

01 Локальный прокторинг

Итерация 1

Организационные моменты

Собрана команда и настроен канал связи  05.02.2025

Проведена проверка оформления профилей GitHub  07.02.2025

Получена тема и первый созвон команды  12.02.2025

Получен доступ к репозиторию и чатам  13.02.2025

Первый созвон с заказчиком и командное обсуждение  14.02.2025

Определены задачи на первую итерацию  16.02.2025

Введены Labels, Milestones  16.02.2025

Введена в работу GitHub wiki  17.02.2025 (после согласования)

Введен в работу GitHub project  19.02.2025 (после выдачи прав)

План на 1-ю итерацию

Подготовить вики-страницу с подробной постановкой задачи, с собранными и проанализированными требованиями, с описанием основной проблемы, которую решает проект, с категориями пользователей, что для каждого важно в проекте

Подготовить вики-страницу со сценариями использования и макетами UI

Ознакомление бакалавров с используемыми технологиями и заготовками основы для программы:

1. Прототип расширения для записи видео.
2. Dockerfile или Docker-compose, который содержит образы для сервера с Python Flask и базы данных MongoDB.
3. Инструкция о запуске и функционале прототипа в md формате.

Результаты 1-ой итерации

Задача 1.

Подготовлена вики о проекте.

<https://github.com/moevm/mse1h2025-prctr/wiki/%D0%9E%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0>

Локальный прокторинг (Local proctoring)

Ключевая проблема традиционных онлайн-прокторингов заключается в полной ответственности владельца прокторинга за технологическую инфраструктуру: настройка работы сервера, поддержание стабильной связи, обработка потокового обмена данными в режиме реального времени и распределение нагрузки в узкий промежуток времени. Наш локальный прокторинг решает эту проблему, переложив основную техническую нагрузку на клиентское устройство. Все, что требуется от сервера – это обеспечить успешную инициализацию с клиентом и своевременно получить данные после прокторинга, что существенно упрощает техническую реализацию и обслуживание. Такой подход снижает риск сбоев в работе системы, использует принцип зеркалирования при взаимодействии с сервером и повышает надежность обработки и хранения данных.

Данный проект разрабатывается на базе университета и имеет ключевое применение в нем, где студенты заинтересованы в корректной работе локального прокторинга со стороны клиента, а преподаватели – в успешном получении данных о локальных прокторингах на сервере. Расширение должно стабильно работать в браузере Google Chrome и быть независимым от операционной системы: Windows или Linux. Серверная часть использует для работы Docker с образами сервера Python Flask и базы данных MongoDB.

На базе клиента локально производится прокторинг со сбором метаданных прокторинга, логированием действий, визуальными подсказками и записью экрана со звуком и микрофона в локальный файл на компьютере клиента. С сервером происходит два взаимодействия: до записи – инициализация, после записи – передача данных. Данные на сервере разделены на базу данных MongoDB и директорию с записями.

Дополнительные моменты, которые также отмечены заказчиком:

- Расширение должно делать скриншоты, если не получается реализовать работу с видео.
- Запись веб-камеры во время прокторинга является отличным дополнением.
- Серверное GUI является отличным дополнением. В него входит веб-интерфейс, визуализация базы данных с таблицей и метаданными, а также видео-список с возможностью проигрывания.
- Авторизация пользователя и верификация клиента не важны.

Результаты 1-ой итерации

Задача 2.

Подготовлены вики о сценариях использования и макетах UI.

Сценарии использования:

GitHub wiki:

<https://github.com/moevm/mse1h2025-prctr/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%B8%D0%B8-%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F>

Макет UI:

Графическое представление: https://miro.com/app/board/uXjVlanJGFc=

GitHub wiki: <https://github.com/moevm/mse1h2025-prctr/wiki/%D0%9C%D0%B0%D0%BA%D0%B5%D1%82-UI>

Результаты 1-ой итерации

Задача 3. Issue 2 Basics.

