**МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ**

Профиль: Информационные технологии

**Практический командный кейс №2**

**«Управление столовой»**

**ИТОГОВОЕ ОПИСАНИЕ РЕШЕНИЯ**

Команда: <вставить название команды>

Участники: <ФИО участников>

Образовательная организация: <вставить>

Руководитель: <ФИО руководителя>

Москва, 2026 г.

**СОДЕРЖАНИЕ**

*Содержание будет сформировано автоматически. В Word: ПКМ по полю -> Обновить поле.*

**1. Введение**

Проект «Управление столовой» разработан как прикладное решение для школьной среды в рамках командного кейса Московской предпрофессиональной олимпиады. Система автоматизирует ключевые процессы питания: работу с меню, фиксацию выдачи, закупки продуктов, учет пользовательских действий и формирование отчетности для администрации.

В основе реализации выбран подход server-rendered web-приложения на Flask с базой данных SQLite. Такой стек обеспечивает быстрый запуск, прозрачную архитектуру и минимальные инфраструктурные риски в условиях ограниченного времени на проектирование и отладку.

**2. Актуальность**

Школьное питание затрагивает несколько групп пользователей с различными задачами и уровнями доступа. На практике это приводит к конфликтам данных, ошибкам ручного учета и непрозрачности закупочных решений. Для кейса важна не только реализация функций, но и эксплуатационная устойчивость: корректные числовые значения, защищенные действия и удобный интерфейс без визуальных сбоев.

Предложенное решение закрывает эти риски за счет жесткой ролевой модели, валидаций числовых данных, CSRF-защиты всех POST-операций, а также единой отчетной подсистемы с выгрузкой данных в формат, удобный для проверки и защиты результата.

**3. Цель и задачи проекта**

Цель проекта: создать устойчивую информационную систему управления школьным питанием для трех ролей (ученик, повар, администратор) с контролируемым доступом, предсказуемой обработкой данных и проверяемой отчетностью.

**Задачи проекта:**

1. Реализовать ролевую модель и ограничения доступа для всех бизнес-операций.

2. Обеспечить публичную регистрацию только для роли ученика.

3. Внедрить защиту форм от CSRF и единые правила обработки ошибок доступа.

4. Обеспечить числовую корректность для количеств и денежных значений без scientific notation.

5. Реализовать устойчивые интерфейсы для desktop и mobile без наложений элементов.

6. Подготовить админ-аналитику с фильтрами периода и CSV-экспортом.

7. Подтвердить корректность решения интеграционными и визуальными проверками.

**4. Требования кейса и принятые ограничения**

При подготовке решения учитывались официальные требования сезона 2025/2026: командный формат, выполнение практического кейса профиля «Информационные технологии», подготовка решения и документации для загрузки капитаном команды в систему практического тура.

В текущую итерацию включен весь обязательный функционал кейса №2. За рамки вынесены интеграции с внешними платежными шлюзами, миграция на PostgreSQL и продакшн-инфраструктура, так как они не влияют на прохождение регламентных сценариев защиты.

**5. Архитектура и стек решения**

Технологический стек: Python 3.12+, Flask 3, SQLite, Jinja2 templates, unittest для интеграционных проверок, Playwright для визуального debug loop.

Архитектурно проект разделен на модуль инициализации приложения, модуль аутентификации и ролей, модуль бизнес-маршрутов, модуль базы данных с миграциями и runtime-нормализацией, а также набор шаблонов пользовательского интерфейса. Такое разделение упрощает тестирование и снижает риск регрессий.

**6. Реализация сценариев для всех пользователей**

**6.1. Роль «Ученик»**

Ученик проходит публичную регистрацию без выбора роли, авторизуется, просматривает меню, выполняет демо-оплату, отмечает получение питания и ведет профиль (аллергии/предпочтения). Повторная отметка получения одного и того же приема пищи блокируется бизнес-правилом.

