|  |
| --- |
| Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Новгородский государственный университет имени Ярослава Мудрого»  ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ  ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ |

ПРОИЗВОДСТВЕННАЯ ПРАКТИКА

ПМ.08 РАЗРАБОТКА ДИЗАЙНА ВЕБ-ПРИЛОЖЕНИЯ

09.02.07 Информационные системы и программирование

код и наименование специальности

ОТЧЁТ

|  |
| --- |
| Руководитель от организации:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_г. |
| Руководитель от колледжа:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_г. |
| Выполнил:  Студент гр. \_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_г. |

Содержание

[1 Постановка задачи на практику 4](#_Toc1)

[2 Техника безопасности на предприятии 5](#_Toc2)

[2.1. Общие требования охраны труда 5](#_Toc3)

[2.2. Требования охраны труда перед началом работы 8](#_Toc4)

[2.3. Требования охраны труда во время работы 8](#_Toc5)

[2.4 Требования охраны труда в аварийных ситуациях 10](#_Toc6)

[2.5 Требования охраны труда по окончании работы 11](#_Toc7)

[3 Должностные обязанности 12](#_Toc8)

[4 Программное обеспечение 14](#_Toc9)

[5 Спецификация программного продукта 15](#_Toc10)

[5.1 Техническое задание: gRPC API 15](#_Toc11)

[5.2 Разработка: gRPC API 15](#_Toc12)

[5.3 Тестирование: gRPC API 16](#_Toc13)

[5.4 Техническое задание: GraphQl API 17](#_Toc14)

[5.5 Разработка: GraphQl API 18](#_Toc15)

[5.6 Тестирование: GraphQl API 18](#_Toc16)

[5.7 Техническое задание: WebSocket 19](#_Toc17)

[5.8 Разработка: WebSocket 20](#_Toc18)

[5.9 Тестирование: WebSocket 21](#_Toc19)

[5.10 Техническое задание: Вёрстка по макету Figma 22](#_Toc20)

[5.11 Разработка: Вёрстка по макету Figma 23](#_Toc21)

[5.12 Тестирование: Вёрстка по макету Figma 23](#_Toc22)

[5.13 Руководство пользователя 24](#_Toc23)

[Выводы 27](#_Toc24)

[Приложение А 28](#_Toc25)

# 1 Постановка задачи на практику

Цель: Получение практического опыта в проектировании, разработке и тестировании современных веб-приложений с использованием различных технологий и подходов.

В процессе практики мне были поставлены следующие задачи:

— Разработка сервиса управления товарами на складе с использованием gRPC API;

— Реализация Inventory Management API на GraphQL;

— Разработка сервиса трансляции котировок акций в реальном времени на WebSocket;

— Вёрстка пользовательского интерфейса по макету Figma с применением Bootstrap и CSS.

# 2 Техника безопасности на предприятии

## 2.1. Общие требования охраны труда

К самостоятельной работе в качестве веб-программиста допускаются лица не моложе 18 лет, имеющие соответствующее образование, прошедшие медицинский осмотр и не имеющие противопоказаний по состоянию здоровья, вводный инструктаж по охране труда, проверку знаний требований охраны труда, обучение и проверку знаний правил пожарной безопасности в объеме должностных обязанностей.

Веб-программист обязан:

— знать и соблюдать требования настоящей инструкции, правила и нормы охраны труда и производственной санитарии, электробезопасности, правила внутреннего трудового распорядка;

— соблюдать правила поведения на территории и в помещениях предприятия;

— заботиться о личной безопасности и личном здоровье;

— выполнять требования пожаро- и взрывобезопасности, знать сигналы оповещения о пожаре, порядок действий при нем, места расположения средств пожаротушения и уметь пользоваться ими;

— знать месторасположение аптечки и уметь оказывать первую помощь пострадавшему;

— знать порядок действий в случае возникновения чрезвычайных происшествий.

Веб-программист должен проходить:

— периодический медицинский осмотр в соответствии с действующим законодательством РФ;

— очередную проверку знаний требований охраны труда не реже 1 раза в год.

В процессе работы на веб-программиста могут воздействовать следующие опасные и вредные производственные факторы:

— повышенные уровни электромагнитного излучения;

— повышенные уровни ультрафиолетового и инфракрасного излучения;

— повышенный уровень напряженности статического электричества;

— повышенное содержание положительных и отрицательных аэроионов в воздухе рабочей зоны;

— повышенный или пониженный уровень освещенности;

— повышенный уровень прямой и отраженной блесткости;

— неравномерность распределения яркости в поле зрения;

— повышенная яркость светового изображения;

— повышенный уровень пульсации светового потока;

— повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека;

— повышенные уровни запыленности воздуха рабочей зоны;

— работы на высоте;

— острые кромки листов документации.

К оборудованию и помещению, где работает веб-программист, предъявляются следующие требования:

— металлические корпуса электропотребляющих установок должны быть заземлены (занулены). Категорически запрещается использовать в качестве контура заземления паропроводные, водопроводные, газовые, отопительные и другие трубы, радиаторы и т.п.;

— питающие электрические кабели должны иметь ненарушенную изоляцию и сечение, соответствующее передаваемой мощности;

— мебель, оргтехника, оборудование, расположенные в помещении, их окраска, а также окраска и конструкция стен помещения должны соответствовать эргономическим требованиям, а также нормам пожарной безопасности;

— помещение, в котором расположены ПК и сервер в соответствии с санитарными и другими нормами безопасности должно быть оборудовано: системами освещения (естественного или искусственного, общего или комбинированного – по необходимости); системой вентиляции, кондиционирования и отопления; устройствами защиты от поражения электрическим током (защитным заземлением, занулением), защитным отключением – УЗО; первичными средствами пожаротушения;

— оборудование должно содержаться в исправном состоянии, размещаться в предусмотренных технологическим процессом местах, не мешать работе, свободному проходу и проезду;

— на все виды используемого оборудования должны быть инструкции по эксплуатации, содержащие требования по безопасному обслуживанию;

— оборудование должно использоваться по назначению и применяться в условиях, установленных предприятием-изготовителем.

Веб-программист должен знать и соблюдать правила личной гигиены. Принимать пищу, курить, отдыхать только в специально отведенных для этого помещениях и местах. Пить воду только из специально предназначенных для этого установок.

Запрещается употребление спиртных напитков и появление на работе в нетрезвом состоянии, в состоянии наркотического или токсического опьянения.

Веб-программист обязан немедленно извещать руководителя о любой ситуации, угрожающей жизни и здоровью людей, о каждом несчастном случае, происшедшем на производстве, или об ухудшении своего здоровья, в том числе о появлении острого профессионального заболевания (отравления), а также обо всех замеченных неисправностях оборудования, устройств.

Требования настоящей инструкции по охране труда являются обязательными для веб-программиста. Невыполнение этих требований рассматривается как нарушение трудовой дисциплины и влечет ответственность согласно действующему законодательству РФ.

## 2.2. Требования охраны труда перед началом работы

Произвести визуальный осмотр ПК, убедиться в исправности электророзеток, штепсельных вилок, питающих электрошнуров.

Убедиться, что вентиляционные отверстия в корпусе включаемого оборудования не завалены бумагой и не перекрыты каким-либо другим способом.

Включить ПК в сеть 220В, при этом штепсельную вилку держать за корпус.

Запрещается подключать сетевые устройства мокрыми руками.

Отрегулировать освещенность на рабочем месте, убедиться в достаточности освещенности, отсутствии отражений на экране, отсутствии встречного светового потока.

При обнаружении каких-либо неисправностей оборудования, электропроводки сообщить об этом руководителю и до их устранения к работе не приступать.

## 2.3. Требования охраны труда во время работы

Подчиняться правилам внутреннего трудового распорядка, иным документам, регламентирующим вопросы дисциплины труда.

В течение всего рабочего времени содержать в порядке и чистоте рабочее место.

Держать открытыми все вентиляционные отверстия устройств.

Выполнять санитарно-гигиенические требования.

Соблюдать правила эксплуатации оборудования в соответствии с инструкциями по эксплуатации.

Соблюдать расстояние от глаз до экрана в пределах 60-80 см.

Следить, чтобы не загромождались проходы между оборудованием, приборами и рабочими местами, а также пути эвакуации.

Переключение разъемов интерфейсных кабелей периферийных устройств производить только при выключенном питании.

При необходимости использования лестниц-стремянок следует предварительно проверить их исправность. Запрещается применять случайные подставки (ящики, стулья и т.д.). Лестницы-стремянки должны иметь на тетивах резиновые башмаки.

При выполнении работ на высоте свыше 1,3 метров веб-программист обязан использовать монтажный пояс.

Во время работы запрещается:

— прикасаться к задней панели системного блока (процессора) при включенном питании;

— производить переключение разъемов интерфейсных кабелей периферийных устройств при включенном питании;

— загромождать верхние панели устройств бумагами и посторонними предметами;

— допускать попадание влаги на поверхность системного блока (процессора), монитора, рабочую поверхность клавиатуры, дисководов, принтеров и др. устройств;

— захламлять рабочее место бумагой и другими предметами во избежание накапливания пыли;

— выполнять работу на высоте с настилов и на случайных подставках (ящиках, бочках и т.п.).

Соблюдать правила поведения на территории и в помещениях предприятия.

Не принимать пищу, не курить на рабочем месте.

В случае плохого самочувствия прекратить работу, поставить в известность руководителя и обратиться к врачу.

## 2.4 Требования охраны труда в аварийных ситуациях

В любых аварийных ситуациях или при возникновении опасности для жизни и здоровья работников следует остановить работу, сообщить руководителю, принять меры к устранению таких ситуаций и опасностей.

При обнаружении на металлических частях оборудования напряжения (ощущение действия электротока) необходимо отключить оборудование от сети и доложить руководителю.

При обнаружении дыма и возникновении пожара немедленно объявить пожарную тревогу, принять меры к ликвидации пожара с помощью имеющихся первичных средств пожаротушения, поставить в известность руководителя. При необходимости вызвать пожарную бригаду.

Запрещается применять воду и пенные огнетушители для тушения электропроводок и оборудования под напряжением, так как пена является хорошим проводником электрического тока. Для этих целей используются углекислотные и порошковые огнетушители.

В условиях задымления и наличия огня в помещении передвигаться вдоль стен, согнувшись или ползком; для облегчения дыхания рот и нос прикрыть платком (тканью), смоченной водой; через пламя передвигаться, накрывшись с головой верхней одеждой или покрывалом, по возможности облиться водой, загоревшуюся одежду сорвать или погасить, а при охвате огнем большей части одежды плотно закатать работника в ткань (кошму), но не накрывать с головой.

При несчастном случае немедленно освободить пострадавшего от действия травмирующего фактора, соблюдая собственную безопасность, оказать пострадавшему первую помощь, при необходимости вызвать бригаду скорой помощи. По возможности сохранить обстановку, при которой произошел несчастный случай, если это не угрожает жизни и здоровью окружающих, для проведения расследования причин возникновения несчастного случая, или зафиксировать на фото или видео. Сообщить руководителю и специалисту по охране труда.

В случае ухудшения самочувствия, появления рези в глазах, резком ухудшении видимости - невозможности сфокусировать взгляд или навести его на резкость, появлении боли в пальцах и кистях рук, усилении сердцебиения немедленно покинуть рабочее место, сообщить о произошедшем руководителю и обратиться в медицинское учреждение.

## 2.5 Требования охраны труда по окончании работы

Отключить питание компьютера и всех периферийных устройств от электросети.

Привести в порядок рабочее место, убрать отходы бумаги.

Вымыть руки с мылом.

