfileActivity.this, "Неверный формат суммы", Toast.LENGTH\_SHORT).show();  
 }  
 });  
   
 builder.setNegativeButton("Отмена", (dialog, which) -> dialog.cancel());  
   
 builder.show();  
 }  
   
   
 private void topUpBalance(final double amount) {  
 progressBar.setVisibility(View.VISIBLE);  
   
 new Thread(() -> {  
   
 User user = dbHelper.getUserById(currentUser.getId());  
 if (user != null) {  
   
 double newBalance = user.getBalance() + amount;  
   
 boolean success = dbHelper.updateUserBalance(user.getId(), newBalance);  
   
 runOnUiThread(() -> {  
 progressBar.setVisibility(View.GONE);  
   
 if (success) {  
   
 currentUser.setBalance(newBalance);  
   
 updateBalanceText();  
   
 Toast.makeText(ProfileActivity.this,   
 "Баланс успешно пополнен на " + String.format("%.2f ₽", amount),   
 Toast.LENGTH\_SHORT).show();  
 } else {  
 Toast.makeText(ProfileActivity.this,   
 "Ошибка при пополнении баланса",   
 Toast.LENGTH\_SHORT).show();  
 }  
 });  
 } else {  
 runOnUiThread(() -> {  
 progressBar.setVisibility(View.GONE);  
 Toast.makeText(ProfileActivity.this,   
 "Ошибка: пользователь не найден",   
 Toast.LENGTH\_SHORT).show();  
 });  
 }  
 }).start();  
 }  
  
   
 private void saveProfileDataToDB() {  
   
 String phone = phoneEditText.getText().toString().trim();  
 String address = addressEditText.getText().toString().trim();  
 String city = cityEditText.getText().toString().trim();  
 String postalCode = postalCodeEditText.getText().toString().trim();  
  
   
 userProfile.setPhone(phone);  
 userProfile.setAddress(address);  
 userProfile.setCity(city);  
 userProfile.setPostalCode(postalCode);  
  
   
 new Thread(new Runnable() {  
 @Override  
 public void run() {  
 final boolean success = dbHelper.saveUserProfile(  
 currentUser.getId(),  
 userProfile.getPhone(),  
 userProfile.getAddress(),  
 userProfile.getCity(),  
 userProfile.getPostalCode()  
 );  
   
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 progressBar.setVisibility(View.GONE);  
 saveProfileButton.setEnabled(true);  
   
 if (success) {  
 Toast.makeText(ProfileActivity.this, "Профиль успешно сохранен", Toast.LENGTH\_SHORT).show();  
 } else {  
 Toast.makeText(ProfileActivity.this, "Ошибка при сохранении профиля", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
 }  
 }).start();  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
 if (item.getItemId() == android.R.id.home) {  
 onBackPressed();  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
}

package com.example.upavlika;  
  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.os.Bundle;  
import android.util.Patterns;  
import android.view.View;  
import android.widget.ArrayAdapter;  
import android.widget.Button;  
import android.widget.LinearLayout;  
import android.widget.RadioButton;  
import android.widget.RadioGroup;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import com.example.upavlika.model.User;  
import com.example.upavlika.model.UserRole;  
import com.example.upavlika.util.UserManager;  
import com.google.android.material.textfield.TextInputEditText;  
import com.google.android.material.textfield.TextInputLayout;  
  
import java.util.regex.Pattern;  
  
public class RegisterActivity extends AppCompatActivity {  
  
 private TextInputLayout nameInputLayout;  
 private TextInputLayout emailInputLayout;  
 private TextInputLayout passwordInputLayout;  
 private TextInputLayout confirmPasswordInputLayout;  
 private TextInputEditText nameEditText;  
 private TextInputEditText emailEditText;  
 private TextInputEditText passwordEditText;  
 private TextInputEditText confirmPasswordEditText;  
 private Button registerButton;  
 private TextView loginTextView;  
 private UserManager userManager;  
 private boolean returnToCart = false;  
 private LinearLayout roleContainer;  
 private RadioGroup roleRadioGroup;  
 private RadioButton customerRadioButton;  
 private RadioButton adminRadioButton;  
 private boolean isAdminRegistering = false;  
  
 // Константа для минимальной длины пароля  
 private static final int MIN\_PASSWORD\_LENGTH = 6;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_register);  
  
   
 if (getIntent().hasExtra("RETURN\_TO\_CART")) {  
 returnToCart = getIntent().getBooleanExtra("RETURN\_TO\_CART", false);  
 }  
   
   
 if (getIntent().hasExtra("IS\_ADMIN\_CREATING")) {  
 isAdminRegistering = getIntent().getBooleanExtra("IS\_ADMIN\_CREATING", false);  
 }  
  
   
 userManager = UserManager.getInstance(this);  
  
 // Инициализация TextInputLayout  
 nameInputLayout = findViewById(R.id.nameInputLayout);  
 emailInputLayout = findViewById(R.id.emailInputLayout);  
 passwordInputLayout = findViewById(R.id.passwordInputLayout);  
 confirmPasswordInputLayout = findViewById(R.id.confirmPasswordInputLayout);  
   
