2.Техники тестирования Black-box

Техники тестирования Black-box (черный ящик) — это методы тестирования программного обеспечения, которые не требуют знания внутренней структуры или кода программы. Вместо этого тестировщики взаимодействуют с системой через её интерфейсы и проверяют её поведение на основе спецификаций и требований. Основная цель тестирования черного ящика — убедиться, что система работает правильно с точки зрения пользователя.

Определить подходящие техники тестирования Black-box для заданной спецификации. Спроектировать необходимые и достаточные тесты и оформление результатов в соответствии с описанным стандартом.

Задание:

1. Применить следующие методы/техники тестирования на основе спецификаций (Black Box):
   - Эквивалентное разбиение
   - Анализ граничных значений
   - Таблица решений
   - Таблица перехода
   - Сценарии использования

   для проектирования тестов по спецификации.
2. Используя изученные методы, спроектировать тесты (необходимые и достаточные) и оформить результаты в соответствии с описанным стандартом.

1. Эквивалентное разбиение (Equivalence Partitioning)

Описание: Эквивалентное разбиение — это техника, при которой входные данные разделяются на группы (классы), которые считаются эквивалентными. Предполагается, что если один элемент класса работает правильно, то и все остальные элементы этого класса также будут работать правильно.

Как сделать: Разделите входные данные на эквивалентные классы и выберите по одному представителю из каждого класса для тестирования. Например, если у вас

есть поле ввода для возраста, разделите его на классы: "возраст меньше 18", "возраст от 18 до 65", "возраст больше 65".

Пример:

1.2. Покупка кофе Cappuccino, проверка баланса, что денег хватает.

Положительные:

- [35; +inf) — покупка возможна

Негативные

- [0, 35) — недостаточно средств
- (-inf, 0) — некорректное значение баланса

2. Анализ граничных значений (Boundary Value Analysis)

Описание: Анализ граничных значений — это техника, при которой проверяются граничные значения эквивалентных классов, так как ошибки часто возникают на границах.

Как сделать: Проверьте минимальные и максимальные значения, а также значения, близкие к этим границам. Например, для поля ввода возраста проверьте значения 17, 18, 65 и 66.

Двухточечный и трехточечный методы — это техники тестирования, которые используются для проверки граничных значений и соседних значений входных данных. Эти методы помогают выявить ошибки, которые часто возникают на границах допустимых значений.

Двухточечный метод (Two-Point Method)

Описание: Двухточечный метод включает тестирование минимального и максимального значений допустимого диапазона.

Как сделать: Проверьте минимальное и максимальное значения допустимого диапазона. Например, если допустимый диапазон возраста от 18 до 65, проверьте значения 18 и 65.

Трехточечный метод (Three-Point Method)

Описание: Трехточечный метод включает тестирование минимального, максимального и среднего значений допустимого диапазона.

Как сделать: Проверьте минимальное, максимальное и среднее значения допустимого диапазона. Например, если допустимый диапазон возраста от 18 до 65, проверьте значения 18, 41 (среднее значение) и 65.

Пример:

Покупка кофе Cappuccino, проверка баланса, что денег хватает.

Области:

- [35; +inf) — покупка возможна
- [0, 35) — недостаточно средств
- (-inf, 0) — некорректное значение баланса

Двухточечный метод:

- 35, 36
- 0, 1 и 34, 35
- -1, 0

Трехточечный метод:

- 34, 35, 36
- -1, 0, 1 и 34, 35, 36
- -2, -1, 0

3. Таблица решений (Decision Table Testing)

Описание: Таблица решений — это техника, при которой используются таблицы для определения всех возможных комбинаций условий и действий, что помогает систематически проверить все возможные сценарии.

Как сделать: Создайте таблицу, где строки представляют собой условия, а столбцы — действия. Заполните таблицу всеми возможными комбинациями и проверьте каждую из них. Например, таблица решений для определения скидки на основе возраста и типа клиента.

