Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Институт математики и информационных технологий

Отчет о прохождении практики

Разработка веб-приложения рекомендаций для инвесторов

Выполнил студент группы 22307:

Бровкин А.П.

Направление подготовки:

Программная инженерия системное и прикладное ПО

Сроки прохождения практики:

29 мая 2023 г. - 9 июня 2023 г.

Место прохождения практики:

кафедра ИМО

Руководитель практики:

к.т.н., доцент
Богоявленская О. Ю.

Оценка _____

Дата _____

Петрозаводск 2023 г.

Оглавление

Введение

Всех инвесторов фондового рынка можно разделить на два типа: долгосрочные и краткосрочные. Долгосрочный инвестор – инвестор, вкладывающий финансы в активы, с целью получить выгоду через несколько лет или десятилетий. Краткосрочный инвестор продает свои активы в более ранний срок.

Для всех инвесторов главной проблемой является потеря денег.

Для долгосрочного инвестора она проявляется в сложности самоконтроля следования своей стратегии инвестирования. Эта сложность выражается в попытке погони за краткосрочной прибылью, которая зачастую бывает безуспешной. Если человек будет продавать ценные бумаги после окончания бурного роста или в моменты падения, то он будет терять финансы. Он не учитывает долгосрочной тенденции роста рынка и продает свои активы слишком рано от страха локальной просадки цены, теряя возможность продать их позже, но дороже. Подобные решения увеличивают риск потери денежных вложений человека

Целью проекта является разработка веб-приложения для автоматизации покупок ценных бумаг на фондовом рынке по алгоритму усреднения. Разрабатываемое веб-приложение предназначено для создания стратегии инвестирования и автоматизации покупок ценных бумаг на фондовом рынке с использованием алгоритма усреднения. Алгоритм усреднения – метод инвестирования, при котором инвестор покупает ценные бумаги постепенно и на протяжении длительного периода. Стратегия основана на тенденции, что рынок в долгосрочной перспективе в среднем растёт, несмотря на краткосрочные колебания. В результате инвестор покупает ценные бумаги постепенно, что позволяет усреднить стоимость приобретенных ценных бумаг. В итоге при росте стоимости ценных бумаг в долгосрочной перспективе, инвестор получает прибыль. Ему не придется следить за рынком самостоятельно и принимать решения о покупке и продаже, что позволит снизить риски ошибок.

Благодаря использованию современных технологий, веб-приложение будет способен проводить автоматические операции.

Задачи проекта:

1. Изучить язык Dart и framework flutter.
2. Изучить информацию о правильном инвестировании и стратегиях.
3. Разработать алгоритм расчёта рекомендаций.
4. Разработать веб-приложение, его функционал и пользовательский интерфейс.

Глава 1

Описание проектных решений

1.1 Стратегия инвестирования

Инвестиции – это вложение денежных средств с целью получения прибыли или сохранения капитала. Покупая ценные бумаги и продавая их через длительное время, каждый человек может заработать. Долгосрочное инвестирование — это инвестирование на протяжении от 3 лет. Долгосрочным инвесторам тяжело составлять стратегию для инвестирования и её придерживаться. Стратегия усреднения является простой и довольно эффективной.

Стратегия усреднения заключается в том, что инвестор покупает ценные бумаги не сразу, а постепенно. Это позволяет снизить среднюю цену бумаги, а в будущем получить больше прибыли, чем при единовременной покупке. Стратегия усреднения подразумевает, что вы регулярно или время от времени докупаете какой-то актив. Она позволяет снизить риск изменчивости и рассчитывать на большую доходность за счет усреднения стоимости актива в портфеле.

Долгосрочные инвесторы, которые инвестируют в фондовый рынок, используя брокеров. Проблема этой целевой аудитории – неудобство процесса инвестирования на фондовом рынке продолжительное время. Им нужно тратить время на составление стратегии и следование ей. Автоматизированная стратегия усреднения позволит инвестору забыть о хлопотах ручного отслеживания стратегии, и разрабатываемый программный продукт позволяет не допустить совершения этих ошибок.

