Отчет на тему 1

**На начальной стадии** мы организовали персональное хранилище кода, включив в него:

* Файл README.md с деталями проекта
* Файл .gitignore для выбранного языка программирования
* Директорию src/ с базовым скриптом

Выполняя последовательные сохранения изменений с содержательными комментариями, мы освоили ключевые возможности GitHub.

**На следующей фазе** мы исследовали функционал репозитория, изучая открытый проект на GitHub:

1. Изучили историю коммитов : в проект были внесены обновления README.md, дополнительные слияния pull request, удаление повторяющихся шагов и точек, незначительные исправления орфографических и грамматических ошибок в описаниях в data.json;
2. Проанализировали структуру проекта: присутствуют папки такие как .github, contributiong.md , README.md, data.json. Да, имеются файлы README.md, .gitignore, а также документация такая как сценарий и рабочие процессы.

**Мини-ревью на основе данных репозитория:**

1. **Читаемость и понятность**:
   * Логика реализации прозрачна и соответствует названию проекта
   * Применен единый стиль кодирования
   * Использованы информативные, но лаконичные наименования
2. **Структура и организация**:
   * Архитектура тщательно продумана
   * Отсутствует избыточное дублирование
   * Сложные алгоритмы сопровождаются пояснениями
   * Соблюден принцип единой ответственности методов

**Практики работы с версиями:**

* **Качество коммитов**:
  + Четкие, содержательные сообщения
  + Описание характера и цели изменений
  + Избегание расплывчатых формулировок
* **Частота фиксаций**:
  + Атомарные изменения (одна задача = один коммит)
  + Изолированные правки
  + Упрощение процесса код-ревью

**Преимущества подхода:**

* Прозрачная история изменений
* Упрощенный анализ модификаций
* Эффективный процесс проверки кода
* Быстрая идентификация проблем

Проект демонстрирует зрелый подход к разработке, сочетающий техническое совершенство с эффективными практиками контроля версий.

В проекте для код ревью использовались следующие инструменты:

GitHub Pull Request для отслеживания изменений и обсуждений.

Автоматические инструменты для проверки кода на соответствие стилистическим стандартам и выявления потенциальных ошибок и уязвимостей

Эффективность этих инструментов оценивалась по следующим критериям:

1. Выявление ошибок
2. Оптимизация производительности
3. Соблюдение стандартов
4. Обмен знаниями

Некоторые проблемы, которые можно заметить в истории и структуре коммитов:

* Неправильно скомпонованные изменения. В одном коммите сочетаются исправление ошибки, новые фичи, правки документации и тестов, что делает историю сложной для понимания.
* Незавершённые изменения. Более поздний коммит отменяет что-то из более раннего частичного коммита, что усложняет установление связей между изменениями.
* Ошибки в сообщении коммита. Некорректное описание изменений может привести к неправильному пониманию истории.
* Ненужные файлы в истории. Например, логи, файлы IDE или временный вывод при тестировании проекта, которые остались в истории, хотя в текущей версии проекта их нет.

Отчет на тему 2

1. Цели ревью для приведённого фрагмента кода:

проверить, соответствует ли код идиомам и шаблонам языка;

убедиться, что в коде используются возможности языка и стандартные библиотеки;

оценить, делает ли код то, что должен, и не содержит ли очевидных логических ошибок.

2. Задачи для ревьюера:

проверить, соответствует ли код стандарту PEP-8;

убедиться, что соблюдаются все соглашения языка и проекта;

проверить, что идентификаторам присвоены значимые и соответствующие стандартам имена;

оценить, свободен ли код от обфускации и ненужной сложности.

3. Объекты проверки:

использование функций языка и стандартных библиотек;

читаемость кода;

соответствие стандартам оформления;

тесты безопасности;

пригодность кода к обслуживанию.

Чек лист

1 **Работоспособность кода** - Проверить, что модуль выполняет свои функции корректно, логика реализована правильно, ошибок при запуске нет.

2 **Понятность кода** - Оценить читаемость: ясные названия методов и переменных, структура кода, наличие комментариев и документации.

3 **Соответствие стилю оформления** Соблюдение стандартов оформления (PEP-8 для Python, Google Java Style для Java): правильные отступы, имена, форматирование.

4 **Избыточность кода** - Обнаружить повторяющиеся участки, которые можно вынести в отдельные функции или методы для повышения повторного использования.

5 **Обработка исключений** - Проверить, что исключения используются по назначению, есть обработка ошибок, логирование или информирование пользователя.

6 **Безопасность** - Проверить валидацию входных данных: типы, диапазоны, формат, отсутствие уязвимостей (например, SQL-инъекций, XSS).