Пример:

Обнуление информации

	Правило 1	Правило 2
Условие 1. После выдачи напитка прошло меньше 10 с.	N	Y

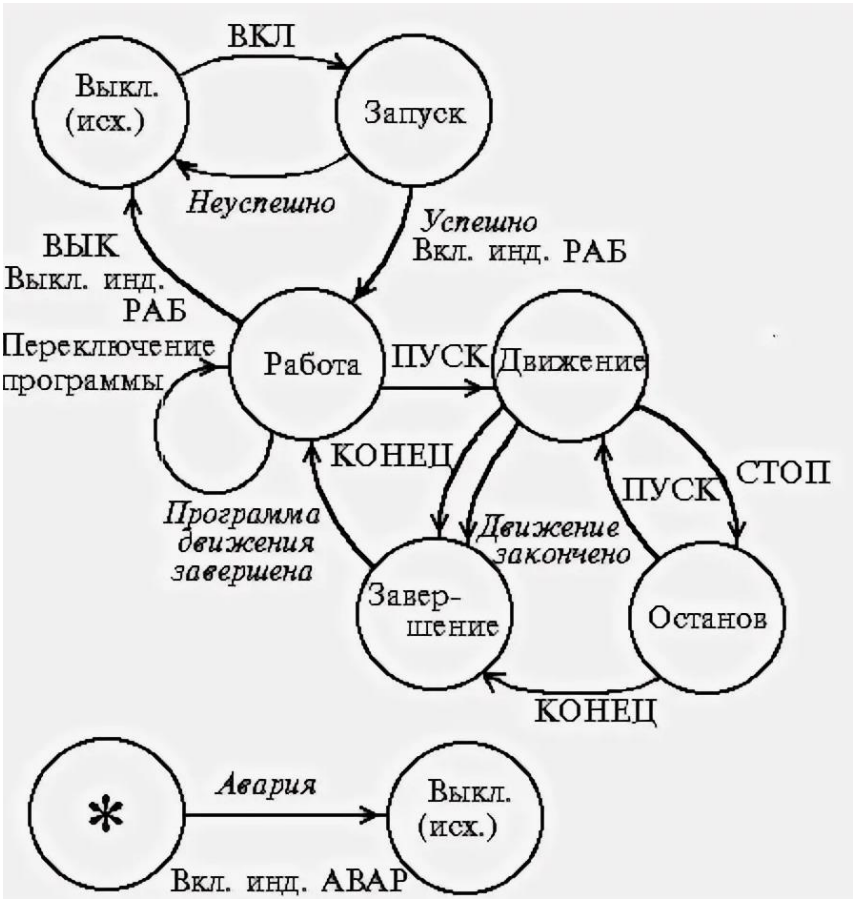
Условие 2. После выдачи напитка прошло 10 с или более.	Y	N
Действие 1. Отображение текущего баланса		X
Действие 2. Обнуление интерфейса	X	

4. Таблица перехода (State Transition Testing)

Описание: Таблица перехода — это техника, при которой проверяется поведение системы при переходах между различными состояниями, что особенно полезно для систем, которые имеют множество состояний.

Как сделать: Создайте таблицу переходов, где строки представляют собой текущие состояния, а столбцы — события, которые могут изменить состояние. Проверьте все возможные переходы. Например, таблица переходов для тестирования работы банкомата.

Пример графа состояний и событий:



Пример таблицы переходов:

