Краткое описание основной концепции разрабатываемого проекта

Основная концепция игры “Казино” заключается в том, что игроку предлагается сделать ставку и выбрать число от 1 до 10, и затем программа случайным образом определяет выигрышное число. Если число игрока совпадает с выигрышным числом, то игрок получает 10-кратную ставку, иначе игрок теряет сумму ставки. Это простая и быстрая игра, которая основана на удаче.

Список требований в формате product backlog

1.Создание основного интерфейса:

Разработка меню.

Правила игры.

2.Механика игры:

Реализация загадывания случайного числа.

3.Интерактивность:

Возможность ввода имени игрока и его ставки.

4.Управление игрой:

Реализация выбора режима игры.

Реализация выбора для начала новой игры.

5.Завершение игры:

Определение условий завершения игры (недостаточно денег для игры, добровольный выход из игры).

Вывод сообщения о результате игры.

6.Тестирование:

Разработка тестов для проверки корректности работы игры.

7.Документация:

Написание документации.

Список пользовательских историй из product backlog

1.Как игрок, я хочу видеть меню, чтобы иметь возможность начать игру в одиночном, либо в двойном режиме.

Как пользователь, я должен видеть меню, чтобы иметь возможность начать игру в одиночном либо в двойном режиме.

2.Как игрок, я хочу вводить своё имя и начальный баланс.

Как пользователь, я должен вводить своё имя и баланс, а также видеть его.

3.Как игрок, я хочу вводить числа, чтобы угадывать их и выигрывать ставки.

Как пользователь, я должен иметь возможность вводить числа, чтобы угадывать их и выигрывать ставки.

4.Как игрок, я хочу завершить игру, угадав число.

Как пользователь, я должен иметь возможность начать новую игру.

Разработка первой версии игры “Казино”

Задача: Создание основного интерфейса

Подзадача: Разработка главного меню.

Задача: Механика игры

Подзадача: Реализация загадывания случайного числа.

Задача: Интерактивность

Подзадача: Возможность ввода имени игрока и его ставки.

Задача: Управление игрой

Подзадача: Реализация выбора режима игры.

Подзадача: Реализация выбора для начала новой игры.

Задача: Завершение игры

Подзадача: Определение условий завершения игры (недостаточно денег для игры, добровольный выход из игры).

Подзадача: Вывод сообщения о результате игры.

Задача: Тестирование

Подзадача: Разработка тестов для проверки корректности работы игры.

Задача: Документация

Подзадача: Написание документации.

Вопросы

1. Что такое жизненный цикл разработки программного обеспечения?

Жизненный цикл разработки программного обеспечения – это период времени, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент полного его изъятия из эксплуатации.

2. Какие модели жизненного цикла разработки ПО вы знаете?

Каскадная модель.

Инкрементная модель.

Спиральная модель.

Модель V-образа.

Модель гибкой разработки.

3. Какими способами можно организовать общение с заказчиком?

Общение с заказчиком можно организовать через встречи, видеоконференции, электронную почту, обмен документами, создание прототипов, использование систем отслеживания ошибок и т.д.

4. С какой целью выполняется планирование разработки ПО, составление технического задания?

Планирование разработки ПО и составление технического задания проводится для определения объема работ, сроков выполнения, распределения ресурсов и создания основы для последующего процесса разработки.

5. Поясните, как выполняется детальный анализ предметной области и принимается окончательное решения о необходимости создания ПО.

Детальный анализ предметной области включает в себя изучение требований заказчика, выявление проблем и определение, как программное решение может их решить. Окончательное решение о создании ПО принимается на основе оценки выгод и затрат.

6. Перечислите технологии быстрой разработки программного обеспечения

Технологии быстрой разработки включают в себя RAD (Rapid Application Development), Agile (гибкие методологии разработки, такие как Scrum и Kanban), XP (Extreme Programming) и другие.

7. Что такое технология экстремального программирования?

Технология экстремального программирования (XP) — это методология разработки программного обеспечения, акцентирующая внимание на быстром реагировании на изменения, тесном взаимодействии с заказчиком и тестировании.

8. Что такое SCRUM-технология?

Scrum — это гибкая методология разработки ПО, основанная на итеративном и инкрементальном подходах. Основные элементы включают в себя итерации (спринты), роли (Product Owner, Scrum Master, Team), и постоянную обратную связь.

9. Что такое Kanban-технология?

Kanban — это метод управления разработкой, при котором акцент делается на визуализации процесса работы, ограничении рабочего запаса и непрерывной оптимизации производства.

10.Какие преимущества и недостатки технологий быстрой разработки программного обеспечения?

Преимущества: быстрые результаты, легкость внесения изменений. Недостатки: не всегда подходит для крупных проектов, требует высокой вовлеченности команды.

11.Как организована коллективная работа над проектом при использовании технологий быстрой разработки?

Преимущества: быстрые результаты, легкость внесения изменений. Недостатки: не всегда подходит для крупных проектов, требует высокой вовлеченности команды.

12.Назначение диаграммы вариантов использования.

Диаграмма вариантов использования описывает взаимодействие пользователя с системой, позволяя понять, как система будет использоваться в реальных сценариях.