Сообщить руководителю обо всех нарушениях и замечаниях, выявленных в процессе работы, и принятых мерах по их устранению.

# 3 Должностные обязанности

Должностные обязанности Веб-разработчика:

— Разработка и поддержка пользовательского интерфейса: Веб-разработчик отвечает за создание и поддержку веб-интерфейса, который пользователи видят и с которым взаимодействуют. Он должен обеспечивать высокое качество пользовательского опыта, создавая удобный и интуитивно понятный интерфейс;

— Реализация дизайн-макетов: Веб-разработчик должен иметь навыки верстки и превращения графических дизайнов в рабочий код. Он должен точно воспроизводить внешний вид и стиль, заданные дизайнером, и обеспечивать их совместимость с различными браузерами и устройствами;

— Разработка интерактивных элементов: Веб-разработчик отвечает за создание интерактивных элементов интерфейса, таких как кнопки, формы, выпадающие списки и т. д. Он должен обеспечивать правильное функционирование и отзывчивость таких элементов, а также их валидацию и обработку данных;

— Интеграция с бэкендом: Веб-разработчик должен уметь взаимодействовать с серверной частью приложения, обращаться к API и обрабатывать полученные данные. Он должен понимать основы работы с HTTP-запросами, обработку ошибок и управление состоянием приложения;

— Оптимизация производительности и загрузки страниц: Веб-разработчик должен стремиться к оптимизации производительности веб-страниц, уменьшению размера файлов, снижению числа запросов к серверу и использованию кэширования. Он также должен обеспечивать быструю загрузку и отзывчивость интерфейса для улучшения пользовательского опыта;

— Тестирование и отладка: Веб-разработчик должен создавать и запускать тесты для проверки правильности работы интерфейса и обнаружения возможных ошибок. Он также должен уметь искать и устранять проблемы в коде, связанные с отображением и взаимодействием с элементами интерфейса;

— Кросс-браузерная и кросс-платформенная совместимость: Веб-разработчик должен проверять и обеспечивать совместимость создаваемого интерфейса с различными браузерами и устройствами. Он должен использовать современные стандарты и технологии, а также учитывать особенности различных платформ и устройств;

— Коллаборация с другими разработчиками и командами: Веб-разработчик должен эффективно работать в команде, взаимодействовать с дизайнерами, бэкенд-разработчиками и другими участниками проекта. Он должен быть готов обмениваться идеями и находить компромиссы для достижения общей цели.

# 4 Программное обеспечение

Для выполнения поставленных задач использовалось следующее программное обеспечение:

— Операционная система: Fedora Linux;

— Языки разметки и стилей: HTML, CSS;

— Языки программирования: Python;

— Фреймворк: Bootstrap;

— IDE: Visual Studio Code;

— Браузеры для тестирования: Google Chrome, Mozilla Firefox.

# 5 Спецификация программного продукта

## 5.1 Техническое задание: gRPC API

Назначение: Создание сервиса управления складскими запасами (Inventory Management) с использованием gRPC, обеспечивающего эффективный учёт продукции и контроль остатков.

Функциональные требования:

— Сервис должен позволять добавлять, редактировать и удалять товары;

— Пользователь должен иметь возможность изменять количество товара на складе;

— Сервис должен предоставлять информацию о текущем количестве каждого товара;

— Сервис должен поддерживать просмотр списка всех товаров;

— Сервис должен реализовывать потоковое оповещение о низком остатке товара (StreamStockAlerts).

Нефункциональные требования:

— Сервис должен иметь простой и понятный API, описанный в proto-файле;

— Сервис должен быть реализован с использованием Python и библиотеки grpcio;

— Для хранения данных может использоваться in-memory структура или JSON-файл;

— Сервис должен быть легко разворачиваемым и тестируемым.

## 5.2 Разработка: gRPC API

Реализация Inventory Management API выполнена с использованием Python и gRPC:

— Proto-файл определяет сущности Product и StockChange, а также сервис InventoryService с необходимыми методами;

— Сервер реализует методы AddProduct, GetProduct, UpdateStock, RemoveProduct, ListProducts, StreamStockAlerts;

— Для хранения данных используется products.json;

— Клиент реализует примеры вызова всех методов, включая стриминговые;

— Все взаимодействие происходит по gRPC-протоколу.

## 5.3 Тестирование: gRPC API

Таблица 1 - Добавление нового товара

|  |  |
| --- | --- |
| Поле | Описание |
| Название проекта | Inventory Management gRPC API |
| Рабочая версия | 1.0 |
| Имя тестирующего | Иван Иванов |
| Дата(ы) теста | 20/06/2025 |
| Тестовый пример # | TC\_INV\_01 |
| Приоритет тестирования *(Низкий/Средний/Высокий)* | Высокий |
| Заголовок/название теста | Добавление нового товара |
| Краткое изложение теста | Проверка функциональности добавления нового товара в систему. |
| Этапы теста | 1. Запустить сервер Inventory Management; 2. Отправить запрос AddProduct с данными нового товара; 3. Получить подтверждение добавления; 4. Проверить, что товар появился в списке товаров; 5. Убедиться, что количество товаров увеличилось на 1. |
| Тестовые данные | Название товара: "Молоко", Количество: 10 |
| Ожидаемый результат | Новый товар "Молоко" добавлен в систему, отображается в списке товаров, количество товаров увеличилось на 1. |
| Фактический результат | Новый товар "Молоко" добавлен в систему, отображается в списке товаров, количество товаров увеличилось на 1. |

Окончание таблицы 1

|  |  |
| --- | --- |
| Предварительное условие | Сервер запущен, клиент готов к отправке запросов. |
| Постусловие | Товар "Молоко" добавлен в систему. |
| Статус *(Зачет/Незачет)* | Зачёт |
| Примечания/комментарии |  |

## 5.4 Техническое задание: GraphQl API

Назначение: Реализация сервиса управления складскими запасами (Inventory Management) с использованием GraphQL, обеспечивающего структурированное, расширяемое и удобное API для работы с товарами и остатками.

Функциональные требования:

— Функциональные требования аналогичны gRPC-версии: добавление, редактирование, удаление товаров, изменение остатков, просмотр списка и информации о товарах, получение оповещений о низком остатке;

— Основное отличие — использование GraphQL для описания схемы, запросов, мутаций и подписок.

Нефункциональные требования:

— Сервис должен быть реализован с использованием Python (Strawberry) или другой подходящей технологии;

— Должна быть реализована валидация входных данных (Non-Null, типы, ограничения);

— API должно быть расширяемым и поддерживать подписки (Subscription) для оповещений.

## 5.5 Разработка: GraphQl API

Реализация Inventory Management API выполнена с использованием Python, библиотеки Strawberry и FastAPI.

Код построен следующим образом:

— Для описания GraphQL-типов (Product, StockChange, StockAlert) используются классы с аннотациями Strawberry;

— Для хранения и управления товарами используются отдельные функции (load\_products, save\_products), которые работают с JSON-файлом;

— Основная бизнес-логика (добавление, удаление, изменение остатков) реализована в методах резолверов GraphQL (Query, Mutation, Subscription);

— Для реализации подписок используется отдельный класс PubSub, обеспечивающий рассылку оповещений о низком остатке товара;

— Взаимодействие с клиентом осуществляется через GraphQL API, поддерживаются все основные операции управления товарами и подписка на события о низком остатке;

— Для хранения данных о товарах используется файл products.json.

## 5.6 Тестирование: GraphQl API

Таблица 2 - Оповещение о низком остатке товара

|  |  |
| --- | --- |
| Поле | Описание |
| Название проекта | Inventory Management (GraphQL) |
| Рабочая версия | 1.0 |
| Имя тестирующего | Иван Иванов |
| Дата(ы) теста | 20/06/2025 |
| Тестовый пример # | TC\_INV\_GQL\_02 |
| Приоритет тестирования *(Низкий/Средний/Высокий)* | Средний |

Окончание таблицы 2

|  |  |
| --- | --- |
| Заголовок/название теста | Проверка работы подписки на событие о низком остатке товара (streamStockAlerts). |
| Краткое изложение теста | Проверка работы подписки на событие о низком остатке товара (streamStockAlerts). |
| Этапы теста | 1. Запустить сервер Inventory Management (GraphQL); 2. Подписаться на streamStockAlerts с порогом, например, 5; 3. Уменьшить количество товара до значения ниже порога; 4. Убедиться, что пришло уведомление о низком остатке. |
| Тестовые данные | id товара: "milk", начальное количество: 10, порог: 5, уменьшить до 4 |
| Ожидаемый результат | После уменьшения количества товара до 4 приходит уведомление о низком остатке. |
| Фактический результат | После уменьшения количества товара до 4 приходит уведомление о низком остатке. |
| Предварительное условие | Сервер запущен, клиент подписан на streamStockAlerts. |
| Постусловие | Получено уведомление о низком остатке. |
| Статус *(Зачет/Незачет)* | Зачёт |
| Примечания/комментарии |  |

## 5.7 Техническое задание: WebSocket

Назначение: Разработка сервиса трансляции котировок акций в реальном времени с использованием протокола WebSocket.

Функциональные требования:

— Сервер должен поддерживать подключение нескольких клиентов одновременно;

— Клиенты могут подписываться на интересующие тикеры акций и получать обновления цен в реальном времени;

— Поддерживаются типы событий: subscribe (подписка на тикер), unsubscribe (отписка), priceUpdate (обновление цены);

— Формат сообщений — JSON с обязательными полями: тип события, тикер, цена (для priceUpdate), список тикеров (для subscribe/unsubscribe);

— Сервер должен обрабатывать ошибки (например, неверный формат сообщения, неизвестный тикер) и отправлять клиенту информативный ответ;

— Реализована аутентификация пользователя при подключении (например, по токену).

Нефункциональные требования:

— Сервер должен быть реализован на Python с использованием библиотеки websockets;

— Решение должно быть легко расширяемым для добавления новых типов событий и логики;

— Примеры JSON-сообщений и ожидаемые ответы сервера должны быть задокументированы для интеграции с разными клиентами.

## 5.8 Разработка: WebSocket

Сервис трансляции цен реализован на Python с использованием библиотеки websockets:

— Сервер обрабатывает подключения клиентов, поддерживает аутентификацию по токену при установлении соединения;

— Клиенты могут отправлять сообщения с типом subscribe или unsubscribe для управления списком интересующих тикеров;

— Сервер периодически отправляет сообщения типа priceUpdate с актуальной ценой для каждого подписанного тикера;

— Формат сообщений — JSON с полями: type (тип события), tickers (список тикеров для подписки/отписки), ticker и price (для обновления цены);

— Для обработки ошибок реализована валидация входящих сообщений и отправка клиенту ответа с описанием ошибки;

— В клиентском примере реализовано подключение к серверу, подписка на тикеры и отписка от тикеров, получение обновлений цен, обработка событий open, message, error, close, а также логирование событий в консоль.