 // Инициализация TextInputEditText  
 nameEditText = findViewById(R.id.nameEditText);  
 emailEditText = findViewById(R.id.emailEditText);  
 passwordEditText = findViewById(R.id.passwordEditText);  
 confirmPasswordEditText = findViewById(R.id.confirmPasswordEditText);  
 registerButton = findViewById(R.id.registerButton);  
 loginTextView = findViewById(R.id.loginTextView);  
 roleContainer = findViewById(R.id.roleContainer);  
 roleRadioGroup = findViewById(R.id.roleRadioGroup);  
 customerRadioButton = findViewById(R.id.customerRadioButton);  
 adminRadioButton = findViewById(R.id.adminRadioButton);  
   
   
 if (isAdminRegistering) {  
 roleContainer.setVisibility(View.VISIBLE);  
 loginTextView.setVisibility(View.GONE);   
 } else {  
 roleContainer.setVisibility(View.GONE);  
 }  
  
   
 registerButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 // Сброс всех ошибок перед валидацией  
 emailInputLayout.setError(null);  
 passwordInputLayout.setError(null);  
 confirmPasswordInputLayout.setError(null);  
   
 String name = nameEditText.getText().toString().trim();  
 String email = emailEditText.getText().toString().trim();  
 String password = passwordEditText.getText().toString().trim();  
 String confirmPassword = confirmPasswordEditText.getText().toString().trim();  
  
 // Проверка на пустые поля  
 boolean hasError = false;  
   
 if (name.isEmpty()) {  
 nameInputLayout.setError("Введите имя");  
 hasError = true;  
 }  
   
 if (email.isEmpty()) {  
 emailInputLayout.setError("Введите email");  
 hasError = true;  
 } else if (!isValidEmail(email)) {  
 emailInputLayout.setError("Введите корректный email");  
 hasError = true;  
 }  
   
 if (password.isEmpty()) {  
 passwordInputLayout.setError("Введите пароль");  
 hasError = true;  
 } else if (password.length() < MIN\_PASSWORD\_LENGTH) {  
 passwordInputLayout.setError("Пароль должен содержать не менее " + MIN\_PASSWORD\_LENGTH + " символов");  
 hasError = true;  
 }  
   
 if (confirmPassword.isEmpty()) {  
 confirmPasswordInputLayout.setError("Подтвердите пароль");  
 hasError = true;  
 } else if (!password.equals(confirmPassword)) {  
 confirmPasswordInputLayout.setError("Пароли не совпадают");  
 hasError = true;  
 }  
   
 if (hasError) {  
 return;  
 }  
  
 // Определение роли пользователя  
 UserRole role = UserRole.ПОЛЬЗОВАТЕЛЬ;   
 if (isAdminRegistering) {  
 role = adminRadioButton.isChecked() ? UserRole.АДМИН : UserRole.ПОЛЬЗОВАТЕЛЬ;  
 }  
  
 // Регистрация пользователя  
 User newUser = userManager.registerUserWithRole(name, email, password, role);  
   
 if (newUser != null) {  
 // Сохранение данных пользователя в любом случае  
 saveUserCredentials(newUser);  
   
 Toast.makeText(RegisterActivity.this, "Пользователь успешно зарегистрирован", Toast.LENGTH\_SHORT).show();  
   
 if (isAdminRegistering) {  
 // Если администратор создаёт аккаунт, просто закрываем экран  
 finish();  
 } else if (returnToCart) {  
 // Если регистрация из корзины, возвращаемся в корзину  
 Intent intent = new Intent(RegisterActivity.this, CartActivity.class);  
 startActivity(intent);  
 finish();  
 } else {  
 // Обычная регистрация - переходим на главный экран  
 // И явно указываем, что это обычный пользователь  
 clearAdminSession();  
 Intent intent = new Intent(RegisterActivity.this, MainActivity.class);  
 intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);  
 startActivity(intent);  
 finish();  
 }  
 } else {  
 Toast.makeText(RegisterActivity.this, "Пользователь с таким email уже существует", Toast.LENGTH\_LONG).show();  
 }  
 }  
 });  
  
 // Обработчик нажатия на ссылку для входа  
 loginTextView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(RegisterActivity.this, LoginActivity.class);  
   
 if (returnToCart) {  
 intent.putExtra("RETURN\_TO\_CART", true);  
 }  
 startActivity(intent);  
 finish();  
 }  
 });  
 }  
   
 /\*\*  
 \* Проверка валидности email  
 \*/  
 private boolean isValidEmail(String email) {  
 return Patterns.EMAIL\_ADDRESS.matcher(email).matches();  
 }  
   
 /\*\*  
 \* Сохранение учетных данных пользователя  
 \*/  
 private void saveUserCredentials(User user) {  
 SharedPreferences sharedPreferences = getSharedPreferences("UserPrefs", MODE\_PRIVATE);  
 SharedPreferences.Editor editor = sharedPreferences.edit();  
   
 editor.putLong("userId", user.getId());  
 editor.putString("userName", user.getName());  
 editor.putString("userEmail", user.getEmail());  
 editor.putString("userRole", user.getRole().name());  
   
 editor.apply();  
 }  
   
 /\*\*  
 \* Очистка сессии админа при регистрации обычного пользователя  
 \*/  
 private void clearAdminSession() {  
 SharedPreferences adminPrefs = getSharedPreferences("AdminPrefs", MODE\_PRIVATE);  
 SharedPreferences.Editor editor = adminPrefs.edit();  
 editor.clear();  
 editor.apply();  
 }  
}

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".AdminPanelActivity">  
  
   
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="?attr/colorPrimary"  
 android:elevation="4dp"  
 android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"  
 app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />  
  
   
 <com.google.android.material.tabs.TabLayout  
 android:id="@+id/tabLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="?attr/colorPrimary"  
 android:elevation="4dp"  
 app:tabGravity="fill"  
 app:tabMode="fixed"  
 app:tabTextColor="@android:color/white" />  
  
