Лабораторная работа №3 Ручное тестирование

Цели:

- Изучение методологии тестирования
- Развитие навыков анализа требований
- Создание структурированных тест-кейсов
- Проверка соответствия требованиям
- Выявление ошибок и дефектов
- Развитие структурированного мышления
- Подготовка к реальной практике
- Работа с документацией
- Понимание жизненного цикла разработки
- Развитие практических навыков

Задание:

- Начните с детального изучения функциональных требований приложения. Разделите их на отдельные функции или возможности.
- Определите, какие сценарии использования (use cases) нужно протестировать.
- Создайте список тестовых сценариев
- Сохраните все тест-кейсы в едином формате (например, в таблице).
- Выполните все тест-кейсы и продемонстрируйте преподавателю.

Дополнительное задание (опционально)

- Начните с детального изучения нефункциональных требований приложения. Разделите их на отдельные функции или возможности.
- Определите, какие сценарии использования (use cases) нужно протестировать.
- Создайте список тестовых сценариев
- Сохраните все тест-кейсы в едином формате (например, в таблице).
- Выполните все тест-кейсы.

Пример:

Тема: **Программа для управления товарами в магазине**

Программа должна предоставлять следующие функции:
1. Добавление товара.
2. Удаление товара.
3. Проверка наличия товара.
4. Отображение списка товаров.
5. Проверка граничных значений (например, цена не может быть отрицательной, количество не может быть отрицательным и т.д.).

Функциональные требования

Функциональные требования описывают, что система должна делать.

1. **Управление товарами:**
    - Система должна позволять создавать товар с указанием названия, цены и количества.
    - Система должна предоставлять возможность проверки доступности товара (наличие на складе).
    - Система должна отображать информацию о товаре, включая название, цену и количество.
    - Система должна позволять удалять товар.
2. **Проверка доступности товара:**
    - Система должна возвращать true, если количество товара больше 0, и false, если количество равно 0.
3. **Отображение информации:**
    - Система должна предоставлять возможность вывода строкового представления товара, включающего название, цену и количество.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**
    - Система должна обрабатывать создание товара и проверку его доступности за время не более 1 секунды.
    - Система должна отображать информацию о товаре за время не более 2 секунд.
2. **Удобство использования:**
    - Интерфейс системы должен быть интуитивно понятным, с четкими сообщениями об успешных или ошибочных действиях.
    - Система должна поддерживать возможность отмены последнего действия (например, создания товара).
3. **Надежность:**

    - Система должна корректно обрабатывать ошибки, такие как попытка создания товара с некорректными данными (например, отрицательная цена или количество).
    - Система должна сохранять целостность данных при сбоях (например, не допускать потери данных при внезапном завершении работы).
4. **Масштабируемость:**
    - Система должна поддерживать работу с большим количеством товаров (до 10 000 элементов).
5. **Безопасность:**
    - Система должна обеспечивать защиту данных от несанкционированного доступа (например, шифрование данных при хранении).
6. **Совместимость:**
    - Система должна поддерживать работу на операционных системах Windows, macOS и Linux.
    - Система должна быть совместима с браузерами Chrome, Firefox и Edge (если используется веб-интерфейс).
7. **Локализация:**
    - Система должна поддерживать несколько языков интерфейса (например, русский и английский).

Детальное изучения функциональных требований приложения. Разделение их на отдельные функции или возможности, а также определение того, что нужно проверить.

1. Управление товарами
    1.1. Создание товара:
        - Проверить, что система позволяет создавать товар с указанием названия, цены и количества.
        - Проверить, что система корректно обрабатывает корректные данные (название, цена > 0, количество >= 0).
        - Проверить, что система возвращает ошибку при попытке создания товара с некорректными данными (например, отрицательная цена или количество).
    1.2. Удаление товара:
        - Проверить, что система позволяет удалить выбранный товар.
        - Проверить, что система информирует о необходимости выбора товара перед удалением.
2. Проверка доступности товара:

- Проверить, что система возвращает true, если количество товара больше 0.
- Проверить, что система возвращает false, если количество товара равно 0.

3. Отображение информации о товаре:
   - Проверить, что система корректно отображает информацию о товаре, включая название, цену и количество.
   - Проверить, что строковое представление товара соответствует формату: "Товар: [название], Цена: [цена], Количество: [количество]".

Создание списка тестовых сценариев. Требования к тестированию.

1. Создание товара
   - **TC-001:** Создание товара с корректными данными.
   - **TC-002:** Создание товара с нулевой ценой.
   - **TC-003:** Создание товара с отрицательной ценой.
   - **TC-004:** Создание товара с отрицательным количеством.
   - **TC-005:** Создание товара с пустым названием.

2. Удаление товара
   - **TC-006:** Удаление товара из списка..
   - **TC-007:** Попытка удаления товара без выбора.

3. Проверка доступности товара
   - **TC-008:** Проверка доступности товара (количество > 0).
   - **TC-009:** Проверка доступности товара (количество = 0).
   - **TC-010:** Проверка доступности товара без выбора.

4. Отображение информации
   - **TC-011:** Проверка строкового представления товара.
   - **TC-012:** Проверка корректности отображаемых данных (название, цена, количество).

Оформленные тест-кейсы для приложения "ProductManager" в соответствии со стандартом ISTQB

1. **Информация о документе**

- Наименование документа: Тест-кейсы для приложения "ProductManager"
- Дата создания: [Указать дату]

- Версия: 1.0
- Автор: [Указать имя автора]

2. Введение

Данный документ содержит описание тест-кейсов для приложения "ProductManager", предназначенных для проверки его функциональности. Тест-кейсы разработаны в соответствии с требованиями ISTQB и включают:

- **Идентификатор тест-кейса** (уникальный номер).
- **Название тест-кейса**.
- **Предусловия** (условия, которые должны быть выполнены перед выполнением теста).
- **Шаги выполнения** (последовательность действий для выполнения теста).
- **Ожидаемый результат** (что должно произойти после выполнения шагов).
- **Фактический результат** (заполняется после выполнения теста).
- **Статус** (Pass(***Пройден***)/Fail(***Не пройден***)).

3. Список использованных стандартов

- ISTQB (International Software Testing Qualifications Board) — международная система квалификации тестировщиков программного обеспечения, унифицирующая стандарты и подходы к тестированию

4. Описание окружения

- Операционная система: [Указать ОС]
- Версия приложения: [Указать версию]
- Прочее ПО: [Указать дополнительное ПО, если требуется]

5. Тест-кейсы

5.1. Добавление товара с корректными данными

- **Идентификатор:** TC-001
- **Название:** Добавление товара с корректными данными.
- **Предусловия:** Нет.
- **Шаги выполнения:**
    1. Ввести в поле "Наименование товара" значение: "Ноутбук".
    2. Ввести в поле "Цена, руб." значение: 50000.
    3. Ввести в поле "Количество" значение: 10.
    4. Нажать кнопку "Добавить".
- **Ожидаемый результат:** Товар успешно добавлен, отображается в списке товаров, появляется сообщение "Товар добавлен!".
- **Фактический результат:**
- **Статус:**

5.2. Добавление товара с нулевой ценой

- **Идентификатор:** ТС-002
- **Название:** Добавление товара с нулевой ценой.
- **Предусловия:** Нет.
- **Шаги выполнения:**
    1. Ввести в поле "Наименование товара" значение: "Ноутбук".
    2. Ввести в поле "Цена, руб." значение: 0.
    3. Ввести в поле "Количество" значение: 10.
    4. Нажать кнопку "Добавить".
- **Ожидаемый результат:** Появляется сообщение об ошибке: "Цена должна быть положительной!", товар не добавлен.
- **Фактический результат:**
- **Статус:**

5.3. Добавление товара с отрицательной ценой

- **Идентификатор:** ТС-003
- **Название:** Добавление товара с отрицательной ценой.
- **Предусловия:** Нет.
- **Шаги выполнения:**
    1. Ввести в поле "Наименование товара" значение: "Ноутбук".
    2. Ввести в поле "Цена, руб." значение: -50000.
    3. Ввести в поле "Количество" значение: 10.
    4. Нажать кнопку "Добавить".
- **Ожидаемый результат:** Появляется сообщение об ошибке: "Цена должна быть положительной!", товар не добавлен.
- **Фактический результат:**
- **Статус:**

5.4. Добавление товара с отрицательным количеством

- **Идентификатор:** ТС-004
- **Название:** Добавление товара с отрицательным количеством.
- **Предусловия:** Нет.
- **Шаги выполнения:**
    1. Ввести в поле "Наименование товара" значение: "Ноутбук".
    2. Ввести в поле "Цена, руб." значение: 50000.
    3. Ввести в поле "Количество" значение: -10.
    4. Нажать кнопку "Добавить".
- **Ожидаемый результат:** Появляется сообщение об ошибке: "Количество не может быть отрицательным!", товар не добавлен.
- **Фактический результат:**

- **Статус:**

5.5. Добавление товара с пустым наименованием

- **Идентификатор:** TC-005
- **Название:** Добавление товара с пустым наименованием.
- **Предусловия:** Нет.
- **Шаги выполнения:**
    1. Оставить поле "Наименование товара" пустым.
    2. Ввести в поле "Цена, руб." значение: 50000.
    3. Ввести в поле "Количество" значение: 10.
    4. Нажать кнопку "Добавить".
- **Ожидаемый результат:** Появляется сообщение об ошибке, товар не добавлен.
- **Фактический результат:**
- **Статус:**

5.6. Удаление товара

- **Идентификатор:** TC-006
- **Название:** Удаление товара из списка.
- **Предусловия:** В списке товаров есть хотя бы один товар.
- **Шаги выполнения:**
    1. Выбрать товар в списке.
    2. Нажать кнопку "Удалить".
- **Ожидаемый результат:** Товар удален из списка, появляется сообщение "Товар удален!".
- **Фактический результат:**
- **Статус:**

5.7. Удаление товара без выбора

- **Идентификатор:** TC-007
- **Название:** Попытка удаления товара без выбора.
- **Предусловия:** В списке товаров есть хотя бы один товар.
- **Шаги выполнения:**
    1. Не выбирать товар в списке.
    2. Нажать кнопку "Удалить".
- **Ожидаемый результат:** Появляется сообщение об ошибке: "Выберите товар для удаления!".
- **Фактический результат:**
- **Статус:**

5.8. Проверка доступности товара (количество > 0)

- **Идентификатор:** TC-008
- **Название:** Проверка доступности товара, если количество больше 0.
- **Предусловия:** В списке товаров есть товар с количеством > 0.
- **Шаги выполнения:**
    1. Выбрать товар с количеством > 0 в списке.
    2. Нажать кнопку "Проверить".
- **Ожидаемый результат:** Появляется сообщение: "Товар доступен!".
- **Фактический результат:**
- **Статус:**

5.9. Проверка доступности товара (количество = 0)

- **Идентификатор:** TC-009
- **Название:** Проверка доступности товара, если количество равно 0.
- **Предусловия:** В списке товаров есть товар с количеством = 0.
- **Шаги выполнения:**
    1. Выбрать товар с количеством = 0 в списке.
    2. Нажать кнопку "Проверить".
- **Ожидаемый результат:** Появляется сообщение: "Товар недоступен!".
- **Фактический результат:**
- **Статус:**

5.10. Проверка доступности товара без выбора

- **Идентификатор:** TC-010
- **Название:** Попытка проверки доступности товара без выбора.
- **Предусловия:** В списке товаров есть хотя бы один товар.
- **Шаги выполнения:**
    1. Не выбирать товар в списке.
    2. Нажать кнопку "Проверить".
- **Ожидаемый результат:** Появляется сообщение об ошибке: "Выберите товар для проверки!".
- **Фактический результат:**
- **Статус:**

5.11. Проверка строкового представления товара

- **Идентификатор:** TC-011
- **Название:** Проверка строкового представления товара.
- **Предусловия:** В списке товаров есть хотя бы один товар.
- **Шаги выполнения:**
    1. Посмотреть в списке представление товара.
- **Ожидаемый результат:** Строковое представление товара соответствует формату: "Товар: [название], Цена: [цена], Количество: [количество]".

- **Фактический результат:**
- **Статус:**

5.12. Проверка корректности отображаемых данных

- **Идентификатор:** TC-011
- **Название:** Проверка корректности отображаемых данных.
- **Предусловия:** нет.
- **Шаги выполнения:**
    1. Ввести в поле "Наименование товара" значение: "Ноутбук".
    2. Ввести в поле "Цена, руб." значение: 50000.
    3. Ввести в поле "Количество" значение: 10.
    4. Нажать кнопку "Добавить".
- **Ожидаемый результат:** Добавленный товар,отображается в списке товаров, и соответствует введенным значения.
- **Фактический результат:**
- **Статус:**

---

6. Ниже приведены примеры тест-кейсов, которые охватывают как функциональные, так и нефункциональные требования.

