3. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К КУРСОВОЙ РАБОТЕ

3.1 Задание

Цель работы: Разработать десктопное многооконное приложение на языке C++ в среде Qt Creator, реализующее функционал предметной области по варианту. Распределение вариантов осуществляется преподавателем **до первого контрольного срока**. Далее вся работа по курсовому проекту проходит в рамках назначенной темы.

Требования: Приложение должно содержать минимум 5 классов, отражающих сущности предметной области, поддерживать работу с несколькими пользователями. Обязательно использование минимум двух паттернов проектирования из следующих: Observer, Strategy, Factory, Command, Composite. Данные приложения (пользователи, сущности, настройки) должны сохраняться локально (JSON, XML, или SQLite).

Варианты:

1) **Редактор тестов для проверки знаний по предмету (2 чел.)**

✓ Создание, редактирование и удаление тестов и вопросов.
✓ Поддержка типов вопросов: одиночный выбор, множественный выбор, открытый ответ.
✓ Привязка тестов к предметам и темам.
✓ Прохождение теста с автоматической проверкой и подсчётом баллов.
✓ Поиск по предмету, типу вопроса, сложности.
✓ Сохранение результатов, импорт/экспорт тестов.
✓ Визуализация статистики (рейтинг, прогресс).

Рекомендуется использовать следующие сущности:

- User — студент/преподаватель (разный интерфейс)
- Test — тест (название, предмет, тема)
- Question — вопрос (текст, тип, сложность)
- Answer — ответ (правильный/неправильный, текст)
- TestSession — сессия прохождения, результаты

Рекомендуемые паттерны:

- Factory — создание разных типов вопросов.
- Observer — обновление прогресса и UI при изменении состояния.

2) **Организация работы школьной столовой (2 чел.)**

✓ Управление меню и ценами.
✓ Оформление заказов с автоматическим списанием средств.
✓ Поиск блюд по категории, цене.
✓ Фильтрация заказов по дате, ученику.
✓ Отчёты: выручка, популярные блюда.
✓ Импорт/экспорт меню, сохранение при выходе.

Рекомендуется использовать следующие сущности:

- User — администратор/ученик (разный интерфейс)
- Meal — блюдо (название, цена, категория)
- Order — заказ (дата, блюда)

• Category — завтрак, обед, перекус
• ReportManager — формирование отчётов

Рекомендуемые паттерны:

• Strategy — разные стратегии сортировки и формирования отчётов.
• Observer — обновление баланса ученика при оформлении заказа.

3) Создание комментируемых электронных фотоальбомов (1 чел.)

✓ Добавление, редактирование, удаление фото.
✓ Организация по альбомам и папкам.
✓ Поиск по тегам, дате, описанию.
✓ Импорт/экспорт альбома (со структурой).
✓ Визуализация хронологии (лента).

Рекомендуется использовать следующие сущности:

• User — владелец альбома
• Photo — изображение (файл, описание, дата)
• Album — альбом (название, теги)
• Tag — метка (например, "отпуск", "семья")
• PhotoManager — загрузка, экспорт, поиск

Рекомендуемые паттерны:

• Composite — вложенные альбомы в виде дерева.
• Strategy — разные стратегии поиска (по тегу, по дате).

4) Работа ремонтной мастерской техники (2 чел.)

✓ Оформление заказа клиентом.
✓ Приём заказа, смена статуса "принят", "в работе", "готов", "выдан").
✓ Учёт и списание запчастей при ремонте.
✓ Поиск заказов по клиенту, статусу, дате.
✓ Формирование счёта и истории заказов.
✓ Экспорт истории заказов.

Рекомендуется использовать следующие сущности:

• User — менеджер/клиент (разный интерфейс)
• RepairOrder — заказ (устройство, неисправность)
• Device — тип устройства (ноутбук, телефон)
• Part — запчасть (артикул, количество)
• StatusTracker — отслеживание статуса

Рекомендуемые паттерны:

• Strategy — смена поведения по статусу.
• Observer — уведомления о смене статуса.

5) Планировщик задач по разным темам с уведомлениями (1 чел.)

✓ Создание, редактирование, выполнение задач.
✓ Группировка по проектам, категориям.
✓ Напоминания за заданное время до дедлайна.
✓ Поиск по приоритету, дате, проекту.
✓ Импорт/экспорт списка задач.