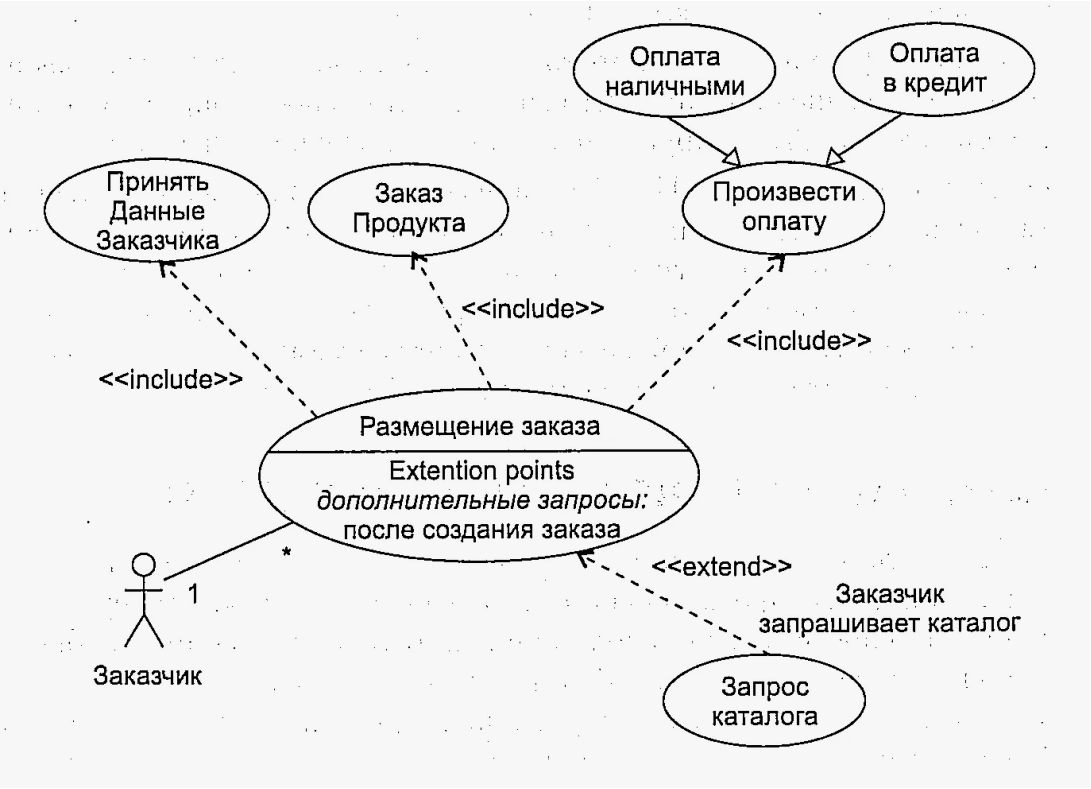
Текущее состояние	Запуск (Start)	Остановка (Stop)	Завершение работы (Shutdown)	Авария (Error Occurred)	Выключение (Turn Off)	Перезапуск (Restart)
Выключен (Off)	Запуск (Starting)	-	-	-	-	-
Запуск (Starting)	-	Остановлен (Stopped)	Завершение работы (Shutting Down)	Авария (Error)	Выключен (Off)	-
Работает (Running)	-	Остановлен (Stopped)	Завершение работы (Shutting Down)	Авария (Error)	Выключен (Off)	-
Остановлен (Stopped)	Запуск (Starting)	-	Завершение работы (Shutting Down)	-	Выключен (Off)	-
Авария (Error)	-	-	Завершение работы (Shutting Down)	-	Выключен (Off)	Запуск (Starting)
Завершение работы (Shutting Down)	-	-	-	-	Выключен (Off)	-

5. Сценарии использования (Use Case Testing)

Описание: Сценарии использования — это техника, при которой система проверяется на основе сценариев использования, которые описывают, как пользователи будут взаимодействовать с системой.

