МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1
по дисциплине
«Тестирование и верификация программного обеспечения»

Тема: «ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА МЕТОДОМ «ЧЕРНОГО ЯЩИКА»»

Выполнил студент группы ИКБО-43-23 Зепалов Е.В.
Зернов Н.А.
Коротаев А.М.
Любишкин С.Н.
Степанов К.К.

Принял Ильичев Г.П.

Практическая работа выполнена «__»_____2025г. *(подпись студента)*

«Зачтено» «__»_____2025 г. *(подпись руководителя)*

Москва 2025

ВВЕДЕНИЕ

Цель работы заключается в знакомстве студентов с процессом тестирования программного обеспечения, включая подготовку технической документации, выявление ошибок и их документирование. Развитие навыков командной работы при тестировании программного продукта. Для достижения поставленной цели работы студентам необходимо выполнить ряд задач:

1) научиться разрабатывать программный продукт с учетом требований для дальнейшего тестирования;

2) овладеть методом тестирования «черного ящика»;

3) разработать и протестировать техническое задание и документацию;

4) научиться находить ошибки в чужом программном продукте и документировать их;

5) оформить итоговый отчет по проделанной работе в соответствии с установленными стандартами;

6) освоить работу с инструментами для управления тестированием и отслеживания багов;

7) изучить распределение ролей в команде тестировщиков

1 Разработка технического задания программного продукта

1.1.1 Наименование программы

GUI UNIX OS — эмулятор UNIX-подобной операционной системы с графическим окном терминала (реализовано с использованием pygame), принимающий образ виртуальной файловой системы в виде zip-архива.

1.1.2 Краткая характеристика области применения.

Программа предназначена для обучения и тестирования базовых команд UNIX, имитации работы с файловой системой и выполнения простых сценариев (start-скриптов) в контролируемой, безопасной (виртуальной) среде. Подходит для учебных лабораторий, демонстраций и самостоятельного обучения

1.2.1 Основание для проведения разработки

Необходимость создания учебного инструмента для отработки базовых операций в shell-среде без риска повредить реальную файловую систему

1.2.2 Наименование и условное обозначение темы разработки

Наименование: «Эмулятор UNIX-подобной ОС — GUI UNIX OS».

Шифр темы: GUI-UNIX-001

1.3.1 Функциональное назначение

Обеспечение пользователю возможности вводить команды, аналогичные базовым UNIX-командам, просматривать и изменять структуру виртуальной файловой системы, загруженной из ZIP-архива, и получать текстовый вывод

результатов.

1.3.2 Эксплуатационное назначение

Использование в учебных и демонстрационных целях: для студентов, преподавателей и сотрудников, желающих изучать работу shell-команд и манипуляции с файловой системой в безопасном окружении.

1.3.3 Задачи, решаемые в ходе разработки

Задачи, решаемые в ходе разработки

- Реализация текстового интерфейса/терминала в окне pygame.
- Обработка и монтирование структуры виртуальной файловой системы из ZIP-архива.
- Поддержка набора команд: help, ls, cd, exit, wc, mv, clear.
- Выполнение стартового скрипта (если указан) — последовательного выполнения команд из файла.

Корректная обработка ошибок, логирование событий, простые тесты/приёмка

1.4.1 Требования к составу выполняемых функций

Запуск программы с параметрами командной строки:

--user — имя пользователя, используемое в приглашении.

--archive — путь к zip-архиву, представляющему виртуальную файловую систему.

--script — имя скрипта внутри архива, который нужно выполнить при старте.

help — вывод списка доступных команд и кратких описаний.

ls — перечисление содержимого текущей директории или указанного пути.

cd — смена текущей директории.

exit — корректное завершение работы эмулятора.

wc — подсчёт: строк, слов или символов в указанном текстовом файле.

mv — перемещение файла/папки или переименование в пределах виртуальной ФС.

clear — очистка выводимой консоли в окне эмулятора.

Логирование: ошибки и важные события записываются в файл логов.

1.4.2 Требования к организации входных данных

Входные данные программы должны быть организованы в виде:

Архив ZIP (стандартный ZIP): корневая папка архива рассматривается как корень виртуальной файловой системы. В архиве допустимы файлы любых форматов (текстовые, бинарные), и вложенные директории.

Файл start-скрипта (если указан) должен быть текстовым и находиться внутри архива по указанному относительному пути.

Программа должна корректно обрабатывать:

отсутствие архива вывод корректной диагностической ошибки и завершение;

повреждённый/некорректный архив аккуратное сообщение об ошибке и запись в лог.

1.4.3 Требования к организации выходных данных

Вывод команд — текст в окне эмулятора; при необходимости — дополнительно запись в лог.

Форматы сообщений: понятные и однозначные тексты ошибок.

При завершении работы — возвратный код процесса 0 при успешном завершении, ненулевой код при фатальных ошибках.

1.4.4Требования к обеспечению надежного (устойчивого) функционирования программы

Корректная обработка некорректных входных данных (несуществующие пути, неверный формат файлов и т.п.) без краха хоста.

Любая ошибка в работе (исключение Python) должна быть перехвачена, краткая информация показана пользователю, подробный стек запись в лог

1.4.5 Время восстановления после отказа

Перезапуск приложения (старт скрипта/монтажа архива) должен занимать не более 1 минуты при стандартных условиях (медленное IO не учитывается)

1.4.6 Отказы из-за некорректных действий пользователя

Некорректные команды должны возвращать сообщение об ошибке и не приводить к изменению реальной файловой системы хоста.

Скрипты и команды запускаются внутри эмулятора и не должны выполнять реальных shell-команд на хосте

1.4.7 Требования к видам обслуживания

ОС: Windows 10/11, Linux (современные дистрибутивы), macOS — при установленном Python 3.10+.

Наличие интернета не обязательно для работы, требуется только при установке зависимостей

1.4.8 Требования к численности и квалификации персонала

Для запуска: пользователь с базовыми навыками работы в командной строке и ОС.

Для сопровождения: разработчик/администратор, знакомый с Python и pygame

1.4.9 Требования к составу и параметрам технических средств

- CPU: 1 ГГц или выше.
- RAM: 512 МБ или выше.
- HDD/SSD: 200 МБ свободного места (зависит от размера архивов).
- (Рекомендуется: 2 ГБ RAM, 1 ГБ свободного диска для логов и тестовых архивов.)

1.4.10 Требования к информационным структурам и методам решения

Внутреннее представление структуры файлов — древовидные структуры/словарь Python; для чтения архива использовать модуль zipfile.

1.4.11 Требования к исходным кодам и языкам программирования

Язык: Python 3.10 и выше.

Структура проекта: main.py, модули fs.py (виртуальная ФС), commands.py, ui.py (pygame), logger.py, tests/

1.4.12 Требования к программным средствам, используемым программой

Программа должна работать под управлением ОС Linux/Windows с установленным интерпретатором Python и необходимыми зависимостями

1.5.1 Общие требования к интерфейсу

Интерфейс реализован в окне pygame: область вывода текста + строка

ввода (prompt). Текст выводится моноширинным шрифтом; новая команда вводится одной строкой

1.5.2 Поведение интерфейса

Кнопки окна: стандартные (свернуть, развернуть, закрыть). При закрытии окна — корректное завершение (сохранение логов).

1.6.1 Ожидаемый эффект

Ускорение усвоения студентами практических навыков работы с командной строкой в безопасной среде. Снижение риска случайного повреждения реальной файловой системы при обучении.

1.6.2 Затраты на разработку

Разработка программного продукта осуществляется в рамках учебного проекта. Финансовые затраты отсутствуют. Затраты выражаются только в трудозатратах исполнителей (время и усилия студентов).

1.7 Состав программной документации

Состав программной документации должен включать в себя:

а) техническое задание;

б) пользовательская документация;

в) техническая документация;

г) текстовая документация;

д) системная документация.

1.8.1 Организация приемки

Приемка готового продукта осуществляется командой тестирования.

1.8.2 Методы контроля качества

Контроль качества осуществляется методом черного ящика.

1.8.3 Тестовые сценарии

Для приемки другой команде предоставляется тестовый сценарий, включающий не менее 10 тест-кейсов, покрывающих все функциональные требования п. 1.4.1

1.8.4 Критерии приемки

Критерием успешной приемки является успешное прохождение не менее 95% всех тест-кейсов

1.8.5 Документация приемки

После успешного прохождения испытаний подписывается Акт о приемке программного обеспечения в опытную эксплуатацию.

1.8.6 Завершение приемки

По итогам успешной опытной эксплуатации в течение установленного срока подписывается Акт о приемке программного обеспечения в промышленную эксплуатацию.

1.9.1 План-график разработки

Разработка программного обеспечения должна быть выполнена в следующие сроки (таблица 1).

Таблица 1 - План-график разработки

№	Наименование этапа	Срок исполнения	Исполнитель
1	Анализ требований и проектирование архитектуры	3 рабочих дня	Разработчик
2	Написание кода и модульное тестирование	8 рабочих дней	Разработчик
3	Наполнение контентом	2 рабочих дня	Заказчик
4	Пилотная эксплуатация и приемочные испытания	3 рабочих дней	Совместно
5	Ввод в промышленную эксплуатацию	1 рабочий день	Разработчик
Итого		17 дней	Совместно

.

1.9.2 Форматы представления документации

Вся документация должна быть предоставлена в электронном виде в форматах PDF (для руководств) и .py (для исходного кода).

2 ДОПОЛНИТЕЛЬНАЯ ДОКУМЕНТАЦИЯ

2.1.1 Архитектура**

GUI UNIX OS — учебное графическое приложение на pygame, эмулирующее базовые команды UNIX поверх виртуальной файловой системы, упакованной в ZIP-архив.

Поток работы:

1.Пользователь вводит /start.

2. Бот спрашивает имя и переходит в MAIN_MENU.

3. В главном меню доступны кнопки: «Кто мы», «Заповеди», «Отделы», «Словарь», «Перезапустить».

4. Выбор «Отделы» переводит в состояние DEPARTMENTS.

5. «Перезапустить» возвращает к главному меню, «/cancel» завершает диалог.

Данные (миссия, заповеди, словарь, описания отделов) хранятся в словаре TEXTS.

Клавиатуры формируются через ReplyKeyboardMarkup.

2.1.2 Основные функции

- help — показать список доступных команд.

- ls — вывести содержимое текущего каталога виртуальной ФС.

- cd <путь> — сменить каталог. Поддерживаются '/', '..' и относительные пути.

- wc <файл> — посчитать количество строк, слов и символов в файле из архива.

- mv <источник> <назначение> — переместить или переименовать файл внутри ZIP.

- clear — очистка экрана консоли.

- exit — завершить работу приложения.

2.2 Тестовая документация

2.2.1 Юз-кейсы

UC-01. Просмотр корня виртуальной ФС

- Открыть приложение с архивом systeam.zip.

- Ввести команду: ls.

- Ожидаемый результат: выведен список элементов каталога systeam/ (dir1/, so.txt, text.txt).