Рекомендуется использовать следующие сущности:

• User — владелец задач
• Task — задача (название, дата, приоритет)

• Project — проект или категория
• Reminder — напоминание (время до дедлайна)
• TaskManager — сортировка, фильтрация

Рекомендуемые паттерны:

• Command — отмена/повтор действия.
• Observer — автоматическое обновление при наступлении напоминания.

6) Обучающее приложение для детей дошкольного возраста (2 чел.)

✓ Прохождение интерактивных заданий.
✓ Система баллов и уровней сложности.
✓ Призы и поощрения за выполнение.
✓ Поиск заданий по типу (цвета, формы).
✓ Визуализация прогресса (прогресс-бар, диаграммы).
✓ Импорт/экспорт заданий и профиля ребенка с прогрессом.

Рекомендуется использовать следующие сущности:

• User — ребёнок и родитель/куратор (разный интерфейс)
• Exercise — задание (буквы, цифры, фигуры)
• Reward — награда (баллы, значки)
• Level — уровень сложности
• ProgressTracker — статистика выполнения

Рекомендуемые паттерны:

• Strategy — выбор типа задания.
• Observer — обновление прогресса после выполнения.

7) Система ведения личного дневника с тегами и поиском (1 чел.)

✓ Добавление, редактирование, удаление записей.
✓ Поиск по тегам, дате, тексту.
✓ Фильтрация по месяцу, настроению.
✓ Визуализация активности (календарь записей).
✓ Экспорт и импорт дневника.

Рекомендуется использовать следующие сущности:

• User — автор записей
• Entry — запись (текст, дата, теги)
• Tag — метка (настроение, тема)
• SearchManager — поиск по ключевым словам
• ExportManager — импорт/экспорт

Рекомендуемые паттерны:

• Strategy — разные стратегии поиска (по тегу, по дате).
• Command — отмена удаления записи.

8) Организация работы магазина бытовых товаров (3 чел.)

✓ Учёт товаров: артикул, название, цена, остаток.
✓ Покупка товаров, заказ.
✓ Оформление продаж и поставок товаров.
✓ Поиск по артикулу, названию, категории.
✓ Отчёты: прибыль, популярные товары.

✓ Импорт/экспорт данных о товарах и операциях.
Рекомендуется использовать следующие сущности:
• User — кассир, администратор, клиент (разные интерфейсы)
• Product — товар (артикул, цена, остаток)
• Sale — продажа (чек, состав)
• Supply — поставка товаров
• ReportManager — прибыль, топ-товары
Рекомендуемые паттерны:
• Factory — создание разных типов операций (продажа/поставка).
• Observer — обновление остатков в реальном времени.

9) Учет персональной библиотеки книг (1 чел.)

✓ Добавление, редактирование книг.
✓ Поиск по автору, названию, жанру.
✓ Сортировка по алфавиту, году.
✓ Формирование списков «Прочитанные», «Планирую прочитать».
✓ Импорт/экспорт библиотеки.
Рекомендуется использовать следующие сущности:
• User — владелец библиотеки
• Book — книга (автор, название, жанр)
• Status — статус (читаю, прочитана)
• Tag — метки (например, "фантастика")
• ListManager — списки "Прочитанные", "Хочу прочитать".
Рекомендуемые паттерны:
• Strategy — разные способы сортировки.
• Composite — списки книг как коллекции.

10) Приложение для трекинга полезных привычек (1 чел.)

✓ Создание привычек с расписанием.
✓ Отметка выполнения за день.
✓ Визуализация прогресса (стрики, календарь)..
✓ Напоминания.
✓ Поиск по названию, частоте.
Рекомендуется использовать следующие сущности:
• User — пользователь
• Habit — привычка (название, периодичность)
• Record — отметка выполнения (дата)
• CalendarView — календарь выполнений
• StreakManager — подсчёт "серий"
Рекомендуемые паттерны:
• Observer — обновление календаря при изменении.
• Strategy — разные стратегии визуализации (тепловая карта, график).

11) Редактор целей и достижений в разных сферах жизни (1 чел.)

✓ Создание целей с подцелями.
✓ Отслеживание прогресса.
✓ Напоминания о сроках.
✓ Поиск по сроку, сфере (работа, здоровье).

