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»

Выполнили студенты группы ИКБО-43-23 Хайров Эмиль, Чумаков Сергей, Рогачев Николай, Сергеев Андрей, Ерёмин Вадим.

Принял _____

Практическая работа выполнена «__»_____2025 г. *(подпись студента)*

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

Москва 2025

Содержание

Название команды «IBA CHETKO»

1 Разработка технического задания и программного продукта

2 Введение

Настоящее техническое задание (далее ТЗ) определяет цели, требования и условия для разработки настольного приложения «Менеджер задач». Программа предназначена для индивидуального использования или применения в малых рабочих группах с целью организации, планирования и отслеживания выполнения персональных или рабочих задач. Продукт позволяет пользователям эффективно управлять своим временем, устанавливать приоритеты и контролировать сроки.

Данное приложение представляет собой комплексное решение для индивидуального использования и малых рабочих групп, функционирующее в качестве централизованной системы учета задач различного уровня сложности и приоритета. В условиях растущих требований к эффективности временного менеджмента программа предлагает интуитивно понятный интерфейс для организации рабочих процессов, установки четких дедлайнов и визуального контроля за выполнением поставленных целей.

3 Основания для разработки

Разработка программного продукта инициирована для решения фундаментальной проблемы неэффективного управления временем и задачами, которая характерна для современных специалистов различных областей. Ручное управление задачами через бумажные носители или базовые текстовые редакторы демонстрирует крайне низкую эффективность: отсутствие систематизации, сложность поиска исторических данных, невозможность оперативного изменения приоритетов и высокий риск постановки конфликтующих дедлайнов.

Анализ рынка показывает растущий спрос на простые и функциональные решения для управления задачами, особенно в сегменте малого бизнеса и индивидуального использования. Существующие аналоги либо избыточно сложны, либо требуют постоянного подключения к интернету, либо не предоставляют необходимой гибкости в настройке параметров задач.

Основанием для выбора конкретной архитектуры решения послужила необходимость создания легковесного, но мощного инструмента, который мог бы работать в условиях ограниченного интернет-соединения и обеспечивать максимальную конфиденциальность данных пользователя.

4 Назначение разработки

Основная цель разработки - создание высокоэффективного инструмента управления задачами, который позволит радикально улучшить процессы планирования и исполнения работ как для индивидуальных пользователей, так и для малых рабочих групп.

Количественные цели:

- Сокращение времени на ежедневное планирование на 40-50% за счет интуитивного интерфейса и шаблонных решений
- Увеличение соблюдения установленных сроков на 25-35% благодаря системе визуальных напоминаний и приоритизации
- Снижение количества пропущенных задач на 60-70% через внедрение системы напоминаний и дублирования
- Уменьшение времени на поиск исторических данных и аналитику выполненных задач на 80%

Целевые показатели эффективности (KPI):

- Время добавления новой задачи: не более 15 секунд
- Время поиска задачи в архиве: не более 10 секунд
- Точность фильтрации и сортировки: 99.9%
- Надежность хранения данных: 99.95% uptime
- Совместимость с различными ОС: 100% функциональность

Программа должна стать универсальным инструментом, который интегрируется в ежедневную рутину пользователей и становится неотъемлемой частью их рабочего процесса, обеспечивая измеримое улучшение продуктивности.

Качественные цели:

- Повышение прозрачности рабочих процессов для руководителей малых отделов
- Улучшение личной эффективности и снижение стресса у пользователей
- Создание унифицированной системы постановки и контроля задач
- Формирование культуры дедлайнов и ответственного отношения к срокам
- Накопление исторических данных для последующего анализа

5 Требования к программе

5.1 Функциональные требования

5.1.1 Управление задачами

– Создание задач с обязательными полями: заголовок (до 255 символов), срок (формат YYYY-MM-DD), приоритет (Низкий, Средний, Высокий)

– Расширенное редактирование с поддержкой rich-text в поле описания

– Множественное удаление задач с подтверждением операции

– Изменение статуса выполнения (Выполнено/Невыполнено) с возможностью массового применения

– Дублирование задач с наследованием основных параметров

– Архивирование выполненных задач с возможностью восстановления

5.1.2 Организация и представление данных

– Табличное представление с колонками: Заголовок, Срок, Приоритет,

Статус, Дата создания

– Поддержка кастомной сортировки по всем колонкам таблицы

– Многоуровневая фильтрация по комбинации параметров

– Группировка задач по статусу, приоритету, датам

– Поиск с поддержкой полнотекстового индексирования по заголовку и описанию

– История изменений задач (лог модификаций)

