Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

ОТЧЕТ

По проекту
«Разработка платформы для обучения студентов с использованием модели взаимного обучения»

по дисциплине «Проектный практикум»

Заказчик: Хлебников Н.А.

Куратор: Ананьева Алена Борисовна
ученая степень, ученое звание, должность

Студенты команды ___Performance_____

Колесник Кирилл Павлович	Разработчик back-end
Сычкин Игорь Александрович	Аналитик
Сайдулин Денис Маратович	Разработчик front-end
Сахно Никита Сергеевич	Тимлид + аналитик
Ларченко Мария Андреевна	Дизайнер

Екатеринбург 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Основной целью проекта является создание P2P сервиса для студентов УрФУ на основе принципов взаимного обучения студентов по предмету программирование, который будет обладать необходимым функционалом для реализации этой идеи и будет помогать студентам учиться программированию более продуктивно.

Таким образом, основными задачами проекта являются:

1. Реализация функционала для создания тематических постов с вопросами о проблеме;
2. Реализация функционала для создания курсов от студентов по темам ;
3. Создание интуитивного дизайна сайта, знакомого типичным пользователям сайтов;
4. Реализация личных чатов между участниками для взаимной помощи по вопросам из постов;
5. Создание возможности прикрепить решение по вопросу к соответствующему посту;
6. Реализация рейтинга, который в будущем может влиять на брс и служить мотивацией для помощи друг другу через данный сервис;
7. Создание возможности комментирования кода в личном чате(режим кода).

ЦА данного продукта — студенты УрФУ, которые изучают программирование. Актуальность этого сервиса заключается в том, что ЦА постоянно работает с большим количеством информации в сфере IT, у ЦА постоянно возникают вопросы как решить ту или иную задачу по программированию.

Студентам приходится искать решения на различных форумах, читать тематические статьи на Habr и прочих площадках. Разобраться в теме лучше всего помогает наставник — более опытный человек, но у преподавателя далеко не всегда есть время и возможность помочь каждому нуждающемуся студенту. Как раз эту проблему и будет решать наш сервис. Каждый студент сможет обратиться с вопросом или просьбе о помощи, выложив пост на сайт, а более опытные студенты, которые хорошо разбираются в той или иной теме, будут помогать нуждающимся в помощи студентам. Так, любой студент сможет и помогать, и получать помощь от других студентов, это сделает обучение студентов более быстрым и продуктивным, т.к обучая других, помогающие студенты будут сами лучше разбираться в теме, а получающие помощь смогут разобраться в теме быстрее, чем если бы они делали это полностью самостоятельно. Таким образом, областью применения данного продукта является образование студентов.

По завершении проекта мы ожидаем получить продукт, который как минимум содержит в себе весь необходимый функционал для обсуждения проблем при решении задач, вопросов по темам программирования, помощи друг другу в чате, имеет понятную структуру, а также простой, но приятный дизайн. В наилучшем случае, мы ожидаем получить сайт с красивым дизайном и анимациями, а также с более сложными функциями для работы с кодом и обучения друг друга, которые добавят больше удобства пользователям. Так же в таком случае, т.е при идеальных условиях, основным достижением проекта будет запуск сайта(сервиса) на общем (не локальном) сервере и использование сайта реальными пользователями в целях, для которых и задумывался этот проект, с последующим сбором отзывов о продукте.

ПЛАН ВЫПОЛНЕНИЯ ПРОЕКТА

1.1 Роли участников в проекте

Первым делом мы распределились по ролям, чтобы разработка проекта шла одновременно по все фронтам, и не было больших застоев в том или ином направлении. Так , у нас в команде получился следующий состав:

1. Аналитик;
2. Тим-лид + аналитик;
3. Дизайнер;
4. Back-end разработчик;
5. Front-end разработчик.

Для написания кода выделили 2 разработчика, чтобы оптимизировать время разработки кода. Также тим-лид выступает в нашей команде в роли аналитика, чтобы быстрее анализировать информацию, а также сократить время для написания таких важных для разработки документов, как Use Cases, User Story и бэклог проекта.

1.2 План разработки

Общий план разработки продуктов состоит из 3 этапов:

1. Анализ требований заказчика и ЦА, поиск и изучение необходимых инструментов для создания данного проекта, наброски дизайна, бэклог;
2. Доработка набросков дизайна сайта, цвета, составление описания функционала сайта, критериев MVP, составление Use Cases, User Story,

прочих документов для создания продукта, начало разработки(написание первого кода и верстка первых страниц без дизайна), разработка архитектуры базы данных, основная разработка back части сайта;

3. Доработка дизайна, правки для критериев MVP на основе оценки возможностей разработчиков и оставшегося времени, доработка и адаптация функционала, верстка оставшихся страниц сайта, доработка кода, доработка внешнего вида сайта, исправление ошибок в работе сайта и его функционала.

Этот план разработки включает в себя на каждом этапе, за исключением первого, анализ прогресса в разработки проекта и корректировку итогового результата в соответствии с оставшимся временем и возможностями для реализации задуманных функции. В случае возникновения проблем с тем или иным пунктом, в разработке функционала выдвигалось альтернативное, упрощенное решение реализации, которое помогало сохранить суть и «полезность» проекта без значимых изменений в структуре и работе сайта. При составлении плана разработки мы основывались на Agile методологии разработки проектов.

2 Первый этап

2.1 Анализ требований к проекту

Мы начали с ЦА и общих требований к продукту. Сначала провели анализ нашей ЦА с помощью опроса и выявили, что основными потребностями и проблемами будущих пользователей являются:

1. Возможность обратиться к более опытному человеку/наставнику;
2. Возможность обучаться онлайн;
3. Часто испытывают трудности в изучении новой темы по программированию;
4. Некоторые задания при обучении непонятны и требуют пояснения от тех, кто смог найти решение;
5. Преподаватель долго отвечает на вопросы из-за недостатка времени.

Также мы проанализировали аналоги и составили таблицу сравнения нашего будущего проекта с уже существующими. Мы сравнивали основные функции типичные для такого типа сервисов, а также выяснили, что нашей уникальностью будет то, что сервис содержит все необходимой для обучения в одном месте, это можно увидеть в таблице:

	Лента с вопросами и ответами	Возможность выкладывать курсы	Рейтинг участников	Чаты	Режим работы с кодом	Звонки
Продукт команды	+	+	+	+	+	+
Brainly	+	-	+	-	-	-
Coursera	-	+	-	-	+	-
Joyteka	-	+	-	-	-	-
Stepic	-	+	-	-	+	-
Microsoft Teams	-	-	-	+	-	+
Ответы Mail.ru	+	-	+	-	-	-

Рис.1. Таблица сравнения с аналогами

2.2 Анализ архитектуры проекта

После анализа требований, мы составили план архитектуры проекта, при выборе тех или иных инструментов для реализации идей мы в основном опирались на время, за которое можно реализовать то, что мы хотим с помощью данного инструмента, и на сложность реализации с помощью этого инструмента, т. е мы подобрали такой набор инструментов, который позволил выполнить задуманное просто и с умеренными тратами по времени. В итоге у нас получился следующий набор инструментов для разработки:

1. Figma – для разработки дизайна;
2. Python — т.к гибкий язык и удобно работать на нем с Django;
3. Django Rest Framework (DRF) - фреймворк для разработки API на Django;
4. SQLite - встроенная СУБД в Django, удобна при работе с GITом;

5. JavaScript – для интерактива сайта;
6. HTML5, CSS3 – для верстки страниц и настройки стилей страниц сайта;
7. DB Designer – для проектировки архитектуры БД;
8. React – для более быстрой разработки и оптимизации работы сайта, быстрой загрузки.

Так же для всего прочего мы использовали такие инструменты как:

1. Интернет — поиск информации и идей;
2. Google Forms – для создание опросов и получения результатов;
3. Текстовые редакторы — для создание отчетов, критериев к MVP и т. п.;
4. Git Hub – для хранения и безопасного изменения кода;
5. Yougile — для отслеживания задачи и дедлайнов;
6. Google Docs – для описание проекта и прочих документов и отчетов;
7. Google Drive – для хранения всех артефактов.

