Оценочные материалы для промежуточной аттестации по дисциплине

Разработка финансовых продуктов, 2 семестр

Код, направление подготовки	38.04.08 Финансы и кредит
Направленность (профиль)	Финансовые технологии
Форма обучения	Очная
Кафедра-разработчик	Экономики, учета и финансов
Выпускающая кафедра	Экономики, учета и финансов

Типовые задания для контрольной работы:

Решение практических задач.

Задача №1.

В компании разработали новое платёжное API. Перед внедрением в прод решили провести апробацию (бета-тестирование) в течение 4 недель.
План апробации:

- Цель: система должна выдерживать 10 000 успешных транзакций в час без критических ошибок.
- Критерий успеха: стабильность системы (процент успешных запросов) не менее

- Этапы тестирования:
  - -я неделя: ручное тестирование (10% от целевой нагрузки).
  - 2-я неделя: автоматизированное нагрузочное тестирование (70% нагрузки).
  - 3-я неделя: стресс-тест (120% нагрузки на коротких пиках).
  - 4-я неделя: эксплуатационная апробация (100% нагрузки + 5% случайных сбоев оборудования).

Результаты недельных испытаний:

Неделя	Кол-во запросов	Успешных ответов	Серьёзных сбоев (краш сервера)
			1 (восст. 15 мин)
			(восст. 30 мин и 45 мин)
			2 (восст. 10 мин и 20 мин)

Примечание: сбой считается, если все запросы в этот период завершились ошибкой на 5+ минут.

Вопросы:

остигнута ли цель по стабильности (≥99.5% успешных ответов) в целом за апробацию?

ожно ли считать апробацию успешной, если добавить условие: суммарное время недоступности из-за крашей не должно превышать 0.1% от времени тестирования? Общее время тестирования — 4 недели (672 часа), работа 24/7.

акое технологическое решение нужно срочно доработать перед релизом?

Задача №2.

Напишите функцию process_payment_webhook(data, current_timestamp), которая:

- Проверяет подпись
- Проверяет актуальность timestamp
- Сверяет сумму и комиссию с ожидаемыми
  (функция get_expected_from_db(payment_id) возвращает (expected_amount, expected_commission, status) — считай, что она уже реализована)
- Возвращает:
  - ("ok", payment_id) — если всё корректно и статус был "pending" (списание выполнено)
  - ("already_processed", payment_id) — если статус уже "paid"
  - ("invalid_signature", payment_id) — если подпись не совпала
  - ("expired", payment_id) — если timestamp старый
  - ("amount_mismatch", payment_id) — если сумма или комиссия не совпадают
  - ("invalid_status", payment_id) — если статус не "pending" и не "paid"

Пример
Вход:
python
```
data = {
   "payment_id": "p777",
   "user_id": 42,
   "amount": 100.00,
   "commission": 5.00,
   "timestamp": 1732123400,
   "signature": "расчетная_подпись"
}
current_timestamp = 1732123410   # разница 10 секунд (допустимо)
get_expected_from_db("p777") возвращает (100.00, 5.00, "pending")
```
Расчёт подписи:
Строка: "p77742100.005.001732123400" + "SECRET_KEY_2024" → MD5 → если совпадает со значением в data["signature"], то возвращаем ("ok", "p777").

Задача №3.

Вы разрабатываете backend-модуль для сервиса «Долями». Пользователь выбирает товар за total_price рублей и хочет разбить его на равные платежи (кроме, возможно, последнего) на parts частей. Банк берет комиссию commission_percent процентов от суммы каждой части (комиссия списывается сразу в момент оплаты части).
Входные данные (через аргументы функции):

- total_price (float) — полная стоимость товара (>0).
- parts (int) — количество частей (от 2 до 12).
- commission_percent (float) — комиссия банка за одну часть (от 0 до 10).

Правила начисления платежей:

1. Первые parts - 1 платежей — равные, округленные вниз до 2 копеек (т.е. до 0.01).
2. Последний платеж погашает остаток долга.
3. Важно: К сумме каждого списания (за каждую часть) сверху прибавляется комиссия банка. Пользователь в момент каждого платежа платит: Часть_долга + (Часть_долга * commission_percent / 100).
4. Если в какой-то момент из-за округлений суммарный долг становится отрицательным или пользователь переплачивает — система должна скорректировать последний платеж (уменьшить комиссию или последнюю часть).