5.1.3 Работа с данными

– Экспорт в JSON с поддержкой выбора диапазона данных

– Автоматическое резервное копирование с настраиваемой периодичностью

– Миграция данных между различными версиями программы

– Очистка старых данных с сохранением статистики

5.2 Условия эксплуатации

5.2.1 Программные среды

– Windows 10/11 (x64) с установленным .NET Framework 4.8 или выше

– Linux: Ubuntu 20.04 LTS+, Fedora 35+, CentOS 8+ с поддержкой GTK3

– macOS: версии 11.0 (Big Sur) и новее

5.2.2 Эксплуатационные характеристики

- Работа 24/7 без деградации производительности
- Поддержка многопользовательского режима (для будущих версий)
- Минимальное энергопотребление в фоновом режиме
- Совместимость с системами родительского контроля и корпоративными политиками

5.3 Требования к совместимости

5.3.1 Кроссплатформенность

- Единая кодовая база для всех поддерживаемых ОС
- Нативные интерфейсы для каждой платформы

5.3.2 Форматы данных

- JSON согласно RFC 8259 с UTF-8 кодировкой
- SQLite версии 3.35+ с обратной совместимостью

6 Требования к интерфейсу

6.1 Основные элементы управления

6.1.1 Панель управления задачами

- Поле "Заголовок": текстовое поле с автодополнением на основе предыдущих задач
- Поле "Срок": календарь с быстрым выбором дат + текстовый ввод с валидацией
- Выпадающий список "Приоритет": (высокий, средний, низкий)
- Поле "Описание": многострочное текстовое поле с поддержкой базового форматирования
- Кнопка "Добавить задачу": всегда видимая, с иконкой "+"
- Кнопка "Редактировать": активируется только при выборе задачи
- Кнопка "Удалить": с подтверждением и отменой операции
- Кнопка "Отметить/Снять выполнение"

6.1.2 Панель управления данными

- Кнопка "Экспорт в JSON": с выбором файла и опциями экспорта
- Кнопка "Импорт из JSON"
- Выпадающий список "Фильтр": Все/Выполненные/Невыполненные
- Поле "Поиск": с инкрементальным поиском и подсказками
- Кнопка "Сброс фильтра": мгновенный сброс всех фильтров

– Кнопка "Сортировать по сроку": с индикацией направления сортировки

7 Критерии приемки

7.1 Функциональное тестирование

- Выполнение 95% всех разработанных тест-кейсов
- Отсутствие блокирующих и критических дефектов
- Соответствие всем заявленным функциональным требованиям

7.2 Производительность

- Время запуска приложения: не более 3 секунд
- Время отклика на пользовательские действия: не более 200 мс
- Потребление памяти: не более 500 MB при 10 000 задач

7.3 Требования к документации

Состав программной документации должен включать в себя:

- Техническая документация
- Пользовательская документация

7.4 Порядок контроля и приемки

Тестирование методом «черного ящика»: подбор входных данных (разные варианты ответов), анализ правильности реакций программы.

Проверка всех функциональных сценариев: прохождение викторины до конца, выбор неверных и верных ответов, переходы между вопросами.

7.5 Этапы и сроки разработки

Таблица 1. Этапы и сроки разработки

№	Наименование этапа	Срок исполнения	Исполнитель
1	Анализ требований и проектирование архитектуры приложения	2 рабочих дня	Разработчик
2	Разработка системы и модуля работы с базой данных	3 рабочих дня	Разработчик
3	Создание пользовательского интерфейса и визуальных компонентов	3 рабочих дня	Разработчик
4	Тестирование, откладка и подготовка финальной версии	5 рабочих дней	Разработчик
5	Сдача проекта и ввод в эксплуатацию	2 рабочих дня	Разработчик

