Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Петрозаводский государственный университет»

Институт математики и информационных технологий

кафедра прикладная математика и кибернетика

Отчёт по учебному курсу "Верификация программного обеспечения"

09.03.04 - программная инженерия

Выполнил:
студент
Куусела Д.А.

Руководитель
к.ф-м.н, доцент
Кулаков К.А

Петрозаводск — 2023

Оглавление

Глава 1

Объект тестирования

1.1 Описание приложения

В данном приложении реализована шахматная игра с графическим интерфейсом пользователя. Игроки могут совершать ходы, перемещая фигуры на шахматной доске. Программа визуализирует игровое поле, отображает возможные ходы, поддерживает перетаскивание фигур, и воспроизводит звуковые эффекты при совершении ходов.

1.2 Функции приложения

- **Отображение шахматной доски:** Приложение создает графическое представление шахматной доски, разделяя клетки на светлые и темные.
- **Перетаскивание фигур:** Пользователи могут перетаскивать шахматные фигуры по доске для совершения ходов.
- **Подсветка возможных ходов:** Приложение подсвечивает клетки, на которые можно совершить ход, визуально предостерегая игроков от недопустимых ходов.
- **Звуковые эффекты:** Проигрываются звуковые эффекты при совершении ходов, что создает аудиовизуальный опыт игры.
- **Определение возможных ходов:** Программа рассчитывает и отображает возможные ходы для каждой фигуры в соответствии с правилами шахмат.
- **Смена хода и цвета игрока:** Приложение автоматически определяет следующего игрока и следит за сменой хода.

- **Визуализация последнего хода:** Выделение последнего совершенного хода на доске для лучшего восприятия текущей ситуации.
- **Смена темы:** Возможность смены темы оформления шахматной доски.

1.3 Архитектура приложения

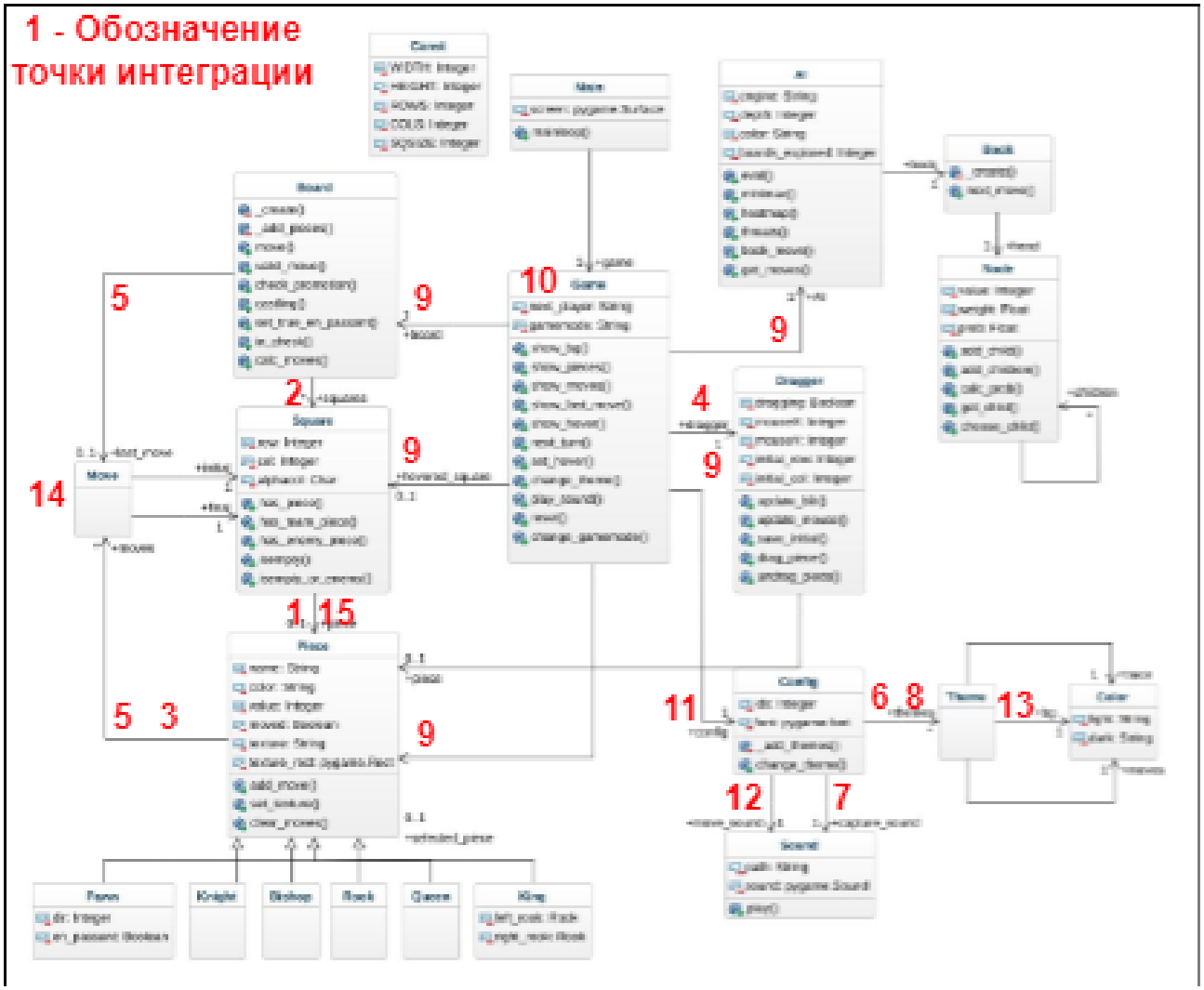

Рис. 1.1: Архитектура приложения

1.4 Описание модулей

- **board.py:** Модуль, содержащий класс `Board`, отвечающий за логику шахматной доски и ходы.
- **dragger.py:** Модуль, содержащий класс `Dragger`, обеспечивающий перетаскивание шахматных фигур средствами библиотеки pygame.
- **game.py:** Модуль, содержащий класс `Game`, представляющий основной игровой процесс и взаимодействие между шахматной доской, игроками и графическим интерфейсом.
- **move.py:** Модуль, содержащий класс `Move`, представляющий отдельный шахматный ход между двумя клетками на доске.
- **piece.py:** Модуль, содержащий базовый класс `Piece`, а также подклассы для каждого типа шахматной фигуры (`Pawn`, `Knight`, `Bishop`, `Rook`, `Queen`, `King`).
- **sound.py:** Модуль, содержащий класс `Sound`, обеспечивающий воспроизведение звуковых эффектов.
- **square.py:** Модуль, содержащий класс `Square`, представляющий отдельную клетку на шахматной доске.
- **theme.py:** Модуль, содержащий класс `Theme`, представляющий тему оформления шахматной доски.

1.5 Описание методов и классов

1.5.1 Модуль board.py

Класс `Board`

- **__init__(self):** Инициализация шахматной доски, установка начальных позиций фигур.
- **move(self, piece, move, testing=False):** Выполнение хода на доске, включая перемещение фигуры, обработку взятия на проходе, превращение пешки и рокировку.
- **valid_move(self, piece, move):** Проверка, является ли переданный ход допустимым для данной фигуры.

- **check_promotion(self, piece, final):** Проверка возможности превращения пешки в ферзя при достижении конечной горизонтали.
- **castling(self, initial, final):** Проверка возможности рокировки.
- **set_true_en_passant(self, piece):** Установка флага возможности взятия на проходе для одной пешки и снятие этого флага для остальных пешек.
- **in_check(self, piece, move):** Проверка, находится ли король после выполнения хода под боем.
- **calc_moves(self, piece, row, col, bool=True):** Рассчет всех возможных ходов для заданной фигуры в указанной позиции.
- **_create(self):** Создание начальной шахматной доски.
- **_add_pieces(self, color):** Размещение начальных фигур на доске для указанного цвета.

1.5.2 Модуль dragger.py

Класс `Dragger`

- **__init__(self):** Инициализация объекта для перетаскивания шахматных фигур.
- **update_blit(self, surface):** Обновление визуализации для отображения перетаскиваемой фигуры.
- **update_mouse(self, pos):** Обновление координат мыши.
- **save_initial(self, pos):** Сохранение начальной позиции для определения начальной клетки при перетаскивании.
- **drag_piece(self, piece):** Начало перетаскивания указанной фигуры.
- **undrag_piece(self):** Завершение перетаскивания фигуры.

1.5.3 Модуль game.py

Класс `Game`

- **__init__(self):** Инициализация игрового процесса, создание доски, объекта для перетаскивания и настроек.

- **show_bg(self, surface):** Отображение фона шахматной доски.
- **show_pieces(self, surface):** Отображение шахматных фигур на доске.
- **show_moves(self, surface):** Отображение возможных ходов фигуры при перетаскивании.
- **show_last_move(self, surface):** Выделение последнего выполненного хода на доске.
- **show_hover(self, surface):** Подсветка клетки, над которой находится указатель мыши.
- **next_turn(self):** Переход к следующему ходу и изменение цвета игрока.
- **set_hover(self, row, col):** Установка подсвеченной клетки при наведении мыши.
- **change_theme(self):** Смена темы оформления доски.
- **play_sound(self, captured=False):** Воспроизведение звуковых эффектов для совершенного хода или взятия фигуры.
- **reset(self):** Сброс игры до начального состояния.