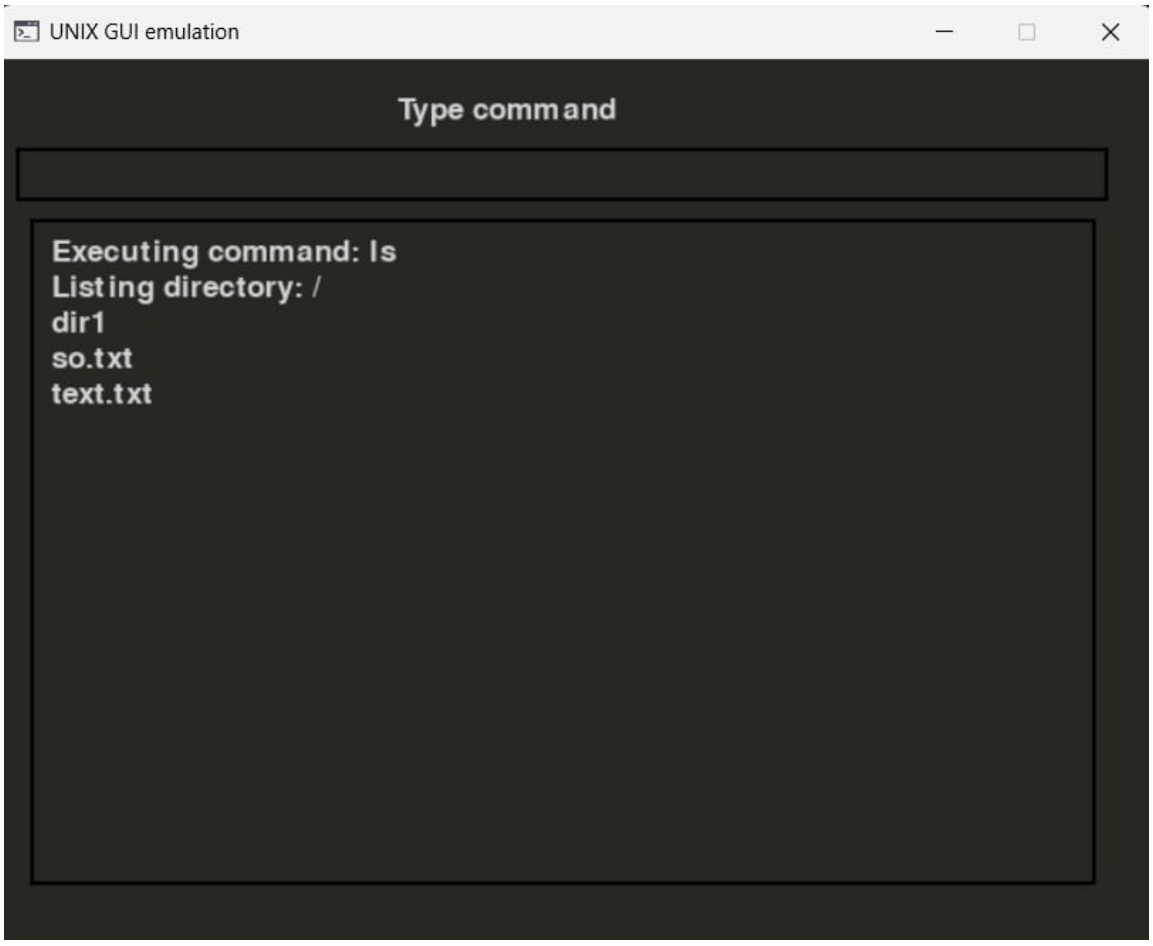

Рисунок 1 – Просмотр корня

UC-02. Переход в подкаталог и возврат

- Команда: cd dir1

- Команда: ls

- Ожидаемый результат: виден файл a.txt.

- Команда: cd ..

- Ожидаемый результат: снова каталог system/.

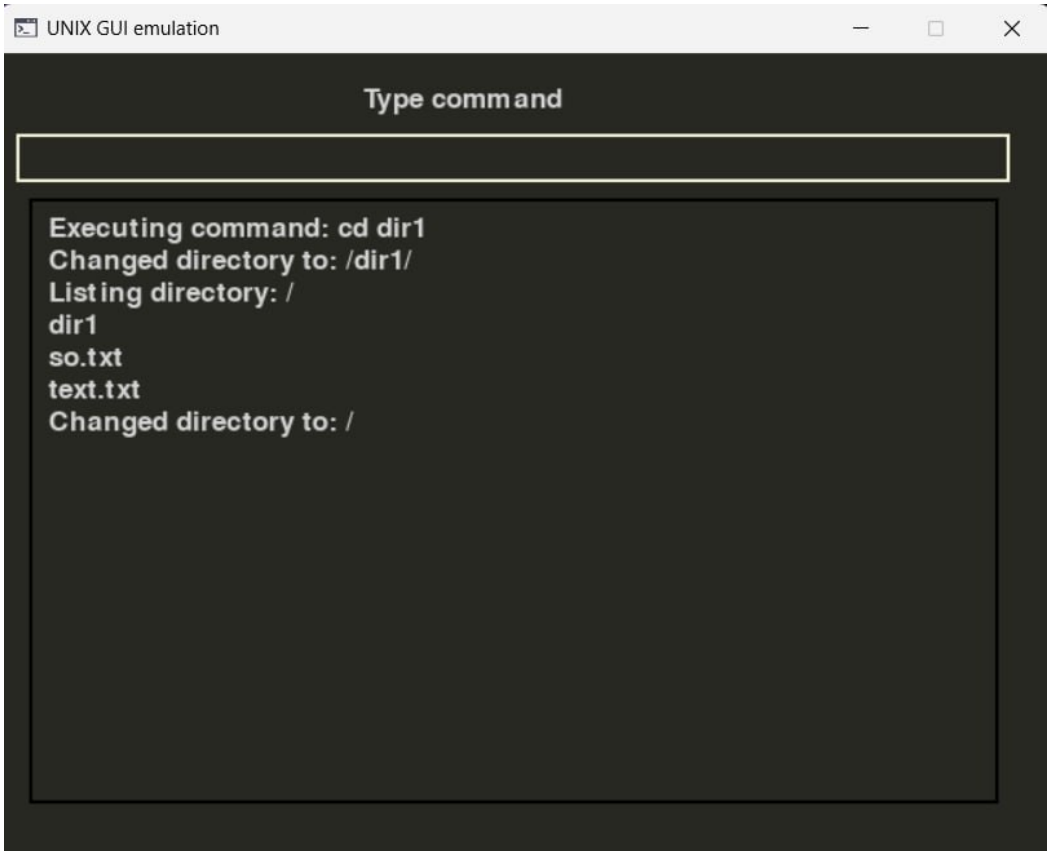

Рисунок 2– Переход в подкаталог и обратно

UC-03. Подсчёт строк/слов/символов

- Команда: wc text.txt

- Ожидаемый результат: корректные значения Lines/Words/Characters для файла в ZIP

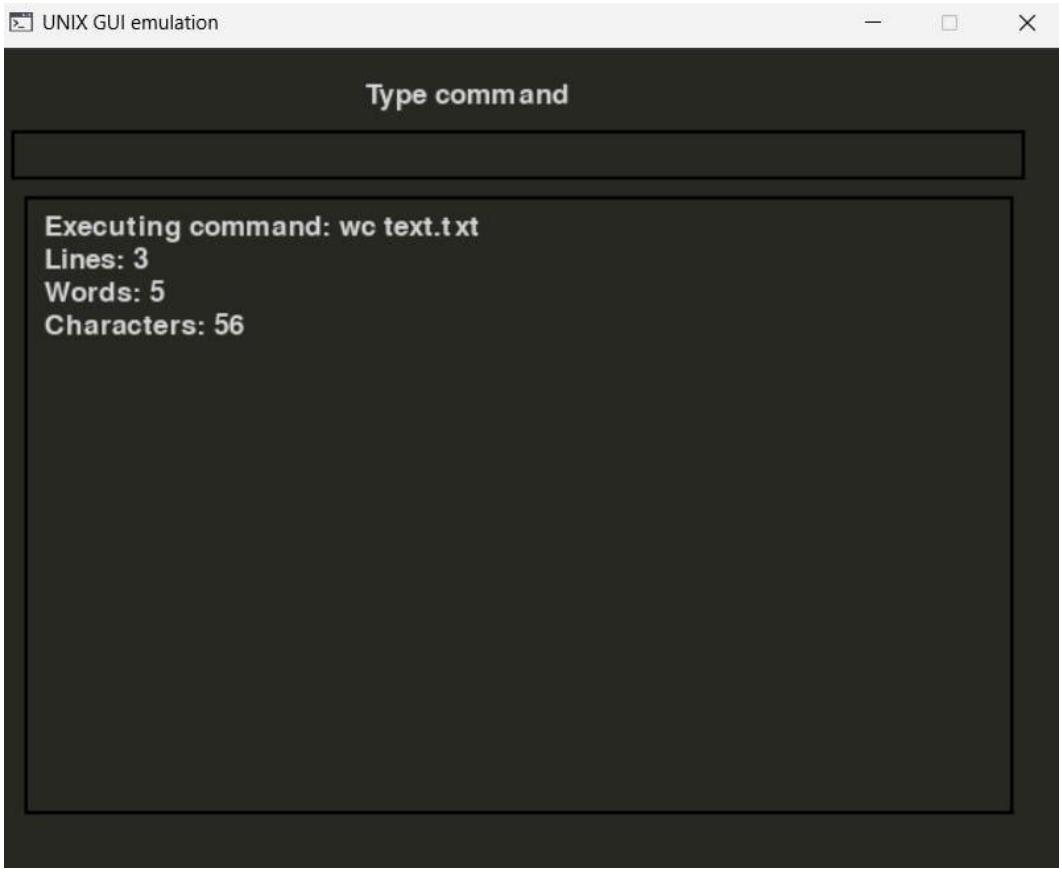

Рисунок 3– Подсчет строк и слов

UC-04. Перемещение/переименование файла

- Команда: mv text.txt dir1/text.txt

- Ожидаемый результат: файл перемещён в systeam/dir1/, подтверждающее сообщение 'Moved ...'.

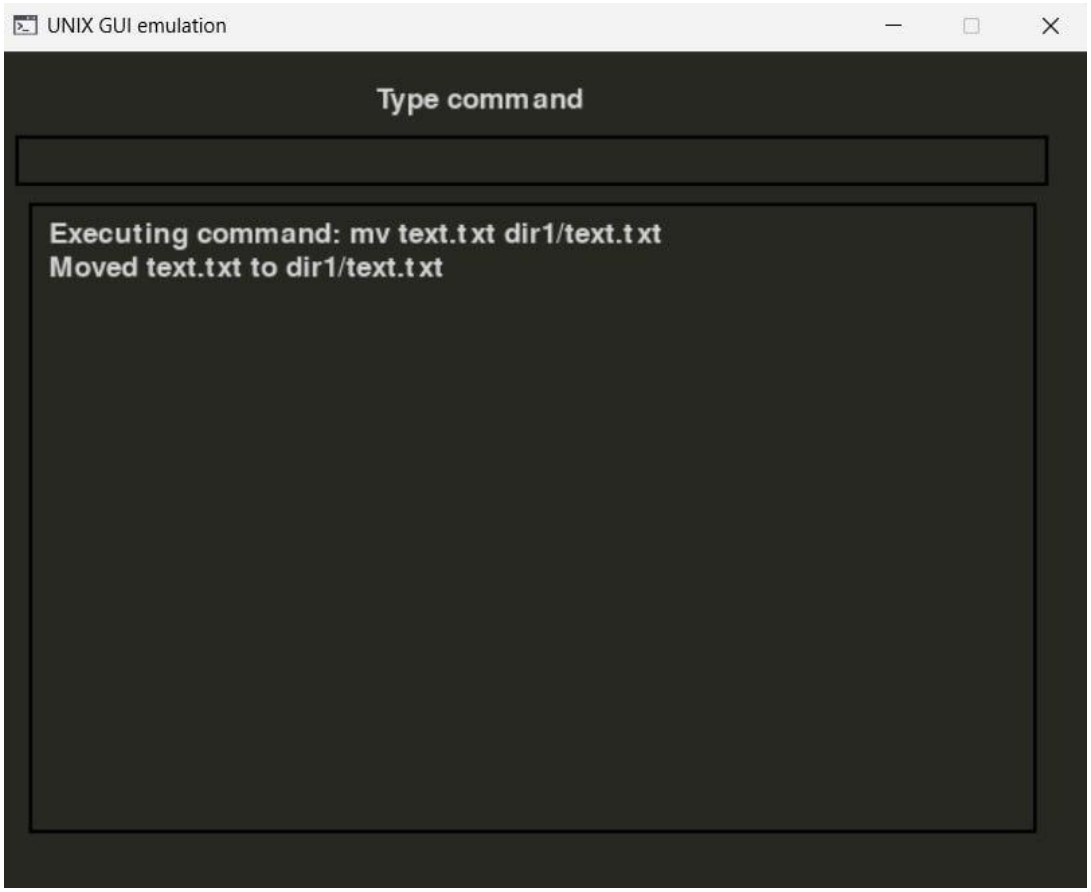

Рисунок 4– Передвижение файла

UC-05. Очистка экрана

- Команда: clear

- Ожидаемый результат: история вывода очищена.