6.1. Производительность: Добавление товара

- **Идентификатор:** TC-NF-001
- **Название:** Проверка времени добавления товара.
- **Предусловия:** Нет.
- **Шаги выполнения:**
    1. Ввести в поле "Наименование товара" значение: "Ноутбук".
    2. Ввести в поле "Цена, руб." значение: 50000.
    3. Ввести в поле "Количество" значение: 10.
    4. Нажать кнопку "Добавить".
- **Ожидаемый результат:** Товар добавлен за время не более 1 секунды.
- **Фактический результат:**
- **Статус:**

6.2. Производительность: Удаление товара

- **Идентификатор:** TC-NF-002
- **Название:** Проверка времени удаления товара.
- **Предусловия:** В списке товаров есть хотя бы один товар.
- **Шаги выполнения:**
    1. Выбрать товар в списке.
    2. Нажать кнопку "Удалить".

- **Ожидаемый результат:** Товар удален за время не более 1 секунды.
- **Фактический результат:**
- **Статус:**

6.3. Удобство использования: Корректность сообщений об ошибках

- **Идентификатор:** TC-NF-003
- **Название:** Проверка корректности сообщений об ошибках.
- **Предусловия:** Нет.
- **Шаги выполнения:**
    1. Ввести в поле "Цена, руб." значение: -50000.
    2. Нажать кнопку "Добавить".
- **Ожидаемый результат:** Появляется сообщение: "Цена должна быть положительной!".
- **Фактический результат:**
- **Статус:**

6.4. Удобство использования: Отображение списка товаров

- **Идентификатор:** TC-NF-004
- **Название:** Проверка отображения списка товаров.
- **Предусловия:** В списке товаров есть хотя бы один товар.
- **Шаги выполнения:**
    1. Открыть главное окно программы.
- **Ожидаемый результат:** Список товаров отображается корректно, без искажений.
- **Фактический результат:**
- **Статус:**

6.5. Надежность: Обработка исключений при добавлении товара

- **Идентификатор:** TC-NF-005
- **Название:** Проверка обработки исключений при добавлении товара.
- **Предусловия:** Нет.
- **Шаги выполнения:**
    1. Ввести в поле "Наименование товара" значение: "Ноутбук".
    2. Ввести в поле "Цена, руб." значение: "abc" (некорректное значение).
    3. Нажать кнопку "Добавить".
- **Ожидаемый результат:** Появляется сообщение об ошибке, программа не завершает работу аварийно.
- **Фактический результат:**
- **Статус:**

6.6. Надежность: Обработка исключений при удалении товара

- **Идентификатор:** TC-NF-006
- **Название:** Проверка обработки исключений при удалении товара.
- **Предусловия:** В списке товаров есть хотя бы один товар.
- **Шаги выполнения:**
    1. Выбрать товар в списке.
    2. Имитировать сбой (например, отключение базы данных).
    3. Нажать кнопку "Удалить".
- **Ожидаемый результат:** Появляется сообщение об ошибке, программа не завершает работу аварийно.
- **Фактический результат:**
- **Статус:**

6.7. Безопасность: Защита от SQL-инъекций

- **Идентификатор:** TC-NF-007
- **Название:** Проверка защиты от SQL-инъекций.
- **Предусловия:** Нет.
- **Шаги выполнения:**
    1. Ввести в поле "Наименование товара" значение: "'; DROP TABLE products; --.
    2. Ввести в поле "Цена, руб." значение: 50000.
    3. Ввести в поле "Количество" значение: 10.
    4. Нажать кнопку "Добавить".
- **Ожидаемый результат:** Товар добавлен, данные защищены от SQL-инъекций.
- **Фактический результат:**
- **Статус:**

6.8. Совместимость: Работа на разных операционных системах

- **Идентификатор:** TC-NF-008
- **Название:** Проверка работы программы на Windows, macOS и Linux.
- **Предусловия:** Программа установлена на Windows, macOS и Linux.
- **Шаги выполнения:**
    1. Запустить программу на каждой из операционных систем.
    2. Добавить товар с параметрами: Name = "Ноутбук", Price = 50000, Quantity = 10.
- **Ожидаемый результат:** Программа работает корректно на всех операционных системах.
- **Фактический результат:**
- **Статус:**

6.9. Локализация: Поддержка нескольких языков

- **Идентификатор:** TC-NF-009
- **Название:** Проверка поддержки нескольких языков интерфейса.
- **Предусловия:** Программа поддерживает русский и английский языки.
- **Шаги выполнения:**
    1. Изменить язык интерфейса на русский.
    2. Проверить отображение текста на кнопках и полях ввода.
    3. Изменить язык интерфейса на английский.
    4. Проверить отображение текста на кнопках и полях ввода.
- **Ожидаемый результат:** Интерфейс корректно отображается на обоих языках.
- **Фактический результат:**
- **Статус:**

6.10. Логирование: Запись действий пользователя

- **Идентификатор:** TC-NF-010
- **Название:** Проверка записи действий пользователя в журнал.
- **Предусловия:** Включено логирование.
- **Шаги выполнения:**
    1. Добавить товар с параметрами: Name = "Ноутбук", Price = 50000, Quantity = 10.
    2. Проверить журнал логов.
- **Ожидаемый результат:** В журнале записано действие "Товар добавлен: Ноутбук".
- **Фактический результат:**
- **Статус:**

6.11. Резервное копирование: Автоматическое создание резервных копий

- **Идентификатор:** TC-NF-011
- **Название:** Проверка автоматического создания резервных копий.
- **Предусловия:** Настроено автоматическое резервное копирование.
- **Шаги выполнения:**
    1. Добавить товар с параметрами: Name = "Ноутбук", Price = 50000, Quantity = 10.
    2. Дождаться выполнения резервного копирования.
    3. Проверить наличие резервной копии.
- **Ожидаемый результат:** Резервная копия создана и содержит добавленный товар.
- **Фактический результат:**
- **Статус:**

Варианты

1. Управление банковским счётом

Описание: Создать программу для управления банковским счётом, которая позволяет создавать новый счёт, пополнять его, снимать средства, проверять баланс и проверять граничные значения (например, отрицательные суммы, превышение лимита).

Функциональные требования

1. **Создание банковского счёта**
    - Система должна позволять создавать банковский счёт с указанием имени владельца и начального баланса.
    - Начальный баланс должен быть неотрицательным числом.
2. **Получение информации о владельце счёта**
    - Система должна предоставлять возможность получения имени владельца счёта.
3. **Получение текущего баланса**
    - Система должна предоставлять возможность получения текущего баланса на счёте.
4. **Пополнение счёта**
    - Система должна позволять пополнять счёт на указанную сумму.
    - Сумма пополнения должна быть неотрицательной.
    - Сумма пополнения не должна превышать максимально допустимую сумму (1 000 000).
    - В случае попытки пополнения на отрицательную сумму или сумму, превышающую максимально допустимую, система должна выбрасывать исключение с соответствующим сообщением.
5. **Снятие средств со счёта**
    - Система должна позволять снимать средства со счёта на указанную сумму.
    - Сумма снятия должна быть неотрицательной.
    - Сумма снятия не должна превышать максимально допустимую сумму (1 000 000).
    - Сумма снятия не должна превышать текущий баланс на счёте.
    - В случае попытки снятия отрицательной суммы, суммы, превышающей максимально допустимую, или суммы, превышающей текущий баланс, система должна выбрасывать исключение с соответствующим сообщением.

Нефункциональные требования

1. **Производительность**

    - Операции пополнения и снятия средств должны выполняться за время, не превышающее 1 секунду.
2. **Надёжность**
    - Система должна гарантировать корректное выполнение операций пополнения и снятия средств без потери данных.
    - В случае возникновения исключений система должна сохранять целостность данных (например, баланс не должен изменяться при некорректных операциях).
3. **Безопасность**
    - Система должна предотвращать несанкционированный доступ к данным счёта (например, к имени владельца и балансу).
4. **Удобство использования**
    - Сообщения об ошибках должны быть понятными и информативными для пользователя.

2. Конвертер валют

Описание: Создать программу для конвертации валют. Пользователь может выбирать валюты для конвертации, вводить сумму и получать результат. Программа должна использовать актуальные курсы валют и проверять корректность ввода.

Функциональные требования:

1. **Конвертация валют:**
    - Система должна корректно конвертировать сумму из одной валюты в другую на основе заданных курсов.
    - Поддерживаемые пары валют:
        - USD → EUR
        - EUR → USD
    - Для других пар валют система должна выбрасывать исключение NotSupportedException.
2. **Обработка отрицательных значений:**
    - Если сумма для конвертации (amount) меньше нуля, система должна выбрасывать исключение ArgumentException с сообщением "Сумма не может быть отрицательной."
3. **Курсы валют:**
    - Курсы валют должны быть заданы как константы:
        - USD → EUR: 0.88
        - EUR → USD: 1.12

Нефункциональные требования:

1. **Производительность:**
    - Конвертация валют должна выполняться за время, не превышающее 100 мс для суммы до 1 000 000 единиц.
2. **Надежность:**
    - Система должна корректно обрабатывать все возможные входные данные, включая граничные случаи (например, сумма = 0).
3. **Удобство использования:**
    - Сообщения об ошибках должны быть понятными и информативными для пользователя.

3. Управление задачами

Описание: Создать программу для управления задачами. Пользователь может добавлять, удалять, редактировать задачи и отмечать их как выполненные. Программа должна сохранять задачи в файл и загружать их при запуске.

Функциональные требования

1. **Управление задачами**
    - 1.1. Система должна позволять добавлять новую задачу с описанием.
        - 1.1.1. Если описание задачи пустое или null, система должна выбрасывать исключение ArgumentException.
    - 1.2. Система должна позволять удалять задачу по указанному индексу.
        - 1.2.1. Если индекс задачи выходит за пределы допустимого диапазона, система должна выбрасывать исключение IndexOutOfRangeException.
    - 1.3. Система должна позволять изменять статус выполнения задачи (выполнена/не выполнена) по указанному индексу.
        - 1.3.1. Если индекс задачи выходит за пределы допустимого диапазона, система должна выбрасывать исключение IndexOutOfRangeException.
2. **Сохранение и загрузка задач**
    - 2.1. Система должна сохранять список задач в файл tasks.txt после каждого изменения (добавление, удаление, изменение статуса задачи).
        - 2.1.1. Каждая задача должна сохраняться в формате: {статус выполнения}|{описание задачи}.
    - 2.2. Система должна загружать список задач из файла tasks.txt при инициализации.
        - 2.2.1. Если файл tasks.txt не существует, система должна инициализировать пустой список задач.
3. **Хранение задач**

- 3.1. Система должна хранить список задач в памяти в виде коллекции объектов типа Task.
- 3.2. Каждая задача должна содержать:
  - 3.2.1. Описание задачи (строка).
  - 3.2.2. Статус выполнения задачи (булево значение).

Нефункциональные требования

1. **Производительность**
   - 1.1. Система должна загружать задачи из файла tasks.txt за время, не превышающее 1 секунду для 1000 задач.
   - 1.2. Система должна сохранять задачи в файл tasks.txt за время, не превышающее 1 секунду для 1000 задач.
2. **Надежность**
   - 2.1. Система должна корректно обрабатывать ошибки при работе с файловой системой (например, отсутствие прав на запись или чтение файла).
   - 2.2. Система должна сохранять целостность данных при сбоях (например, при завершении работы во время записи в файл).
3. **Удобство использования**
   - 3.1. Система должна предоставлять четкие сообщения об ошибках при некорректных действиях пользователя (например, пустое описание задачи или неверный индекс).
4. **Совместимость**
   - 4.1. Система должна работать на платформах, поддерживающих .NET Core или .NET Framework.
   - 4.2. Файл tasks.txt должен быть читаемым и редактируемым в текстовом редакторе.
5. **Безопасность**
   - 5.1. Система не должна допускать утечки данных (например, описание задач должно храниться только в файле tasks.txt и в памяти программы).

4. Календарь событий

Описание: Создать программу, которая позволяет добавлять, удалять и редактировать события в календаре. События должны сохраняться в файл и загружаться при запуске.

Функциональные требования

1. **Управление событиями**
   - 1.1. Система должна позволять добавлять новое событие.

        - 1.1.1. Если добавляемое событие равно null, система должна выбрасывать исключение ArgumentNullException.
    - 1.2. Система должна позволять удалять существующее событие.
        - 1.2.1. Если удаляемое событие равно null, система должна выбрасывать исключение ArgumentNullException.
2. **Сохранение и загрузка событий**
    - 2.1. Система должна сохранять список событий в файл events.txt после каждого изменения (добавление или удаление события).
        - 2.1.1. Каждое событие должно сохраняться в формате: {дата}|{описание}.
        - 2.1.2. Дата должна сохраняться в формате yyyy-MM-dd.
    - 2.2. Система должна загружать список событий из файла events.txt при инициализации.
        - 2.2.1. Если файл events.txt не существует, система должна инициализировать пустой список событий.
        - 2.2.2. Если строка в файле имеет некорректный формат, она должна быть пропущена.
3. **Хранение событий**
    - 3.1. Система должна хранить список событий в памяти в виде коллекции объектов типа Event.
    - 3.2. Каждое событие должно содержать:
        - 3.2.1. Дату события (тип DateTime).
        - 3.2.2. Описание события (строка).

