



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

*Ермоленко В.М., Кавказский И.К.,
Дементиевский Д.В., Тарасов А.М.*

Проверил:

Ильичев Г.П.

2025 г.

Состав команды:

Ермоленко В.М.

Кавказский И.К.

Дементиевский Д.В.

Тарасов А.М.

Часть 1. Разработка технического задания и программного продукта

1. Введение

Программа «Текстовый чат» (условное обозначение: ТЧ-01) представляет собой клиент-серверное приложение, предназначенное для организации быстрого и безопасного обмена текстовыми сообщениями в реальном времени.

Областью применения программы является предоставление универсальной платформы для свободного общения, кооперации и взаимодействия пользователей по интересам через публичные сети (Интернет).

2. Основания для разработки

2.1. Причины разработки программы:

Разработка программы инициирована по следующим причинам:

2.1.1. Образовательная цель: Освоение на практике современных технологий и методов разработки кроссплатформенных клиент-серверных приложений, включая сетевое программирование, работу с базами данных и создание пользовательских интерфейсов.

2.1.2. Простота и минимализм: Существует потребность в простом и легком мессенджере без рекламы, алгоритмических лент, историй и избыточных функций, который бы фокусировался исключительно на текстовом общении.

2.1.3. Контроль над данными: Позволяет организовать частное и безопасное общение для небольшой группы людей с полным контролем над сервером и данными, в отличие от публичных крупных платформ.

2.1.4. Технический эксперимент: Создание прототипа для отработки архитектурных решений, которые могут быть использованы в более крупных проектах в будущем.

2.2. Нормативные и исходные документы:

-Техническое задание (настоящий документ).

3. Назначение разработки

Разработка программы «Текстовый чат» (условное обозначение: «ТЧ-01») предназначена для решения следующих основных задач:

3.1. Организация коммуникационного пространства:

- Создание универсальной платформы для текстового общения пользователей в реальном времени

3.2. Повышение эффективности взаимодействия:

- Сокращение времени на установление контакта между пользователями со схожими интересами
- Упрощение процесса координации совместной деятельности в группах

3.3. Обеспечение доступности и удобства:

- Предоставление кроссплатформенного решения с возможностью использования на различных устройствах
- Обеспечение интуитивно понятного интерфейса, не требующего специального обучения

3.4. Гарантия безопасности общения:

- Обеспечение конфиденциальности личной переписки пользователей

Цель разработки: Создание современной и удобной платформы для текстовой коммуникации, позволяющей пользователям эффективно организовывать общение по интересам и совместную деятельность через интернет.

4. Требование к программе

4.1. Что должна уметь программа (функции):

- Регистрация и вход пользователей по логину и паролю
- Отправка и получение текстовых сообщений
- Просмотр списка активных пользователей (команда /online)
- Хранение истории чата и вывод последних 10 сообщений при входе пользователя

4.2. Надежность:

- Программа не должна зависать при обычном использовании
- Сообщения не должны теряться при отправке
- При перезагрузке программы история чатов не должна теряться

4.3. Где будет работать:

- **Компьютеры:** Windows 10, 11; современные версии macOS; Linux

4.4. Совместимость:

- Работа в локальной сети и через интернет

- Возможность работы на разных устройствах одновременно

4.5. Безопасность:

- Пароли пользователей должны надежно храниться
- Защита от спама и флуда

5. Требования к интерфейсу

Интерфейс программы является текстовым (консольным) и должен быть организован просто и интуитивно. Все элементы управления реализуются с помощью текста, цветового выделения и управления с клавиатуры.

Текстовое описание экранов и элементов управления

5.1 Экран аутентификации (логин)

- Экран должен содержать область для ввода данных.
- Поля "Логин:" и "Пароль:" должны быть четко обозначены.
- Вход должен происходить по нажатию клавиши Enter после ввода пароля.

5.2 Основной экран чата

После успешного входа пользователь сразу попадает в общий чат.