7 **Документация** - Наличие комментариев, docstring для функций, описание параметров и возвращаемых значений.

8 **Тестирование** - Есть ли юнит-тесты, покрывающие основные сценарии, граничные случаи, ошибки.

9 **Наличие unit-тестов для всех методов** - Проверить, что все публичные и важные методы протестированы.

10 **Проверка уязвимостей** - Анализ потенциальных уязвимостей, таких как SQL-инъекции, уязвимости безопасности, связанные с вводом данных.

**План этапов ревью**

|  |  |  |  |
| --- | --- | --- | --- |
| **Этап** | **Действия** | **Срок** | **Участники** |
| Подготовка | Выбор кода, назначение ревьюера, ознакомление с модулем | 45 мин | TechLead, Ревьюер |
| Анализ | Проверка кода по чек-листу, выявление проблем | 2 часа | Ревьюер |
| Обсуждение | Обсуждение выявленных вопросов, уточнение деталей | 30 мин | Автор, Ревьюер |
| Исправление | Внесение правок по результатам ревью | 1-2 дня | Автор |
| Повторная проверка | Анализ исправленного кода | 1 час | Ревьюер |
| Финальное утверждение | Подписание итогового варианта | 30 мин | TechLead, Автор |

**Метрики код-ревью модуля Test**

**1. Основные параметры:**

* **Объем кода:** ~200 строк
* **Сложность:** Средняя (требует детального анализа)
* **Временные затраты:**
  + **Анализ кода:** 2 часа
  + **Подготовка замечаний:** 1 час
  + **Обсуждение/доработки:** 1-2 часа
  + **Общее время:** 4-5 часов

Выводы (что было сложным в планировании? Как улучшить процесс?)

**Трудности при планировании заключались в следующем:**

* **Критерии оценки:** Необходимо было найти золотую середину при составлении чек-листа, чтобы пункты были понятными и охватывали все важные аспекты, но при этом не были слишком детализированными и сложными.
* **Временные рамки:** Важно было точно рассчитать, сколько времени потребуется на каждый этап, чтобы избежать как спешки, так и излишнего затягивания процесса.
* **Полнота проверки:** Было необходимо убедиться, что учтены все возможные аспекты, касающиеся безопасности, тестирования и стиля кода, чтобы не пропустить критические моменты.
* **Координация этапов:** Важно было четко распределить роли и обязанности между участниками, особенно при повторных итерациях, чтобы избежать путаницы и задержек.

**Для улучшения процесса можно предпринять следующие шаги:**

* **Автоматизация:** Внедрение инструментов статического анализа кода (таких как pylint, flake8, SonarQube) поможет снизить нагрузку на проверяющих и повысить точность выявления ошибок.
* **Чек-листы и шаблоны:** Использование заранее подготовленных шаблонов позволит ускорить подготовку к проверке и убедиться, что все необходимые аспекты учтены.
* **Ретроспективы:** После каждой проверки следует проводить анализ, чтобы выявить успешные моменты и области, которые можно улучшить, что позволит оптимизировать сроки и качество процесса.
* **Обучение:** Организация внутренних тренингов по лучшим практикам code review поможет повысить эффективность проверок и снизить влияние субъективных факторов.
* **Планирование времени:** Выделение конкретных временных интервалов для каждого этапа позволит избежать затягивания и обеспечить своевременное завершение процесса.

Отчет на тему 3

**Таблица сравнения офисных пакетов**

| Критерий | Microsoft Office 365 | Google Workspace | LibreOffice |
| --- | --- | --- | --- |
| **Совместимость форматов (DOCX ↔ ODT)** | В основном хорошая, но при конвертации возможны незначительные искажения форматирования | Умеренная, конвертация осуществляется через экспорт и импорт файлов | Превосходная, благодаря нативному формату ODT и хорошей поддержке формата DOCX |
| **Стоимость** | Платная подписка (от $6.99 в месяц) | Платная подписка (от $6 в месяц) | Бесплатный (Open Source) |
| **Совместное редактирование** | Поддерживается редактирование в реальном времени (через OneDrive/Teams) | Лучшая в своем классе поддержка (Google Docs) | Ограниченная функциональность (через облачные платформы, например, NextCloud) |
| **Автоматизация и скрипты** | VBA, Office Scripts, Power Automate | Google Apps Script (на основе JavaScript) | LibreOffice Basic, Python, API |
| **Мобильные приложения** | Полноценные приложения для iOS и Android | Отличные мобильные приложения (онлайн-офис) | Ограниченные возможности (только просмотр файлов) |
| **Автономная работа** | Полная поддержка | Работа в офлайн-режиме возможна, но с некоторыми ограничениями | Полная поддержка автономной работы |
| **Безопасность и приватность** | Корпоративный уровень (но данные хранятся у Microsoft) | Зависит от политики конфиденциальности Google | Локальное хранение данных, что обеспечивает высокий уровень приватности |