2.3 Работа дизайнера

На этом этапе дизайнер, проанализировал аналоги с точки зрения дизайна, продумал понятный интерфейс для типичного пользователя сайтов и составил эскизы дизайна сайта:

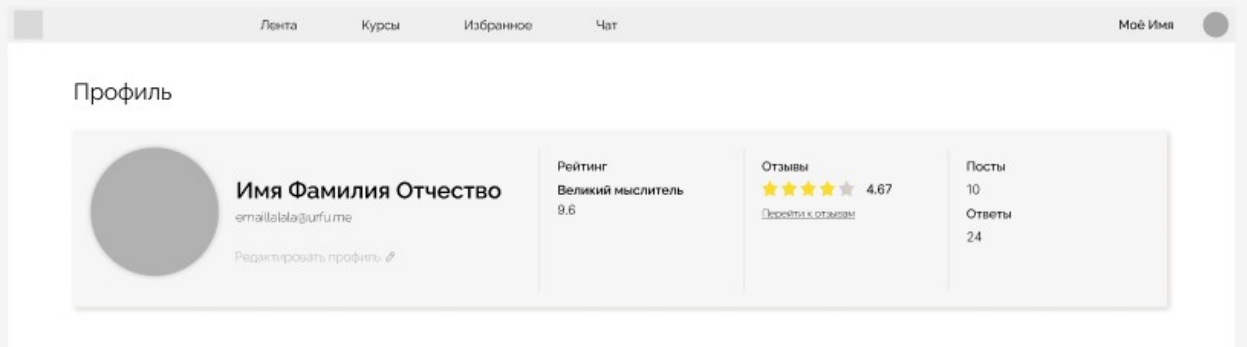


Рис.2. Пример эскиза 1

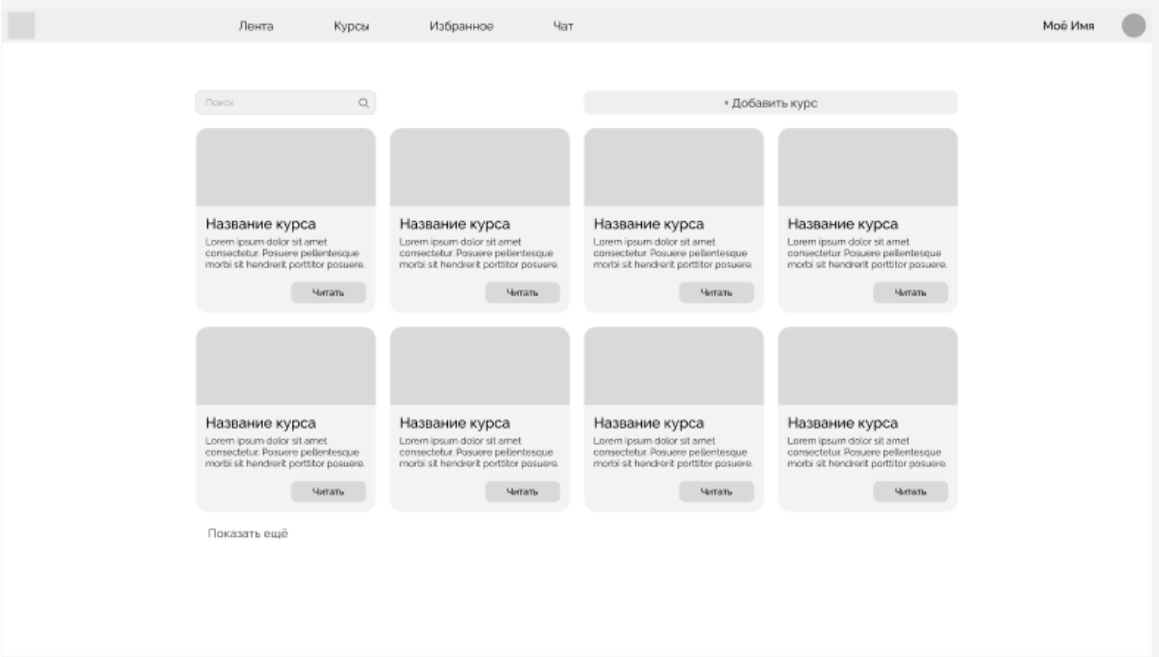


Рис.3. Пример эскиза 2

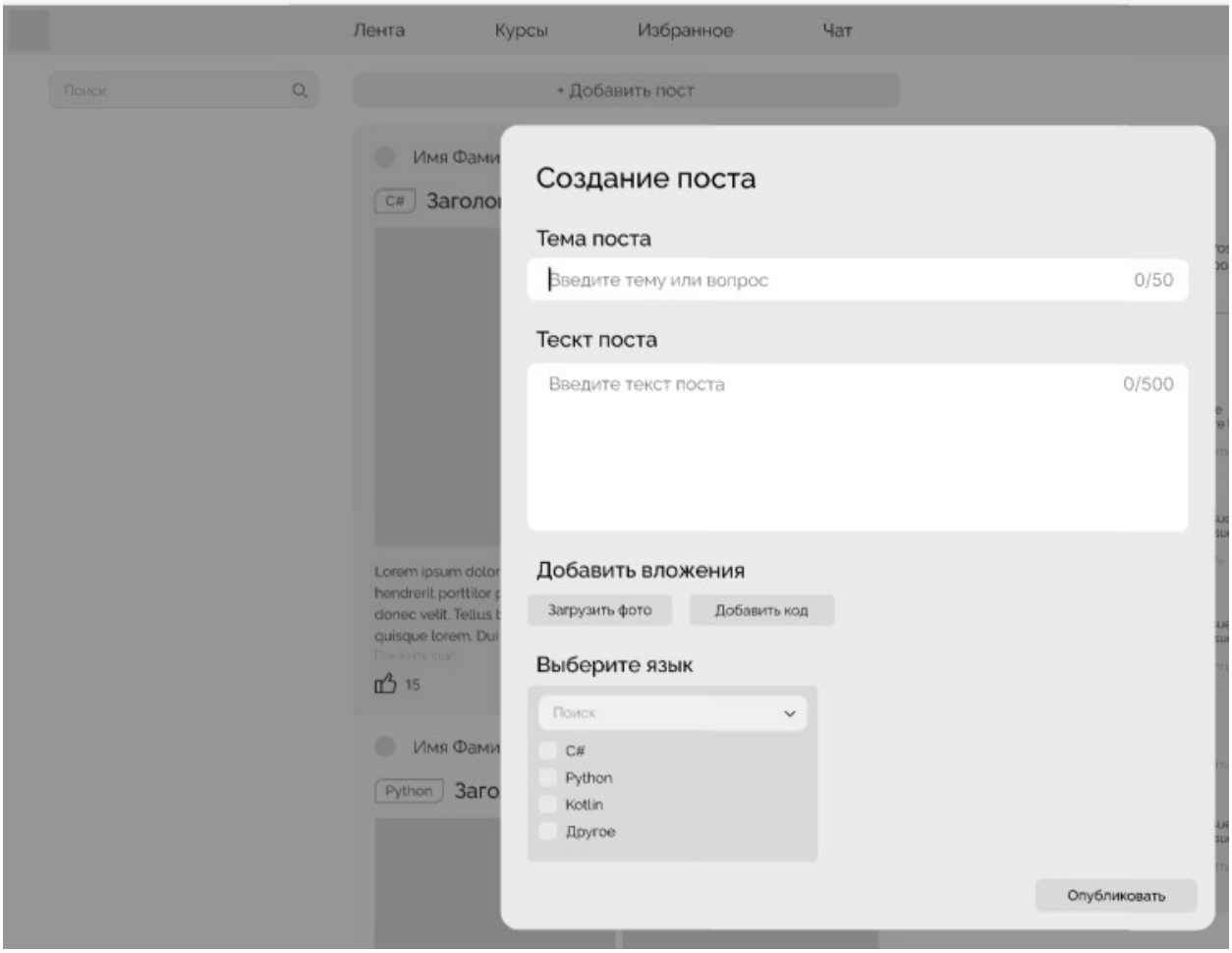


Рис.4. Пример эскиза 3

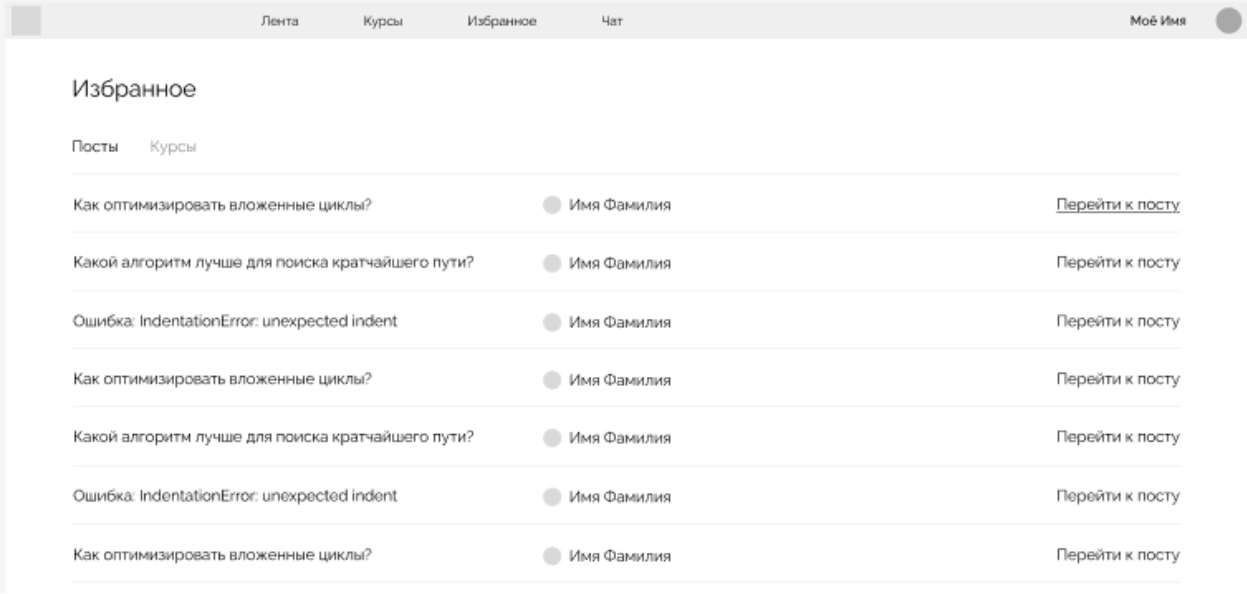
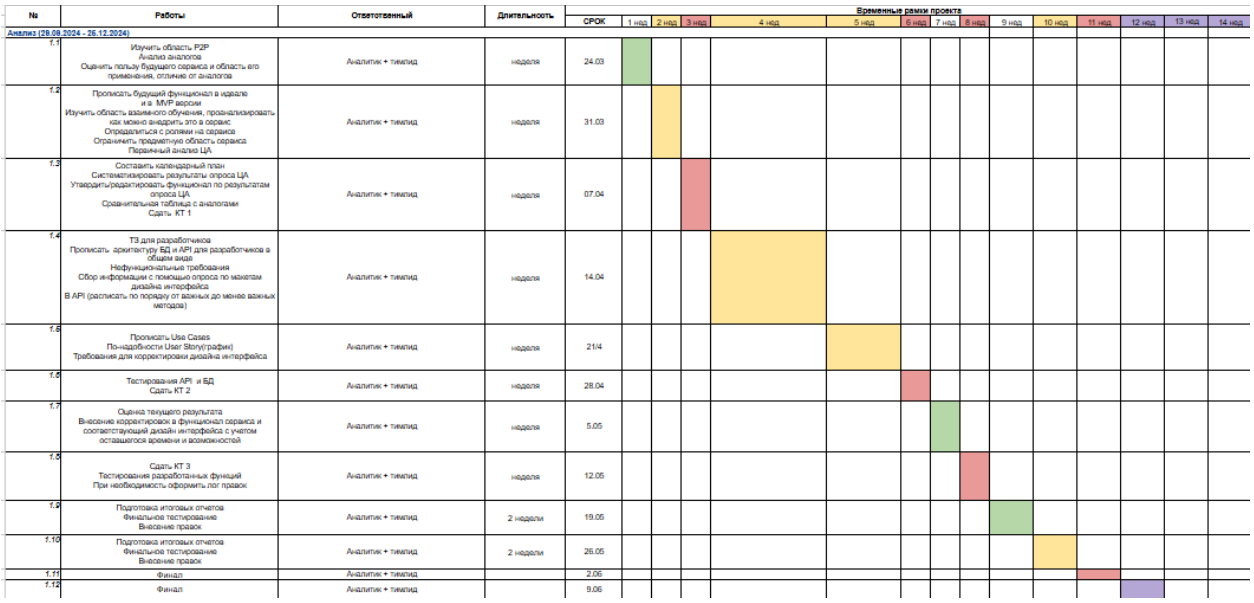


Рис.5. Пример эскиза 4

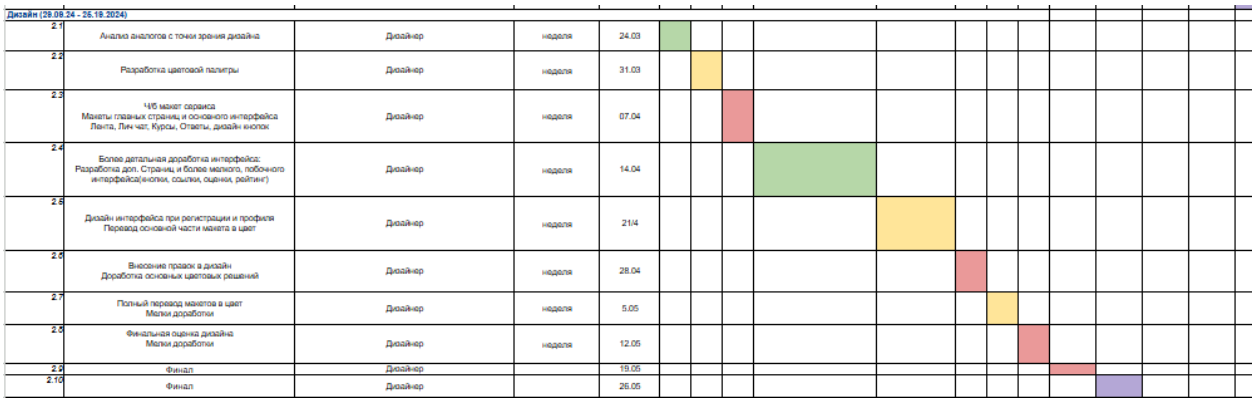
2.4 Работа аналитиков

На этом этапе аналитики проанализировали ЦА, опрос ЦА, составили сравнительную таблицу (Рис.1.), сделали набросок функционала сайта и расписали бэклог проекта:



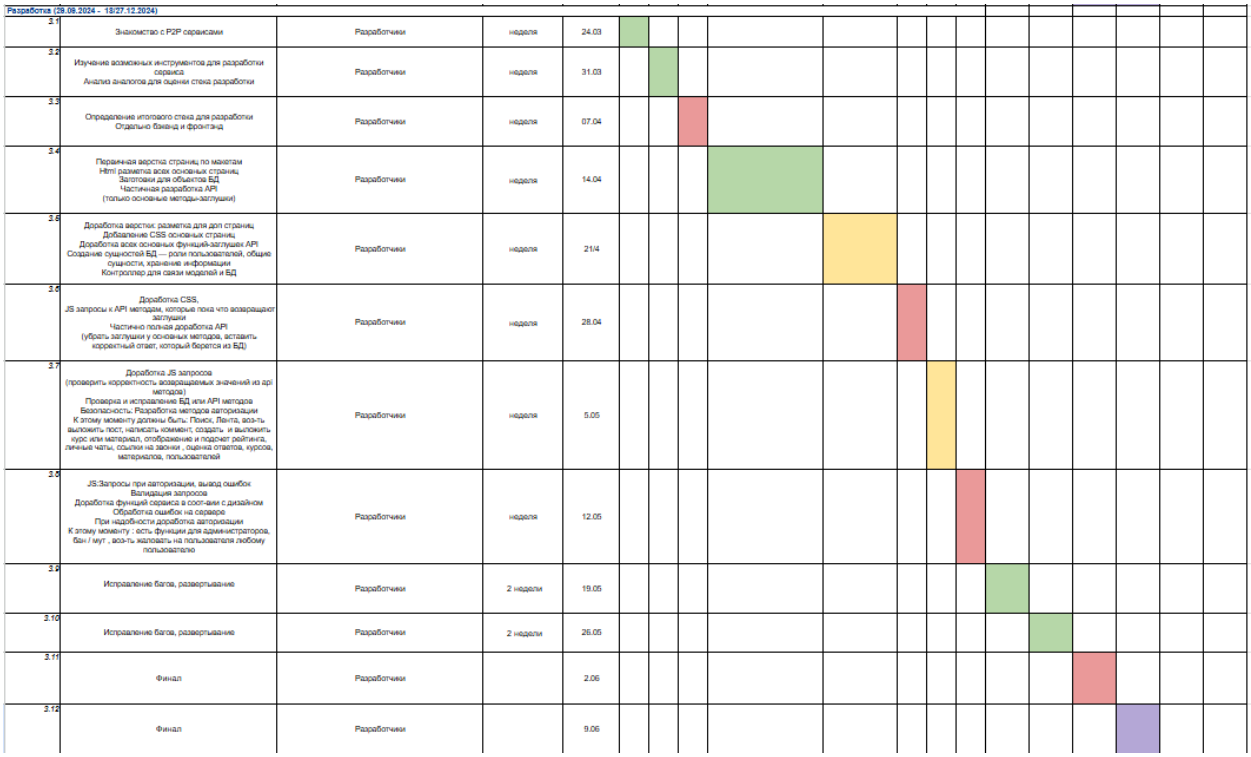
№	Работы	Ответственный	Длительность	Временные рамки проекта														
				СРОК	1 нед	2 нед	3 нед	4 нед	5 нед	6 нед	7 нед	8 нед	9 нед	10 нед	11 нед	12 нед	13 нед	14 нед
Анализ (28.09.2024 - 26.12.2024)																		
1.1	Изучить область P2P Анализ аналогов Оценить пользу будущего сервиса и область его применения, отличие от аналогов	Аналитик + тимлид	неделя	24.03														
1.2	Прописать будущий функционал в идеале и в MVP версии Изучить область машинного обучения, проанализировать как можно внедрить это в сервис Определиться с ролями на сервисе Ограничить предметную область сервиса Первичный анализ ЦА	Аналитик + тимлид	неделя	31.03														
