Заглавна страница  
Борис Кирилов Костадинов  
Проектиране и разработване на web базирана система за поръчки и плащания в заведения

Утвърдено дипломно задание

Съдържание

[УВОД 6](#_Toc200932965)

[ГЛАВА 1. ПОСТАНОВКА НА ДИПЛОМНАТА РАБОТА, ЦЕЛИ, ЗАДАЧИ И ФУНКЦИОНАЛНОСТИ НА SMARTBAR 8](#_Toc200932966)

[1.1 Постановка на задачата 8](#_Toc200932967)

[1.2 Цели на дипломната работа 8](#_Toc200932968)

[1.3 Основни функционалности на SmartBar 9](#_Toc200932969)

[ГЛАВА 2: ПРОЕКТИРАНЕ НА СИСТЕМАТА SMARTBAR 12](#_Toc200932970)

[2.1 Общ преглед на архитектурата 12](#_Toc200932971)

[2.2 Избор на технологии 12](#_Toc200932972)

[2.3 Структура на приложението 13](#_Toc200932973)

[2.4 Проектиране на базата от данни 14](#_Toc200932974)

[2.5 Поток на данни и логика 15](#_Toc200932975)

[2.6 Предимства на архитектурата 16](#_Toc200932976)

[2.7 Възможности за надграждане 16](#_Toc200932977)

[ГЛАВА 3: РЕАЛИЗАЦИЯ НА УЕБ ПРИЛОЖЕНИЕТО SMARTBAR 17](#_Toc200932978)

[3.1 Обща представа за реализацията 17](#_Toc200932979)

[3.2 Настройка на проекта 17](#_Toc200932980)

[3.3 Регистрация и автентикация на потребители 17](#_Toc200932981)

[3.4 Управление на продукти и категории 18](#_Toc200932982)

[3.5 Количка и добавяне на продукти 19](#_Toc200932983)

[3.6 Създаване на поръчка 20](#_Toc200932984)

[3.7 Онлайн плащане с PayPal 20](#_Toc200932985)

[3.8 Генериране на фактура 23](#_Toc200932986)

[3.9 Административен панел 23](#_Toc200932987)

[3.10 Имейл система 23](#_Toc200932988)

[3.11 Стил и визуализация 23](#_Toc200932989)

[3.12 Тестване и отстраняване на грешки 24](#_Toc200932990)

[ГЛАВА 4: РЪКОВОДСТВО ЗА РАБОТА С УЕБ ПРИЛОЖЕНИЕТО SMARTBAR 25](#_Toc200932991)

[🔹 4.1 Начална страница (Home) 25](#_Toc200932992)

[🔹 4.2 Регистрация и вход 26](#_Toc200932993)

[🔹 4.3 Преглед на продукти 27](#_Toc200932994)

[🔹 4.4 Количка 27](#_Toc200932995)

[🔹 4.5 Поръчка 28](#_Toc200932996)

[🔹 4.6 Плащане с PayPal 28](#_Toc200932997)

[🔹 4.7 Преглед на завършена поръчка 29](#_Toc200932998)

[🔹 4.8 История на поръчките 29](#_Toc200932999)

[🔹 4.9 Административен панел (Django Admin) 30](#_Toc200933000)

[🔹 4.10 Работа с имейли 30](#_Toc200933001)

[ГЛАВА 5: ЗАКЛЮЧЕНИЕ 31](#_Toc200933002)

[🔷 5.1 Обобщение на проекта 31](#_Toc200933003)

[🔷 5.2 Постигнати резултати 31](#_Toc200933004)

[🔷 5.3 Използвани технологии и практики 32](#_Toc200933005)

[🔷 5.4 Ползи от системата SmartBar 33](#_Toc200933006)

[🔷 5.5 Проблеми и предизвикателства 34](#_Toc200933007)

[🔷 5.6 Възможности за бъдещо развитие 34](#_Toc200933008)

[🔷 5.7 Личен принос 35](#_Toc200933009)

[ГЛАВА 6: ИЗПОЛЗВАНИ ИЗТОЧНИЦИ И ПРИЛОЖЕНИЯ 37](#_Toc200933010)

# УВОД

В условията на съвременния дигитален свят, все повече заведения за хранене и развлечение – като ресторанти, барове и кафенета – се стремят да модернизират и автоматизират процесите на обслужване на клиентите. Това включва не само подобряване на вътрешната организация на работа, но и предоставяне на интуитивни, бързи и сигурни начини за поръчване и разплащане. В този контекст, уеб базираните системи за поръчки и онлайн плащания се превръщат в необходимост за всяко заведение, което цели да отговори на очакванията на модерните потребители.

Настоящата дипломна работа има за цел да проектира и реализира уеб приложение с наименование **SmartBar**, което предоставя цялостна система за дигитално обслужване в заведения – от избора и поръчката на продукти, до извършване на плащания през интернет. Системата е насочена към крайни потребители (клиенти на заведението), както и към администраторите и персонала, които управляват менюто, продуктите, поръчките и транзакциите.