**Краткие выводы:**

* **Microsoft Office 365:** Подходит для пользователей, которым необходима широкая функциональность, корпоративные инструменты и полная поддержка офлайн-работы, но готовы платить за подписку и полагаться на облачные сервисы Microsoft.
* **Google Workspace:** Идеален для командной работы и пользователей, активно использующих облачные сервисы Google. Предлагает отличную функциональность для совместного редактирования, но менее надежен в автономном режиме.
* **LibreOffice:** Превосходный выбор для тех, кто ищет бесплатный, полнофункциональный офисный пакет с хорошей совместимостью форматов и высоким уровнем конфиденциальности, благодаря возможности локального хранения данных.

**Таблица сравнения браузеров**

| Критерий | Google Chrome | Mozilla Firefox | Microsoft Edge |
| --- | --- | --- | --- |
| **Использование оперативной памяти (10 вкладок)** | Высокое (около 1.5–2.5 ГБ) | Умеренное (около 1–1.8 ГБ) | Оптимизированное (около 1–2 ГБ) |
| **Влияние на батарею (10 минут работы)** | Среднее (относительно быстрый разряд) | Лучшее (более медленный разряд) | Лучшее (имеется режим экономии энергии) |
| **Скорость запуска** | Быстрый запуск (1–3 секунды) | Запуск средней скорости (2–4 секунды) | Очень быстрый запуск (1–2 секунды) |
| **Доступность расширений** | Обширная библиотека (Chrome Web Store) | Большая коллекция, но меньше, чем у Chrome (Firefox Add-ons) | Поддерживает расширения Chrome + собственные расширения |

**Краткий вывод:**

* **Google Chrome:** Браузер с самой большой библиотекой расширений, но потребляет больше оперативной памяти и может быстрее разряжать батарею.
* **Mozilla Firefox:** Более экономно использует ресурсы системы, чем Chrome, и имеет достойный выбор расширений.
* **Microsoft Edge:** Отличается высокой скоростью запуска и оптимизированным потреблением ресурсов, а также поддерживает расширения как Chrome, так и собственные.

**Таблица критериев средств просмотра видео**

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | KMPlayer | MX Player | VLC Media Player |
| Поддержка автоматизации и скриптов | Средняя (макросы, плагины) | Ограниченная (только основные функции) | Продвинутая (Lua-скрипты, API, CLI) |
| Функциональность на мобильных устройствах | Хорошая (Android, iOS) | Отличная (оптимизирован для Android) | Широкая (доступен на всех основных платформах) |
| Работа с субтитрами | Настраиваемые параметры, поддержка ASS/SSA | Автоматический поиск, синхронизация | Расширенная поддержка (включая субтитры, встроенные в видео) |
| Воспроизведение редких кодеков | Хорошая совместимость | Зависит от аппаратного декодирования | Максимальная совместимость (встроенные кодеки) |
| Автоматическая коррекция изображения/звука | Балансировка цвета и звука | Улучшение качества видео (Dolby Audio) | Ручная настройка (отсутствует автоматическая коррекция) |

**Краткий вывод:**

* **KMPlayer:** Предлагает сбалансированный набор функций, включая среднюю поддержку автоматизации и хорошую совместимость с различными кодеками.
* **MX Player:** Отлично подходит для Android-устройств благодаря оптимизации и функциям улучшения видео и звука.
* **VLC Media Player:** Универсальный плеер с расширенной поддержкой кодеков, скриптов и широкой доступностью на различных платформах, но не имеет автоматической коррекции изображения.

**Подведём итог:**

* **Для тесной интеграции и единой экосистемы:** Остановите свой выбор на продуктах Microsoft (Office 365, Edge) или Google (Workspace, Chrome), если вам важна слаженная работа приложений в рамках одного бренда.
* **Для тех, кто ценит бесплатность и открытый исходный код:** LibreOffice, Firefox и VLC станут отличным выбором, предлагая широкие возможности без каких-либо финансовых затрат.
* **Для мобильных устройств:** Google Workspace (для работы с документами), MX Player (для просмотра видео) и Firefox или Chrome (в качестве браузеров) обеспечат оптимальный опыт использования на смартфонах и планшетах.
* **Для профессионалов:** Office 365 (для работы с документами), Chrome/Edge (в качестве браузеров) и VLC (для работы с мультимедиа) предоставляют инструменты, необходимые для решения сложных задач.