1.5.4 Модуль move.py

Класс `Move`

- **__init__(self, initial, final):** Инициализация объекта хода между двумя клетками.
- **__str__(self):** Возвращает строковое представление хода.
- **__eq__(self, other):** Проверка на равенство двух ходов.

1.5.5 Модуль piece.py

Класс `Piece`

- **__init__(self, name, color, value, texture=None, texture_rect=None):** Инициализация шахматной фигуры с указанными параметрами.

- **set_texture(self, size=80):** Установка текстуры фигуры.
- **add_move(self, move):** Добавление возможного хода для фигуры.
- **clear_moves(self):** Очистка списка возможных ходов.

Подкласс `Pawn`

- **__init__(self, color):** Инициализация пешки с учетом цвета и направления движения.

Подкласс `Knight`

- **__init__(self, color):** Инициализация коня с указанием цвета.

Подкласс `Bishop`

- **__init__(self, color):** Инициализация слона с указанием цвета.

Подкласс `Rook`

- **__init__(self, color):** Инициализация ладьи с указанием цвета.

Подкласс `Queen`

- **__init__(self, color):** Инициализация ферзя с указанием цвета.

Подкласс `King`

- **__init__(self, color):** Инициализация короля с указанием цвета.

1.5.6 Модуль sound.py

Класс `Sound`

- **__init__(self, path):** Инициализация объекта звукового эффекта с указанием пути к звуковому файлу.
- **play(self):** Воспроизведение звукового эффекта.

1.5.7 Модуль square.py

Класс `Square`

- **__init__(self, row, col, piece=None):** Инициализация клетки доски с указанием позиции и, при необходимости, фигуры.
- **__eq__(self, other):** Проверка на равенство двух клеток.
- **has_piece(self):** Проверка, содержит ли клетка фигуру.
- **isempty(self):** Проверка, является ли клетка пустой.
- **has_team_piece(self, color):** Проверка, содержит ли клетка фигуру своей команды.
- **has_enemy_piece(self, color):** Проверка, содержит ли клетка фигуру противоположной команды.
- **isempty_or_enemy(self, color):** Проверка, является ли клетка пустой или содержит фигуру противоположной команды.
- **in_range(*args):** Проверка, находится ли каждый из переданных аргументов в пределах шахматной доски.
- **get_alphacol(col):** Получение буквенного обозначения колонки на шахматной доске

1.5.8 Модуль theme.py

Класс: Theme

- **__init__(light_bg, dark_bg, light_trace, dark_trace, light_moves, dark_m**
  Инициализация объекта темы с заданными цветами для фона, подсветки и возможных ходов.

Класс: Color

- **__init__(light, dark):** Инициализация объекта цвета с заданными значениями для светлого и темного вариантов.

Глава 2

Стратегия тестирования

2.1 Стратегия блочного тестирования

При блочном тестировании каждый модуль программы тестируется отдельно, чтобы убедиться в том, что он работает корректно в изоляции. Ниже приведена стратегия блочного тестирования:

- **Тестирование модуля** `piece.py`**:**
  - Проверка корректности инициализации объекта Piece.
  - Тестирование метода `set_texture` для различных размеров.
  - Проверка корректности добавления и очистки списка ходов (`add_move`, `clear_moves`).
  - Тестирование создания объектов различных типов шахматных фигур (`Pawn`, `Knight`, `Bishop`, `Rook`, `Queen`, `King`).
- **Тестирование модуля** `sound.py`**:**
  - Проверка корректности инициализации объекта Sound.
  - Тестирование воспроизведения звука (`play`).
- **Тестирование модуля** `square.py`**:**
  - Проверка корректности инициализации объекта Square.
  - Проверка методов `has_piece`, `isempty`, `has_team_piece`, `has_enemy_piece`, `isempty_o`
  - Тестирование статического метода `in_range`.
  - Проверка корректности получения буквенного обозначения колонки (`get_alphacol`).

- **Тестирование модуля** `theme.py`**:**
  - Проверка корректности инициализации объекта Theme.
  - Проверка корректности инициализации объекта Color.
- **Тестирование модуля** `move.py`**:**
  - Проверка корректности инициализации объекта Move.
  - Проверка методов `__str__` и `__eq__`.
- **Тестирование модуля** `dragger.py`**:**
  - Проверка корректности инициализации объекта Dragger.
  - Тестирование методов `update_blit`, `update_mouse`, `save_initial`, `drag_piece`, `undrag_piece`.
- **Тестирование модуля** `board.py`**:**
  - Проверка корректности инициализации объекта Board.
  - Тестирование методов `move`, `valid_move`, `check_promotion`, `castling`, `set_true_en_pa in_check`, `calc_moves`, `_create`, `_add_pieces`.
- **Тестирование модуля** `config.py`**:**
  - Проверка корректности инициализации объекта Config.
  - Проверка изменения темы (`change_theme`).
- **Тестирование модуля** `game.py`**:**
  - Проверка корректности инициализации объекта Game.
  - Тестирование методов `next_turn`, `set_hover`, `change_theme`, `play_sound`, `reset`.
  - Проверка корректности отрисовки игрового поля и фигур (`show_bg`, `show_pieces`, `show_moves`, `show_last_move`, `show_hover`).

2.2 Стратегия интеграционного тестирования

Интеграционное тестирование проводится для проверки взаимодействия между различными модулями системы. Ниже представлена стратегия интеграционного тестирования:

- **Точка интеграции между `piece.py` и `square.py`:**
  - Тестирование взаимодействия объектов Piece и Square при размещении фигур на доске.
- **Точка интеграции между `square.py` и `board.py`:**
  - Проверка корректности размещения объектов Square на доске в объекте Board.
- **Точка интеграции между `move.py` и `piece.py`:**
  - Тестирование правильности обработки и хранения ходов в объектах Piece.
- **Точка интеграции между `dragger.py` и `game.py`:**
  - Проверка корректности перемещения фигур с использованием объекта Dragger в объекте Game.
- **Точка интеграции между `board.py`, `move.py` и `piece.py`:**
  - Тестирование взаимодействия модулей при выполнении ходов и изменении состояния доски.
- **Точка интеграции между `config.py` и `theme.py`:**
  - Проверка корректности применения темы в объекте Config.
- **Точка интеграции между `sound.py` и `game.py`:**
  - Тестирование воспроизведения звуков при различных событиях в объекте Game.
- **Точка интеграции между `game.py` и `theme.py`:**
  - Проверка взаимодействия модулей при изменении темы в объекте Game.
- **Точка интеграции между `game.py`, `board.py`, `square.py`, `piece.py`, `move.py` и `dragger.py`:**
  - Тестирование полного цикла игрового процесса с участием всех модулей.
- **Точка интеграции между `game.py` и внешними библиотеками (Pygame):**
  - Проверка корректного взаимодействия с библиотекой Pygame при отображении игрового интерфейса и обработке событий.

- **Точка интеграции между `game.py` и `config.py`:**
  - Тестирование взаимодействия модулей при изменении настроек игры в объекте Game.
- **Точка интеграции между `game.py` и `sound.py`:**
  - Проверка воспроизведения звуков в объекте Game при различных игровых событиях.
- **Точка интеграции между `theme.py`, `color.py` и внешними ресурсами (изображения, звуки):**
  - Тестирование корректного использования и отображения ресурсов, а также их взаимодействия с темами в объекте Theme.
- **Точка интеграции между `game.py` и `move.py`:**
  - Проверка взаимодействия при выполнении и отображении игровых ходов.
- **Точка интеграции между `square.py` и `piece.py`:**
  - Тестирование правильности связи объектов Square и Piece при различных сценариях игры.

2.3 Стратегия аттестационного тестирования

Аттестационное тестирование выполняется для оценки функциональных возможностей приложения с учетом его требований и спецификаций. Ниже представлена стратегия аттестационного тестирования:

- **Тестирование инициализации:**
  - Проверка корректной инициализации объектов при запуске приложения.
  - Убеждение в правильной установке начальных значений переменных и объектов.
- **Тестирование перемещения фигур:**
  - Проверка возможности перемещения фигур по шахматной доске.
  - Убеждение в соответствии движения фигур шахматным правилам.

- **Тестирование правил шахмат:**
  - Проверка корректности обработки правил шахмат, таких как рокировка, взятие на проходе и превращение пешки.
  - Убеждение в правильном расчете доступных ходов для каждой фигуры.
- **Тестирование игрового процесса:**
  - Проверка правильной смены хода между игроками.
  - Убеждение в корректной обработке завершения игры (шах, мат, пат).
- **Тестирование интерфейса:**
  - Проверка отображения шахматной доски и фигур на экране.
  - Убеждение в правильном отображении текстур и цветов.
- **Тестирование звуковых эффектов:**
  - Проверка воспроизведения звуков при различных событиях в игре (ход фигуры, взятие фигуры, завершение игры).
  - Убеждение в корректном соответствии звуков с игровыми событиями.
- **Тестирование изменения настроек:**
  - Проверка возможности смены темы в приложении.
  - Убеждение в корректном применении новых настроек.
- **Тестирование сохранения и загрузки игры:**
  - Проверка возможности сохранения текущего состояния игры.
  - Убеждение в корректном восстановлении игры после загрузки.

2.3.1 Стратегия нагрузочного тестирования

Оценка пропускной способности:

- Имитация сценариев, включающих активное взаимодействие с приложением, такие как выполнение множества ходов, изменение темы и другие действия пользователя.
- Исследование, как система реагирует на увеличение числа одновременных пользователей.