   
 <androidx.viewpager2.widget.ViewPager2  
 android:id="@+id/viewPager"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1" />  
  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".CartActivity">  
  
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/cartToolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="@color/colorPrimary"  
 android:elevation="4dp"  
 android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:popupTheme="@style/ThemeOverlay.AppCompat.Light"  
 app:title="Корзина" />  
  
 <TextView  
 android:id="@+id/empty\_cart\_text\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/cart\_empty"  
 android:textSize="18sp"  
 android:visibility="gone"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/cart\_recycler\_view"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:clipToPadding="false"  
 android:padding="8dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/totalLayout"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/cartToolbar"  
 tools:listitem="@layout/item\_cart\_product" />  
  
 <LinearLayout  
 android:id="@+id/totalLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@color/white"  
 android:elevation="4dp"  
 android:orientation="vertical"  
 android:padding="16dp"  
 app:layout\_constraintBottom\_toBottomOf="parent">  
  
 <TextView  
 android:id="@+id/total\_price\_text\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:gravity="end"  
 android:textColor="@color/black"  
 android:textSize="18sp"  
 android:textStyle="bold"  
 tools:text="Итого: 1800 ₽" />  
  
 <Button  
 android:id="@+id/checkout\_button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/colorPrimary"  
 android:padding="12dp"  
 android:text="@string/proceed\_to\_checkout"  
 android:textColor="@color/white" />  
 </LinearLayout>  
  
 <ProgressBar  
 android:id="@+id/progressBar"  
 style="?android:attr/progressBarStyle"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:visibility="gone"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.coordinatorlayout.widget.CoordinatorLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".DeliveryTrackingActivity">  
  
 <com.google.android.material.appbar.AppBarLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:theme="@style/Theme.AppCompat.Light.DarkActionBar">  
  
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="?attr/colorPrimary" />  
  
 </com.google.android.material.appbar.AppBarLayout>  
  
 <androidx.core.widget.NestedScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
   
 <androidx.cardview.widget.CardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="16dp"  
 app:cardCornerRadius="8dp"  
 app:cardElevation="4dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <TextView  
 android:id="@+id/textViewOrderNumber"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textSize="18sp"  
 android:textStyle="bold"  
 android:text="Заказ № \_\_\_" />  
  
 <TextView  
 android:id="@+id/textViewETA"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:textSize="16sp"  
 android:textColor="#4CAF50"  
 android:text="Ожидаемое время доставки: \_\_:\_\_" />  
  
 </LinearLayout>  
  
 </androidx.cardview.widget.CardView>  
  
   
 <androidx.cardview.widget.CardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="16dp"  
 app:cardCornerRadius="8dp"  
 app:cardElevation="4dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:padding="16dp">  
  
 <ImageView  
 android:id="@+id/imageViewDriver"  
 android:layout\_width="80dp"  
 android:layout\_height="80dp"  
 android:src="@android:drawable/ic\_menu\_myplaces"  
 android:scaleType="centerCrop"  
 android:background="#EEEEEE" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:layout\_marginStart="16dp">  
  
 <TextView  
 android:id="@+id/textViewDriverName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 android:text="Курьер: \_\_\_" />  
  
 <TextView  
 android:id="@+id/textViewRating"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="4dp"  
 android:textSize="14sp"  
 android:text="Рейтинг: \_\_\_ ★" />  
  
 <Button  
 android:id="@+id/buttonChat"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="4dp"  
 android:text="Чат с курьером"  
 android:textSize="12sp" />  
  
 </LinearLayout>  
  
 </LinearLayout>  
  
 </androidx.cardview.widget.CardView>  
  
   
 <androidx.cardview.widget.CardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="16dp"  
 app:cardCornerRadius="8dp"  
 app:cardElevation="4dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:padding="16dp">  
  
 <ImageView  
 android:id="@+id/imageViewVehicle"  
 android:layout\_width="120dp"  
 android:layout\_height="80dp"  
 android:src="@android:drawable/ic\_menu\_directions"  
 android:scaleType="centerCrop"  
 android:background="#EEEEEE" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:layout\_marginStart="16dp">  
  
 <TextView  
 android:id="@+id/textViewVehicleInfo"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textSize="14sp"  
 android:text="Автомобиль: \_\_\_\nНомер: \_\_\_" />  
  
 </LinearLayout>  
  
 </LinearLayout>  
  
 </androidx.cardview.widget.CardView>  
  
 </LinearLayout>  
  
 </androidx.core.widget.NestedScrollView>  
  
</androidx.coordinatorlayout.widget.CoordinatorLayout>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:padding="16dp"  
 tools:context=".LoginActivity">  
  
 <ImageView  
 android:id="@+id/logoImageView"  
 android:layout\_width="120dp"  
 android:layout\_height="120dp"  
 android:layout\_marginTop="48dp"  
 android:contentDescription="@string/app\_logo"  
 android:src="@drawable/ic\_universal"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/titleTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="@string/login\_title"  
 android:textSize="24sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/logoImageView" />  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/emailInputLayout"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="24dp"  
 android:hint="@string/email\_hint"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/titleTextView">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/emailEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textEmailAddress" />  
  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/passwordInputLayout"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:hint="@string/password\_hint"  
 app:endIconMode="password\_toggle"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/emailInputLayout">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/passwordEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPassword" />  
  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <Button  
 android:id="@+id/loginButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="24dp"  
 android:padding="12dp"  
 android:text="@string/login\_button"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/passwordInputLayout" />  
  
 <TextView  
 android:id="@+id/forgotPasswordTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:text="@string/forgot\_password"  
 android:textColor="@android:color/holo\_blue\_dark"  
 android:textSize="14sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/loginButton" />  
  