Итоговый программный продукт – это веб-приложение, берущее на себя все сложности, связанные с долгосрочным инвестированием. Пользователь веб-приложения может собрать портфель ценных бумаг, которые он хочет приобрести. Следуя стратегии усреднения, для каждой ценной бумаги он может указать количество закупки и срок, за который нужно достичь этого количества. Также, можно указать валюту, сумму инвестирования и интервал покупки ценных бумаг.

Исходя из всех введённых пользователем данных, веб-приложение даёт рекомендации по приоритизации и количеству закупаемых ценных бумаг. Благодаря настройке портфеля, разрабатываемое веб-приложение даёт пользователю возможность создать именно тот портфель, который он хочет, не ограничивая его ни в чём.

1.2 Алгоритм расчёта рекомендаций

Для закупки портфеля ценных бумаг нужен алгоритм, позволяющий распределить имеющиеся средства в соответствии с ограничениями в виде срока закупки и необходимого количества. В соответствии с этими требованиями был разработан следующий алгоритм:

1. Вызывается функция по составлению рекомендаций со следующими параметрами: портфель, бюджет на портфель, срок портфеля, периодичность.
2. Считается текущий день инвестирования.
3. Формируются рекомендации по закупки для бумаг со сроком:
    1 Определяется количество оставшихся периодов инвестирования для бумаги.
    2 Берётся величина, равная частному от деления оставшегося к закупке количества ценных бумаг на оставшееся количество периодов инвестирования. Эта величина – количество закупаемых ценных бумаг за один период.
    3 Считается стоимость этих ценных бумаг.
        1 Если стоимость больше, чем оставшийся бюджет, то бумага пропускается.
        2 Иначе стоимость вычитается из оставшегося периода, днём закупки этой ценной бумаги назначается текущее значение дня инвестирования, а количеством – рассчитанное ранее количество.
    3 Если среди бумаг со сроком ещё остались те, которым не назначена рекомендация, то к текущему дню инвестирования добавляется период инвестирования, а к текущему бюджету прибавляется бюджет на период.
4. Считается наивысший процент закупки у бумаг без срока.
5. Формируются рекомендации по закупки для бумаг без срока:
    1 Бумаги сортируются по возрастанию процента закупки.
        1 Считается, сколько необходимо купить акций, чтобы достичь высшего процента закупки.
        2 Считается, сколько денег нужно для достижения этого процента:
            1 Если стоимость больше, чем оставшийся бюджет, то бумага пропускается.

2 Иначе стоимость вычитается из оставшегося периода, днём закупки этой ценной бумаги назначается текущее значение дня инвестирования, а количеством – рассчитанное ранее количество.

3 Если среди бумаг со сроком ещё остались те, которым не назначена рекомендация, то к текущему дню инвестирования добавляется период инвестирования, а к текущему бюджету прибавляется бюджет на период.

6. Рекомендации готовы.

```
budjet_period - Бюджет на период
budjet - Текущий бюджет
period - Период регулярных покупок
invest_day - День покупки ценных бумаг
stocks[] - Массив ценных бумаг
recommendation[] - Список рекомендаций к закупке

while не у всех бумаг со сроком есть рекомендации:
    for each in stocks[] со сроком и без рекомендаций:
        count = сколько нужно купить цб
        price = сколько это стоит
        if price > budjet:
            continue
        else:
            добавить цб в рекомендации со следующим днём инвестирования и количеством
while не у всех бумаг без срока есть рекомендации:
    отсортировать stocks[] по проценту закупки
    for each in stocks[] без срока и без рекомендаций:
        count = сколько нужно купить цб
        price = сколько это стоит
        if price > budjet:
            continue
        else:
            добавить цб в рекомендации со следующим днём инвестирования и количеством
return recomendation[]
```

Рис. 1.1: Псевдокод

Реализованный код алгоритма предоставлен в пункте "Приложение".