По време на разработката на проекта бяха извършени редица инженерни дейности – анализ на изискванията, проектиране на архитектурата, изграждане на базата данни, създаване на интуитивен потребителски интерфейс, и интегриране на онлайн платежен модул (PayPal). Уеб приложението е разработено с помощта на съвременни технологии, сред които **Python**, **Django**, **HTML**, **CSS**, **Bootstrap** и **JavaScript**. За осигуряване на достъпност и надеждност, приложението е подготвено за хостване в облачна среда с използване на **AWS** и възможност за интеграция с имейл услуги и база данни.

Основната цел на проекта е да предложи решение, което:

* улеснява клиентите при заявяване на поръчки;
* автоматизира процеса на обработка и плащане;
* намалява натоварването на обслужващия персонал;
* подобрява цялостното потребителско изживяване в заведението.

В рамките на дипломната работа се представят всички основни фази от разработката на системата SmartBar – от формулирането на изискванията, през проектирането и реализирането ѝ, до извършване на тестове и анализ на резултатите. Освен това е включено ръководство за работа с приложението, както и препоръки за бъдещо разширяване на функционалността.

С настоящия проект се цели демонстрация на умения за проектиране, програмиране, интегриране на технологии и създаване на цялостно функциониращ софтуерен продукт, ориентиран към реален бизнес казус в сферата на ресторантьорството.

# ГЛАВА 1. ПОСТАНОВКА НА ДИПЛОМНАТА РАБОТА, ЦЕЛИ, ЗАДАЧИ И ФУНКЦИОНАЛНОСТИ НА SMARTBAR

## 1.1 Постановка на задачата

В последните години дигитализацията навлезе интензивно във всички сфери на обществения живот, включително и в ресторантьорството. Технологиите трансформират начина, по който клиентите взаимодействат със заведенията – от разглеждане на меню, през поръчване, до извършване на плащания. Стремежът към бързо, ефективно и безконтактно обслужване породи необходимостта от внедряване на цялостни уеб базирани решения, които улесняват както клиентите, така и служителите.

Проблемите, които традиционната организация на обслужване често създава – като забавяне при обслужване, липса на проследимост на поръчките, затруднения в комуникацията между сервитьори и кухня – налагат нуждата от софтуерни системи, които да автоматизират ключови процеси. Допълнително, в условията на пандемии, необходимостта от безконтактни методи на плащане и самообслужване се превърна от удобство в изискване.

Заданието на дипломната работа е да се проектира, разработи и тества цялостно уеб базирано приложение за поръчки и онлайн разплащания в заведения от тип ресторант или бар. Разработената система трябва да предлага функционалност за клиентски поръчки, визуализация на менюто, добавяне в количка, плащане онлайн, както и административен панел за управление на продукти, категории и поръчки. Основната идея е чрез тази система да се подобри клиентското изживяване и да се намалят административните усилия от страна на персонала.

## 1.2 Цели на дипломната работа

Основната цел на дипломната работа е изграждането на напълно функционираща уеб базирана система за заведения, която да изпълнява следните функции:

* Предоставяне на дигитален интерфейс за преглед и избор на продукти от менюто на заведението.
* Възможност за регистриране и влизане на потребители.
* Обработка на поръчки с детайлна информация за избраните артикули и вариации (например размер, вкус, добавки).
* Онлайн плащане чрез интеграция с платформа за електронни разплащания (PayPal).
* Достъп до административен панел за управление на продуктите, категориите и постъпилите поръчки.
* Изпращане на автоматични имейли при регистрация, направена поръчка и плащане.
* Подобряване на процеса на обслужване чрез автоматизация и дигитализация.

Допълнителните цели включват:

* Сигурно съхранение на данни чрез използване на база данни.
* Добро потребителско изживяване (UX) чрез съвременен дизайн с помощта на Bootstrap.
* Модулна архитектура, позволяваща лесно разширение в бъдеще (напр. интеграция с мобилно приложение или QR меню).
* Хостване на приложението в облачна среда (напр. AWS или друг публичен хостинг).

## 1.3 Основни функционалности на SmartBar

SmartBar е уеб приложение, ориентирано към две основни групи потребители: **клиенти на заведението** и **администратори/служители**. По-долу са описани всички основни модули и функционалности, които системата предлага:

**➤ 1. Регистрация и автентикация на потребители**

* Възможност за създаване на потребителски акаунт чрез регистрационна форма.
* Вход и изход от системата с валидни данни.
* Потвърждение на регистрацията чрез имейл (активираща връзка).
* Възможност за възстановяване на забравена парола.

**➤ 2. Преглед на менюто**

* Менюто е разделено в категории (напр. Напитки, Храни, Десерти).
* Визуализация на продукти със снимка, цена и кратко описание.
* Поддържане на вариации на продукти (например: размер на напитка, добавки към храна).

