



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

Отчет по практической работе

по дисциплине «Тестирование и верификация ПО»

Выполнили:

Студенты группы ИКБО-74-23

Селянин Н.М.

Чихачев И.А.

xxxxx x.x.

xxxx x.x.

Проверил:

Преподаватель

Ильичев Г.П.

2025 г.

Техническое задание:

1. Введение

Настоящий документ описывает требования к небольшому программному продукту «Учёт экспедиции». Программа предназначена для хранения списка участников экспедиции с их ролями, просмотра списка, фильтрации и подсчёта численности. Реализация ориентирована на тестирование методом «чёрного ящика».

2. Основания для разработки

Работа выполняется в рамках учебной практики по дисциплине «Тестирование программного обеспечения» для подгруппы «Экспедиция 74-23».

3. Назначение разработки

Программный продукт обеспечивает базовый учёт состава экспедиции:

- добавление участника с указанием роли;
- вывод полного списка участников;
- вывод участников по выбранной роли;
- удаление участника;
- подсчёт общего количества участников.

4. Требования к программе

4.1 Функциональные требования

- **F1. Добавление участника.** Команда: `add <Имя> <Роль>`.
- **F2. Вывод всех участников.** Команда: `list` — сортировка по алфавиту по полю «Имя».
- **F3. Фильтрация по роли.** Команда: `list --role <Роль>` — список участников выбранной роли.
- **F4. Подсчёт численности.** Команда: `count` — вывод общего количества участников.
- **F5. Удаление участника.** Команда: `remove <Имя>` — удаление записи по имени.

4.2 Ограничения и валидация ввода

- **Имя:** от 2 до 40 символов; допустимы буквы кириллицы/латиницы, пробел и дефис.
- **Роль:** выбирается из фиксированного списка: штурман, водитель, грузчик, механик. Сравнение — без учёта регистра.
- **Уникальность:** имена уникальны (сравнение — без учёта регистра).
- Недопустимые значения сопровождаются понятным сообщением об ошибке, выполнение команды не приводит к изменению данных.

4.3 Нефункциональные требования

- **Тип приложения:** консольная утилита (CLI).
- **Язык интерфейса:** русский.
- **Производительность:** операции выполняются интерактивно (менее 0,5 сек для списков до 10 000 записей).
- **Надёжность:** при ошибке ввода программа не завершается аварийно.
- **Безопасность:** работа с локальным файлом данных без сетевого доступа.
- **Портируемость:** Windows 10/11, macOS 12+, Linux (совместимость командного интерфейса).

4.4 Совместимость и зависимости

- Интерпретатор/движок выбранного языка программирования и стандартная библиотека. Внешние сетевые или БД-зависимости отсутствуют.

4.5 Хранение данных

- Персистентность обеспечивается локальным файлом members.json в рабочей директории приложения.
- Формат данных — JSON (см. Приложение А).

5. Требования к интерфейсу

Интерфейс командной строки. Формат сообщений:

- **Успех:** ОК: <сообщение>

- **Ошибка валидации:** ERR: <описание проблемы>
- **Неизвестная команда:** ERR: unknown command

Примеры ожидаемых ответов приведены в «Руководстве пользователя».

6. Критерии приёмки

- Реализованы функции F1–F5 и вся валидация из п. 4.2.
- Команды возвращают корректные сообщения об успехе/ошибках.
- Данные сохраняются и корректно восстанавливаются между запусками.
- Тестовое покрытие сценариями «чёрного ящика»: не менее 15 тест-кейсов, включая не менее 5 негативных; успешно выполняется $\geq 95\%$ тест-кейсов при корректной конфигурации.

7. Состав документации

- Настоящее ТЗ.
- Руководство пользователя.
- Инструкция по установке и запуску.
- Приложение: описание формата данных.

8. Порядок контроля и приёмки

- Предварительная проверка корректности ввода и сообщений об ошибках.
- Функциональное тестирование по сценариям из Руководства пользователя.
- Приёмочные испытания: демонстрация выполнения F1–F5, проверки персистентности, сортировки и фильтрации по роли.