- **История сообщений:** В основной области экрана построчно выводятся сообщения всех пользователей. Каждая строка должна иметь четкий формат: «Имя пользователя: сообщение»
- **Новые сообщения:** Новые входящие сообщения должны немедленно появляться в истории сообщений.
- **Поле ввода:** Приглашение для ввода должно быть всегда видимым. Ввод сообщения осуществляется в строке.
- **Отправка сообщения:** Сообщение отправляется исключительно по нажатию клавиши Enter. После отправки строка ввода должна немедленно очищаться, а отправленное сообщение — добавляться в историю.
- **Системные уведомления:** Сообщения о входе ({nickname} присоединился к чату!) и выходе ({nickname} покинул чат!) пользователей должны выводиться в основную область и быть выделены отдельным цветом.

5.3 Выход из программы

- Ввод команды /quit должен завершать работу программы.
- Перед завершением работы выводится системное сообщение.

5.4 Элементы взаимодействия.

- **Простота:** Интерфейс должен быть минималистичным, без излишних элементов, отвлекающих от процесса общения.
- **Дополнительные команды:** /online - просмотр списка пользователей, которые сейчас онлайн; /? - справка по командам

6. Критерии приемки

Программа считается принятой и соответствующей настоящему Техническому заданию, если **Заказчик (или уполномоченное лицо) лично подтверждает**, что выполнены все следующие условия:

6.1. Критерии функциональной готовности:

6.1.1 Рабочий цикл "вход-сообщение-выход": Пользователь может успешно запустить программу, ввести свой логин и пароль, попасть в общий чат, отправить текстовое сообщение, увидеть свое сообщение в истории чата, завершить работу программы с помощью команды /quit без аварийного завершения (краша).

6.1.2 Доставка сообщений в реальном времени: Сообщение, отправленное одним пользователем, немедленно (с задержкой, не заметной пользователю) появляется в консоли любого другого пользователя, уже находящегося в чате.

6.1.3 Корректное отображение информации: Все сообщения в истории чата отображаются в формате: Имя отправителя: текст. Системные сообщения имеют цветное выделение.

6.1.4 Системные события: При подключении нового пользователя или его отключении в общем чате появляется соответствующее текстовое уведомление с цветовым выделением.

6.2. Критерии надежности и завершенности:

6.2.1 Сохраняемость истории: История сообщений в чате сохраняется между перезапусками клиентской части программы. При повторном входе пользователи видят историю последних сообщений.

6.2.2 Отсутствие критических ошибок: В течение 30 минут непрерывного использования программы (подключение 2-3 пользователей, обмен сообщениями) не возникает ситуаций, приводящих к аварийному завершению работы клиентской или серверной части.

6.3. Критерии соответствия требованиям:

6.3.1 Соответствие интерфейса: Пользовательский интерфейс соответствует текстовому описанию, приведенному в разделе 5 настоящего ТЗ, включая цветное выделение различных типов сообщений.

6.3.2 Кроссплатформенность: Клиентская часть программы успешно запускается и выполняет свой основной функционал (п. 6.1) на двух различных операционных системах (например, Windows и Linux).

Приемка осуществляется путем демонстрации выполнения каждого пункта критериев Заказчику.

7. Требования к документации

В состав программной документации должны входить следующие документы, оформленные в виде текстовых файлов в формате Markdown (.md) или PDF:

7.1. Руководство пользователя (User Manual):

Документ должен содержать исчерпывающее описание процедур работы с программой для конечного пользователя.

Содержание:

- Назначение программы.
- Требования к системе для запуска (минимальные версии ОС, необходимое ПО).
- Инструкция по установке и запуску (например, команды для терминала).
- Пошаговое описание работы: как запустить, войти, отправить сообщение, выйти.
- Описание формата отображения сообщений и системных уведомлений.
- Список доступных команд (например, /quit).
- Описание возможных проблем и способы их решения (например, "ошибка подключения").

7.2. Краткое техническое описание (Technical Overview):

Документ предназначен для разработчика или администратора, разворачивающего систему.

Содержание:

- Описание архитектуры системы (клиент-серверная модель).
- Протокол обмена данными между клиентом и сервером (например, "текстовые команды по TCP").
- Форматы хранения данных (описание структуры файла с историей сообщений).
- Инструкция по развертыванию серверной части (порты, конфигурация, команда для запуска).
- Описание клиентских аргументов командной строки (если есть).