Нефункциональные требования

1. **Производительность**
    - 1.1. Система должна загружать события из файла events.txt за время, не превышающее 1 секунду для 1000 событий.
    - 1.2. Система должна сохранять события в файл events.txt за время, не превышающее 1 секунду для 1000 событий.
2. **Надежность**
    - 2.1. Система должна корректно обрабатывать ошибки при работе с файловой системой (например, отсутствие прав на запись или чтение файла).
    - 2.2. Система должна сохранять целостность данных при сбоях (например, при завершении работы во время записи в файл).
3. **Удобство использования**
    - 3.1. Система должна предоставлять четкие сообщения об ошибках при некорректных действиях пользователя (например, попытка добавить null событие).
4. **Совместимость**

    - 4.1. Система должна работать на платформах, поддерживающих .NET Core или .NET Framework.
    - 4.2. Файл events.txt должен быть читаемым и редактируемым в текстовом редакторе.
5. **Безопасность**
    - 5.1. Система не должна допускать утечки данных (например, описание событий должно храниться только в файле events.txt и в памяти программы).

5. Управление книгами в библиотеке

Описание: Создать программу для управления книгами в библиотеке. Пользователь может добавлять, удалять, редактировать книги и искать по автору или названию. Программа должна сохранять данные в файл.

Функциональные требования

1. **Управление книгами**
    - 1.1. Система должна позволять добавлять новую книгу.
        - 1.1.1. Если добавляемая книга равна null, система должна выбрасывать исключение ArgumentNullException.
    - 1.2. Система должна позволять удалять существующую книгу.
        - 1.2.1. Если удаляемая книга равна null, система должна выбрасывать исключение ArgumentNullException.
2. **Поиск книг**
    - 2.1. Система должна позволять искать книги по запросу.
        - 2.1.1. Поиск должен осуществляться по полям Author и Title.
        - 2.1.2. Поиск должен возвращать список книг, содержащих запрос в любом из указанных полей.
3. **Сохранение и загрузка книг**
    - 3.1. Система должна сохранять список книг в файл books.txt после каждого изменения (добавление или удаление книги).
        - 3.1.1. Каждая книга должна сохраняться в формате: {автор}|{название}|{год}.
    - 3.2. Система должна загружать список книг из файла books.txt при инициализации.
        - 3.2.1. Если файл books.txt не существует, система должна инициализировать пустой список книг.
        - 3.2.2. Если строка в файле имеет некорректный формат, она должна быть пропущена.
4. **Хранение книг**
    - 4.1. Система должна хранить список книг в памяти в виде коллекции объектов типа Book.
    - 4.2. Каждая книга должна содержать:

        - 4.2.1. Автора книги (строка).
        - 4.2.2. Название книги (строка).
        - 4.2.3. Год издания книги (строка).

Нефункциональные требования

1. **Производительность**
    - 1.1. Система должна загружать книги из файла books.txt за время, не превышающее 1 секунду для 1000 книг.
    - 1.2. Система должна сохранять книги в файл books.txt за время, не превышающее 1 секунду для 1000 книг.
    - 1.3. Поиск книг должен выполняться за время, не превышающее 0,5 секунды для 1000 книг.
2. **Надежность**
    - 2.1. Система должна корректно обрабатывать ошибки при работе с файловой системой (например, отсутствие прав на запись или чтение файла).
    - 2.2. Система должна сохранять целостность данных при сбоях (например, при завершении работы во время записи в файл).
3. **Удобство использования**
    - 3.1. Система должна предоставлять четкие сообщения об ошибках при некорректных действиях пользователя (например, попытка добавить null книгу).
4. **Совместимость**
    - 4.1. Система должна работать на платформах, поддерживающих .NET Core или .NET Framework.
    - 4.2. Файл books.txt должен быть читаемым и редактируемым в текстовом редакторе.
5. **Безопасность**
    - 5.1. Система не должна допускать утечки данных (например, информация о книгах должна храниться только в файле books.txt и в памяти программы).

6. Управление заметками

Описание: Создать программу для управления заметками. Пользователь может добавлять, удалять, редактировать заметки и сохранять их в файл.

Функциональные требования

1. **Управление заметками**
    - 1.1. Система должна позволять добавлять новую заметку.
        - 1.1.1. Если добавляемая заметка равна null, система должна выбрасывать исключение ArgumentNullException.

    - 1.2. Система должна позволять удалять существующую заметку.
        - 1.2.1. Если удаляемая заметка равна null, система должна выбрасывать исключение ArgumentNullException.
2. **Сохранение и загрузка заметок**
    - 2.1. Система должна сохранять список заметок в файл notes.txt после каждого изменения (добавление или удаление заметки).
        - 2.1.1. Каждая заметка должна сохраняться в формате: {заголовок}|{содержание}|{дата и время}.
        - 2.1.2. Дата и время должны сохраняться в формате yyyy-MM-dd HH:mm:ss.
    - 2.2. Система должна загружать список заметок из файла notes.txt при инициализации.
        - 2.2.1. Если файл notes.txt не существует, система должна инициализировать пустой список заметок.
        - 2.2.2. Если строка в файле имеет некорректный формат, она должна быть пропущена.
3. **Хранение заметок**
    - 3.1. Система должна хранить список заметок в памяти в виде коллекции объектов типа Note.
    - 3.2. Каждая заметка должна содержать:
        - 3.2.1. Заголовок заметки (строка).
        - 3.2.2. Содержание заметки (строка).
        - 3.2.3. Дата и время создания заметки (тип DateTime).

Нефункциональные требования

1. **Производительность**
    - 1.1. Система должна загружать заметки из файла notes.txt за время, не превышающее 1 секунду для 1000 заметок.
    - 1.2. Система должна сохранять заметки в файл notes.txt за время, не превышающее 1 секунду для 1000 заметок.
2. **Надежность**
    - 2.1. Система должна корректно обрабатывать ошибки при работе с файловой системой (например, отсутствие прав на запись или чтение файла).
    - 2.2. Система должна сохранять целостность данных при сбоях (например, при завершении работы во время записи в файл).
3. **Удобство использования**
    - 3.1. Система должна предоставлять четкие сообщения об ошибках при некорректных действиях пользователя (например, попытка добавить null заметку).
4. **Совместимость**

    - 4.1. Система должна работать на платформах, поддерживающих .NET Core или .NET Framework.
    - 4.2. Файл notes.txt должен быть читаемым и редактируемым в текстовом редакторе.
5. **Безопасность**
    - 5.1. Система не должна допускать утечки данных (например, информация о заметках должна храниться только в файле notes.txt и в памяти программы).

7. Управление контактами

Описание: Создать программу для управления контактами. Пользователь может добавлять, удалять, редактировать контакты и искать по имени или номеру телефона. Программа должна сохранять контакты в файл.

Функциональные требования

1. **Управление контактами**
    - 1.1. Система должна позволять добавлять новый контакт.
        - 1.1.1. Если добавляемый контакт равен null, система должна выбрасывать исключение ArgumentNullException.
    - 1.2. Система должна позволять удалять существующий контакт.
        - 1.2.1. Если удаляемый контакт равен null, система должна выбрасывать исключение ArgumentNullException.
2. **Поиск контактов**
    - 2.1. Система должна позволять искать контакты по запросу.
        - 2.1.1. Поиск должен осуществляться по полям Name и PhoneNumber.
        - 2.1.2. Поиск должен возвращать список контактов, содержащих запрос в любом из указанных полей.
3. **Сохранение и загрузка контактов**
    - 3.1. Система должна сохранять список контактов в файл contacts.txt после каждого изменения (добавление или удаление контакта).
        - 3.1.1. Каждый контакт должен сохраняться в формате: {имя}|{номер телефона}.
    - 3.2. Система должна загружать список контактов из файла contacts.txt при инициализации.
        - 3.2.1. Если файл contacts.txt не существует, система должна инициализировать пустой список контактов.
        - 3.2.2. Если строка в файле имеет некорректный формат, она должна быть пропущена.
4. **Хранение контактов**

    - 4.1. Система должна хранить список контактов в памяти в виде коллекции объектов типа Contact.
    - 4.2. Каждый контакт должен содержать:
        - 4.2.1. Имя контакта (строка).
        - 4.2.2. Номер телефона контакта (строка).

Нефункциональные требования

1. **Производительность**
    - 1.1. Система должна загружать контакты из файла contacts.txt за время, не превышающее 1 секунду для 1000 контактов.
    - 1.2. Система должна сохранять контакты в файл contacts.txt за время, не превышающее 1 секунду для 1000 контактов.
    - 1.3. Поиск контактов должен выполняться за время, не превышающее 0,5 секунды для 1000 контактов.
2. **Надежность**
    - 2.1. Система должна корректно обрабатывать ошибки при работе с файловой системой (например, отсутствие прав на запись или чтение файла).
    - 2.2. Система должна сохранять целостность данных при сбоях (например, при завершении работы во время записи в файл).
3. **Удобство использования**
    - 3.1. Система должна предоставлять четкие сообщения об ошибках при некорректных действиях пользователя (например, попытка добавить null контакт).
4. **Совместимость**
    - 4.1. Система должна работать на платформах, поддерживающих .NET Core или .NET Framework.
    - 4.2. Файл contacts.txt должен быть читаемым и редактируемым в текстовом редакторе.
5. **Безопасность**
    - 5.1. Система не должна допускать утечки данных (например, информация о контактах должна храниться только в файле contacts.txt и в памяти программы).

8. Управление покупками

Описание: Создать программу для управления покупками. Пользователь может добавлять, удалять, редактировать покупки и сортировать их по категориям. Программа должна сохранять данные в файл.

Функциональные требования

1. **Управление покупками**

- 1.1. Система должна позволять добавлять новую покупку.
  - 1.1.1. Если добавляемая покупка равна null, система должна выбрасывать исключение ArgumentNullException.
- 1.2. Система должна позволять удалять существующую покупку.
  - 1.2.1. Если удаляемая покупка равна null, система должна выбрасывать исключение ArgumentNullException.

2. **Фильтрация покупок по категории**
   - 2.1. Система должна позволять получать список покупок по указанной категории.
     - 2.1.1. Категория должна быть одной из предопределенных: Продукты, Техника, Одежда, Прочее.
3. **Сохранение и загрузка покупок**
   - 3.1. Система должна сохранять список покупок в файл purchases.txt после каждого изменения (добавление или удаление покупки).
     - 3.1.1. Каждая покупка должна сохраняться в формате: {название}|{цена}|{категория}|{дата и время}.
     - 3.1.2. Категория должна сохраняться как целое число, соответствующее значению перечисления Category.
     - 3.1.3. Дата и время должны сохраняться в формате yyyy-MM-dd HH:mm:ss.
   - 3.2. Система должна загружать список покупок из файла purchases.txt при инициализации.
     - 3.2.1. Если файл purchases.txt не существует, система должна инициализировать пустой список покупок.
     - 3.2.2. Если строка в файле имеет некорректный формат, она должна быть пропущена.
4. **Хранение покупок**
   - 4.1. Система должна хранить список покупок в памяти в виде коллекции объектов типа Purchase.
   - 4.2. Каждая покупка должна содержать:
     - 4.2.1. Название покупки (строка).
     - 4.2.2. Цену покупки (тип decimal).
     - 4.2.3. Категорию покупки (перечисление Category).
     - 4.2.4. Дата и время покупки (тип DateTime).

Нефункциональные требования

1. **Производительность**
   - 1.1. Система должна загружать покупки из файла purchases.txt за время, не превышающее 1 секунду для 1000 покупок.
   - 1.2. Система должна сохранять покупки в файл purchases.txt за время, не превышающее 1 секунду для 1000 покупок.

    - 1.3. Фильтрация покупок по категории должна выполняться за время, не превышающее 0,5 секунды для 1000 покупок.
2. **Надежность**
    - 2.1. Система должна корректно обрабатывать ошибки при работе с файловой системой (например, отсутствие прав на запись или чтение файла).
    - 2.2. Система должна сохранять целостность данных при сбоях (например, при завершении работы во время записи в файл).
3. **Удобство использования**
    - 3.1. Система должна предоставлять четкие сообщения об ошибках при некорректных действиях пользователя (например, попытка добавить null покупку).
4. **Совместимость**
    - 4.1. Система должна работать на платформах, поддерживающих .NET Core или .NET Framework.
    - 4.2. Файл purchases.txt должен быть читаемым и редактируемым в текстовом редакторе.
5. **Безопасность**
    - 5.1. Система не должна допускать утечки данных (например, информация о покупках должна храниться только в файле purchases.txt и в памяти программы).

9. Управление задачами с приоритетом

Описание: Создать программу для управления задачами с приоритетом. Пользователь может добавлять, удалять, редактировать задачи и сортировать их по приоритету. Программа должна сохранять задачи в файл.

Функциональные требования

1. **Управление задачами с приоритетами**
    - Система должна позволять создавать задачи с указанием описания, приоритета (Низкий, Средний, Высокий) и срока выполнения (Deadline).
    - Система должна сохранять состояние задачи (выполнена или не выполнена).
2. **Добавление задачи**
    - Система должна предоставлять возможность добавления новой задачи.
    - При попытке добавить задачу с пустыми или некорректными данными система должна выбрасывать исключение ArgumentNullException.
3. **Удаление задачи**

    - Система должна предоставлять возможность удаления существующей задачи.
    - При попытке удалить задачу с пустыми или некорректными данными система должна выбрасывать исключение ArgumentNullException.
4. **Изменение статуса задачи**
    - Система должна предоставлять возможность изменения статуса задачи (выполнена/не выполнена).
5. **Сортировка задач по приоритету**
    - Система должна предоставлять возможность сортировки задач по приоритету в порядке убывания (Высокий, Средний, Низкий).
6. **Сохранение и загрузка задач**
    - Система должна сохранять задачи в файл tasks.txt в формате: Описание|Приоритет|Статус|Deadline.
    - Система должна загружать задачи из файла tasks.txt при запуске.