✓ Экспорт данных в файл.

Рекомендуется использовать следующие сущности:

- User — владелец целей
- Goal — цель (название, срок)
- Subgoal — подцель
- Progress — прогресс (в %)
- Reminder — напоминание о дедлайне

Рекомендуемые паттерны:

• Composite — цели и подцели как дерево.

• Observer — обновление прогресса при изменении подцелей.

12) Система учета домашних рецептов (1 чел.)

✓ Добавление рецептов: название, фото, ингредиенты, шаги приготовления, время.

✓ Поиск по ингредиентам, времени приготовления.

✓ Фильтрация по типу блюда (первое, десерт и т.д.).

✓ Возможность пометки «избранное».

✓ Импорт/экспорт списка рецептов.

Рекомендуется использовать следующие сущности:

- User — повар
- Recipe — рецепт (название, ингредиенты, шаги)
- Ingredient — ингредиент
- Category — тип блюда (первое, десерт)
- FavouritesManager — избранное

Рекомендуемые паттерны:

- Strategy — разные фильтры.
- Factory — создание рецепта по шаблону.

13) Приложение для изучения музыкальных аккордов и гамм (2 чел.)

✓ Создание и редактирование базы аккордов и гамм для гитары и фортепиано.

✓ Просмотр схем аккордов и гамм с возможностью масштабирования и поворота (для гитары — вид схемы ладов, для фортепиано — клавиатура).

✓ Поиск по названию, тональности, типу (мажор, минор), инструменту.

✓ Фильтрация по сложности (начинающий, средний, продвинутый).

✓ Режимы обучения: «Угадай аккорд по схеме», «Найди схему по названию».

✓ Возможность создавать кастомные наборы (например, "Аккорды для блюза", "Минорные гаммы") — функция для учителя.

✓ Импорт/экспорт базы данных (в JSON) — для обмена наборами между пользователями.

✓ Сохранение прогресса при выходе, загрузка при запуске.

Рекомендуется использовать следующие сущности:

- User — учитель (автор)/ученик (разный интерфейс)

• Chord — аккорд (название, инструмент, схема)
• Scale — гамма
• Instrument — тип (гитара, фортепиано)
• TrainingMode — режим "угадай по схеме"

Рекомендуемые паттерны:

• Factory — создание схемы под инструмент.
• Strategy — выбор режима обучения.

14) Работа компьютерного клуба (2 чел.)

✓ Регистрация клиента.
✓ Выбор времени посещения и запись клиента.
✓ Подтверждение записи клиента администратором.
✓ Учет времени сессии.
✓ Расчет стоимости по тарифу.
✓ Поиск по клиенту, дате, ПК.
✓ Отчёты: доход, загруженность.
✓ Сохранение истории.

Рекомендуется использовать следующие сущности:

• User — клиент/администратор (разный интерфейс)
• Session — сессия (время входа/выхода)
• Rate — тариф (руб/час)
• Computer — номер ПК
• ReportManager — доход, загруженность

Рекомендуемые паттерны:

• Strategy — разные тарифы.
• Observer — обновление стоимости в реальном времени.

15) Приложение для составления расписания дополнительных занятий детей (2 чел.)

✓ Создание расписания для каждого ребенка.
✓ Повторяющиеся занятия с автозаполнением.
✓ Поиск по дню, учителю, предмету.
✓ Фильтрация по направлению развития ребенка.
✓ Импорт/экспорт расписания.
✓ Визуализация по дням недели, календарь.
✓ Напоминания о занятиях для родителя.

Рекомендуется использовать следующие сущности:

• User — родитель/ребенок (разный интерфейс)
• Lesson — занятие (название, время, место, педагог)
• Schedule — расписание по дням
• Teacher — учитель
• ExportManager — экспорт расписания

Рекомендуемые паттерны:

• Composite — расписание как коллекция дней
• Command — отмена изменений

16) Организация работы агентства по сдаче автомобилей в аренду (2 чел.)

✓ Учет автомобилей: марка, модель, статус.
✓ Заказ аренды.
✓ Оформление аренды: клиент, даты, стоимость.
✓ Расчет по дням, штрафы за просрочку.
✓ Поиск по марке, дате, клиенту.
✓ Отчёты по доходу.
✓ Импорт/экспорт данных.