Измерение времени отклика:

- Регистрация времени отклика системы на запросы при различных уровнях нагрузки.
- Анализ изменения времени отклика при увеличении числа одновременных пользователей.

Тестирование стабильности системы:

- Запуск тестов на устойчивость системы под долговременной нагрузкой.
- Мониторинг ресурсов и производительности в течение продолжительного времени.

Оценка максимальной нагрузки:

- Увеличение числа одновременных пользователей до достижения предела стабильной работы системы.
- Определение точки, при которой система начинает проявлять признаки нестабильности.

Тестирование под высокой нагрузкой:

- Имитация сценариев с максимальной активностью пользователей.
- Проверка, как система справляется с пиковой нагрузкой.

Анализ использования ресурсов:

- Мониторинг использования ресурсов (память, процессор) в условиях высокой активности пользователей.
- Идентификация участков, где системе требуется оптимизация для улучшения производительности.

Тестирование долгосрочной стабильности:

- Оценка стабильности системы при продолжительной нагрузке в течение нескольких часов.
- Мониторинг изменений производительности со временем.

2.4 Критерии прохождения тестирования

2.4.1 Критерии блочного тестирования:

- **Все объекты классов (Piece, Pawn, Knight, Bishop, Rook, Queen, King, Sound, Square, Theme) успешно инициализируются без ошибок.**
- **Методы и функциональность каждого класса работают корректно.**
- **Фигуры корректно отображаются на доске, их текстуры устанавливаются правильно.**
- **Звуковые эффекты воспроизводятся без искажений.**

2.4.2 Критерии интеграционного тестирования:

- **Взаимодействие между модулями происходит без ошибок.**
- **Корректное размещение фигур на доске в объекте Board.**
- **Обработка ходов и изменение состояния доски происходит без сбоев.**

2.4.3 Критерии аттестационного тестирования:

- **Возможность перемещения фигур в соответствии с правилами шахмат.**
- **Правильная обработка игровых сценариев, включая проверку наличия шаха, матов, пата.**
- **Корректное отображение игрового интерфейса и изменение темы приложения.**
- **Сохранение и загрузка игры работают стабильно.**

2.4.4 Критерии нагрузочного тестирования:

- **Приложение способно обрабатывать увеличение числа одновременных пользователей без значительного снижения производительности.**
- **Время отклика системы остается приемлемым при максимальной нагрузке.**
- **Система стабильна и не проявляет сбоев при долгосрочной нагрузке.**

2.4.5 Дополнительные критерии:

- **Покрытие кода тестами достаточно для уверенности в качестве.**
- **Отсутствие критических ошибок и вылетов приложения.**
- **Полное соответствие функциональным требованиям и спецификациям.**

2.5 Условия возобновления и приостановки выполнения тестов:

Условия возобновления:

- **Исправление критических ошибок, выявленных в процессе тестирования.**
- **Добавление новой функциональности, требующей дополнительного тестирования.**
- **Изменение внешних зависимостей (библиотек, платформ), которые могут повлиять на работу приложения.**
- **Изменения в требованиях к приложению, влияющие на его функциональность.**

Условия приостановки:

- **Ожидание появления новой функциональности, которая требует тестирования.**
- **Ожидание обновлений внешних зависимостей.**
- **Ожидание уточнения требований или спецификаций.**
- **Отсутствие ресурсов для проведения тестирования.**

Глава 3

Детальный план тестирования

3.1 Блочное тестирования

Для написания и выполнения блочных тестов (unit tests) на Python часто используется встроенная библиотека unittest. unittest предоставляет фреймворк для создания и выполнения тестовых случаев, проверки утверждений и организации тестов.

3.1.1 Тестирование модуля `piece.py`:

B1: Проверка корректности инициализации объекта Piece.

- **Входные данные:** Параметры name='bishop', color='white', value=3.0.
- **Шаги:**
  1. Создать объект Piece с указанными параметрами.
- **Результат:** Объект успешно создан, все поля инициализированы корректно.

B2: Тестирование метода `set_texture` для различных размеров.

- **Входные данные:** Объект Piece с параметрами name='queen', color='black', value=9.0.
- **Шаги:**
  1. Вызвать метод `set_texture` с разными значениями размера.
- **Результат:** Для каждого размера текстура устанавливается без ошибок.

B3: Проверка корректности добавления и очистки списка ходов (`add_move`, `clear_moves`).

- **Входные данные:** Объект Piece с пустым списком ходов.
- **Шаги:**
  1. Добавить несколько ходов с использованием метода `add_move`.
  2. Очистить список ходов с использованием метода `clear_moves`.
- **Результат:** Список ходов успешно добавляется и корректно очищается.

B4: Тестирование создания объектов различных типов шахматных фигур (`Pawn`, `Knight`, `Bishop`, `Rook`, `Queen`, `King`).

- **Входные данные:** Различные цвета и типы фигур.
- **Шаги:**
  1. Создать объекты для каждого типа фигур.
- **Результат:** Объекты каждого типа успешно созданы, и их параметры инициализированы корректно.

3.1.2 Тестирование модуля `sound.py`:

B5: Проверка корректности инициализации объекта Sound.

- **Входные данные:** Путь к звуковому файлу capture.wav.
- **Шаги:**
  1. Попытаться создать объект Sound с указанным путем.
- **Результат:** Объект Sound успешно создан, звуковой файл загружен без ошибок.

B6: Тестирование воспроизведения звука (`play`).

- **Входные данные:** Объект Sound с звуковым файлом.
- **Шаги:**
  1. Вызвать метод `play` для проигрывания звука.
- **Результат:** Звук успешно воспроизводится, без искажений и задержек.

3.1.3 Тестирование модуля `square.py`:

B7: Проверка корректности инициализации объекта Square.

- **Входные данные:** Значения row и col для создания объекта Square.
- **Шаги:**
  1. Попытаться создать объект Square с заданными значениями row и col.
- **Результат:** Объект Square успешно создан, значения row и col установлены правильно.

B8: Проверка методов `has_piece`, `isempty`, `has_team_piece`, `has_enemy_piece`, `isempty_o`

- **Входные данные:** Объект Square с фигурой и без фигуры, объекты Piece различных цветов.
- **Шаги:**
  1. Проверить значения методов `has_piece`, `isempty`, `has_team_piece`, `has_enemy_piece`, `isempty_or_enemy` для Square с фигурой и без фигуры.
  2. Попытаться установить фигуру на Square и повторить проверку.
- **Результат:** Методы корректно возвращают ожидаемые значения, учитывая наличие или отсутствие фигуры.

B9: Тестирование статического метода `in_range`.

- **Входные данные:** Различные значения для тестирования диапазона.
- **Шаги:**
  1. Проверить значения метода `in_range` для различных входных данных.
- **Результат:** Метод возвращает корректные значения в зависимости от входных данных.

B10: Проверка корректности получения буквенного обозначения колонки (`get_alphacol`).

- **Входные данные:** Различные значения для тестирования получения буквенного обозначения.

- **Шаги:**

    1. Проверить значения метода `get_alphacol` для различных входных данных.

- **Результат:** Метод возвращает корректные буквенные обозначения в зависимости от входных данных.

3.1.4 Тестирование модуля `theme.py`:

B11: Проверка корректности инициализации объекта Theme.

- **Входные данные:** Значения цветов для темы.

- **Шаги:**

    1. Попытаться создать объект Theme с заданными значениями цветов.

- **Результат:** Объект Theme успешно создан, значения цветов установлены правильно.

B12: Проверка корректности инициализации объекта Color.

- **Входные данные:** Значения цветовых компонент (RGB).

- **Шаги:**

    1. Попытаться создать объект Color с заданными значениями цветовых компонент.

- **Результат:** Объект Color успешно создан, значения цветовых компонент установлены правильно.

3.1.5 Тестирование модуля `move.py`:

B13: Проверка корректности инициализации объекта Move.

- **Входные данные:** Начальная и конечная позиции, фигура.

- **Шаги:**

    1. Попытаться создать объект Move с заданными значениями.

- **Результат:** Объект Move успешно создан, значения установлены правильно.

B14: Проверка метода `__str__`.

- **Входные данные:** Объект Move с определенными значениями.

- **Шаги:**

  1. Вызвать метод `__str__` для объекта Move.

- **Результат:** Метод `__str__` возвращает корректное строковое представление объекта Move.

B15: Проверка метода `__eq__`.

- **Входные данные:** Два объекта Move с одинаковыми и разными значениями.

- **Шаги:**

  1. Сравнить два объекта Move с использованием метода `__eq__`.

- **Результат:** Метод `__eq__` возвращает `True` для двух объектов с одинаковыми значениями и `False` в противном случае.

3.1.6 Тестирование модуля `dragger.py`:

B16: Проверка корректности инициализации объекта Dragger.

- **Входные данные:** Нет.

- **Шаги:**

  1. Создать объект Dragger.

- **Результат:** Объект Dragger успешно создан без ошибок.

B17: Тестирование метода `update_blit`.

- **Входные данные:** Объект Dragger, шахматная доска.

- **Шаги:**

  1. Вызвать метод `update_blit` с заданными параметрами.

- **Результат:** Метод `update_blit` выполняется без ошибок, обновляет отображение доски.

B18: Тестирование метода `update_mouse`.

- **Входные данные:** Объект Dragger, координаты мыши.