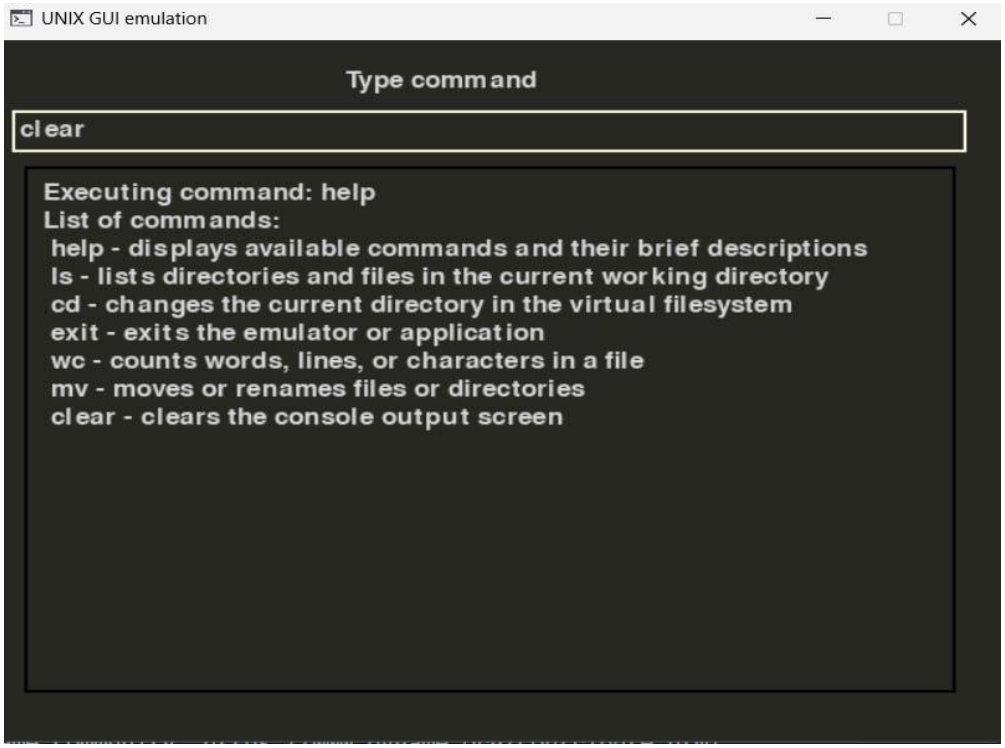

Рисунок 5– Полный экран

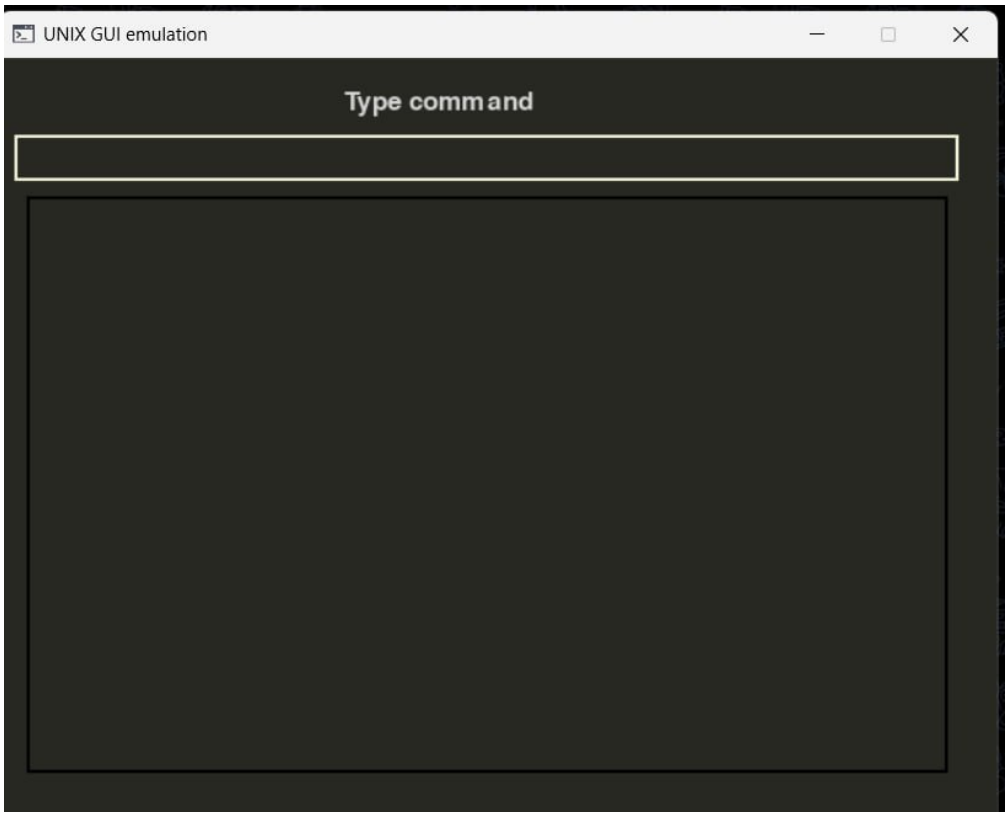

Рисунок 6– Приминение clear

UC-06. Справка

- Команда: help

- Ожидаемый результат: выведен список доступных команд.

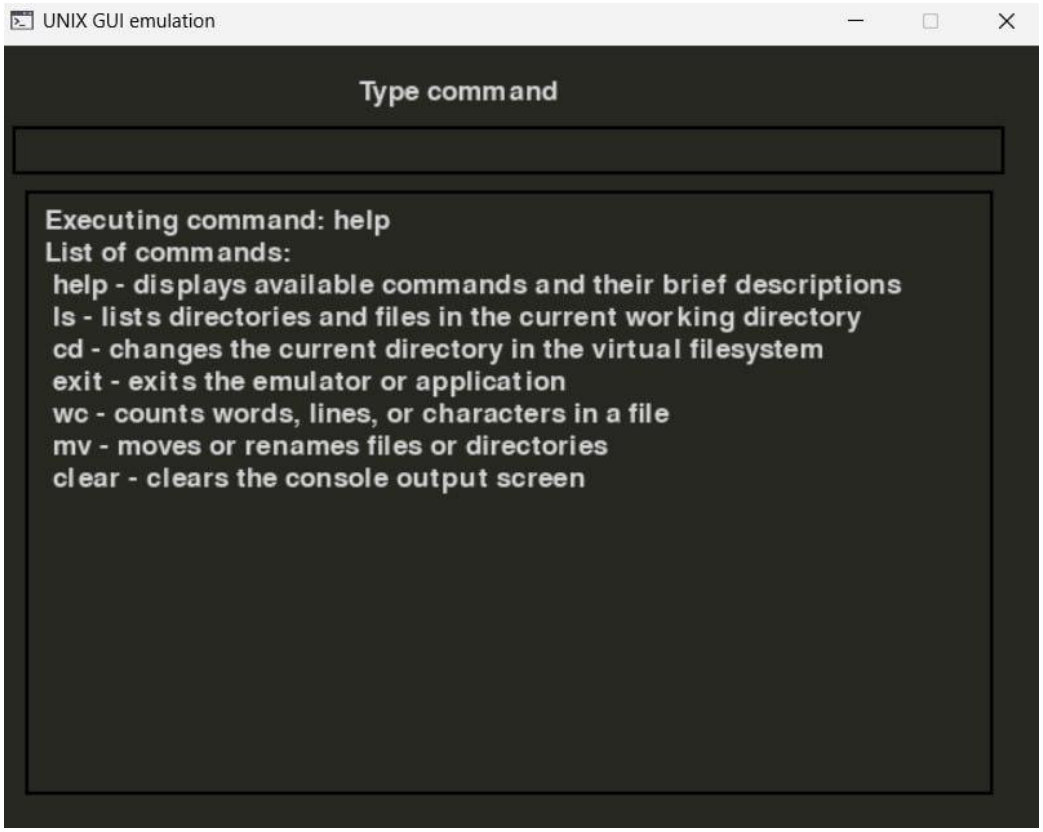

Рисунок 7– Вывод справки

UC-07. Обработка ошибок

- Команда: wc not_exist.txt → ожидается 'ERROR: File ... not found'.

- Команда: cd not_exist → ожидается 'ERROR: Directory not exist'.

- Команда: foo → ожидается 'ERROR: Invalid command'.

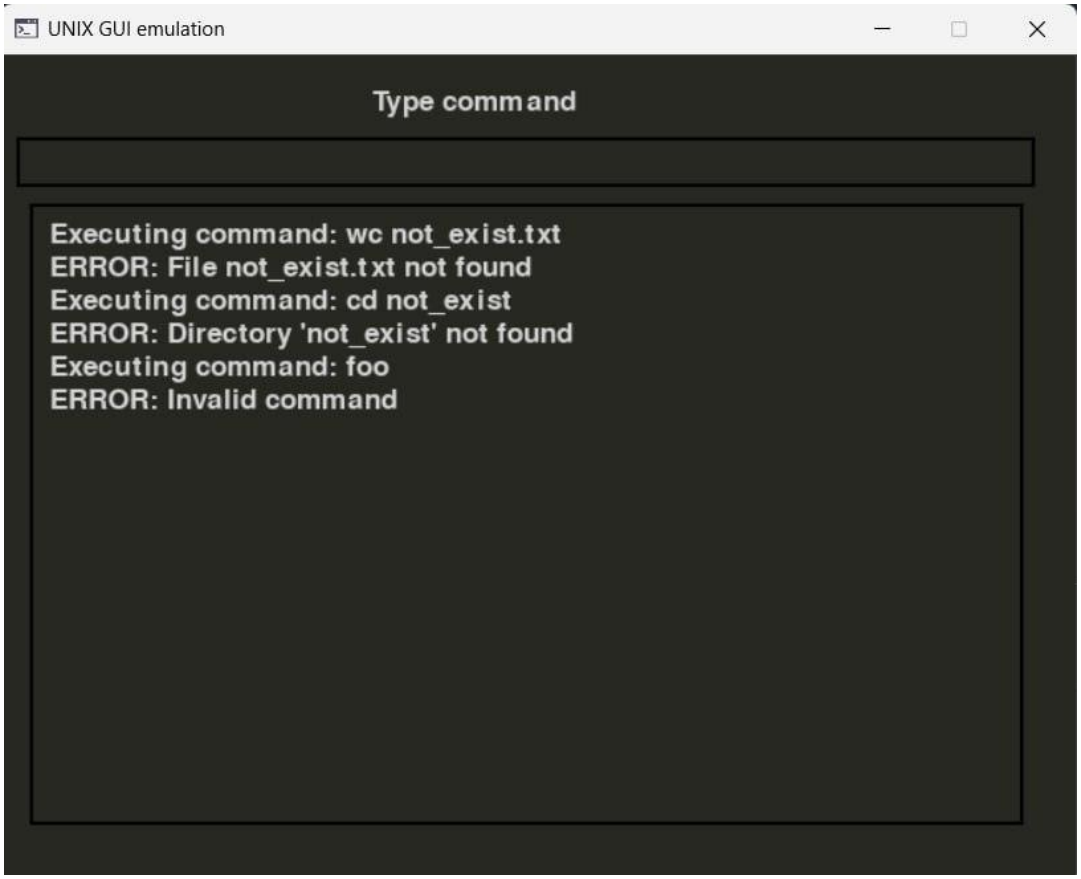

Рисунок 8– Обработка ошибок

UC-08. Автовыполнение скрипта

- Запуск с --script start.sh.