13.Назначение этапа внедрение и сопровождение программных продуктов.

Этап внедрения включает в себя развертывание программного продукта на живую среду. Сопровождение включает в себя обеспечение корректной работы продукта, устранение ошибок и обновление.

14.Назовите основные задачи, решаемые на этапе внедрения.

Задачи включают в себя установку программы на серверы, обучение пользователей, конфигурирование системы под нужды заказчика.

15.Как выполняется процесс устранения ошибок на этапе внедрения.

Обнаружение ошибок:

Ошибки могут быть обнаружены различными способами, такими как тестирование продукта, обратная связь от конечных пользователей, мониторинг работы системы в реальной среде.

Логирование ошибок:

Выявленные ошибки фиксируются в виде логов или отчетов, содержащих информацию о характере ошибки, условиях ее возникновения и других сведениях.

Приоритизация ошибок:

Ошибки классифицируются по их важности и воздействию на работу системы. Определяется, какие из них требуют немедленного внимания, а какие могут быть отложены.

Исправление ошибок:

Разработчики приступают к исправлению обнаруженных ошибок. Это включает в себя изменение исходного кода, проведение тестирования исправленной версии и подготовку к выпуску обновления.

Тестирование исправлений:

После внесения изменений продукт проходит повторное тестирование, чтобы убедиться, что исправления успешно устранили обнаруженные ошибки и не вызвали новых проблем.

Релиз обновления:

Исправленная версия программы выпускается в виде обновления. Это может включать в себя предоставление пользователям новой версии программного обеспечения или внесение изменений в онлайн-сервис.

Обратная связь и мониторинг:

После релиза обновления важно следить за обратной связью от пользователей и мониторить работу системы в реальной среде. Это помогает выявить возможные новые проблемы или подтвердить успешность устранения ошибок.

Документирование:

Все шаги по устранению ошибок и внесению изменений должны быть документированы. Это помогает команде и будущим разработчикам лучше понимать историю изменений в продукте.

16.Какие решения могут приниматься при выявлении ошибок на этапе внедрения.

Исправление ошибок:

Разработчики могут провести анализ и внести необходимые изменения в исходный код для устранения обнаруженных ошибок.

Выпуск патчей или обновлений:

Если ошибки требуют немедленного внимания, команда может решить выпустить патчи или обновления для пользователей, чтобы исправить проблемы.

Переработка функциональности:

В случае, если обнаруживается серьезная проблема в определенной функциональности, разработчики могут принять решение о переработке этой части продукта.

Откат к предыдущей версии:

Если новая версия программы вызывает слишком много проблем, команда может решить временно откатиться к предыдущей стабильной версии.

Внесение изменений в документацию:

Если ошибка связана с документацией, требуется ее исправление. Это может включать в себя обновление руководств пользователя, инструкций по установке и т.д.

Уточнение требований:

Иногда обнаружившиеся ошибки могут быть следствием неполного или недостаточно четкого понимания требований. В этом случае, команда может обратиться к заказчику для уточнения требований и предоставления ясных указаний.

Улучшение процесса тестирования:

При частых обнаружениях ошибок команда может принять решение об улучшении процесса тестирования, добавлении тестовых сценариев, автоматизации тестирования и т.д.

Обучение пользователей:

Если ошибка связана с неправильным использованием продукта пользователями, может потребоваться обучение или предоставление дополнительной документации.

17.Назначение документирования программного обеспечения.

Понимание и улучшение кода:

Документация помогает разработчикам понимать структуру, логику и особенности кода. Это особенно важно при работе с чужим кодом или при возвращении к собственному коду после длительного времени.

Облегчение сопровождения:

Документация упрощает задачу сопровождения и обновления программного обеспечения. Сведения о том, какие изменения были внесены и почему, помогают новым разработчикам и текущей команде поддержки быстро ориентироваться в коде.

Управление проектом:

Документация играет важную роль в управлении проектом. Она включает в себя планы, расписания, отчеты о статусе, анализ требований, архитектурные документы и др. Эти документы обеспечивают прозрачность в процессе разработки и помогают принимать информированные решения.

Согласование с требованиями:

Документация описывает требования к программному продукту, включая функциональные и нефункциональные характеристики. Это позволяет разработчикам и заказчикам согласовывать ожидания и обеспечивать соответствие результатов разработки требованиям.

Тестирование и отладка:

Документация о том, как должен работать программный продукт, служит основой для создания тестовых сценариев. Тестировщики могут использовать документацию для проверки соответствия продукта требованиям и выявления ошибок.

Обучение пользователей:

Документация, включая руководства пользователя, помогает конечным пользователям понять, как использовать программный продукт. Это особенно важно для сложных или многозадачных систем.

Безопасность и конфиденциальность:

В документации могут быть указаны меры безопасности и требования по обеспечению конфиденциальности данных. Это важно для того, чтобы предотвратить возможные угрозы безопасности.

Создание базы знаний:

Документация становится основой для создания базы знаний в организации. Это улучшает передачу знаний между членами команды и обеспечивает непрерывность в случае изменения состава команды.