Рекомендуется использовать следующие сущности:

- User — клиент/администратор (разный интерфейс)
- Car — автомобиль (марка, модель, статус)
- Rental — аренда (даты, стоимость)
- Fine — штраф за просрочку
- ReportManager — доход, загруженность

Рекомендуемые паттерны:

- Strategy — расчёт стоимости (день/неделя/штраф)
- Observer — обновление статуса авто при возврате

17) Адаптивный комплекс тренировок для мозга (2 чел.)

✓ Несколько типов мини-игр: память, внимание, логика.
✓ Уровни сложности в зависимости от набранных баллов.
✓ Поиск по типу игры.
✓ Визуализация прогресса по разным направлениям тренировок.
✓ Импорт/экспорт статистики за указанный период.

Рекомендуется использовать следующие сущности:

- User — игрок
- Game — тип игры (память, логика)
- Level — уровень сложности
- Result — результат прохождения
- ProgressTracker — статистика

Рекомендуемые паттерны:

- Factory — создание игры по типу
- Strategy — выбор алгоритма сложности на основе прогресса.

18) Создание и симуляция жизни фантастических питомцев (2 чел.)

✓ Создание питомца по критериям (скорость изменения потребностей, внешний вид).
✓ Выбор комнаты для питомца.
✓ Управление несколькими питомцами.
✓ Потребности питомца: голод, сон, настроение, здоровье.
✓ Действия: кормить, играть, спать.
✓ Монеты за игры для покупки еды и предметов в комнату.
✓ Прогресс возраста со сменой аватара с течением времени.
✓ Сохранение состояния игры.

Рекомендуется использовать следующие сущности:

- User — владелец
- Pet — питомец (голод, сон, настроение)

• Action — действие (кормить, играть)
• Need — потребность
• SaveManager — сохранение состояния
Рекомендуемые паттерны:
• Observer — обновление состояния при действии
• Command — отмена действия

19) Пополняемая энциклопедия полезных лесных трав (2 чел.)

✓ Добавление, редактирование растений.
✓ Поиск и фильтрация по сезону, типу.
✓ Просмотр и добавление фото и описаний.
✓ Список избранного для пользователя.
✓ Импорт/экспорт базы.
✓ Визуализация по сезонам.
Рекомендуется использовать следующие сущности:
• User — ботаник/пользователь (разный интерфейс)
• Plant — растение (название, фото, описание)
• Family — семейство
• Season — сезон сбора
• SearchManager — поиск по названию, семейству
Рекомендуемые паттерны:
• Strategy — разные стратегии фильтрации.
• Composite — группы растений по сезону

20) Учет и контроль работы в туристическом агентстве (2 чел.)

✓ Управление турами: направление, даты, стоимость, вместимость.
✓ Бронирование тура клиентом с подтверждением.
✓ Оформление оплаты и отслеживание статуса.
✓ Поиск туров по направлению, дате, цене.
✓ Фильтрация бронирований по клиенту, статусу.
✓ Формирование отчётов: доход, популярные направления.
✓ Импорт/экспорт данных о турах и клиентах.
✓ Сохранение состояния при выходе.
Рекомендуется использовать следующие сущности:
• User — клиент / менеджер агентства (разный интерфейс)
• Tour — тур (направление, даты, стоимость, количество мест)
• Booking — бронирование (клиент, тур, статус оплаты)
• Payment — оплата (сумма, дата, статус)
• ReportManager — формирование отчётов
Рекомендуемые паттерны:
• Strategy — выбор стратегии расчёта стоимости (горящий тур, раннее бронирование).
• Observer — уведомление о заполнении тура или изменении статуса брони.

21) Адаптивный дневник питания и тренировок (1 чел.)

✓ Учёт приёмов пищи: продукты, порции, калории.
✓ Учёт тренировок: тип, длительность, сожжённые калории.