Нефункциональные требования

1. **Производительность**
    - Система должна загружать задачи из файла tasks.txt за время, не превышающее 1 секунду при количестве задач до 1000.
2. **Надежность**
    - Система должна корректно обрабатывать ошибки при чтении/записи файла tasks.txt (например, отсутствие файла или некорректный формат данных).
3. **Удобство использования**
    - Система должна предоставлять понятный интерфейс для управления задачами (в контексте консольного или графического интерфейса).
4. **Совместимость**
    - Система должна поддерживать работу на платформах Windows, Linux и macOS.
5. **Безопасность**
    - Система должна обеспечивать целостность данных при сохранении и загрузке задач (например, проверка корректности формата данных).

10. Управление продажами

Описание: Создать программу для управления продажами. Пользователь может добавлять, удалять, редактировать продажи и генерировать отчёты. Программа должна сохранять данные в файл.

Функциональные требования

1. **Управление продажами**
    - Система должна позволять добавлять новую продажу с указанием:
        - Названия продукта (ProductName).
        - Цены продукта (Price).
        - Количества проданных единиц (Quantity).
        - Даты продажи (Date).
    - Система должна позволять удалять существующую продажу.
    - Система должна сохранять изменения в списке продаж после добавления или удаления продажи.
2. **Расчет выручки**
    - Система должна рассчитывать общую выручку (TotalRevenue) для каждой продажи как произведение цены (Price) на количество (Quantity).
    - Система должна рассчитывать общую выручку для всех продаж как сумму выручки по каждой продаже.
3. **Сохранение и загрузка данных**
    - Система должна сохранять данные о продажах в файл sales.txt в формате: **ProductName|Price|Quantity|Date**
    - Система должна загружать данные о продажах из файла sales.txt при запуске.
    - Система должна корректно обрабатывать файл sales.txt, если он существует, и игнорировать его, если он отсутствует.
4. **Валидация данных**
    - Система должна проверять, что добавляемая продажа не является null. В случае передачи null, система должна выбрасывать исключение ArgumentNullException.
    - Система должна корректно обрабатывать данные при загрузке из файла, игнорируя строки с некорректным форматом.

Нефункциональные требования

1. **Производительность**
    - Система должна загружать данные о продажах из файла sales.txt за время, не превышающее 1 секунду для файла размером до 10 000 строк.
    - Система должна сохранять данные о продажах в файл sales.txt за время, не превышающее 1 секунду для списка из 10 000 продаж.
2. **Надежность**
    - Система должна корректно работать при наличии или отсутствии файла sales.txt.
    - Система должна обрабатывать ошибки чтения/записи файла и не завершать работу аварийно.
3. **Удобство использования**

    - Система должна предоставлять возможность легко добавлять и удалять продажи через API.
4. **Совместимость**
    - Система должна поддерживать работу с файлами sales.txt в кодировке UTF-8.
5. **Безопасность**
    - Система должна обеспечивать целостность данных при сохранении и загрузке.
    - Система должна игнорировать строки в файле sales.txt, которые не соответствуют ожидаемому формату.

11. Управление инвентарём

Описание: Создать программу для управления инвентарём. Пользователь может добавлять, удалять, редактировать товары и отслеживать их количество. Программа должна сохранять данные в файл.

Функциональные требования

1. **Управление инвентарем**
    - Система должна позволять добавлять новый элемент инвентаря с указанием:
        - Названия элемента (Name).
        - Количества (Quantity).
        - Цены (Price).
        - Категории (Category).
    - Система должна позволять удалять существующий элемент инвентаря.
    - Система должна позволять обновлять количество элемента инвентаря.
2. **Сохранение и загрузка данных**
    - Система должна сохранять данные об инвентаре в файл inventory.txt в формате: **Name|Quantity|Price|Category**
    - Система должна загружать данные об инвентаре из файла inventory.txt при запуске.
    - Система должна корректно обрабатывать файл inventory.txt, если он существует, и игнорировать его, если он отсутствует.
3. **Валидация данных**
    - Система должна проверять, что добавляемый или обновляемый элемент инвентаря не является null. В случае передачи null, система должна выбрасывать исключение ArgumentNullException.
    - Система должна корректно обрабатывать данные при загрузке из файла, игнорируя строки с некорректным форматом.

Нефункциональные требования

1. **Производительность**
    - Система должна загружать данные об инвентаре из файла inventory.txt за время, не превышающее 1 секунду для файла размером до 10 000 строк.
    - Система должна сохранять данные об инвентаре в файл inventory.txt за время, не превышающее 1 секунду для списка из 10 000 элементов.
2. **Надежность**
    - Система должна корректно работать при наличии или отсутствии файла inventory.txt.
    - Система должна обрабатывать ошибки чтения/записи файла и не завершать работу аварийно.
3. **Удобство использования**
    - Система должна предоставлять возможность легко добавлять, удалять и обновлять элементы инвентаря через API.
4. **Совместимость**
    - Система должна поддерживать работу с файлами inventory.txt в кодировке UTF-8.
5. **Безопасность**
    - Система должна обеспечивать целостность данных при сохранении и загрузке.
    - Система должна игнорировать строки в файле inventory.txt, которые не соответствуют ожидаемому формату.

12. Управление заказами

Описание: Создать программу для управления заказами. Пользователь может добавлять, удалять, редактировать заказы и отслеживать их статус. Программа должна сохранять данные в файл.

Функциональные требования

1. **Управление заказами**
    - Система должна позволять добавлять новый заказ с указанием:
        - Имени клиента (CustomerName).
        - Описания заказа (Description).
        - Даты создания заказа (CreationDate).
    - Система должна автоматически устанавливать статус заказа как "Новый" при его создании.
    - Система должна позволять удалять существующий заказ.
    - Система должна позволять обновлять статус заказа на один из следующих:
        - Новый.

        - В обработке.
        - Завершён.
2. **Сохранение и загрузка данных**
    - Система должна сохранять данные о заказах в файл orders.txt в формате: **CustomerName|Description|Status|CreationDate**
    - Система должна загружать данные о заказах из файла orders.txt при запуске.
    - Система должна корректно обрабатывать файл orders.txt, если он существует, и игнорировать его, если он отсутствует.
3. **Валидация данных**
    - Система должна проверять, что добавляемый или обновляемый заказ не является null. В случае передачи null, система должна выбрасывать исключение ArgumentNullException.
    - Система должна корректно обрабатывать данные при загрузке из файла, игнорируя строки с некорректным форматом.

Нефункциональные требования

1. **Производительность**
    - Система должна загружать данные о заказах из файла orders.txt за время, не превышающее 1 секунду для файла размером до 10 000 строк.
    - Система должна сохранять данные о заказах в файл orders.txt за время, не превышающее 1 секунду для списка из 10 000 заказов.
2. **Надежность**
    - Система должна корректно работать при наличии или отсутствии файла orders.txt.
    - Система должна обрабатывать ошибки чтения/записи файла и не завершать работу аварийно.
3. **Удобство использования**
    - Система должна предоставлять возможность легко добавлять, удалять и обновлять заказы через API.
4. **Совместимость**
    - Система должна поддерживать работу с файлами orders.txt в кодировке UTF-8.
5. **Безопасность**
    - Система должна обеспечивать целостность данных при сохранении и загрузке.
    - Система должна игнорировать строки в файле orders.txt, которые не соответствуют ожидаемому формату.

13. Управление сотрудниками

Описание: Создать программу для управления сотрудниками. Пользователь может добавлять, удалять, редактировать данные сотрудников и отслеживать их отпуска. Программа должна сохранять данные в файл.

Функциональные требования

1. **Управление сотрудниками**
    - Система должна позволять добавлять нового сотрудника с указанием:
        - Имени сотрудника (Name).
        - Должности сотрудника (Position).
        - Даты приема на работу (HireDate).
    - Система должна позволять удалять существующего сотрудника.
    - Система должна позволять обновлять данные о сотруднике, включая:
        - Начало отпуска (VacationStart).
        - Конец отпуска (VacationEnd).
2. **Управление отпусками**
    - Система должна позволять устанавливать даты начала и окончания отпуска для сотрудника.
    - Система должна определять, находится ли сотрудник в отпуске в текущий момент времени, на основе дат VacationStart и VacationEnd.
3. **Сохранение и загрузка данных**
    - Система должна сохранять данные о сотрудниках в файл employees.txt в формате:
      Name|Position|HireDate|VacationStart|VacationEnd
    - Система должна загружать данные о сотрудниках из файла employees.txt при запуске.
    - Система должна корректно обрабатывать файл employees.txt, если он существует, и игнорировать его, если он отсутствует.
4. **Валидация данных**
    - Система должна проверять, что добавляемый или обновляемый сотрудник не является null. В случае передачи null, система должна выбрасывать исключение ArgumentNullException.
    - Система должна корректно обрабатывать данные при загрузке из файла, игнорируя строки с некорректным форматом.

Нефункциональные требования

1. **Производительность**

    - Система должна загружать данные о сотрудниках из файла employees.txt за время, не превышающее 1 секунду для файла размером до 10 000 строк.
    - Система должна сохранять данные о сотрудниках в файл employees.txt за время, не превышающее 1 секунду для списка из 10 000 сотрудников.
2. **Надежность**
    - Система должна корректно работать при наличии или отсутствии файла employees.txt.
    - Система должна обрабатывать ошибки чтения/записи файла и не завершать работу аварийно.
3. **Удобство использования**
    - Система должна предоставлять возможность легко добавлять, удалять и обновлять данные о сотрудниках через API.
4. **Совместимость**
    - Система должна поддерживать работу с файлами employees.txt в кодировке UTF-8.
5. **Безопасность**
    - Система должна обеспечивать целостность данных при сохранении и загрузке.
    - Система должна игнорировать строки в файле employees.txt, которые не соответствуют ожидаемому формату.

14. Управление проектами

Описание: Создать программу для управления проектами. Пользователь может добавлять, удалять, редактировать проекты и отслеживать их прогресс. Программа должна сохранять данные в файл.

Функциональные требования

1. **Управление проектами**
    - Система должна позволять создавать новый проект с указанием:
        - Названия проекта (Name).
        - Описания проекта (Description).
        - Даты начала проекта (StartDate).
        - Даты окончания проекта (EndDate).
    - Система должна инициализировать прогресс проекта (Progress) значением 0 при создании.
2. **Обновление прогресса проекта**
    - Система должна позволять обновлять прогресс проекта (Progress) на значение от 0 до 100.
    - Система должна выбрасывать исключение ArgumentOutOfRangeException, если значение прогресса выходит за пределы допустимого диапазона (0–100).

3. **Добавление и удаление проектов**
    - Система должна позволять добавлять новый проект в список проектов.
    - Система должна позволять удалять проект из списка проектов.
    - Система должна выбрасывать исключение ArgumentNullException, если передается null вместо объекта проекта.
4. **Сохранение и загрузка проектов**
    - Система должна сохранять список проектов в файл projects.txt в формате: **Название|Описание|Дата начала|Дата окончания|Прогресс**
    - Система должна загружать список проектов из файла projects.txt при инициализации.

Нефункциональные требования

1. **Производительность**
    - Система должна загружать список проектов из файла за время, не превышающее 1 секунду при количестве проектов до 1000.
2. **Надежность**
    - Система должна корректно обрабатывать ошибки при загрузке и сохранении файла (например, отсутствие файла или некорректный формат данных).
3. **Удобство использования**
    - Система должна предоставлять понятные сообщения об ошибках при некорректных действиях пользователя (например, ввод недопустимого значения прогресса).

15. Управление клиентами

Описание: Создать программу для управления клиентами. Пользователь может добавлять, удалять, редактировать данные клиентов и искать по различным критериям. Программа должна сохранять данные в файл.

Функциональные требования

1. **Управление клиентами**
    - Система должна позволять создавать нового клиента с указанием:
        - Имени клиента (Name).
        - Электронной почты клиента (Email).
        - Телефона клиента (Phone).
        - Адреса клиента (Address).
2. **Добавление и удаление клиентов**

    - Система должна позволять добавлять нового клиента в список клиентов.
    - Система должна позволять удалять клиента из списка клиентов.
    - Система должна выбрасывать исключение ArgumentNullException, если передается null вместо объекта клиента.
3. **Поиск клиентов**
    - Система должна позволять осуществлять поиск клиентов по:
        - Имени (Name).
        - Электронной почте (Email).
        - Телефону (Phone).
        - Адресу (Address).
    - Система должна возвращать список клиентов, соответствующих критериям поиска.
4. **Сохранение и загрузка клиентов**
    - Система должна сохранять список клиентов в файл clients.txt в формате: **Имя|Электронная почта|Телефон|Адрес**
    - Система должна загружать список клиентов из файла clients.txt при инициализации.