**➤ 3. Количка и преглед на поръчка**

* Добавяне на продукти в количката с избрани вариации.
* Преглед на количката с възможност за корекция (увеличаване/намаляване на количество, изтриване).
* Автоматично изчисляване на междинна сума и крайна цена.

**➤ 4. Онлайн плащания**

* Интеграция с PayPal за извършване на реални онлайн плащания.
* Преобразуване на валута (напр. EUR).
* Записване на статус на транзакцията (успешна, неуспешна, чакаща).
* Създаване на поръчка само след успешно плащане.

**➤ 5. Административен панел**

* Създаване, редакция и изтриване на продукти и категории.
* Преглед на всички поръчки с филтри по статус.
* Промяна на статус на поръчка (напр. „Приета“, „Завършена“, „Отказана“).
* Достъп до списъци с потребители и техните поръчки.

**➤ 6. Имейл известия**

* Изпращане на имейл при:
  + Регистрация на нов потребител
  + Потвърждение на поръчка
  + Завършено плащане
* Имейл шаблони с автоматично попълнени данни (име, поръчка, дата и т.н.)

**➤ 7. Фактуриране и преглед на завършена поръчка**

* Потребителят може да прегледа направената поръчка.
* Създава се автоматично „фактура“ с всички детайли по транзакцията.
* Възможност за принтиране на документа директно от интерфейса.

**➤ 8. Допълнителни възможности**

* Отзивчив (responsive) дизайн — работи на компютър, таблет и телефон.
* Страници за „За нас“, „Контакт“ и друга информационна структура.
* Визуални съобщения и предупреждения при грешки, празна количка и др.

Тези функционалности превръщат SmartBar в завършена система, приложима в реална бизнес среда, и създават основа за по-нататъшно развитие – като мобилна версия, QR поръчки от маса, интеграция с POS терминали и др.

# ГЛАВА 2: ПРОЕКТИРАНЕ НА СИСТЕМАТА SMARTBAR

## 2.1 Общ преглед на архитектурата

Уеб системата SmartBar е реализирана като многослойна архитектура, базирана на **модела Model-View-Template (MVT)**, характерен за фреймуърка Django. Подобно на архитектурния модел Model-View-Controller (MVC), MVT разделя логиката на приложението на три основни части:

* **Model** – определя структурата на данните и начините за тяхното взаимодействие с базата данни;
* **View** – съдържа логиката за обработка на заявки и генериране на отговори;
* **Template** – определя визуалната част на уеб приложението (HTML + динамично съдържание).

Тази архитектура е избрана, тъй като осигурява ясно разделение между бизнес логиката, потребителския интерфейс и данните. Това улеснява разработката, поддръжката и бъдещото разширение на системата.

## 2.2 Избор на технологии

Изборът на технологии за системата SmartBar е базиран на съвременни, добре поддържани и широко използвани инструменти, които предлагат стабилност, сигурност и гъвкавост:

**➤ Backend:**

* **Python** – популярен, четим и лесен за поддръжка език, идеален за уеб разработка.
* **Django** – мощен фреймуърк за изграждане на уеб приложения, който предоставя множество вградени функционалности като ORM (Object-Relational Mapping), администрация, сигурност, маршрутизиране и шаблони.

**➤ Frontend:**

* **HTML5** – за структуриране на уеб съдържанието.
* **CSS3** и **Bootstrap** – за стилизиране и адаптивност на интерфейса, с цел постигане на модерен и отзивчив дизайн.
* **JavaScript** – използван за допълнителна интерактивност и манипулиране на DOM.

**➤ База от данни:**

* **SQLite** – използвана по време на разработка заради лекотата и простотата си; при нужда може да се замени с PostgreSQL за продакшън среда.

**➤ Интеграции:**

* **PayPal API** – за реални онлайн разплащания.
* **SMTP (Gmail)** – за изпращане на автоматични имейли за потвърждение на регистрация, поръчки и транзакции.

**➤ Облак и хостинг (по избор):**

* AWS, Heroku, PythonAnywhere или локален сървър с Apache/Nginx и uWSGI/Gunicorn.

## 2.3 Структура на приложението

Приложението е разделено на следните модули (Django apps):

| **Модул** | **Описание** |
| --- | --- |
| **accounts** | Управлява регистрация, вход и профили на потребителите |
| **store** | Съдържа всички продукти, категории и вариации |
| **carts** | Управлява количката и поръчките в нея |
| **orders** | Отговаря за създаване на поръчки, плащания и фактуриране |
| **category** | Обособява категории за продуктите |
| **templates** | Всички HTML файлове с визуализация на интерфейса |

## 2.4 Проектиране на базата от данни

Базата от данни е проектирана така, че да осигури максимална нормализация, логическа последователност и възможност за разширение. Ето описанието на основните модели:

**🔹 Account (User):**

* Име, фамилия, имейл, телефон, парола (шифрована)
* Дата на създаване
* Активиращ флаг (is\_active)
* Роля (по избор: клиент, служител, админ)

