МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«Дальневосточный федеральный университет»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ
Департамент программной инженерии и искусственного интеллекта

РАЗРАБОТКА СИСТЕМЫ ПО УСТРАНЕНИЮ ДЕФЕКТОВ НА ИЗОБРАЖЕНИИ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ КОЛЛЕКТИВНОЙ ПРОМЫШЛЕННОЙ РАЗРАБОТКИ

КУРСОВОЙ ПРОЕКТ

по дисциплине «Технологии коллективной промышленной разработки информационных систем» по образовательной программе подготовки бакалавров по направлению 09.03.04 «Программная инженерия»

Выполнили:
студенты гр. Б9119-09.03.04прогин
_____Кузнецов Е.А.
_____Пак К.С.
_____Селютин А.В.
_____Черепанова И.А.
_____Югалдина Ю.К.
Руководитель:
Ассистент ДПИиИИ
_____Логачев Е.М.

г. Владивосток
2024

Содержание

Введение

Промышленная разработка информационных систем включает в себя множество этапов, начиная от создания плана проекта и заканчивая его тестированием. Для успешного выполнения этих этапов необходимо привлечение специалистов различных профилей, которые будут отвечать за различные аспекты разработки. Однако, чтобы эта работа проходила эффективно, необходимы унифицированные методы коммуникации между членами команды, которые позволят им четко делить обязанности и координировать свои действия в соответствии с их специализацией.

Для достижения этой цели рекомендуется использовать технологии коллективной разработки. Это подразумевает применение инструментов, которые позволяют командам эффективно сотрудничать и совместно работать над проектом.

В данном курсовом проекте рассматривается задача коллективной разработки программного средства «Программная система для устранения дефектов на изображении» и составление технической документации к нему.

Таким образом, целью курсового проекта является разработка программного средства с использованием подходов коллективной промышленной разработки.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработать план проекта.
2. Разработать регламент проведения инспекции.
3. Разработать модель состояний задач.
4. Разработать презентацию проекта.
5. Разработать требования к проекту.
6. Разработать архитектуру проекта.
7. Разработать измерения проекта.
8. Разработать перечень задач проекта.
9. Разработать рекомендации по кодированию.

10. Разработать план тестирования проекта.
11. Протестировать проект.

1 План проекта

План проекта – это завершающий документ планирования, содержащий подробную информацию о проекте, включая участников, задачи и сроки. Он утверждается перед началом работы и является основным источником информации о проекте.

В нашем случае исполнителями являются следующие лица:

Team Leader – Пак Ксения;

Coder 1 – Кузнецов Евгений;

Coder 2 – Селютин Алексей;

Coder 3 – Черепанова Ирина;

Build Engineer – Югалдина Юлия;

Technical Writer – Пак Ксения.

На рисунке ниже представлен перечень задач для выполнения и примерные сроки их реализации.

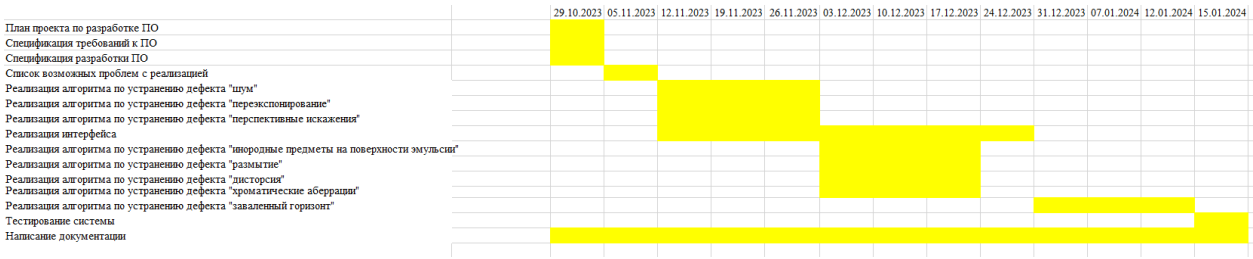

Рисунок 1 - План проекта

2 Регламент проведения инспекции

Проверка рабочих продуктов является неотъемлемой частью процесса обеспечения их качества. В современной программной индустрии разработаны специальные стандарты, методы и инструменты для проведения проверок рабочих продуктов, которые называются инспекциями.

Инспекция — это мероприятие по обеспечению качества рабочих продуктов проектов по разработке ПО и иной деятельности, которая проводится разработчиками, возможно - с участием представителей заказчика. Концептуально инспекция имеет следующие цели:

– Обнаружить ошибки в функциях, логике, содержании или реализации рабочих продуктов на ранних этапах их разработки и предотвратить их наследование;

– Выразить концепцию или воплотить продукт с участием всех заинтересованных лиц в наиболее эффективной форме;

– Оптимизировать, оценить или улучшить рабочий продукт.

Критерии формальности инспекции

Неформальная инспекция проводится:

– В случае изменения участка документации, содержащего не более 6 строк;

– В случае изменения не более 5 элементов для документов дизайна;

– В случае каких-либо правок фрагмента кода (не более 3 строк кода);

– В случае изменения не более 2 тестовых ситуаций.