8. Порядок контроля и приемки

8.1 Методы тестирования:

8.1.1 Модульное тестирование с покрытием 100 % логики функций формирования заказа.

8.1.2 Интеграционное тестирование API: 50 + эндпоинтов GET/POST по спецификации.

8.1.3 Системное и приемочное тестирование по чек-листам и пользовательским сценариям.

8.2 Фиксация и устранение дефектов:

8.2.1 Все баги регистрируются в системе Jira с меткой «Приемка».

8.2.2 Критические ошибки (блокирующие расчёт или сохранение отчёта) устраняются в течение 1 рабочего дня.

8.2.3 Неблокирующие — до конца этапа 11.5.

8.3 Приемочный акт:

8.3.1 Составляется после успешного прохождения всех пунктов 6.1–6.4.8

8.3.2 Включает перечень выполненных проверок, подписи сторон, дату приемки.

9. Этапы и сроки разработки

Этап	Длительность, Дни	Выходные артефакты
9.1 Сбор и анализ требований	3	Спецификация исходных данных, согласованный ТЗ
9.2 Проектирование архитектуры	4	Диаграммы компонентов и потоков данных
9.3 Разработка прототипа интерфейса	5	Прототипы экранов (PDF/PNG), описание UI-стиля
9.4 Модульное тестирование	4	Отчет по unit-тестам, исправленные баги
9.5 Интеграция и системное тестирование	5	Сценарии интеграции, отчёт по нагрузочному тестированию
9.6 Приемочные испытания и документация	3	Полный пакет документации, акт приемки

Руководство пользователя “Текстовый чат”

Руководство пользователя Текстовый Чат (ТЧ-01)

Назначение программы

Текстовый Чат (ТЧ-01) - это простое клиент-серверное приложение для обмена текстовыми сообщениями в реальном времени через сеть.

Требования к системе

- Операционные системы: Windows 10/11, macOS, Linux
- Python: версия 3.6 или выше
- Дополнительные библиотеки: termcolor (`pip install termcolor`)

Установка и запуск

Запуск сервера

- Сохраните файлы `server.py` и `utils.py` в одну папку
- Откройте терминал в этой папке
- Запустите сервер:

```
python server.py
```

Запуск клиента

- Сохраните файлы `client.py` и `utils.py` в одну папку
- Откройте терминал в этой папке
- Запустите клиент:

```
python client.py
```

Пошаговое описание работы

1. Запуск программы

После запуска клиента откроется интерфейс аутентификации.

2. Аутентификация

```
==== ВХОД ====
Регистрация (1) или Вход (2):
Логин:
Пароль:
```

- Выберите "1" для регистрации нового аккаунта
- Выберите "2" для входа в существующий аккаунт
- Введите логин и пароль

3. Работа в чате

После успешной авторизации вы попадете в общий чат:

```
~[WELCOME TO EASY_CHAT]~
Добро пожаловать в чат, username! Сейчас онлайн: 1
(Нажмите ?? для подсказки)
username:
```

4. Отправка сообщений

- Вводите текст сообщения и нажимайте Enter для отправки
- Ваши сообщения отображаются в формате: `username: текст сообщения`

5. Команды чата

- `/quit` - выход из программы
- `/online` - показать список онлайн-пользователей
- `/?` - показать справку по командам

6. Выход из программы

Введите команду `/quit` для завершения работы.

Формат отображения сообщений

- Обычные сообщения: `username: текст сообщения`
- Системные уведомления: Цветное оформление
- Приветственные сообщения: Желтый и зеленый цвет
- История чата: Серый цвет (последние 10 сообщений)

Цветовая схема:

- [YLW]: - желтый (приветствия)
- [GRN]: - зеленый (системные сообщения)
- [GRY]: - серый (история чата)
- [RED]: - красный (ошибки)
- [WHT]: - белый (обычный текст)

Возможные проблемы и решения

Ошибка подключения

Симптомы: "Ошибка подключения: [Errno 111] Connection refused" Решение: Убедитесь, что сервер запущен на правильном IP и порту