**🔹 UserProfile:**

* Връзка с Account (OneToOne)
* Снимка, адрес, град, държава, пощенски код

**🔹 Category:**

* Име на категория (напр. „Напитки“, „Храна“)
* Снимка и слаг за URL достъп

**🔹 Product:**

* Име, описание, цена, наличност, снимка
* Връзка към категория
* Поле „is\_available“

**🔹 Variation:**

* Продукт
* Тип вариация (напр. „Размер“)
* Стойност (напр. „Малка“, „Голяма“)

**🔹 Cart и CartItem:**

* Временна „кошница“, идентифицирана чрез сесия
* CartItem съдържа продукт, количество, вариации, сума

**🔹 Order и OrderProduct:**

* Информация за поръчката: потребител, адрес, сума, статус
* Свързани продукти, количество, цена, вариации
* Връзка с транзакцията (платежен запис)

**🔹 Payment:**

* ID на транзакция (от PayPal)
* Статус, метод, обща сума, дата

## 2.5 Поток на данни и логика

1. **Потребителят** влиза в системата или се регистрира.
2. Разглежда продуктите и избира конкретни артикули с техните вариации.
3. Продуктите се добавят в количката, съхранявани във временно състояние.
4. При потвърждение, се генерира поръчка и се извиква PayPal API за разплащане.
5. След потвърдено плащане се записва транзакцията и поръчката става активна.
6. Администраторът вижда новата поръчка в панела и може да промени статуса.
7. Потребителят получава имейл с потвърждение и фактура.

## 2.6 Предимства на архитектурата

* **Модулност:** всяка функционалност е в отделен app.
* **Сигурност:** Django предоставя защита срещу CSRF, SQL инжекции, XSS.
* **Гъвкавост:** лесно разширяване чрез добавяне на нови модели и изгледи.
* **Скалируемост:** в бъдеще може да се добавят REST API и мобилно приложение.

## 2.7 Възможности за надграждане

Проектираната архитектура позволява лесно надграждане със следните възможности:

* Добавяне на REST API с Django REST Framework.
* Изграждане на мобилно приложение, което комуникира с уеб услугите.
* Интеграция с QR кодове за самостоятелни поръчки от маса.
* Поддръжка на други платежни платформи (Stripe, Revolut).
* Създаване на панел за персонала (бармани, готвачи).

С тази архитектура системата SmartBar е готова да покрие реалните нужди на заведение, като същевременно предлага стабилна и мащабируема основа за бъдещо развитие.

# ГЛАВА 3: РЕАЛИЗАЦИЯ НА УЕБ ПРИЛОЖЕНИЕТО SMARTBAR

## 3.1 Обща представа за реализацията

Реализацията на уеб приложението **SmartBar** се извършва с помощта на фреймуърка **Django**, който предоставя пълна среда за изграждане на уеб базирани системи, използвайки езика **Python**. Django е избран заради своите предимства като високо ниво на сигурност, вградена администрация, автоматично генерирани маршрути, шаблони и ORM (Object-Relational Mapping) за работа с бази от данни.

Уеб приложението е разделено на логически модули (Django приложения), всяко от които отговаря за отделна функционалност: потребители, продукти, количка, поръчки и др.

## 3.2 Настройка на проекта

**▶ 3.2.1 Създаване на Django проект**

Стартирането на проекта става с командата:

django-admin startproject greatbar

cd greatbar

**▶ 3.2.2 Създаване на приложения**

В Django всеки модул е отделно "приложение". В проекта са създадени:

python manage.py startapp accounts

python manage.py startapp store

python manage.py startapp carts

python manage.py startapp orders

python manage.py startapp category

## 3.3 Регистрация и автентикация на потребители

**▶ Терминология:**

* **Регистрация** – процес, в който потребител създава акаунт в системата.
* **Автентикация (authentication)** – проверка на самоличността чрез потребителско име и парола.
* **Авторизация (authorization)** – определяне какви права има даден потребител след като е автентикиран.

**▶ Реализация:**

Моделът Account наследява AbstractBaseUser и BaseUserManager, като съдържа всички необходими полета (име, имейл, телефон, парола, дата на създаване). Използва се потребителски мениджър за създаване на потребители и суперпотребители (админи).

Регистрацията и входът се осъществяват чрез изгледи (views.py), които обработват данни от форми (forms.py). При успешно създаване на потребител, се изпраща имейл за активация.

**Пример от кода:**

def register(request):

if request.method == 'POST':

form = RegistrationForm(request.POST)

if form.is\_valid():

user = Account.objects.create\_user(...)

# изпращане на имейл за активация

## 3.4 Управление на продукти и категории

**▶ Категории**

Категориите (напр. „Храни“, „Напитки“) се създават чрез модел Category и се използват за групиране на продуктите. Те се показват в навигацията чрез context processor.