- Ожидаемый результат: в начале работы выполнены команды из файла (в поставке — 'help').

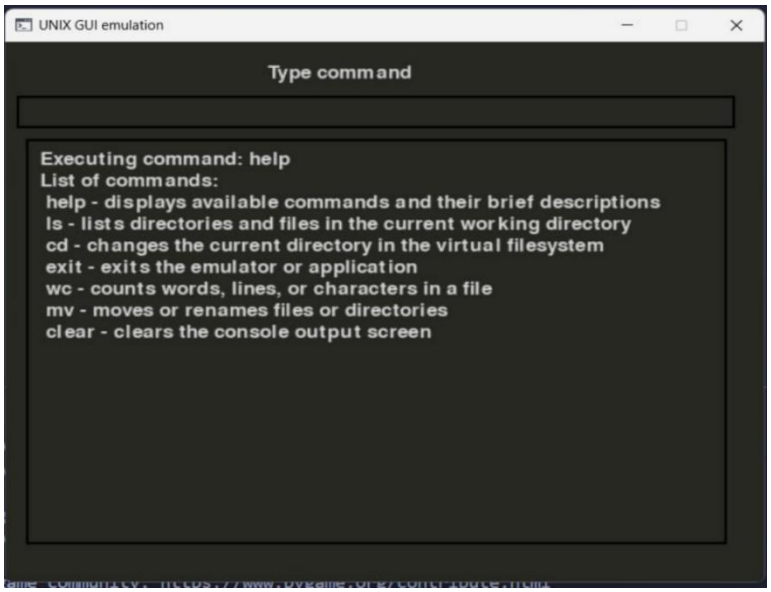

Рисунок 9– Автовыполнение скрипта

2.3 Руководство пользователя

2.3.1 Введение

GUI UNIX OS позволяет тренировочно выполнять базовые команды UNIX в изолированной среде внутри ZIP-архива; реальная файловая система не изменяется

2.3.2 Начало работы

- 1) Установите Python 3.10+ (рекомендуется 3.11).
- 2) Создайте и активируйте виртуальное окружение:
- Windows (PowerShell):
- python -m venv .venv
- .\.venv\Scripts\Activate.ps1
- macOS/Linux (bash):
- python3 -m venv .venv
- source .venv/bin/activate
- 3) Установите зависимости:
- pip install pygame pytest

4) Убедитесь, что архив виртуальной ФС (например, systeam.zip) находится в корне проекта

2.3.3 Основное меню

Область вывода (console): история сообщений и результатов команд.

Поле ввода (input): ввод текста команд; Enter — выполнить, Backspace — удалить символ.

2.3.4 Работа с меню отделов

- help → перечень команд.
- ls → содержимое текущего каталога.
- cd dir1 → перейти в подкаталог; cd .. → вернуться; cd / → корень systeam/.
- wc text.txt → показать Lines/Words/Characters.
- mv text.txt dir1/text.txt → перемещение/переименование.
- clear → очистка вывода.
- exit → выход из приложения

2.3.5 Решение возможных проблем

- ModuleNotFoundError: pygame — установите пакет: pip install pygame.
- Графическое окно не открывается — требуется машина с GUI/X-сервером.
- wc: файл не найден — проверьте путь и наличие в архиве.
- pytest не видит пакет core — запускайте из корня и/или задайте PYTHONPATH=.

2.4 Системная документация

2.4.1 Что вам понадобится

- Python 3.10+ (Windows/macOS/Linux).
- pygame (GUI), pytest (для тестов — опционально).
- Текстовый редактор/IDE (VS Code/PyCharm).

2.4.2 Организация файлов

- main.py — точка входа; принимает аргументы: --user, --archive, --script.
- core/ — логика приложений (console.py, emulator.py, input_box.py).

- source/ — цвета и ресурсы (img/UNIX_GUI_icon.png).
- start.sh — стартовый сценарий (по умолчанию: help).
- system.zip / test_sys.zip — примеры виртуальной ФС (корневой каталог 'system/').
- tests/test_emulator.py — модульные тесты эмулятора.

2.4.3 Организация файлов

Запуск приложения:

python main.py --user Alice --archive systeam.zip --script start.sh

Запуск тестов:

PYTHONPATH=. pytest -q

2.4.4 Взаимодействие с приложением

- Запустите приложение с корректным путём к ZIP и стартовому скрипту.
- После старта выполнится сценарий из --script (в комплекте — команда 'help').
- Введите команды в поле ввода и наблюдайте результаты в области вывода.

3 ОПИСАНИЕ ВНЕСЕННЫХ ИЗМЕНЕНИЙ В ПРОГРАММНЫЙ КОД

В рамках учебного задания в исходный код эмулятора UNIX-подобной ОС были преднамеренно внесены изменения, имитирующие типичные ошибки, возникающие в процессе разработки. Эти модификации затрагивают логику работы, обработку исключений, пользовательский интерфейс и синтаксис. Ниже приведено детальное описание внесенных дефектов.

3.1 Изменения в логике обработки команд

Были изменены ключевые участки кода, отвечающие за парсинг и выполнение команд, что привело к нарушению их работоспособности.

В методе read_command модуля emulator.py была изменена логика разделения строки команды на компоненты. Вместо разделения по пробелу (split(" ")) был установлен разделитель по двум пробелам (split(" ")). В результате команды с аргументами (например, cd systeam или mv file1 file2) перестали корректно обрабатываться, так как вся строка воспринималась как одна неверная команда.

```
135     def read_command(
136         shel_medved_po_lesu_vidit_kamen_a_nem_nadpis_nalevo_pojdyosh_v_mashine_sgorish_napravo_pojdyosh_v_mashine_sgorish_prjamo_pojdyosh_v_mashine_sgorish_poshel_medved_nazad_vidit_k
137         command,
138     ):
139         parts = command.split("  ")
140 
141         if command == "help":
142             shel_medved_po_lesu_vidit_kamen_a_nem_nadpis_nalevo_pojdyosh_v_mashine_sgorish_napravo_pojdyosh_v_mashine_sgorish_prjamo_pojdyosh_v_mashine_sgorish_poshel_medved_nazad_vid
143         elif command == "clear":
144             shel_medved_po_lesu_vidit_kamen_a_nem_nadpis_nalevo_pojdyosh_v_mashine_sgorish_napravo_pojdyosh_v_mashine_sgorish_prjamo_pojdyosh_v_mashine_sgorish_poshel_medved_nazad_vid
145         elif command == "exit":
146             exit()
147         elif command == "ls":
148             shel_medved_po_lesu_vidit_kamen_a_nem_nadpis_nalevo_pojdyosh_v_mashine_sgorish_napravo_pojdyosh_v_mashine_sgorish_prjamo_pojdyosh_v_mashine_sgorish_poshel_medved_nazad_vid
149         elif parts[0] == "cd" and len(parts) > 1:
150             shel_medved_po_lesu_vidit_kamen_a_nem_nadpis_nalevo_pojdyosh_v_mashine_sgorish_napravo_pojdyosh_v_mashine_sgorish_prjamo_pojdyosh_v_mashine_sgorish_poshel_medved_nazad_vid
151                 parts[1]
152             )
153         elif parts[0] == "wc" and len(parts) > 1:
154             shel_medved_po_lesu_vidit_kamen_a_nem_nadpis_nalevo_pojdyosh_v_mashine_sgorish_napravo_pojdyosh_v_mashine_sgorish_prjamo_pojdyosh_v_mashine_sgorish_poshel_medved_nazad_vid
155                 parts[1]
156             )
157         elif parts[0] == "mv" and len(parts) > 1:
158             shel_medved_po_lesu_vidit_kamen_a_nem_nadpis_nalevo_pojdyosh_v_mashine_sgorish_napravo_pojdyosh_v_mashine_sgorish_prjamo_pojdyosh_v_mashine_sgorish_poshel_medved_nazad_vid
159                 parts[1], parts[2]
160             )
161         else:
162             console.text_list.append("ERROR: Invalid command")
163 
```

Рисунок 12 – *Некорректный парсинг команд*

Была изменена проверка количества аргументов для команды mv. Корректная проверка len(parts) > 2 (команда + 2 аргумента) была заменена на неверную len(parts) > 1. Это привело к тому, что в случаях, если с коммандой mv

ввели один аргумент вместо двух может происходить ошибка, которая не проверяется, потому что в таком случае вместо попадания в обработчик команды mv программа должна была сообщить о неверной команде.

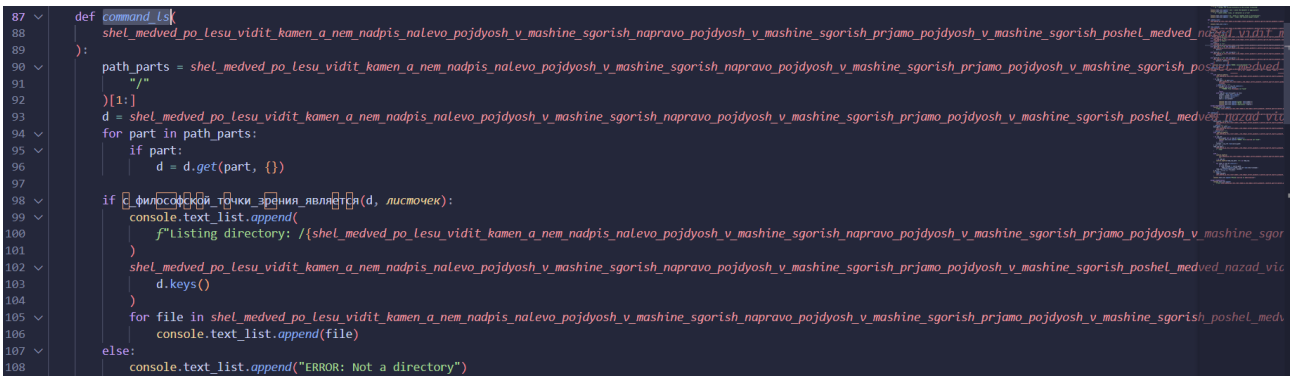

Рисунок 13 – Ошибка в проверке количества аргументов для команды mv

В методе command_ls была изменена проверка типа структуры данных, представляющей директорию. Вместо корректной проверки isinstance(d, dict) была внедрена проверка на тип list. В результате при попытке просмотра содержимого любой директории функция всегда возвращала ошибку "Not a directory", так как тип данных не соответствовал ожидаемому в ошибочной логике.

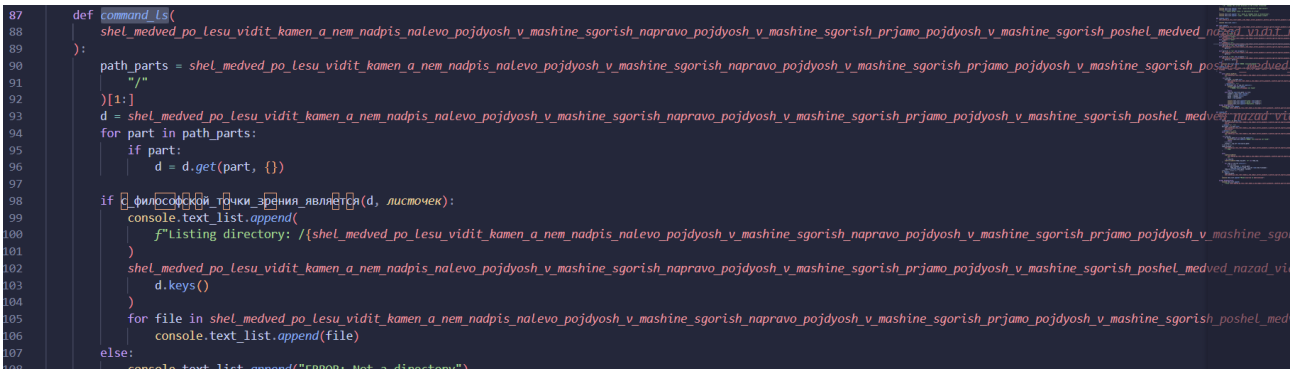

Рисунок 14 – Неверная проверка типа данных для команды ls

3.2 Изменения в обработке исключений

Была ослаблена система обработки ошибок, связанных с работой с файлами, что снизило отказоустойчивость приложения.