Задание на программирование
Напиши функцию calculate_payment_schedule(total_price, parts, commission_percent), которая возвращает список сумм, которые пользователь реально заплатит (с комиссией) за каждую часть.
А также проверяет корректность: в конце список должен быть длиной parts, а сумма всех элементов должна быть равна total_price * (1 + commission_percent / 100) с погрешностью не более 0.01.

Задача №4.

Реализуйте функцию calculate_schedule(price, parts, commission), которая рассчитывает график платежей для сервиса «Долями».
Условия:

- Стоимость товара price (руб, >0) делится на parts равных частей (2–12).
- Первые parts-1 частей округляются вниз до копеек.
- Последняя часть = остаток долга.
- На каждую часть начисляется комиссия commission (%) от чистой части и округляется вверх до копеек.
- Итоговый платёж = чистая часть + комиссия.

Результат: список реальных сумм к оплате (с комиссией) для каждой части.
Пример:

- price = 100, parts = 3, commission = 10 → [36.67, 36.67, 36.68]

Требования: точность до копейки, обработка округлений, учёт остатка.

Задача №5.

Вы разрабатываете модуль финтех-агрегатора, который одновременно работает с тремя банками (А — REST, Б — SOAP, В — GraphQL). В логах обнаружена следующая ситуация:

1. При вызове fetch_account_balance("ACC123", "A") банк А вернул HTTP 429.
2. При повторном вызове через 2 секунды — снова 429.

3. При вызове fetch_account_balance("ACC123", "Б") банк Б вернул <fault><code>SOAP-ENV:Server</code><faultstring>DB timeout</faultstring></fault>.
4. При вызове fetch_account_balance("ACC123", "В") банк В вернул HTTP 200 OK, но в теле ответа: {"errors": [{"message": "Rate limit exceeded for token"}], "data": null}.
5. Вызов initiate_payment(..., bank_code="Б") для банка Б отправился, но ответ не пришёл (тайм-аут 5 секунд). Вы использовали идемпотентный ключ.

Вопрос: Какое из следующих утверждений верно в контексте корректной реализации финтех-агрегатора (согласно условиям кейса)?

Варианты ответов:

а) Для банка А после получения HTTP 429 нужно сразу вернуть success: false с error_code = "SERVER_ERROR", так как 429 — это ошибка сервера. Повторять запрос не требуется, клиент сам должен подождать.

б) Для банка Б при тайм-ауте платежа метод initiate_payment должен автоматически повторить запрос через 5 секунд с тем же идемпотентным ключом, чтобы гарантировать доставку, так как банк Б не поддерживает идемпотентность на своей стороне.

в) Для банка В, несмотря на HTTP 200 OK, агрегатор должен проверить наличие поля errors в ответе, и если оно есть — преобразовать ошибку в унифицированный формат, например, error_code = "RATE_LIMIT", даже если статус HTTP 200.

г) Rate limit банка А (100 запросов/мин) не нужно учитывать в функции fetch_all_transactions, так как пагинация и так замедляет запросы, и превысить лимит практически невозможно при выгрузке 50 000 транзакций.

Задача №6.

Вы финтех-стартап, запускаете BNPL (Buy Now, Pay Later) для e-commerce. Модель: платёж разбивается на 4 части, первые 25% — сразу, остальные — каждые 2 недели, без процентов для клиента. Доход вы получаете с комиссии от магазина (5% от суммы чека) и штрафов за просрочку (10% от платежа, но не более 15% от общей суммы заказа). Магазину платите транзакционные издержки (эквайринг 1,5% + шлюз 0,3%).
Средний чек — 10 000 руб.
Конверсия из визита в оплату через BNPL — 6%.
Повторная выдача BNPL тому же клиенту возможна только после погашения предыдущего.
Условия

- CAC (привлечение покупателя) = 800 руб. через соцсети.
- COC (привлечение магазина) = 15 000 руб. (партнёрский бонус, интеграция).
- Операционные расходы на клиента в месяц (серверы, поддержка, KYC, коллекшн): 30 руб.
- Когорта клиентов: 100 000 уникальных покупателей в месяц 1.
- Просрочка более 7 дней — 12% клиентов, из них 60% в итоге платят с комиссией (штраф), 40% уходят в коллектор (списание 100% остатка долга, но вы не несёте траты по корп. коллекшну — комиссия 50% от взысканного).
- Юнит-экономика считается на одного нового платного покупателя за 6 месяцев (3 цикла BNPL).