**▶ Продукти**

Моделът Product съдържа полета като:

* product\_name
* description
* price
* stock
* image
* category (ForeignKey)

Използват се слагове (slug) за изграждане на SEO-приятелски URL-и.

**Пример:**

/store/napitki/bezalkoholno/

**▶ Вариации**

Някои продукти имат варианти – напр. напитка с размер **малка**, **средна**, **голяма**. Това се реализира чрез модел Variation.

## 3.5 Количка и добавяне на продукти

**▶ Какво е количка?**

„Количка“ (cart) е временно място, където потребителят съхранява избраните продукти преди да ги поръча. Работи по аналогия с физическата количка в магазин.

**▶ Реализация:**

* Използва се Cart модел – всяка количка е свързана със сесия или с потребител.
* CartItem – всеки ред в количката: продукт, брой, вариации.

**▶ Логика:**

* При натискане на "Добави", изгледът проверява дали такъв CartItem вече съществува – ако да, увеличава количеството; ако не – създава нов запис.

def add\_cart(request, product\_id):

# взима продукт, добавя го в количката или увеличава количеството

## 3.6 Създаване на поръчка

**▶ Форма за поръчка**

Потребителят попълва данни за доставка чрез OrderForm, където въвежда:

* Име
* Адрес
* Град, държава
* Бележка

**▶ Модел Order**

Съдържа всички въведени от клиента данни + стойности като:

* Сума
* Данък
* IP адрес
* Номер на поръчка

order\_number = current\_date + str(data.id)

## 3.7 Онлайн плащане с PayPal

PayPal е платформа за електронни плащания, която позволява извършване на трансакции между клиенти и търговци без да се разкриват банкови данни.

По време на разработката на функционалността за онлайн разплащания в SmartBar, беше използвана **PayPal Sandbox среда** – специална тестова инфраструктура, предоставена от PayPal, която симулира реални транзакции без реални пари.

PayPal Sandbox е среда за разработчици, която позволява:

* Да се **създават тестови акаунти** (лични и бизнес)
* Да се извършват тестови плащания без реални средства
* Да се тестват съобщения за успешно, неуспешно или отложено плащане
* Да се валидира интеграцията между уеб приложението и PayPal API

**🔹 Приложение в проекта SmartBar**

При внедряване на онлайн плащанията:

* Беше създаден **PayPal Developer акаунт**
* Създадени бяха два sandbox акаунта:
  + Един **личен акаунт** – симулира клиента
  + Един **бизнес акаунт** – симулира сметката на заведението
* В проекта е използван официалният JavaScript SDK на PayPal с линк към Sandbox:

<script src="https://www.paypal.com/sdk/js?client-id=YOUR\_SANDBOX\_CLIENT\_ID&currency=EUR"></script>

* Всички тестови транзакции се извършват чрез sandbox.paypal.com и **не генерираха реални разходи**.

**🔹 Предимства**

* Без риск от реални загуби по време на разработка
* Може да се симулират различни сценарии: отменено плащане, грешка, успешно плащане
* Позволява пълна автоматизация на тестове

**🔹 Преход към продукция**

След тестването, PayPal SDK линкът трябва да бъде заменен със **production live client ID**, и се премахва параметърът sandbox.

**▶ Интеграция**

Използва се JavaScript SDK от PayPal. При натискане на „Плати“, се създава PayPal поръчка:

paypal.Buttons({

createOrder: function(data, actions) {

return actions.order.create({

purchase\_units: [{ amount: { value: amount, currency\_code: 'EUR' } }]

});

},

onApprove: function(data, actions) {

return actions.order.capture().then(function(details) {

// Изпращане на данни към Django (fetch -> /payments/)

});

}

}).render('#paypal-button-container');

**▶ Обработка на плащане в Django**

След одобрение на плащането, се записва информацията:

def payments(request):

body = json.loads(request.body)

payment = Payment.objects.create(...)

order.payment = payment

order.is\_ordered = True

order.save()

## 3.8 Генериране на фактура

* Използва се шаблон order\_detail.html.
* Данните за поръчката се показват в табличен вид.
* Създаден е **бутон за принтиране** чрез JavaScript, който отпечатва само div.container.

## 3.9 Административен панел

* Админите могат да управляват потребители, продукти, категории, поръчки.
* Всичко това се извършва чрез Django Admin, като моделите са регистрирани и с допълнителна конфигурация (напр. списъчни изгледи, филтри и полета за търсене).

## 3.10 Имейл система

Използва се SMTP през Gmail за:

* Потвърждение на регистрация (активационен линк)
* Потвърждение на поръчка
* Известия за статус на плащане

email = EmailMessage(subject, message, to=[user.email])

email.send()

## 3.11 Стил и визуализация

* Използван е Bootstrap за responsive дизайн.
* Изображения на продукти се зареждат от /media/ или /static/.
* Страниците са съобразени с UX практики: лесна навигация, ясно видими бутони, съобщения при грешки и успешни действия.