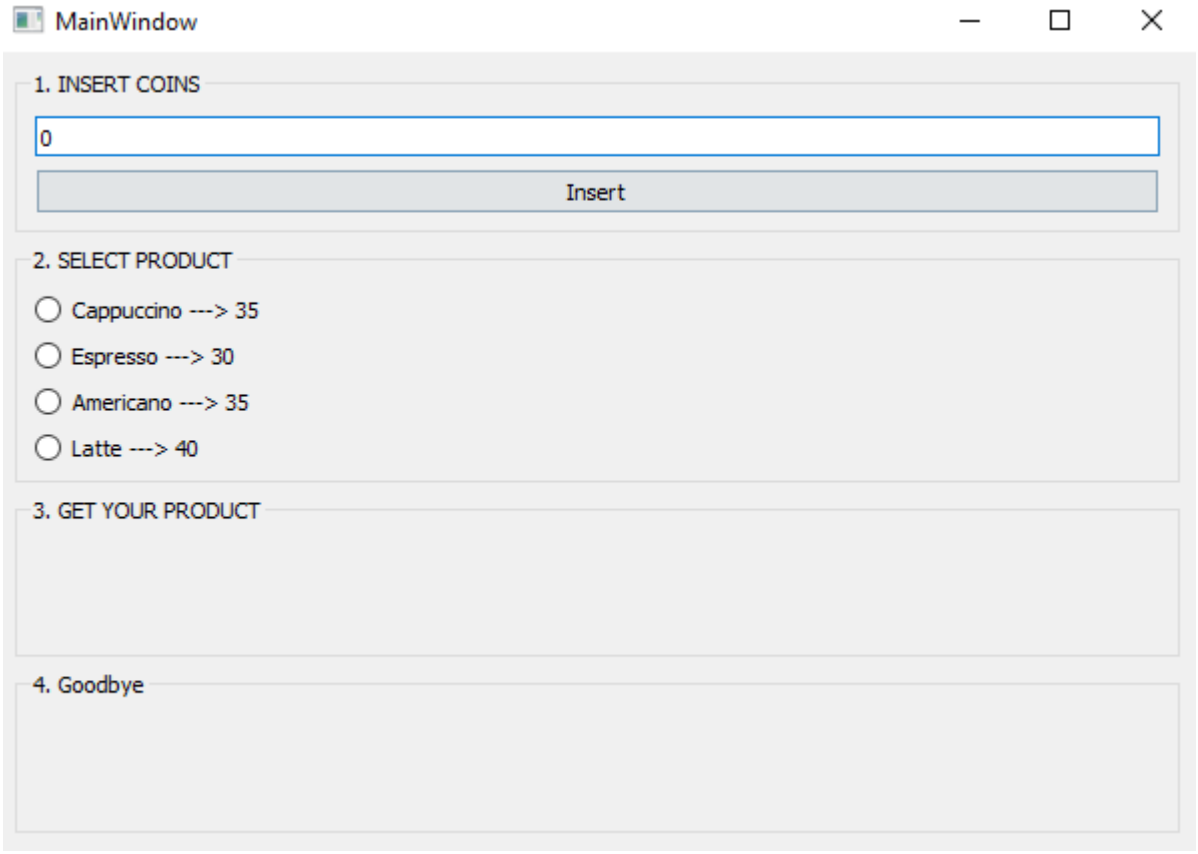
Как сделать: Определите основные сценарии использования и проверьте их на всех этапах. Например, тестирование процесса регистрации нового пользователя на веб-сайте, включая ввод данных, подтверждение email и вход в систему.

Пример:



Исходный вариант спецификации:

- Программа по продаже кофе имеет интерфейс на английском (утвержденный прототип на рис.)



- Поле «Insert coins» соответствует сумме денег, внесённой в автомат. Для внесения средств требуется ввести значение монеты или купюры РФ от 1 до 500 и нажать на кнопку «Insert». Для дополнительного внесения денег следует повторно ввести в указанное поле значение и нажать на кнопку «Insert». Вводить можно только реально существующие в РФ значения монет и купюр.
- После нажатия «Insert» в поле «Get your product» должна отобразится внесённая сумма.
- После выдачи напитка через 10 секунд интерфейс программы должен обнулиться и выглядеть снова как на рисунке.
- После внесения денег пользователь должен выбрать напиток: «Cappuccino», «Espresso», «Americano» или «Latte», каждый напиток имеет свою цену.
    - Если внесена достаточная сумма средств, то в поле «Goodbye» появится пожелание хорошего дня
    - В поле «Get your product» - сообщение, содержащее следующую информацию:
        - Внесённая сумма
        - Сдача

3.Разработка тестовых сценариев

Тестовые сценарии (test cases) — это набор шагов, условий и ожидаемых результатов, которые используются для проверки функциональности программного обеспечения. Существуют различные стандарты и методы для создания тестовых сценариев, которые помогают обеспечить систематический и эффективный процесс тестирования. Вот краткое описание некоторых из них:

Стандарты тестовых сценариев:

IEEE 829 (Standard for Software Test Documentation):

Описывает формат и содержание документов, используемых в процессе тестирования программного обеспечения. Включает описание тестовых сценариев, тестовых планов, отчетов о тестировании и других документов.