1.3	Составить календарный план Систематизировать результаты опроса ЦА Утвердить/редактировать функционал по результатам опроса ЦА Сравнительная таблица с аналогами Сдать КТ 1	Аналитик + тимлид	неделя	07.04														
1.4	ТЗ для разработчиков Прописать архитектуру БД и API для разработчиков в общем виде Нефункциональные требования Сбор информации с помощью опроса по макетам дизайна интерфейса В API (расписать по порядку от важных до менее важных методов)	Аналитик + тимлид	неделя	14.04														
1.5	Прописать Use Cases По-надобности User Story(график) Требования для корректировки дизайна интерфейса	Аналитик + тимлид	неделя	21/4														
1.6	Тестирования API и БД Сдать КТ 2	Аналитик + тимлид	неделя	28.04														
1.7	Оценка текущего результата Внесение корректировок в функционал сервиса и соответствующий дизайн интерфейса с учетом оставшегося времени и возможностей	Аналитик + тимлид	неделя	5.05														
1.8	Сдать КТ 3 Тестирования разработаных функций При необходимости оформить лог правок	Аналитик + тимлид	неделя	12.05														
1.9	Подготовка итоговых отчетов Финальное тестирование Внесение правок	Аналитик + тимлид	2 недели	19.05														
1.10	Подготовка итоговых отчетов Финальное тестирование Внесение правок	Аналитик + тимлид	2 недели	26.05														
1.11	Финал	Аналитик + тимлид		2.06														
1.12	Финал	Аналитик + тимлид		9.06														