## 5.9 Тестирование: WebSocket

Таблица 3 - Получение обновления цены по подписке

|  |  |
| --- | --- |
| Поле | Описание |
| Название проекта | Stock Price Broadcast WebSocket |
| Рабочая версия | 1.0 |
| Имя тестирующего | Иван Иванов |
| Дата(ы) теста | 20/06/2025 |
| Тестовый пример # | TC\_STOCK\_03 |
| Приоритет тестирования *(Низкий/Средний/Высокий)* | Высокий |
| Заголовок/название теста | Получение обновления цены по подписке |
| Краткое изложение теста | Проверка корректности получения обновлений цен по подписанному тикеру. |
| Этапы теста | 1. Запустить сервер WebSocket; 2. Подключить клиента; 3. Отправить сообщение с типом subscribe и тикером "AAPL"; 4. Убедиться, что клиент получает сообщения priceUpdate с актуальной ценой для "AAPL". |
| Тестовые данные | type="subscribe", tickers=["AAPL"], token: "demo\_token" |
| Ожидаемый результат | Клиент получает сообщения priceUpdate с тикером "AAPL" и актуальной ценой. |
| Фактический результат | Клиент получает сообщения priceUpdate с тикером "AAPL" и актуальной ценой. |
| Предварительное условие | Сервер запущен, клиент подключён. |
| Постусловие | Клиент подписан и получает обновления цен. |
| Статус *(Зачет/Незачет)* | Зачёт |
| Примечания/комментарии |  |

## 5.10 Техническое задание: Вёрстка по макету Figma

Назначение: Разработка адаптивного и современного пользовательского интерфейса веб-приложения на основе предоставленного макета Figma.

Функциональные требования:

— Интерфейс должен полностью соответствовать визуальному стилю и структуре макета Figma;

— Все элементы (шапка, меню, карточки, кнопки, формы и т.д.) должны быть реализованы согласно макету;

— Вёрстка должна быть адаптивной и корректно отображаться на различных устройствах и разрешениях;

— Используются современные подходы к вёрстке: Flexbox, Grid, медиазапросы;

— Все интерактивные элементы должны иметь стили состояний (hover, active, focus) согласно макету;

— Шрифты, цвета, отступы и размеры должны соответствовать спецификации Figma.

Нефункциональные требования:

— Все страницы и компоненты сверстаны в полном соответствии с макетом;

— Для построения сеток и позиционирования элементов применяются классы Bootstrap, Flexbox и CSS Grid;

— Реализована адаптивность с помощью медиазапросов, обеспечена корректная работа на мобильных и десктопных устройствах;

— Проектная структура и имена классов логичны и соответствуют методологии.

## 5.11 Разработка: Вёрстка по макету Figma

Вёрстка выполнена на основе макета Figma с использованием HTML, Bootstrap 5 и CSS:

— Все страницы и компоненты сверстаны в полном соответствии с макетом;

— Для построения сеток и позиционирования элементов применяются классы Bootstrap, Flexbox и CSS Grid;

— Реализована адаптивность с помощью медиазапросов, обеспечена корректная работа на мобильных и десктопных устройствах;

— Использованы кастомные шрифты, цвета и размеры, указанные в макете;

— Для интерактивных элементов реализованы стили состояний (hover, active, focus);

— Вся графика и иконки экспортированы из Figma и оптимизированы для web;

— Структура проекта организована по папкам (отдельные папки для стилей, изображений, компонентов).

## 5.12 Тестирование: Вёрстка по макету Figma

Таблица 4 - Проверка соответствия вёрстки макету

|  |  |
| --- | --- |
| Поле | Описание |
| Название проекта | Вёрстка по макету Figma |
| Рабочая версия | 1.0 |
| Имя тестирующего | Иван Иванов |
| Дата(ы) теста | 20/06/2025 |
| Тестовый пример # | TC\_LAYOUT\_04 |

Окончание таблицы 4

|  |  |
| --- | --- |
| Приоритет тестирования *(Низкий/Средний/Высокий)* | Высокий |
| Заголовок/название теста | Проверка соответствия вёрстки макету |
| Краткое изложение теста | Проверка визуального и структурного соответствия страниц макету Figma |
| Этапы теста | 1. Открыть страницу в браузере; 2. Сравнить с макетом Figma; 3. Проверить адаптивность; 4. Проверить стили интерактивных элементов; 5. Проверить корректность отображения на разных устройствах. |
| Тестовые данные | Макет Figma, различные разрешения экрана |
| Ожидаемый результат | Интерфейс полностью соответствует макету, корректно отображается на всех устройствах, интерактивные элементы работают согласно спецификации. |
| Фактический результат | Интерфейс соответствует макету, адаптивность и интерактивность реализованы корректно. |
| Предварительное условие | Макет Figma доступен, проект развернут локально |
| Постусловие | Вёрстка прошла визуальное и функциональное тестирование |
| Статус *(Зачет/Незачет)* | Зачёт |

|  |  |
| --- | --- |
| Примечания/комментарии |  |

## 5.13 Руководство пользователя

gRPC API:

— Как добавить товар: "Для добавления товара используйте метод gRPC AddProduct, передав название и количество. Пример запроса приведён в клиенте и README."

— Как получить товар: "Для получения товара используйте метод gRPC GetProduct, указав идентификатор товара."

— Как изменить остаток товара: "Для изменения количества товара используйте метод UpdateStock, указав идентификатор товара и изменение (delta)."

— Как удалить товар: "Для удаления товара вызовите метод RemoveProduct с идентификатором товара."

— Как просмотреть список товаров: "Вызовите метод ListProducts для получения всех товаров с возможностью пагинации."

— Как подписаться на оповещения о низком остатке: "Используйте стриминговый метод StreamStockAlerts, указав пороговое значение остатка."

GraphQl API:

— Как добавить товар: "mutation { addProduct(name: "Молоко", quantity: 10) { id name quantity } }"

— Как получить товар по id: "query { getProduct(id: "ID\_товара") { id name quantity } }"

— Как изменить остаток товара: "mutation { updateStock(productId: "ID\_товара", delta: -2) { id name quantity } }"

— Как удалить товар: "mutation { removeProduct(id: "ID\_товара") }"

— Как просмотреть список товаров: "query { listProducts(limit: 10, offset: 0) { id name quantity } }"

— Как подписаться на оповещения о низком остатке: "subscription { streamStockAlerts(threshold: 5) { product { name quantity } message } }"

WebSocket:

— Как подключиться к серверу: "Откройте клиент, укажите адрес WebSocket-сервера и выполните подключение."

— Как подписаться на тикер: "Отправьте сообщение в формате JSON: { "event": "subscribe", "tickers": ["AAPL", "GOOG"], "token": "demo\_token" } и сервер начнёт присылать обновления цен по этим тикерам."

— Как отписаться от тикера: "Отправьте сообщение: { "event": "unsubscribe", "tickers": ["AAPL"] } и сервер прекратит присылать обновления по этим тикерам."

— Как получать обновления цен: "После подписки клиент будет получать сообщения: { "event": "priceUpdate", "ticker": "AAPL", "price": 123.45 } и обновления будут отображаться в консоли."

— Как обрабатывать ошибки: "В случае некорректного сообщения сервер отправит ответ с типом "error" и описанием проблемы."

Вёрстка по макету Figma:

— Для просмотра интерфейса откройте файл index.html (или другую страницу) в браузере.

— Все элементы и взаимодействия реализованы согласно макету Figma с помощью Bootstrap и CSS.

— Дополнительных действий от пользователя не требуется.

# Выводы

В ходе прохождения производственной практики в ООО "41T" был получен практический опыт разработки веб-приложений с использованием современных технологий, таких как HTML, CSS, Python и Bootstrap.

В процессе работы были улучшены навыки адаптивной вёрстки, изучены принципы построения API на основе gRPC и GraphQL, освоены основы работы с WebSocket для реализации обмена данными в реальном времени, а также рассмотрены возможности использования фреймворков и библиотек для ускорения разработки.

Разработанные приложения — Inventory Management gRPC API, Inventory Management GraphQL API, Stock Price Broadcast WebSocket и Вёрстка по макету Figma — могут быть использованы компанией, например:

— как внутренние инструменты для повышения эффективности работы сотрудников;

— интегрированы в разрабатываемые веб-приложения для расширения функциональности

— как база для разработки более сложных приложений.

# Приложение А

(обязательное)

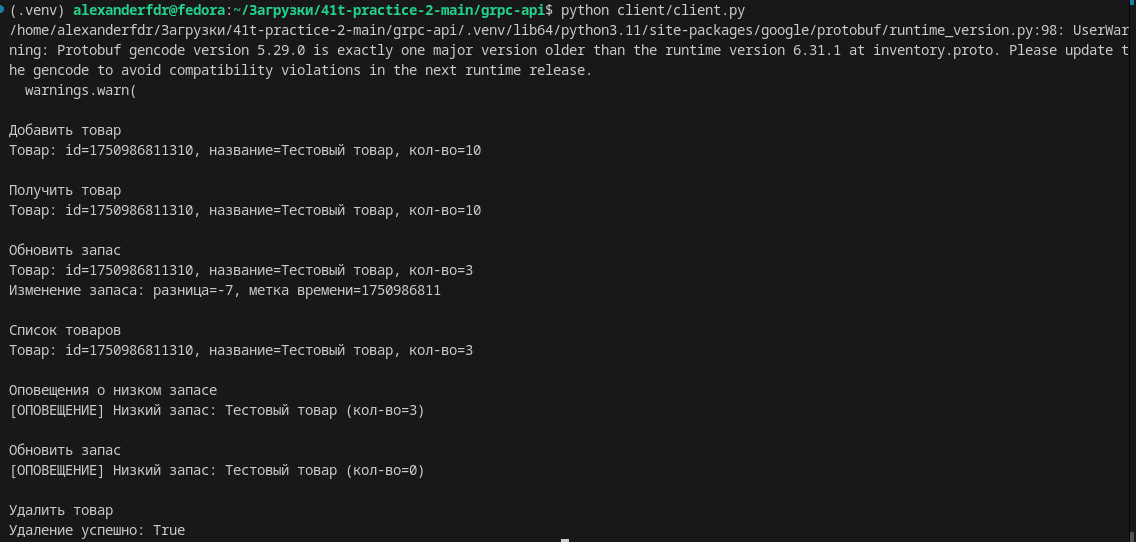


Рисунок А.1 – Логи консоли gRPC

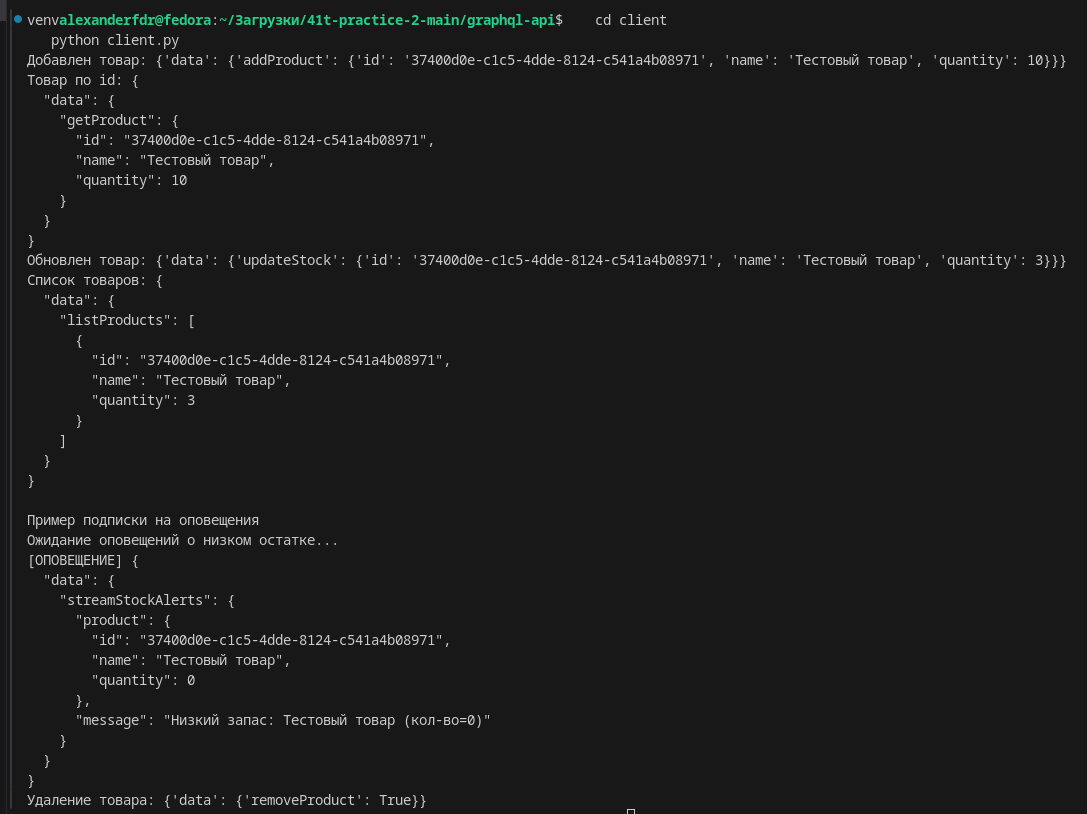


Рисунок А.2 – Логи консоли GraphQL



Рисунок А.3 – Mutation GraphQL (Strawberry)