8 Документация

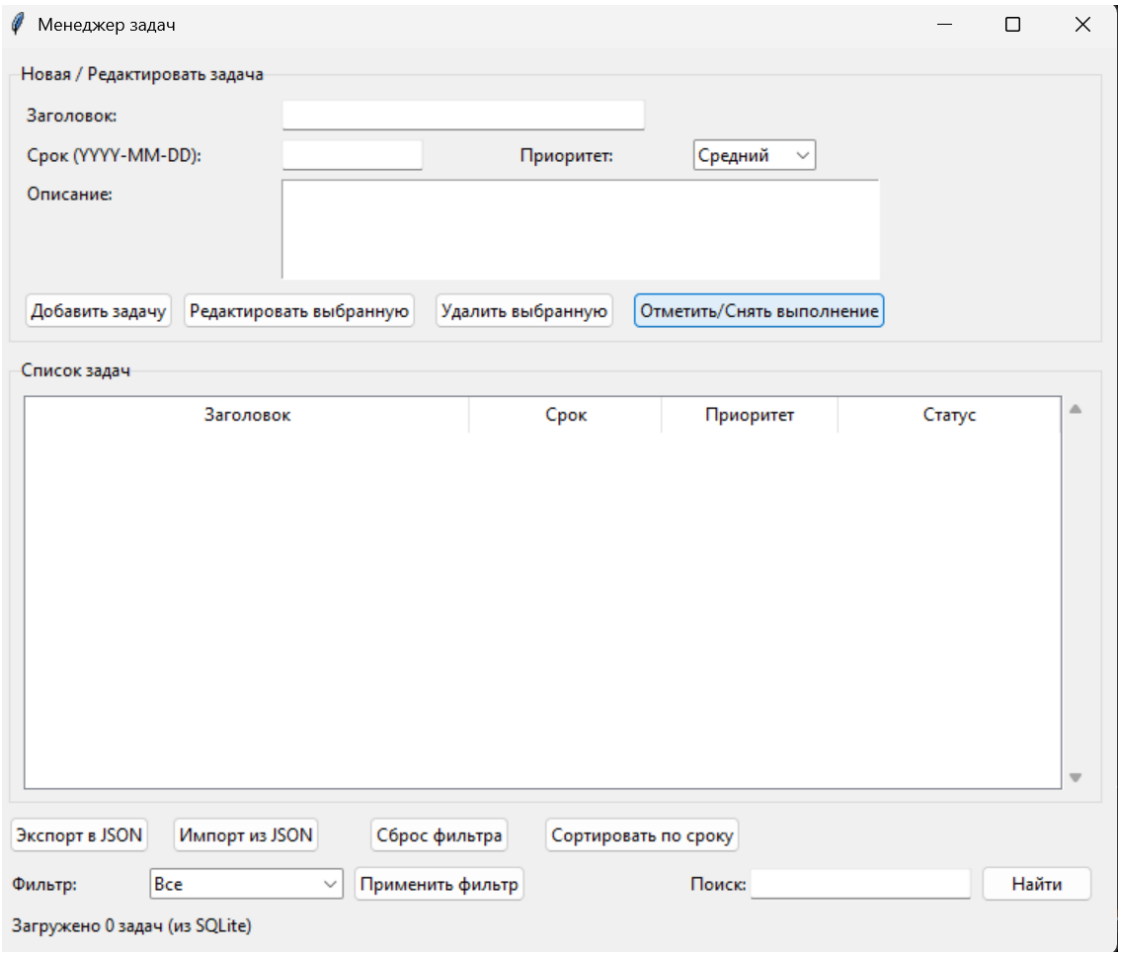


Рисунок 1 – интерфейс программы

Руководство пользователя

Запуск: открыть файл main.exe

При старте отображается интерфейс с различными функциями

9 Описание внесённых ошибок в собственное ПО

- Неправильный формат даты (DATE_FORMAT) не соответствует подписи поля.
- Кнопки Добавить и Редактировать перетрутся (нажатие вызывает другую функцию).
- Удаление сделано через id, но неверно обрабатывается (при удалении записи по id — иногда удаляется не та запись).
- Переключение статуса (выполнено) меняет только память, но не сохраняется сразу в БД (потеряется при перезапуске).
- Нужно реализовать через startswith и чувствительный к регистру — частичные/регулярные выражения пропускаются.
- Кнопка Сохранить изменения не убирается после сохранения (накладывается при повторном редактировании).