В методах, работающих с ZIP-архивом (open_zip_sys, command_wc, command_mv), корректная обработка исключения zipfile.BadZipFile была заменена на обработку другого,

неподходящего исключения StopIteration. Вследствие этого, если пользователь указывал поврежденный или не являющийся ZIP-архивом файл, приложение аварийно завершалось с необработанным исключением BadZipFile вместо вывода информативного сообщения об ошибке.

```
    filename,
):
    try:
        with zipfile.ZipFile(
            shel_medved_po_lesu_vidit_kamen_a_nem_nadpis_nalevo_po
            "r",
        ) as zip_ref:
            file_path = os.path.join(
                shel_medved_po_lesu_vidit_kamen_a_nem_nadpis_nalev
                filename,
            ).replace("\\", "/")
            if file_path not in zip_ref.namelist():
                console.text_list.append(
                    f"ERROR: File {filename} not found"
                )
                return
            with zip_ref.open(file_path) as file:
                content = file.read().decode()
                lines = content.splitlines()
                words = content.split()
                chars = len(content)

                console.text_list.append(f"Lines: {len(lines)}")
                console.text_list.append(f"Words: {len(words)}")
                console.text_list.append(f"Characters: {chars}")
    except StopIteration:
        console.text_list.append(
            f"Файл {shel_medved_po_lesu_vidit_kamen_a_nem_nadpis_na
        )
```

Рисунок 15 – Неверная обработка ошибок работы с ZIP-архивом

3.4 Внесение синтаксических ошибок

В параметрах конструктора класса ConsoleOutput пропущена запятая перед параметром font_size.

```
class Co
    def
        font_size,
        x,
        y,
        w,
        h,
    ):
```

SyntaxError: Expected ',', found name Ruff
invalid syntax Mypy(syntax)
View Problem (Alt+F8) Quick Fix... (Ctrl+.) Fix (Ctrl+I)

Рисунок 16 – Синтаксическая ошибка в конструкторе ConsoleOutput

В теле конструктора Emulator при присвоении атрибута current_dir использовалось двойное равенство вместо одинарного.

```
_nee_i_sgorel.archive_path = archive_path
_nee_i_sgorel.current_dir == "system/"
_nee_i_sgorel.files_list = []
_nee_i_sgorel.file_structure = {}
_nee_i_sgorel.open_zip_sys()


_nee_i_sgorel,
```

Рисунок 17 – Синтаксическая ошибка в конструкторе Emulator

3.5 Внесение ошибок в интерфейс

В интерфейсе были допущены 2 ошибки расположения элементов. Вводимый пользователем текст располагается вне поля для ввода. Также и с основным текстом консольного интерфейса – он находится правее, чем необходимо и в некоторых случаях обрезается вместо переноса строки.

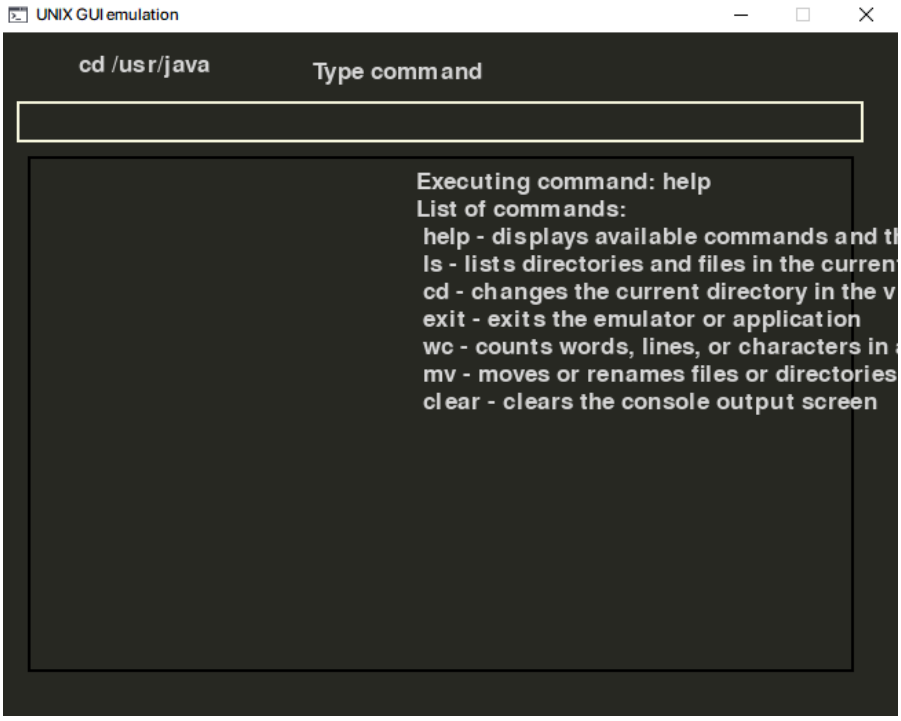

Рисунок 18 – Ошибки в интерфейсе

4 ТЕХНИЧЕСКОЕ ЗАДАНИЕ И АНАЛИЗ ДОКУМЕНТАЦИИ ПРОГРАММНОГО ПРОДУКТА ДРУГОЙ КОМАНДЫ

4.1 Техническое задание к программному продукту

Программный продукт «Менеджер задач» представляет собой кроссплатформенное настольное приложение, разработанное для управления персональными и командными задачами. Продукт предназначен для работы в операционных системах Windows, Linux и macOS. Основная цель — предоставить пользователям интуитивно понятный инструмент для создания, отслеживания, фильтрации и архивирования задач, повышая личную и групповую продуктивность.

Ключевые заявленные функции включают: создание задач с заголовком, описанием, сроком и приоритетом; расширенное редактирование с поддержкой rich-text; множественное удаление; сортировку и фильтрацию по различным параметрам; а также импорт и экспорт данных в формате JSON для обеспечения переносимости.

Основанием для разработки служит потребность рынка в автономном, безопасном и функциональном решении, не зависящем от постоянного интернет-соединения и облачных сервисов, что обеспечивает максимальную конфиденциальность пользовательских данных.

4.2 Анализ технической документации

В ходе анализа было изучено предоставленное техническое задание (ТЗ) версии 1.0. Была проведена оценка полноты, логичности и непротиворечивости требований.

4.3 Оценка полноты технического задания

Техническое задание структурировано и охватывает основные аспекты продукта, однако при детальном рассмотрении выявляется ряд существенных пробелов:

- Отсутствие требований к обработке граничных и ошибочных ситуаций. В документе не описано поведение системы в следующих случаях:
  - Ввод данных, превышающих установленные лимиты (например, заголовок более 255 символов).
  - Попытка создания задачи без обязательных полей (например, без заголовка).
  - Ввод некорректных форматов данных (например, текст в поле даты "YYYY-MM-DD").
  - Действия пользователя с пустым списком задач (например, попытка сортировки или поиска).
  - Импорт поврежденного, пустого или некорректно структурированного JSON-файла.
- **Недостаточная спецификация формата данных.** Требование «Экспорт в JSON» не подкреплено схемой, примером структуры или описанием полей. Это создает неопределенность для разработки и тестирования функций импорта/экспорта. Неясно, как должны обрабатываться специальные символы, переносы строк и rich-text форматирование при сериализации в JSON.
- **Отсутствие нефункциональных требований.** Кроме общих метрик производительности, в ТЗ отсутствуют требования к безопасности (например, шифрование локального файла с задачами), удобству использования (поддержка горячих клавиш, доступность для людей с ограниченными возможностями) и восстановлению после сбоев (механизмы автоматического сохранения).
- **Слабая детализация требований к пользовательскому интерфейсу.** Отсутствуют эскизы, макеты или даже словесное описание

компоновки основного окна, что ведет к риску создания нелогичного и неудобного интерфейса.

4.4 Оценка логичности и согласованности

Требования в целом непротиворечивы, однако некоторые формулировки допускают двойственное толкование:

- **Неоднозначность поиска.** Указан «поиск с поддержкой полнотекстового индексирования», но не определены его параметры: должен ли он быть регистронезависимым, поддерживать ли поиск по части слова, искать ли только в заголовках или также в описаниях.
- **Неконкретные требования к rich-text.** Не указан точный перечень поддерживаемых тегов форматирования (например, жирный, курсив, списки, ссылки, таблицы). Это может привести к несоответствию между ожиданиями пользователя и реализацией.
- **Неясный механизм «Архивирования».** В пункте 4.1.1 упоминается «Архивирование выполненных задач с возможностью восстановления», но в остальном документе нет никаких деталей о том, как этот механизм должен работать, где хранится архив и как выглядит интерфейс для восстановления.

4.5 Соответствие реализации требованиям ТЗ

Тестирование выявило расхождения между заявленными требованиями и фактической реализацией.

Таблица 4.5.1 – Соответствие реализации требованиям ТЗ

Требование ТЗ	Соответствие	Примечание
Множественное удаление задач	Не выполняется	При выделении нескольких задач и нажатии "Удалить" удаляется только первая запись в списке выделенных, а не все.

Продолжение таблицы 4.5.1

Экспорт в JSON	Выполняется	Функция экспорта создает JSON-файл с нарушенной структурой (незакрытые скобки), что делает его непригодным для последующего импорта.
Сортировка по сроку	Частично выполняется	Сортировка работает некорректно, так как обрабатывает даты как строки, а не как объекты дат (например, "10.01.2025" будет идти раньше "09.02.2025").
Сохранение данных между сеансами	Не выполняется	Приложение не сохраняет данные автоматически при закрытии. Для сохранения требуется ручное действие (например, через меню "Файл -> Сохранить"), которое не описано в ТЗ и не является очевидным.

4.6 Рекомендации по улучшению документации

Детализация требований:

- Добавить раздел «Обработка исключительных ситуаций» с таблицей, описывающей реакцию системы на некорректные действия пользователя и сбои.
- Включить в ТЗ точную схему JSON-файла с примером для функций импорта/экспорта.
- Четко определить алгоритм поиска (регистронезависимый, по всем

текстовым полям) и перечень поддерживаемых rich-text тегов.

Визуализация:

- Приложить к ТЗ эскизы (wireframes) основного окна и диалоговых окон для обеспечения единого видения интерфейса.
- Добавить диаграмму компонентов верхнего уровня для описания архитектуры.

5 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ПРОДУКТА

В процессе функционального, юзабилити и нагрузочного тестирования программного продукта «Менеджер задач» был выявлен ряд дефектов, которые существенно влияют на корректность работы, удобство использования и общую стабильность приложения.

5.1 Классификация выявленных дефектов

Выявленные дефекты были классифицированы по их критичности:

- Критический
- Высокий
- Средний
- Низкий

А также по типу:

- Функциональный : несоответствие поведения системы заявленным или ожидаемым требованиям.
- Дефект интерфейса: визуальные дефекты, некорректное отображение элементов интерфейса.
- Дефект валидации: ошибки, связанные с некорректной проверкой вводимых данных или данных, поступающих извне.
- Производительность: дефекты, связанные с низкой скоростью работы, высоким потреблением ресурсов или нестабильностью под нагрузкой.
- Ошибки данных: нарушение целостности или консистентности данных.