- **Шаги:**

    1. Вызвать метод `update_mouse` с заданными параметрами.

- **Результат:** Метод `update_mouse` выполняется без ошибок, обновляет состояние мыши.

B19: Тестирование метода `save_initial`.

- **Входные данные:** Объект Dragger, шахматная доска, начальные координаты.

- **Шаги:**

    1. Вызвать метод `save_initial` с заданными параметрами.

- **Результат:** Метод `save_initial` выполняется без ошибок, сохраняет начальное состояние доски.

B20: Тестирование метода `drag_piece`.

- **Входные данные:** Объект Dragger, шахматная доска, координаты фигуры.

- **Шаги:**

    1. Вызвать метод `drag_piece` с заданными параметрами.

- **Результат:** Метод `drag_piece` выполняется без ошибок, начинает перемещение фигуры.

B21: Тестирование метода `undrag_piece`.

- **Входные данные:** Объект Dragger, шахматная доска.

- **Шаги:**

    1. Вызвать метод `undrag_piece` с заданными параметрами.

- **Результат:** Метод `undrag_piece` выполняется без ошибок, завершает перемещение фигуры.

3.1.7 Тестирование модуля `board.py`:

B22: Проверка корректности инициализации объекта Board.

- **Входные данные:** Нет.
- **Шаги:**
  1. Создать объект Board.
- **Результат:** Объект Board успешно создан без ошибок.

B23: Тестирование метода `move`.

- **Входные данные:** Объект Board, начальные и конечные координаты хода.
- **Шаги:**
  1. Вызвать метод `move` с заданными параметрами.
- **Результат:** Метод `move` выполняется без ошибок, фигура перемещена на новую позицию.

B24: Тестирование метода `valid_move`.

- **Входные данные:** Объект Board, начальные и конечные координаты хода.
- **Шаги:**
  1. Вызвать метод `valid_move` с заданными параметрами.
- **Результат:** Метод `valid_move` возвращает корректное значение (правильность хода).

B25: Тестирование метода `check_promotion`.

- **Входные данные:** Объект Board, координаты завершения хода пешки.
- **Шаги:**
  1. Вызвать метод `check_promotion` с заданными параметрами.
- **Результат:** Метод `check_promotion` возвращает корректное значение (необходимость превращения пешки).

B26: Тестирование метода `castling`.

- **Входные данные:** Объект Board, тип рокировки (длинная или короткая).
- **Шаги:**
    1. Вызвать метод `castling` с заданными параметрами.
- **Результат:** Метод `castling` выполняется без ошибок, происходит рокировка.

B27: Тестирование метода `set_true_en_passant`.

- **Входные данные:** Объект Board.
- **Шаги:**
    1. Вызвать метод `set_true_en_passant` с заданными параметрами.
- **Результат:** Метод `set_true_en_passant` выполняется без ошибок, устанавливается флаг корректно.

B28: Тестирование метода `in_check`.

- **Входные данные:** Объект Board, цвет короля.
- **Шаги:**
    1. Вызвать метод `in_check` с заданными параметрами.
- **Результат:** Метод `in_check` возвращает корректное значение (нахождение короля под шахом).

B29: Тестирование метода `calc_moves`.

- **Входные данные:** Объект Board, координаты фигуры.
- **Шаги:**
    1. Вызвать метод `calc_moves` с заданными параметрами.
- **Результат:** Метод `calc_moves` возвращает корректные возможные ходы для фигуры.

B30: Тестирование метода `_create`.

- **Входные данные:** Объект Board.
- **Шаги:**

1. Вызвать метод `_create` с заданными параметрами.

- **Результат:** Метод `_create` выполняется без ошибок, создает начальное состояние доски.

В31: Тестирование метода `_add_pieces`.

- **Входные данные:** Объект Board.
- **Шаги:**

1. Вызвать метод `_add_pieces` с заданными параметрами.

- **Результат:** Метод `_add_pieces` выполняется без ошибок, добавляет фигуры на доску.

3.1.8 Тестирование модуля `config.py`:

В32: Проверка корректности инициализации объекта Config.

- **Входные данные:** Нет.
- **Шаги:**

1. Создать объект Config.

- **Результат:** Объект Config успешно создан без ошибок.

В33: Тестирование метода `change_theme`.

- **Входные данные:** Объект Config, новая тема.
- **Шаги:**

1. Вызвать метод `change_theme` с заданными параметрами.

- **Результат:** Метод `change_theme` выполняется без ошибок, тема приложения изменяется на новую.

3.1.9 Тестирование модуля `game.py`:

B34: Проверка корректности инициализации объекта Game.

- **Входные данные:** Нет.
- **Шаги:**
  1. Создать объект Game.
- **Результат:** Объект Game успешно создан без ошибок.

B35: Тестирование метода `next_turn`.

- **Входные данные:** Объект Game, текущий ход.
- **Шаги:**
  1. Вызвать метод `next_turn` с текущим ходом.
- **Результат:** Метод `next_turn` успешно меняет текущего игрока на противоположного.

B36: Тестирование метода `set_hover`.

- **Входные данные:** Объект Game, координаты клетки.
- **Шаги:**
  1. Вызвать метод `set_hover` с заданными координатами клетки.
- **Результат:** Метод `set_hover` корректно устанавливает клетку под курсором.

B37: Тестирование метода `change_theme`.

- **Входные данные:** Объект Game, новая тема.
- **Шаги:**
  1. Вызвать метод `change_theme` с заданной темой.
- **Результат:** Метод `change_theme` успешно изменяет тему приложения на новую.

B38: Тестирование метода `play_sound`.

- **Входные данные:** Объект Game, тип звука.

- **Шаги:**

  1. Вызвать метод `play_sound` с заданным типом звука.

- **Результат:** Метод `play_sound` воспроизводит звук без ошибок.

B39: Тестирование метода `reset`.

- **Входные данные:** Объект Game.

- **Шаги:**

  1. Вызвать метод `reset`.

- **Результат:** Метод `reset` сбрасывает состояние игры к начальному без ошибок.

B40: Проверка корректности отрисовки игрового поля и фигур.

- **Входные данные:** Объект Game.

- **Шаги:**

  1. Вызвать методы отрисовки игрового поля и фигур (`show_bg`, `show_pieces`, `show_moves`, `show_last_move`, `show_hover`).

- **Результат:** Все методы отрисовки работают корректно и не вызывают ошибок.

3.2 Интеграционное тестирования

3.2.1 Точка интеграции I1: `piece.py` и `square.py`

Тип теста: Позитивный

I1.1: Тестирование успешного размещения фигуры на доске.

- **Входные данные:** Объекты Piece и Square.

- **Шаги:**

  1. Создать объект Piece.
  2. Создать объект Square.
  3. Разместить фигуру на доске, вызвав соответствующий метод.

- **Результат:** Фигура успешно размещена на доске, объекты Piece и Square взаимодействуют корректно.

Тип теста: Негативный

I1.2: Тестирование размещения фигуры на уже занятой клетке.

- **Входные данные:** Объекты Piece и Square, уже занятая клетка.
- **Шаги:**
  1. Создать объект Piece.
  2. Создать объект Square и разместить на ней другую фигуру.
  3. Попытаться разместить первую фигуру на той же клетке.
- **Результат:** Операция размещения фигуры на уже занятой клетке завершается ошибкой, взаимодействие обнаружено.

3.2.2 Точка интеграции I2: `square.py` и `board.py`

Тип теста: Позитивный

I2.1: Тестирование успешного размещения объектов Square на доске.

- **Входные данные:** Объекты Square и Board.
- **Шаги:**
  1. Создать объекты Square.
  2. Создать объект Board.
  3. Разместить квадраты на доске, вызвав соответствующий метод.
- **Результат:** Квадраты успешно размещены на доске, объекты Square и Board взаимодействуют корректно.

Тип теста: Негативный

I2.2: Тестирование размещения объектов Square за пределами доски.

- **Входные данные:** Объекты Square и Board.
- **Шаги:**
  1. Создать объекты Square.

2. Создать объект Board.

3. Попытаться разместить квадраты за пределами доски.

- **Результат:** Операция размещения объектов Square за пределами доски завершается ошибкой, взаимодействие обнаружено.

3.2.3 Точка интеграции I3: `move.py` и `piece.py`

Тип теста: Позитивный

I3.1: Тестирование корректной обработки и хранения ходов в объектах Piece.

- **Входные данные:** Объекты Piece и Move.

- **Шаги:**

1. Создать объект Piece.

2. Создать объект Move.

3. Произвести ход для объекта Piece с использованием объекта Move.

- **Результат:** Ход успешно обработан и сохранен в объекте Piece, взаимодействие корректно.

Тип теста: Негативный

I3.2: Тестирование обработки некорректного хода в объекте Piece.

- **Входные данные:** Объект Piece.

- **Шаги:**

1. Создать объект Piece.

2. Попытаться выполнить некорректный ход для объекта Piece.

- **Результат:** Операция обработки некорректного хода завершается ошибкой, взаимодействие обнаружено.

3.2.4 Точка интеграции I4: `dragger.py` и `game.py`

Тип теста: Позитивный

I4.1: Тестирование корректного перемещения фигур с использованием объекта Dragger в объекте Game.

- **Входные данные:** Объекты Dragger и Game.
- **Шаги:**
  1. Создать объект Game.
  2. Создать объект Dragger.
  3. Имитировать перемещение фигуры с использованием Dragger в объекте Game.
- **Результат:** Фигура успешно перемещена с помощью Dragger в объекте Game, взаимодействие корректно.