1.3 Использование flutter для создания веб-приложения

Flutter - это фреймворк разработки пользовательского интерфейса, который позволяет создавать кроссплатформенные мобильные, веб и настольные приложения с использованием одного и того же кода. Он разработан компанией Google и основан на языке программирования Dart.

Использование Flutter вышло из следующих его преимуществ:

1. Кроссплатформенность: изначально планируется создание веб-приложения, но используя Flutter на начальной стадии кодирования, в будущем можно быстро создать приложения и под IOS, и под Android, и под Desktop

2. Быстрая разработка и горячая перезагрузка: Flutter предоставляет мощные инструменты разработки, такие как горячая перезагрузка (hot reload), позволяющая мгновенно видеть изменения в приложении при внесении правок в код. Это значительно ускоряет процесс разработки и тестирования приложения.

3. Привлекательный пользовательский интерфейс: Flutter предлагает богатый набор виджетов и возможностей для создания привлекательного и интуитивно понятного пользовательского интерфейса. Это важно для нашего веб-приложения, так как пользователи должны легко взаимодействовать с функциональностью и получать информацию о стратегии усреднения.

4. Высокая производительность: Flutter использует свой собственный движок рендеринга, что обеспечивает высокую производительность и быструю отрисовку пользовательского интерфейса. Это важно для обеспечения плавности работы приложения и отзывчивого пользовательского опыта.

Таким образом, Flutter - это мощный инструментарий и возможность создания высококачественного и многоплатформенного приложения.

1.4 Подключение и работа с Firebase/Firestore

Firebase - это платформа разработки приложений, предоставляемая Google. Она предлагает широкий набор инструментов и сервисов для создания и развертывания приложений, включая хостинг, базу данных, аутентификацию, облачные функции, уведомления и многое другое. Одним из сервисов Firebase является Firestore.

Firestore - это гибкая и масштабируемая облачная база данных, которая предоставляет возможность хранить и синхронизировать данные в реальном времени между клиентскими приложениями и сервером. Firestore является NoSQL базой данных и хранит данные в виде коллекций документов, которые могут содержать различные поля и значения.

Так как при кодирование использовался Visual Studio Code (VSCode), то для подключения к Firebase и использования Firestore, необходимо выполнить следующие шаги:

1. Установление расширения Firebase в VSCode. В панеле расширений в VSCode, вводим "Firebase"в поле поиска и установливаем расширение "Firebase"от Google.

2. Создание проекта Firebase и настройка Firestore. Переходим на сайт Firebase (https://fireba
и создаём новый проект. Затем в настройках проекта активируем Firestore.

3. Подключение к Firebase в VSCode. Открываем палитру команд в VSCode, вводим "Firebase: Sign in"и выполняем вход в свою учетную запись Google, связанную с проектом Firebase.

4. Инициализируем проект Firebase в нашем приложении. Открываем терминал в VSCode и переходим в корневую папку нашего проекта. Затем выполняем команду firebase init‘ и следуем инструкциям, чтобы настроить проект Firebase в нашем приложении.

5. Используем Firebase и Firestore в нашем приложении. После подключения Firestore, используем его API для чтения и записи данных, а также реализовывания функциональности аутентификации пользователей с помощью сервисов Firebase Authentication.

Относительно хранения аккаунтов пользователей, Firebase предлагает сервисы аутентификации, которые позволяют регистрировать пользователей, аутентифицировать их и управлять доступом к ресурсам приложения. Firebase Authentication поддерживает различные методы аутентификации, включая электронную почту и пароль, аутентификацию через социальные сети, а также пользовательские методы аутентификации, мы остановились на электронной почте и пароле

Также Firebase Firestore использовался для хранения данных пользователей, таких как информация о портфеле и ценных бумагах. Создание соответствующие коллекций и документов в Firestore, где каждый документ будет представлять отдельного пользователя и содержать соответствующую информацию. Затем использовался API Firestore для чтения и записи данных пользователя в вашем приложении.