9. Этапы и сроки выполнения

1. Подготовка ТЗ и структуры проекта.
2. Реализация CLI-интерфейса и операций F1–F5.
3. Реализация персистентности (JSON файл).
4. Оформление документации и примерных сценариев.
5. Внутренняя проверка и фиксация версии 1.0.

Руководство пользователя:

1. Назначение

Программа «Учёт экспедиции» поддерживает добавление, просмотр, фильтрацию, удаление участников и подсчёт численности экспедиции.

2. Поддерживаемые роли

штурман, водитель, грузчик, механик (регистр не важен).

3. Команды и примеры

3.1 Добавление участника

Команда:

add <Имя> <Роль>

Примеры и ответы:

add Анна штурман

ОК: добавлен "Анна" (штурман)

add АННА ШТУРМАН

ERR: такое имя уже существует

3.2 Просмотр всех участников

Команда:

list

Ответ: нумерованный список, отсортированный по алфавиту.

Пример:

1) Анна — штурман

2) Борис — механик

3) Иван — водитель

3.3 Фильтрация по роли

Команда:

list --role <Роль>

Пример:

list --role механик

1) Борис — механик

При отсутствии записей выводится:

нет записей

3.4 Подсчёт участников

Команда:

count

Пример ответа:

Всего участников: 3

3.5 Удаление участника

Команда:

remove <Имя>

Примеры и ответы:

remove Иван

ОК: удалён "Иван"

remove Пётр

ERR: запись с именем "Пётр" не найдена

3.6 Сообщения об ошибках (типовые)

- ERR: имя должно быть 2–40 символов (буквы, пробел, дефис)
- ERR: роль должна быть одной из: штурман, водитель, грузчик, механик
- ERR: такое имя уже существует
- ERR: запись с именем "<Имя>" не найдена
- ERR: unknown command

Инструкция по установке и запуску

1. Системные требования

- Windows 10/11, macOS 12+ или Linux.
- Установленный интерпретатор/рантайм выбранного языка программирования.

2. Запуск

- Запуск выполняется из командной строки/терминала в каталоге с программой.
- Файл данных `members.json` создаётся автоматически при первом успешном добавлении участника.
- Завершение работы — стандартным способом ОС (например, `Ctrl+C` в терминале).

Приложение А. Формат файла данных (JSON)

А.1 Структура

Файл `members.json` содержит массив объектов следующего вида:

```
[  
  {  
    "name": "Анна",  
    "role": "штурман"  
  },  
  {  
    "name": "Борис",  
    "role": "механик"  
  }  
]
```

А.2 Требования к данным

- Поле `name` — строка 2–40 символов; уникальна без учёта регистра.
- Поле `role` — одно из: штурман, водитель, грузчик, механик (хранится в нижнем регистре либо в том виде, в котором введено — по реализации; поведение интерфейса при сравнении регистр-независимое).
- При повреждении файла программа должна корректно сообщить об ошибке чтения без аварийного завершения.

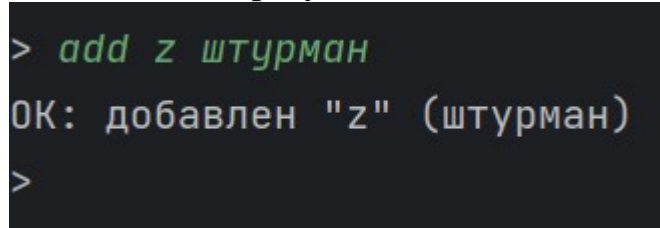
4. Ошибки:

Ошибка №1: Проверка валидации длины имени при добавлении участника

Описание Убедиться, что система корректно проверяет длину имени участника (2-40 символов)

Ожидаемый результат Система возвращает ошибку: "имя должно быть (2–40 символов)"