Задача

1. Построить юнит-экономику на 6 месяцев: доход с одного клиента (комиссии со всех его заказов + штрафы), переменные расходы (кредитные потери, когортный

CAC, транзакционные издержки, COC амортизированный на одного покупателя через магазины).

2. Рассчитать CAC payback period в месяцах.
3. Определить, при каком уровне дефолтности (сразу на 14+ дней) юнит-экономика становится отрицательной, если фикс. расходы не меняются.

Дополнительное испытание (финмодель)

Спрогнозировать EBITDA когорты за 12 месяцев с учётом того, что 30% покупателей совершают повторную BNPL-покупку на 8000 руб. с теми же условиями. Допустим, LTV (6 мес) = доход – кредитные потери – переменные издержки. Построить чувствительность к штрафной комиссии (8% вместо 10%).

Задача №7.

Апробация и тестирование новых технологических решений: «Внедрение voice-ID для антифрода»

Контекст

Банк с 5 млн активных цифровых клиентов. Хочет заменить часть SMS/IVR-аутентификации в контакт-центре на голосовую биометрию (voice-ID). Ожидаемые эффекты:

- Снижение времени верификации с 45 сек до 5 сек.
- Падение фрода на каналах удалённого обслуживания на 40%.
- Экономия на OTP-сообщениях (SMS стоит 1,5 руб./шт., их будет меньше).

Технология

Поставщик А предлагает активную биометрию (произнести фразу «В моём банке безопасно»). Точность FAR = 0,1%, FRR = 5%. Поставщик Б — пассивную (анализ 10 сек разговора с оператором). FAR = 0,01%, FRR = 8%.

Условия тестирования

- Нагрузка на контакт-центр: 300 тыс. верификаций в месяц.
- Текущий фрод через подмену голоса/социальную инженерию в ЦОД: 0,3% от звонков → средний ущерб на инцидент 15 000 руб.
- Длительность пилота: 2 месяца.
- Ложные срабатывания (FRR) требуют ручной аутентификации (оператор + KYC-вопросы) — дополнительные 180 руб. на случай.
- Ложные пропуски (FAR → реальный фрод, который не отсекли) = ущерб по 15 000 руб. на инцидент.

Задача (проектирование апробации)

1. Выберите метрики успеха не только технические (FAR/FRR), но и бизнес-метрики: время аутентификации, рост CSAT, снижение операционных затрат.
2. Опишите MVP-пилот:
   - На какой доле трафика тестировать (5%? с каких клиентов?).
   - Как обеспечить А/Б тест: voice-ID vs существующий процесс.
   - Что делать с «тихими» клиентами (не могут/не хотят голос).
3. Рассчитайте экономический эффект пилота за 2 месяца для каждого поставщика, с учётом FRR/FAR. Какой тип голосовой биометрии вы выберете и почему?
4. Какие риски внедрения в продуктивной среде вы видите (регуляторика, хранение голосовых слепков, атаки речевыми генераторами)? Предложите митигацию.

Спроектируйте план постепенного роллаута на все каналы (мобильное приложение + банкоматы голосовые команды). Какие A/B тесты нужно провести после пилота, прежде чем включать 100%?

Типовые тестовые вопросы:

В концепцию продукта должен входить анализ:
а) Только прямых конкурентов
б) Регуляторных ограничений (например, 115-ФЗ)
в) Вакансий в банках
г) Курсов криптовалют
ервый этап создания финтех-продукта:
а) Написание кода
б) Исследование и валидация идеи
в) Получение банковской лицензии
г) Найм команды поддержки
а каком этапе проводится A/B тестирование гипотез о конверсии в платежи?
а) Этап идеи
б) Этап разработки MVP
в) Этап масштабирования
г) Этап вывода на рынок
то такое «soft launch» в финтехе?
а) Публикация whitepaper
б) Запуск продукта на ограниченной аудитории (до 1000 чел)
в) Релиз без юридического лица
г) Бесплатное распространение
тап пост-релиза (post-launch) включает:
а) Сбор метрик и итерации
б) Проектирование архитектуры
в) Подписание контракта с эквайером
г) Формирование команды
акой из этапов наиболее критичен для продукта P2P-кредитования с точки зрения регулятора?
а) Дизайн интерфейса
б) Регистрация в реестре ЦБ РФ (оператор ПД, МФО или платформа)
в) Выбор хостинга
г) Написание пользовательского соглашения на 100 страниц
акой метод HTTP обычно используется для создания нового платежа в REST API?
а) GET
б) POST
в) PUT
г) DELETE
то означает HTTP статус 429 при вызове банковского API?
а) Серверная ошибка
б) Неверный API-ключ
в) Превышен лимит запросов (rate limit)
г) Платёж не найден
ачем нужен заголовок Idempotency-Key в API платежей?
а) Для шифрования запроса
б) Чтобы предотвратить двойное списание при повторе запроса
в) Для указания валюты

г) Для логирования IP-адреса

акая схема авторизации чаще всего используется для machine-to-machine (M2M) в банковских API?