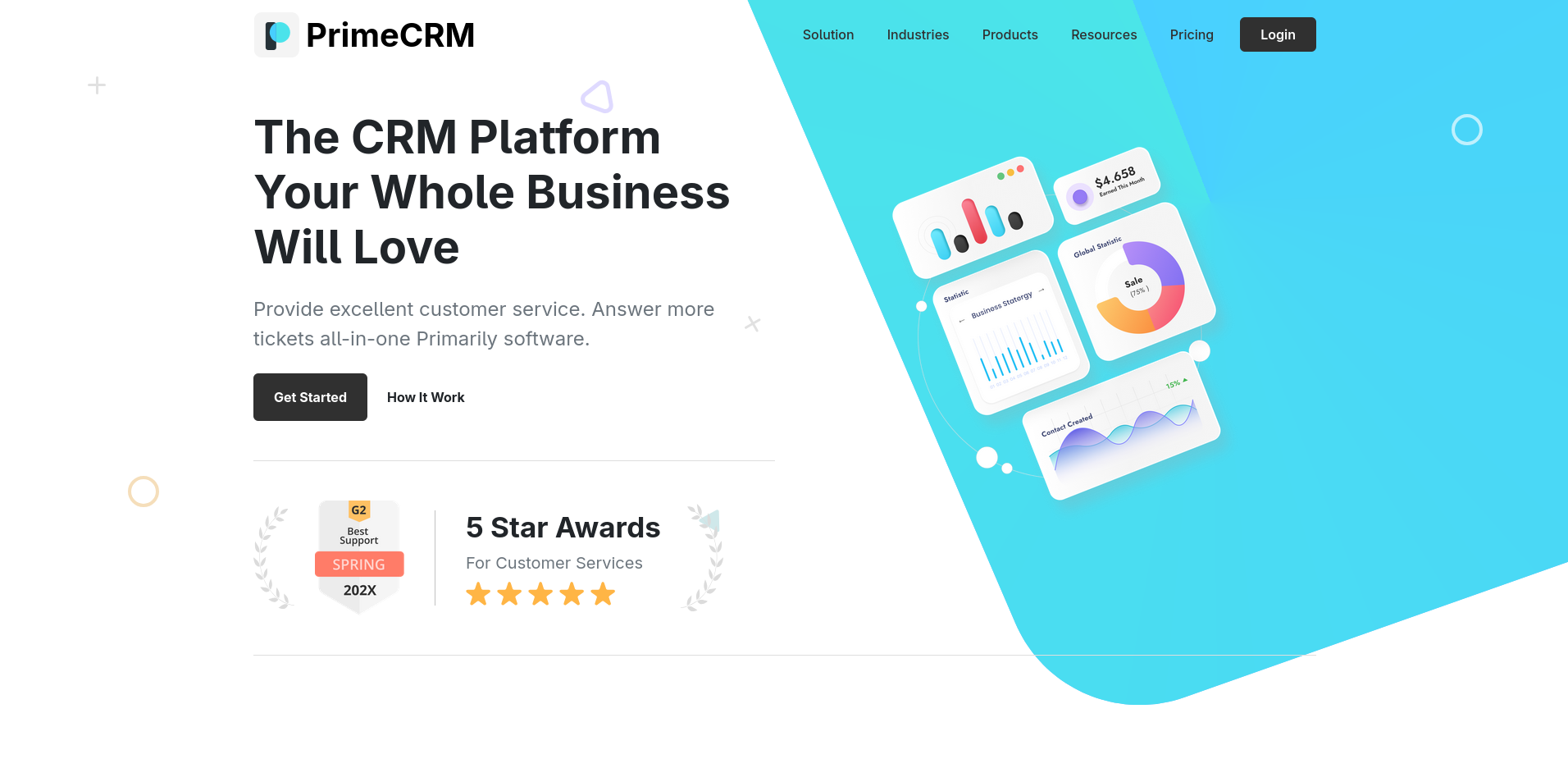


Рисунок А.4 – Вёрстка по макету Figma (Index)

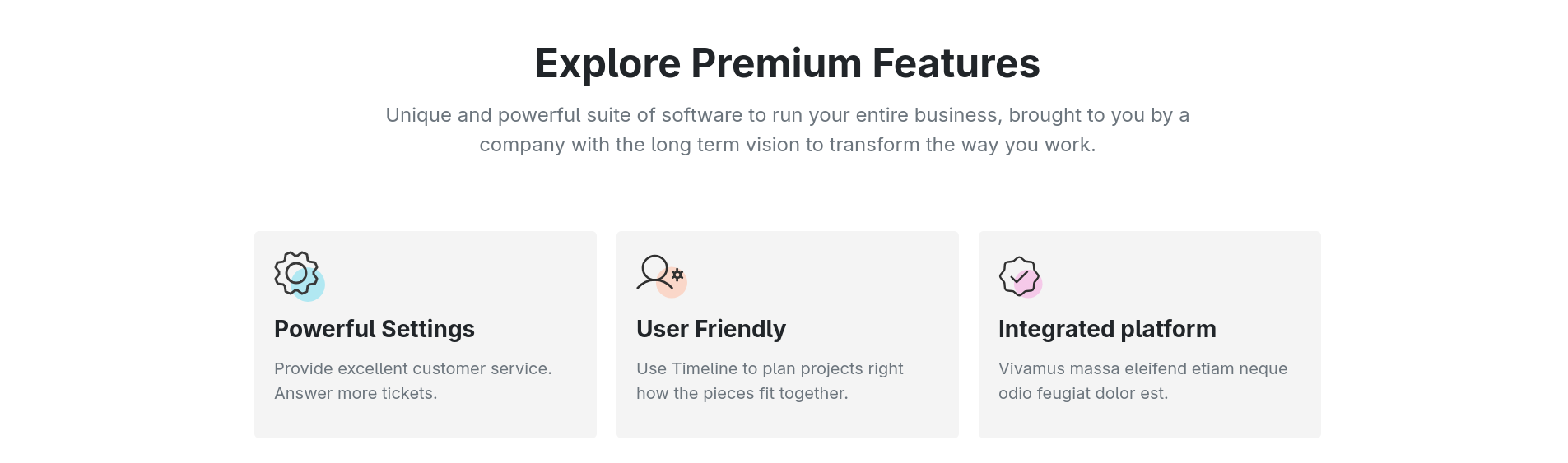


Рисунок А.5 – Вёрстка по макету Figma (Index)

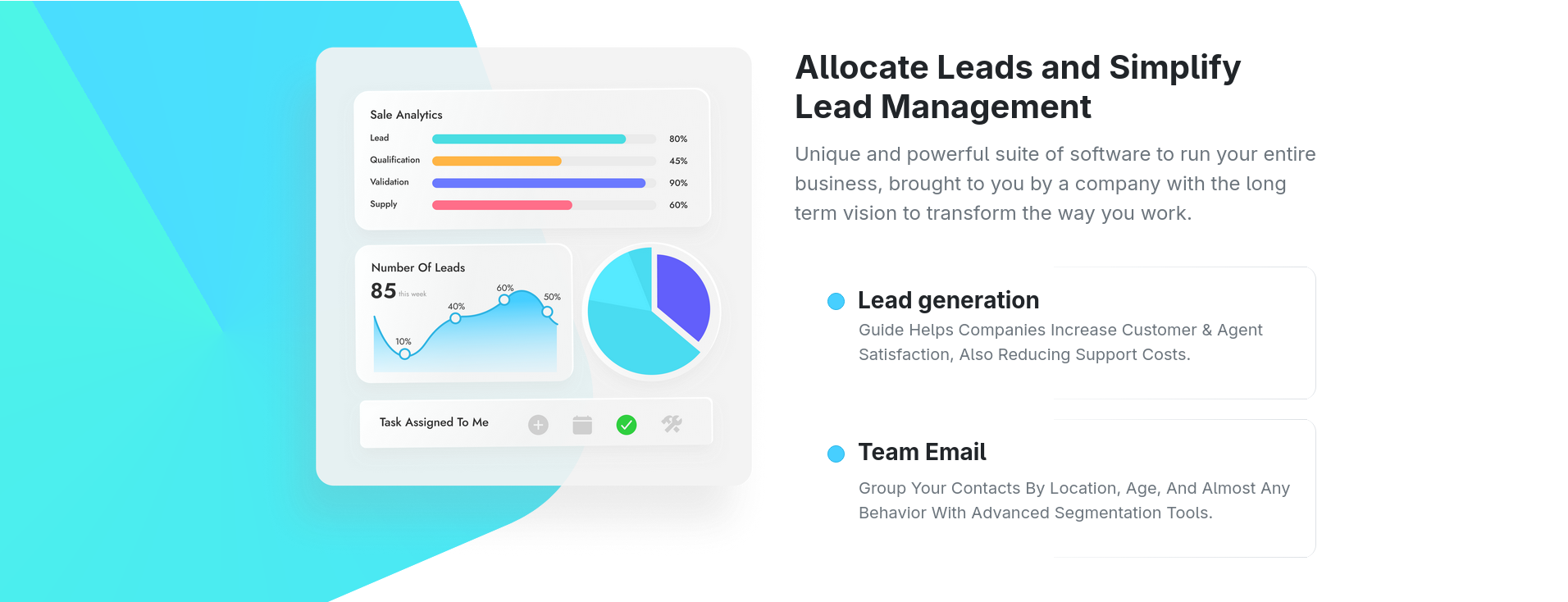


Рисунок А.6 – Вёрстка по макету Figma (Index)