Нефункциональные требования

1. **Производительность**
    - Система должна загружать список клиентов из файла за время, не превышающее 1 секунду при количестве клиентов до 1000.
    - Система должна выполнять поиск клиентов за время, не превышающее 500 миллисекунд при количестве клиентов до 1000.
2. **Надежность**
    - Система должна корректно обрабатывать ошибки при загрузке и сохранении файла (например, отсутствие файла или некорректный формат данных).
3. **Удобство использования**
    - Система должна предоставлять понятные сообщения об ошибках при некорректных действиях пользователя (например, передача null вместо объекта клиента).

16. Управление доставкой

Описание: Создать программу для управления доставкой. Пользователь может добавлять, удалять, редактировать доставки и отслеживать их статус. Программа должна сохранять данные в файл.

1. Функциональные требования

1.1. **Управление доставками**

- Система должна позволять создавать новые доставки с указанием имени клиента, адреса и даты доставки.
- Система должна автоматически устанавливать статус доставки "Новый" при создании новой доставки.

1.2. **Обновление статуса доставки**
- Система должна позволять изменять статус доставки на один из следующих: "Новый", "В_пути", "Доставлен".

1.3. **Добавление и удаление доставок**
- Система должна позволять добавлять новые доставки в список.
- Система должна позволять удалять доставки из списка.

1.4. **Сохранение и загрузка данных**
- Система должна сохранять данные о доставках в файл "deliveries.txt" после каждого изменения (добавление, удаление, обновление статуса).
- Система должна загружать данные о доставках из файла "deliveries.txt" при запуске.

1.5. **Обработка ошибок**
- Система должна выбрасывать исключение ArgumentNullException, если передается null-объект доставки при добавлении, удалении или обновлении статуса.

---

2. Нефункциональные требования

2.1. **Производительность**
- Система должна загружать данные о доставках из файла менее чем за 1 секунду при наличии до 1000 записей.

2.2. **Надежность**
- Система должна корректно обрабатывать ошибки чтения/записи файла "deliveries.txt".

2.3. **Удобство использования**
- Система должна предоставлять четкие сообщения об ошибках в случае некорректных данных.

---

3. Требования к данным

3.1. **Формат данных**
- Данные о доставках должны сохраняться в файл "deliveries.txt" в формате:
  ИмяКлиента|Адрес|ДатаДоставки|Статус
  Пример: Иван Иванов|ул. Ленина, 10|2023-10-15|0

3.2. **Типы данных**
- Имя клиента: строка (string).
- Адрес: строка (string).
- Дата доставки: дата в формате "yyyy-MM-dd".

- Статус доставки: целое число, соответствующее значению перечисления DeliveryStatus.

17. Управление отчётами

Описание: Создать программу для управления отчётами. Пользователь может генерировать, просматривать, редактировать и сохранять отчёты. Программа должна сохранять данные в файл.

Функциональные требования

1. **Управление отчетами**
    - Система должна позволять создавать новый отчет с указанием заголовка (Title), содержимого (Content) и даты создания (CreationDate).
    - Система должна сохранять созданные отчеты в файл reports.txt в формате: Title|Content|CreationDate.
    - Система должна загружать отчеты из файла reports.txt при инициализации объекта ReportManager.
2. **Добавление отчета**
    - Система должна предоставлять возможность добавления нового отчета с помощью метода AddReport.
    - При попытке добавления отчета со значением null система должна выбрасывать исключение ArgumentNullException.
3. **Удаление отчета**
    - Система должна предоставлять возможность удаления существующего отчета с помощью метода RemoveReport.
    - При попытке удаления отчета со значением null система должна выбрасывать исключение ArgumentNullException.
4. **Обновление отчета**
    - Система должна предоставлять возможность обновления заголовка и содержимого существующего отчета с помощью метода UpdateReport.
    - При попытке обновления отчета со значением null система должна выбрасывать исключение ArgumentNullException.
5. **Сохранение отчетов**
    - Система должна автоматически сохранять все изменения в отчетах (добавление, удаление, обновление) в файл reports.txt.
6. **Загрузка отчетов**
    - Система должна автоматически загружать все отчеты из файла reports.txt при создании объекта ReportManager.

Нефункциональные требования

1. **Производительность**
    - Загрузка отчетов из файла reports.txt должна выполняться за время, не превышающее 1 секунду для 1000 отчетов.
    - Сохранение отчетов в файл reports.txt должно выполняться за время, не превышающее 1 секунду для 1000 отчетов.
2. **Надежность**
    - Система должна корректно обрабатывать отсутствие файла reports.txt при загрузке отчетов (не должна выбрасывать исключение, если файл отсутствует).
    - Система должна корректно обрабатывать некорректный формат данных в файле reports.txt (пропускать строки, которые не соответствуют ожидаемому формату).
3. **Удобство использования**
    - Формат даты и времени в файле reports.txt должен быть читаемым и соответствовать стандарту yyyy-MM-dd HH:mm:ss.
4. **Безопасность**
    - Система должна обеспечивать целостность данных: при сбое во время сохранения файла reports.txt данные не должны быть потеряны или повреждены.
5. **Совместимость**
    - Система должна быть совместима с операционными системами, поддерживающими .NET (Windows, Linux, macOS).

18. Управление резервированием

Описание: Создать программу для управления резервированием. Пользователь может добавлять, удалять, редактировать резервы и отслеживать их статус. Программа должна сохранять данные в файл.

Функциональные требования

Функциональные требования описывают, что система должна делать. Они основаны на функциональности, реализованной в коде.

1. **Управление статусами бронирования:**
    - Система должна поддерживать три статуса бронирования: "Активно", "Отменено", "Завершено".
    - Система должна позволять изменять статус бронирования на любой из доступных статусов.
2. **Создание бронирования:**
    - Система должна позволять создавать новое бронирование с указанием имени клиента, времени начала и времени окончания.
    - При создании бронирования статус должен автоматически устанавливаться на "Активно".
3. **Добавление бронирования:**

    - Система должна позволять добавлять новое бронирование в список бронирований.
    - При добавлении бронирования система должна сохранять данные в файл.
4. **Удаление бронирования:**
    - Система должна позволять удалять существующее бронирование из списка.
    - При удалении бронирования система должна обновлять данные в файле.
5. **Обновление статуса бронирования:**
    - Система должна позволять изменять статус существующего бронирования.
    - При изменении статуса система должна сохранять обновленные данные в файл.
6. **Загрузка данных:**
    - Система должна загружать список бронирований из файла при запуске.
    - Система должна корректно обрабатывать данные из файла, включая имя клиента, время начала, время окончания и статус.
7. **Сохранение данных:**
    - Система должна сохранять все изменения в списке бронирований в файл.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, надежности, удобства использования и т.д.

1. **Производительность:**
    - Система должна загружать список бронирований из файла за время, не превышающее 1 секунду при количестве записей до 1000.
    - Система должна сохранять изменения в файл за время, не превышающее 500 мс.
2. **Надежность:**
    - Система должна корректно обрабатывать ошибки при чтении и записи файла (например, отсутствие файла или некорректный формат данных).
    - Система должна гарантировать целостность данных при сбоях (например, использовать временные файлы для записи данных).
3. **Удобство использования:**

    - Система должна предоставлять понятные сообщения об ошибках при некорректных входных данных (например, неверный формат времени).
    - Система должна поддерживать возможность расширения списка статусов бронирования без изменения основной логики.
4. **Совместимость:**
    - Система должна поддерживать работу на платформах Windows, Linux и macOS.
    - Система должна поддерживать кодировку UTF-8 для корректного отображения кириллицы.
5. **Масштабируемость:**
    - Система должна поддерживать увеличение количества бронирований до 10 000 записей без потери производительности.
6. **Безопасность:**
    - Система должна обеспечивать защиту данных бронирований от несанкционированного доступа (например, шифрование файла с данными).
    - Система должна проверять корректность входных данных для предотвращения инъекций или других атак.
7. **Локализация:**
    - Система должна поддерживать локализацию статусов бронирования на разные языки.

19. Управление бюджетом

Описание: Создать программу для управления бюджетом. Пользователь может добавлять, удалять, редактировать доходы и расходы, а также отслеживать общий бюджет. Программа должна сохранять данные в файл.

Функциональные требования:

1. **Управление транзакциями**:
    - ПО должно позволять добавлять новые транзакции с указанием описания, суммы, типа (доход/расход) и даты.
    - ПО должно позволять удалять существующие транзакции.
    - ПО должно позволять обновлять существующие транзакции (изменять описание, сумму, тип).
2. **Хранение данных**:
    - ПО должно сохранять все транзакции в файл transactions.txt в формате: Описание|Сумма|Тип|Дата.
    - ПО должно загружать транзакции из файла transactions.txt при запуске.
3. **Расчет бюджета**:

    - ПО должно автоматически рассчитывать общий бюджет на основе всех транзакций, где доходы увеличивают бюджет, а расходы уменьшают его.
4. **Валидация данных**:
    - ПО должно проверять, что добавляемая или изменяемая транзакция не является null.
    - ПО должно корректно обрабатывать ошибки при загрузке и сохранении транзакций (например, некорректный формат данных в файле).

Нефункциональные требования:

1. **Производительность**:
    - Загрузка транзакций из файла должна занимать не более 2 секунд при количестве транзакций до 10 000.
    - Расчет общего бюджета должен выполняться за время, не превышающее 1 секунду.
2. **Надежность**:
    - ПО должно корректно работать при отсутствии файла transactions.txt (создавать новый файл при первом сохранении).
    - ПО должно обрабатывать ошибки чтения/записи файла и уведомлять пользователя о проблемах.
3. **Удобство использования**:
    - Интерфейс ПО должен быть интуитивно понятным, с возможностью легко добавлять, удалять и редактировать транзакции.
    - ПО должно предоставлять пользователю информацию об общем бюджете в понятном формате.
4. **Совместимость**:
    - ПО должно поддерживать работу на операционных системах Windows, Linux и macOS.
    - Файл transactions.txt должен быть читаемым и редактируемым в стандартных текстовых редакторах.
5. **Безопасность**:
    - ПО должно обеспечивать целостность данных при сохранении и загрузке транзакций.
    - Доступ к файлу transactions.txt должен быть ограничен для предотвращения несанкционированного изменения данных.
6. **Масштабируемость**:
    - ПО должно поддерживать увеличение количества транзакций до 100 000 без значительного ухудшения производительности.
7. **Локализация**:

- ПО должно поддерживать русский язык для отображения типов транзакций (доход/расход).

20. Управление файлами и папками

Описание:
Создать программу для управления файлами и папками на компьютере. Функционал должен включать создание, удаление, переименование файлов и папок, а также сортировку по различным критериям.

Функциональные требования

Функциональные требования описывают, что система должна делать. Они основаны на функциональности, реализованной в коде.

1. **Загрузка файлов и папок**
   - ПО должно отображать список файлов и папок в текущей директории (по умолчанию — "Мои документы") в виде списка с колонками: имя, дата последнего изменения, размер (для файлов) или метка "Папка" (для директорий).
   - Папки должны выделяться синим цветом.
2. **Создание файла**
   - ПО должно предоставлять возможность создания нового текстового файла (с расширением .txt) в текущей директории через диалоговое окно.
   - После создания файла список файлов и папок должен обновляться автоматически.
3. **Удаление файла или папки**
   - ПО должно предоставлять возможность удаления выбранного файла или папки.
   - При попытке удаления без выбора элемента должно отображаться предупреждение.
   - После удаления список файлов и папок должен обновляться автоматически.
4. **Переименование файла или папки**
   - ПО должно предоставлять возможность переименования выбранного файла или папки через диалоговое окно.
   - При попытке переименования без выбора элемента должно отображаться предупреждение.
   - После переименования список файлов и папок должен обновляться автоматически.
5. **Сортировка по дате**
   - ПО должно предоставлять возможность сортировки файлов и папок по дате последнего изменения (от старых к новым).
   - После сортировки список должен обновляться.
6. **Сортировка по имени**

- ПО должно предоставлять возможность сортировки файлов и папок по имени (в алфавитном порядке).
- После сортировки список должен обновляться.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность**
   - Загрузка списка файлов и папок должна выполняться не более чем за 2 секунды для директории, содержащей до 1000 элементов.
   - Сортировка файлов и папок должна выполняться не более чем за 1 секунду для директории, содержащей до 1000 элементов.
2. **Удобство использования (Usability)**
   - Интерфейс должен быть интуитивно понятным, с четкими подсказками и сообщениями об ошибках.
   - Цветовое выделение папок должно быть легко различимым.
   - Диалоговые окна (например, для создания или переименования файла) должны быть простыми и понятными.
3. **Надежность (Reliability)**
   - ПО должно корректно обрабатывать ошибки, такие как отсутствие доступа к файлу или папке, и отображать соответствующие сообщения пользователю.
   - При удалении или переименовании файлов и папок должна быть обеспечена целостность данных.
4. **Совместимость (Compatibility)**
   - ПО должно работать на операционных системах Windows (версии 10 и выше).
   - ПО должно поддерживать файловые системы NTFS и FAT32.
5. **Безопасность (Security)**
   - ПО не должно предоставлять доступ к системным файлам и папкам, которые находятся за пределами текущей директории.
   - При удалении файлов и папок должна быть предусмотрена защита от случайного удаления (например, запрос подтверждения).
6. **Масштабируемость (Scalability)**
   - ПО должно корректно работать с директориями, содержащими до 10 000 файлов и папок, без значительного снижения производительности.
7. **Локализация (Localization)**
   - Текстовые сообщения и интерфейс должны поддерживать русский язык.