а) Basic Auth с паролем

в) JWT, подписанный пользователем

г) SMS-пароль

Типовые вопросы к экзамену (2 семестр):

11. Дайте развернутое определение финтех-продукта. Чем он принципиально отличается от традиционной банковской услуги с точки зрения архитектуры и пользовательского опыта?

12. Опишите классификацию финтех-продуктов по сегментам (B2B, B2C, B2G). Приведите примеры ключевых характеристик для каждого сегмента.

13. Перечислите и раскройте суть пяти ключевых характеристик успешного финтех-продукта (например, скорость онбординга, персонализация, автоматизация).

14. Сравните необанк (например, Revolut, Tinkoff) и традиционный банк с точки зрения продуктовой линейки, клиентского пути и технологической инфраструктуры.

15. Объясните, как такие характеристики, как «бесшовность» (seamlessness) и «гиперперсонализация», реализуются на практике в мобильных кошельках и инвестиционных робоэдвайзерах.

16. Предложите три альтернативных способа применения технологии распределенного реестра (DLT) для решения проблемы высоких комиссий при международных переводах. Оцените плюсы и минусы каждого.

17. хочет автоматизировать закупку товаров у поставщиков в Китае. Сформулируйте альтернативные финтех-решения (не менее двух) для управления ликвидностью и валютными рисками.

18. Опишите, как можно по-разному решить задачу кредитования малого бизнеса без кредитной истории: с помощью альтернативных данных (альтдата), P2P-платформ и блокчейн-гарантий.

19. Разработайте альтернативные способы идентификации клиента (KYC) для жителей отдаленных регионов без возможности физического визита в отделение.

20. Поставщик платежей предлагает только рекуррентные платежи. Какие альтернативные финтех-архитектуры вы предложите для SaaS-проекта с моделью подписки?

21. Предложите алгоритм (последовательность шагов) для выбора между Open Banking API и Screen Scraping при интеграции с устаревшей банковской системой. Какие критерии будут решающими?

22. Кейс: стартап по микро-кредитованию. Обоснуйте выбор между созданием собственного скорингового движка на ML и покупкой готового облачного API решения.

23. Сравните технологические решения для хранения чувствительных платежных данных: токенизация, аппаратный HSM (Hardware Security Module) и облачный KMS (Key Management Service). В каком случае что предпочтительнее?

24. Как обосновать выбор между использованием готового белого лейбла (white label) и кастомной разработкой финтех-продукта с нуля? Приведите формулу расчета TCO (совокупной стоимости владения).

25. Разработайте критериальную матрицу для выбора технологии анализа транзакций (стриминг против батч-обработки) для антифрод-системы реального времени.

26. Опишите структуру концепции финтех-продукта: от гипотезы ценности (Value Proposition) до каналов дистрибуции и метрик успеха.

27. Разработайте концепцию «умной зарплатной карты» для фрилансеров, включая ключевые фичи (налоговый кешбэк, автоматическое резервирование, моментальные выплаты).

28. Как сформулировать гипотезы для нового BNPL (Buy Now Pay Later) сервиса при покупке цифровых товаров? Какие метрики подтвердят или опровергнут эти гипотезы?

29. Объясните роль JTBD (Jobs To Be Done) в разработке концепции финтех-продукта. Приведите пример для приложения по управлению семейным бюджетом.

30. Какие разделы обязательно должны быть в дорожной карте (Product Roadmap) финтех-продукта на первые 6 месяцев? Обоснуйте приоритет MVP-функций.

31. Перечислите полный цикл создания финтех-продукта от идеи до пострелиза. Выделите «воронки» рисков на каждом этапе.

32. Раскройте этап «Бизнес-анализ и формализация требований» для платежного шлюза. Какие артефакты создаются на этом этапе (User Story Mapping, BPMN диаграммы)?