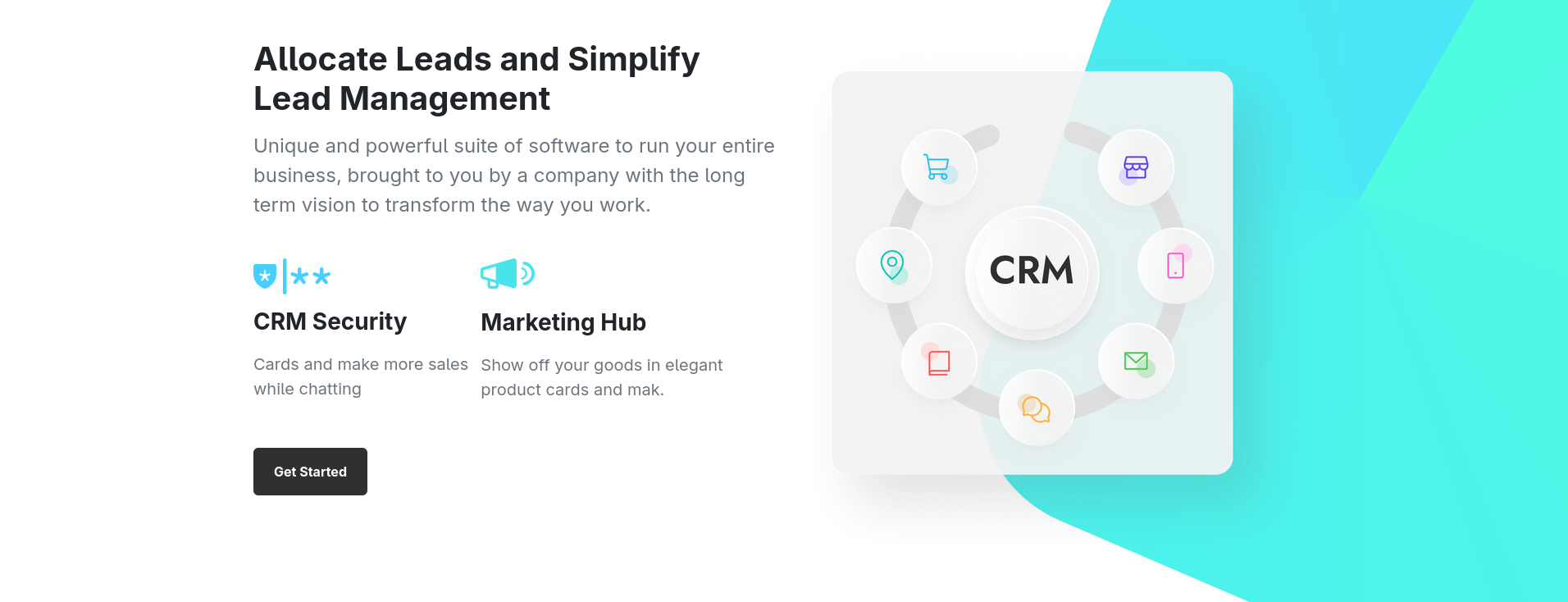


Рисунок А.7 – Вёрстка по макету Figma (Index)

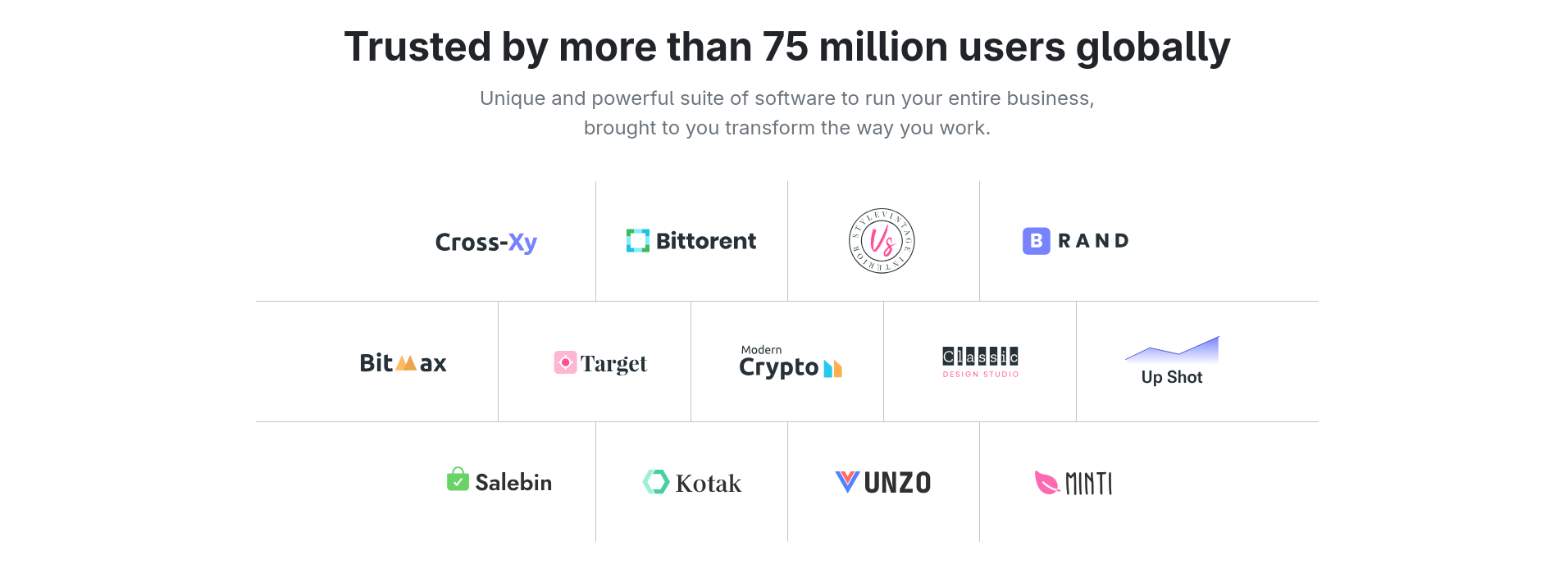


Рисунок А.8 – Вёрстка по макету Figma (Index)

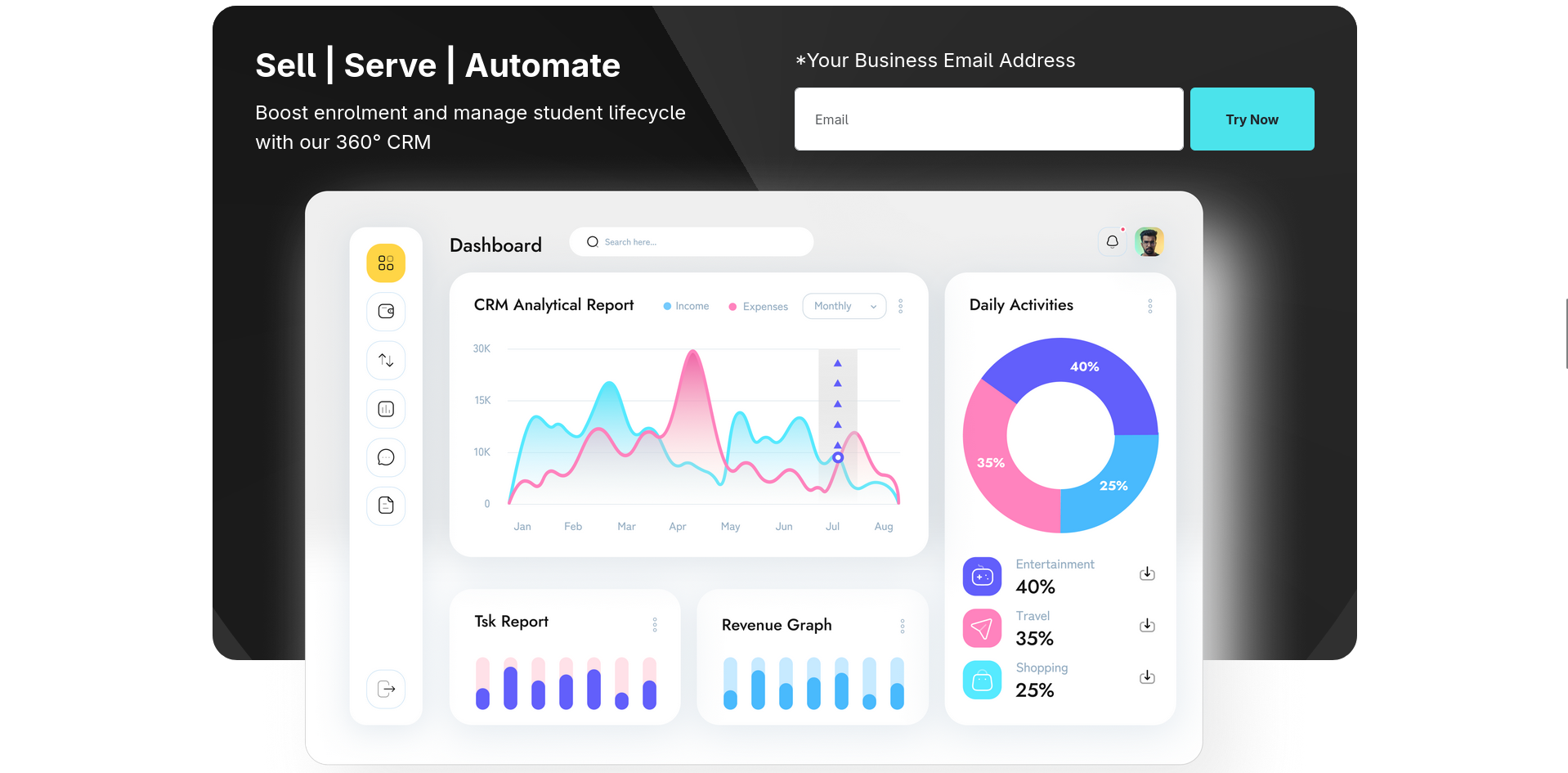


Рисунок А.9 – Вёрстка по макету Figma (Index)

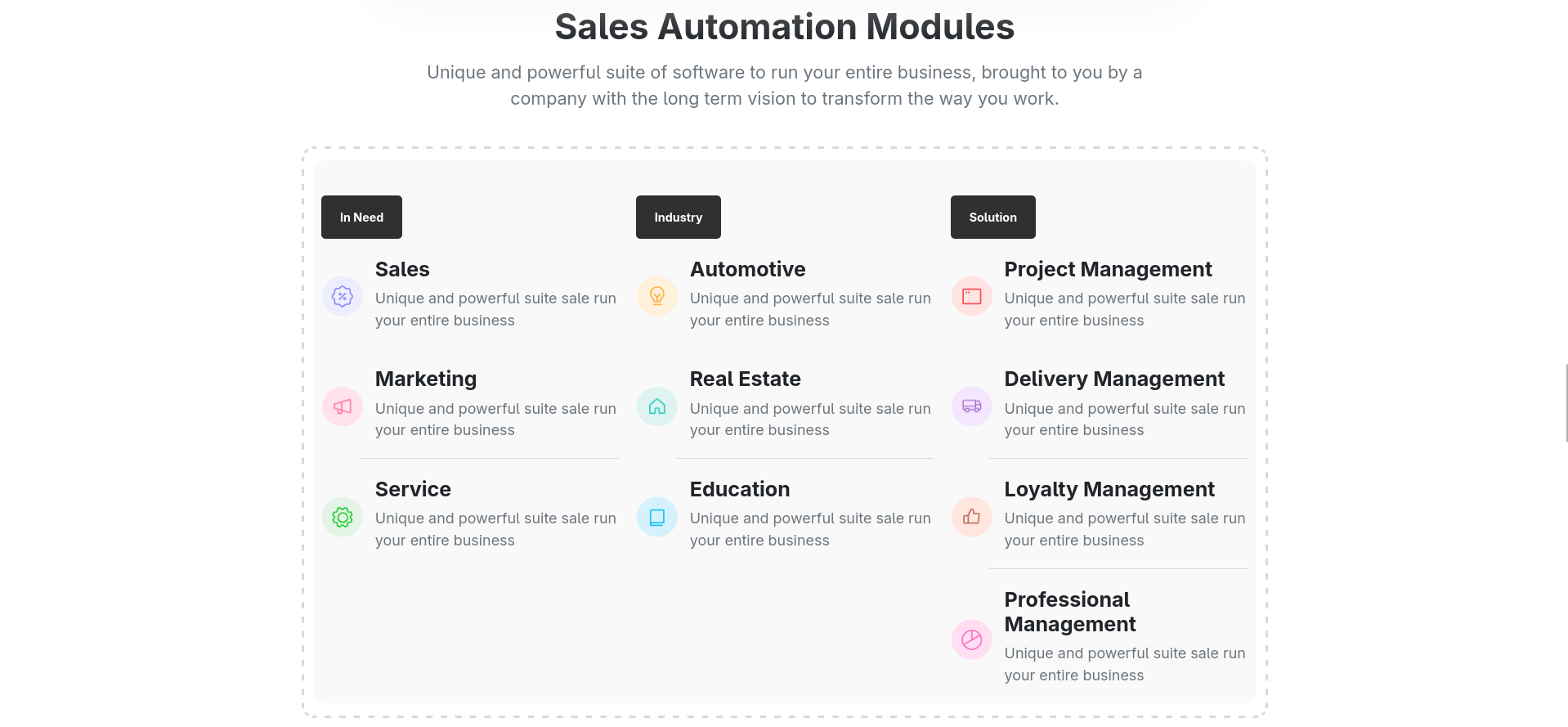


Рисунок А.10 – Вёрстка по макету Figma (Index)