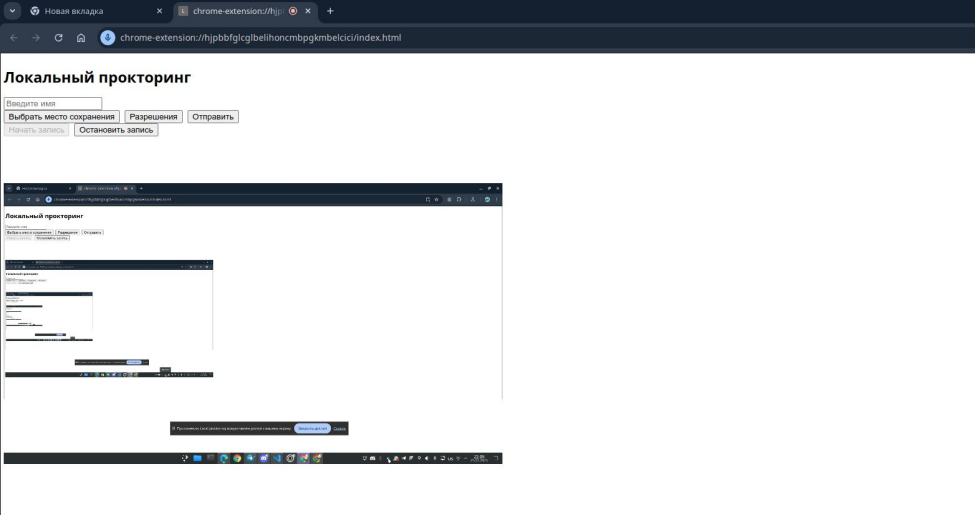
<https://github.com/moevm/mse1h2025-prctr/issues/2>

Бакалавры ознакомились с используемыми технологиями и сделали основы для программы:

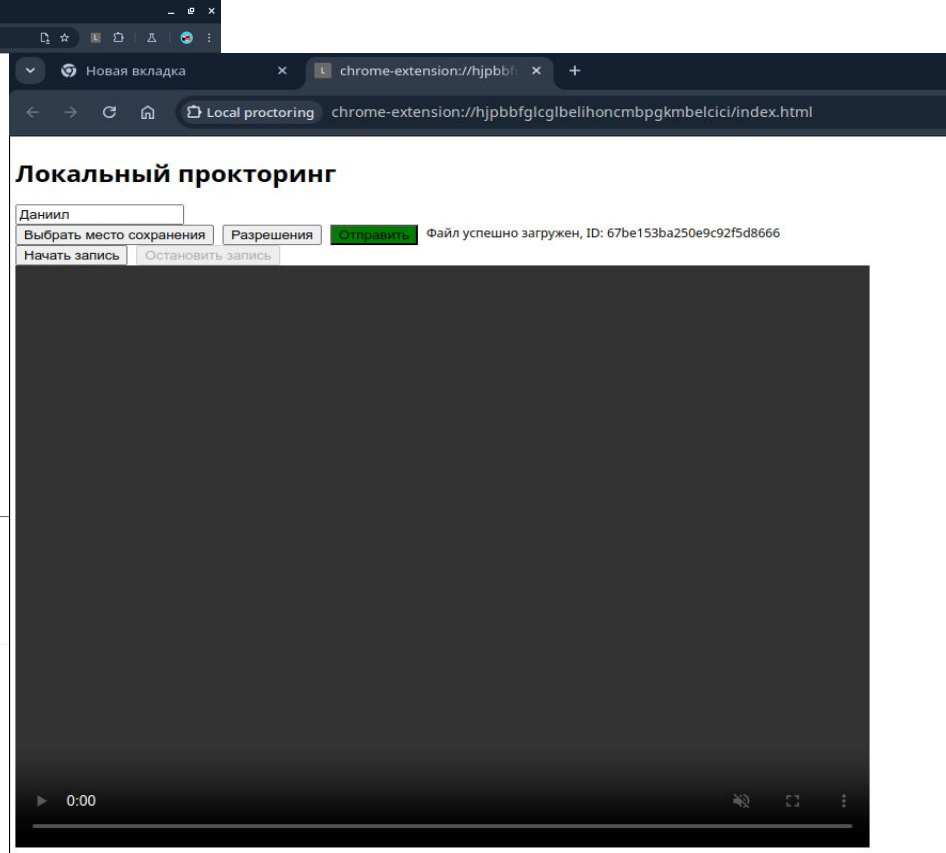
1. Прототип расширения для записи видео.
2. Dockerfile или Docker-compose, который содержит образы для сервера с Python Flask и базы данных MongoDB.
3. Инструкция о запуске и функционале прототипа в md формате.