- Формат даты и времени должен соответствовать региональным настройкам пользователя.

21. Управление системой безопасности данных

Описание:
Создать программу для управления безопасностью данных, включая шифрование, дешифрование и проверку целостности данных.

Функциональные требования

Функциональные требования описывают, что система должна делать. Они основаны на функциональности, реализованной в коде.

1. **Шифрование файлов**
   - ПО должно предоставлять возможность выбора файла для шифрования через диалоговое окно.
   - ПО должно запрашивать у пользователя пароль для шифрования файла.
   - ПО должно использовать алгоритм шифрования AES с ключом, сгенерированным на основе пароля и соли.
   - Зашифрованный файл должен сохраняться с расширением .enc в той же директории, что и исходный файл.
   - После успешного шифрования должно отображаться уведомление об успешном завершении операции.
2. **Дешифрование файлов**
   - ПО должно предоставлять возможность выбора зашифрованного файла (с расширением .enc) через диалоговое окно.
   - ПО должно запрашивать у пользователя пароль для дешифрования файла.
   - ПО должно использовать алгоритм AES для дешифрования файла на основе пароля, соли и вектора инициализации (IV).
   - Расшифрованный файл должен сохраняться с расширением .dec в той же директории, что и зашифрованный файл.
   - После успешного дешифрования должно отображаться уведомление об успешном завершении операции.
3. **Обработка ошибок**
   - ПО должно проверять наличие выбранного файла перед выполнением операций шифрования или дешифрования.
   - ПО должно корректно обрабатывать случаи, когда введен неверный пароль, и уведомлять пользователя об ошибке.
   - ПО должно предотвращать сбои при работе с файлами (например, отсутствие доступа к файлу или недостаток места на диске).

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, безопасности, удобства использования и т.д.

1. **Производительность**
    - Шифрование и дешифрование файлов размером до 1 ГБ должны выполняться не более чем за 5 минут на стандартном оборудовании.
    - Операции шифрования и дешифрования должны выполняться в фоновом режиме, чтобы не блокировать интерфейс пользователя.
2. **Безопасность**
    - ПО должно использовать стойкий алгоритм шифрования AES с ключом длиной 256 бит.
    - Пароль должен хэшироваться с использованием PBKDF2 (Rfc2898DeriveBytes) с 100 000 итераций для повышения стойкости к атакам методом перебора.
    - Соль и вектор инициализации (IV) должны генерироваться с использованием криптографически стойкого генератора случайных чисел (RandomNumberGenerator).
    - Зашифрованные файлы должны быть защищены от несанкционированного доступа.
3. **Удобство использования (Usability)**
    - Интерфейс должен быть интуитивно понятным, с четкими подсказками и сообщениями об ошибках.
    - Диалоговые окна для выбора файлов должны быть простыми и понятными.
    - Пользователь должен получать уведомления о завершении операций шифрования и дешифрования.
4. **Совместимость (Compatibility)**
    - ПО должно работать на операционных системах Windows (версии 10 и выше).
    - ПО должно поддерживать шифрование и дешифрование файлов любого типа (текстовые, бинарные, изображения и т.д.).
5. **Надежность (Reliability)**
    - ПО должно корректно обрабатывать ошибки, такие как отсутствие доступа к файлу, недостаток места на диске или повреждение зашифрованного файла.
    - При возникновении ошибок должно отображаться понятное сообщение пользователю.
6. **Масштабируемость (Scalability)**
    - ПО должно поддерживать шифрование и дешифрование файлов размером до 10 ГБ без значительного снижения производительности.
7. **Локализация (Localization)**

- Текстовые сообщения и интерфейс должны поддерживать русский язык.
- Формат уведомлений и сообщений об ошибках должен быть понятным для пользователя.

22. Управление здоровьем

Описание:
Создать программу для отслеживания здоровья, включая физическую активность, питание и сон.

Функциональные требования

Функциональные требования описывают, что система должна делать. Они основаны на функциональности, реализованной в коде.

1. **Трекинг активности:**
   - Система должна позволять пользователю добавлять информацию о физической активности, включая тип активности и её продолжительность.
   - Если тип активности уже существует в системе, система должна обновить общую продолжительность для этого типа.
   - Система должна отображать сообщение об успешном добавлении активности.
2. **Трекинг питания:**
   - Система должна позволять пользователю добавлять информацию о потребляемой пище, включая название продукта и количество калорий.
   - Если продукт уже существует в системе, система должна обновить общее количество калорий для этого продукта.
   - Система должна отображать сообщение об успешном добавлении продукта.
3. **Трекинг сна:**
   - Система должна позволять пользователю добавлять информацию о сне, включая дату и количество часов сна.
   - Если запись о сне на указанную дату уже существует, система должна обновить количество часов сна.
   - Система должна отображать сообщение об успешном добавлении информации о сне.
4. **Отчет о данных:**
   - Система должна предоставлять возможность просмотра отчета, содержащего информацию о всех отслеживаемых активностях, питании и сне.
   - Отчет должен отображаться в отдельном окне.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**
    - Система должна обрабатывать добавление новой активности, продукта или записи о сне за время не более 1 секунды.
    - Отчет должен формироваться и отображаться за время не более 2 секунд.
2. **Удобство использования:**
    - Система должна предоставлять интуитивно понятный интерфейс для ввода данных и просмотра отчетов.
    - Все сообщения об успешном добавлении данных должны быть четкими и легко читаемыми.
3. **Надежность:**
    - Система должна корректно обрабатывать повторяющиеся записи (например, обновление данных для уже существующих активностей, продуктов или дат сна).
    - Система должна быть устойчива к некорректным данным (например, отрицательные значения продолжительности активности, калорий или часов сна).
4. **Совместимость:**
    - Система должна быть совместима с операционными системами Windows (версии 10 и выше).
    - Система должна поддерживать работу с .NET Framework 4.7.2 и выше.
5. **Безопасность:**
    - Система должна обеспечивать сохранность данных пользователя (активности, питание, сон) в течение всего времени использования.
    - Доступ к данным должен быть ограничен только текущим пользователем.
6. **Масштабируемость:**
    - Система должна поддерживать добавление до 10 000 записей для каждого типа данных (активности, питание, сон) без потери производительности.
7. **Локализация:**
    - Все сообщения и интерфейс системы должны быть на русском языке.

23. Управление музыкальной коллекцией

Описание:

Создать программу для управления музыкальной коллекцией, включая добавление, удаление, поиск и сортировку треков.

Функциональные требования

Функциональные требования описывают, что система должна делать. Они основаны на функциональности, реализованной в коде.

1. **Управление музыкальной коллекцией:**
    - Система должна позволять добавлять музыкальные треки с указанием исполнителя, названия трека, жанра и года выпуска.
    - Система должна позволять удалять музыкальные треки из коллекции.
    - Система должна отображать сообщение об успешном добавлении или удалении трека.
2. **Отображение коллекции:**
    - Система должна отображать список всех музыкальных треков в коллекции, включая информацию об исполнителе, названии трека, жанре и годе выпуска.
    - Система должна обновлять отображение коллекции после добавления или удаления трека.
3. **Поиск по исполнителю:**
    - Система должна позволять пользователю искать треки по имени исполнителя.
    - Система должна отображать все треки, соответствующие запросу, или сообщение, если треки не найдены.
4. **Сортировка по году выпуска:**
    - Система должна позволять сортировать музыкальные треки по году выпуска в порядке возрастания.
    - Система должна обновлять отображение коллекции после сортировки.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**
    - Система должна обрабатывать добавление или удаление трека за время не более 1 секунды.
    - Поиск по исполнителю должен выполняться за время не более 2 секунд.
    - Сортировка треков по году выпуска должна выполняться за время не более 1 секунды.
2. **Удобство использования:**

    - Интерфейс системы должен быть интуитивно понятным, с четким отображением списка треков.
    - Все сообщения (успешное добавление, удаление, поиск, сортировка) должны быть четкими и легко читаемыми.
3. **Надежность:**
    - Система должна корректно обрабатывать попытки удаления несуществующего трека.
    - Система должна быть устойчива к некорректным данным (например, пустые строки или неверные форматы года выпуска).
4. **Совместимость:**
    - Система должна быть совместима с операционными системами Windows (версии 10 и выше).
    - Система должна поддерживать работу с .NET Framework 4.7.2 и выше.
5. **Безопасность:**
    - Система должна обеспечивать сохранность данных пользователя (музыкальная коллекция) в течение всего времени использования.
    - Доступ к данным должен быть ограничен только текущим пользователем.
6. **Масштабируемость:**
    - Система должна поддерживать добавление до 10 000 треков без потери производительности.
7. **Локализация:**
    - Все сообщения и интерфейс системы должны быть на русском языке.

24. Управление путешествиями

Описание:
Создать программу для планирования и управления путешествиями, включая маршруты, бронирование отелей и отслеживание расходов.

Функциональные требования

Функциональные требования описывают, что система должна делать. Они основаны на функциональности, реализованной в коде.

1. **Управление поездками:**
    - Система должна позволять создавать поездки с указанием пункта назначения, даты начала, даты окончания и бюджета.
    - Система должна хранить информацию о каждой поездке, включая список расходов.
2. **Управление расходами:**
    - Система должна позволять добавлять расходы к поездке, включая описание и сумму.

    - Система должна отображать сообщение об успешном добавлении расхода.
3. **Расчеты по поездке:**
    - Система должна рассчитывать общую сумму всех расходов для поездки.
    - Система должна рассчитывать оставшийся бюджет для поездки на основе общего бюджета и суммы расходов.
4. **Отображение деталей поездки:**
    - Система должна предоставлять возможность просмотра деталей поездки, включая пункт назначения, даты, бюджет, список расходов, общую сумму расходов и оставшийся бюджет.
    - Детали поездки должны отображаться в отдельном окне.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**
    - Система должна обрабатывать добавление нового расхода за время не более 1 секунды.
    - Расчет общей суммы расходов и оставшегося бюджета должен выполняться за время не более 1 секунды.
    - Отображение деталей поездки должно происходить за время не более 2 секунд.
2. **Удобство использования:**
    - Интерфейс системы должен быть интуитивно понятным, с четким отображением деталей поездки и расходов.
    - Все сообщения (успешное добавление расхода) должны быть четкими и легко читаемыми.
3. **Надежность:**
    - Система должна корректно обрабатывать попытки добавления расходов с некорректными данными (например, отрицательные суммы).
    - Система должна быть устойчива к ошибкам при расчете бюджета и расходов.
4. **Совместимость:**
    - Система должна быть совместима с операционными системами Windows (версии 10 и выше).
    - Система должна поддерживать работу с .NET Framework 4.7.2 и выше.
5. **Безопасность:**

    - Система должна обеспечивать сохранность данных пользователя (информация о поездках и расходах) в течение всего времени использования.
    - Доступ к данным должен быть ограничен только текущим пользователем.
6. **Масштабируемость:**
    - Система должна поддерживать добавление до 1 000 расходов для каждой поездки без потери производительности.
7. **Локализация:**
    - Все сообщения и интерфейс системы должны быть на русском языке.

25. Управление фотографиями

Описание:
Создать программу для управления фотографиями, включая их организацию в альбомы, добавление описаний и сортировку.

Функциональные требования

Функциональные требования описывают, что система должна делать. Они основаны на функциональности, реализованной в коде.

1. **Управление фотографиями:**
    - Система должна позволять добавлять фотографии в альбом, включая путь к файлу, описание и дату съемки.
    - Система должна отображать сообщение об успешном добавлении фотографии.
2. **Удаление фотографий:**
    - Система должна позволять удалять фотографии из альбома.
    - Система должна отображать сообщение об успешном удалении фотографии.
    - Система должна предупреждать пользователя, если фотография для удаления не выбрана.
3. **Отображение фотографий:**
    - Система должна отображать список всех фотографий в альбоме, включая путь к файлу, описание и дату съемки.
    - Система должна обновлять отображение списка фотографий после добавления или удаления.
4. **Сортировка фотографий:**
    - Система должна позволять сортировать фотографии по дате съемки в порядке возрастания.
    - Система должна отображать сообщение об успешной сортировке.
5. **Получение описания:**

- Система должна предоставлять пользователю возможность ввода описания для добавляемой фотографии через отдельное диалоговое окно.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**
   - Система должна обрабатывать добавление новой фотографии за время не более 1 секунды.
   - Удаление фотографии должно выполняться за время не более 1 секунды.
   - Сортировка фотографий должна выполняться за время не более 2 секунд.
2. **Удобство использования:**
   - Интерфейс системы должен быть интуитивно понятным, с четким отображением списка фотографий.
   - Все сообщения (успешное добавление, удаление, сортировка) должны быть четкими и легко читаемыми.
   - Диалоговое окно для ввода описания должно быть простым и удобным для пользователя.
3. **Надежность:**
   - Система должна корректно обрабатывать попытки удаления фотографии, если она не выбрана.
   - Система должна быть устойчива к некорректным данным (например, пустые строки или неверные форматы даты).
4. **Совместимость:**
   - Система должна быть совместима с операционными системами Windows (версии 10 и выше).
   - Система должна поддерживать работу с .NET Framework 4.7.2 и выше.
5. **Безопасность:**
   - Система должна обеспечивать сохранность данных пользователя (информация о фотографиях) в течение всего времени использования.
   - Доступ к данным должен быть ограничен только текущим пользователем.
6. **Масштабируемость:**
   - Система должна поддерживать добавление до 10 000 фотографий без потери производительности.
7. **Локализация:**

- Все сообщения и интерфейс системы должны быть на русском языке.