Пользователь не найден

Симптомы: "Ошибка авторизации" после ввода логина/пароля Решение: Проверьте правильность ввода или зарегистрируйте новый аккаунт

Неправильный пароль

Симптомы: "Ошибка авторизации" Решение: Проверьте правильность ввода пароля

Разрыв соединения

Симптомы: Программа внезапно закрывается Решение: Перезапустите клиент и проверьте соединение с сервером

Краткое техническое описание (Technical Overview):

Техническое описание Текстовый Чат (ТЧ-01)

Архитектура системы

Система использует клиент-серверную архитектуру:

- Сервер: Центральный узел, обрабатывающий все соединения и сообщения
- Клиенты: Подключаются к серверу для отправки и получения сообщений

Протокол обмена данными

Связь осуществляется по TCP с использованием текстовых команд:

Команды аутентификации

- REGISTER:nickname:password - регистрация нового пользователя
- LOGIN:nickname:password - вход существующего пользователя

Ответы сервера

- REGISTER_SUCCESS - успешная регистрация
- LOGIN_SUCCESS - успешный вход
- USER_EXISTS - пользователь уже существует
- USER_NOT_FOUND - пользователь не найден
- WRONG_PASSWORD - неверный пароль
- INVALID_COMMAND - неизвестная команда

Команды чата

- /quit - отключение от сервера
- /online - запрос списка онлайн-пользователей
- /? (на клиенте) - вывод списка команд
- Любой другой текст - отправка сообщения в чат

Формат сообщений

- Обычные сообщения: username: text
- Системные сообщения: [COLOR]:text
- Цветовые коды: YLW, GRN, RED, WHT

Форматы хранения данных

Хранение пользователей

Пользователи хранятся в памяти сервера в словаре:

```
user_database = {
    'nickname1': 'password1',
    'nickname2': 'password2'
}
```

Хранение истории сообщений

История хранится в памяти сервера как список строк:

```
history = [
    'user1: message1',
    'user2: message2',
    ...
]
```

Онлайн-пользователи

Список активных пользователей хранится в памяти:

```
online_users = ['user1', 'user2']
```

Развертывание серверной части

Конфигурация сервера

По умолчанию сервер использует:

- Хост: localhost
- Порт: 12345
- Максимальное кол-во подключений: 5

Изменение настроек

Для изменения настроек отредактируйте конструктор класса ChatServer:

```
server = ChatServer(host='сам_хост', port=сам_порт)
```

Запуск сервера

```
python server.py
```

Сервер запускается в фоновом режиме и обрабатывает подключения в отдельных потоках.

Клиентские настройки

Подключение к серверу

По умолчанию клиент подключается к localhost:12345

Изменение настроек подключения

Отредактируйте конструктор класса SimpleChatClient:

```
client = SimpleChatClient(host='адрес_сервера', port=порт)
```

Аргументы командной строки

В текущей версии аргументы командной строки не поддерживаются. Настройки изменяются путем редактирования кода.

Особенности реализации

Многопоточность

- Сервер использует отдельный поток для каждого клиента
- Клиент использует отдельный поток для прослушивания сообщений от сервера

Обработка ошибок

- Автоматическое удаление отключившихся клиентов
- Защита от обрывов соединения
- Валидация входящих команд

Цветовое оформление

Используется библиотека `termcolor` для цветового выделения различных типов сообщений

Безопасность

- Пароли хранятся в открытом виде в памяти сервера
- Шифрование передаваемых данных не реализовано
- Рекомендуется использовать в доверенных сетях

Описание намеренно внесенных ошибок:

1. Идентификатор

ТС-001

2. Название.

Проверка выхода из онлайн.

3. Описание

Убедиться, что сервер корректно отображает активных пользователей

4. Предварительные условия:

Сервер запущен, пользователь некорректно завершил работу клиента.

5. Шаги выполнения:

1. Зайти в клиент
2. Некорректно завершить работу

6. Ожидаемый результат.

Сервер отобразит, что пользователь оффлайн

7. Фактический результат.