Таблица 5.1.1 – Классификация выявленных дефектов

№	Описание дефекта	Критичность	Тип
1	Функционал кнопок "Добавить задачу" и "Редактировать выбранную" перепутан	Высокий	Функциональный

Продолжение таблицы 5.1.1

2	Неправильное распознавание формата даты	Высокий	Валидация
3	Ошибка удаления задач после создания/удаления нескольких действий	Высокий	Функциональный
4	Сброс статуса всех задач в "Невыполнено" при перезапуске	Высокий	Функциональный, ошибка данных
5	Отсутствие валидации даты срока (срок в прошлом)	Средний	Ошибка данных, валидация
6	Пересечение элементов интерфейса при уменьшении окна	Средний	Интерфейс
7	Отсутствие валидации JSON при импорте (пустые заголовки/даты)	Высокий	Ошибка данных, валидация
8	Замедление работы и высокое потребление ОЗУ при значительном количестве добавленных задач	Критический	Производительность
9	Приложение сильно лагает при множественном нажатии "Добавить задачу"	Высокий	Производительность
10	Поиск задач чувствителен к регистру	Средний	Функциональный
11	Возможность добавить несколько дублирующихся задач	Низкий	Функциональный
12	Кнопка "Сохранить изменения" остается активной после сохранения	Низкий	Интерфейс

5.2 Детальное описание дефектов

В ходе тестирования выявлен функциональный дефект, заключающийся в инверсии логики работы кнопок "Добавить задачу" и "Редактировать выбранную". Активация кнопки "Добавить задачу" приводит к открытию формы редактирования, предзаполненной данными существующей задачи.

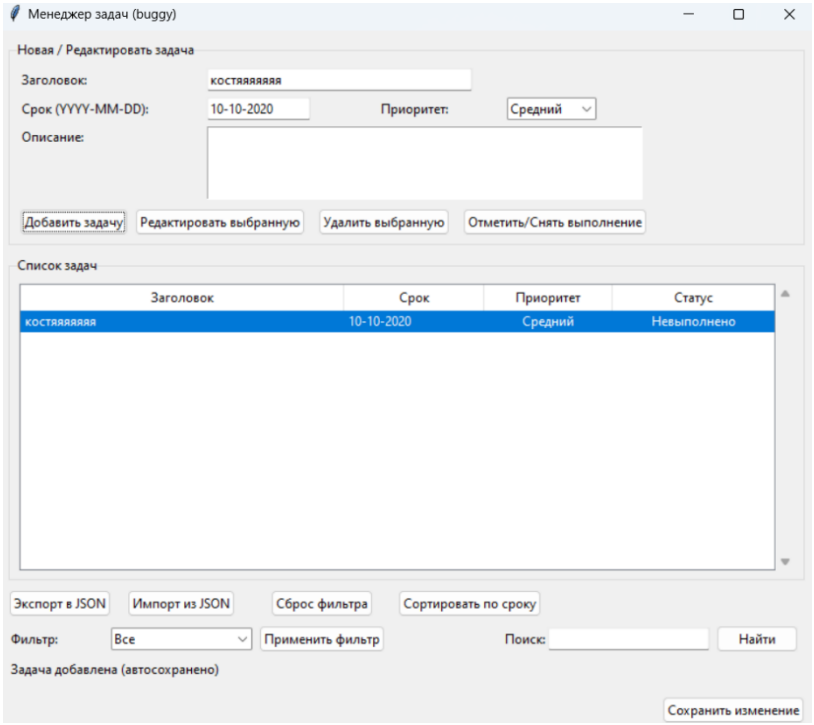

Рисунок 19 – Результат нажатия на кнопку «Добавить задачу»

Тогда как нажатие "Редактировать выбранную" инициирует открытие пустой формы создания новой задачи. Данное несоответствие нарушает базовые пользовательские сценарии и снижает интуитивность интерфейса.

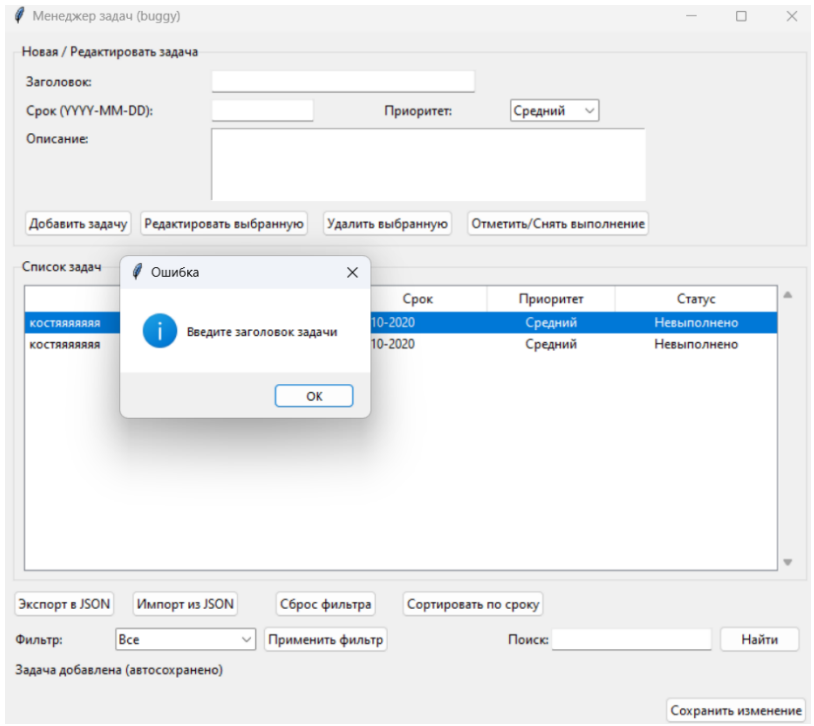

Рисунок 20 – Результат нажатия на кнопку «Редактировать выбранную»

Обнаружена ошибка в обработке форматов даты. Несмотря на заявленный в интерфейсе формат YYYY-MM-DD, приложение фактически ожидает и валидирует даты в формате DD-MM-YYYY. Ввод даты в формате YYYY-MM-

DD генерирует ошибку "Неверный формат даты", тогда как ввод в DD-MM-YYYY принимается корректно, что свидетельствует о внутренней несогласованности валидации данных.

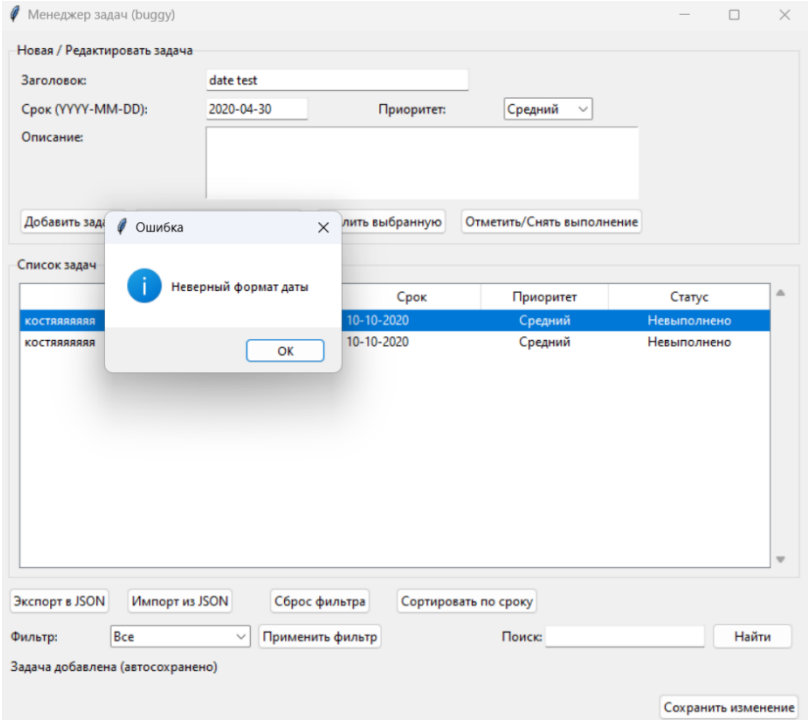

Рисунок 21 – Результат попытки ввести дату в указанном формате YYYY-MM-DD

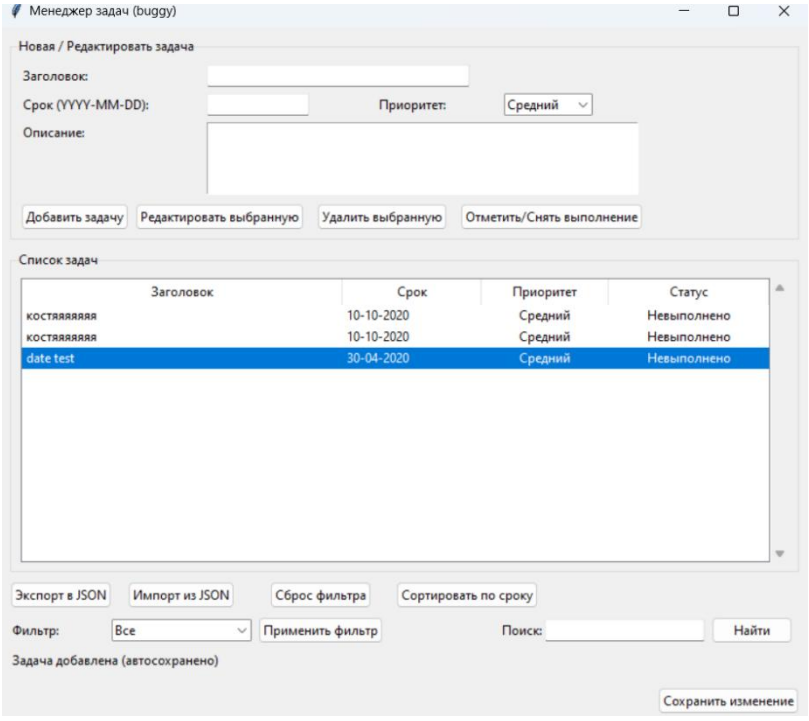

Рисунок 22 – Успешное добавление заметки с датой в формате DD-MM-YYYY

Зафиксирован функциональный дефект в механизме удаления задач, проявляющийся после выполнения множественных операций создания и удаления. После 5-7 циклов создания/удаления задач, попытка удалить любую

задачу приводит к отображению предупреждения "Неверный выбор" и блокировке функции удаления. Восстановление функциональности требует перезапуска приложения.

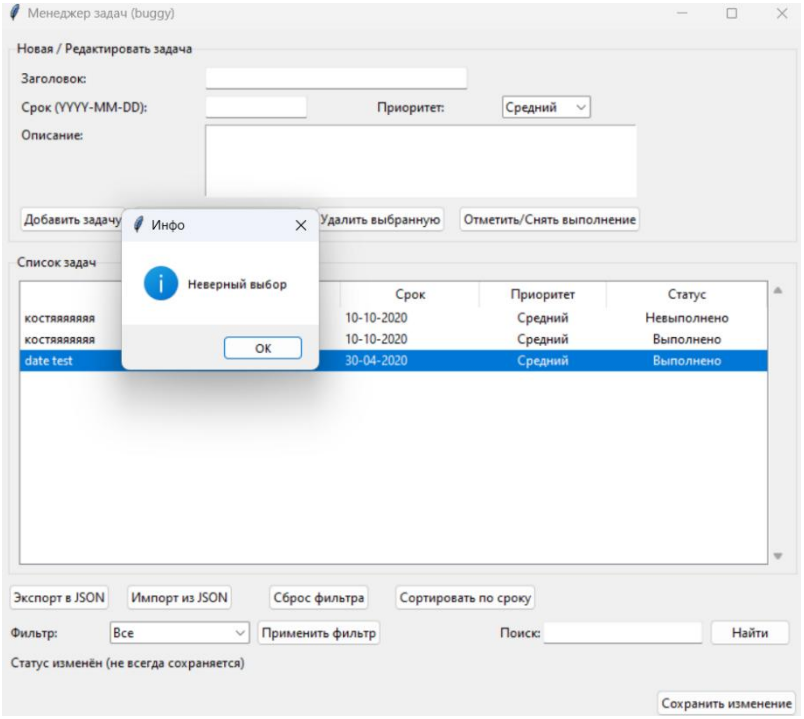

Рисунок 23 – Попытка удалить заметку

Выявлена проблема с сохранением состояния данных: при перезапуске приложения статус всех задач, включая те, что были помечены как "Выполнено", сбрасывается в исходное состояние "Невыполнено". Это ведет к потере информации о прогрессе выполнения задач и нарушает целостность хранимых данных между сеансами работы.