33. Чем отличается этап проектирования финтех-продукта с высокой регуляторной нагрузкой (например, краудлендинг) от продукта без лицензирования (например, бюджетный трекер)?

34. Опишите этап «Техническое прототипирование» в финтехе. Какие инструменты и методы (mockups, clickable prototype, wireframes) используются и зачем?

35. Как этап «Масштабирование» отличается от этапа «Рост» для финтех-продукта с точки зрения архитектуры, команды и бюджета?

36. Объясните разницу между REST API и GraphQL на примере интеграции с банковским API для получения истории транзакций и баланса.

37. Опишите типичную последовательность шагов при интеграции платежного API (например, Stripe или Tinkoff): от получения токена до подтверждения операции вебхуком.

38. Какие методы обеспечения безопасности API в финтехе вы знаете? Опишите OAuth 2.0 и взаимные TLS (mTLS) применительно к Open Banking (PSD2).

39. Что такое идемпотентность запросов к платежному API? Приведите пример проблемы и решения, когда повторный запрос может привести к двойному списанию.

40. Разработайте план тестирования интеграции с банковским API, включая функциональные, нагрузочные и негативные сценарии (timeout, невалидный ответ).

41. Опишите архитектуру типичной P2P-кредитной платформы. Как обеспечивается соответствие заявок (матчинг) между инвесторами и заемщиками?

42. Какие уникальные скоринговые модели нужны для P2P-кредитования по сравнению с банковским? Как использовать социальные данные и поведенческие факторы?

43. Объясните механизм работы резервных фондов (Provision Fund) на P2P-платформе. Как это реализуется технически и юридически?

44. Предложите решение для автоматического реинвестирования возвращенных средств на P2P-платформе с учетом риск-профиля инвестора.

45. Как в продукте для P2P-кредитования решается проблема асимметрии информации и предотвращения мошенничества со стороны заемщиков (лицо-двойник, групповая подача)?

46. Раскройте формулу юнит-экономики для необанка с бесплатным тарифом. Как рассчитать CAC, LTV, ARPU и Gross Margin на одного активного пользователя?

47. Постройте финансовую модель для сервиса денежных переводов. Включите в ответ переменные: процент отказа транзакции, кэшбэк, комиссия за экспресс-перевод.

48. Как учесть в юнит-экономике финтех-продукта риск-метрики (например, долгосрочный дефолт по кредиту или чарджбэк по платежу)?

49. Кейс: вашему P2P-сервису нужно снизить Commission Fee на 30%, чтобы конкурировать. Пересчитайте модель: за счет каких юнитов (CAC, COGS, операционные расходы) вы компенсируете потерю прибыли?

50. Объясните понятие «Payback Period CAC» применительно к инвестиционному робоэдвайзеру. Что значит «хороший» и «плохой» период окупаемости в финтехе?

51. Опишите методику A/B тестирования для внедрения новой технологии биометрической аутентификации в мобильном банке. Как измерить влияние на конверсию и фрод?

52. Что такое песочница (sandbox) регулятора? Как использовать регуляторную песочницу для апробации блокчейн-решений в условиях, близких к реальным?

53. Разработайте план нагрузочного тестирования для API мгновенных платежей на Черную пятницу (Black Friday). Какие метрики (TPS, latency, error rate) критичны?

54. Как проводить канареечное развертывание (canary deployment) новой версии антифрод-модуля на реальном трафике без риска для клиентов?

55. Опишите процесс тестирования бэк-офиса (операционной панели) кредитного конвейера: как проверить корректность автоматического принятия/отказа заявок на исторических данных?

56. Составьте Go-to-market стратегию для нового финтех-продукта «Гарант сделок для маркетплейса мебели». Включите каналы привлечения и план коммуникации с ранними адаптерами.

57. Как управлять сопротивлением сотрудников legacy-банка при внедрении нового мобильного приложения вместо старого интернет-банка? Назовите 3 инструмента управления изменениями.

58. Опишите цикл вывода финтех-продукта на рынок по Agile-методологии: релизные циклы, фидбек-лупы с пользователями и priority-патчи.

59. Какие юридические и регуляторные изменения нужно провести до выхода на рынок продукта для краудфандинга? Опишите план взаимодействия с ЦБ (или другим регулятором).

60. Разработайте план пост-релизного сопровождения: SLA для платежного шлюза, мониторинг отказавших транзакций, процедура rollback при критическом баге в API.

Примерные темы курсовых работ.