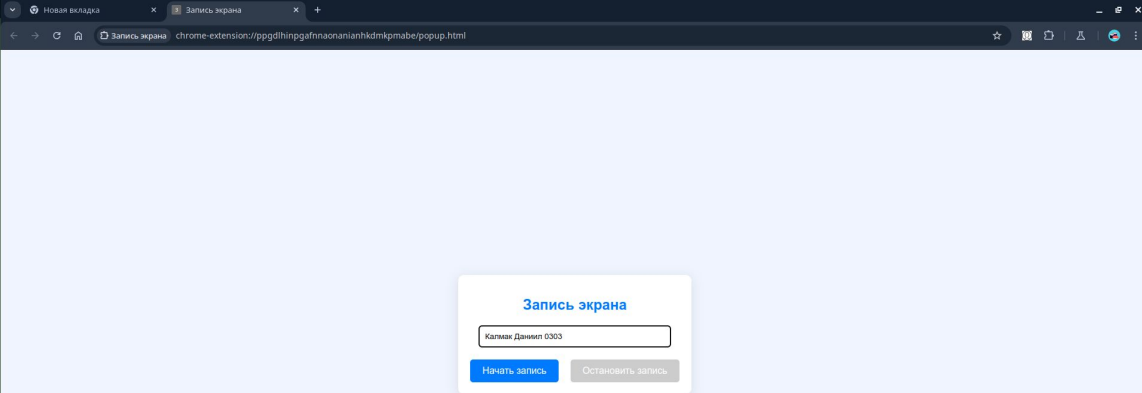
В качестве основы проекта выбрана программа, подготовленная Зазулей Илей.

На следующих слайдах представлены прототипы расширений.

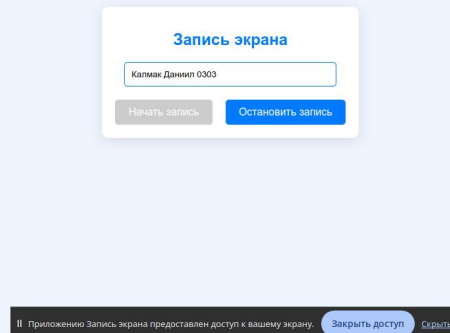
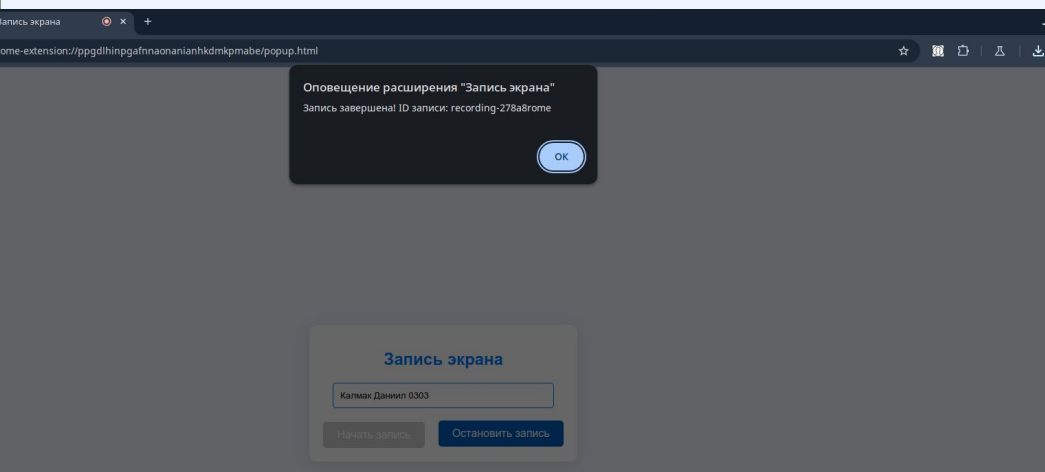
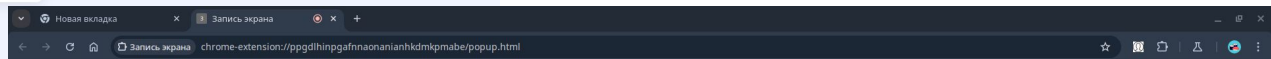