Сервер считает, что пользователь активен

```
=== ВХОД ===
Регистрация (1) или Вход (2): 1
Логин: prokuror
Пароль: 123
Успешная авторизация!
prokuror:
~[WELCOME TO EASY_CHAT]~

Добро пожаловать в чат, prokuror! Сейчас онлайн: 2
(Напишите /? для помощи)

# Последние сообщения #
GeorgeDroyd: hi guys!
prokuror: /online
Онлайн (2): GeorgeDroyd, prokuror
prokuror: █
```

8. Статус.

Failed

1. Идентификатор

ТС-002

2. Название.

Проверка на возможность регистрации пользователей с одинаковыми никнеймами.

3. Описание

Убедиться, что сервер корректно работает при регистрации пользователей с одинаковыми никнеймами

4. Предварительные условия:

Сервер запущен, пользователи зарегистрировали два одинаковых никнейма.

5. Шаги выполнения:

1. Зайти в клиент
2. Зарегистрировать два одинаковых никнейма

6. Ожидаемый результат.

Сервер выдаст ошибку "USER_EXISTS"

7. Фактический результат.

Сервер позволяет зарегистрировать два одинаковых никнейма

```

=== ВХОД ===
Регистрация (1) или Вход (2): 1
Логин: litvin
Пароль: 123
Успешная авторизация!
litvin:
~[WELCOME TO EASY_CHAT]~

Добро пожаловать в чат, litvin! Сейчас онлайн: 2
(Напишите /? для помощи)

# Последние сообщения #
GeorgeDroyd: hi guys!
prokuror: /exit
prokuror: exit
litvin: /quit
Выход...
Отключено
PS C:\Users\Barabumkin\Downloads\cli_msgr> ^C
PS C:\Users\Barabumkin\Downloads\cli_msgr>
PS C:\Users\Barabumkin\Downloads\cli_msgr> c::; c
python.debugpy-2025.10.0-win32-x64\bundled\libs\d
=== ВХОД ===
Регистрация (1) или Вход (2): 1
Логин: litvin
Пароль: 123
Успешная авторизация!
litvin:
~[WELCOME TO EASY_CHAT]~

Добро пожаловать в чат, litvin! Сейчас онлайн: 2
(Напишите /? для помощи)

```

8. Статус.

Failed

1. Идентификатор

ТС-003

2. Название.

Отображение надписи «выход»

3. Описание

Убедиться, что надпись «выход» отображается корректно при выходе пользователя из чата

4. Предварительные условия:

Сервер запущен, пользователь вышел из онлайн

5. Шаги выполнения:

1. Зайти в клиент
2. Выйти из клиента

6. Ожидаемый результат.

Корректное отображение надпись «выход»

7. Фактический результат.

Надпись «Выход» выводится как ошибка, красным цветом

```
Регистрация (1) или Вход (2): 1
Логин: litvin
Пароль: 123
Успешная авторизация!
litvin:
~[WELCOME TO EASY_CHAT]~

Добро пожаловать в чат, litvin! Сейчас онлайн: 2
(Напишите /? для помощи)

# Последние сообщения #
GeorgeDroyd: hi guys!
prokuror: /exit
prokuror: exit
litvin: /quit
Выход...
```

8. Статус.

Failed

1. Идентификатор

ТС-004

2. Название.

Отображение надписи «Пользователь отключился»

3. Описание

Убедиться, что надпись «Пользователь отключился» отображается при выходе пользователя из чата

4. Предварительные условия:

Сервер запущен, пользователь вышел из онлайн

5. Шаги выполнения:

1. Зайти в клиент
2. Выйти из клиента

6. Ожидаемый результат.

Корректное отображение надпись «Пользователь отключился»

7. Фактический результат.

Надпись не отображается

```
Tamaev2:
~[WELCOME TO EASY_CHAT]~

Добро пожаловать в чат, Татаев2! Сейчас онлайн: 4
(Напишите /? для помощи)

# Последние сообщения #
GeorgeDroyd: hi guys!
prokuror: /exit
prokuror: exit
Tamaev2: /quit
Выход...
Отключено
```

8. Статус.

Failed

1. Идентификатор

ТС-005

2. Название.