Рис.6. Бэклог для анализа



Дизайн (29.09.24 - 26.10.2024)																		
2.1	Анализ аналогов с точки зрения дизайна	Дизайнер	неделя	24.03														
2.2	Разработка цветовой палитры	Дизайнер	неделя	31.03														
2.3	Ч/б макет сервиса Макеты главных страниц и основного интерфейса Лента, Лич чат, Курсы, Ответы, дизайн кнопок	Дизайнер	неделя	07.04														
2.4	Более детальная доработка интерфейса: Разработка доп. Страниц и более мелкого, побочного интерфейса(кнопки, ссылки, оценки, рейтинг)	Дизайнер	неделя	14.04														
2.5	Дизайн интерфейса при регистрации и профиля Перевод основной части макета в цвет	Дизайнер	неделя	21/4														
2.6	Внесение правок в дизайн Доработка основных цветовых решений	Дизайнер	неделя	28.04														
2.7	Полный перевод макетов в цвет Мелки доработки	Дизайнер	неделя	5.05														
2.8	Финальная оценка дизайна Мелки доработки	Дизайнер	неделя	12.05														
2.9	Финал	Дизайнер		19.05														
2.10	Финал	Дизайнер		26.05														

Рис.7. Бэклог для дизайна



Разработка (29.09.2024 - 18/27.12.2024)					
3.1	Знакомство с P2P сервисами	Разработчики	неделя	24.03	
3.2	Изучение возможных инструментов для разработки сервиса Анализ аналогов для оценки стека разработки	Разработчики	неделя	31.03	
3.3	Определение итогового стека для разработки Отдельно бэкенд и фронтенд	Разработчики	неделя	07.04	
3.4	Первичная верстка страниц по макетам Html разметка всех основных страниц Заготовки для объектов БД Частичная разработка API (только основные методы-заглушки)	Разработчики	неделя	14.04	
3.5	Доработка верстки: разметка для доп страниц Добавление CSS основных страниц Доработка всех основных функций-заглушек API Создание сущностей БД — роли пользователей, общие сущности, хранение информации Контроллер для связи моделей и БД	Разработчики	неделя	21/4	
3.6	Доработка CSS, JS запросы к API методам, которые пока что возвращают заглушки Частично полная доработка API (убрать заглушки у основных методов, вставить корректный ответ, который берется из БД)	Разработчики	неделя	28.04	
3.7	Доработка JS запросов (проверить корректность возвращаемых значений из api методов) Проверка и исправление БД или API методов Безопасность. Разработка методов авторизации К этому моменту должны быть: Поиск, Лента, воз-ть выложить пост, написать коммент, создать и выложить курс или материал, отображение и подсчет рейтинга, личные чаты, ссылки на звонки , оценка ответов, курсов, материалов, пользователей	Разработчики	неделя	5.05	
3.8	JS:Запросы при авторизации, вывод ошибок Валидация запросов Доработка функций сервиса в соот-вии с дизайном Обработка ошибок на сервере При надобности доработка авторизации К этому моменту : есть функции для администраторов, бан / мут , воз-ть жаловать на пользователя любому пользователю	Разработчики	неделя	12.05	
3.9	Исправление багов, развертывание	Разработчики	2 недели	19.05	
3.10	Исправление багов, развертывание	Разработчики	2 недели	26.05	
3.11	Финал	Разработчики		2.06	
3.12	Финал	Разработчики		9.06	

Рис.8. Бэклог для разработки

Мы также проанализировали результаты опроса нашей ЦА о предполагаемом образе нашего продукта и трудностях ЦА. По результатам опроса мы выявили, что ЦА нуждается в сервисе, который будет:

1. Давать возможность учиться онлайн;
2. Давать возможность каждому и помогать, и спрашивать, делиться материалам;
3. Давать возможность получать ответ быстро

На вопрос: «ищите ли вы ответы на вопросы на специальных платформах по программированию» около 79% людей дало положительный ответ, а также на вопрос о наличии трудностей в изучении программирования 66%

подтвердили, что проблемы присутствуют, поэтому курсы от самих студентов будет полезной функцией нашего сервиса для ЦА.

За онлайн обучение проголосовало около 67% всех опрашиваемых, поэтому решение оформить сервис в виде сайта — хорошее решение, т.к можно использовать наш сервис с любого устройства и в любом месте, где есть интернет, без необходимости устанавливать отдельное приложение.

Многие студенты по результатам опроса обращаются друг к дургу за помощью, поэтому сделать ленту с постами-вопросами с возможностью решения вопросов из поста в личном чате - будет хорошим решением, а также вид ленты является привычным видом отображения информации для студентов.

Дополнительная мотивация в виде наличия рейтинга, который, возможно, в будущем будет влиять на БРС поможет получать ответ на свой вопрос довольно быстро, т.к активные студенты, которые помогают другим, будут получать за это баллы рейтинга.

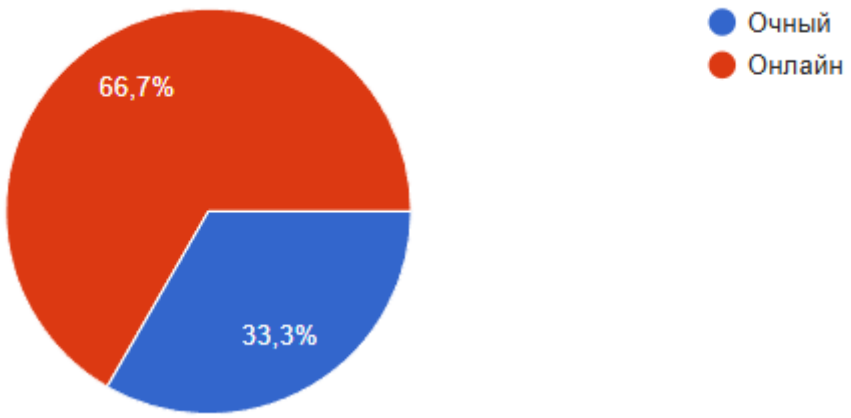


Рис.9. Пример результатов опроса о формате обучения

2.5 Работа разработчиков

На этом этапе разработчики проанализировали аналоги с точки зрения архитектуры, определили стек разработки для back-end и front-end разработки и ознакомились с инструментами, которые в дальнейшем использовались при создании проекта.

Таким образом, на 1 этапе команда выполнила все пункты, которые были обозначены в плане выполнения проекта:

1. Проанализированы требования и ЦА;
2. Составлены эскизы дизайна;
3. Определен стек и архитектура разработки.

3 Второй этап

3.1 Корректировки

На этом этапе был проведен анализ проделанной работы, в архитектуру проекта были внесены изменения из-за сложности реализации и адаптации проекта под новые условия: сначала потребовалось изменить функционал, чтобы он больше соответствовал требованиям заказчика и самой идеи P2P. Далее вносились изменения в архитектуру базы данных, упрощалась реализация функционала сервиса в соответствии с новыми требованиями и оставшимся временем на разработку, корректировался план работ. Также по результатам опроса ЦА о дизайне(эскизах) сайта были внесены некоторые корректировки и в дизайн.

3.2 Работа дизайнера.

На этом этапе дизайнер разработал дизайн всех основных страниц сайта, составил цветовую палитру сайта. Внес корректировки в детали дизайна после оценки ЦА дизайна в соответствующем опросе.