Зазуля Илья

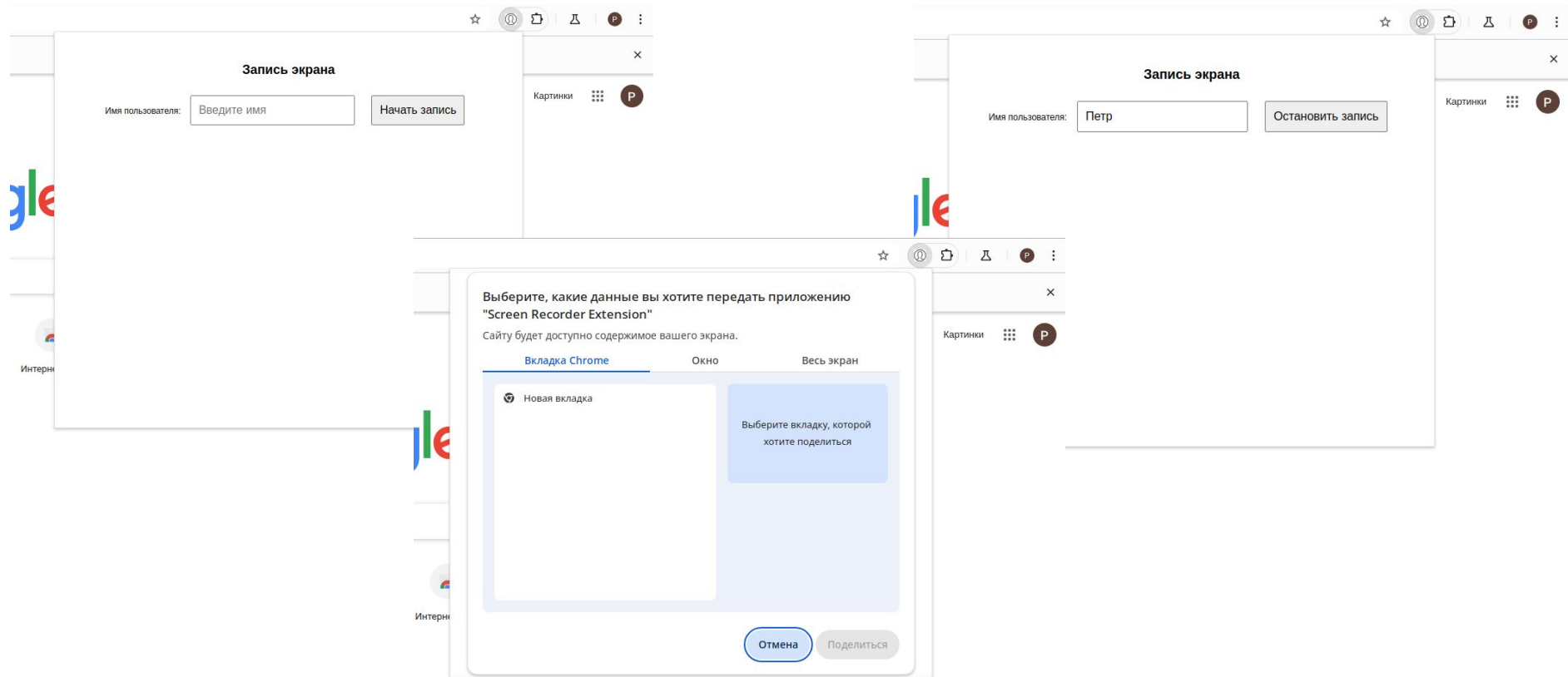




Заметаев Михаил



Шляхтин Максим



Долотов Никита

Screen Recorder

Имя пользователя:

Начать запись

Screen Recorder

Имя пользователя:

Остановить запись

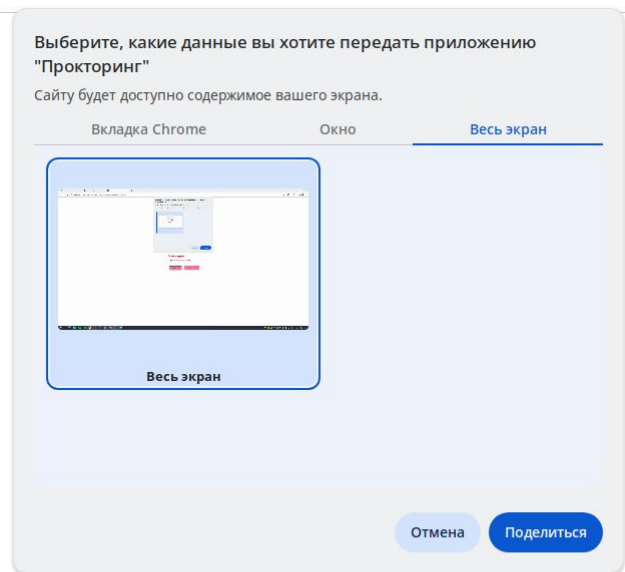
Резник Александра

Запись экрана

Введите ваше имя

Начать запись

Закончить запись



Запись экрана

Петр

Начать запись

Закончить запись

Оповещение расширения "Прокторинг"

Запись успешно завершена!

ОК

Запись экрана

Петр

Начать запись

Закончить запись

План на 2-ую итерацию

1. Подготовить версию 1 – частично работоспособная версия
2. Приложение корректно запускается: без ошибок и сбоев
3. Приложение реализует минимум один сценарий использования
4. Есть инструкция по развертыванию программы
5. Подготовить презентационные материалы

Первые задачи

Задачи разделены на пять направлений:

Кодовая база расширения.

Внешний вид расширения.

Сервер Python Flask.

База данных MongoDB.

DevOps. Docker.

Старт 2-ой итерации

1. Убрать сохранение в выбранную папку, заменив сохранением по умолчанию в папку Загрузки (Downloads) без запроса доступа у пользователя. Сделать поддержку фонового состояния окна расширения без явного открытия вкладок.
2. Перевести расширение в режим всплывающего окна без открытия дополнительных вкладок. Расширить внешний вид до полей Группа, Фамилия, Имя, Отчество.
3. Заглушка инициализации с клиентом, выдача ID сессии, запись в базу данных с датой выдачи. Написать запросы для первого сценария использования при работе с сервером.
4. Подготовить БД по заданному шаблону: {ID сессии (ObjectId), Группа, Фамилия, Имя, Отчество, Начало сессии (время в UTC0), Конец сессии (время в UTC0), Путь до записи, Статус записи (плохая/хорошая)}
5. Скорректировать Dockerfile и Docker-compose.yml под последние актуальные версии Python и MongoDB. Проверить соответствие последним стандартам Docker.