ISTQB (International Software Testing Qualifications Board):

Определяет стандарты и лучшие практики для тестирования программного обеспечения. Включает рекомендации по созданию тестовых сценариев, управлению тестированием и другим аспектам тестирования.

ISO/IEC/IEEE 29119 (Software Testing):

Описывает процессы, методы и документы, используемые в тестировании программного обеспечения. Включает стандарты для тестовых сценариев, тестовых планов и отчетов о тестировании.

Задание:

Требуется разработать тестовые сценарии в количестве 5 (test-case) согласно принятым стандартам на основании выбранных методов.

Структура тестового сценария по стандарту IEEE 829

- Идентификатор тестового сценария
- Название тестового сценария
- Предусловия
- Входные значения теста
- Ожидаемый результат

- Постусловия
- Позитивные сценарии
- Негативные сценарии

Пример:

Тестовый сценарий: Покупка Capuccino

1. Идентификатор тестового сценария

Идентификатор: TC-001

Название: Покупка Capuccino

2. Предусловия

Предусловие 1: Программа запущена.

Предусловие 2: Пользователь внес сумму денег.

3. Входные значения теста

Шаг 1: Нажать на кнопку «Capuccino».

4. Ожидаемый результат

Ожидаемый результат 1: Если внесена достаточная сумма средств, то в поле «Goodbye» появится изображение выбранного напитка Capuccino, а в поле «Get your product» - сообщение, содержащее следующую информацию: (внесённая сумма из предусловия, сдача, пожелание хорошего дня).

Ожидаемый результат 2: Через 10 секунд интерфейс программы должен обнулиться и выглядеть снова как на рисунке 1.

5. Постусловия

Постусловие 1: Интерфейс программы должен обнулиться через 10 секунд.

6. Позитивные сценарии

Тестовый сценарий 1:

Идентификатор: TC-001-P1

Название: Покупка Capuccino с балансом 35р

Входные значения: Баланс 35р, нажата кнопка «Capuccino»

Ожидаемый результат: В поле «Goodbye» появится изображение Capuccino, в поле «Get your product» - сообщение с информацией о внесённой сумме, сдаче и пожеланием хорошего дня.

Тестовый сценарий 2:

Идентификатор: TC-001-P2

Название: Покупка Capuccino с балансом 36р

Входные значения: Баланс 36р, нажата кнопка «Capuccino»

Ожидаемый результат: В поле «Goodbye» появится изображение Capuccino, в поле «Get your product» - сообщение с информацией о внесённой сумме, сдаче и пожеланием хорошего дня.

7. Негативные сценарии

Тестовый сценарий 3:

Идентификатор: TC-001-N1

Название: Покупка Capuccino с балансом 34р

Входные значения: Баланс 34р, нажата кнопка «Capuccino»

Ожидаемый результат: В поле «Get your product» появится сообщение "Not enough coins!".

Тестовый сценарий 4:

Идентификатор: TC-001-N2

Название: Покупка Capuccino с балансом 0р

Входные значения: Баланс 0р, нажата кнопка «Capuccino»

Ожидаемый результат: В поле «Get your product» появится сообщение "Not enough coins!".

Тестовый сценарий 5:

Идентификатор: TC-001-N3

Название: Покупка Capuccino с балансом 1р

Входные значения: Баланс 1р, нажата кнопка «Capuccino»

Ожидаемый результат: В поле «Get your product» появится сообщение "Not enough coins!".

3. Статическое тестирование. Рецензирование

Статическое тестирование - это метод тестирования программного обеспечения, который включает в себя проверку кода, требований и документации без выполнения кода. Это один из этапов в процессе тестирования, который помогает выявлять дефекты на ранних стадиях разработки.

Основные характеристики статического тестирования:

1. **Анализ кода**:
    - Проверка кода на соответствие стандартам и правилам кодирования.
    - Использование статических анализаторов кода для выявления потенциальных ошибок и уязвимостей.
2. **Анализ документации**:
    - Проверка требований на полноту и непротиворечивость.
    - Анализ проектной документации и спецификаций.
3. **Методы статического тестирования**:
    - **Ревью кода (Code Review)**: Процесс, при котором другие разработчики проверяют код на наличие ошибок и дают свои рекомендации.
    - **Статический анализ кода**: Использование инструментов для автоматической проверки кода.
    - **Рецензирование (Inspection/Walkthrough)**: Формальный процесс, включающий в себя анализ различных артефактов разработки (требования, дизайн, код и т.д.) группой специалистов.