Формальная инспекция проводиться в случае невозможности проведения неформальной инспекции.

Участники инспекции

Участники могут иметь следующие роли:

1. *Автор* — участник, который вносит изменения в рабочий продукт. Загружает изменения в систему контроля версий и инициирует инспекцию.

2. *Председатель* — человек, контролирующий процесс работы других участников и инспекций. Назначает инспектора и одобряет внесение ключевых изменений в основную версию продукта.

3. *Инспектор* — участник, проводящий проверку внесённых изменений. Оставляет замечания и выносит вердикт о внесении изменений в продукт.

В инспекции в обязательном порядке присутствуют два участника, имеющие роли автора и инспектора соответственно. При внесении изменений в дизайн рабочего продукта, необходимо присутствие председателя, который может также выполнять роль инспектора.

Этапы инспекции

1. Инициация — процесс создания автором запроса на внесение изменений в продукт, корректировка каких-либо мелких недочетов и назначение инспектора.

2. Планирование – назначение даты и списка обязательно присутствующих участников инспекции.

3. Подготовка и проведение — анализ изменений и внесение замечаний инспектором.

4. Завершение — вынесение вердикта о внесении изменений в продукт.

Порядок организации инспекции

Проектный продукт подвергается работе в системе контроля версий GIT. При внесении изменений автор заполняет запрос на изменения, указывает

инспектора и отправляет сообщение в рабочий чат в Telegram с упоминанием инспектора.

Когда инспектор сталкивается с изменением в проекте, он обращается к председателю инспекции в рабочем чате в Telegram. По завершении своей работы инспектор либо одобряет запрос на изменения, либо отправляет его на доработку автору. При этом инспектор уведомляет автора в рабочем чате в Telegram.

После доработки создается снова инспекция, если всё сделано верно и удовлетворяет инспектора и председателя, то изменения засчитываются.

Порядок подготовки к инспекции

Инспекция должна быть проведена в течение 7 дней с момента её инициации.

До момента инспекции должны быть подготовлены все вопросы и уточнения от автора запроса по внесению изменений.

Порядок проведения инспекции

После просмотра и анализа изменений инспектор оставляет в системе контроля замечания, обозначая степень их важности. При наличии замечаний, требующих исправлений, работа передаётся автору на доработку и требования показать исправления. При отсутствии подобных замечаний инспекция считается завершённой, и изменения вступают в силу.

Перечень статусов и степени важности замечаний

1. Комментарий – рекомендация по улучшению продукта, не требующая обязательных изменений.

2. Ошибка – замечание, сообщающее о необходимости исправления.

3. Отклонение – запрос на внесение изменений не является актуальной или в нем нет необходимости.

4. Одобрение – запрос на внесение изменений одобрено и вступает в силу.

Порядок верификации учёта замечаний

После внесения повторных изменений инспектор просматривает замечания и проверяет соответствующие изменения. По окончании верификации выносится вердикт о внесении изменений в продукт или выносятся новые замечания.

Метрики, характеризующие эффективность инспекций

– Производительность инспектирования (ПИ): ПИ = Размер продукта / Общее время инспектирования;

– 80% инспекций привели к улучшению ПО;

– У автора не остается не закрытых вопросов и запросы на внесение изменений были актуальны и одобрены.

3 Модель состояний задач

Каждая задача проходит через несколько состояний в рамках делового процесса. В начале, задача создается, где определяются все ее параметры и критерии успешного выполнения. Затем, переходим к этапу выполнения работ, где назначаются исполнители и ресурсы для выполнения задачи. На этом этапе, происходит непосредственное выполнение задачи согласно определенным требованиям. После завершения работ по задаче, наступает этап завершения, где проверяется выполнение всех заданных критериев и происходит оценка результата работы. Если все выполнено успешно, задача считается завершенной.

Перечень состояний задач:

1. New subtask – новая подзадача.
2. Subtask analysis – подзадача находится в процессе анализирования, то есть, как только участник команды начнет анализ подзадачи, подзадача перейдет в статус анализирования.
3. Development – это состояние задачи подразумевает переход в стадию «Дальнейшая разработка». Подзадача может перейти в это состояние только после анализирования и назначения определенного участника команды, как ответственного.
4. Coding – кодирование. В это состояние подзадача переводится сотрудником/разработчиком, при начале работы по кодированию, связанному с задачей.
5. Inspected – проинспектировано. После того, как сотрудник-разработчик закончил кодирование и проверил изменения в рабочем продукте, подзадаче присваивается статус «Inspected».
6. Tested – протестировано. Переводится сотрудником, осуществляющим тестирование изменений в рабочий продукт (tester).
7. Closed – закрыто. В это состояние задача переводится ССВ по результатам отчёта о тестировании сделанных изменений.