Например, Создана коллекцию "users"в Firestore и для каждого пользователя создаётся документ, где хранится информация о его портфеле и ценных бумагах.

1.5 Подключение и работа с TwelveData

Twelve Data – это финансовый API-сервис, который позволяет получать доступ к рыночным данным, таким как котировки ценных бумаг, фьючерсы, валюты и другие финансовые инструменты.

1. Регистрируемся на сайте Twelve Data
2. Получаем API-ключ. Для этого входим в свою учетную запись на сайте Twelve Data и переходим в раздел "API".
3. Настраиваем приложение, чтобы использовать Twelve Data API. Для этого нам нужно будет настроить соединение через IP-адрес. Для начала узнаём IP-адрес, с которого наше приложение будет отправлять запросы к Twelve Data. Если используется хостинг-провайдера, у него может быть уже назначен статический IP-адрес. Если нет, можем запросить его у нашего провайдера.
4. Добавляем свой IP-адрес в список разрешенных на странице "API Access"на сайте Twelve Data. Это позволит нашему приложению отправлять запросы к API.
5. Пишем код для запросов к Twelve Data API, используя API-ключ и IP-адрес.
6. Обрабатываем ответы от API в нашем приложении. Twelve Data API предоставляет различные методы для получения различных типов данных, включая котировки ценных бумаг, новости и другие финансовые данные.

Глава 2

Сравнение с аналогами

2.1 Анализ конкурентов

Наш сервис предоставляет следующие функциональные возможности:

- Автоматизация покупки ценных бумаг. Наш продукт автоматически проводит операции на рынке в соответствии с определённой стратегией усреднения.
- Аналитика портфеля. Мы предоставляем пользователю информацию о дохожности портфеля на данный момент относительно вложенных средств и на моменты в прошлом как для портфеля в целом, так и для каждой ценной бумаги в отдельности.
- Гибкая стратегия. Мы не навязываем пользователю портфель ценных бумаг или конкертную ценную бумагу к покупке, он сам выбирает, что он хочет купить. На рынке представлено множество продуктов, целью которых является помощь в инвестировании, но прямых аналогов обнаружено не было. Но многие из них используют одни и теже решения, поэтому для описания я выбрал 3 основных аналога, предлашающих разные варианты автоматизации покупки ценных бумаг.

1. Tinkoff invest
(ссылка на Tinkoff invest находится в списке использованных источников стр 18)

Tinkoff invest - это сервис, предоставляющий услуги тогровли на фондовой бирже, имеет доступ более чем к 10000 акций. Также они предоставляют инструмент "Следования функциональность которого отчасти является аналогом нашего сервиса. "Следование это опция повторения сделок за каким-то другим пользователем приложения. Такой подход позволяет автоматизировать процесс инвестирования, но не является гибким, то есть пользователь

не можем сам выбрать покупаемую бумагу. Также в приложении имеется функциональность аналитики портфеля, дополняющая инструмент "Следование".

2. Betterment
(ссылка на Betterment находится в списке использованных источников стр 18)

Betterment - это финансовый калькулятор. Этот продукт предлагает пользователю настроить параметры портфеля, на основании которых даёт прогнох по доходности, после чего берёт деньги пользователя и самостоятельно оперирует ими, совершая сделки на фондовом рынке, в соответствии с определёнными заранее параметрами. Параметры, которые сервис даёт определить это:

- Бюджет инвестирования. Это количество денег, которые пользователь готов вложить как стартовый капитал.
- Срок инвестирования. Это период, в который сервис будет распоряжаться деньгами пользователя, в этом и подобных сервисах обычно срок инвестирования исчисляется в годах, а максимальный составляет в районе 20 лет.
- Уровень риска. Из-за того, что фондовый рынок не является полностью прогнозируемым, пользователь всегда имеет риск потерять свои деньги, подобные сервсиы также не дают полной уверенности в получении желаемого дохода, поэтому они дают возможность пользователю самому определить риск потери своих финансов. Данный параметр влияет на набор ценных бумаг, которыми будет оперировать сервис, так как на рынке имеются стабильные ценные бумаги. Также этот параметр прямо пропорционально влияет на прогнозируемую итоговую прибыль, т.к. менее рисковые бумаги имеют меньший рост в цене.