✓ Подсчёт суточного баланса калорий.
✓ Анализ прогресса: графики веса, активности, питания.
✓ Поиск записей по дате, типу (еда/тренировка).
✓ Фильтрация по цели (похудение, набор массы).
✓ Рекомендации на основе данных (упрощённые, по шаблонам).
✓ Импорт/экспорт дневника.
Рекомендуется использовать следующие сущности:
• User — пользователь (цель, вес, рост)
• Meal — приём пищи (продукты, калории)
• Workout — тренировка (тип, время, нагрузка)
• CalorieCounter — подсчёт калорий
• ProgressTracker — визуализация прогресса
Рекомендуемые паттерны:
• Strategy — выбор стратегии подсчёта нормы калорий (по полу, возрасту).
• Observer — обновление прогресса при добавлении записи.

22) Учет заказов и расходных материалов швейного ателье (2 чел.)

✓ Учёт заказов: клиент, изделие, срок выполнения, стоимость.
✓ Учёт тканей и фурнитуры: остатки, типы материалов.
✓ Списание материалов при выполнении заказа.
✓ Поиск заказов по клиенту, сроку, статусу.
✓ Фильтрация по типу изделия (платье, костюм).
✓ Отчёты: расход материалов, прибыль по заказам.
✓ Импорт/экспорт данных об остатках и заказах.
Рекомендуется использовать следующие сущности:
• User — закройщик / администратор (разный интерфейс)
• Order — заказ (клиент, изделие, срок, стоимость)
• Material — ткань или фурнитура (название, остаток)
• Supply — поставка материалов
• ReportManager — отчёты по расходам и прибыли

Рекомендуемые паттерны:
• Strategy — выбор стратегии расчёта стоимости пошивки.
• Observer — обновление остатков при списании материалов.

23) Организация работы свадебного агентства (2 чел.)

✓ Учёт заказов: клиенты, дата свадьбы, бюджет.
✓ Назначение услуг: фотограф, ведущий, декор, транспорт.
✓ Ведение чек-листа задач (до, во время, после свадьбы).
✓ Отслеживание оплат по частям.
✓ Поиск по дате, клиенту, статусу оплаты.
✓ Фильтрация по бюджету, типу услуги.
✓ Импорт/экспорт чек-листов и заказов.
Рекомендуется использовать следующие сущности:
• User — клиент / организатор (разный интерфейс)
• WeddingOrder — заказ свадьбы (дата, бюджет)

• Service — услуга (фото, ведущий, декор)
• TaskList — чек-лист задач
• PaymentTracker — отслеживание оплат
Рекомендуемые паттерны:
• Composite — чек-лист как иерархия задач.
• Observer — обновление статуса заказа при завершении задач.

24) Администрирование фитнес клуба (2 чел.)

✓ Учёт абонементов: тип (разовый, месячный), срок, стоимость.
✓ Отметка посещений клиентов.
✓ Напоминания о завершении абонемента.
✓ Поиск по клиенту, типу абонемента, дате.
✓ Фильтрация по статусу (активен, истёк).
✓ Отчёты: посещаемость, доход.
✓ Импорт/экспорт данных о клиентах и абонементах.
Рекомендуется использовать следующие сущности:
• User — клиент / администратор (разный интерфейс)
• Membership — абонемент (тип, срок, стоимость)
• Visit — посещение (дата, время)
• Trainer — тренер (опционально)
• ReportManager — отчёты по посещаемости
Рекомендуемые паттерны:
• Strategy — выбор стратегии начисления (разовый, групповой).
• Observer — уведомление о завершении абонемента.

25) Редактируемый календарь садовода (1 чел.)

✓ Планирование работ: посадка, полив, обрезка, удобрение.
✓ Привязка к культурам (огурцы, помидоры) и сезонам.
✓ Добавление напоминаний о предстоящих работах.
✓ Цветовая маркировка по типу работ.
✓ Поиск по культуре, типу работы, дате.
✓ Фильтрация по сезону (весна, лето).
✓ Импорт/экспорт календаря.
✓ Сохранение при выходе.
Рекомендуется использовать следующие сущности:
• User — садовод
• GardenTask — задача (посадка, полив)
• Culture — культура (томат, картофель)
• Season — сезон (весна, лето)
• Reminder — напоминание о работе
Рекомендуемые паттерны:
• Strategy — выбор цветовой схемы по типу работы.
• Observer — обновление календаря при добавлении задачи.

26) Работа сервиса аренды техники (2 чел.)

✓ Учёт инвентаря: тип (дрель, генератор), статус, стоимость аренды.
✓ Оформление аренды: клиент, срок, залог.
✓ Расчёт стоимости по дням и типу техники.