26. Управление рецептами и планированием меню

Описание:
Создать программу для управления кулинарными рецептами и планирования меню на неделю.

Функциональные требования

Функциональные требования описывают, что система должна делать. Они основаны на функциональности, реализованной в коде.

1. **Управление рецептами:**
   - Система должна позволять создавать рецепты с указанием названия, описания, списка ингредиентов, инструкций по приготовлению и количества калорий.
   - Система должна хранить информацию о каждом рецепте.
2. **Управление планом питания:**
   - Система должна позволять добавлять рецепты в план питания на определенную дату.
   - Система должна отображать сообщение об успешном добавлении рецепта в план.
   - Система должна предотвращать добавление рецепта на дату, если на эту дату уже есть запись.
3. **Удаление рецептов из плана:**
   - Система должна позволять удалять рецепты из плана питания.
   - Система должна отображать сообщение об успешном удалении рецепта.
   - Система должна предупреждать пользователя, если рецепт для удаления не выбран.
4. **Поиск рецептов:**
   - Система должна позволять искать рецепты по названию.
   - Система должна отображать найденный рецепт или сообщение, если рецепт не найден.
5. **Отображение плана питания:**
   - Система должна отображать список всех рецептов в плане питания, включая дату и название рецепта.
   - Система должна обновлять отображение плана питания после добавления или удаления рецепта.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**
    - Система должна обрабатывать добавление нового рецепта в план питания за время не более 1 секунды.
    - Удаление рецепта из плана питания должно выполняться за время не более 1 секунды.
    - Поиск рецепта по названию должен выполняться за время не более 2 секунд.
2. **Удобство использования:**
    - Интерфейс системы должен быть интуитивно понятным, с четким отображением плана питания и рецептов.
    - Все сообщения (успешное добавление, удаление, поиск) должны быть четкими и легко читаемыми.
    - Формы для добавления и поиска рецептов должны быть простыми и удобными для пользователя.
3. **Надежность:**
    - Система должна корректно обрабатывать попытки добавления рецепта на уже занятую дату.
    - Система должна быть устойчива к некорректным данным (например, пустые строки или неверные форматы даты).
4. **Совместимость:**
    - Система должна быть совместима с операционными системами Windows (версии 10 и выше).
    - Система должна поддерживать работу с .NET Framework 4.7.2 и выше.
5. **Безопасность:**
    - Система должна обеспечивать сохранность данных пользователя (информация о рецептах и плане питания) в течение всего времени использования.
    - Доступ к данным должен быть ограничен только текущим пользователем.
6. **Масштабируемость:**
    - Система должна поддерживать добавление до 1 000 рецептов в план питания без потери производительности.
7. **Локализация:**
    - Все сообщения и интерфейс системы должны быть на русском языке.

27. Управление обучением и курсами

Описание:

Создать программу для управления курсами и отслеживания прогресса обучения.

Функциональные требования

Функциональные требования описывают, что система должна делать, и основываются на функциональности, представленной в коде.

1. **Управление курсами:**
    - Система должна позволять создавать курс с указанием его названия, описания, даты начала и даты окончания.
    - Система должна предоставлять возможность добавлять и удалять модули в курсе.
    - Система должна отображать информацию о курсе, включая его название, описание, даты и список модулей.
2. **Управление модулями:**
    - Система должна позволять создавать модуль с указанием его названия.
    - Система должна предоставлять возможность добавлять и удалять темы в модуле.
    - Система должна позволять обновлять прогресс модуля в диапазоне от 0% до 100%.
    - Система должна отображать информацию о модуле, включая его название, список тем и текущий прогресс.
3. **Управление темами:**
    - Система должна позволять создавать тему с указанием её названия.
    - Система должна предоставлять возможность удалять тему из модуля.
4. **Взаимодействие с пользователем:**
    - Система должна отображать сообщения пользователю при добавлении, удалении или обновлении элементов (курсов, модулей, тем).
    - Система должна отображать предупреждения при попытке установить недопустимое значение прогресса (например, меньше 0% или больше 100%).

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**

    - Система должна обрабатывать добавление и удаление модулей и тем за время не более 1 секунды при количестве элементов до 1000.
    - Система должна отображать информацию о курсе или модуле за время не более 2 секунд.
2. **Удобство использования:**
    - Интерфейс системы должен быть интуитивно понятным, с четкими сообщениями об успешных или ошибочных действиях.
    - Система должна поддерживать возможность отмены последнего действия (например, удаления модуля или темы).
3. **Надежность:**
    - Система должна корректно обрабатывать ошибки, такие как попытка удалить несуществующий модуль или тему.
    - Система должна сохранять целостность данных при сбоях (например, не допускать потери данных при внезапном завершении работы).
4. **Масштабируемость:**
    - Система должна поддерживать работу с большим количеством курсов, модулей и тем (до 10 000 элементов каждого типа).
5. **Безопасность:**
    - Система должна обеспечивать защиту данных от несанкционированного доступа (например, шифрование данных при хранении).
6. **Совместимость:**
    - Система должна поддерживать работу на операционных системах Windows, macOS и Linux.
    - Система должна быть совместима с браузерами Chrome, Firefox и Edge (если используется веб-интерфейс).
7. **Локализация:**
    - Система должна поддерживать несколько языков интерфейса (например, русский и английский).

28. Управление резюме и поиском работы

Описание:
Создать программу для управления резюме и отслеживания вакансий.

Функциональные требования

Функциональные требования описывают, что система должна делать, и основываются на функциональности, представленной в коде.

1. **Управление резюме:**
    - Система должна позволять создавать резюме с указанием имени, контактной информации и цели.

    - Система должна предоставлять возможность добавлять и отображать навыки в резюме.
    - Система должна позволять добавлять и отображать опыт работы (должность, компания, период, описание).
    - Система должна позволять добавлять и отображать образование (учебное заведение, степень, период).
    - Система должна отображать полное резюме, включая имя, контактную информацию, цель, навыки, опыт работы и образование.
2. **Управление вакансиями:**
    - Система должна позволять добавлять вакансии с указанием должности, компании, описания и требований.
    - Система должна предоставлять возможность поиска и отображения списка вакансий.
3. **Взаимодействие с пользователем:**
    - Система должна отображать сообщения пользователю при успешном создании резюме, добавлении навыков, опыта работы, образования или вакансии.
    - Система должна отображать предупреждения, если список резюме или вакансий пуст.
4. **Поиск и отображение:**
    - Система должна предоставлять возможность поиска вакансий по ключевым словам (например, должность, компания).
    - Система должна отображать детали вакансии, включая должность, компанию, описание и требования.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**
    - Система должна обрабатывать добавление и отображение резюме и вакансий за время не более 1 секунды при количестве элементов до 1000.
    - Система должна выполнять поиск вакансий за время не более 2 секунд.
2. **Удобство использования:**
    - Интерфейс системы должен быть интуитивно понятным, с четкими сообщениями об успешных или ошибочных действиях.
    - Система должна поддерживать возможность отмены последнего действия (например, удаления навыка или вакансии).
3. **Надежность:**

    - Система должна корректно обрабатывать ошибки, такие как попытка добавить пустое значение или удалить несуществующий элемент.
    - Система должна сохранять целостность данных при сбоях (например, не допускать потери данных при внезапном завершении работы).
4. **Масштабируемость:**
    - Система должна поддерживать работу с большим количеством резюме и вакансий (до 10 000 элементов каждого типа).
5. **Безопасность:**
    - Система должна обеспечивать защиту данных от несанкционированного доступа (например, шифрование данных при хранении).
    - Система должна требовать авторизации для доступа к конфиденциальным данным (например, контактной информации в резюме).
6. **Совместимость:**
    - Система должна поддерживать работу на операционных системах Windows, macOS и Linux.
    - Система должна быть совместима с браузерами Chrome, Firefox и Edge (если используется веб-интерфейс).
7. **Локализация:**
    - Система должна поддерживать несколько языков интерфейса (например, русский и английский).

29. Управление встречами и мероприятиями

Описание:
Создать программу для управления встречами и мероприятиями, включая создание, редактирование и уведомления.

Функциональные требования

Функциональные требования описывают, что система должна делать, и основываются на функциональности, представленной в коде.

1. **Управление событиями:**
    - Система должна позволять создавать событие с указанием названия, времени начала, времени окончания, места и описания.
    - Система должна предоставлять возможность редактирования существующих событий.
    - Система должна позволять удалять события.
    - Система должна отображать список событий с их названиями, временем начала, временем окончания и местом проведения.
2. **Управление напоминаниями:**

    - Система должна позволять устанавливать напоминание для события.
    - Система должна позволять снимать напоминание для события.
    - Система должна отображать статус напоминания (установлено/не установлено) для каждого события.
3. **Отображение информации о событиях:**
    - Система должна предоставлять возможность просмотра детальной информации о событии, включая название, время начала и окончания, место, описание и статус напоминания.
4. **Взаимодействие с пользователем:**
    - Система должна отображать сообщения пользователю при успешном создании, редактировании или удалении события.
    - Система должна отображать предупреждения, если список событий пуст.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**
    - Система должна обрабатывать создание, редактирование и удаление событий за время не более 1 секунды при количестве событий до 1000.
    - Система должна отображать список событий за время не более 2 секунд.
2. **Удобство использования:**
    - Интерфейс системы должен быть интуитивно понятным, с четкими сообщениями об успешных или ошибочных действиях.
    - Система должна поддерживать возможность отмены последнего действия (например, удаления события).
3. **Надежность:**
    - Система должна корректно обрабатывать ошибки, такие как попытка редактирования или удаления несуществующего события.
    - Система должна сохранять целостность данных при сбоях (например, не допускать потери данных при внезапном завершении работы).
4. **Масштабируемость:**
    - Система должна поддерживать работу с большим количеством событий (до 10 000 элементов).
5. **Безопасность:**

    - Система должна обеспечивать защиту данных от несанкционированного доступа (например, шифрование данных при хранении).
6. **Совместимость:**
    - Система должна поддерживать работу на операционных системах Windows, macOS и Linux.
    - Система должна быть совместима с браузерами Chrome, Firefox и Edge (если используется веб-интерфейс).
7. **Локализация:**
    - Система должна поддерживать несколько языков интерфейса (например, русский и английский).

30. Управление техникой и оборудованием

Описание:
Создать программу для управления техникой и оборудованием, включая их состояние и обслуживание.

Функциональные требования

Функциональные требования описывают, что система должна делать, и основываются на функциональности, представленной в коде.

1. **Управление оборудованием:**
    - Система должна позволять добавлять оборудование с указанием названия, типа, серийного номера, даты покупки, даты последнего обслуживания и текущего состояния.
    - Система должна предоставлять возможность удаления оборудования по серийному номеру.
    - Система должна отображать список оборудования с его названием, типом, серийным номером и текущим состоянием.
2. **Обслуживание оборудования:**
    - Система должна позволять выполнять обслуживание оборудования, обновляя дату последнего обслуживания и устанавливая состояние "В хорошем состоянии".
    - Система должна позволять помечать оборудование как неисправное.
    - Система должна позволять помечать оборудование как находящееся в ремонте.
3. **Отображение информации об оборудовании:**
    - Система должна предоставлять возможность просмотра детальной информации об оборудовании, включая название, тип, серийный номер, дату покупки, дату последнего обслуживания и текущее состояние.
4. **Взаимодействие с пользователем:**

- Система должна отображать сообщения пользователю при успешном добавлении, удалении или обновлении оборудования.
- Система должна отображать предупреждения, если список оборудования пуст или оборудование с указанным серийным номером не найдено.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**
   - Система должна обрабатывать добавление и удаление оборудования за время не более 1 секунды при количестве элементов до 1000.
   - Система должна отображать список оборудования за время не более 2 секунд.
2. **Удобство использования:**
   - Интерфейс системы должен быть интуитивно понятным, с четкими сообщениями об успешных или ошибочных действиях.
   - Система должна поддерживать возможность отмены последнего действия (например, удаления оборудования).
3. **Надежность:**
   - Система должна корректно обрабатывать ошибки, такие как попытка удаления несуществующего оборудования.
   - Система должна сохранять целостность данных при сбоях (например, не допускать потери данных при внезапном завершении работы).
4. **Масштабируемость:**
   - Система должна поддерживать работу с большим количеством оборудования (до 10 000 элементов).
5. **Безопасность:**
   - Система должна обеспечивать защиту данных от несанкционированного доступа (например, шифрование данных при хранении).
6. **Совместимость:**
   - Система должна поддерживать работу на операционных системах Windows, macOS и Linux.
   - Система должна быть совместима с браузерами Chrome, Firefox и Edge (если используется веб-интерфейс).
7. **Локализация:**
   - Система должна поддерживать несколько языков интерфейса (например, русский и английский).