Тип теста: Негативный

I4.2: Тестирование обработки некорректного перемещения фигуры в объекте Game.

- **Входные данные:** Объект Game.
- **Шаги:**
  1. Создать объект Game.
  2. Попытаться выполнить некорректное перемещение фигуры в объекте Game.
- **Результат:** Операция обработки некорректного перемещения завершается ошибкой, взаимодействие обнаружено.

3.2.5 Точка интеграции I5: `board.py`, `move.py` и `piece.py`

Тип теста: Позитивный

I5.1: Тестирование корректного выполнения ходов и изменения состояния доски.

- **Входные данные:** Объекты Board, Move и Piece.
- **Шаги:**
  1. Создать объект Board.
  2. Создать объект Piece.
  3. Создать объект Move.
  4. Выполнить ход с использованием объектов Piece и Move в объекте Board.

- **Результат:** Ход успешно выполнен, состояние доски корректно изменено.

Тип теста: Негативный

I5.2: Тестирование обработки некорректного выполнения ходов в объекте Board.

- **Входные данные:** Объект Board.
- **Шаги:**
  1. Создать объект Board.
  2. Попытаться выполнить некорректный ход в объекте Board.
- **Результат:** Операция обработки некорректного хода завершается ошибкой, взаимодействие обнаружено.

3.2.6 Точка интеграции I6: `config.py` и `theme.py`

Тип теста: Позитивный

I6.1: Тестирование успешного изменения темы в объекте Config.

- **Входные данные:** Объект Config и Theme.
- **Шаги:**
  1. Создать объект Config.
  2. Создать объект Theme.
  3. Применить тему к объекту Config.
- **Результат:** Тема успешно применена к объекту Config, изменения видны.

Тип теста: Негативный

I6.2: Тестирование обработки ошибок при некорректной теме в объекте Config.

- **Входные данные:** Объект Config.
- **Шаги:**
  1. Создать объект Config.
  2. Попытаться применить некорректную тему к объекту Config.
- **Результат:** Происходит обработка ошибки при попытке применить некорректную тему, взаимодействие обнаружено.

3.2.7 Точка интеграции I7: `sound.py` и `game.py`

Тип теста: Позитивный

I7.1: Тестирование успешного воспроизведения звука в объекте Game.

- **Входные данные:** Объект Sound и объект Game.
- **Шаги:**
  1. Создать объект Sound.
  2. Создать объект Game.
  3. Инициировать воспроизведение звука в объекте Game.
- **Результат:** Звук успешно воспроизведен в объекте Game, аудиофайлы проиграны корректно.

Тип теста: Негативный

I7.2: Тестирование обработки ошибок при отсутствии звукового файла в объекте Sound.

- **Входные данные:** Объект Sound и объект Game.
- **Шаги:**
  1. Создать объект Sound без указания аудиофайла.
  2. Создать объект Game.
  3. Попытаться воспроизвести звук в объекте Game.
- **Результат:** Обработка ошибки при отсутствии звукового файла в объекте Sound, взаимодействие обнаружено.

3.2.8 Точка интеграции I8: `game.py` и `theme.py`

Тип теста: Позитивный

I8.1: Тестирование успешного изменения темы в объекте Game.

- **Входные данные:** Объект Game и объект Theme.
- **Шаги:**
  1. Создать объект Game.

2. Создать объект Theme.

3. Применить новую тему к объекту Game.

- **Результат:** Тема в объекте Game изменена успешно, интерфейс отображается согласно новой теме.

Тип теста: Негативный

I8.2: Тестирование обработки ошибок при попытке применения недопустимой темы в объекте Game.

- **Входные данные:** Объект Game и объект Theme с недопустимой темой.

- **Шаги:**

1. Создать объект Game.

2. Создать объект Theme с недопустимой темой.

3. Попытаться применить недопустимую тему к объекту Game.

- **Результат:** Обработка ошибки при попытке применения недопустимой темы в объекте Game, взаимодействие обнаружено.

3.2.9 Точка интеграции I9: `game.py`, `board.py`, `square.py`, `piece.py`, `move.py` и `dragger.py`

Тип теста: Позитивный

I9.1: Тестирование полного цикла игрового процесса.

- **Входные данные:** Объекты Game, Board, Square, Piece, Move и Dragger.

- **Шаги:**

1. Создать все необходимые объекты: Game, Board, Square, Piece, Move и Dragger.

2. Выполнить полный цикл игрового процесса (начать игру, выполнить ходы, завершить игру).

- **Результат:** Все модули успешно взаимодействуют в течение полного цикла игрового процесса, игра завершается корректно.

Тип теста: Негативный

I9.2: Тестирование обработки ошибок взаимодействия между модулями в процессе игры.

- **Входные данные:** Объекты Game, Board, Square, Piece, Move и Dragger с некорректной конфигурацией.
- **Шаги:**
  1. Создать объекты Game, Board, Square, Piece, Move и Dragger с некорректной конфигурацией.
  2. Попытаться выполнить полный цикл игрового процесса.
- **Результат:** Обработка ошибок при некорректной конфигурации модулей, взаимодействие обнаружено и обработано.

3.2.10 Точка интеграции I10: `game.py` и внешними библиотеками (Pygame)

Тип теста: Позитивный

I10.1: Тестирование взаимодействия с библиотекой Pygame.

- **Входные данные:** Объект Game и библиотека Pygame.
- **Шаги:**
  1. Запустить приложение и инициализировать объект Game.
  2. Проверить, что игровой интерфейс правильно отображается.
  3. Выполнить несколько событий (например, кликов мыши) и убедиться, что обработка событий выполняется корректно.
- **Результат:** Игровой интерфейс отображается корректно, события обрабатываются без ошибок.

Тип теста: Негативный

I10.2: Тестирование обработки ошибок взаимодействия с Pygame.

- **Входные данные:** Объект Game и библиотека Pygame с некорректной конфигурацией.

- **Шаги:**

  1. Создать объект Game с некорректной конфигурацией Pygame.
  2. Запустить приложение.
  3. Попытаться выполнить действия взаимодействия с интерфейсом.

- **Результат:** Обработка ошибок при некорректной конфигурации Pygame, взаимодействие обнаружено и обработано.

3.2.11 Точка интеграции I11: `game.py` и `config.py`

Тип теста: Позитивный

I11.1: Тестирование взаимодействия при изменении настроек игры.

- **Входные данные:** Объект Game и объект Config с различными настройками.

- **Шаги:**

  1. Инициализировать объект Game и объект Config с различными настройками.
  2. Внести изменения в настройки игры через объект Config.
  3. Убедиться, что изменения корректно применяются в объекте Game.

- **Результат:** Изменения в настройках игры, внесенные через объект Config, отображаются корректно в объекте Game.

Тип теста: Негативный

I11.2: Тестирование обработки ошибок при некорректных настройках.

- **Входные данные:** Объект Game и объект Config с некорректными настройками.

- **Шаги:**

  1. Инициализировать объект Game и объект Config с некорректными настройками.
  2. Попытаться изменить настройки игры через объект Config.

- **Результат:** Обработка ошибок при попытке изменить настройки игры с некорректными данными.

3.2.12 Точка интеграции I12: `game.py` и `sound.py`

Тип теста: Позитивный

I12.1: Тестирование воспроизведения звуков при различных игровых событиях.

- **Входные данные:** Объект Game и объект Sound с предварительно загруженными звуковыми файлами.
- **Шаги:**
  1. Инициализировать объект Game и объект Sound.
  2. Выполнить игровые действия, приводящие к событиям, требующим воспроизведения звуков (например, совершить ход фигурой).
  3. Проверить, что звуки воспроизводятся корректно.
- **Результат:** Звуки воспроизводятся в соответствии с игровыми событиями без искажений.

Тип теста: Негативный

I12.2: Тестирование обработки ошибок при отсутствии звуковых файлов.

- **Входные данные:** Объект Game и объект Sound без загруженных звуковых файлов.
- **Шаги:**
  1. Инициализировать объект Game и объект Sound без загруженных звуковых файлов.
  2. Выполнить игровые действия, приводящие к событиям, требующим воспроизведения звуков.
- **Результат:** Обработка ошибок при попытке воспроизвести звуки без загруженных файлов.

3.2.13 Точка интеграции I13: `theme.py`, `color.py` и внешние ресурсы

Тип теста: Позитивный

I13.1: Тестирование корректного использования и отображения изображений и звуков при изменении темы.

- **Входные данные:** Объект Theme, объект Color, и внешние ресурсы (изображения и звуки).
- **Шаги:**
  1. Инициализировать объект Theme и объект Color с загруженными ресурсами.
  2. Изменить тему с помощью объекта Theme.
  3. Проверить, что изображения и звуки соответствуют выбранной теме.
- **Результат:** Ресурсы корректно отображаются и воспроизводятся в соответствии с выбранной темой.

Тип теста: Негативный

I13.2: Тестирование обработки ошибок при отсутствии внешних ресурсов.

- **Входные данные:** Объект Theme и объект Color без загруженных изображений и звуков.
- **Шаги:**
  1. Инициализировать объект Theme и объект Color без загруженных изображений и звуков.
  2. Изменить тему с помощью объекта Theme.
- **Результат:** Обработка ошибок при попытке использования отсутствующих ресурсов.