Корректность сохранения пароля

3. Описание

Убедиться, что пароль чувствителен к регистру

4. Предварительные условия:

Сервер запущен, пользователь зарегистрировался в системе

5. Шаги выполнения:

1. Зайти в клиент
2. Зарегистрироваться, используя пароль с различным регистром
3. Выйти из клиента
4. Войти, используя пароль в одном регистре

6. Ожидаемый результат.

Ошибка «неправильный пароль»

7. Фактический результат.

Программа игнорирует регистр

```
=== ВХОД ===
Регистрация (1) или Вход (2): 1
Логин: Татаев2
Пароль: Abc
Успешная авторизация!
Татаев2:
~[WELCOME TO EASY_CHAT]~

Добро пожаловать в чат, Татаев2! Сейчас онлайн: 4
(Напишите /? для помощи)

# Последние сообщения #
GeorgeDroyd: hi guys!
prokuror: /exit
prokuror: exit
Татаев2: /quit
Выход...
Отключено
PS C:\Users\Barabumkin\Downloads\cli_msgn> ^C
PS C:\Users\Barabumkin\Downloads\cli_msgn>
PS C:\Users\Barabumkin\Downloads\cli_msgn> c::; cd
s\Barabumkin\.vscode\extensions\ms-python.debugpy-
=== ВХОД ===
Регистрация (1) или Вход (2): 2
Логин: Татаев2
Пароль: abc
Успешная авторизация!
Татаев2:
~[WELCOME TO EASY_CHAT]~
```

8. Статус.

Failed

Часть 2. Тестирование ПП

ТЗ другой группы:

Состав команды “Экспедиция”: Селянин Н.М; Чихачев И.А.

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.

Ошибки, найденные в ПП другой группы:

1. Идентификатор

ТС-001

2. Название

Проверка валидации длины имени при добавлении участника

3. Описание

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

4. Предварительные условия

Приложение запущено, база данных пуста

5. Шаги выполнения

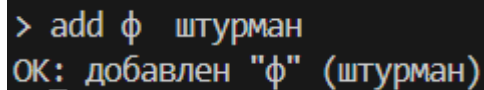
а) Ввести команду: add ф штурман (имя из 1 символа)

б) Нажать Enter

6. Ожидаемый результат

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

7. Фактический результат



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

Система принимает короткое имя: "OK: добавлен 'ф' (штурман)"

8. Статус

Failed

1. Идентификатор

ТС-002

2. Название

Проверка регистронезависимости при проверке дубликатов имен

3. Описание

Убедиться, что система корректно обрабатывает имена в разных регистрах как дубликаты

4. Предварительные условия

В базе есть участник с именем "Анна"

5. Шаги выполнения

а) Ввести команду: add "АННА грузчик"

б) Нажать Enter

6. Ожидаемый результат

Система возвращает ошибку: "такое имя уже существует"

7. Фактический результат


```
> add Анна грузчик
OK: добавлен "Анна" (грузчик)
> add АННА грузчик
OK: добавлен "АННА" (грузчик)
> list
1) ян – штурман
2) ф – штурман
3) джордждройд – грузчик
4) вадим – штурман
5) Анна – грузчик
6) АННА – грузчик
```

Система принимает имя как уникальное: "OK: добавлен 'АННА' (грузчик)"

8. Статус

Failed

1. Идентификатор

ТС-003

2. Название

Проверка сортировки списка участников по возрастанию

3. Описание

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

4. Предварительные условия

В базе есть участники

5. Шаги выполнения

а) Ввести команду: list

б) Нажать Enter

6. Ожидаемый результат

Список отображается в порядке по алфавиту

7. Фактический результат

```
> list
1) ян – штурман
2) ф – штурман
3) джордждройд – грузчик
4) вадим – штурман
5) Анна – грузчик
6) АННА – грузчик
```

Список отображается в обратном порядке

8. Статус

Failed

1. Идентификатор

ТС-004

2. Название

Проверка регистронезависимости фильтрации по роли

3. Описание

Убедиться, что фильтр по роли работает независимо от регистра ввода

4. Предварительные условия

В базе есть участник с ролью "водитель"