Этот сервис полностью автоматизирует процесс покупки ценных бумаг, но так же, как и Tinkoff Invest, не является гибким. Также в сервисе присутствует аналитика портфеля енных бумаг

3. Финансовый консультант

Финансовый консультант - это человек или компания, дающие личные консультации по вопросам инвестирования своим клиентам. Подобные продукты предоставляют индиви-

дуальные рекомендации для каждого из своих клиентов. Это позволяет собирать гибкий портфель для пользователя, в отличие от предыдущих решений. Но подобный вариант не предоставляет возможности автоматизации покупки ценных бумаг. Подобные сервисы предоставляют детализированную индивидуальную аналитику, что является одним из главных плюсов таких решений.

Для наглядности эта информация приведена в таблице

Возможности сервиса	Финансовый Консультант	Tinkoff Invest	Betterment
Гибкая стратегия	Да	Нет	Нет
Аналитика портфеля	Да	Да	Да
Автоматизация покупок	Нет	Да	Да

Рис. 2.1: Возможности сервисов

2.2 Поиск по патентным базам

Также помимо прямого поиска конкурентов использовался поиск с помощью патентных баз: WIPO Patentscope и Google patents

Поиск происходил с помощью различных комбинаций ключевых слов и ключевых предложение. Далее отбирались патенты которые наиболее всего подходили на наш проект и анализировались на сходство. Среди таких был только один, "АВТОМАТИЗИРОВАННАЯ СИСТЕМА ИНФОРМАЦИОННОЙ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ ПО ФОРМИРОВАНИЮ ИНВЕСТИЦИОННЫХ ПОРТФЕЛЕЙ НА ОСНОВЕ ПОИСКА И АНАЛИЗА МНОЖЕСТВА ЭФФЕКТИВНЫХ РЕШЕНИЙ"
(Ссылка на патент находится в списке использованных источников, стр 18)

Найденная система представляет собой комплексный инструмент, который осуществляет поиск и анализ большого количества эффективных решений для формирования инвестиционных портфелей. Она может использовать различные алгоритмы и модели для определения оптимальных портфелей с учетом заданных параметров и критериев, таких как ожидаемая доходность, риск, ликвидность и др. Эта система может предоставить инвестору широкий выбор вариантов и помочь ему принять информированное решение.

Наше веб-приложение, основанное на стратегии усреднения, фокусируется на конкретной стратегии инвестирования. Стратегия усреднения предполагает покупку определенного количества ценных бумаг по порядку и на протяжении определенного временного периода. В отличие от первой системы, наше веб-приложение ограничено использованием конкретной стратегии и рекомендует инвестору только соответствующие активы и порядок их покупки.

Уникальность нашего продукта заключается в его специализации на стратегии усреднения и предоставлении пользователю конкретных рекомендаций в рамках этой стратегии. Он может предлагать простой и легко понятный интерфейс, упрощенные инструменты анализа и рекомендации, которые ориентированы на конкретную стратегию и удовлетворяют потребности инвесторов, предпочитающих использовать стратегию усреднения.