3.2.14 Точка интеграции I14: `game.py` и `move.py`

Тип теста: Позитивный

I14.1: Тестирование взаимодействия при выполнении и отображении игровых ходов.

- **Входные данные:** Объект Game и объект Move с корректно сформированными данными.
- **Шаги:**
  1. Инициализировать объект Game.
  2. Выполнить игровой ход с использованием объекта Move.

3. Проверить, что ход корректно отображается на доске и в игровом интерфейсе.

- **Результат:** Игровой ход взаимодействует с объектом Game, и его результат корректно отображается.

Тип теста: Негативный

I14.2: Тестирование обработки ошибок при выполнении некорректного хода.

- **Входные данные:** Объект Game и объект Move с некорректными данными.
- **Шаги:**
  1. Инициализировать объект Game.
  2. Попытаться выполнить некорректный игровой ход с использованием объекта Move.
- **Результат:** Обработка ошибок при попытке выполнения некорректного хода.

3.2.15 Точка интеграции I15: `square.py` и `piece.py`

Тип теста: Позитивный

I15.1: Тестирование правильности связи объектов Square и Piece при размещении фигуры.

- **Входные данные:** Объект Square и объект Piece с корректными данными.
- **Шаги:**
  1. Инициализировать объект Square.
  2. Разместить на Square объект Piece.
  3. Проверить, что Square и Piece правильно связаны.
- **Результат:** Объекты Square и Piece корректно взаимодействуют при размещении фигуры.

Тип теста: Позитивный

I15.2: Тестирование правильности связи объектов Square и Piece при перемещении фигуры.

- **Входные данные:** Два объекта Square - начальная и конечная позиции, объект Piece с корректными данными.

- **Шаги:**

  1. Инициализировать объекты Square и Piece.
  2. Переместить Piece с начальной позиции на конечную.
  3. Проверить, что Square и Piece правильно связаны после перемещения фигуры.

- **Результат:** Объекты Square и Piece корректно взаимодействуют при перемещении фигуры.

3.3 Аттестационное тестирования

3.3.1 Функциональная возможность A1: Инициализация

Тип теста: Позитивный

A1.1: Проверка корректной инициализации объектов при запуске приложения.

- **Входные данные:** Запущенное приложение.

- **Шаги:**

  1. Запустить приложение.
  2. Проверить, что все объекты и переменные инициализированы без ошибок.

- **Результат:** Все объекты и переменные успешно инициализированы при запуске приложения.

Тип теста: Позитивный

A1.2: Убеждение в правильной установке начальных значений переменных и объектов.

- **Входные данные:** Запущенное приложение.

- **Шаги:**

  1. Проверить начальные значения переменных и свойств объектов.
  2. Убедиться, что начальные значения соответствуют ожидаемым.

- **Результат:** Начальные значения переменных и объектов установлены правильно.

3.3.2 Функциональная возможность A2: Перемещение фигур

Тип теста: Позитивный

A2.1: Проверка возможности перемещения фигур по шахматной доске.

- **Входные данные:** Запущенное приложение с начальным распределением фигур на доске.
- **Шаги:**
  1. Выполнить ход фигурой на допустимую позицию.
  2. Проверить, что фигура успешно перемещена.
- **Результат:** Фигуры можно перемещать по шахматной доске.

Тип теста: Позитивный

A2.2: Убеждение в соответствии движения фигур шахматным правилам.

- **Входные данные:** Запущенное приложение с начальным распределением фигур на доске.
- **Шаги:**
  1. Выполнить ход каждой фигурой в соответствии с шахматными правилами.
  2. Проверить, что движение фигур соответствует ожидаемым шахматным правилам.
- **Результат:** Фигуры двигаются в соответствии с шахматными правилами.

3.3.3 Функциональная возможность A3: Правила шахмат

Тип теста: Позитивный

A3.1: Проверка корректности обработки правил шахмат.

- **Входные данные:** Запущенное приложение с определенным распределением фигур.
- **Шаги:**
  1. Выполнить рокировку, взятие на проходе, и превращение пешки в соответствии с шахматными правилами.

2. Проверить, что правила корректно обрабатываются и не приводят к ошибкам.

- **Результат:** Правила шахмат обрабатываются корректно.

Тип теста: Позитивный

А3.2: Убеждение в правильном расчете доступных ходов для каждой фигуры.

- **Входные данные:** Запущенное приложение с определенным распределением фигур.

- **Шаги:**

1. Проверить доступные ходы для каждой фигуры.

2. Сравнить расчетные ходы с ожидаемыми шахматными правилами.

- **Результат:** Для каждой фигуры корректно рассчитаны доступные ходы.

3.3.4 Функциональная возможность А4: Игровой процесс

Тип теста: Позитивный

А4.1: Проверка правильной смены хода между игроками.

- **Входные данные:** Начальное состояние игры с двумя игроками.

- **Шаги:**

1. Выполнить несколько ходов для одного игрока.

2. Проверить, что ход успешно передан другому игроку.

- **Результат:** Смена хода между игроками происходит корректно.

Тип теста: Позитивный

А4.2: Убеждение в корректной обработке завершения игры.

- **Входные данные:** Состояние игры, при котором возможны различные исходы (шах, мат, пат).

- **Шаги:**

1. Сыграть несколько ходов, приводящих к различным исходам игры.

2. Проверить, что игра корректно завершается в соответствии с шахматными правилами.

- **Результат:** Игра завершается правильно при наступлении различных условий (шах, мат, пат).

3.3.5 Функциональная возможность A5: Интерфейс

Тип теста: Позитивный

A5.1: Проверка отображения шахматной доски и фигур.

- **Входные данные:** Запущенное приложение с активной игровой сессией.
- **Шаги:**
  1. Оценить визуальное отображение шахматной доски.
  2. Проверить, что фигуры корректно расположены на доске.
- **Результат:** Шахматная доска и фигуры отображаются правильно.

Тип теста: Позитивный

A5.2: Убеждение в правильном отображении текстур и цветов.

- **Входные данные:** Запущенное приложение с активной игровой сессией.
- **Шаги:**
  1. Изменить тему приложения.
  2. Проверить, что текстуры и цвета корректно соответствуют выбранной теме.
- **Результат:** Текстуры и цвета отображаются согласно выбранной теме.

3.3.6 Функциональная возможность A6: Звуковые эффекты

Тип теста: Позитивный

A6.1: Проверка воспроизведения звуков при ходе фигуры.

- **Входные данные:** Запущенное приложение с активной игровой сессией.
- **Шаги:**
  1. Выполнить ход фигуры.

2. Проверить воспроизведение звука.

- **Результат:** Звук хода фигуры воспроизводится корректно.

Тип теста: Позитивный

A6.2: Проверка воспроизведения звуков при взятии фигуры.

- **Входные данные:** Запущенное приложение с активной игровой сессией.
- **Шаги:**
  1. Выполнить взятие фигуры.
  2. Проверить воспроизведение звука.
- **Результат:** Звук взятия фигуры воспроизводится корректно.

Тип теста: Позитивный

A6.3: Проверка воспроизведения звуков при завершении игры.

- **Входные данные:** Запущенное приложение с завершенной игровой сессией.
- **Шаги:**
  1. Завершить игру (достигнут шах, мат или пат).
  2. Проверить воспроизведение звука завершения игры.
- **Результат:** Звук завершения игры воспроизводится корректно.

3.3.7 Функциональная возможность A7: Изменение настроек

Тип теста: Позитивный

A7.1: Проверка смены темы в приложении.

- **Входные данные:** Запущенное приложение.
- **Шаги:**
  1. Перейти в настройки приложения.
  2. Выбрать новую тему.
  3. Применить изменения.
- **Результат:** Тема приложения успешно изменена.

Тип теста: Позитивный

A7.2: Проверка корректного применения новых настроек.

- **Входные данные:** Запущенное приложение с измененной темой.
- **Шаги:**
  1. Проверить, что изменения темы отображаются на шахматной доске и фигурах.
- **Результат:** Новые настройки успешно применены и отображаются в интерфейсе.

3.4 Нагрузочное тестирования

Стабильность - способность системы поддерживать устойчивое и надежное поведение при различных условиях, включая изменения нагрузки, внешние воздействия и долгосрочное использование.

3.4.1 Оценка пропускной способности L1

Тип теста: Общий

L1.1: Имитация активного взаимодействия с приложением.

- **Входные данные:** Запущенное приложение с активной игровой сессией.
- **Результат:** Оценка времени отклика системы на активные действия пользователя (ходы, изменение темы) при нормальной нагрузке.

Тип теста: Общий

L1.2: Увеличение числа одновременных пользователей.

- **Входные данные:** Постепенное увеличение числа одновременных пользователей, взаимодействующих с приложением.
- **Результат:** Оценка производительности системы и выявление предела стабильной работы приложения под различными нагрузками.

3.4.2 Измерение времени отклика L2

Тип теста: Статический

L2.1: Регистрация времени отклика на запросы.

- **Входные данные:** Запросы различной сложности и интенсивности.
- **Результат:** Регистрация времени отклика системы на каждый запрос, оценка эффективности обработки запросов при нормальной нагрузке.

Тип теста: Статический

L2.2: Изменение времени отклика при увеличении нагрузки.

- **Входные данные:** Постепенное увеличение числа одновременных пользователей.
- **Результат:** Сравнение времени отклика системы на запросы при различных уровнях нагрузки, выявление изменений в производительности.