 <TextView  
 android:id="@+id/registerTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="@string/register\_prompt"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/forgotPasswordTextView" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <com.google.android.material.appbar.AppBarLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">  
  
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="?attr/colorPrimary"  
 app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />  
  
 <com.google.android.material.tabs.TabLayout  
 android:id="@+id/categoriesTabLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:tabMode="scrollable"  
 app:tabGravity="start"  
 app:tabIndicatorColor="@android:color/white" />  
  
 </com.google.android.material.appbar.AppBarLayout>  
  
 <androidx.core.widget.NestedScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:fillViewport="true"  
 app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior">  
  
 <FrameLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/productsRecyclerView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:clipToPadding="false"  
 android:padding="8dp" />  
  
 <TextView  
 android:id="@+id/emptyView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:text="@string/no\_products"  
 android:textSize="18sp"  
 android:visibility="gone" />  
  
 <ProgressBar  
 android:id="@+id/progressBar"  
 style="?android:attr/progressBarStyle"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:visibility="gone" />  
  
 </FrameLayout>  
  
 </androidx.core.widget.NestedScrollView>  
  
 <com.google.android.material.floatingactionbutton.FloatingActionButton  
 android:id="@+id/cartFab"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="bottom|end"  
 android:layout\_margin="16dp"  
 android:contentDescription="@string/cart"  
 app:srcCompat="@android:drawable/ic\_menu\_send" />  
  
</androidx.coordinatorlayout.widget.CoordinatorLayout>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="?attr/colorPrimary"  
 android:elevation="4dp"  
 android:theme="@style/ThemeOverlay.AppCompat.ActionBar"  
 app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />  
  
 <ScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="Спасибо за заказ!"  
 android:textSize="24sp"  
 android:textStyle="bold" />  
  
 <TextView  
 android:id="@+id/order\_id\_text\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:gravity="center"  
 android:textSize="18sp" />  
  
 <androidx.cardview.widget.CardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 app:cardCornerRadius="8dp"  
 app:cardElevation="4dp"  
 app:contentPadding="16dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Информация о заказе"  
 android:textSize="18sp"  
 android:textStyle="bold" />  
  
 <TextView  
 android:id="@+id/order\_date\_text\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:textSize="16sp" />  
  
 <TextView  
 android:id="@+id/order\_status\_text\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:textSize="16sp" />  
  
 <TextView  
 android:id="@+id/payment\_status\_text\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:textSize="16sp" />  
  
 <TextView  
 android:id="@+id/order\_total\_text\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:textSize="16sp"  
 android:textStyle="bold" />  
 </LinearLayout>  
 </androidx.cardview.widget.CardView>  
  
 <androidx.cardview.widget.CardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 app:cardCornerRadius="8dp"  
 app:cardElevation="4dp"  
 app:contentPadding="16dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Товары в заказе"  
 android:textSize="18sp"  
 android:textStyle="bold" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/order\_items\_recycler\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp" />  
 </LinearLayout>  
 </androidx.cardview.widget.CardView>  
  
 <Button  
 android:id="@+id/track\_delivery\_button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Отследить доставку"  
 android:visibility="gone" />  
  
 <Button  
 android:id="@+id/download\_receipt\_button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:text="Скачать чек" />  
  
 <Button  
 android:id="@+id/return\_to\_shop\_button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:text="Вернуться в магазин" />  
  
 </LinearLayout>  
  
 </ScrollView>  
  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="?attr/colorPrimary"  
 android:elevation="4dp"  
 android:theme="@style/ThemeOverlay.AppCompat.ActionBar"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:title="История заказов" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/recyclerViewOrders"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:padding="8dp"  
 android:clipToPadding="false"  
 app:layout\_constraintTop\_toBottomOf="@id/toolbar"  
 app:layout\_constraintBottom\_toBottomOf="parent" />  
  
 <TextView  
 android:id="@+id/textViewNoOrders"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Нет заказов"  
 android:visibility="gone"  
 android:textSize="18sp"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".ProductDetailActivity">  
  
 <com.google.android.material.appbar.AppBarLayout  
 android:id="@+id/appBarLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">  
  
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="@color/colorPrimary"  
 app:popupTheme="@style/ThemeOverlay.AppCompat.Light"  
 app:navigationIcon="?attr/homeAsUpIndicator" />  
  
 </com.google.android.material.appbar.AppBarLayout>  
  
 <androidx.core.widget.NestedScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <ImageView  
 android:id="@+id/productImageView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="250dp"  
 android:contentDescription="@string/product\_image"  
 android:scaleType="centerCrop"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 tools:src="@drawable/ic\_cake" />  
  
 <TextView  
 android:id="@+id/productNameTextView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginEnd="16dp"  
 android:textSize="24sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/productImageView"  
 tools:text="Шоколадный торт" />  
  
 <TextView  
 android:id="@+id/productPriceTextView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="16dp"  
 android:textColor="@color/purple\_700"  
 android:textSize="20sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/productNameTextView"  
 tools:text="1299 ₽" />  
  
 <TextView  
 android:id="@+id/categoryLabelTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="16dp"  
 android:text="Категория:"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/productPriceTextView" />  
  
 <TextView  
 android:id="@+id/categoryTextView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginEnd="16dp"  
 android:textSize="16sp"  
 app:layout\_constraintBaseline\_toBaselineOf="@id/categoryLabelTextView"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@id/categoryLabelTextView"  
 tools:text="Торты" />  
  