✓ Поиск по типу техники, клиенту, дате.
✓ Фильтрация по статусу (в аренде, свободен).
✓ Отчёты: доход, загруженность инвентаря.
✓ Импорт/экспорт данных об оборудовании.
Рекомендуется использовать следующие сущности:
• User — клиент / администратор (разный интерфейс)
• Equipment — единица техники (тип, статус, цена в сутки)
• Rental — аренда (клиент, даты, залог)
• Rate — тариф (по типу техники)
• ReportManager — отчёты по доходу и использованию
Рекомендуемые паттерны:
• Strategy — расчёт стоимости (день/неделя/сезонная скидка).
• Observer — обновление статуса техники при оформлении аренды.

27) Создание собственного журнала комиксов (1 чел.)

✓ Создание страниц: панели, текст, изображения.
✓ Использование шаблонов разметки.
✓ Добавление текста и изображений в панели.
✓ Сохранение и загрузка журнала.
✓ Поиск по названию страницы, дате.
✓ Фильтрация по статусу (черновик, готово).
✓ Импорт/экспорт журнала (с сохранением структуры).
Рекомендуется использовать следующие сущности:
• User — автор комикса
• ComicPage — страница комикса (панели, текст)
• Panel — панель (изображение, текст)
• Template — шаблон разметки
• SaveManager — сохранение и загрузка проекта
Рекомендуемые паттерны:
• Composite — страница как коллекция панелей.
• Command — отмена/повтор действия (например, удаление панели).

28) Учет занятий спортивной школы для разных направлений (3 чел.)

✓ Создание и редактирование групп администратором: направление (футбол, танцы, плавание), тренер, расписание (дни, время, зал).
✓ Добавление/удаление тренеров, учеников.
✓ Учёт посещаемости учеников тренерами.
✓ Просмотр расписания своих групп тренером.
✓ Поиск по ученику, группе, дате.
✓ Просмотр своего расписания и прогресса учениками.
✓ Фильтрация по направлению, тренеру.
✓ Отчёты: посещаемость, прогресс учеников.
✓ Импорт/экспорт расписания и посещаемости.
Рекомендуется использовать следующие сущности:
• User — ученик / тренер / администратор (разные интерфейсы)
• Group — группа (направление, тренер, расписание)

• Attendance — посещаемость (ученик, дата, статус)
• ScheduleManager — управление расписанием, проверка конфликтов
• Trainer — тренер (ФИО, направление)
• ReportManager — отчёты по посещаемости
Рекомендуемые паттерны:
• Strategy — выбор стратегии подсчёта прогресса (по посещениям).
• Observer — обновление статуса ученика при отметке посещения.

29) Альбом фотографий и достижений ребенка от 0 до 14 лет (1 чел.)

✓ Добавление фото, описания, даты события.
✓ Организация по возрастным этапам (0–1, 1–3, 3–7 и т.д.).
✓ Хронология развития: рост, речь, навыки.
✓ Поиск по возрасту, событию, дате.
✓ Фильтрация по типу события (день рождения, первый шаг).
✓ Импорт/экспорт альбома (с сохранением структуры).
✓ Сохранение при выходе.
Рекомендуется использовать следующие сущности:
• User — родитель / опекун
• Photo — фотография (файл, описание, дата)
• Milestone — важное событие (первый шаг, день рождения)
• AgeGroup — возрастная группа
• Timeline — хронология развития
Рекомендуемые паттерны:
• Composite — альбом как коллекция возрастных групп.
• Observer — обновление хронологии при добавлении события.

30) Приложение для изучения скорочтения (1 чел.)

✓ Режим показа текста по словам/фразам с заданной скоростью.
✓ Настройка скорости (слов в минуту).
✓ Тестирование скорости чтения с замером времени.
✓ Статистика прогресса: средняя скорость, динамика.
✓ Поиск по названию текста, теме, уровню сложности.
✓ Фильтрация по жанру (новости, художественное).
✓ Импорт/экспорт текстов и статистики.
Рекомендуется использовать следующие сущности:
• User — читатель
• Text — текст (название, содержание, тема, сложность)
• SpeedSetting — настройка скорости (слов/мин)
• TestSession — сессия тестирования
• ProgressTracker — статистика прогресса
Рекомендуемые паттерны:
• Strategy — выбор стратегии отображения (по слову, по фразе).
• Observer — обновление статистики после завершения теста.