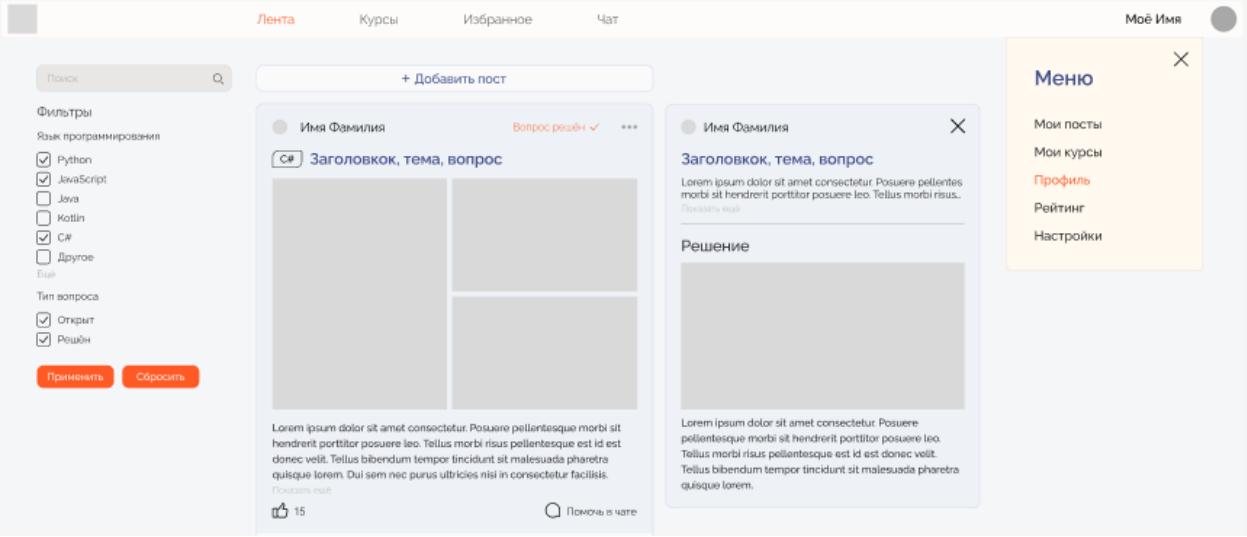


Рис.10. Дизайн ленты

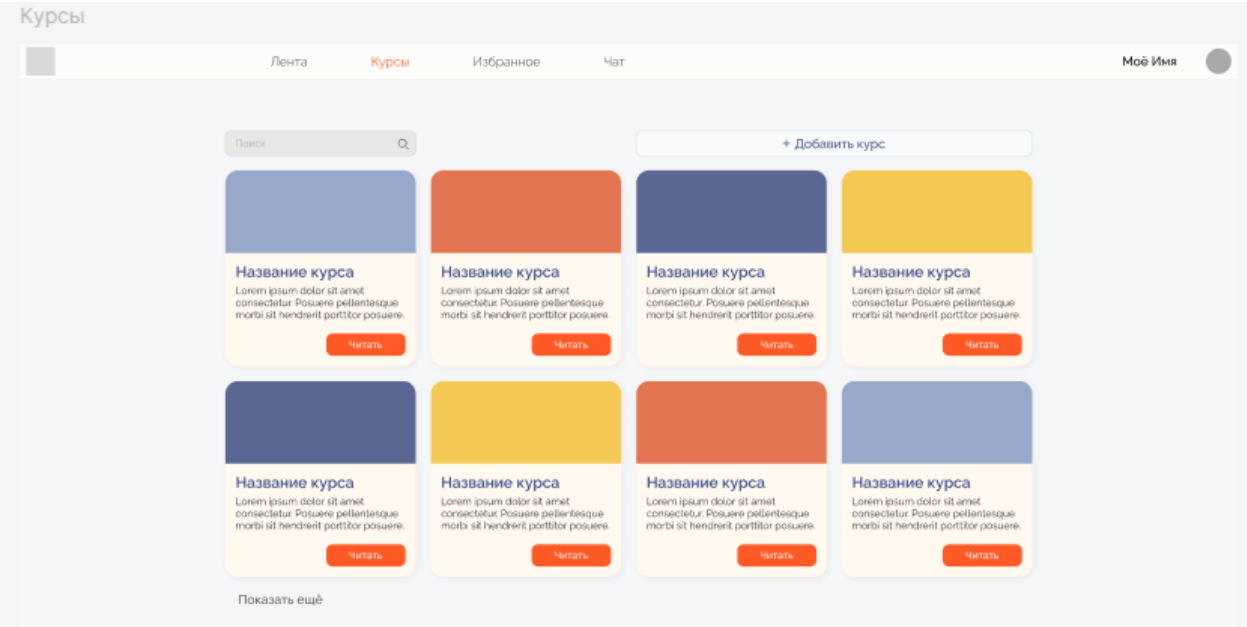


Рис.11. Дизайн страницы «Курсы»

Рис.12. Палитра сайта

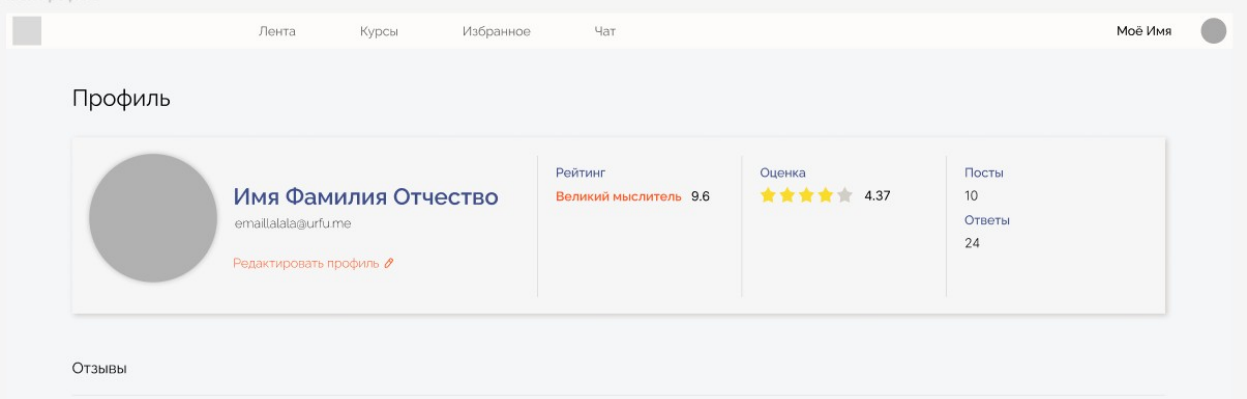


Рис.13. Дизайн профиля

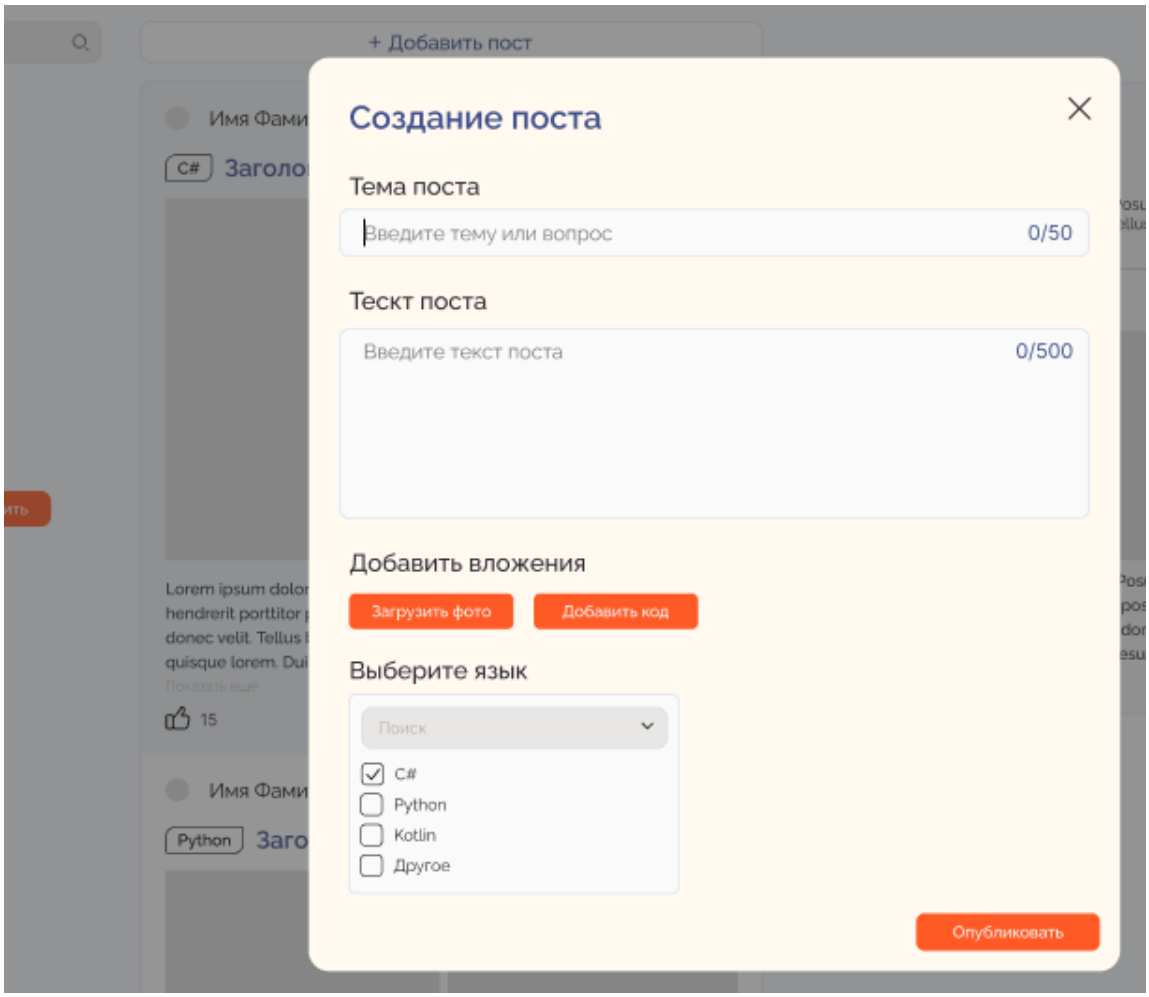


Рис.14. Дизайн создания поста

3.3 Работа Аналитиков

На этом этапе аналитики модифицировали функционал приложения, определили новые критерии MVP проекта, доработали Use Cases, архитектуру БД, составили User Story, провели опрос по дизайну сайта.