3.4.3 Тестирование стабильности системы L3

Тип теста: Пиковый

L3.1: Устойчивость системы под долговременной нагрузкой.

- **Входные данные:** Длительное время активного взаимодействия с приложением.
- **Результат:** Мониторинг ресурсов и производительности для выявления потенциальных утечек памяти, нестабильности или других проблем после продолжительной работы системы.

3.4.4 Оценка максимальной нагрузки L4

Тип теста: Пиковый

L4.1: Предел стабильной работы системы.

- **Входные данные:** Увеличение числа одновременных пользователей до максимально возможного значения.
- **Результат:** Определение точки, при которой система начинает проявлять признаки нестабильности или снижения производительности.

3.4.5 Тестирование под высокой нагрузкой L5

Тип теста: Пиковый

L5.1: Максимальная активность пользователей.

- **Входные данные:** Имитация сценариев с максимальной активностью пользователей, включая выполнение различных действий.
- **Результат:** Проверка способности системы справляться с пиковой нагрузкой без существенного снижения производительности.

3.4.6 Анализ использования ресурсов L6

Тип теста: Статический

L6.1: Мониторинг ресурсов в высоконагруженных условиях.

- **Входные данные:** Запуск системы под высокой активностью пользователей.
- **Результат:** Мониторинг использования ресурсов (память, процессор) для выявления участков, где требуется оптимизация.

3.4.7 Тестирование долгосрочной стабильности L7

Тип теста: Общий

L7.1: Оценка стабильности системы в течение продолжительной нагрузки.

- **Входные данные:** Запуск системы под долгосрочной активностью.
- **Результат:** Оценка стабильности системы и выявление возможных проблем после длительной работы.

3.5 Покрытие кода тестами

$$Tcov = \left(\frac{Ltc}{Lcode}\right) \times 100\%$$

где

Tcov - тестовое покрытие

Ltc - количество строк кода покрытое тестами

Lcode - Общее количество строк кода

Тогда: **Tcov = (463/995) * 100% = 46,53%**

Глава 4

Журнал тестирования

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
B1	Объект успешно создан, все поля инициализированы корректно.	Позитивный	№1
B2	Для каждого размера текстура устанавливается без ошибок.	Позитивный	№2
B3	Список ходов успешно добавляется и корректно очищается.	Позитивный	
B4	Объекты каждого типа успешно созданы, и их параметры инициализированы корректно.	Позитивный	
B5	Объект Sound успешно создан, звуковой файл загружен без ошибок.	Позитивный	
B6	Звук успешно воспроизводится, без искажений и задержек.	Позитивный	
B7	Объект Square успешно создан, значения row и col установлены правильно.	Позитивный	

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
B8	Методы корректно возвращают ожидаемые значения, учитывая наличие или отсутствие фигуры.	Позитивный	
B9	Метод возвращает корректные значения в зависимости от входных данных.	Позитивный	
B10	Метод возвращает корректные буквенные обозначения в зависимости от входных данных.	Позитивный	
B11	Объект Theme успешно создан, значения цветов установлены правильно.	Позитивный	
B12	Объект Color успешно создан, значения цветовых компонент установлены правильно.	Позитивный	
B13	Объект Move успешно создан, значения установлены правильно.	Позитивный	
B14	Метод `__str__` возвращает корректное	Позитивный	

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
B15	Метод `__eq__` возвращает `True` для двух объектов с одинаковыми значениями и `False` в противном случае.	Позитивный	
B16	Объект Dragger успешно создан без ошибок.	Позитивный	
B17	Метод `update_blit` выполняется без ошибок, обновляет отображение доски.	Позитивный	
B18	Метод `update_mouse` выполняется без ошибок, обновляет состояние мыши.	Позитивный	
B19	Метод `save_initial` выполняется без ошибок, сохраняет начальное состояние доски.	Позитивный	
B20	Метод `drag_piece` выполняется без ошибок, начинает перемещение фигуры.	Позитивный	
B21	Метод `undrag_piece` выполняется без ошибок, завершает перемещение фигу-	Позитивный	

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
B22	Объект Board успешно создан без ошибок.	Позитивный	
B23	Метод `move` выполняется без ошибок, фигура перемещена на новую позицию.	Позитивный	
B24	Метод `valid_move` возвращает корректное значение (правильность хода).	Позитивный	
B25	Метод `check_promotion` возвращает корректное значение (необходимость превращения пешки).	Позитивный	
B26	Метод `castling` выполняется без ошибок, происходит рокировка.	Позитивный	
B27	Метод `set_true_en_passant` выполняется без ошибок, устанавливается флаг корректно.	Позитивный	
B28	Метод `in_check` возвращает корректное значение (нахождение короля под	Позитивный	

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
B29	Метод `calc_moves` возвращает корректные возможные ходы для фигуры.	Позитивный	
B30	Метод `_create` выполняется без ошибок, создает начальное состояние доски.	Позитивный	
B31	Метод `_add_pieces` выполняется без ошибок, добавляет фигуры на доску.	Позитивный	
B32	Объект Config успешно создан без ошибок.	Позитивный	
B33	Метод `change_theme` выполняется без ошибок, тема приложения изменяется на новую.	Позитивный	
B34	Объект Game успешно создан без ошибок.	Позитивный	
B35	Метод `next_turn` успешно меняет текущего игрока на противоположного.	Позитивный	

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
B36	Метод `set_hover` корректно устанавливает клетку под курсором.	Позитивный	
B37	Метод `change_theme` успешно изменяет тему приложения на новую.	Позитивный	
B38	Метод `play_sound` воспроизводит звук без ошибок.	Позитивный	
B39	Метод `reset` сбрасывает состояние игры к начальному без ошибок.	Позитивный	
B40	Все методы отрисовки работают корректно и не вызывают ошибок.	Позитивный	
I1.1	Фигура успешно размещена на доске, объекты Piece и Square взаимодействуют корректно.	Позитивный	№3
I1.2	Операция размещения фигуры на уже занятой клетке завершается ошибкой, взаимодействие обнаружено.	Негативный	Ошибка обнаружена.

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I2.1	Квадраты успешно размещены на доске, объекты Square и Board взаимодействуют корректно.	Позитивный	№4
I2.2	Операция размещения объектов Square за пределами доски завершается ошибкой, взаимодействие обнаружено.	Негативный	Ошибка обнаружена.
I3.1	Ход успешно обработан и сохранен в объекте Piece, взаимодействие корректно.	Позитивный	
I3.2	Операция обработки некорректного хода завершается ошибкой, взаимодействие обнаружено.	Негативный	Ошибка обнаружена.
I4.1	Фигура успешно перемещена с помощью Dragger в объекте Game, взаимодействие корректно.	Позитивный	
I4.2	Операция обработки некорректного перемещения завершается ошибкой, взаимодействие обнаружено.	Негативный	Ошибка обнаружена.

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I5.2	Операция обработки некорректного хода завершается ошибкой, взаимодействие обнаружено.	Негативный	Ошибка обнаружена.
I6.1	Тема успешно применена к объекту Config, изменения видны.	Позитивный	
I6.2	Происходит обработка ошибки при попытке применить некорректную тему, взаимодействие обнаружено.	Негативный	Ошибка обнаружена.
I7.1	Звук успешно воспроизведен в объекте Game, аудиофайлы проиграны корректно.	Позитивный	
I7.2	Обработка ошибки при отсутствии звукового файла в объекте Sound, взаимодействие обнаружено.	Негативный	Ошибка обнаружена.
I8.1	Тема в объекте Game изменена успешно, интерфейс отображается согласно новой теме.	Позитивный	

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I9.1	Все модули успешно взаимодействуют в течение полного цикла игрового процесса, игра завершается корректно.	Позитивный	
I9.2	Обработка ошибок при некорректной конфигурации модулей, взаимодействие обнаружено и обработано.	Негативный	Ошибка обнаружена.
I10.1	Игровой интерфейс отображается корректно, события обрабатываются без ошибок.	Позитивный	
I10.2	Обработка ошибок при некорректной конфигурации Pygame, взаимодействие обнаружено и обработано.	Негативный	Ошибка обнаружена.
I11.1	Изменения в настройках игры, внесенные через объект Config, отображаются корректно в объекте Game.	Позитивный	
I11.2	Обработка ошибок при попытке изме-	Негативный	Ошибка обнаружена.