1. Разработка Telegram-бота для P2P-кредитования с базовой скоринговой моделью. Программная часть: Python (aiogram), SQLite, простой ML-модель (sklearn). Суть: Бот принимает заявки, собирает альтернативные данные (например, ответы на вопросы), строит скоринг и матчит заёмщика с инвестором.

2. API-шлюз для агрегации банковских транзакций через Open Banking (Mock API) Программная часть: Node.js / FastAPI, REST API, OAuth 2.0 (симуляция), Docker. Суть: Разработка сервиса, который через единый интерфейс обращается к fake-банкам, нормализует данные транзакций и отдаёт их в JSON.

3. Калькулятор юнит-экономики финтех-продукта с веб-интерфейсом Программная часть: React + TypeScript (front), Flask (back), математическая модель LTV/CAC/ARPU. Суть: Приложение, где пользователь вводит параметры (комиссия, отток, CAPEX), а система строит прогноз и графики окупаемости.

4. Симулятор отказоустойчивости платежного API (Chaos Engineering для финтеха) Программная часть: Go / Python, async/await, эмуляция таймаутов, идемпотентности и вебхуков. Суть: Генератор запросов к тестовому платежному API с проверкой, что повторные запросы не приводят к двойным списаниям.

5. Веб-приложение для A/B-тестирования методов аутентификации (биометрия vs OTP) Программная часть: Django, Redis для хранения сессий, интеграция с простым эмулятором биометрии. Суть: Инструмент для анализа, какой метод аутентификации повышает конверсию входа в "банк" по заданным метрикам.

6. Анализатор чарджбэков (Chargeback Predictor) на основе истории транзакций Программная часть: Pandas / Polars, LightGBM, FastAPI. Суть: Модель, которая загружает

CSV-лог транзакций, обучается предсказывать риск возврата платежа и отдаёт результат через простое API.

7. Дэшборд мониторинга API-метрик для финтех-продукта (QLiksense-style, но самописное) Программная часть: Python (Streamlit / Dash), получение метрик из Prometheus или из файлов логов. Суть: Визуализация TPS, latency и error rate для интеграции с платежным шлюзом.

8. Система автоматического резервирования средств (Escrow Agent) для маркетплейса на смарт-контрактах. Программная часть: Solidity (Ethereum), Hardhat, Web3.js, локальная сеть Ganache. Суть: Умный контракт, который удерживает средства до подтверждения сделки – аналог "Гаранта сделок".

9. Плагин для блокировки Screen Scraping (анти-парсинг) для учебного банковского сайта. Программная часть: JavaScript (изменение DOM), Headless-детекция, Python для демонстрации атаки и защиты. Суть: Демонстрация борьбы с несанкционированным сбором данных – защита от альтернативных решений без Open Banking.

10. Рекуррентный платежный движок (Subscription Engine) с веб-интерфейсом. Программная часть: FastAPI, Celery + Redis, PostgreSQL, интеграция с любым mock-банком. Суть: Сервис, который создаёт подписки и автоматически списывает средства по расписанию, обрабатывая отказы.

11. Прототип агрегатора альтернативных данных для скоринга (альтдата-бюро) Программная часть: Scrapy (парсинг открытых источников), Pandas, простой REST API. Суть: Сбор данных из соцсетей или платёжных чеков (агрегированных) для оценки кредитоспособности.

12. Классификатор транзакций (Transaction Enricher) для личного бюджета Программная часть: Python + spaCy / NLTK, либо ruGPT для категоризации. Суть: Модель, которая получает строку "Оплата в Магнит" и возвращает категорию ("Продукты") и логотип – ключевой компонент необанков.

13. Имитационное моделирование P2P-платформы (NetLogo или Mesa на Python). Программная часть: Mesa (agent-based modeling), Matplotlib. Суть: Агентная модель, где заёмщики дефолтят с определённой вероятностью, а инвесторы распределяют средства. Анализ эффективности резервного фонда.

14. CLI-утилита для нагрузочного тестирования платежного API. Программная часть: Go (goroutines) или Python (asyncio + aiohttp). Суть: Утилита запускает N параллельных запросов к API, замеряет перцентили задержек и проверяет идемпотентность.

15. Простой краудфандинговый смарт-контракт с механизмом голосования за выпуск средств Программная часть: Solidity, JavaScript (тесты на Mocha/Chai). Суть: Если проект собирает нужную сумму, владелец может потратить средства только после одобрения >50% бекеров (децентрализованная гарантия).