Пользователь пришел, чтобы получить информацию:

Первым делом в ленте я ищу по поиску постов(ввожу название темы и язык программирования), есть ли уже пост-вопрос на интересующую меня тему, если да, то смотрю решение моей проблемы, сам пост тогда отмечен как “решенный”, а решение прикреплено к посту
Если решение помогло/вопрос о том,что мне и надо было ставлю лайк

Иначе

Я в поиске курсов я ищу курс на мою тему, если курс есть - читаю, если помог ставлю лайк

Рис.15. Пример User Story

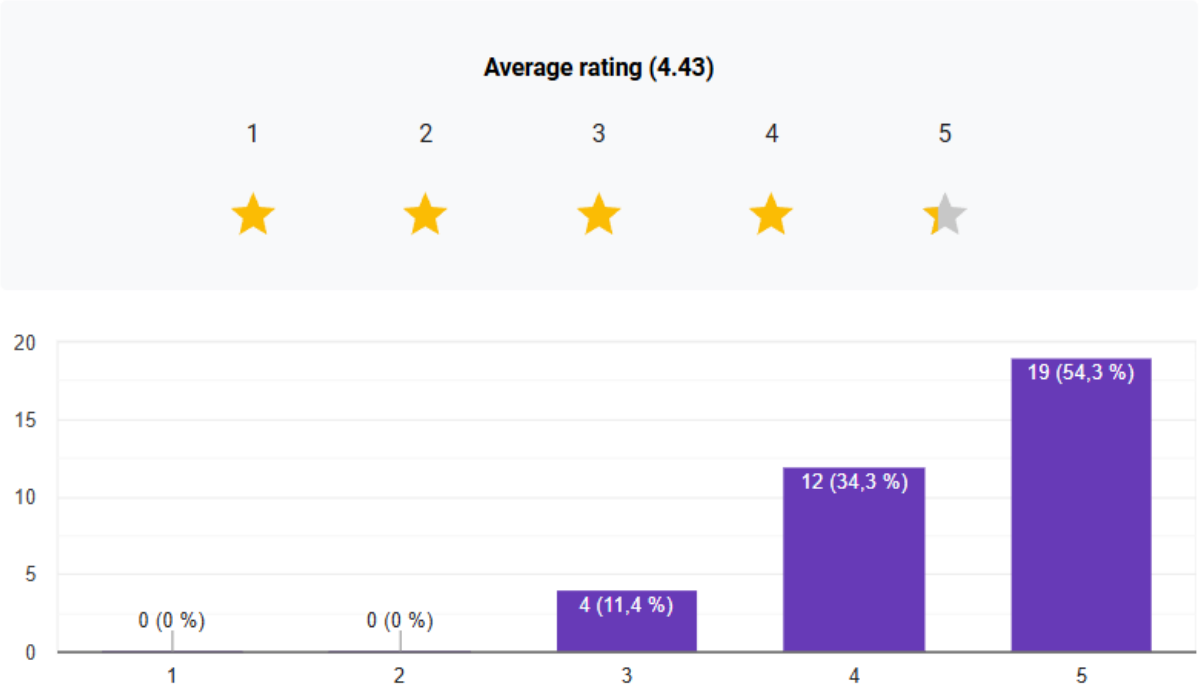


Рис.16. Пример опроса по дизайну: «Насколько приятна палитра цветов»

Результаты опроса мы проанализировали и пришли к выводу, что :

По вопросу об удобстве и полноте дизайна главной страницы(Ленты) мы получили среднюю оценку 4.46 - это значит, что дизайнерское решение оправдалось и будущую ЦА устраивает интерфейс этой страницы нашего сайта. Окно создания поста также имеет высокую оценку и не требует изменений. Также страницы профиля, курсов и избранного имеют хорошие оценки, значит интерфейс этих страниц устраивается ЦА.

Вывод: большинству понравился интерфейс и дизайн страниц нашего сайта, средняя оценка по всем критериям: 4.5, поэтому каких-либо крупных изменений в дизайне не требуется.

Помимо этого критерии к MVP функционалу нашего проекта получились следующие:

1. Главная страница с постами — Лента;

2. Добавление постов(название, тема), отметка решено/не решено;
3. Фильтр поиска постов, установка языка программирования при создании поста, добавление кода;
4. Возможность выложить свой курс в pdf формате, добавить название и описание;
5. Возможность читать выложенный курсы и искать их по названию;
6. Авторизация и регистрация;
7. Личный чат между пользователями;
8. Лайки к постам и возможность перейти в личный чат прямо из поста;
9. Режим кода (как минимум нумерация строк кода, чтоб можно было при ответе на сообщение указать к каким строчка относится комментарий);
10. Избранное — возможность сохранять понравившиеся посты и курсы;
11. Вкладка «Мои курсы» и «Мои посты»;
12. Страница профиля и отображение рейтинга на ней.

3.4 Работа back-end разработчика

На этом этапе back-end разработчик выполнил основную часть работы: создал модели основных сущностей для базы данных, поработал над API, создал шаблоны для работы с динамическими данными, создал представления, написал код для маршрутизации запросов.

```python
from django.db import models
from django.contrib.auth.models import AbstractUser
from django.utils import timezone
#models.py
class User(AbstractUser):
    profile_img_url = models.TextField(default='')
    role = models.TextField(default='user')
    total_questions = models.IntegerField(default=0)
    total_answers = models.IntegerField(default=0)
    is_blocked = models.BooleanField(default=False)
    post_likes_cnt = models.IntegerField(default=0)
    comment_likes_cnt = models.IntegerField(default=0)
    course_likes_cnt = models.IntegerField(default=0)
    chat_help_likes_cnt = models.IntegerField(default=0)

    def __str__(self):
        return self.username

```

Рис.21. Пример создание моделей сущности User - пользователь

```
<!DOCTYPE html>
<html>
<head>
    <title>Profile</title>
</head>
<body>
    <h1>{{ profile_user.username }}'s Profile</h1>
    <p>Email: {{ profile_user.email }}</p>
    <p>Role: {{ profile_user.role }}</p>
    {% if profile_user == user %}
        <p><a href="{% url 'core:edit_profile' %}">Edit Profile</a></p>
    {% endif %}
    <h2>Posts</h2>
    <ul>
    {% for post in posts %}
        <li><a href="{% url 'core:post_detail' post.id %}">{{ post.title }}</a></li>
    {% empty %}
        <li>No posts</li>
    {% endfor %}
    </ul>
    <p><a href="{% url 'core:index' %}">Home</a></p>
</body>
</html>
```

Рис.17. Создание шаблона структуры страницы для работы с динамическими данными

```python
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from . import views

router = DefaultRouter()
router.register(r'posts', views.PostViewSet, basename='post')
router.register(r'courses', views.CourseViewSet, basename='course')
router.register(r'chats', views.ChatViewSet, basename='chat')
router.register(r'messages', views.MessageViewSet, basename='message')
router.register(r'reports', views.ReportViewSet, basename='report')
router.register(r'codes', views.CodeViewSet, basename='code')

urlpatterns = [
    path('', include(router.urls)),
    path('register/', views.register, name='register'),
    path('login/', views.login_view, name='login'),
    path('logout/', views.logout_view, name='logout'),
    path('profile/<int:user_id>/', views.profile_view, name='profile'),
    path('profile/edit/', views.edit_profile, name='edit_profile'),
    path('posts/<int:post_id>/mark-resolved/', views.mark_post_resolved, name='mark_post_resolved'),
    path('bookmarks/', views.bookmark_list, name='bookmark_list'),
    path('bookmarks/add/<int:post_id>/', views.add_bookmark, name='add_bookmark'),
    path('bookmarks/remove/<int:post_id>/', views.remove_bookmark, name='remove_bookmark'),
```

Рис.18. Пример urls файла маршрутизации запросов

```python
def test_login(self):
    data = {'email': 'user1@example.com', 'password': 'test123'}
    response = self.client.post(reverse('login'), data, format='json')
    self.assertEqual(response.status_code, status.HTTP_200_OK)
    self.assertEqual(response.data['username'], 'user1')
```

Рис.19. Пример тестов

```python
@login_required
def create_post(request):
    """Создание нового поста"""
    if request.method == 'POST':
        # Логика создания поста
        return redirect('core:post_list')
    return render(request, 'core/create_post.html')

def post_detail(request, post_id):
    """Детальный просмотр поста"""
    post = get_object_or_404(Post, id=post_id)
    comments = post.comments.all().order_by('-created_at')
    return render(request, 'core/post_detail.html', {'post': post, 'comments': comments})
```