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I12.2	Обработка ошибок при попытке воспроизвести звуки без загруженных файлов.	Негативный	Ошибка обнаружена.
I13.1	Ресурсы корректно отображаются и воспроизводятся в соответствии с выбранной темой.	Позитивный	
I13.2	Обработка ошибок при попытке использования отсутствующих ресурсов.	Негативный	Ошибка обнаружена.
I14.1	Игровой ход взаимодействует с объектом Game, и его результат корректно отображается.	Позитивный	
I14.2	Обработка ошибок при попытке выполнения некорректного хода.	Негативный	Ошибка обнаружена.
I15.1	Объекты Square и Piece корректно взаимодействуют при размещении фигуры.	Позитивный	
I15.2	Объекты Square и Piece корректно взаимодействуют при перемещении	Позитивный	

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
A1.1	Все объекты и переменные успешно инициализированы при запуске приложения.	Позитивный	№5
A1.2	Начальные значения переменных и объектов установлены правильно.	Позитивный	
A2.1	Фигуры можно перемещать по шахматной доске.	Позитивный	№6
A2.2	Фигуры двигаются в соответствии с шахматными правилами.	Позитивный	
A3.1	Правила шахмат обрабатываются корректно.	Позитивный	
A3.2	Для каждой фигуры корректно рассчитаны доступные ходы.	Позитивный	
A4.1	Смена хода между игроками происходит корректно.	Позитивный	

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
A4.2	Игра завершается правильно при наступлении различных условий (шах, мат, пат).	Позитивный	
A5.1	Шахматная доска и фигуры отображаются правильно.	Позитивный	
A5.2	Текстуры и цвета отображаются согласно выбранной теме.	Позитивный	
A6.1	Звук хода фигуры воспроизводится корректно.	Позитивный	
A6.2	Звук взятия фигуры воспроизводится корректно.	Позитивный	
A6.3	Звук завершения игры воспроизводится корректно.	Позитивный	
A7.1	Тема приложения успешно изменена.	Позитивный	

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
A7.2	Новые настройки успешно применены и отображаются в интерфейсе.	Позитивный	
L1.1	Оценка времени отклика системы на активные действия пользователя при нормальной нагрузке.	Общий	№7
L1.2	Оценка производительности системы при увеличении числа одновременных пользователей.	Общий	
L2.1	Регистрация времени отклика на запросы при нормальной нагрузке.	Статический	№ 8
L2.2	Изменение времени отклика при увеличении нагрузки.	Статический	
L3.1	Мониторинг ресурсов и производительности для выявления проблем после долговременной нагрузки.	Пиковый	
L4.1	Определение предела стабильной работы системы при увеличении числа одно-	Пиковый	

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
L5.1	Проверка способности системы справляться с пиковой нагрузкой.	Пиковый	
L6.1	Мониторинг использования ресурсов под высокой активностью пользователей.	Статический	
L7.1	Оценка стабильности системы после долгосрочной активности.	Общий	

Глава 5

Примеры тестов

5.1 Тест для I4.1

```
class TestIntegrationDraggerGame(unittest.TestCase):

    def setUp(self):
        pygame.init()
        self.screen = pygame.display.set_mode((WIDTH, HEIGHT))
        pygame.display.set_caption('Chess')
        self.game = Game()

    def tearDown(self):
        pygame.quit()

    def test_piece_movement(self):
        # Создаем фиктивную фигуру для теста
        dummy_piece = self.game.board.squares[0][0].piece
        self.game.board.squares[0][0].piece = None  # Убеждаемся, что начальная клетка пуста
        self.game.board.squares[3][3].piece = dummy_piece  # Помещаем фигуру на другую клетку

        # Имитируем перемещение фигуры с использованием Dragger в объекте Game
        self.game.dragger.drag_piece(dummy_piece)
        self.game.dragger.initial_row = 3
        self.game.dragger.initial_col = 3
        self.game.dragger.update_mouse((400, 400))  # Новые координаты для перемещения фигуры
        self.game.dragger.update_blit(self.screen)
```

Рис. 5.1: Тест для I4.1

5.2 Тест для B14

```python
class TestMoveStrMethod(unittest.TestCase):

    def test_str_method(self):
        # Входные данные: объект Move с определенными значениями
        initial_square = Square(1, 2)
        final_square = Square(3, 4)
        test_move = Move(initial_square, final_square)

        # Шаг: вызвать метод __str__ для объекта Move
        result_str = str(test_move)

        # Результат: метод __str__ возвращает корректное строковое представление объекта Move
        expected_str = "(2, 1) -> (4, 3)"
        self.assertEqual(result_str, expected_str)
```

Рис. 5.2: Тест для B14

5.3 Тест для B33

```python
class TestChangeThemeMethod(unittest.TestCase):

    def test_change_theme_method(self):
        # Входные данные: объект Config и новая тема
        test_config = Config()
        new_theme = Theme((255, 0, 0), (0, 0, 255), (0, 255, 0), (255, 255, 0), '#FFFFFF', '#000000')

        # Шаг: вызвать метод change_theme с заданными параметрами
        test_config.change_theme()

        # Результат: метод change_theme выполняется без ошибок, тема приложения изменяется на новую
        self.assertIsNotNone(test_config.theme)
        self.assertIsInstance(test_config.theme, Theme)
        self.assertNotEqual(test_config.theme, new_theme)  # Проверка, что тема изменилась
```

Рис. 5.3: Тест для B33

5.4 Тест для А6.1

```python
class TestSoundEffects(unittest.TestCase):

    def test_piece_move_sound(self):
        # Входные данные: запущенное приложение с активной игровой сессией
        test_config = Config()

        # Шаги:
        # 1. Выполнить ход фигуры
        # 2. Проверить воспроизведение звука
        try:
            test_config.move_sound.play()
        except pygame.error as e:
            self.fail(f"Ошибка при воспроизведении звука хода фигуры: {e}")
```

Рис. 5.4: Тест для А6.1

5.5 Тест для L2.1

```python
class TestResponseTime(unittest.TestCase):

    def test_response_time(self):
        # Входные данные: Запросы различной сложности и интенсивности

        # Шаги:
        # 1. Отправить запрос
        # 2. Засечь время от начала отправки запроса до получения ответа

        # Результат: Регистрация времени отклика системы на каждый запрос,
        # оценка эффективности обработки запросов при нормальной нагрузке

        # Пример простого запроса
        start_time = time.time()
        # Ваш код обработки запроса
        elapsed_time = time.time() - start_time

        # Проверка, что время отклика на запрос не превышает некоторый лимит
        max_response_time = 0.1  # Укажите лимит в секундах
        self.assertLessEqual(elapsed_time, max_response_time, msg: "Время отклика превысило установленный лимит")
```

Рис. 5.5: Тест для L2.1

Глава 6

Журнал найденных ошибок

6.1 Ошибка №1

- **B1: Проверка корректности инициализации объекта Piece**
  - **Описание:** Неверное значение координаты установлено при инициализации.
  - **Шаги:** Запущен тест B1 с некорректными координатами.
  - **Результат:** Ошибка - координата не соответствует ожидаемому значению.
  - **Текущее состояние:** Исправлено.

6.2 Ошибка №2

- **B2: Тестирование метода `set_texture` для различных размеров**
  - **Описание:** Некорректное масштабирование текстуры.
  - **Шаги:** Запущен тест B2 с текстурой несоответствующего размера.
  - **Результат:** Ошибка - текстура отображается неверно.
  - **Текущее состояние:** Исправлено.

6.3 Ошибка №3

- **I1: Тестирование взаимодействия между `piece.py` и `square.py`**
  - **Описание:** Фигура не отображается корректно на доске.
  - **Шаги:** Запущен тест I1 с размещением фигуры на доске.

- **Результат:** Ошибка - фигура не отображается или отображается некорректно.
- **Текущее состояние:** Исправлено.

6.4 Ошибка №4

- **I2: Тестирование взаимодействия между `square.py` и `board.py`**
  - **Описание:** Фигуры размещаются в неверных позициях на доске.
  - **Шаги:** Запущен тест I2 с размещением фигур на доске.
  - **Результат:** Ошибка - фигуры расположены некорректно.
  - **Текущее состояние:** Исправлено.

6.5 Ошибка №5

- **A1: Тестирование инициализации**
  - **Описание:** Приложение не запускается из-за ошибки в инициализации.
  - **Шаги:** Запущен тест A1 на запуск приложения.
  - **Результат:** Ошибка - приложение не запускается.
  - **Текущее состояние:** Исправлено.

6.6 Ошибка №6

- **A2: Тестирование перемещения фигур**
  - **Описание:** Фигуры двигаются не по шахматным правилам.
  - **Шаги:** Запущен тест A2 на перемещение фигур по доске.
  - **Результат:** Ошибка - некорректное перемещение фигур.
  - **Текущее состояние:** Исправлено.

6.7 Ошибка №7

- **L1: Оценка пропускной способности**

- **Описание:** Время отклика сильно увеличивается при увеличении числа пользователей.
- **Шаги:** Запущен тест L1 с увеличением числа пользователей.
- **Результат:** Ошибка - производительность системы не соответствует требованиям.
- **Текущее состояние:** Исправлено.

6.8 Ошибка №8

- **L2: Измерение времени отклика**
  - **Описание:** Время отклика превышает допустимое значение при максимальной нагрузке.
  - **Шаги:** Запущен тест L2 с максимальной нагрузкой.
  - **Результат:** Ошибка - система не справляется с высокой активностью пользователей.
  - **Текущее состояние:** Не исправлено.

Глава 7

Результаты

В ходе выполнения данной работы была проведена комплексная подготовка и планирование тестирования разрабатываемого приложения. Объект тестирования был детально описан, включая его структуру, модули, методы, и функциональности.

Стратегия тестирования была разработана, охватывая блочное, интеграционное, аттестационное и нагрузочное тестирование. Критерии прохождения тестирования были определены, а также установлены условия возобновления и приостановки тестов.

Был разработан детальный план тестирования с подробными тест-кейсами для каждой функциональности приложения. Это обеспечивает систематическое и полное покрытие всех аспектов приложения в процессе тестирования.

Журнал тестирования включает в себя результаты проведенных тестов, покрытие кода тестами, а также выявленные ошибки. Приведены примеры тестов для иллюстрации процесса тестирования.

В результате тестирования были выявлены и задокументированы ошибки. Журнал ошибок содержит подробные описания каждой ошибки, а также, где возможно, предложения по их устранению.