3.2 Требования к отчету курсовой работы

Отчет по курсовой работе выполняется каждым студентом в отдельности. Для совместных тем в отчет вносится работа по той части программного продукта, которой занимался студент, следуя общему плану работы.

Оформление. Титульный лист должен содержать: название учебного заведения, наименование кафедры, название дисциплины и тему работы, ФИО и группу учащегося, ФИО и должность руководителя; город и год написания курсовой работы. Текст должен быть набран в текстовом редакторе Microsoft Word или текстовом редакторе свободно распространяемого ПО, сохранен в формате docx или совместимом с docx формате. Основной шрифт – Times New Roman, 14 кегль, межстрочный интервал – одинарный. Каждый абзац с новой красной строки, отступ 1,25 см. Отступы на полях документа: слева 3 см, справа 1 см, верхний и нижний — по 2 см. Выравнивание текста по ширине.

Все рисунки и таблицы должны быть озаглавлены и иметь ссылку на них в тексте. Подрисуночные подписи должны располагаться по центру рисунка без точки в конце. Надпись над таблицей выравнивается по левому краю без точки в конце. Листы должны быть пронумерованы, начиная со второй страницы (1 – титульный лист, 2 – оборот титульного листа). Страницы нумеруются снизу, строго по центру. В текст должна быть установлена автоматическая расстановка переносов.

В содержание включают номера и наименования разделов и подразделов с указанием номеров листов (страниц). В содержание также включают все приложения, вошедшие в данный документ, с указанием номера листа (страницы).

Введение. Обзор предметной области по заданной теме в объеме одной страницы. Описать какие существуют приложения по данной и смежной тематике и их возможности, какие видите перспективы по этой теме.

Основная часть. Проектирование интерфейса программного продукта должно соответствовать следующему плану:

1. Анализ задач и пользователей.

1.1. Анализ целевой аудитории:
- общее описание аудитории;
- структура аудитории, выделение сегментов;
- предпочтения целевой аудитории;
- социально-демографические характеристики целевой аудитории;
- тенденции по составу и предпочтениям целевой аудитории на ближайшее будущее.

1.2. Профиль пользователя.

Составить обобщенный профиль представителя целевой аудитории:
- демографические и психографические особенности пользователя;
- задачи и цели пользователя;
- рабочая среда пользователя;
- термины пользователя

1.3. Общие требования и задачи.

2. Выбор репрезентативных задач.

2.1. Список задач пользователя.

2.2. Подробное описание задач в терминах пользователя.

2.3. Список необходимых материалов для выполнения задач.

3. Заимствование.

- Найти и описать 1-2 приложения, в которых реализованы задачи по вашей теме либо похожие задачи.
- Описать существующие низкоуровневые интерфейсные решения, которые можно применить в вашей теме.
- Проанализировать в соответствии с профилем пользователя, какие интерфейсные решения можно заимствовать и выбрать наиболее подходящие для вашей работы;

4. Черновое описание интерфейса.

- Выбрать и обосновать какие модели будут использоваться и для каких частей описания интерфейса.
- Описать интерфейс с помощью выбранных моделей:

а) для матрицы прямого манипулирования:
- описать используемые метафоры;
- в заданной предметной области определить список объектов, их типы и представления,
- построить матрицу прямого манипулирования объектами;

б) для модели действий пользователя:
- определить роли пользователей,
- составить список действий,
- построить модель действий пользователя;

в) для структурной модели:
- описать используемые элементы интерфейса,
- составить схемы экранных кадров,
- графически представить структурную модель;

г) для последовательно-динамической модели:
- определить роли пользователей,
- составить список объектов интерфейса,
- привести таблицу динамического взаимодействия.

5. Анализ интерфейса.

Проанализировать описанный интерфейс и оптимизировать его на основе полученных результатов.

1) Выбрать три главные репрезентативные задачи для исследования.
2) Провести CWT анализ выбранных репрезентативных задач и оформить таблицей вида:

Формулировка задачи			
Действие	История	Проблемы	Пути решения

3) Провести GOMS анализ выбранных задач.
4) Исследовать производительность экранных форм по методу Дж. Раскина касательно выбранных репрезентативных задач.