**6.2. Роль «Повар»**

Повар фиксирует выдачу, контролирует остатки и формирует заявки на закупку. При недостатке складских позиций используется отдельный сценарий исключения. Ввод количества и цены проходит строгую валидацию: запрещены экспоненциальная запись, NaN/inf, а также выход за лимиты.

**6.3. Роль «Администратор»**

Администратор обрабатывает заявки (одобрение/отклонение), управляет пользователями (роль, блокировка/разблокировка), анализирует метрики за период и выгружает отчет в CSV. Все критичные операции фиксируются в журнале admin\_actions с детализацией изменений.

**7. Безопасность и надежность данных**

Безопасность построена на сочетании RBAC и CSRF. Для всех POST-форм обязателен валидный CSRF-токен. Доступ к маршрутам ограничен по роли. При нарушении прав доступа система возвращает 403 с понятным русскоязычным сообщением.

Для устойчивости данных реализованы idempotent-миграции и таблица версий schema\_migrations. На старте приложения выполняется нормализация legacy-данных: записи закупок со статусом approved и нулевой ценой переводятся в rejected с системной причиной. Это исключает искажение отчетов и баланса.

**8. Отчетность и аналитика**

В админ-интерфейсе реализован периодный анализ по параметрам date\_from/date\_to. Отчет включает суммы оплат, количество получений питания, показатели выдачи, затраты по валидным одобренным закупкам и итоговый баланс. Метрики отображаются в веб-интерфейсе и выгружаются в CSV с одинаковой логикой расчета.

CSV-выгрузка сформирована в Excel-совместимом формате: UTF-8 с BOM, разделитель «;», включены поля generated\_at, period\_from, period\_to. Даты в интерфейсе и отчетах отображаются в формате DD.MM.YYYY HH:MM:SS.

**9. Тестирование и верификация**

Проверка решения выполнена набором интеграционных тестов (32 сценария) и визуальными проверками на нескольких брейкпоинтах. Автотесты покрывают RBAC, регистрацию только ученика, CSRF positive/negative path, блокировку пользователей, numeric-границы, legacy-policy, корректность отчетов и аудит админ-действий.

Дополнительно выполнен browser debug loop для ролей ученик/повар/администратор с фиксацией скриншотов интерфейса на разрешениях 360, 390, 768, 1024 и 1280 пикселей. Критерий приемки: отсутствие наложений и выходов элементов за границы карточек.

**10. План демонстрации на защите**

1. Регистрация нового пользователя и подтверждение автоматического назначения роли «Ученик».

2. Сценарии ученика: меню, оплата, отметка получения, проверка запрета повторной отметки.

3. Сценарии повара: фиксация выдачи, создание закупочной заявки с дробными значениями.

4. Сценарии администратора: обработка заявки, управление пользователем, проверка журнала действий.

5. Показ периодной аналитики и выгрузка CSV с сопоставлением ключевых метрик.

6. Короткий блок по безопасности: RBAC, CSRF, обработка 403.

**11. Выводы и готовность к сдаче**

Результат проекта соответствует обязательным требованиям кейса №2 и готов к демонстрации в рамках регламентной защиты. Реализация охватывает весь функциональный контур для трех ролей, имеет подтвержденную устойчивость пользовательского интерфейса, а также проверяемую корректность данных и отчетов.

Подготовлен полный пакет сопроводительных материалов: описание реализации, матрица покрытия требований, сценарий демонстрации, протокол тестирования и визуальные артефакты.

**12. Приложение: материалы проекта**

• Репозиторий: https://github.com/Pelmeshka126/Predprof-case2-canteen

• Полный текст кейса (Markdown): docs/full\_case\_text.md

• Матрица покрытия: docs/case2\_coverage.md

• Сценарий демонстрации: docs/demo\_script.md

• Протокол тестирования: docs/test\_protocol.md

• План по ролям: docs/role\_plan\_all\_users.md

• Визуальные артефакты: output/playwright\_v3/