Рецензирование (Inspection) - это систематический и формальный процесс статического тестирования, при котором артефакты разработки проверяются на соответствие установленным требованиям и стандартам. Цель рецензирования - обнаружение и устранение дефектов на ранних стадиях разработки.

Основные этапы рецензирования:

4. **Планирование**:
    - Определение целей рецензирования.
    - Назначение участников (автор, рецензенты, модератор, секретарь).
5. **Подготовка**:
    - Распределение материалов для рецензирования среди участников.
    - Изучение материалов рецензентами.

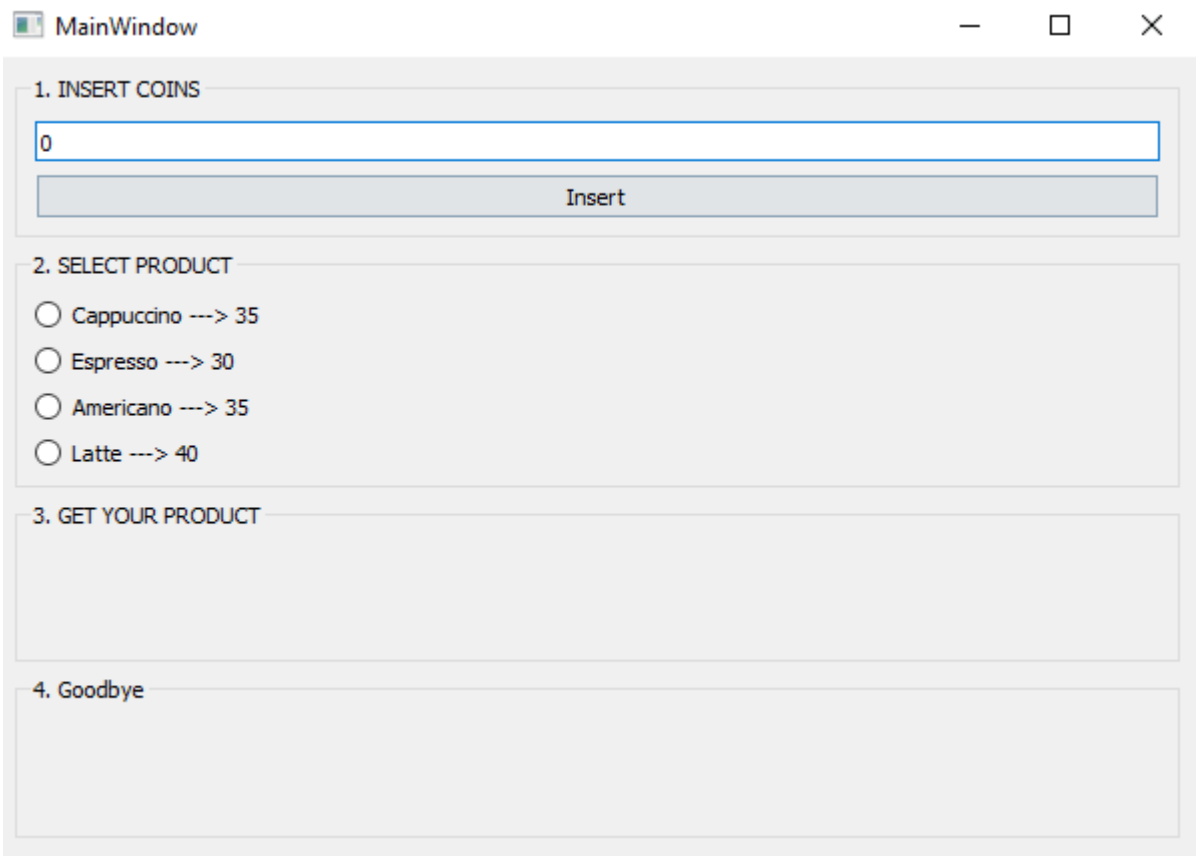
6. **Проведение рецензирования**:
    - Формальная встреча, где участники обсуждают найденные дефекты и предложения по улучшению.
    - Модератор контролирует ход встречи и фиксирует все замечания.
7. **Анализ**:
    - Оценка замечаний и их классификация (критические, серьезные, мелкие).
    - Решение, какие изменения необходимо внести.
8. **Исправление**:
    - Внесение изменений в артефакты разработки на основе замечаний.
9. **Проверка исправлений**:
    - Проверка внесенных изменений для подтверждения исправления дефектов.