6. Прототип интерфейса.

- Привести прототипы всех экранных форм, включая сообщения.

- Произвести оценку трех основных экранных форм с помощью законов Фитса и Хика.
- Оптимизировать разработанный вариант интерфейса на основе полученной оценки, а также золотых правил построения интерфейсов.

7. **Тестирование с пользователями.**
   - Описать двоих реальных пользователей либо персонажей в соответствии с профилем пользователя.
   - Провести use-case для всех репрезентативных задач для двоих пользователей по форме:

Действие пользователя	Реакция системы	Реакция и ожидания пользователя

   - Сделать выводы по возможным улучшениям интерфейса на основании тестирования.

8. **Программная реализация.**
   - Сформулировать основные цели разработки.
   - Составить требования к программному продукту.
   - Сформировать перечень и характеристики исходных данных.
   - Описать получаемые результаты и способы их представления.
   - Перечислить использованные классы Qt и их назначение.
   - Привести скриншоты основных окон приложения.
   - Описать базу данных или структуру сохраняемых файлов.
   - Выложить исходный программный код на любой облачный ресурс и привести ссылку в отчете.

Заключение. Сделать выводы о проделанной работе, описать возникшие проблемы в процессе выполнения работы и пути их решения.

Список используемых источников. Перечислить как минимум 3 источника литературы, используемых при подготовке по ГОСТ Р 7.0.5–2008.

3.3 Контрольные вопросы для защиты курсовой работы

1. Перечислите этапы проектирования интерфейса пользователя согласно проблемно-центрированному подходу.
2. Что включает в себя анализ целевой аудитории на этапе проектирования интерфейса, и зачем он необходим?
3. Какова цель этапа заимствования при разработке интерфейса?
4. Какие модели используются для чернового описания интерфейса, и что они отражают в структуре будущего приложения?
5. Что представляет собой CWT-анализ и какую роль он играет при оптимизации пользовательского интерфейса?
6. В чем заключается особенность использования законов Фитса и Хика при оценке пользовательского интерфейса?
7. Что такое GOMS-анализ? Как он применяется для оценки эффективности интерфейса?

8. Какие паттерны проектирования вы использовали в своей работе? Объясните, зачем они нужны и как реализованы в вашем приложении.
9. В чём разница между паттернами Factory и Strategy? Приведите примеры их использования в вашем проекте.
10. Как паттерн Observer помогает в реализации многопользовательского или многокомпонентного интерфейса?
11. Как паттерн Composite может быть полезен при работе с иерархическими структурами (например, альбомы, цели, чек-листы)?
12. Какие данные в вашем приложении сохраняются локально и в каком формате (JSON, XML, SQLite)? Обоснуйте выбор формата.
13. Как реализована поддержка нескольких пользователей в вашем приложении? Какие данные разделяются, а какие — персональны?
14. Как реализованы функции импорта и экспорта данных? Какие ограничения или особенности возникли при их реализации?

СПИСОК ЛИТЕРАТУРЫ

1) Qt, Tools for Each Stage of Software Development Lifecycle [Электронный ресурс]. URL: https://www.qt.io (дата обращения: 5.05.2025).

2) Алексеев Е.Р., Злобин Г.Г., Костюк Д.А. [и др.] Программирование на языке C++ в среде Qt Creator : Курс лекций. — Москва : Интуит НОУ, 2016. — 715 с. [Электронный ресурс]. URL: https://book.ru/book/918128 (дата обращения: 05.05.2025).

3) Шлее М. Qt 5.10. Профессиональное программирование на C++. БХВ-Петербург, 2018. 1052 с.

4) Зубкова Т.М. Технология разработки программного обеспечения: учебное пособие. Оренбургский гос. ун-т.- Оренбург: ОГУ, 2017. – 468 с.

5) Корнипаев И.Требования для программного обеспечения: рекомендации по сбору и документированию. М.: Издательство «Книга по Требованию», 2013. – 118 с.

6) Мерзлякова Е. Ю. Визуальное программирование и человеко-машинное взаимодействие : практикум / Сибирский государственный университет телекоммуникаций и информатики. Новосибирск, 2022. 49 с.