Заключение

В заключение хотелось бы подчеркнуть важность разработки веб-приложения для автоматизации покупок ценных бумаг на фондовом рынке. Как было упомянуто во введении, основной проблемой для всех инвесторов является потеря денег. Для долгосрочных инвесторов это проявляется в сложности самоконтроля следования своей стратегии инвестирования. Разработка веб-приложения, которое позволит автоматически покупать ценные бумаги по алгоритму усреднения, поможет решить эту проблему. Инвесторы смогут избежать ошибок, связанных с продажей ценных бумаг после окончания бурного роста или в моменты падения, что поможет им сохранить свои денежные вложения и даже получить прибыль в долгосрочной перспективе.

В рамках работы над проектом были достигнуты все задачи, в том числе рассмотрена информация об инвестировании и стратегиях, проведена разработка алгоритма расчета рекомендаций, а также создано само веб-приложение.

Список используемых источников

1. Как стратегия усреднения может помочь снизить риски инвестиций [Электронный ресурс]. Режим доступа: https://dokhodchivo.ru/cost-averaging.
2. Инвестиционные советники [Электронный ресурс]. Режим доступа: https://conomy.ru/analysis/articles/349
3. Пост с Пульса о стратегии Лесенка [Электронный ресурс]. Режим доступа: https://www.tinkoff.ru/invest/social/profile/Mistika911/88f1a7ba-8e54-45b3-95ad-ceff592755b5/.
4. WIki проект [Электронный ресурс]. Режим доступа: https://se.cs.petrsu.ru/wiki/BBBG.
5. Flutter документация [Электронный ресурс]. Режим доступа: https://docs.flutter.dev/
6. Dart документация [Электронный ресурс]. Режим доступа: https://dart.dev/guides
7. twelveData документация [Электронный ресурс]. Режим доступа: https://twelvedata.com/docs
8. Firebase документация [Электронный ресурс]. Режим доступа: https://firebase.google.com/docs?hl=ru
9. Добавления плагинов для работы с Firebase в проекте на Flutter [Электронный ресурс]. Режим доступа: https://firebase.google.com/docs/flutter/setup?hl=ruplatform=iosavailable-plugins
10. Поиск патентов - WIPO Patentscope [Электронный ресурс]. Режим доступа: https://www.wipo.int/patentscope/en/
11. Поиск патентов - Google patents [Электронный ресурс]. Режим доступа: https://patents.google.com/

12. Патент "Автоматизированная система информационной поддержки принятия решений по формированию инвестиционных портфелей на основе поиска и анализа множества эффективных решений"[Электронный ресурс]. Режим доступа: https://patentscope.wipo.int/search/en/detail.jsf?docId=RU250392622_cid=P10-LILOPK-97125-1

13. Конкурент 1 - Tinkoff invest [Электронный ресурс]. Режим доступа: https://www.tinkoff.ru/invest/

14. Конкурент 2 - Betterment [Электронный ресурс]. Режим доступа: https://www.betterment.com/

Приложения

Листинг 2.1: Код алгоритма, представленого в П 1.2 – Алгоритм расчёта рекомендация. Стратегия усреднения – AVGstrategy – принимает портфель ценных бумаг, бюджет на период и период, в течение которого бюджет нужно использовать, и возвращает рекомендации по закупкам ценных бумаг с учетом дней и сроков дедлайнов. Функция разделяет ценные бумаги на те, которые имеют дедлайны и те, которые не имеют, а затем вычисляет рекомендации для каждой группы, с учетом бюджета и процентов покупки

```
    body: Stack(
      children: [
        Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            Expanded(
              child: FutureBuilder<QuerySnapshot>(
                future: FirebaseFirestore.instance
                    .collection('users')
                    .doc(FirebaseAuth.instance.currentUser!.uid)
                    .collection('securities')
                    .get(),
                builder: (context, snapshot) {
                  if (snapshot.connectionState == ConnectionState.waiting) {
                    return Center(
                      child: CircularProgressIndicator(),
                    );
                  }
                  if (snapshot.hasError) {
                    return Text('Data receiver error: ${snapshot.error}');
```

```
                }
                if (snapshot.data == null || snapshot.data!.docs.isEmpty) {
                  return Container(
                    alignment: Alignment.center,
                    child: Text(
                      'You dont have securities, refer to the search',
                      style: TextStyle(fontSize: 24),
                    ),
                  );
                }
                final tickerDocs = snapshot.data!.docs;
                List<Stock> stocks = [];
                var nowDate = DateTime.now().toString();
                for (var tickerDoc in tickerDocs) {
                  stocks.add(Stock(tickerDoc['ticker'], tickerDoc['amount'],
                      tickerDoc['bought'], tickerDoc['term'], nowDate));
                }

                var schedule2 = AVGstrategy(stocks, 3000, 2);

                List<Recomendation> recos = [];
                schedule2.forEach((key, value) {
                  var reca = Recomendation(
                      key, value['count'], value['date'].toString());
                  recos.add(reca);
                });

                return ListView.builder(
                  itemCount: recos.length,
                  itemBuilder: (context, index) {
                    final recommendation = recos[index];
                    final ticker = recommendation.ticker;
                    final amountOrStocksPerPeriod =
                    final nextDate =
                        DateTime.parse(recommendation.investDay);