## 3.12 Тестване и отстраняване на грешки

Извършени са тестове на:

* Регистрация и вход
* Добавяне на продукти в количка
* Правилна сума на поръчката
* Успешно и неуспешно плащане
* Имейл изпращане
* Работа с админ панел

**Заключение на главата**

Главата представи в детайли процеса на реализация на уеб системата SmartBar — от структурата на приложенията до конкретна логика за потребители, поръчки, плащания и интерфейс. Реализацията е извършена съгласно добрите практики на уеб разработка с акцент върху сигурност, модулност и удобство за крайния потребител.

# ГЛАВА 4: РЪКОВОДСТВО ЗА РАБОТА С УЕБ ПРИЛОЖЕНИЕТО SMARTBAR

Тази глава описва подробно начина, по който се използва системата **SmartBar** както от **обикновени потребители (клиенти)**, така и от **администратори и персонал**. Целта е да се даде ясно и систематизирано обяснение на всички интерфейсни елементи и логика на поведение в реална работна среда. Всички действия ще бъдат описани **стъпка по стъпка**, заедно с обяснение на функционалността, навигацията и резултатите.

Тази глава ще даде пълно ръководство за работа със SmartBar – от първото посещение на сайта до успешно плащане и администриране. Системата е проектирана с интуитивен интерфейс, сигурна логика и пълна проследимост на действията. Всяка стъпка е съобразена както с очакванията на потребителите, така и с нуждите на служителите и управленския екип.

## 🔹 4.1 Начална страница (Home)

След отваряне на сайта (например http://127.0.0.1:8000/), потребителят попада на **началната страница**, която съдържа:

* Навигационна лента с:
  + **Начало**
  + **Магазин (Меню)**
  + **Вход / Регистрация**
  + **Профил (ако е логнат)**
  + **Количка**
* **Банер изображение** или слайдър с промоционално съдържание
* Препратки към различните категории

➡ *От тук потребителят може да се насочи към менюто и да започне да разглежда продукти, дори без да е влязъл в профил.*

## 🔹 4.2 Регистрация и вход

**➤ Регистрация**

1. Натиска се бутон **„Регистрация“**.
2. Зарежда се форма със следните полета:
   * Име
   * Фамилия
   * Имейл (използва се за вход)
   * Телефонен номер
   * Парола + потвърждение
3. При изпращане на формата:
   * В системата се създава нов потребител.
   * Изпраща се **имейл с активационен линк**.
   * До активиране на акаунта потребителят не може да влезе.

**➤ Вход**

След активиране:

1. Потребителят въвежда имейл и парола.
2. Системата го идентифицира и пренасочва към началната страница.
3. Навигацията се променя – появява се „Моят профил“, „Изход“, „История на поръчките“.

## 🔹 4.3 Преглед на продукти

След кликване на **„Магазин“**, се зарежда списък с продукти:

* Продуктите са подредени в мрежа (grid), по категории.
* Всеки продукт има:
  + Снимка
  + Име
  + Цена
  + Бутон „Виж продукта“

**➤ Продукт детайл:**

1. След натискане на „Виж продукта“, се зарежда подробна страница.
2. В нея има:
   * Подробно описание
   * Възможност за избор на **вариации** (напр. размер: малък/голям)
   * Бутон **„Добави в количка“**

## 🔹 4.4 Количка

След добавяне, потребителят може да отиде в **Количката**, където:

* Вижда всички добавени продукти, със:
  + Избрани вариации
  + Количество
  + Единична цена
  + Обща цена за продукта
* Има възможности:
  + Увеличаване/намаляване на броя
  + Премахване на продукт
* В долната част:
  + **Обща сума**
  + **Данък (ако има)**
  + **Крайна сума**
  + **Бутон „Продължи към плащане“**

## 🔹 4.5 Поръчка

След натискане на „Продължи към плащане“:

1. Зарежда се страница с **форма за адрес и доставка**, включваща:
   * Име и фамилия
   * Телефон
   * Имейл
   * Държава, област, град
   * Адрес
   * Бележка към поръчката (по избор)
2. След потвърждение се създава **поръчка в режим „в изчакване“** и потребителят се пренасочва към **страницата за плащане.**

## 🔹 4.6 Плащане с PayPal

На страницата за плащане:

* Зарежда се бутон на **PayPal**, чрез вграден SDK.
* Потребителят:
  1. Натиска бутона.
  2. Влиза в акаунта си в PayPal.
  3. Потвърждава плащането.
* След успешно плащане:
  1. В базата се създава запис в таблицата **Payment**.
  2. Поръчката се маркира като „Завършена“.
  3. Изпраща се **имейл с фактура и потвърждение**.

## 🔹 4.7 Преглед на завършена поръчка

След плащането, потребителят се пренасочва към страница order\_complete/, където вижда:

* Уникален номер на поръчка
* Дата
* Статус на транзакцията (например „Завършена“)
* Таблица с продуктите
* Крайна цена
* **Бутон за принтиране на фактурата** – принтира се само избраната част от страницата.