 <TextView  
 android:id="@+id/stockLabelTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="8dp"  
 android:text="В наличии:"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/categoryLabelTextView" />  
  
 <TextView  
 android:id="@+id/stockTextView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginEnd="16dp"  
 android:textSize="16sp"  
 app:layout\_constraintBaseline\_toBaselineOf="@id/stockLabelTextView"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@id/stockLabelTextView"  
 tools:text="15 шт." />  
  
 <View  
 android:id="@+id/divider"  
 android:layout\_width="0dp"  
 android:layout\_height="1dp"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginEnd="16dp"  
 android:background="@android:color/darker\_gray"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/stockTextView" />  
  
 <TextView  
 android:id="@+id/descriptionLabelTextView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginEnd="16dp"  
 android:text="Описание"  
 android:textSize="18sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/divider" />  
  
 <TextView  
 android:id="@+id/descriptionTextView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="16dp"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/descriptionLabelTextView"  
 tools:text="Нежный шоколадный бисквит с кремом из темного шоколада и вишневым наполнителем. Идеально подходит для праздничных мероприятий и семейных чаепитий." />  
  
 <com.google.android.material.card.MaterialCardView  
 android:id="@+id/orderCardView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="24dp"  
 android:layout\_marginEnd="16dp"  
 android:layout\_marginBottom="24dp"  
 app:cardCornerRadius="8dp"  
 app:cardElevation="4dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/descriptionTextView">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="16dp">  
  
 <TextView  
 android:id="@+id/quantityLabelTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Количество:"  
 android:textSize="16sp"  
 app:layout\_constraintBottom\_toBottomOf="@id/quantityLayout"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="@id/quantityLayout" />  
  
 <LinearLayout  
 android:id="@+id/quantityLayout"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:orientation="horizontal"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@id/quantityLabelTextView"  
 app:layout\_constraintTop\_toTopOf="parent">  
  
 <Button  
 android:id="@+id/decreaseButton"  
 style="@style/Widget.MaterialComponents.Button.OutlinedButton"  
 android:layout\_width="48dp"  
 android:layout\_height="48dp"  
 android:insetTop="0dp"  
 android:insetBottom="0dp"  
 android:text="-"  
 android:textSize="18sp" />  
  
 <TextView  
 android:id="@+id/quantityTextView"  
 android:layout\_width="48dp"  
 android:layout\_height="48dp"  
 android:gravity="center"  
 android:text="1"  
 android:textSize="18sp"  
 android:textStyle="bold" />  
  
 <Button  
 android:id="@+id/increaseButton"  
 style="@style/Widget.MaterialComponents.Button.OutlinedButton"  
 android:layout\_width="48dp"  
 android:layout\_height="48dp"  
 android:insetTop="0dp"  
 android:insetBottom="0dp"  
 android:text="+"  
 android:textSize="18sp" />  
  
 </LinearLayout>  
  
 <Button  
 android:id="@+id/addToCartButton"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Добавить в корзину"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/quantityLayout" />  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
 </com.google.android.material.card.MaterialCardView>  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
 </androidx.core.widget.NestedScrollView>  
  
</androidx.coordinatorlayout.widget.CoordinatorLayout>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".ProfileActivity">  
  
 <com.google.android.material.appbar.AppBarLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/profileToolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="?attr/colorPrimary"  
 app:title="@string/profile"  
 app:titleTextColor="@color/white"  
 app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />  
  
 </com.google.android.material.appbar.AppBarLayout>  
  
 <androidx.core.widget.NestedScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <ProgressBar  
 android:id="@+id/progressBar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="16dp"  
 android:visibility="gone" />  
  
 <ImageView  
 android:id="@+id/profileImageView"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginBottom="16dp"  
 android:contentDescription="@string/profile"  
 android:src="@drawable/ic\_profile"  
 android:tint="@color/purple\_500" />  
  
 <TextView  
 android:id="@+id/userNameTextView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:gravity="center"  
 android:textSize="20sp"  
 android:textStyle="bold"  
 tools:text="Имя пользователя" />  
  
 <TextView  
 android:id="@+id/userEmailTextView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="24dp"  
 android:gravity="center"  
 android:textSize="16sp"  
 tools:text="user@example.com" />  
  
 <com.google.android.material.card.MaterialCardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginBottom="16dp"  
 app:cardCornerRadius="8dp"  
 app:cardElevation="4dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="16dp"  
 android:text="Баланс"  
 android:textSize="18sp"  
 android:textStyle="bold" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:gravity="center\_vertical">  
  
 <TextView  
 android:id="@+id/balanceTextView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:textSize="24sp"  
 android:textStyle="bold"  
 android:textColor="@color/colorPrimary"  
 tools:text="10000.00 ₽" />  
  
 <Button  
 android:id="@+id/topUpBalanceButton"  
 style="@style/Widget.MaterialComponents.Button.OutlinedButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Пополнить" />  
 </LinearLayout>  
  
 </LinearLayout>  
  
 </com.google.android.material.card.MaterialCardView>  
  
 <com.google.android.material.card.MaterialCardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="16dp"  
 app:cardCornerRadius="8dp"  
 app:cardElevation="4dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="16dp"  
 android:text="Контактная информация"  
 android:textSize="18sp"  
 android:textStyle="bold" />  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/phoneInputLayout"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:hint="Телефон">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/phoneEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="phone" />  
  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/addressInputLayout"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:hint="Адрес">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/addressEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPostalAddress" />  
  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/cityInputLayout"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:hint="Город">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/cityEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="text" />  
  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/postalCodeInputLayout"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:hint="Почтовый индекс">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/postalCodeEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="number" />  
  