Тестирование программного обеспечения охватывает множество аспектов, включая функциональное, нагрузочное и регрессионное тестирование. Каждый из этих типов тестирования имеет свои цели и методы, что позволяет выявить различные виды дефектов на разных этапах разработки. Например, функциональное тестирование направлено на проверку корректности работы всех функций программы и её соответствие заданным требованиям, в то время как нагрузочное тестирование помогает оценить

производительность системы при высоких нагрузках. Современные инструменты автоматизации тестирования значительно повышают эффективность процесса проверки. Использование автоматизированных тестов позволяет не только ускорить выявление ошибок, но и снизить вероятность человеческого фактора, который может привести к пропускам критически важных дефектов. Однако, несмотря на все преимущества

автоматизации, ручное тестирование по-прежнему остается важной частью QA-процесса, особенно в случаях, когда необходима интуитивная оценка пользовательского опыта.

10 Техническое задание (ТЗ) и документацию программного продукта другой команды

10.1 Введение

GUI UNIX OS — это учебный эмулятор командной строки UNIX-подобной операционной системы.
Программа позволяет работать с виртуальной файловой системой (в формате .zip) и исполнять базовые команды, имитируя поведение shell-сессии.

10.2 Основания для разработки

Проект создан в рамках практической работы по дисциплине *Тестирование программного обеспечения методом «черного ящика»*.
Основная цель — изучение принципов тестирования ПО, создание документации, а также развитие навыков поиска и документирования ошибок.

10.3 Назначение разработки

- Программа предназначена для:
- обучения работе с базовыми командами UNIX;
- демонстрации принципов эмуляции виртуальной файловой системы;
- практики тестирования методом «черного ящика».

10.4 Требования к программе

10.4.1 Функциональные требования

- Приём виртуальной ФС в виде zip-архива;
- Запуск стартового скрипта (.sh);
- Поддержка базовых команд (ls, cd, mv, wc, help, exit, clear).

10.4.2 Надёжность

- Программа должна корректно завершать работу при команде exit.
- При некорректных параметрах запуска выводить сообщение об ошибке.

10.4.3 Условия эксплуатации

- ОС: Windows, Linux, MacOS.
- Python 3.8+ с установленной библиотекой pygame.