5. Шаги выполнения

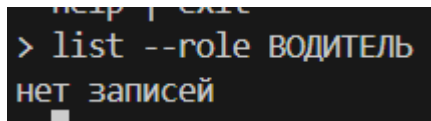
а) Ввести команду: `list --role ВОДИТЕЛЬ`

б) Нажать Enter

6. Ожидаемый результат

Система отображает участника с ролью "водитель"

7. Фактический результат



```
непр + exit
> list --role ВОДИТЕЛЬ
нет записей
```

Система возвращает: "нет записей"

8. Статус

Failed

1. Идентификатор

ТС-005

2. Название

Проверка корректности подсчета участников

3. Описание

Убедиться, что команда `count` возвращает точное количество участников

4. Предварительные условия

В базе 7 участников

5. Шаги выполнения

а) Ввести команду: `count`

б) Нажать Enter

6. Ожидаемый результат

Система возвращает: "Всего участников: 7"

7. Фактический результат

```
> list
1) ян – штурман
2) ф – штурман
3) джордждройд – грузчик
4) вадим – штурман
5) Анна – грузчик
6) Азазлинг – ВОДИТЕЛЬ
7) АННА – грузчик
> count
Всего участников: 6
```

Система возвращает: "Всего участников: 6"

8. Статус

Failed

1. Идентификатор

ТС-006

2. Название

Проверка сохранения данных после удаления последнего участника

3. Описание

Убедиться, что данные сохраняются на диск после удаления последней записи

4. Предварительные условия

В базе 1 участник

5. Шаги выполнения

- а) Ввести команду: remove вадим
- б) Ввести команду: list
- в) Нажать Enter

6. Ожидаемый результат

После удаления list показывает "нет записей"

7. Фактический результат

```
OK: удалён джордж  
> remove вадим  
OK: удалён "вадим"  
> list  
1) вадим – штурман
```

List показывает удаленного участника (устаревшие данные)

8. Статус

Failed

1. Идентификатор

ТС-007

2. Название

Проверка создания файла данных для пустой базы

3. Описание

Убедиться, что файл данных создается даже для пустой базы участников

4. Предварительные условия

Файл members.json отсутствует

5. Шаги выполнения

- а) Запустить приложение
- б) Ввести команду: list
- в) Закрыть приложение

6. Ожидаемый результат

Создается файл members.json с содержимым []

7. Фактический результат

```
> add друн механик  
OK: добавлен "друн" (механик)  
> list  
1) друн – механик  
> remove друн  
OK: удалён "друн"  
> list  
1) друн – механик  
> exit
```

Файл members.json не создается

8. Статус

Failed

1. Идентификатор

ТС-008

2. Название

Проверка текста ошибки при неизвестной роли

3. Описание

Убедиться, что система возвращает понятное сообщение об ошибке при неизвестной роли

4. Предварительные условия

Приложение запущено

5. Шаги выполнения

а) Ввести команду: add Папич водител

б) Нажать Enter

6. Ожидаемый результат

Система возвращает: "роль должна быть одной из: штурман, водитель, грузчик, механик"

7. Фактический результат

```
> add Папич водител  
ERR: unknown command
```

Система возвращает: "ERR: unknown command"

8. Статус

Failed

1. Идентификатор

ТС-009

2. Название

Проверка сообщения об ошибке при неполной команде add

3. Описание

Убедиться, что система возвращает понятное сообщение при неполном вводе команды

4. Предварительные условия

Приложение запущено

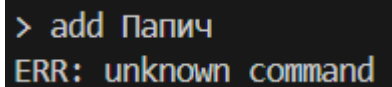
5. Шаги выполнения

- а) Ввести команду: add Папич
- б) Нажать Enter

6. Ожидаемый результат

Система возвращает: "ERR: используйте: add <Имя> <Роль>"

7. Фактический результат



```
> add Папич
ERR: unknown command
```

Система возвращает: "ERR: unknown command"

8. Статус

Failed

Вывод по техническому заданию другой команды: ТЗ и руководство пользователя другой группы поставлено и выполнено удовлетворительно.