 </com.google.android.material.textfield.TextInputLayout>  
  
 </LinearLayout>  
  
 </com.google.android.material.card.MaterialCardView>  
  
 <Button  
 android:id="@+id/saveProfileButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Сохранить изменения" />  
  
 <Button  
 android:id="@+id/ordersHistoryButton"  
 style="@style/Widget.MaterialComponents.Button.OutlinedButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:text="История заказов" />  
  
 </LinearLayout>  
  
 </androidx.core.widget.NestedScrollView>  
  
</androidx.coordinatorlayout.widget.CoordinatorLayout>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:padding="16dp"  
 tools:context=".RegisterActivity">  
  
 <ImageView  
 android:id="@+id/logoImageView"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:layout\_marginTop="24dp"  
 android:contentDescription="@string/app\_logo"  
 android:src="@drawable/ic\_universal"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/titleTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="@string/register\_title"  
 android:textSize="24sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/logoImageView" />  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/nameInputLayout"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:hint="@string/name\_hint"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/titleTextView">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/nameEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPersonName" />  
  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/emailInputLayout"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:hint="@string/email\_hint"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/nameInputLayout">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/emailEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textEmailAddress" />  
  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/passwordInputLayout"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:hint="@string/password\_hint"  
 app:endIconMode="password\_toggle"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/emailInputLayout">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/passwordEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPassword" />  
  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/confirmPasswordInputLayout"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:hint="@string/confirm\_password\_hint"  
 app:endIconMode="password\_toggle"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/passwordInputLayout">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/confirmPasswordEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPassword" />  
  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <LinearLayout  
 android:id="@+id/roleContainer"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:orientation="vertical"  
 android:visibility="gone"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/confirmPasswordInputLayout">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:text="Роль пользователя:"  
 android:textSize="16sp"  
 android:textStyle="bold" />  
  
 <RadioGroup  
 android:id="@+id/roleRadioGroup"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp">  
  
 <RadioButton  
 android:id="@+id/customerRadioButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Покупатель"  
 android:textSize="16sp"  
 android:checked="true"  
 android:padding="8dp"/>  
  
 <RadioButton  
 android:id="@+id/adminRadioButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Администратор"  
 android:textSize="16sp"  
 android:padding="8dp"/>  
 </RadioGroup>  
 </LinearLayout>  
  
 <Button  
 android:id="@+id/registerButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="24dp"  
 android:padding="12dp"  
 android:text="@string/register\_button"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/roleContainer" />  
  
 <TextView  
 android:id="@+id/loginTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="@string/login\_prompt"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/registerButton" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:hint="Имя">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/nameEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPersonName" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:hint="Email">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/emailEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textEmailAddress" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:hint="Пароль">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/passwordEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPassword" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginBottom="4dp"  
 android:text="Роль пользователя:"  
 android:textSize="16sp" />  
  
 <RadioGroup  
 android:id="@+id/roleRadioGroup"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp">  
  
 <RadioButton  
 android:id="@+id/customerRadioButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Покупатель"  
 android:textSize="16sp"  
 android:checked="true"  
 android:padding="8dp"/>  
  
 <RadioButton  
 android:id="@+id/adminRadioButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Администратор"  
 android:textSize="16sp"  
 android:padding="8dp"/>  
 </RadioGroup>  
  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <TextView  
 android:id="@+id/titleTextView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="16dp"  
 android:text="Изменить роль пользователя"  
 android:textSize="18sp"  
 android:textStyle="bold" />  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginBottom="4dp"  
 android:text="Выберите роль:"  
 android:textSize="16sp" />  
  
 <RadioGroup  
 android:id="@+id/roleRadioGroup"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp">  
  
 <RadioButton  
 android:id="@+id/customerRadioButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Покупатель"  
 android:textSize="16sp"  
 android:padding="8dp"/>  
  
 <RadioButton  
 android:id="@+id/adminRadioButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Администратор"  
 android:textSize="16sp"  
 android:padding="8dp"/>  
 </RadioGroup>  
  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Оцените доставку"  
 android:textSize="18sp"  
 android:textStyle="bold"  
 android:layout\_marginBottom="16dp" />  
  
 <RatingBar  
 android:id="@+id/ratingBar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:numStars="5"  
 android:stepSize="0.5"  
 android:rating="4.0" />  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Ваш отзыв поможет нам улучшить сервис доставки"  
 android:textSize="14sp"  
 android:layout\_marginTop="16dp"  
 android:gravity="center" />  
  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:hint="Название товара\*">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/nameEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="text" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:hint="Описание">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/descriptionEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textMultiLine"  
 android:minLines="2" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:hint="Цена\*">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/priceEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="numberDecimal" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:hint="Количество на складе">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/stockEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="number" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginBottom="4dp"  
 android:text="Категория\*"  
 android:textSize="16sp" />  
  
 <Spinner  
 android:id="@+id/categorySpinner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp" />  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:hint="Ингредиенты">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/ingredientsEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textMultiLine"  
 android:minLines="2" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:hint="Пищевая ценность">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/nutritionEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textMultiLine"  
 android:minLines="2" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:text="\* - обязательные поля"  
 android:textColor="@android:color/holo\_red\_light"  
 android:textSize="12sp" />  
  