Рис.20. Пример представлений

3.5 Работа front-end разработчика

На этом этапе front-end разработчик осуществил базовую верстку страниц и их стилей.

```css
body {
  background-color: #f5f5f5;
}

.app {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}

/* Шапка */
.header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 1rem 2rem;
  background-color: #fff;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
  position: sticky;
  top: 0;
  z-index: 100;
}
```

Рис.21. Пример настройки стилей страницы

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
```

Рис.22. Пример установки разметки страницы

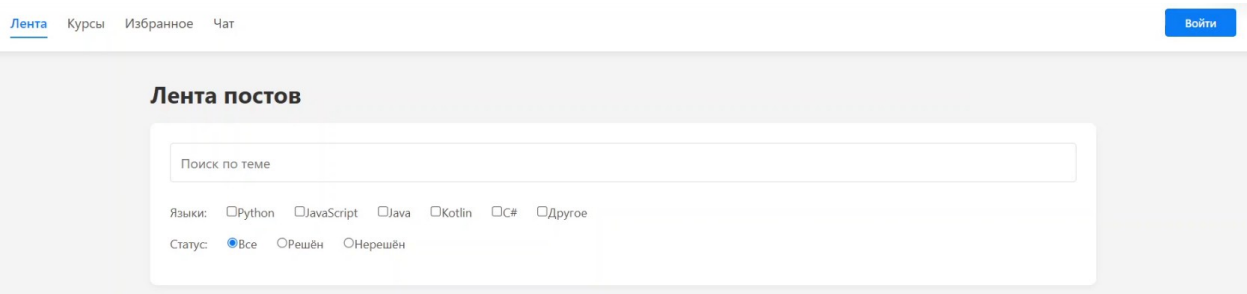


Рис.23. Пример результата верстки

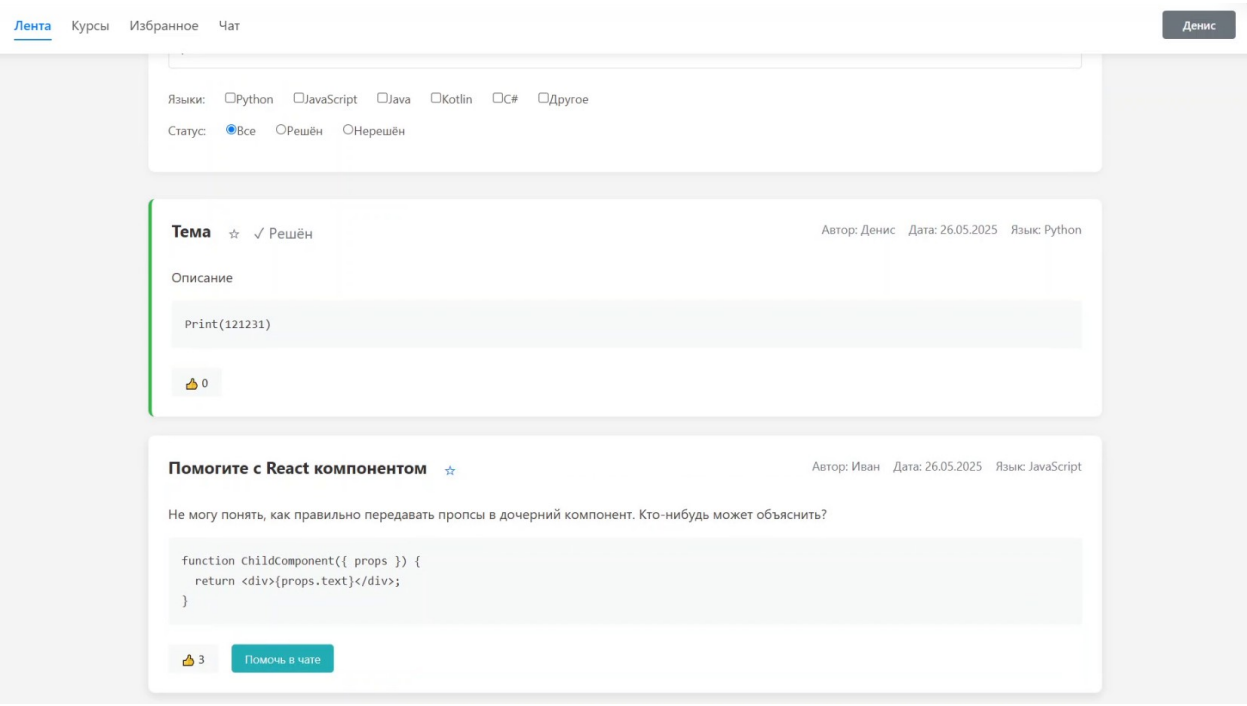


Рис.24. Пример результата верстки

Таким образом, за 2 этап команда успела выполнить основные задачи, поставленные в плане разработки продукта:

1. Осуществлена доработка дизайна, есть палитра сайта;
2. Разработан основной функционал сайта, верстка страниц;
3. Подкорректированы критерии MVP, Use Cases, User Story, разработка архитектуры БД, опрос по дизайну;

4. Разработаны основные составляющие back-end части сайта.

4 Третий этап

4.1 Корректировки

В результате анализа оставшегося времени и возможностей для реализации функций были внесены корректировки в функционал :

1. Режим кода будет иметь упрощенный вид;
2. Возможность создания курса прямо в сервисе удалена;
3. Отдельный профиль админа и панель админа также удалена.

Эти и некоторые второстепенные функции, которые задумывались изначально, мы не сможем реализовать из-за недостатка времени и сложности разработки такого количества сложных функций, поэтому некоторые из них мы упростили или убрали вовсе. Несмотря на это, в back части сайта находятся наработки для большинства из отмененных функций, например, код для реализации профиля админа. После упрощения функционала свое назначение наш сервис выполняет, все необходимые функции для взаимного обучения присутствуют, а основная идея и «полезность» проекта сохранена.

4.2 Работа дизайнера

Вся основная работа дизайнера на этом этапе заканчивается, основные страницы готовы. Дизайнер корректирует, упрощает дизайн в тех местах, где возникли проблемы с переносом дизайна. Такое решение помогает сохранить проект и его работоспособность малыми жертвами.

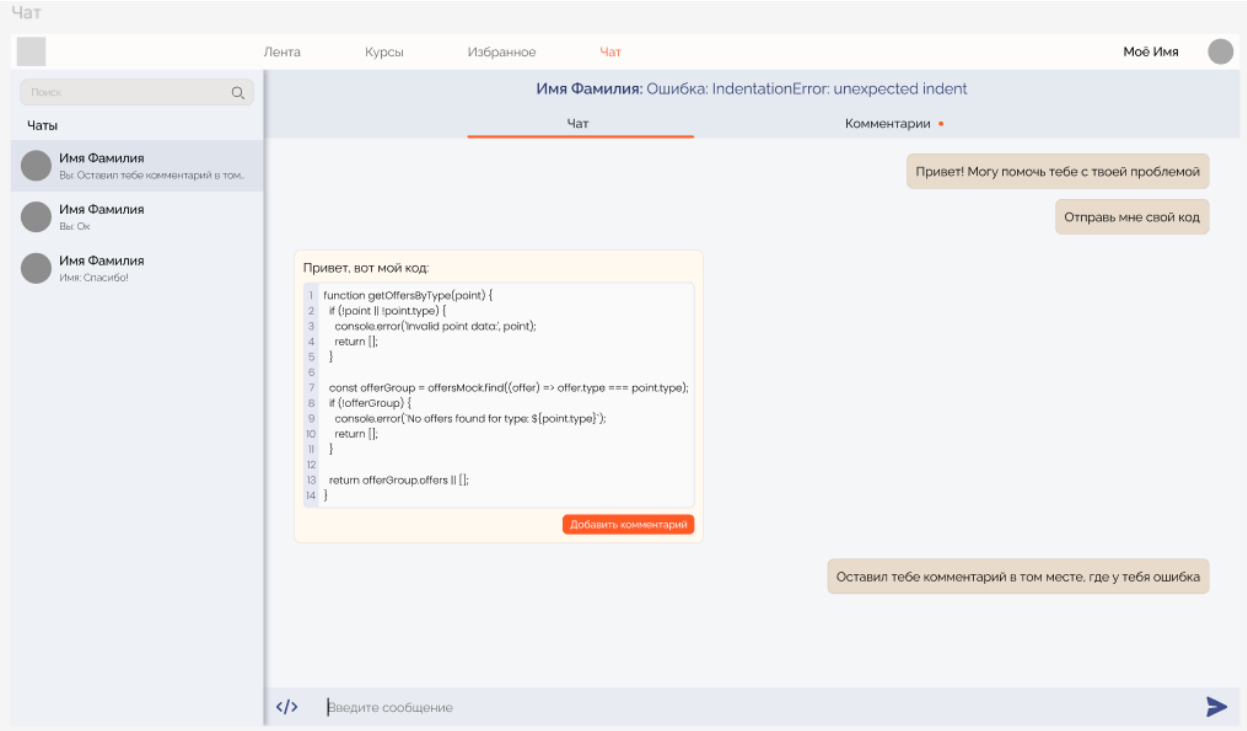


Рис.25. Пример дизайна чата

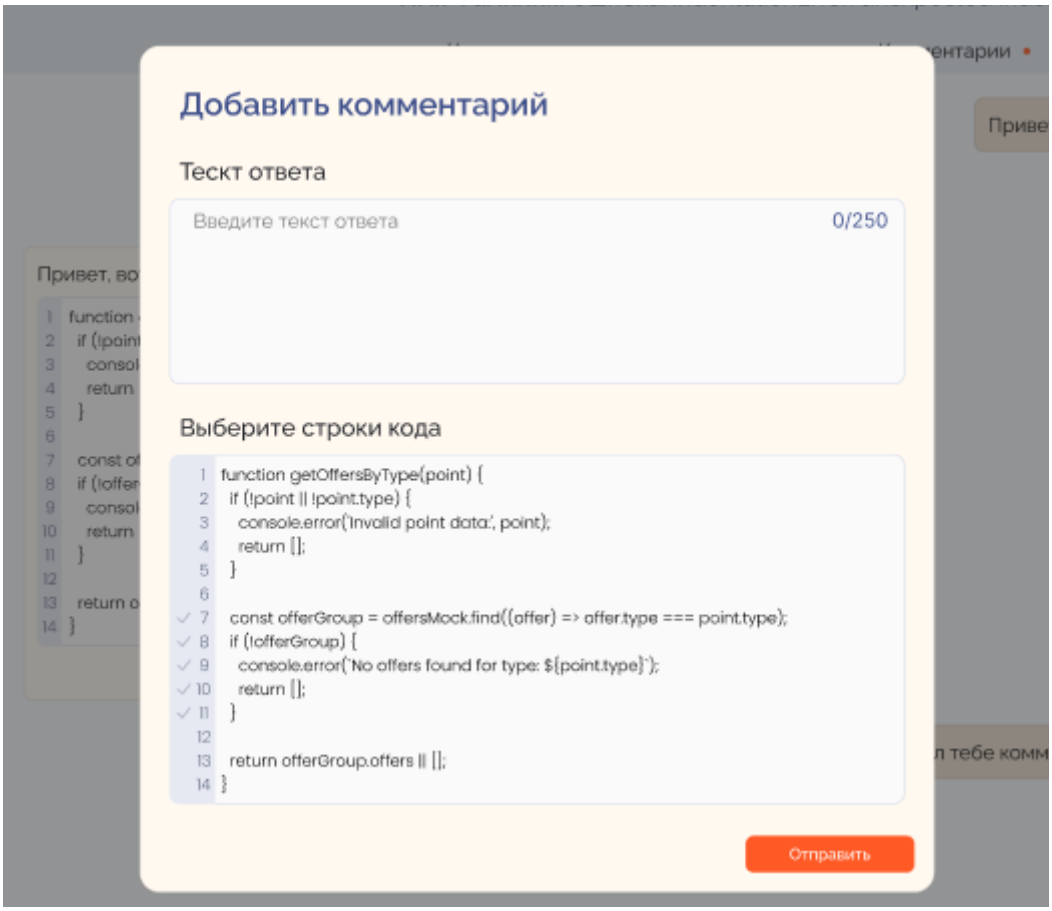


Рис.26. Дизайн режима работы с кодом в чате

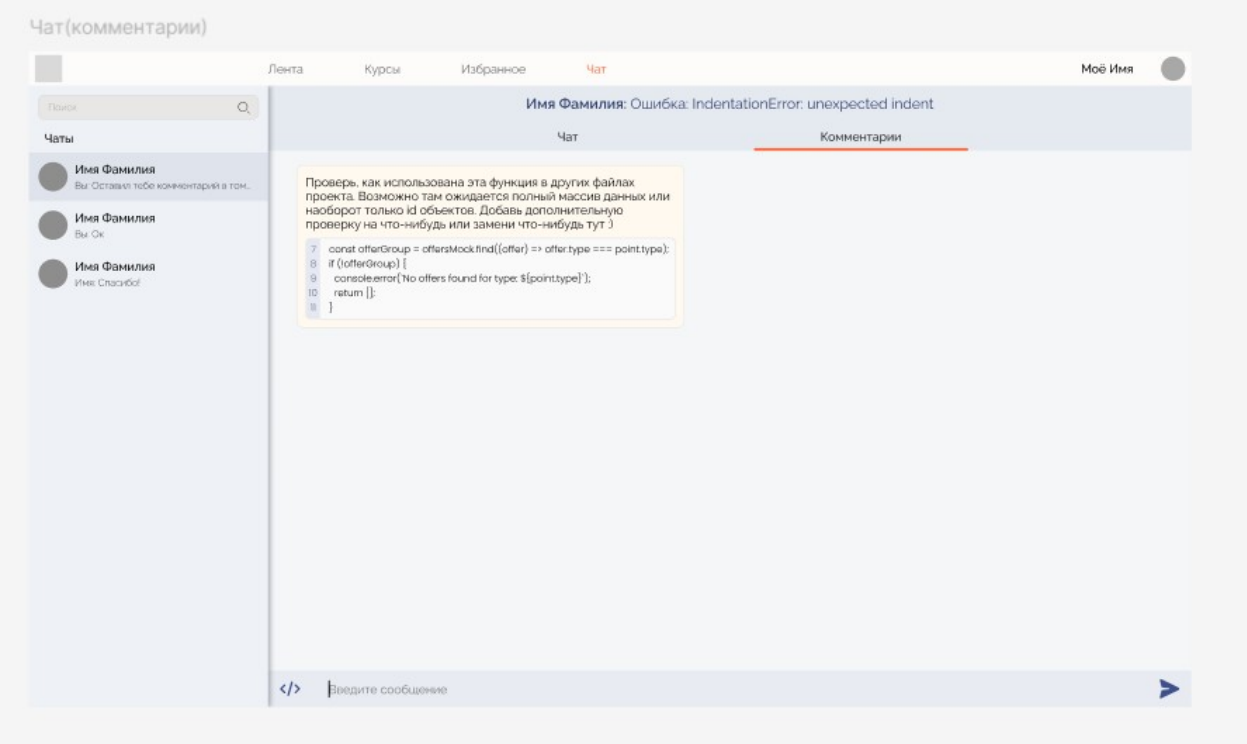


Рис.27. Пример комментария к коду

4.3 Работа аналитиков

На этом этапа аналитики доработали критерии mvp для текущих условий и стадии разработки проекта. Также изменены и дописаны такие документы, как Use Cases , User Story, провели оценку текущей стадии готовности продукта, а также протестировали продукт.