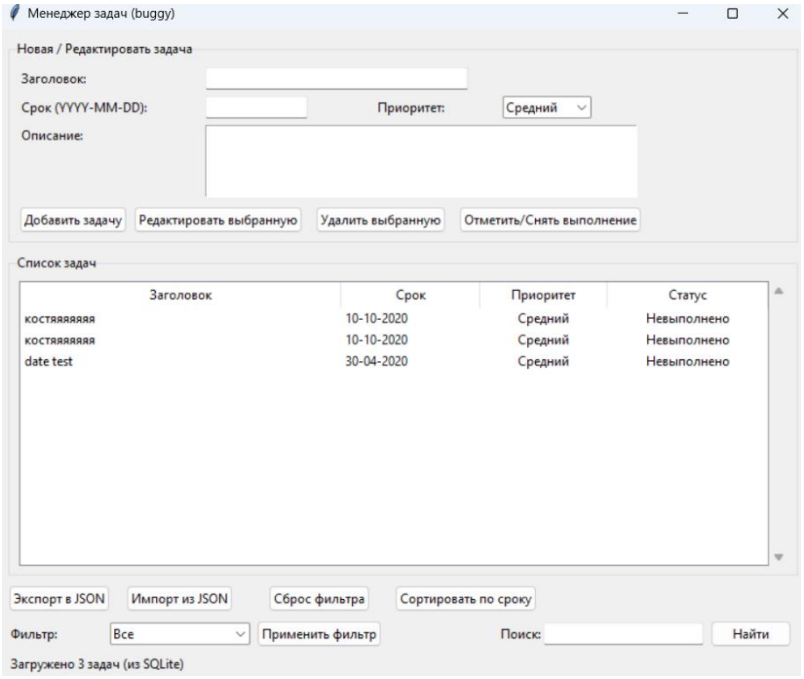

Рисунок 24 – Статусы всех заметок сбросились после перезапуска приложения

Отсутствует валидация данных в поле "Срок", что позволяет пользователю указывать прошедшие даты для выполнения задач. Приложение не предоставляет никаких предупреждений или блокировок при попытке сохранения задачи со сроком в прошлом, что может привести к некорректному планированию и управлению задачами. Например, можно добавит задачу с крайним сроком 9 ноября 1938 года (см. Рисунок 25)

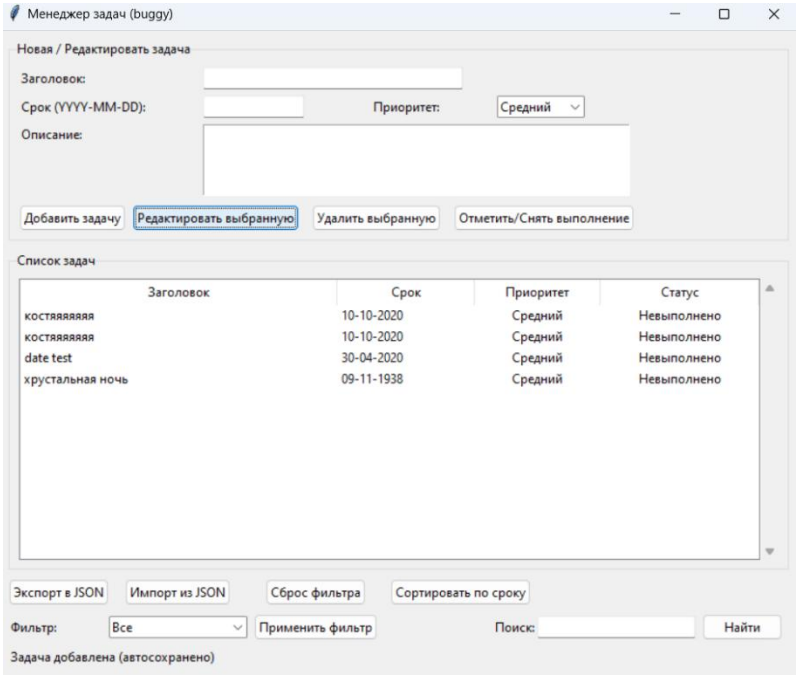

Рисунок 25 – Добавленная заметка с крайним сроком в прошлом

При уменьшении размеров окна приложения наблюдаются дефекты

отображения элементов пользовательского интерфейса. Логотип и кнопки управления окном (свернуть/развернуть/закрыть), а также функциональные кнопки (например, "Найти") начинают пересекаться или обрезаться, что снижает юзабилити и доступность элементов управления.

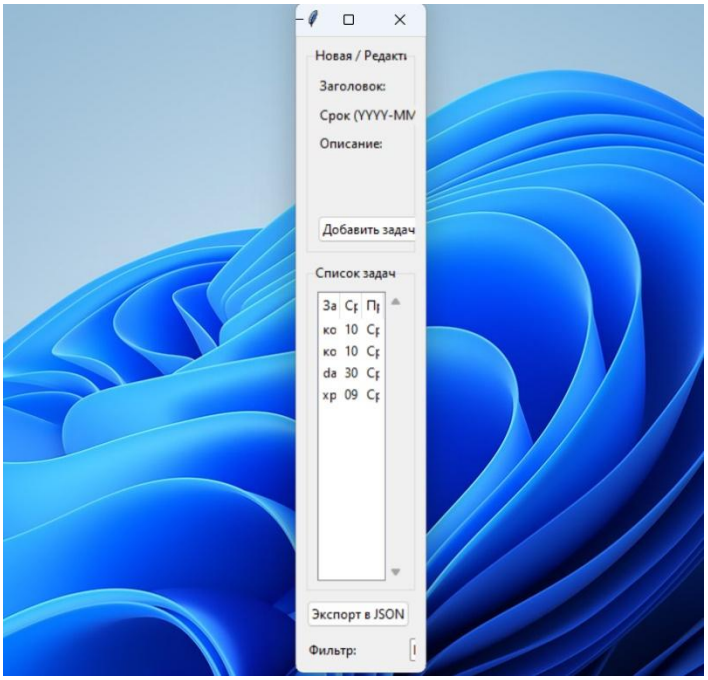

Рисунок 26 – Ошибки в интерфейсе

Функция импорта JSON-файлов обладает недостаточной валидацией содержимого. Приложение позволяет импортировать задачи с пустыми или отсутствующими значениями для обязательных полей, таких как "заголовок" или "срок". Это нарушает целостность данных и может приводить к появлению некорректных записей в списке задач.

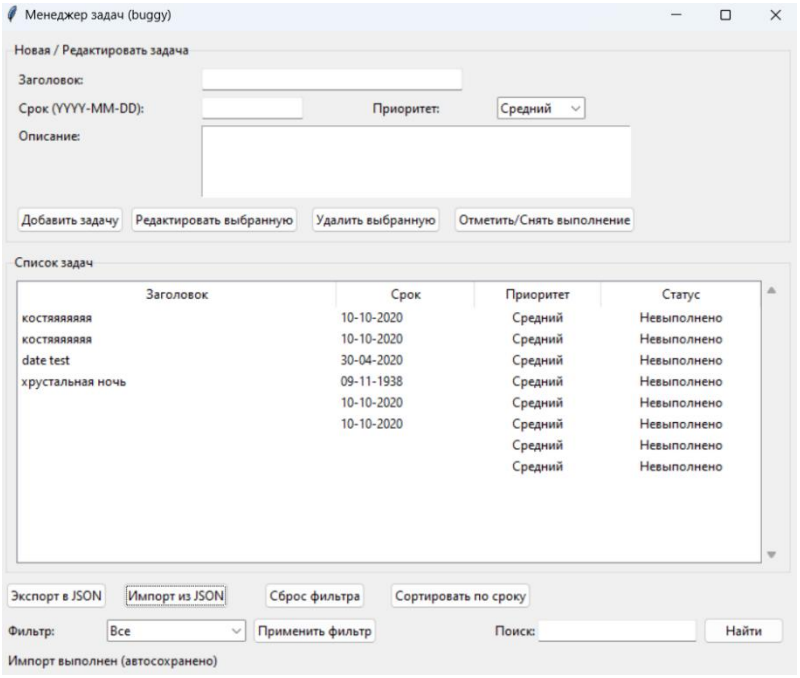

Рисунок 27 – Добавлены задачи с пустыми заголовками и сроками

При добавлении в приложение более 1.000.000 задач наблюдается критическое снижение производительности. Интерфейс демонстрирует значительные задержки, а потребление оперативной памяти возрастает до аномально высоких значений, что делает приложение непригодным для работы с большими объемами данных.

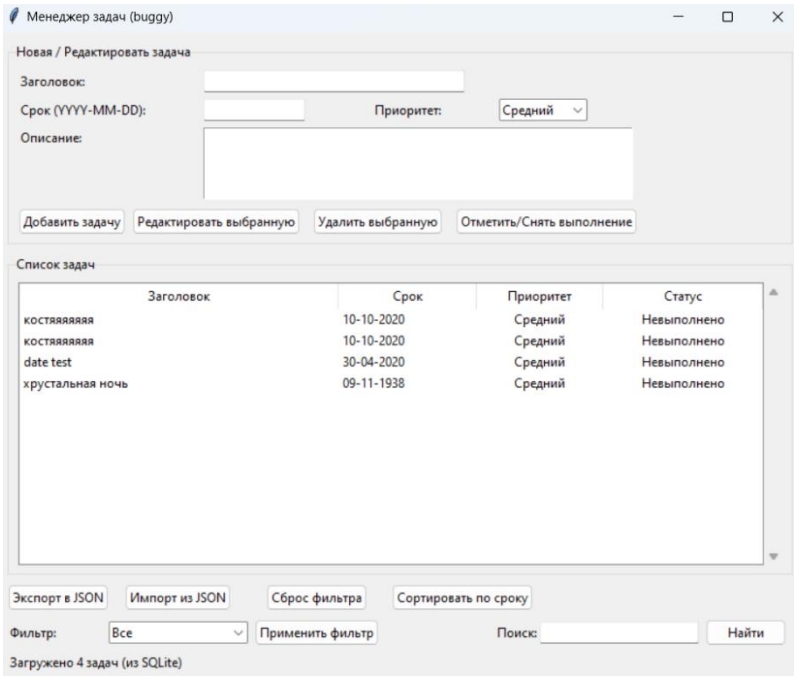

Рисунок 28 – Интерфейс тормозит

Множественное нажатие на кнопку "Добавить задачу" вызывает значительное замедление работы приложения и его "зависание". Это приводит к

неспособности приложения обрабатывать пользовательский ввод и требует принудительного завершения работы.