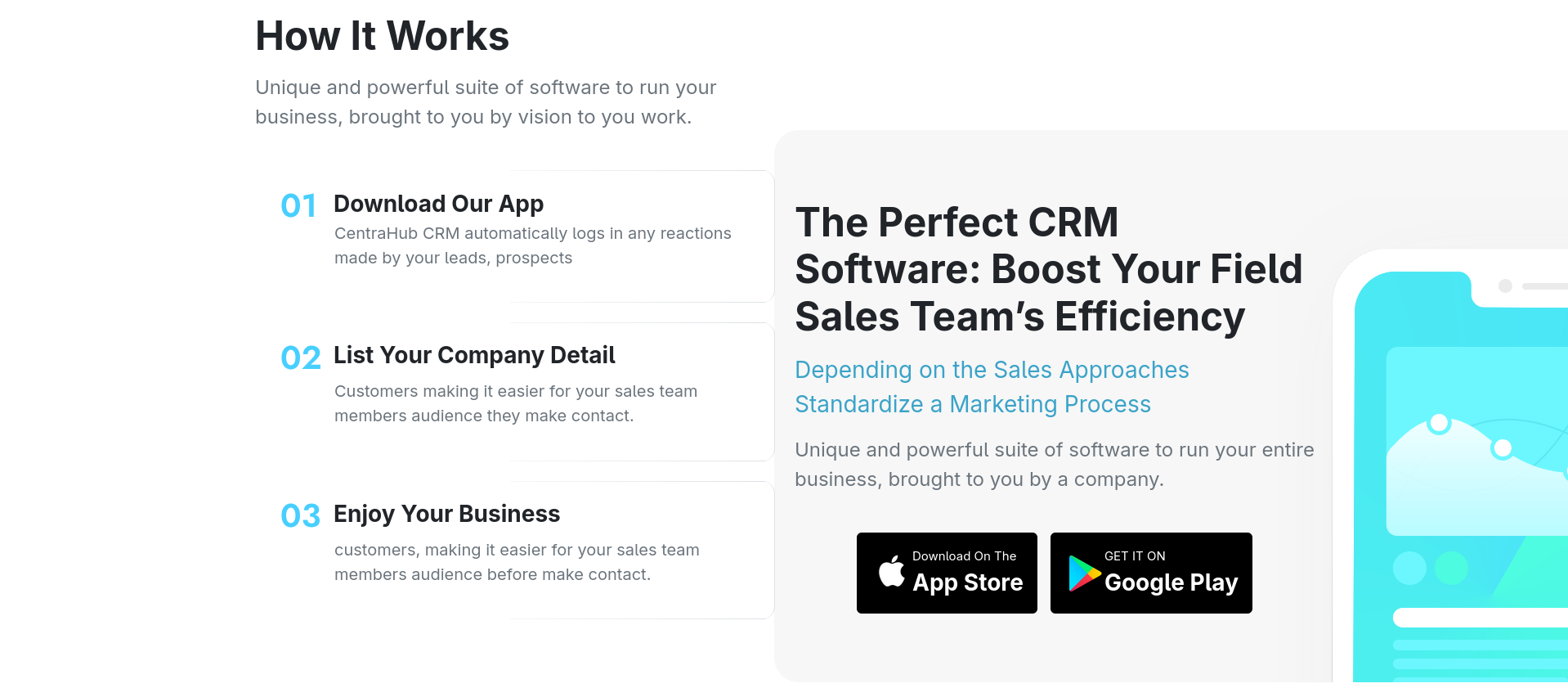


Рисунок А.11 – Вёрстка по макету Figma (Index)

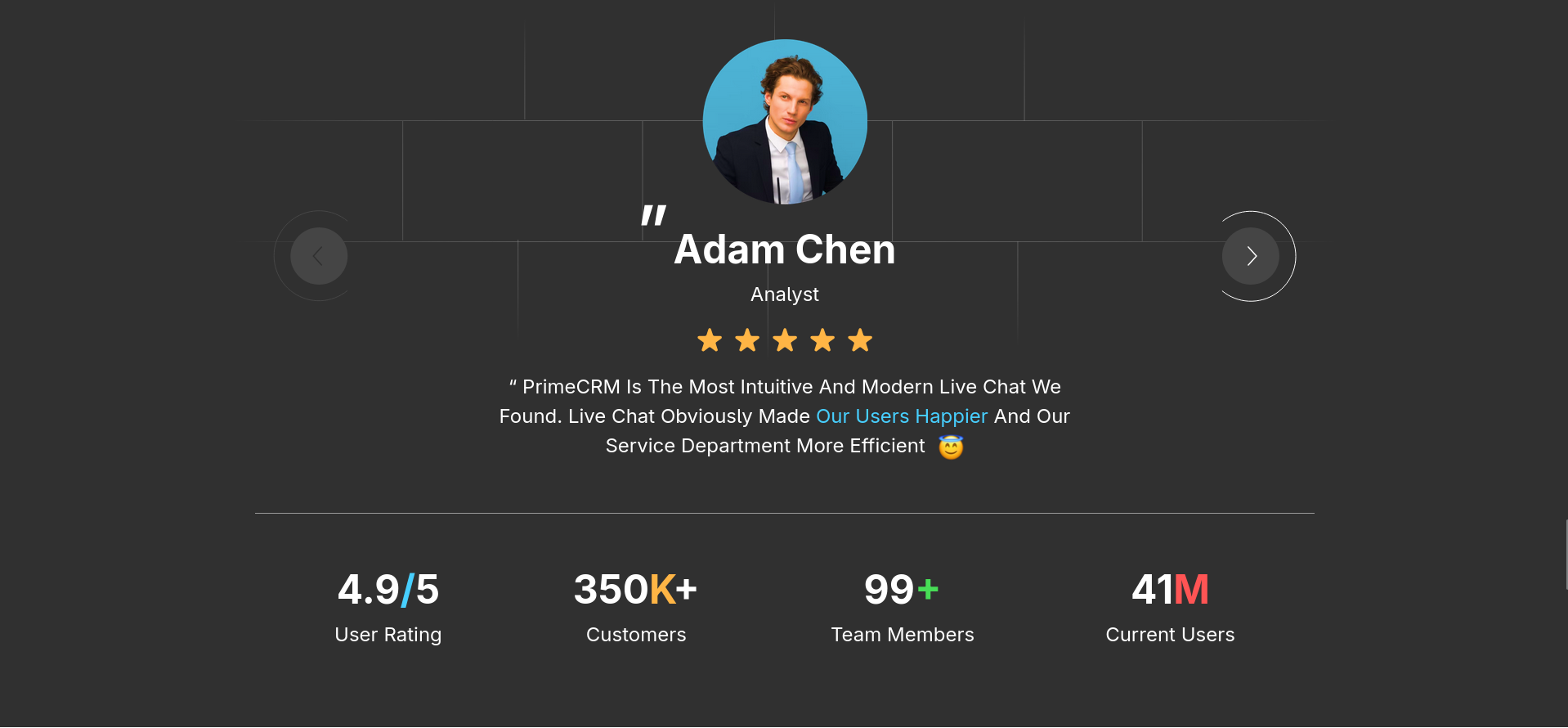


Рисунок А.12 – Вёрстка по макету Figma (Index)

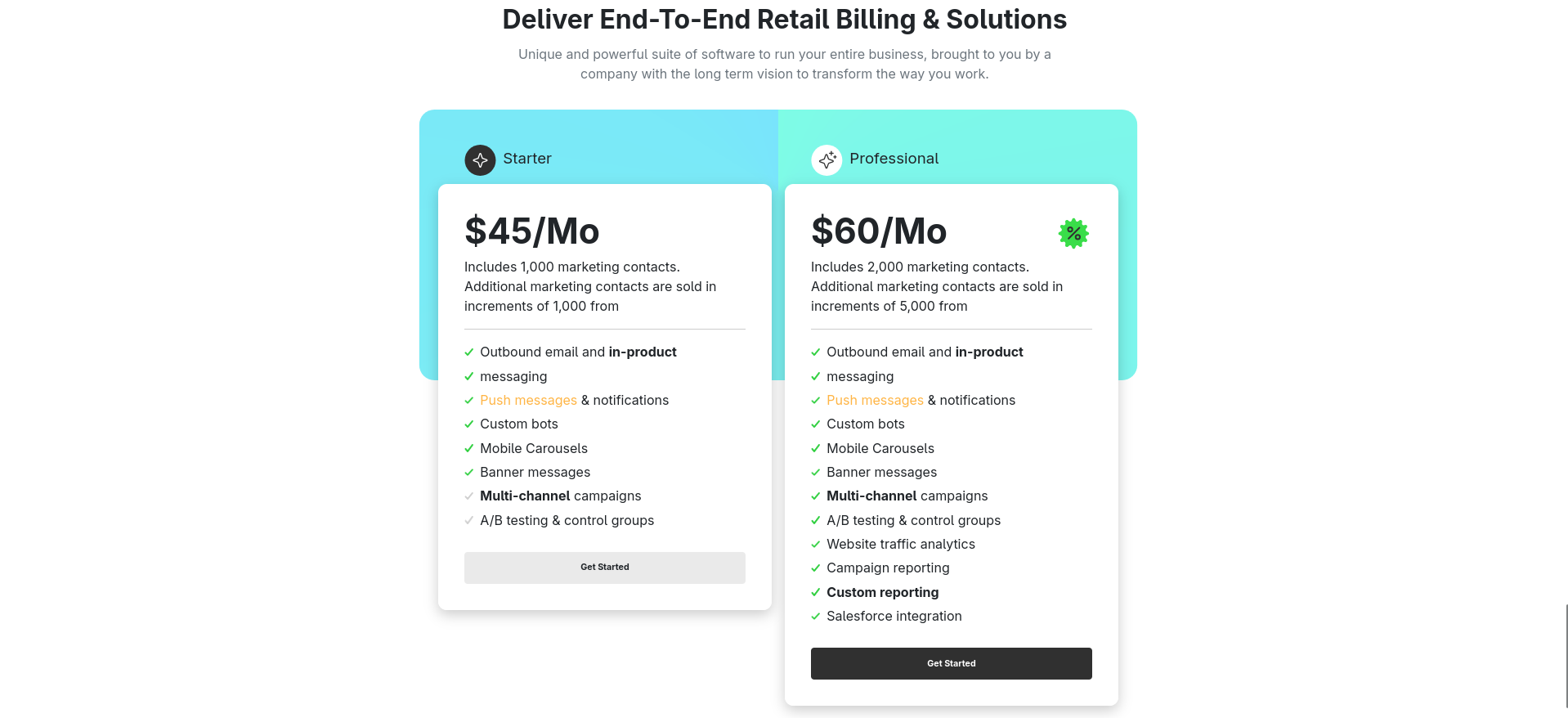


Рисунок А.13 – Вёрстка по макету Figma (Index)