Фактический результат



```
> add z штурман
OK: добавлен "z" (штурман)
>
```

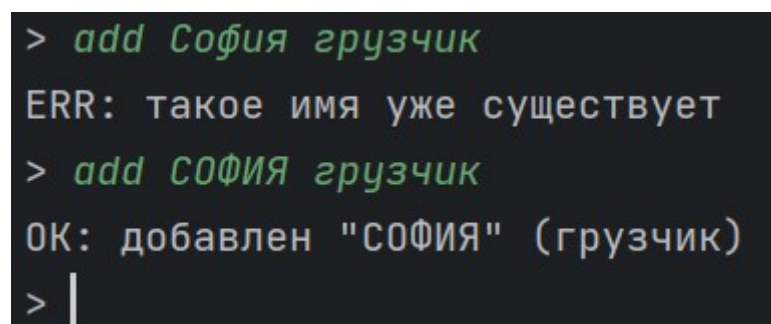
Рисунок 1 – Система принимает короткое имя: "OK: добавлен 'z' (штурман)"

Ошибка №2: Проверка регистра о независимости при проверке дубликатов имен

Описание: Убедиться, что система корректно проверяет длину имени участника (2-40 символов)

Ожидаемый результат: Система возвращает ошибку: "имя должно быть 2–40 символов"

Фактический результат



```
> add София грузчик
ERR: такое имя уже существует
> add СОФИЯ грузчик
OK: добавлен "СОФИЯ" (грузчик)
> |
```

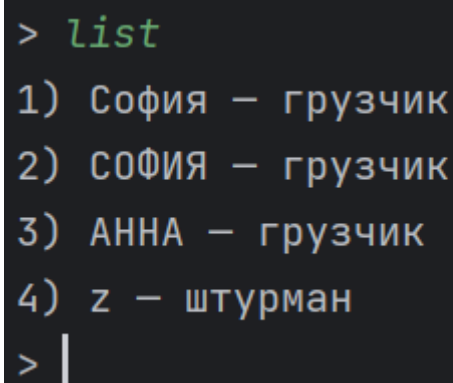
Рисунок 2 – Система принимает имя как уникальное: "OK: добавлен СОФИЯ (грузчик)"

Ошибка №3: Проверка сортировки списка участников по возрастанию

Описание: Убедиться, что список участников сортируется по имени в алфавитном порядке (А→Я)

Ожидаемый результат: Список отображается в порядке по алфавиту

Фактический результат



```
> list
1) София – грузчик
2) СОФИЯ – грузчик
3) АННА – грузчик
4) z – штурман
> |
```

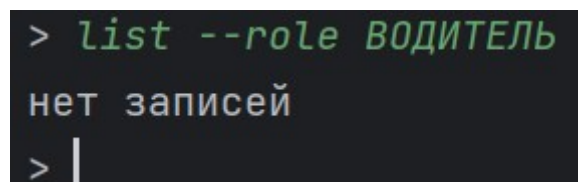
Рисунок 3 – Список отображается в обратном порядке

Ошибка №4: Проверка регистра о независимости фильтрации по роли

Описание: Убедиться, что фильтр по роли работает независимо от регистра ввода

Ожидаемый результат: Система отображает участника с ролью "водитель"

Фактический результат



```
> list --role ВОДИТЕЛЬ
нет записей
> |
```

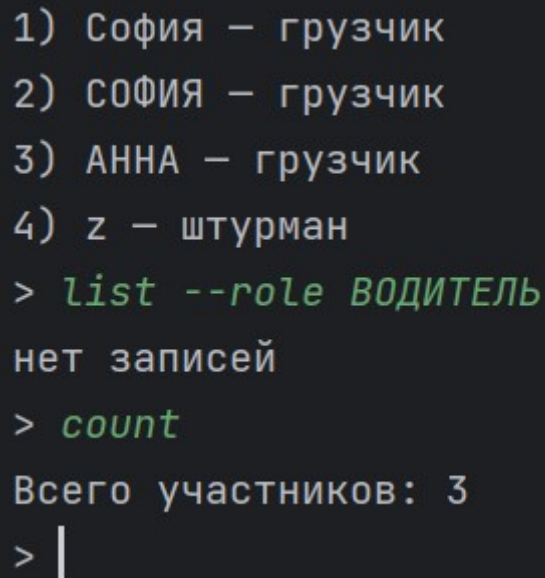
Рисунок 4 – Система возвращает: "нет записей"