## 🔹 4.8 История на поръчките

В потребителския профил има меню **„Моите поръчки“**, където клиентът:

* Вижда списък с предишни поръчки.
* Може да отвори всяка една.
* Преглед на продуктите, датата и статуса.

## 🔹 4.9 Административен панел (Django Admin)

За служителите и администраторите:

* Достъпен е на /admin.
* Използват се Django ModelAdmin класове за управление.
* Възможности:
  + Създаване и редакция на продукти, категории и вариации
  + Преглед на поръчки
  + Промяна на статуси
  + Преглед на транзакции и профили

**Пример:**

* Може да се промени статусът на поръчка от „Завършена“ на „Изпратена“ или „Приета“.
* В админ панела се виждат: дата, стойност, клиент, продукти, статус, транзакция.

## 🔹 4.10 Работа с имейли

* Всички важни действия (регистрация, нова поръчка, плащане) генерират **автоматични имейли**.
* Имейлите са HTML шаблони, изпратени чрез SMTP сървър.
* Работи се с EmailMessage от django.core.mail.

# ГЛАВА 5: ЗАКЛЮЧЕНИЕ

## 🔷 5.1 Обобщение на проекта

В настоящата дипломна работа беше разгледан, проектиран и реализиран цялостен уеб базиран софтуерен продукт за автоматизация на процеса по **поръчване и плащане в заведения** от типа ресторанти, барове и кафенета. Разработеното приложение, наречено **SmartBar**, цели да реши реален проблем, свързан с ръчната обработка на поръчки, бавното обслужване, човешките грешки и липсата на дигитална инфраструктура в редица търговски обекти от този тип.

Проектът следва последователен инженерен процес: анализ на проблема и нуждите, проектиране на система, изграждане на архитектура и база от данни, имплементиране на пълна функционалност за клиентска и административна работа, интегриране на реално работеща платежна система, както и визуално стилизиране на интерфейса.

## 🔷 5.2 Постигнати резултати

В хода на дипломната работа бяха постигнати следните **конкретни резултати**:

**✅ Успешно реализирани функционалности:**

* 📦 **Уеб интерфейс за разглеждане и избор на продукти**, с визуално меню, категории и възможности за вариации (напр. размер, вкус и др.).
* 🛒 **Динамична количка**, в която потребителят може да добавя, премахва и променя броя на избрани продукти.
* 🧾 **Форма за завършване на поръчка**, съдържаща всички необходими полета за доставка и допълнителна информация.
* 💳 **Интеграция с платежна система (PayPal)**, позволяваща **реални онлайн плащания** в евро.
* 📬 **Изпращане на имейл известия**, свързани с регистрация, потвърждение на поръчка и успешна транзакция.
* 👤 **Потребителски акаунти** с регистрация, вход, история на поръчките и профилни данни.
* 🛠️ **Административен панел**, достъпен само за администратори, където могат да се управляват:
  + Категории и продукти
  + Вариации
  + Поръчки и техните статуси
  + Потребители
* 🧾 **Генериране на фактура** с възможност за отпечатване, видима след извършване на плащане.

## 🔷 5.3 Използвани технологии и практики

Разработката на приложението беше извършена с използването на **модерен технологичен стек**, който включва:

| **Компонент** | **Технология / Инструмент** |
| --- | --- |
| Backend | Django (Python) |
| Frontend | HTML5, CSS3, Bootstrap, JavaScript |
| База от данни | SQLite (в dev среда) |
| Онлайн плащания | PayPal API |
| Имейли | Gmail SMTP чрез EmailMessage |
| Хостинг/деплой | Локален + готовност за AWS или PythonAnywhere |
| Работа със сесии | Django Sessions |
| Потребителски роли | Django Custom User Model |

Бяха приложени добри практики като:

* Разделяне на логиката в отделни Django приложения (apps)
* Използване на шаблони и наследяване с {% extends %}
* Защита срещу CSRF атаки
* Проверка на данни и валидиране във формите
* Съхранение на чувствителни настройки (като имейл пароли) чрез .env и python-decouple

## 🔷 5.4 Ползи от системата SmartBar

Системата SmartBar предлага редица **преимущества както за клиентите, така и за бизнеса**, включително:

**➤ За клиентите:**

* Удобен, интуитивен интерфейс за поръчване
* Спестено време чрез бързо онлайн обслужване
* Възможност за безконтактно плащане
* История на направените поръчки
* Получаване на имейл потвърждение и фактура

**➤ За бизнеса (заведенията):**

* Автоматизация на поръчките
* Намаляване на човешките грешки при обслужване
* Статистика и проследимост на всички транзакции
* Лесно добавяне и актуализиране на продукти и категории
* Възможност за разширение с QR менюта и мобилни приложения

## 🔷 5.5 Проблеми и предизвикателства

В процеса на разработка възникнаха и редица технически и логически предизвикателства, сред които:

* Налагане на уникални ограничения в базата (напр. уникален username или slug)
* Изпращане на имейли от Gmail с по-строги SMTP политики
* Интеграция на PayPal с правилна валута (изискващ коректен currency\_code)
* Работа със session колички за неавторизирани потребители
* Кеширане на статични файлове (проблем при промяна на снимки)

Всички те бяха решени чрез дебъгване, тестване, четене на документация и сравнение с примери от реални системи.