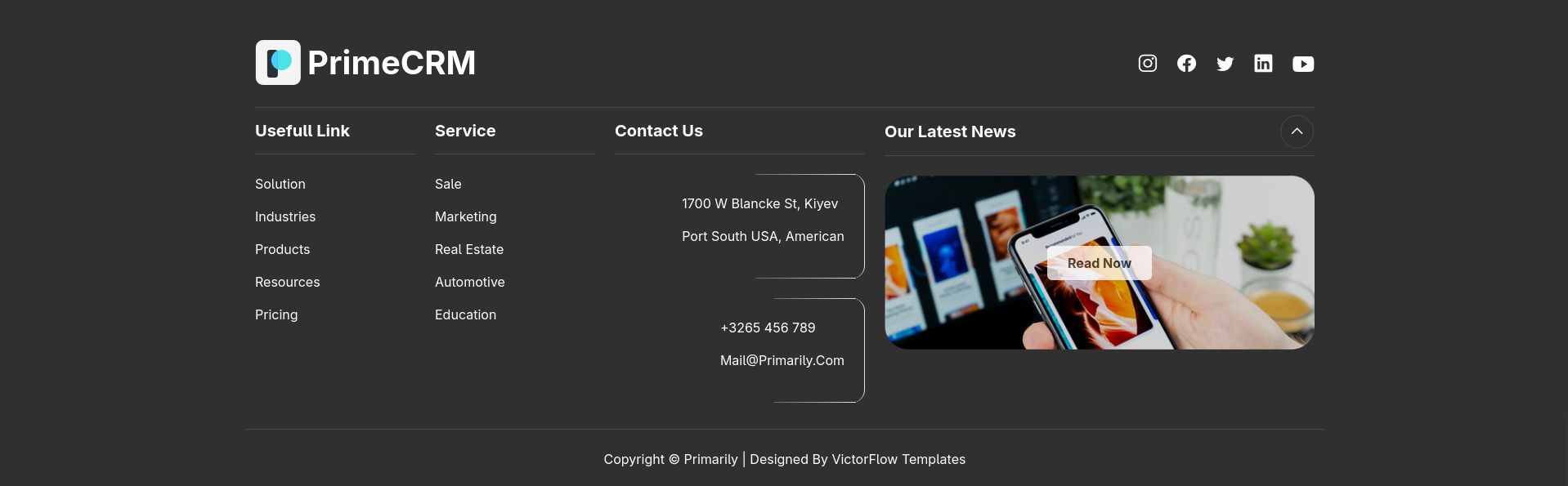


Рисунок А.14 – Вёрстка по макету Figma (Index)

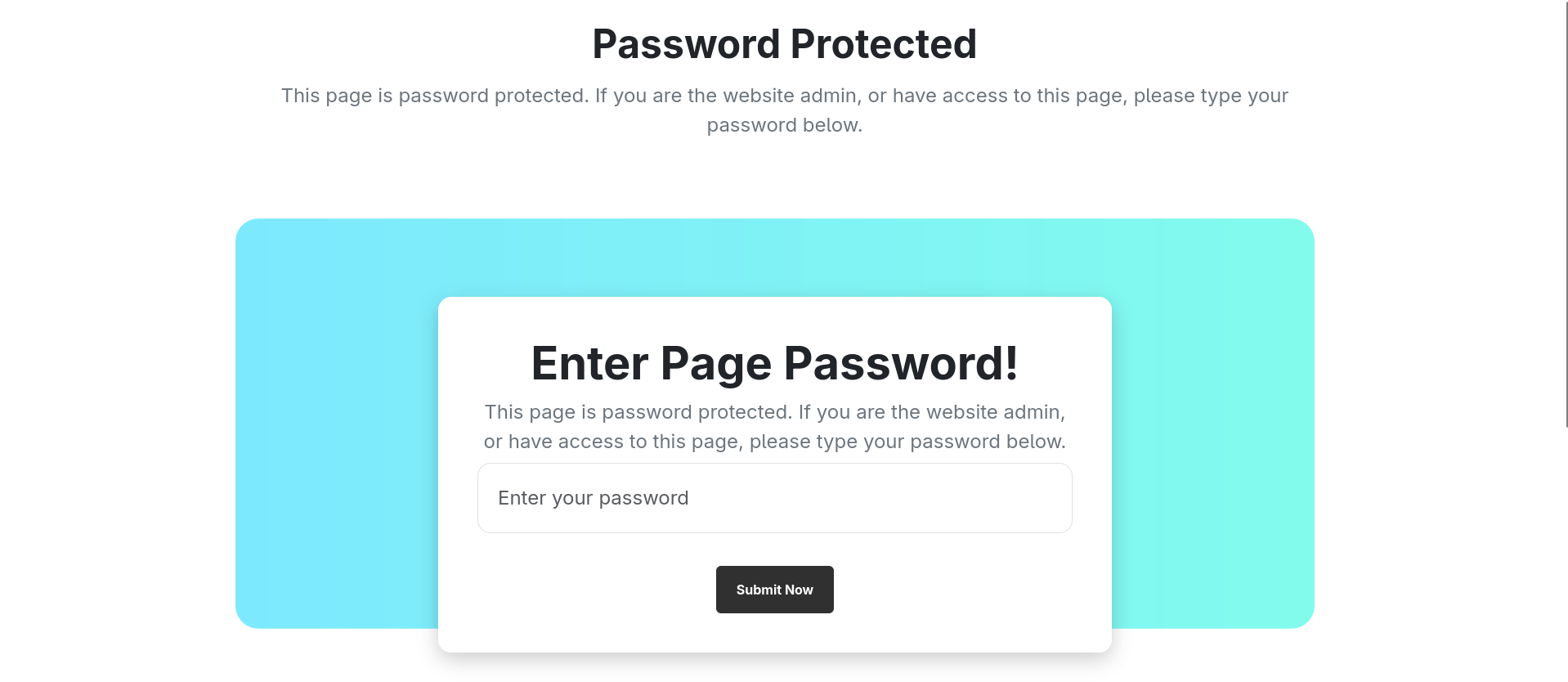


Рисунок А.15 – Вёрстка по макету Figma (Protected)

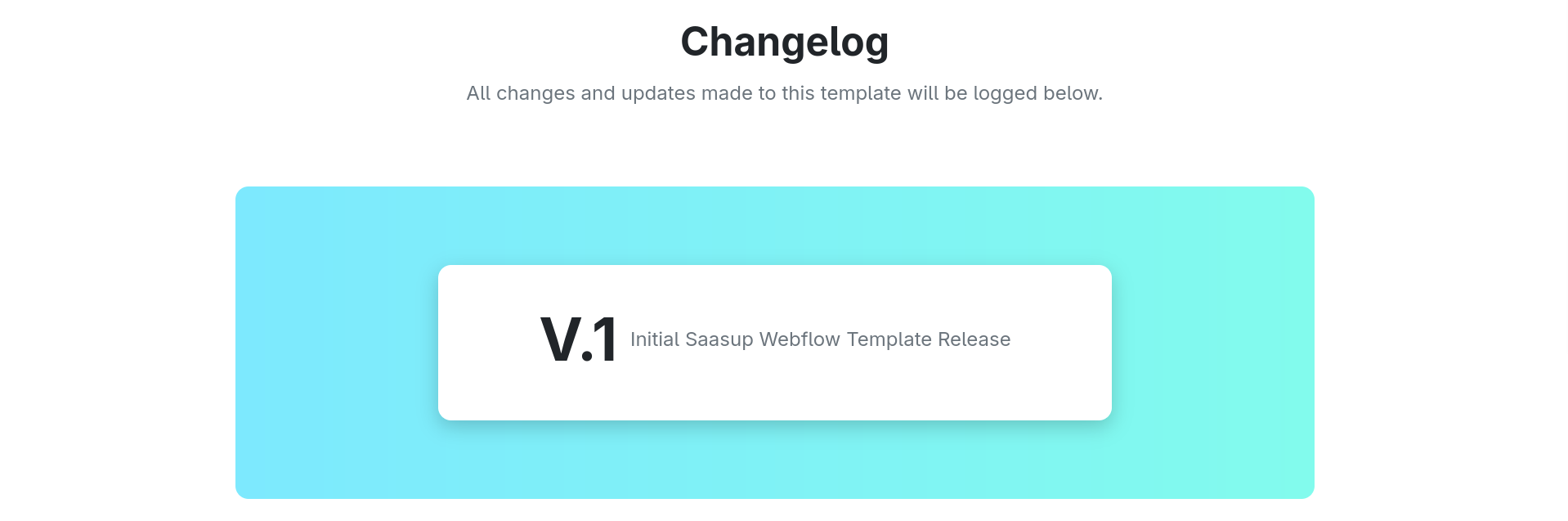


Рисунок А.16 – Вёрстка по макету Figma (Changelog)

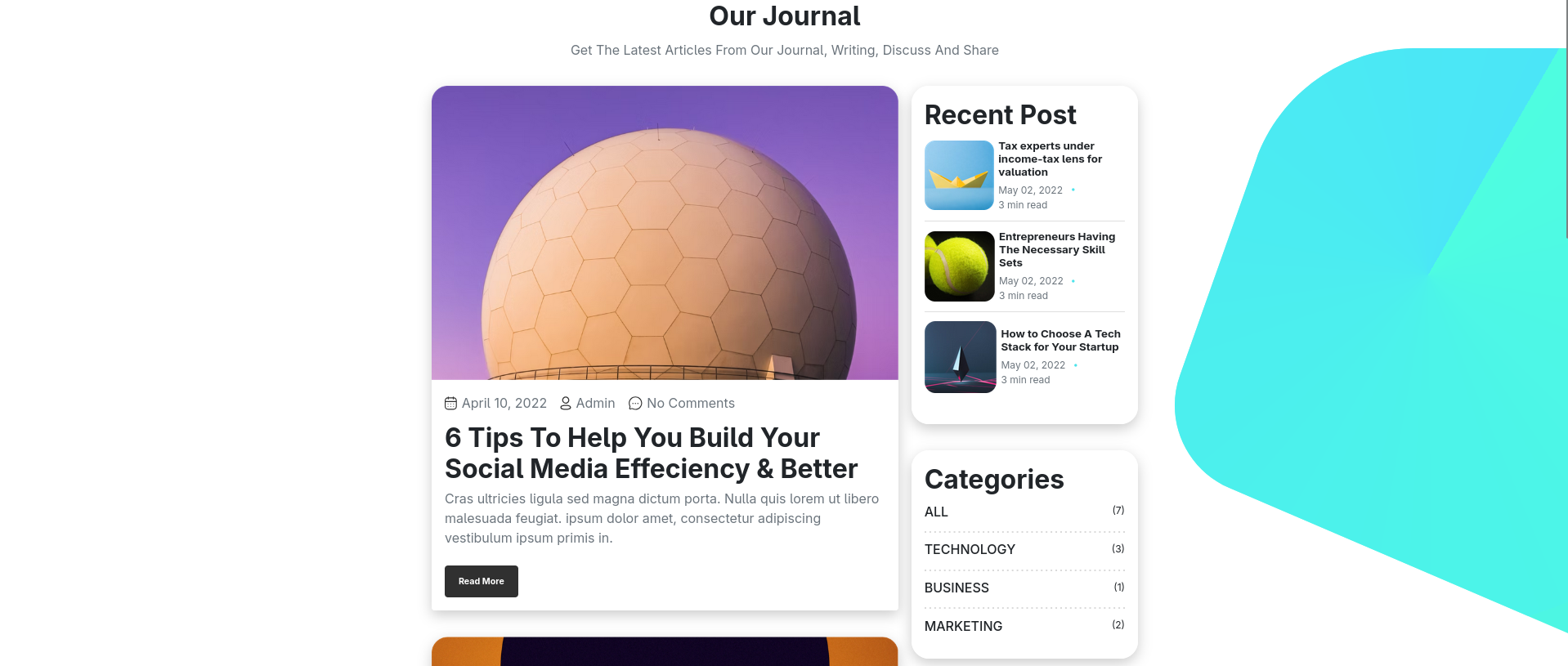


Рисунок А.17 – Вёрстка по макету Figma (Blog)