Ошибка №5: Проверка корректности подсчета участников

Описание: Убедиться, что команда `count` возвращает точное количество участников

Ожидаемый результат Система возвращает: "Всего участников: 4"

Фактический результат



```
1) София – грузчик
2) СОФИЯ – грузчик
3) АННА – грузчик
4) z – штурман
> list --role ВОДИТЕЛЬ
нет записей
> count
Всего участников: 3
> |
```

Рисунок 5 – Система возвращает: "Всего участников: 3"

Техническое задание другой команды: «Калькулятор заказов для кофейни»

1.1 Рассматриваемый программный продукт

Программный продукт «Калькулятор заказов для кофейни» предназначен для автоматизации процесса расчета стоимости заказов в заведениях малого бизнеса. Система позволяет сотрудникам быстро формировать заказ, учитывать добавки, применять скидки и получать итоговую сумму. Основная область применения — кофейни, кафе и другие точки розничной торговли, где важна скорость и точность обслуживания..

1.2 Основания для разработки

Разработка инициирована в рамках учебной дисциплины «Тестирование и верификация программного обеспечения» с целью закрепления навыков проектирования и реализации прикладного программного обеспечения. Основанием служат:

- Методические указания преподавателя по практическому заданию 1
- Требования к оформлению технической документации согласно ГОСТ Р 34.602-2020
- Потребность в демонстрации навыков командной разработки
- Актуальность задачи для малого бизнеса, особенно в сфере HoReCa

1.3 Назначение разработки

Цель разработки — создание удобного инструмента для автоматизации учета заказов, позволяющего:

- Сократить время обработки одного заказа на 30 %
- Минимизировать ошибки при расчетах вручную
- Повысить прозрачность и контроль над скидками и акциями
- Обеспечить простоту использования для сотрудников без технической подготовки
- Упростить формирование отчетов по заказам

Программа должна быть интуитивной, надежной и легко адаптируемой под разные сценарии работы кофейни..

1.4 Требования к программе

1.4.1 Функциональные требования

- Ввод данных о напитке, добавке и количестве
- Расчёт стоимости заказа с учетом базовых цен
- Применение скидки по промокоду
- Очистка формы заказа
- Вывод итоговой суммы
- Возможность расширения каталога напитков и добавок

1.4.2 Надежность

- Все операции должны быть атомарными — частичный сбой не должен влиять на другие функции
- Восстановление последнего состояния при перезапуске

1.4.3 Условия эксплуатации

- Операционная система: Windows 10 и выше
- Язык интерфейса: русский
- Минимальные системные требования: CPU Intel i3, 4 ГБ ОЗУ, экран от 1024x768 px

1.4.4 Совместимость

- Возможность запуска без установки дополнительных библиотек
- Возможность адаптации под Linux при наличии интерпретатора Python

1.5 Требования к интерфейсу

1.5.1 Общие требования

- Все элементы интерфейса должны быть выполнены на русском языке с учетом правил орфографии и терминологии Заказчика.
- Цветовая схема — стандартная для консольных и веб-компонентов (темный фон/светлый текст или на оборот), контраст не менее 4,5:1.
- Шрифты — Monospace 14 pt для консоли, Sans-Serif 14 pt для веб-страниц.