В данном практическом задании вам необходимо провести статическое тестирование (рецензирование) функциональной спецификации на прошивку аппарата для продажи кофе. Выявить дефекты спецификации, связанные с недостаточным описанием функционала; потенциальные дефекты, которые могут возникнуть в разработанном ПО по этой спецификации, а также недостатки, связанные с удобством пользователя при работе с программным обеспечением.
Задание:

1. Произвести анализ спецификации и написать 8-10 дефектов и недостатков (минимум 3 должны быть дефекты «мажорного» уровня, остальные «минорного»). А так же дописать варианты решения для примера номер 3(«Отсутствие описания решения проблем с недостатком средств» описание вариантов решения засчитывается как дефект мажорного уровня).
2. Произвести Ревью кода (Приложение А) приложения и дать свои рекомендации по исправлению 5 предложений и исправлений.

Имеется следующая информация о приложении:

- Программа по продаже кофе имеет интерфейс на английском (утвержденный прототип на рис.)



- Поле «Insert coins» соответствует сумме денег, внесённой в автомат. Для внесения средств требуется ввести значение монеты или купюры РФ от 1 до 500 и нажать на кнопку «Insert». Для дополнительного внесения денег следует повторно ввести в указанное поле значение и нажать на кнопку «Insert». Вводить можно только реально существующие в РФ значения монет и купюр.
- После нажатия «Insert» в поле «Get your product» должна отобразится внесённая сумма.
- После выдачи напитка через 10 секунд интерфейс программы должен обнулиться и выглядеть снова как на рисунке.
- После внесения денег пользователь должен выбрать напиток: «Cappuccino», «Espresso», «Americano» или «Latte», каждый напиток имеет свою цену.
    - Если внесена достаточная сумма средств, то в поле «Goodbye» появится пожелание хорошего дня
    - В поле «Get your product» - сообщение, содержащее следующую информацию:

a. Внесённая сумма
b. Сдача

Пример описание дефектов и недостатков, обнаруженных в результате анализа спецификации:

В результате анализа спецификации были выявлены следующие дефекты и недостатки:

1. Отсутствие подтверждения выбора кофе по нажатию кнопки с кофе. В случае, если пользователь случайно нажмет на кнопку с кофе, имея достаточный баланс, произойдет покупка.

Варианты решения:

- ✓ Фрейм с подтверждением покупки. Форма с текстом «Вы действительно хотите купить НАЗВАНИЕ_КОФЕ?» и выбором по нажатию «Да» и «Нет».

Severity: S4 Minor. Проблема пользовательского интерфейса.

2. Нет единиц измерения стоимости кофе (рубли). Интерфейс программы английский, есть вероятность, что пользоваться будут иностранцы пользователями, не знающими, какую валюту принимает автомат.

Варианты решения:

- ✓ Добавить единицы измерения

Severity: S5 Minor. Незначительна проблема пользовательского интерфейса.

3. Отсутствие описание решения проблемы с недостатком средств у пользователя при выборе кофе со стоимостью большей, чем баланс.

Варианты решения:

….

Severity: S3 Major. Не проработана бизнес логика приложения.

Пример ревью кода:

1. Использование констант:

   Используйте константы для цен продуктов, чтобы избежать "магических чисел" в коде.

2. Улучшение пользовательского интерфейса:

Добавьте кнопку для выбора продукта, чтобы пользователь мог явно выбрать продукт, а не использовать клавишу Enter.