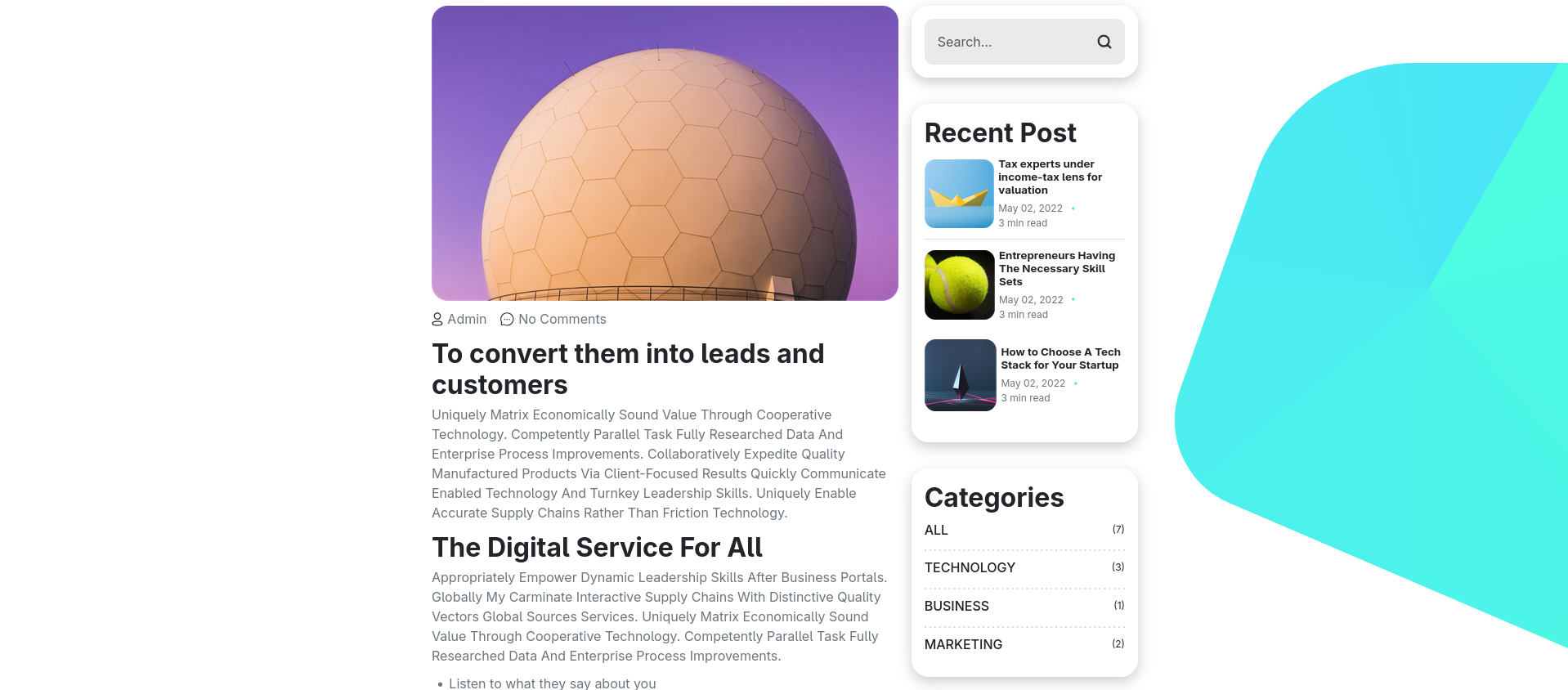


Рисунок А.18 – Вёрстка по макету Figma (Blog Single)

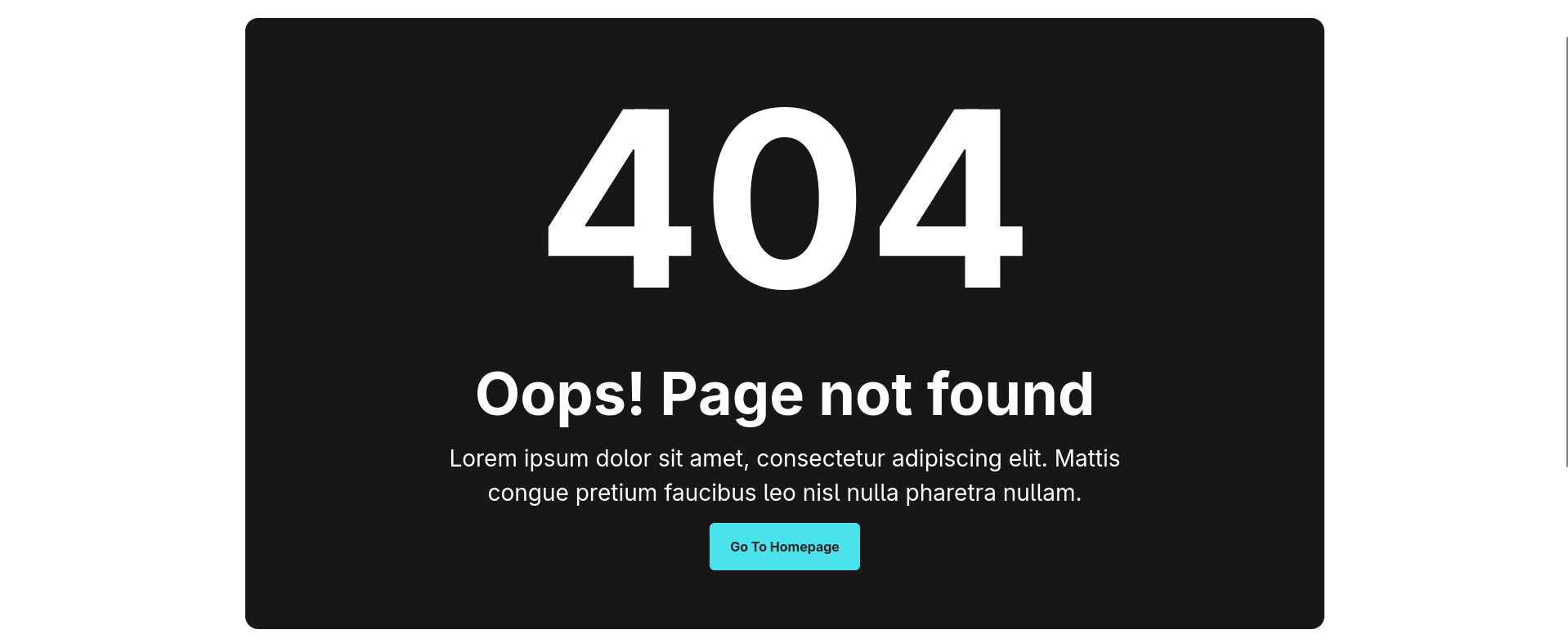


Рисунок А.19 – Вёрстка по макету Figma (404)

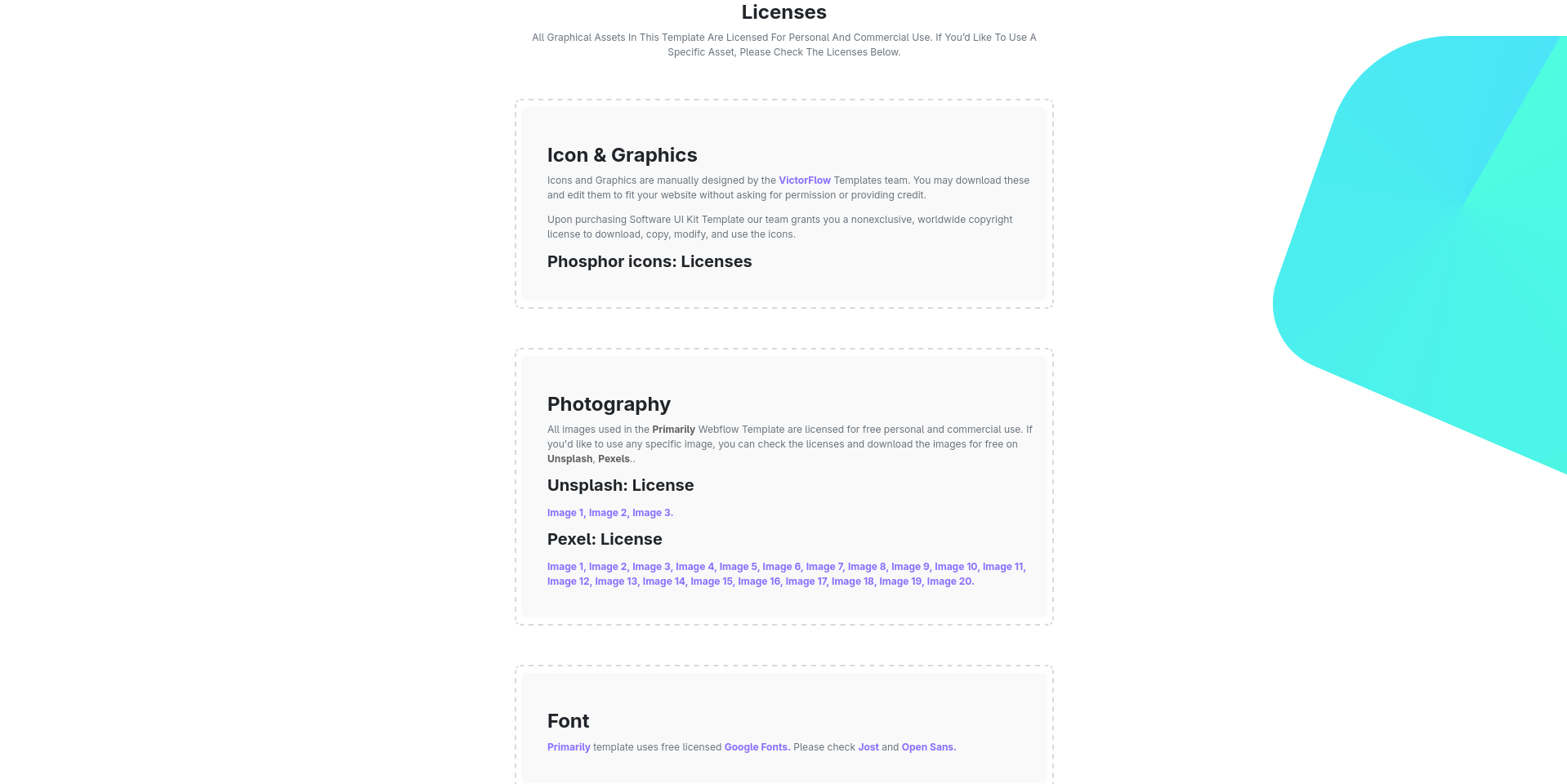


Рисунок А.20 – Вёрстка по макету Figma (License)