 </LinearLayout>  
</ScrollView>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="@string/password\_recovery"  
 android:textSize="20sp"  
 android:textStyle="bold"  
 android:layout\_marginBottom="16dp" />  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/recovery\_instruction"  
 android:layout\_marginBottom="16dp" />  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/usernameRecoveryInputLayout"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/username\_hint"  
 android:layout\_marginBottom="16dp">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/usernameRecoveryEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="text" />  
  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <Button  
 android:id="@+id/sendButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/send\_button" />  
  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="@string/reset\_password"  
 android:textSize="20sp"  
 android:textStyle="bold"  
 android:layout\_marginBottom="16dp" />  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/newPasswordInputLayout"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/new\_password"  
 app:endIconMode="password\_toggle"  
 android:layout\_marginBottom="16dp">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/newPasswordEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPassword" />  
  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/confirmNewPasswordInputLayout"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/confirm\_new\_password"  
 app:endIconMode="password\_toggle"  
 android:layout\_marginBottom="16dp">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/confirmNewPasswordEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPassword" />  
  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <Button  
 android:id="@+id/resetPasswordButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/reset\_password" />  
  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <TextView  
 android:id="@+id/messageTextView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textSize="16sp"  
 android:textColor="#000000"  
 android:layout\_marginBottom="16dp"  
 android:text="Для продолжения сохраните чек с деталями заказа" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:layout\_marginTop="8dp">  
  
 <Button  
 android:id="@+id/saveButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Сохранить чек"  
 android:textColor="#FFFFFF"  
 android:backgroundTint="#4CAF50"  
 android:layout\_marginBottom="8dp"/>  
  
 <Button  
 android:id="@+id/continueButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Продолжить"  
 android:textColor="#FFFFFF"  
 android:enabled="false"/>  
  
 </LinearLayout>  
  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <LinearLayout  
 android:id="@+id/containerLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent">  
  
 <ListView  
 android:id="@+id/productsListView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:divider="@android:color/darker\_gray"  
 android:dividerHeight="0.5dp" />  
  
 </LinearLayout>  
  
 <TextView  
 android:id="@+id/emptyTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Список товаров пуст"  
 android:textSize="18sp"  
 android:visibility="gone"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <com.google.android.material.floatingactionbutton.FloatingActionButton  
 android:id="@+id/addProductFab"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="16dp"  
 android:contentDescription="Добавить товар"  
 android:src="@android:drawable/ic\_input\_add"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <LinearLayout  
 android:id="@+id/containerLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent">  
  
 <ListView  
 android:id="@+id/usersListView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:divider="@android:color/darker\_gray"  
 android:dividerHeight="0.5dp" />  
  
 </LinearLayout>  
  
 <TextView  
 android:id="@+id/emptyTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Список пользователей пуст"  
 android:textSize="18sp"  
 android:visibility="gone"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <com.google.android.material.floatingactionbutton.FloatingActionButton  
 android:id="@+id/addUserFab"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="16dp"  
 android:contentDescription="Добавить пользователя"  
 android:src="@android:drawable/ic\_input\_add"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginHorizontal="16dp"  
 android:layout\_marginVertical="8dp"  
 app:cardCornerRadius="12dp"  
 app:cardElevation="3dp">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="12dp">  
  
 <ImageView  
 android:id="@+id/cartProductImageView"  
 android:layout\_width="80dp"  
 android:layout\_height="80dp"  
 android:scaleType="centerCrop"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 tools:src="@drawable/ic\_cake" />  
  
 <TextView  
 android:id="@+id/cartProductNameTextView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="12dp"  
 android:layout\_marginEnd="8dp"  
 android:ellipsize="end"  
 android:maxLines="2"  
 android:textColor="@color/black"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toStartOf="@+id/removeCartItemButton"  
 app:layout\_constraintStart\_toEndOf="@+id/cartProductImageView"  
 app:layout\_constraintTop\_toTopOf="parent"  
 tools:text="Шоколадный торт" />  
  
 <TextView  
 android:id="@+id/cartItemPriceTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="12dp"  
 android:textColor="@color/colorPrimary"  
 android:textSize="16sp"  
 app:layout\_constraintStart\_toEndOf="@+id/cartProductImageView"  
 app:layout\_constraintTop\_toBottomOf="@+id/cartProductNameTextView"  
 tools:text="800 ₽" />  
  
 <LinearLayout  
 android:id="@+id/quantityControlLayout"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:orientation="horizontal"  
 app:layout\_constraintStart\_toStartOf="@+id/cartItemPriceTextView"  
 app:layout\_constraintTop\_toBottomOf="@+id/cartItemPriceTextView">  
  
 <ImageButton  
 android:id="@+id/cartDecreaseButton"  
 android:layout\_width="40dp"  
 android:layout\_height="40dp"  
 android:background="@drawable/round\_button\_background"  
 android:contentDescription="Уменьшить количество"  
 android:src="@drawable/ic\_minus"  
 android:padding="8dp" />  
  
 <TextView  
 android:id="@+id/cartQuantityTextView"  
 android:layout\_width="40dp"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:textColor="@color/black"  
 android:textSize="16sp"  
 tools:text="1" />  
  
 <ImageButton  
 android:id="@+id/cartIncreaseButton"  
 android:layout\_width="40dp"  
 android:layout\_height="40dp"  
 android:background="@drawable/round\_button\_background"  
 android:contentDescription="Увеличить количество"  
 android:src="@drawable/ic\_plus"  
 android:padding="8dp" />  
  
 </LinearLayout>  
  
 <TextView  
 android:id="@+id/cartItemTotalTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textColor="@color/black"  
 android:textSize="14sp"  
 android:textStyle="bold"  
 app:layout\_constraintBottom\_toBottomOf="@+id/quantityControlLayout"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toTopOf="@+id/quantityControlLayout"  
 tools:text="x2: 1600 ₽" />  
  