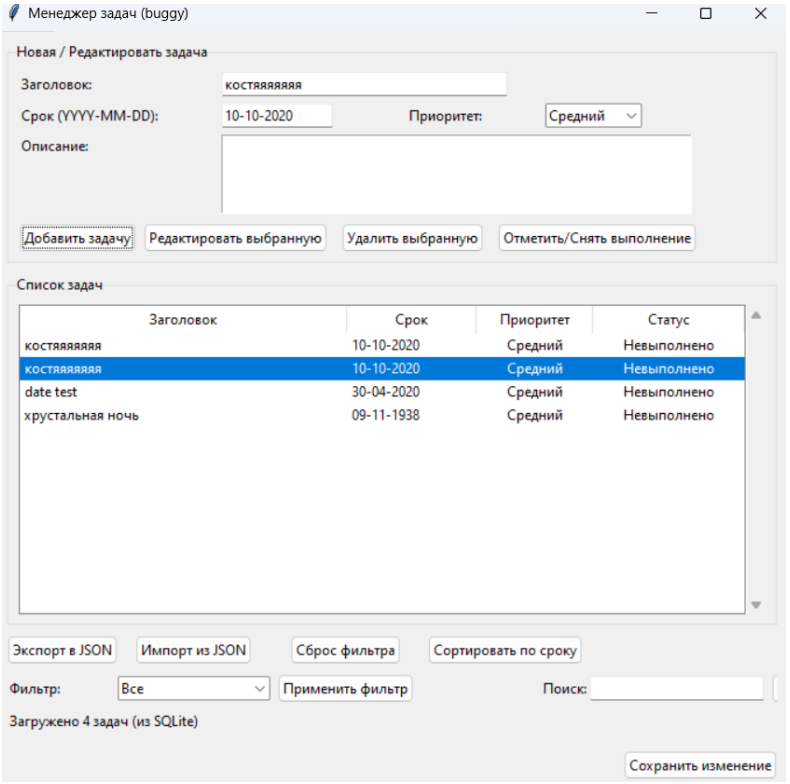

Рисунок 29 – Интерфейс тормозит

Функция поиска задач реализована с учетом регистра символов. Поиск не является регистронезависимым, что снижает эффективность и удобство использования, так как не позволяет найти совпадения при неточном вводе регистра символов.

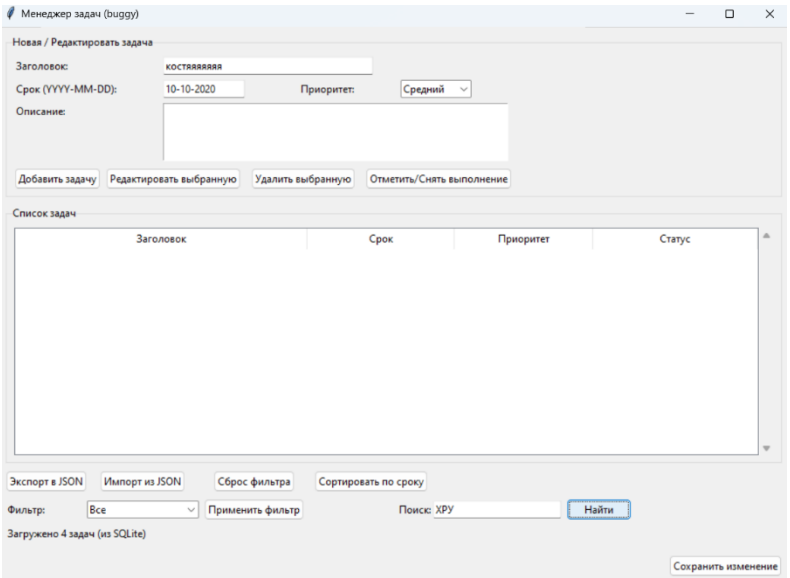

Рисунок 30 – Заметку с названием «Хрустальная ночь» не удалось найти из-за неверного регистра поискового запроса

Отсутствует механизм предотвращения создания дублирующихся задач. Приложение позволяет добавлять несколько задач с идентичными параметрами (заголовок, описание, срок), что приводит к появлению избыточных записей и снижает упорядоченность в списке задач.

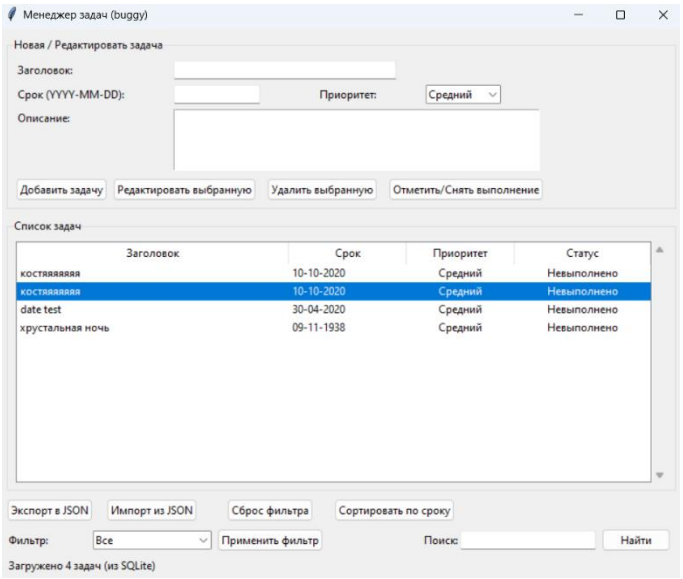

Рисунок 31 – Две идентичные задачи «костяяяяяяя»

После успешного сохранения изменений в форме редактирования задачи кнопка "Сохранить изменения" остается активной, не предоставляя пользователю явной визуальной обратной связи о завершении операции сохранения. Это может ввести в заблуждение относительно статуса сохранения данных.

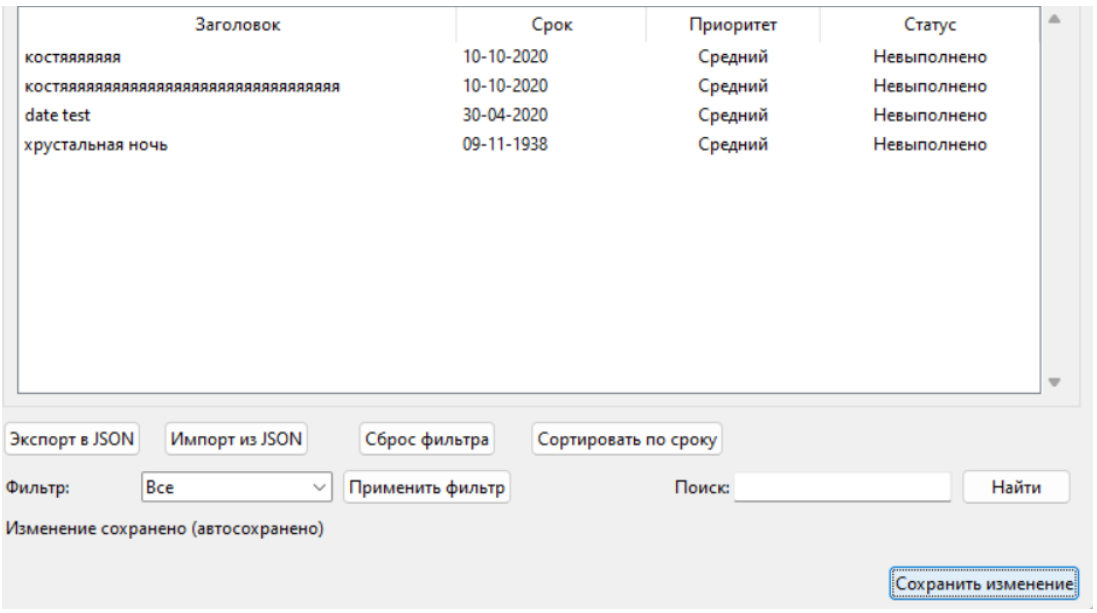

Рисунок 32 – Кнопка «Сохранить изменение» осталась активной после сохранения изменений

5.3 Общая оценка качества

На основании комплексного анализа выявленных дефектов, представленных в разделе 5.2, общая оценка качества программного продукта «Менеджер задач» на текущий момент признается неудовлетворительной.

Существенное количество дефектов классифицировано как критические и высокие, что указывает на фундаментальные проблемы в ключевых аспектах функционирования продукта. В частности, нестабильность базовых операций (например, некорректная работа кнопок добавления/редактирования, сбои при удалении задач), нарушения целостности данных (сброс статусов, отсутствие валидации при импорте), а также критические проблемы с производительностью при работе с большими объемами информации (замедление при 1.000.000+ задач) препятствуют полноценному использованию продукта по его прямому назначению. Кроме того, обнаружены значительные недочеты в пользовательском интерфейсе и эргономике, снижающие удобство взаимодействия с приложением.

Выявленные расхождения между заявленными требованиями (см. раздел 4) и фактической реализацией, в сочетании с перечисленными дефектами, свидетельствуют о необходимости проведения масштабной доработки продукта. Требуется приоритетное устранение критических и высокоприоритетных дефектов, пересмотр архитектурных решений для обеспечения масштабируемости и стабильности, а также существенное улучшение качества пользовательского интерфейса и логики обработки данных. Текущее состояние продукта не позволяет рекомендовать его к развертыванию или активной эксплуатации.

6 АНАЛИЗ ДОКУМЕНТАЦИИ ПРОГРАММНОГО ПРОДУКТА

6.1 Общая характеристика документации

Документация к «Менеджеру задач» представлена техническим заданием. Документ закладывает основу для понимания концепции продукта, но его уровень детализации недостаточен для производственного цикла разработки и тестирования, что является первопричиной многих обнаруженных дефектов.

6.2 Выявленные недостатки документации

На уровне технического задания:

- Критическое упущение описания сценариев обработки ошибок и нештатных ситуаций.
- Крайне общие и неточные формулировки функциональных требований (поиск, rich-text, экспорт).
- Полное отсутствие визуальных материалов (макетов, схем), что ведет к субъективной трактовке требований к интерфейсу.

На уровне (предполагаемого) руководства пользователя:

- Отсутствие детальных инструкций и примеров использования нетривиальных функций (импорт/экспорт, архивация).
- Отсутствие раздела «Часто задаваемые вопросы» или «Решение проблем», который мог бы помочь пользователю справиться с ожидаемыми трудностями.

6.3 Влияние недостатков документации на тестирование

Недостатки документации напрямую и негативно повлияли на процесс тестирования:

- **Увеличение трудозатрат:** Отсутствие четких требований заставило команду тестирования самостоятельно предполагать ожидаемое поведение системы, особенно в негативных сценариях. Это привело к разработке избыточных тест-кейсов для покрытия всех возможных трактовок.
- **Неоднозначность результатов:** При обнаружении дефектов, не описанных в ТЗ (например, регистрозависимый поиск), возникали споры о том, является ли это ошибкой или особенностью реализации. Четкие требования устранили бы эту двусмысленность.
- **Невозможность полноценного тестирования:** Без спецификации формата JSON тестирование функции импорта было ограничено лишь простейшими случаями, без возможности проверить обработку сложных структур или граничных значений.

6.4 Выводы по документации

Представленная документация является лишь начальным наброском и требует серьезной и всесторонней доработки. Для обеспечения эффективного процесса разработки и создания качественного продукта необходимо дополнить ТЗ детализированными спецификациями, сценариями обработки ошибок, макетами интерфейса и четкими критериями приемки. Устранение указанных недостатков позволит снизить риски, сократить время на разработку и тестирование, а также значительно повысить качество конечного программного продукта.

ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы был проведен комплексный анализ процесса тестирования программного обеспечения на примере двух учебных проектов: эмулятора UNIX-подобной ОС («GUI UNIX OS») и «Менеджера задач». Работа позволила закрепить теоретические знания и приобрести практические навыки в области тестирования по методу «чёрного ящика», документирования требований, выявления дефектов и оценки качества ПО.

Основные результаты работы включают:

Разработано и протестировано техническое задание для эмулятора «GUI UNIX OS», что позволило понять важность четкой и полной спецификации требований на ранних этапах разработки.

Внесены и документированы преднамеренные дефекты в код эмулятора, что дало возможность отработать методики поиска и описания ошибок, а также оценить их влияние на функциональность системы.

Проведен критический анализ технического задания и реализации программного продукта «Менеджер задач», выявивший существенные недостатки в документации и расхождения между заявленными требованиями и фактической реализацией.

Классифицированы выявленные дефекты по критичности и типам, что способствовало формированию навыков приоритизации ошибок и оценки общего качества продукта.

Сформулированы рекомендации по улучшению документации и устранению выявленных проблем, подчеркивающие важность детализации требований, визуализации интерфейса и описания обработки исключительных ситуаций.

Работа показала, что качественная документация и тщательное тестирование являются ключевыми факторами успешной разработки ПО. Выявленные в процессе тестирования критические и высокоприоритетные

дефекты в «Менеджере задач» свидетельствуют о необходимости серьезной доработки продукта перед его внедрением