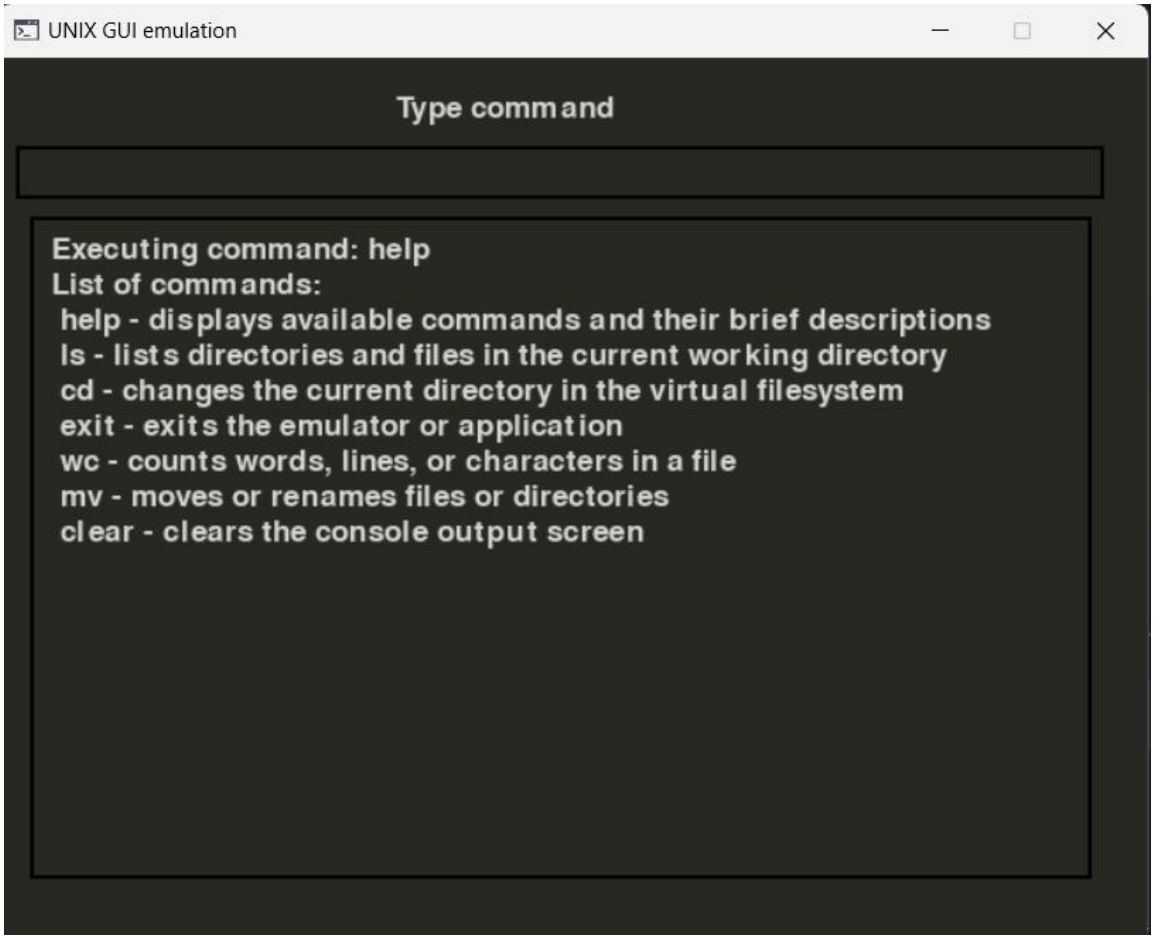
10.4.4 Совместимость

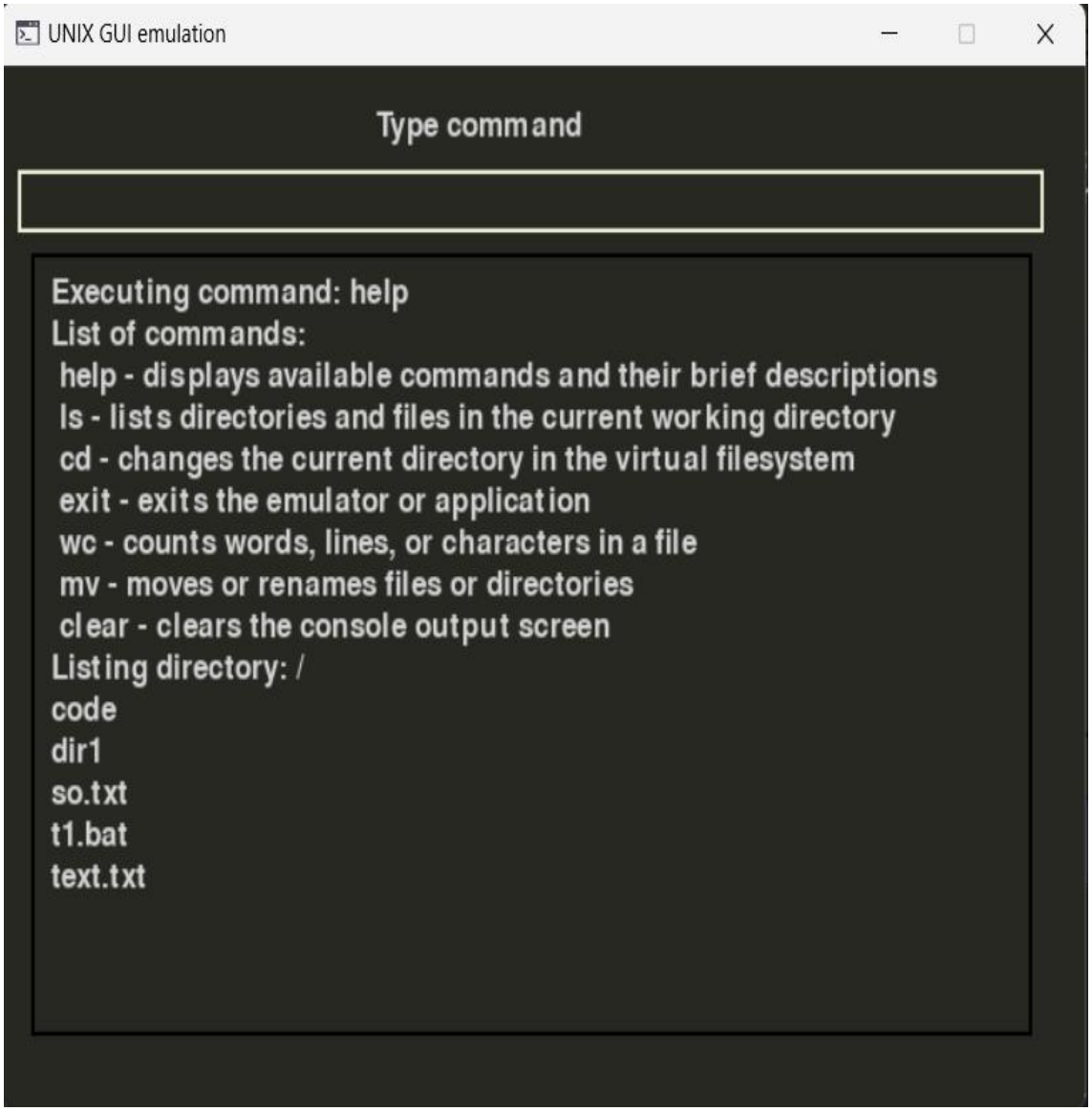
- Работает в терминале с поддержкой UTF-8.
- Тестировалось на Windows 10 и Ubuntu 22