 <ImageButton  
 android:id="@+id/removeCartItemButton"  
 android:layout\_width="30dp"  
 android:layout\_height="30dp"  
 android:background="?attr/selectableItemBackgroundBorderless"  
 android:contentDescription="@string/delete\_item"  
 android:src="@drawable/ic\_close"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:tint="@color/colorPrimary" />  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
</androidx.cardview.widget.CardView>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="8dp"  
 app:cardCornerRadius="8dp"  
 app:cardElevation="4dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <TextView  
 android:id="@+id/textViewOrderId"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textStyle="bold"  
 android:textSize="16sp" />  
  
 <TextView  
 android:id="@+id/textViewOrderDate"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="4dp"  
 android:textSize="14sp" />  
  
 <TextView  
 android:id="@+id/textViewOrderStatus"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="4dp"  
 android:textColor="@android:color/holo\_green\_dark"  
 android:textSize="14sp" />  
  
 <TextView  
 android:id="@+id/textViewOrderTotal"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="4dp"  
 android:textStyle="bold"  
 android:textSize="16sp" />  
  
 <TextView  
 android:id="@+id/textViewOrderItems"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:textSize="14sp" />  
  
 </LinearLayout>  
  
</androidx.cardview.widget.CardView>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="8dp">  
  
 <TextView  
 android:id="@+id/product\_name\_text\_view"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:textSize="16sp"  
 android:maxLines="2"  
 android:ellipsize="end"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/quantity\_text\_view"  
 app:layout\_constraintHorizontal\_weight="2"  
 android:layout\_marginEnd="8dp"/>  
  
 <TextView  
 android:id="@+id/quantity\_text\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="16sp"  
 android:gravity="center"  
 app:layout\_constraintTop\_toTopOf="@+id/product\_name\_text\_view"  
 app:layout\_constraintEnd\_toStartOf="@+id/price\_text\_view"  
 android:layout\_marginEnd="8dp"/>  
  
 <TextView  
 android:id="@+id/price\_text\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="16sp"  
 android:gravity="end"  
 app:layout\_constraintTop\_toTopOf="@+id/product\_name\_text\_view"  
 app:layout\_constraintEnd\_toStartOf="@+id/total\_text\_view"  
 android:layout\_marginEnd="8dp"/>  
  
 <TextView  
 android:id="@+id/total\_text\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 android:gravity="end"  
 app:layout\_constraintTop\_toTopOf="@+id/product\_name\_text\_view"  
 app:layout\_constraintEnd\_toEndOf="parent"/>  
  
 <View  
 android:layout\_width="match\_parent"  
 android:layout\_height="1dp"  
 android:background="#EEEEEE"  
 android:layout\_marginTop="8dp"  
 app:layout\_constraintTop\_toBottomOf="@+id/product\_name\_text\_view"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 android:layout\_marginBottom="1dp"/>  
  
</androidx.constraintlayout.widget.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="8dp"  
 app:cardCornerRadius="8dp"  
 app:cardElevation="4dp">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="8dp">  
  
 <ImageView  
 android:id="@+id/productImageView"  
 android:layout\_width="80dp"  
 android:layout\_height="80dp"  
 android:contentDescription="@string/product\_image"  
 android:scaleType="centerCrop"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 tools:src="@drawable/ic\_cake" />  
  
 <TextView  
 android:id="@+id/productNameTextView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@id/productImageView"  
 app:layout\_constraintTop\_toTopOf="parent"  
 tools:text="Шоколадный торт" />  
  
 <TextView  
 android:id="@+id/productPriceTextView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="4dp"  
 android:textColor="@color/purple\_700"  
 android:textSize="14sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@id/productImageView"  
 app:layout\_constraintTop\_toBottomOf="@id/productNameTextView"  
 tools:text="1299 ₽" />  
  
 <TextView  
 android:id="@+id/productDescriptionTextView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="4dp"  
 android:ellipsize="end"  
 android:maxLines="2"  
 android:textSize="14sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@id/productImageView"  
 app:layout\_constraintTop\_toBottomOf="@id/productPriceTextView"  
 tools:text="Нежный шоколадный торт с кремовой начинкой и шоколадной глазурью" />  
  
 <Button  
 android:id="@+id/addToCartButton"  
 style="@style/Widget.MaterialComponents.Button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:text="@string/add\_to\_cart"  
 android:textSize="12sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/productDescriptionTextView" />  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
</androidx.cardview.widget.CardView>

<?xml version="1.0" encoding="utf-8"?>  
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto">  
   
 <item  
 android:id="@+id/action\_cart"  
 android:icon="@drawable/ic\_cart"  
 android:title="@string/cart"  
 app:showAsAction="ifRoom" />  
  
 <item  
 android:id="@+id/action\_profile"  
 android:icon="@drawable/ic\_profile"  
 android:title="@string/profile"  
 app:showAsAction="never" />  
  
 <item  
 android:id="@+id/action\_order\_history"  
 android:icon="@drawable/ic\_history"  
 android:title="@string/order\_history"  
 app:showAsAction="never" />  
  
 <item  
 android:id="@+id/action\_admin\_panel"  
 android:icon="@drawable/ic\_admin"  
 android:title="@string/admin\_panel"  
 app:showAsAction="never" />  
  
 <item  
 android:id="@+id/action\_debug\_reinstall\_db"  
 android:title="Переустановить базу данных (Debug)"  
 app:showAsAction="never" />  
  
 <item  
 android:id="@+id/action\_logout"  
 android:title="@string/logout"  
 app:showAsAction="never" />  
</menu>