```

```
                    return Container(
                      color: Color(0xFFE6F4F1),
                      padding: EdgeInsets.symmetric(
                          vertical: 8.0, horizontal: 16.0),
                      child: Column(
                        crossAxisAlignment: CrossAxisAlignment.stretch,
                        children: [
                          Text(
                            'Ticker',
                            style: TextStyle(fontWeight: FontWeight.bold),
                            textAlign: TextAlign.center,
                          ),
                          SizedBox(height: 4.0),
                          Text(
                            ticker,
                            textAlign: TextAlign.center,
                          ),
                          SizedBox(height: 8.0),
                          Text(
                            'Recommendation',
                            style: TextStyle(fontWeight: FontWeight.bold),
                            textAlign: TextAlign.center,
                          ),
                          SizedBox(height: 4.0),
                          Text(
                            amountOrStocksPerPeriod,
                            textAlign: TextAlign.center,
                          ),
                          if (nextDate != null) ...[
                            SizedBox(height: 8.0),
                            Text(
                              'Date',
                              style: TextStyle(fontWeight: FontWeight.bold),
                              textAlign: TextAlign.center,
                            ),
                            SizedBox(height: 4.0),
```

```
                              Text(
                                DateFormat.yMMMd().format(nextDate),
                                textAlign: TextAlign.center,
                              ),
                            ],
                            SizedBox(height: 8.0),
                            Divider(),
                          ],
                        ),
                      );
                    },
                  );
                },
              ),
            )
          ],
        ),
      ],
    ),
  );
 }
}

Map AVGstrategy(List<Stock> portfolio, num period_budget, num period) {
  var schedule = new Map();
  if (period_budget == 0) {
    return schedule;
  }
  var stockWithTerm = Map();
  var stockWithoutTerm = Map();
  num goal = 0;
  for (var stock in portfolio) {
    schedule[stock.ticker] = {'count': '0', 'date': DateFormat};
    if (stock.term == 'none') {
      num boughtPercent = 100 * stock.bought ~/ stock.amount;
      if (goal < boughtPercent) {
```

```
129         goal = boughtPercent;
130       }
131       stockWithoutTerm[stock.ticker] = {
132         'stock': stock,
133         'boughtPercent': boughtPercent
134       };
135     }
136     else {
137       stockWithTerm[stock.ticker] = stock;
138     }
139   }
140
141   for (var stock in stockWithTerm.values) {
142     var lastInvestDay = DateTime.parse(stock.lastInvestDay);
143     List<int> termParams = [];
144     for (var element in stock.term.split('.')) {
145       int intel = int.parse(element);
146       termParams.add(intel);
147     }
148     var term = DateTime.utc(termParams[2], termParams[1], termParams[0]);
149     var now = DateTime.now();
150     var diffT = term.difference(now);
151     var interval = diffT.inDays;
152
153     if (interval <= 0) {
154       schedule[stock.ticker]['count'] = -1;
155       schedule[stock.ticker]['date'] = lastInvestDay;
156       continue;
157     }
158
159     var iterations = interval ~/ period;