11 Требования к интерфейсу

Программа запускается в консоли.

Пример интерфейса:




UNIX GUI emulation
Type command
Executing command: help
List of commands:
help - displays available commands and their brief descriptions
ls - lists directories and files in the current working directory
cd - changes the current directory in the virtual filesystem
exit - exits the emulator or application
wc - counts words, lines, or characters in a file
mv - moves or renames files or directories
clear - clears the console output screen
Listing directory: /
code
dir1
so.txt
t1.bat
text.txt

12 Критерии приёмки

Успешное выполнение не менее 95% тест-кейсов;

Корректная работа всех команд;

Адекватная реакция на некорректный ввод.

13 Установка и запуск

Установка

Листинг 1

```
git clone https://github.com/Nikindrik/GUI_UNIX_OS
python -m venv venv
Windows:
.\venv\Scripts\activate
pip3 install pygame
For linux/UNIX/MACLinux/UNIX/Mac:
source venv/bin/activate
pip3 install pygame
Run
Запуск
```

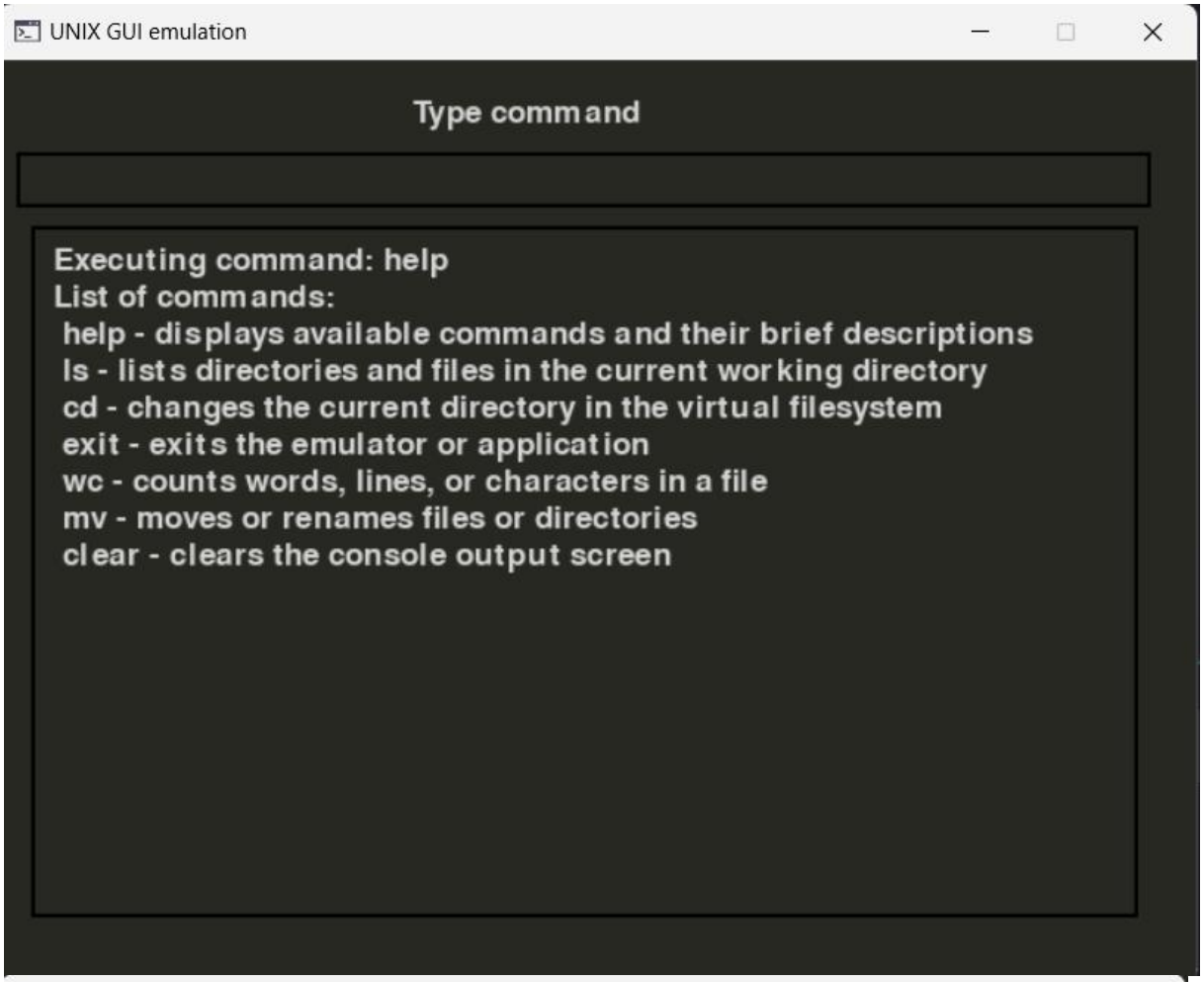
Листинг 2

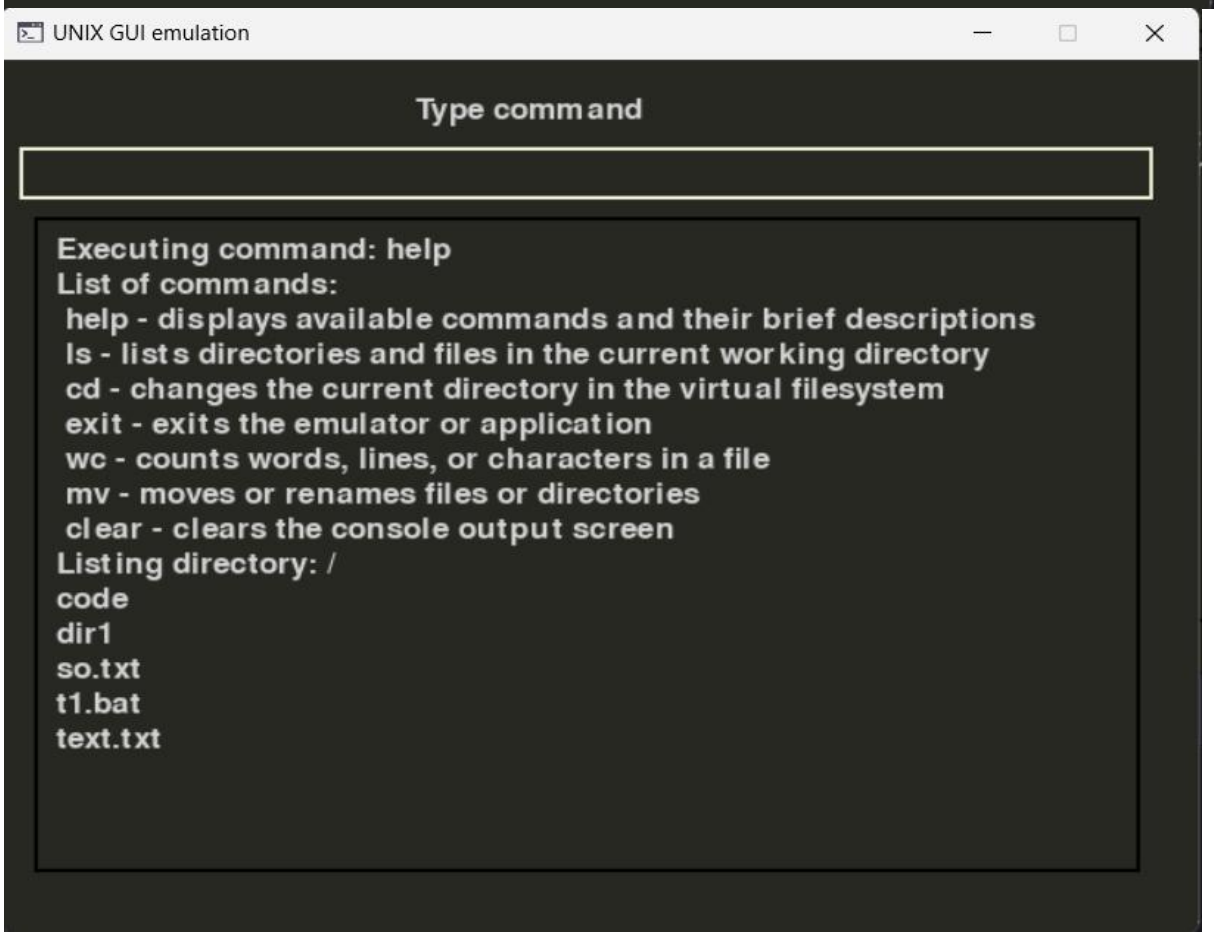
```
python main.py --user <имя_пользователя> --archive <архив.zip> --script <стартовый_скрипт.sh>
Пример:
python main.py --user nick --archive systeam.zip --script start.sh
python main.py --user <user_name> --archive <archive_name.zip> --script <start_script_name.sh>
Example
python main.py --user nick --archive systeam.zip --script start.sh
```