4.4 Работа разработчиков

Разработка front - и back – end части шла отдельно, поэтому на этом этапе разработчики занимались тестированием, исправлением багов, доработкой необходимого функционала сервиса, а также объединением этих двух частей в одно целое. Помимо этого разработчики занимались корректировкой уже написанных функций, т.к упрощение функционала требовало изменение уже существующего кода. Было проведено повторное тестирование тех функций, которые удалось реализовать.

Таким образом, в конце 3 этапа, наша команда имеет готовый сайт с итоговым дизайном, итоговым набором функций (как минимум MVP), протестированными разработчиками. В большинстве команда справилась с 3 этапом разработки и, несмотря на изменения и трудности в разработке, проект сохранил первоначальную идею.

5 Итоги

5.1 Выводы по проекту

В результате проделанной работы у нас получилось создать проект, который будет полезен студентам УрФУ, изучающим программирование — нашей ЦА. Часть функций не удалось реализовать(такие как удобный режим кода, а также профиль и панель админа для контроля работы сервиса, модерации постов, а также контроля поведения пользователей на нашей платформе) в силу недостатка времени и проблем в сфере разработки, также дизайн полностью перенести не удалось, но самый основной функционал и «фичи» сайта реализовать получилось, а дизайн в итоге получился проще, чем ожидали, но он соответствует допустимому формату дизайна для использования реальными пользователями.

5.2 Перспективы развития проекта

В перспективе проект можно будет раскрыть на общедоступном (не локальном) сервере, при условии доработки функционала, облегчающего пользователям работу с кодом, обмен материалами и взаимодействие с сайтом в целом, а также при условии полного переноса дизайна и анимаций, задуманных дизайнером, и добавлении профиля админа и соответствующего функционала для контроля адекватной работы сервиса и взаимодействия пользователей. Доработанный до такого уровня проект сможет стать удобными, уни-

кальным и незаменимым помощником для студентов во время обучения, а также поможет сделать их обучение более быстрым и продуктивным.

ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА

1. Методология Agile. (https://www.atlassian.com/ru/agile);
2. Про API с помощью Django REST framework. (https://habr.com/ru/articles/160117/);
3. Руководство по Django. (https://metanit.com/python/django/);
4. Как сделать сайт. (https://habr.com/ru/articles/273795/);
5. База данных для сайта. (https://habr.com/ru/sandbox/14484/);
6. Как составлять архитектуру базы данных (https://habr.com/ru/articles/514364/);
7. Как написать User Story.(https://habr.com/ru/articles/577420/);
8. Работа с React.(https://habr.com/ru/companies/ruvds/articles/343022/);
9. О технологии P2P.(https://skyeng.ru/magazine/wiki/it-industriya/chtotakoe-p2p/).