160     var count = (stock.amount - stock.bought) ~/ iterations;
161     if (count == 0 && stock.amount > stock.bought) {
162       count += 1;
163     }
164     var summa = stock.price * count;
```

```
165     if (period_budget < summa) {
166       count = period_budget ~/ stock.price;
167       summa = stock.price * count;
168     }
169     period_budget -= summa;
170     schedule[stock.ticker]['count'] = count;
171     schedule[stock.ticker]['date'] = lastInvestDay.toString();
172   }
173   var stabilization = false;
174   var n = 0;
175   var budget = period_budget;
176   while (stabilization == false) {
177     var sortedStock = List.from(stockWithoutTerm.values);
178     sortedStock.sort((a, b) => a.boughtPercent.compareTo(b.boughtPercent));
179     ['boughtPercent']);
180     stabilization = true;
181     for (var stockInfo in sortedStock) {
182       var stock = stockInfo[1]['stock'];
183       var countOnPercent = stock.amount / 100;
184       var needlPercent = goal - stockInfo[1]['boughtPercent'];
185       var count = (needlPercent * countOnPercent ~/ 1).round();
186       var summa = stock.price * count;
187       if (budget < summa) {
188         count = budget ~/ stock.price;
189         summa = stock.price * count;
190       }
191       budget -= summa;
192       stockInfo[1]['boughtPercent'] =
193           (100 * ((stock.bought + count) / stock.amount)).round();
194
195       if (stockInfo[1]['boughtPercent'] < goal) {
196         stabilization = false;
197       }
198
199       if (count != 0 && schedule[stock.ticker]['date'] == 'none') {
200         schedule[stock.ticker]['count'] = count;
```

```
201         var lastInvestDay = DateTime.parse(stock.lastInvestDay)
202             .add((Duration(days: period.toInt() * n)));
203         schedule[stock.ticker]['date'] = lastInvestDay;
204       }
205     }
206
207     if (stabilization) {
208       var totalSumma = 0.0;
209       var count = 0;
210       for (var stock in stockWithoutTerm.values) {
211         var price = stock[1]['stock'].price;
212         totalSumma += price;
213       }
214       if (totalSumma == 0) {
215         break;
216       }
217       if (totalSumma <= budget) {
218         count = (budget ~/ totalSumma).round();
219         for (var stockInfo in stockWithoutTerm.values) {
220           var stock = stockInfo[1]['stock'];
221           if (schedule[stock.ticker]['date'] == 'none') {
222             int(lidParams[2]));
223             var lastInvestDay = DateTime.parse(stock.lastInvestDay);
224             schedule[stock.ticker]['count'] = count;
225             int number = period.toInt() * n;
226             var day = DateTime.parse(stock.lastInvestDay);
227             lastInvestDay = day.add(Duration(days: number));
228             schedule[stock.ticker]['date'] = lastInvestDay;
229           } else {
230             schedule[stock.ticker]['count'] += count;
231           }
232         }
233       } else {
234         for (var stockInfo in stockWithoutTerm.values) {
235           var stock = stockInfo[1]['stock'];
236           while (budget < stock.price) {
```

```
237             budget += period_budget;
238             n += 1;
239           }
240           int(lidParams[2]));
241           var lastInvestDay = DateTime.parse(stock.lastInvestDay)
242               .add((Duration(days: period.toInt() * n)));
243           budget -= stock.price;
244           schedule[stock.ticker]['count'] += 1;
245           schedule[stock.ticker]['date'] = lastInvestDay;
246         }
247         budget += period_budget;
248         n += 1;
249       }
250     }
251     budget += period_budget;
252     n += 1;
253   }
254
255   return schedule;
256 }
```