Команды эмулятора

Displays available commands and their brief descriptions - help

- Lists directories and files in the current working directory - ls
- Changes the current directory in the virtual filesystem - cd
- Exits the emulator or application - exit
- Counts words, lines, or characters in a file - wc
- Moves or renames files or directories - mv
- Clears the console output screen – clear


UNIX GUI emulation
Type command
Executing command: help
List of commands:
help - displays available commands and their brief descriptions
ls - lists directories and files in the current working directory
cd - changes the current directory in the virtual filesystem
exit - exits the emulator or application
wc - counts words, lines, or characters in a file
mv - moves or renames files or directories
clear - clears the console output screen


UNIX GUI emulation
Type command
Executing command: help
List of commands:
help - displays available commands and their brief descriptions
ls - lists directories and files in the current working directory
cd - changes the current directory in the virtual filesystem
exit - exits the emulator or application
wc - counts words, lines, or characters in a file
mv - moves or renames files or directories
clear - clears the console output screen
Listing directory: /
code
dir1
so.txt
t1.bat
text.txt

ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы был проведен всесторонний анализ процесса тестирования программного обеспечения на примере учебного проекта – эмулятора UNIX-подобной операционной системы. Работа позволила закрепить теоретические знания и приобрести практические навыки в области тестирования по методу «чёрного ящика», выявления дефектов различного типа, их документирования и оценки влияния на качество продукта.

Основные результаты работы включают:

Выявление и классификация дефектов. В процессе тестирования эмулятора был обнаружен и систематизирован ряд ошибок, которые были разделены на три ключевые категории: интерфейсные, логические и синтаксические.

Анализ интерфейсных дефектов. Были зафиксированы критические недостатки пользовательского интерфейса, такие как смещение основного текста вправо и некорректное позиционирование текста в поле ввода. Эти ошибки напрямую ухудшают пользовательский опыт и свидетельствуют о проблемах в верстке или логике отрисовки элементов.

Исследование логических ошибок. Наиболее серьезные проблемы были выявлены в бизнес-логике приложения. К ним относятся: некорректная обработка исключений при работе с zip-архивами, ошибочная разбивка введенной команды на составные части, неверная проверка типов в условных конструкциях (if-else), а также сбои в логике выбора функции для команд exit и mv. Данные дефекты приводят к нестабильной работе эмулятора, непредсказуемому поведению и выполнению команд не в соответствии с техническим заданием.

Обнаружение синтаксических ошибок. В файлах console.py и emulator.py были выявлены синтаксические недочеты, которые, хотя и не всегда критичны

сами по себе, указывают на недостаточный уровень внимания к качеству кода на этапе разработки.

Формирование рекомендаций. На основе проведенного анализа сформулированы конкретные рекомендации по устранению выявленных недостатков, включающие исправление алгоритмической логики, рефакторинг кода и доработку пользовательского интерфейса.

Проведенная работа наглядно продемонстрировала, что даже в учебном проекте совокупность незначительных на первый взгляд ошибок (интерфейсных, синтаксических) и серьезных логических дефектов может существенно снизить функциональность и надежность программного продукта. Выявленные проблемы подчеркивают важность тщательного модульного и интеграционного тестирования на всех этапах разработки, а также необходимость строгого следования принципам написания чистого и поддерживаемого кода.