1.5.2 Прототипы и макеты

- Списки категории напитков и добавок
- Поля ввода количества и промокода
- Кнопки “Применить промокод”, “Очистить”, “Подсчитать”
- Итоговая сумма в рублях

1.5.3 Описание поведения элементов

- При клике на “Применить промокод” выводит сумму заказа с учетом скидки
- При клике на “Подсчитать” выводит сумму заказа
- Поля ввода количества и промокода принимают только цифры

1.5.4 Требования к доступности

- Все надписи — не менее 14 pt, интервалы между строками — 1,5;
- Ошибки должны быть очевидны: текст красного цвета для ошибок, зелёного для успешных операций;
- При формировании заказа звук клика не обязателен, но возможен для десктоп-версии.

1.6 Критерии приемки

1.6.1 Функциональные параметры

- Успешное выполнение не менее 95 % всех тест-кейсов (F1–F6).
- Корректный расчет итоговой суммы при стандартных и граничных данных (включая 0, максимальное количество, максимальную цену).

1.6.2 Не функциональные параметры

- Время расчета одного заказа ≤ 1 с на тестовой машине Intel i3/4 ГБ RAM.
- Падение отказов не более 0,1 % при 100 одновременных сессиях.
- Полное восстановление состояния после аварийного завершения в 99 % случаев.

1.6.3 Параметры интерфейса

- Нет Ориентированность окна консоли: весь текст выровнен по ширине;
- Отзывчивость веб-интерфейса: hover-эффект не позже чем через 200 мс.

1.6.4 Отчётность при приемке

- Тест-отчёт по каждому сценарию (unit, интеграция, система) с указанием статуса Passed/Failed.
- Акт приемки-сдаточный с подписью представителя Заказчика и Исполнителя.

1.7 Порядок контроля и приемки

1.7.1 Перечень обязательных документов

- Руководство пользователя: 20–30 строк, описание шагов от запуска до запуска
- Архитектуры системы: диаграммы компонентов, описание API, структура каталогов.
- Техническое описание: пояснение архитектурных решений, протоколы обмена.

1.7.2 Форматы и стандарты

- DOCX или PDF (Confluence для рабочего хранения).
- Шрифт — Times New Roman 14 pt, интервалы 1,5, выравнивание по ширине, поля 20 × 10 × 20 × 10 мм.
- Нумерация разделов по ГОСТ Р 34.602-2020.

1.7.3 Сроки предоставления

- Рабочие версии руководств — при готовности прототипа (этап 11.4).
- Итоговые — не позднее приёмки проекта (этап 11.6).

1.8. Порядок контроля и приемки

1.8.1 Методы тестирования

- Модульное тестирование с покрытием 100 % логики функций формирования заказа.
- Интеграционное тестирование API: 50 + эндпоинтов GET/POST по спецификации.
- Системное и приемочное тестирование по чек-листам и пользовательским сценариям.

1.8.2 Фиксация и устранение дефектов

- Все баги регистрируются в системе Jira с меткой «Приемка».
- Критические ошибки (блокирующие расчёт или сохранение отчёта) устраняются в течение 1 рабочего дня.
- Неблокирующие — до конца этапа 11.5.

1.8.3 Приемочный акт

- Составляется после успешного прохождения всех пунктов 6.1–6.4.
- Включает перечень выполненных проверок, подписи сторон, дату приемки.

1.9. Требования к документации

1.9.1 Руководство пользователя

- Описание процедур формирования заказа, применения скидок и вывода отчетов.

1.9.2 Руководство разработчика

- Архитектурная схема.
- Описание модулей и их интерфейсов.