## 🔷 5.6 Възможности за бъдещо развитие

Системата SmartBar е проектирана с отворена архитектура, което позволява **лесно надграждане** и добавяне на допълнителна функционалност. Някои от възможните посоки са:

**📱 Мобилно приложение**

* Разработка на мобилна версия (Android/iOS), комуникираща чрез REST API
* Уведомления чрез push нотификации

**📈 Анализи и статистика**

* Добавяне на админ панел с графики: най-продавани продукти, обороти по дни/седмици

**🧾 Фискална интеграция**

* Връзка с касов апарат / фискален принтер

**📡 QR код поръчки**

* Генериране на уникален QR код за всяка маса
* Поръчка директно от телефон без нужда от сервитьор

**🌐 Многоезичност**

* Превод на интерфейса на повече от един език (български, английски, немски)

**💰 Разнообразие от платежни методи**

* Добавяне на допълнителни разплащателни системи: Stripe, Revolut Pay, банкови карти

## 🔷 5.7 Личен принос

Разработката на проекта беше извършена изцяло самостоятелно от дипломанта, като включва:

* Анализ и дефиниране на реален проблем
* Планиране и изграждане на архитектурата на системата
* Програмиране на backend и frontend логика
* Интеграция с външни услуги (PayPal, Gmail SMTP)
* Провеждане на тестове и отстраняване на бъгове
* Пълна подготовка на документацията

**📌 Заключителни думи**

Системата SmartBar е доказателство, че чрез съвременни технологии и ясен инженерно-софтуерен подход може да се създаде **реално работещо, полезно и приложимо решение**, което да автоматизира бизнес процеси и подобри потребителското изживяване. Разработеният продукт е стабилна основа, която може да бъде внедрена, надградена и персонализирана според нуждите на всяко заведение.

Тази дипломна работа доказа на практика значимостта на уеб технологиите и демонстрира уменията на дипломанта в пълен цикъл от анализ, разработка, внедряване и документация на софтуерен продукт.

# ГЛАВА 6: ИЗПОЛЗВАНИ ИЗТОЧНИЦИ И ПРИЛОЖЕНИЯ

**Списък на използвани литературни и онлайн източници**

1. Django Software Foundation. *Django Documentation (v5.2)*.  
   Достъпно на: <https://docs.djangoproject.com/en/5.2/>
2. Python Software Foundation. *Python 3.13 Official Documentation*.  
   Достъпно на: <https://docs.python.org/3/>
3. PayPal Developer Portal. *PayPal Checkout Integration Guide*.  
   Достъпно на: <https://developer.paypal.com/docs/checkout/>
4. Bootstrap. *Bootstrap v5.3 Documentation*.  
   Достъпно на: <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
5. W3Schools. *HTML, CSS, JavaScript Tutorials*.  
   Достъпно на: <https://www.w3schools.com/>
6. Mozilla Developer Network (MDN). *JavaScript and Web APIs Documentation*.  
   Достъпно на: <https://developer.mozilla.org/en-US/docs/Web>
7. GitHub – проект на дипломанта: *SmartBar – Django-based Web Ordering System*.  
   Достъпно на: <https://github.com/boriskostadinov/greatbar-django>
8. Django Creating forms from models *ModelForm*  
   Достъпно на: <https://docs.djangoproject.com/en/5.2/topics/forms/modelforms/>
9. Django The messages framework

Достъпно на: <https://docs.djangoproject.com/en/5.2/ref/contrib/messages/>

1. Real Python. *How to Work with Django Models, Forms, and Views*.  
   Достъпно на: <https://realpython.com/>
2. Corey Schafer – YouTube канал: *Django Tutorials*.  
   Достъпно на: <https://www.youtube.com/user/schafer5>
3. Django REST Framework. *Official Documentation*.  
   Достъпно на: <https://www.django-rest-framework.org/>
4. Postman. *Testing APIs and integration debugging*.  
   Достъпно на: <https://www.postman.com/>
5. decouple · PyPI. *Python-Decouple – Settings Management*.  
   Достъпно на: <https://pypi.org/project/python-decouple/>
6. PythonAnywhere – хостинг платформа за Django приложения.  
   Достъпно на: <https://www.pythonanywhere.com/>
7. Gmail SMTP Setup – *Google Workspace Admin Help*.  
   Достъпно на: <https://support.google.com/a/answer/176600?hl=en>