31. Управление автомобилем и его обслуживанием

Описание:
Создать программу для управления автомобилем и его обслуживанием, включая отслеживание пробега, расходов и плановых работ.

Функциональные требования

Функциональные требования описывают, что система должна делать, и основываются на функциональности, представленной в коде.

1. **Управление транспортными средствами:**
    - Система должна позволять создавать транспортное средство с указанием марки, модели, года выпуска и VIN-номера.
    - Система должна отображать информацию о транспортном средстве, включая марку, модель, год выпуска, VIN-номер, текущий пробег и дату последнего обслуживания.
2. **Управление пробегом:**
    - Система должна позволять добавлять пробег к транспортному средству.
    - Система должна обновлять общий пробег транспортного средства и отображать сообщение об успешном обновлении.
3. **Управление обслуживанием:**
    - Система должна позволять выполнять обслуживание транспортного средства с указанием описания и стоимости.
    - Система должна сохранять историю обслуживания, включая описание, стоимость и дату выполнения.
    - Система должна обновлять дату последнего обслуживания после выполнения обслуживания.
    - Система должна позволять планировать плановое обслуживание с указанием описания и даты.
4. **Отображение информации:**
    - Система должна предоставлять возможность просмотра детальной информации о транспортном средстве, включая марку, модель, год выпуска, VIN-номер, пробег, дату последнего обслуживания и историю обслуживания.
5. **Взаимодействие с пользователем:**
    - Система должна отображать сообщения пользователю при успешном добавлении пробега, выполнении обслуживания или планировании обслуживания.
    - Система должна отображать предупреждения, если введены некорректные данные (например, отрицательный пробег).

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**
    - Система должна обрабатывать добавление пробега и выполнение обслуживания за время не более 1 секунды.
    - Система должна отображать информацию о транспортном средстве за время не более 2 секунд.
2. **Удобство использования:**
    - Интерфейс системы должен быть интуитивно понятным, с четкими сообщениями об успешных или ошибочных действиях.
    - Система должна поддерживать возможность отмены последнего действия (например, добавления пробега).
3. **Надежность:**
    - Система должна корректно обрабатывать ошибки, такие как попытка добавить отрицательный пробег или некорректные данные обслуживания.
    - Система должна сохранять целостность данных при сбоях (например, не допускать потери данных при внезапном завершении работы).
4. **Масштабируемость:**
    - Система должна поддерживать работу с большим количеством транспортных средств (до 10 000 элементов).
5. **Безопасность:**
    - Система должна обеспечивать защиту данных от несанкционированного доступа (например, шифрование данных при хранении).
    - Система должна требовать авторизации для доступа к конфиденциальным данным (например, VIN-номеру).
6. **Совместимость:**
    - Система должна поддерживать работу на операционных системах Windows, macOS и Linux.
    - Система должна быть совместима с браузерами Chrome, Firefox и Edge (если используется веб-интерфейс).
7. **Локализация:**
    - Система должна поддерживать несколько языков интерфейса (например, русский и английский).

32. Управление энергопотреблением

Описание:
Создать программу для отслеживания и управления потреблением энергии.

Функциональные требования

Функциональные требования описывают, что система должна делать, и основываются на функциональности, представленной в коде.

1. **Управление измерениями энергии:**
    - Система должна позволять начинать измерение энергии для устройства с указанием его названия и потребляемой мощности.
    - Система должна позволять останавливать измерение энергии для выбранного устройства.
    - Система должна рассчитывать потребление энергии на основе времени работы устройства и его мощности.
2. **Отображение данных:**
    - Система должна отображать список активных измерений с указанием названия устройства, потребляемой мощности и рассчитанного потребления энергии.
    - Система должна предоставлять возможность просмотра детальной информации о потреблении энергии для всех устройств.
3. **Взаимодействие с пользователем:**
    - Система должна отображать сообщения пользователю при успешном начале или завершении измерения.
    - Система должна отображать предупреждения, если нет активных измерений или введены некорректные данные (например, отрицательная мощность).
4. **Расчет энергии:**
    - Система должна корректно рассчитывать потребление энергии на основе времени работы устройства и его мощности.
    - Система должна возвращать нулевое значение энергии, если измерение не было остановлено.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**
    - Система должна обрабатывать начало и завершение измерений за время не более 1 секунды.
    - Система должна отображать список измерений за время не более 2 секунд.
2. **Удобство использования:**
    - Интерфейс системы должен быть интуитивно понятным, с четкими сообщениями об успешных или ошибочных действиях.
    - Система должна поддерживать возможность отмены последнего действия (например, начала измерения).

3. **Надежность:**
    - Система должна корректно обрабатывать ошибки, такие как попытка остановить несуществующее измерение или ввод некорректных данных.
    - Система должна сохранять целостность данных при сбоях (например, не допускать потери данных при внезапном завершении работы).
4. **Масштабируемость:**
    - Система должна поддерживать работу с большим количеством устройств (до 10 000 элементов).
5. **Безопасность:**
    - Система должна обеспечивать защиту данных от несанкционированного доступа (например, шифрование данных при хранении).
6. **Совместимость:**
    - Система должна поддерживать работу на операционных системах Windows, macOS и Linux.
    - Система должна быть совместима с браузерами Chrome, Firefox и Edge (если используется веб-интерфейс).
7. **Локализация:**
    - Система должна поддерживать несколько языков интерфейса (например, русский и английский).

33. Управление личными целями и задачами

Описание:
Создать программу для управления личными целями и задачами, включая их установку, отслеживание прогресса и напоминания.

Функциональные требования

Функциональные требования описывают, что система должна делать, и основываются на функциональности, представленной в коде.

1. **Управление целями:**
    - Система должна позволять создавать цель с указанием названия, описания и дедлайна.
    - Система должна предоставлять возможность редактирования существующих целей.
    - Система должна позволять удалять цели.
    - Система должна отображать список целей с их названиями, дедлайнами и текущим прогрессом.
2. **Управление прогрессом:**
    - Система должна позволять обновлять прогресс выполнения цели в диапазоне от 0% до 100%.

    - Система должна отображать сообщение об ошибке, если значение прогресса выходит за допустимые пределы.
3. **Управление напоминаниями:**
    - Система должна позволять устанавливать напоминание для цели.
    - Система должна позволять снимать напоминание для цели.
    - Система должна отображать статус напоминания (установлено/не установлено) для каждой цели.
4. **Отображение информации:**
    - Система должна предоставлять возможность просмотра детальной информации о цели, включая название, описание, дедлайн, прогресс и статус напоминания.
5. **Взаимодействие с пользователем:**
    - Система должна отображать сообщения пользователю при успешном создании, редактировании или удалении цели.
    - Система должна отображать предупреждения, если список целей пуст или введены некорректные данные (например, отрицательный прогресс).

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**
    - Система должна обрабатывать создание, редактирование и удаление целей за время не более 1 секунды при количестве целей до 1000.
    - Система должна отображать список целей за время не более 2 секунд.
2. **Удобство использования:**
    - Интерфейс системы должен быть интуитивно понятным, с четкими сообщениями об успешных или ошибочных действиях.
    - Система должна поддерживать возможность отмены последнего действия (например, удаления цели).
3. **Надежность:**
    - Система должна корректно обрабатывать ошибки, такие как попытка редактирования или удаления несуществующей цели.
    - Система должна сохранять целостность данных при сбоях (например, не допускать потери данных при внезапном завершении работы).
4. **Масштабируемость:**
    - Система должна поддерживать работу с большим количеством целей (до 10 000 элементов).

5. **Безопасность:**
    - Система должна обеспечивать защиту данных от несанкционированного доступа (например, шифрование данных при хранении).
6. **Совместимость:**
    - Система должна поддерживать работу на операционных системах Windows, macOS и Linux.
    - Система должна быть совместима с браузерами Chrome, Firefox и Edge (если используется веб-интерфейс).
7. **Локализация:**
    - Система должна поддерживать несколько языков интерфейса (например, русский и английский).

34. Управление коллекцией игр

Описание:
Создать программу для управления коллекцией игр, включая добавление, удаление, поиск и сортировку.

Функциональные требования

Функциональные требования описывают, что система должна делать, и основываются на функциональности, представленной в коде.

1. **Управление коллекцией игр:**
    - Система должна позволять добавлять игру с указанием названия, жанра, платформы, года выпуска и рейтинга.
    - Система должна предоставлять возможность удаления игры из коллекции.
    - Система должна отображать список игр с их названиями, жанрами, платформами и годами выпуска.
2. **Сортировка игр:**
    - Система должна позволять сортировать игры по рейтингу в порядке убывания.
    - Система должна позволять сортировать игры по году выпуска в порядке убывания.
3. **Поиск игр:**
    - Система должна предоставлять возможность поиска игр по названию.
    - Система должна предоставлять возможность поиска игр по жанру.
    - Система должна отображать сообщение, если игры по заданным критериям не найдены.
4. **Отображение информации:**
    - Система должна предоставлять возможность просмотра детальной информации о всех играх в коллекции, включая название, жанр, платформу, год выпуска и рейтинг.

5. **Взаимодействие с пользователем:**
    - Система должна отображать сообщения пользователю при успешном добавлении, удалении или сортировке игр.
    - Система должна отображать предупреждения, если коллекция пуста или игры не найдены.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**
    - Система должна обрабатывать добавление, удаление и сортировку игр за время не более 1 секунды при количестве игр до 1000.
    - Система должна выполнять поиск игр за время не более 2 секунд.
2. **Удобство использования:**
    - Интерфейс системы должен быть интуитивно понятным, с четкими сообщениями об успешных или ошибочных действиях.
    - Система должна поддерживать возможность отмены последнего действия (например, удаления игры).
3. **Надежность:**
    - Система должна корректно обрабатывать ошибки, такие как попытка удаления несуществующей игры.
    - Система должна сохранять целостность данных при сбоях (например, не допускать потери данных при внезапном завершении работы).
4. **Масштабируемость:**
    - Система должна поддерживать работу с большим количеством игр (до 10 000 элементов).
5. **Безопасность:**
    - Система должна обеспечивать защиту данных от несанкционированного доступа (например, шифрование данных при хранении).
6. **Совместимость:**
    - Система должна поддерживать работу на операционных системах Windows, macOS и Linux.
    - Система должна быть совместима с браузерами Chrome, Firefox и Edge (если используется веб-интерфейс).
7. **Локализация:**
    - Система должна поддерживать несколько языков интерфейса (например, русский и английский).

35. Управление доставкой

Описание:
Создать программу для управления процессом доставки, включая отслеживание посылок и управление заказами.

Функциональные требования

Функциональные требования описывают, что система должна делать, и основываются на функциональности, представленной в коде.

1. **Управление отправками:**
    - Система должна позволять создавать отправку с указанием номера для отслеживания, отправителя, получателя, пункта отправления, пункта назначения и даты отправки.
    - Система должна предоставлять возможность обновления статуса отправки (например, "В ожидании", "В пути", "Доставлено", "Задержано", "Утеряно").
    - Система должна позволять удалять отправку из списка.
2. **Отображение информации:**
    - Система должна отображать список отправок с их номерами для отслеживания и текущими статусами.
    - Система должна предоставлять возможность просмотра детальной информации об отправке, включая номер для отслеживания, отправителя, получателя, пункт отправления, пункт назначения, дату отправки и статус.
3. **Поиск отправок:**
    - Система должна предоставлять возможность поиска отправки по номеру для отслеживания.
    - Система должна отображать сообщение, если отправка с указанным номером не найдена.
4. **Взаимодействие с пользователем:**
    - Система должна отображать сообщения пользователю при успешном создании, обновлении статуса или удалении отправки.
    - Система должна отображать предупреждения, если список отправок пуст или отправка не найдена.

Нефункциональные требования

Нефункциональные требования описывают, как система должна работать, и включают аспекты производительности, удобства использования, надежности и т.д.

1. **Производительность:**

    - Система должна обрабатывать создание, обновление и удаление отправок за время не более 1 секунды при количестве отправок до 1000.
    - Система должна выполнять поиск отправки по номеру для отслеживания за время не более 2 секунд.
2. **Удобство использования:**
    - Интерфейс системы должен быть интуитивно понятным, с четкими сообщениями об успешных или ошибочных действиях.
    - Система должна поддерживать возможность отмены последнего действия (например, удаления отправки).
3. **Надежность:**
    - Система должна корректно обрабатывать ошибки, такие как попытка обновления статуса несуществующей отправки.
    - Система должна сохранять целостность данных при сбоях (например, не допускать потери данных при внезапном завершении работы).
4. **Масштабируемость:**
    - Система должна поддерживать работу с большим количеством отправок (до 10 000 элементов).
5. **Безопасность:**
    - Система должна обеспечивать защиту данных от несанкционированного доступа (например, шифрование данных при хранении).
    - Система должна требовать авторизации для доступа к конфиденциальным данным (например, информации об отправителе и получателе).
6. **Совместимость:**
    - Система должна поддерживать работу на операционных системах Windows, macOS и Linux.
    - Система должна быть совместима с браузерами Chrome, Firefox и Edge (если используется веб-интерфейс).
7. **Локализация:**
    - Система должна поддерживать несколько языков интерфейса (например, русский и английский).