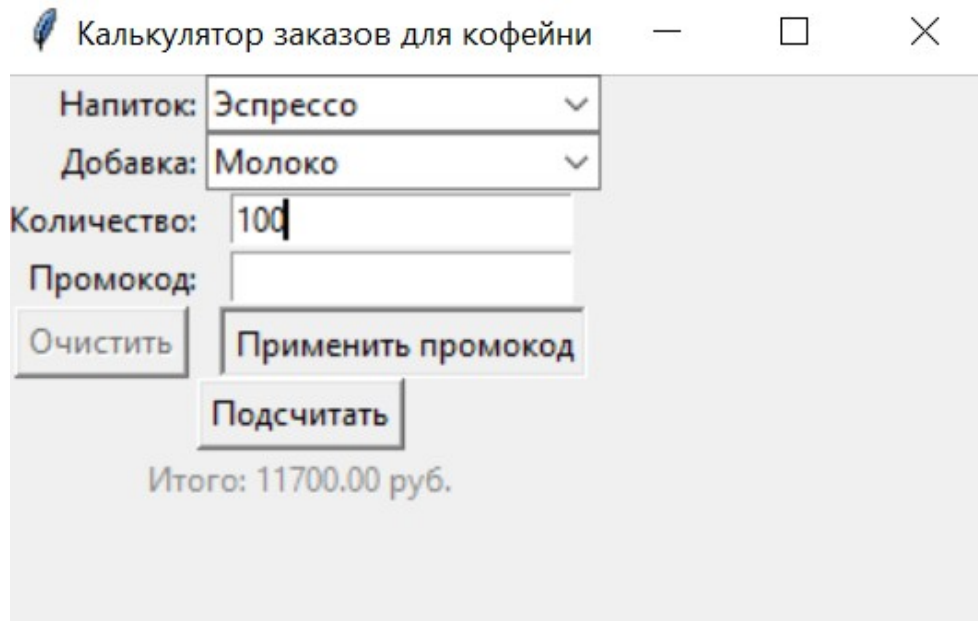
1.9.3 Отчет об испытаниях

- Сценарии функционального, нагрузочного тестирования и результаты.

2 ТЕСТИРОВАНИЕ ПП

2.1 Тестирование программного продукта другой группы

Протестируем собственный программный продукт и задокументируем найденные ошибки.



Ошибка №1: Скидка применяется даже при пустом промокоде

Рисунок 1 – Демонстрация ошибки при пустом промокоде

Ошибка №2: Кнопка «Очистить» визуально неактивна.

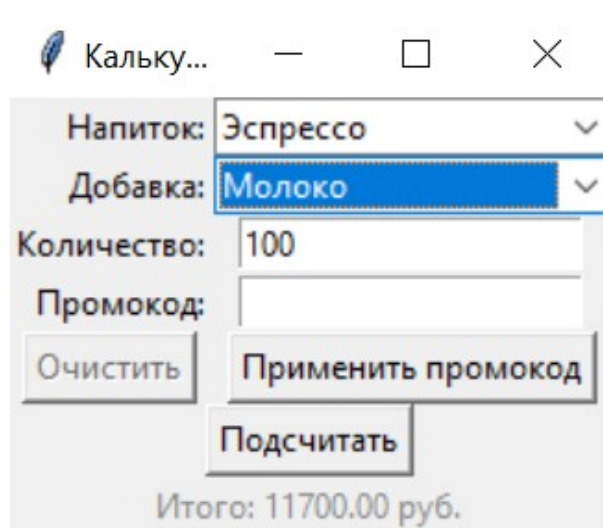


Рисунок 2 – Демонстрация визуальной ошибки кнопки

Ошибка №3: При вводе отрицательного количества программа зависает

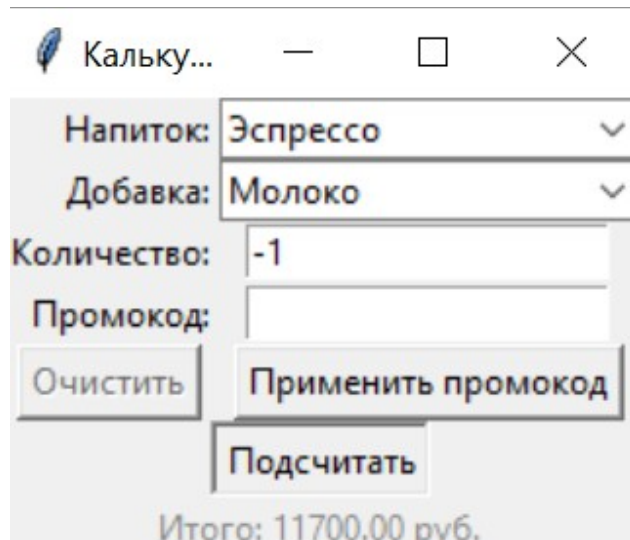
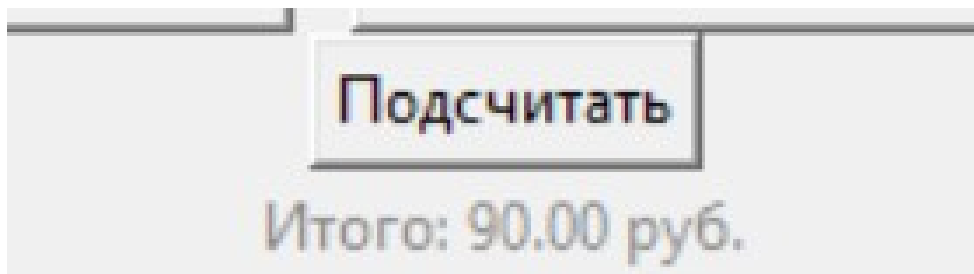


Рисунок 3 – Демонстрация ошибки при вводе отрицательного количества



Ошибка №4: Итоговая сумма отображается серым текстом на сером фоне

Рисунок 4 – Демонстрация ошибки итоговая сумма

Ошибка №5: В инструкции указано, что кнопка «Рассчитать»

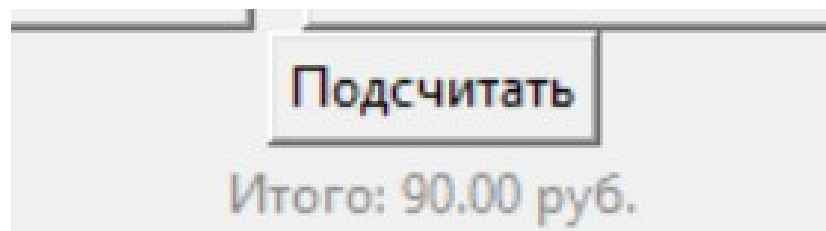


Рисунок 5 – Демонстрация ошибки кнопка «Рассчитать»

3 Тестирование технической документации

В ходе тестирования документации мы выявили критические несоответствия между техническим заданием и реализованным программным продуктом. В частности, приложение не предоставляет возможность смены языка интерфейса, которая была явно указана в функциональных требованиях (п. 1.4 ТЗ). Кроме того, анализ выявленных ошибок (№1–№5) показывает, что в документации отсутствуют или некорректно описаны требования к валидации ввода, обработке исключительных ситуаций и usability-аспектам интерфейса. Например:

- **Ошибка №1** (применение скидки при пустом промокоде) указывает на отсутствие проверки входных данных.
- **Ошибка №3** (зависание при вводе отрицательного количества) свидетельствует о недостатках в описании бизнес-логики и обработки некорректных значений.
- **Ошибки №2 и №4** (визуальные дефекты) говорят о неполноте требований к пользовательскому интерфейсу.

Эти нарушения подтверждают разрыв между этапом формирования требований и этапом реализации, что классифицируется как дефект процесса разработки.

Заключение

Тестирование программного продукта и технической документации выявило ряд критических недочётов. Найденные ошибки (№1–№5) демонстрируют отсутствие нормализации ввода, недостаточную валидацию данных, визуальные и логические дефекты интерфейса, а также несоответствие реализованного функционала требованиям ТЗ (например, отсутствие смены языка). Для исправления ситуации необходимо:

- Уточнить и дополнить требования к валидации ввода и обработке исключений.
- Доработать логику работы с промокодами и количеством товаров.
- Исправить визуальные элементы интерфейса в соответствии с принципами usability.
- Реализовать недостающий функционал (смена языка) и провести модульное тестирование ключевых функций.

Только после устранения указанных недочётов можно говорить о соответствии продукта заявленным требованиям и его готовности к эксплуатации.