Правила создания новой задачи

На начальном этапе проекта все участники команды могут предложить и создать новые задачи. Это необходимо, чтобы организовать систематизированную работу над проектом. Кроме того, в процессе работы над проектом могут возникнуть слишком объемные задачи, требующие разбиения на более мелкие или упрощения. В таких случаях создание новых задач помогает лучше распределить работы. Также, когда возникают проблемы или необходимо добавить новый функционал, участники команды могут создать задачу для баг-фиксинга или разработки нового функционала.

Правила перехода задач из состояния в состояние

Состояния задач всегда идут последовательно друг за другом, в некоторых случаях пункты могут опускаться или повторяться. Схема перехода из состояния в состояние показана на рисунке 2.

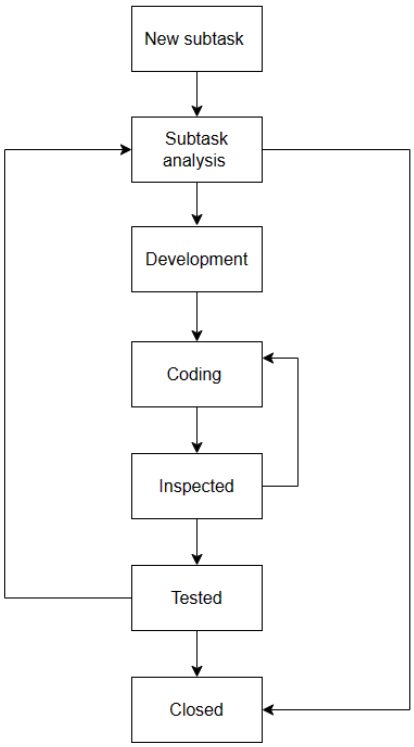

Рисунок 2 -Состояние задачи

4 Презентация проекта

На рисунке 3 представлена титульная страница презентации. Здесь мы можем видеть название нашего программного средства «Устранение дефектов на изображении».

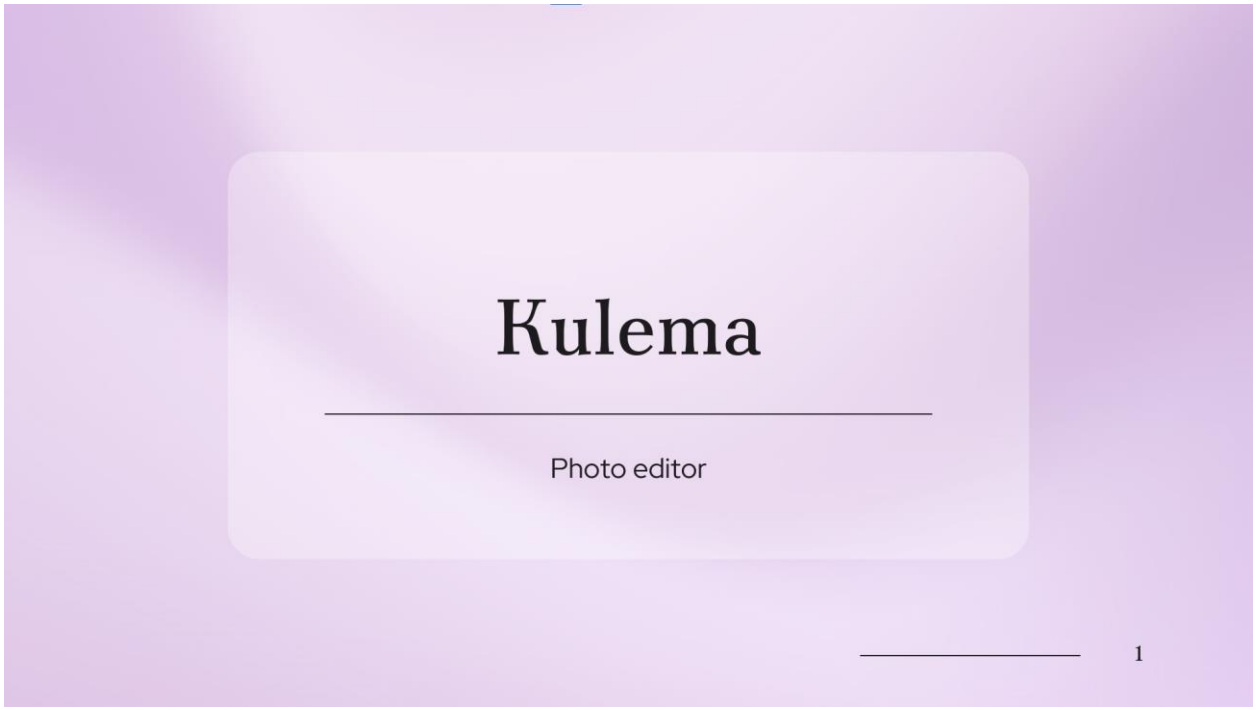

Рисунок 3 - Титульная страница

Актуальность проекта имеет важное значение для его успешной реализации.

Ниже можно ознакомиться с актуальностью предметной области разрабатываемого средства представлена на рисунке 4.

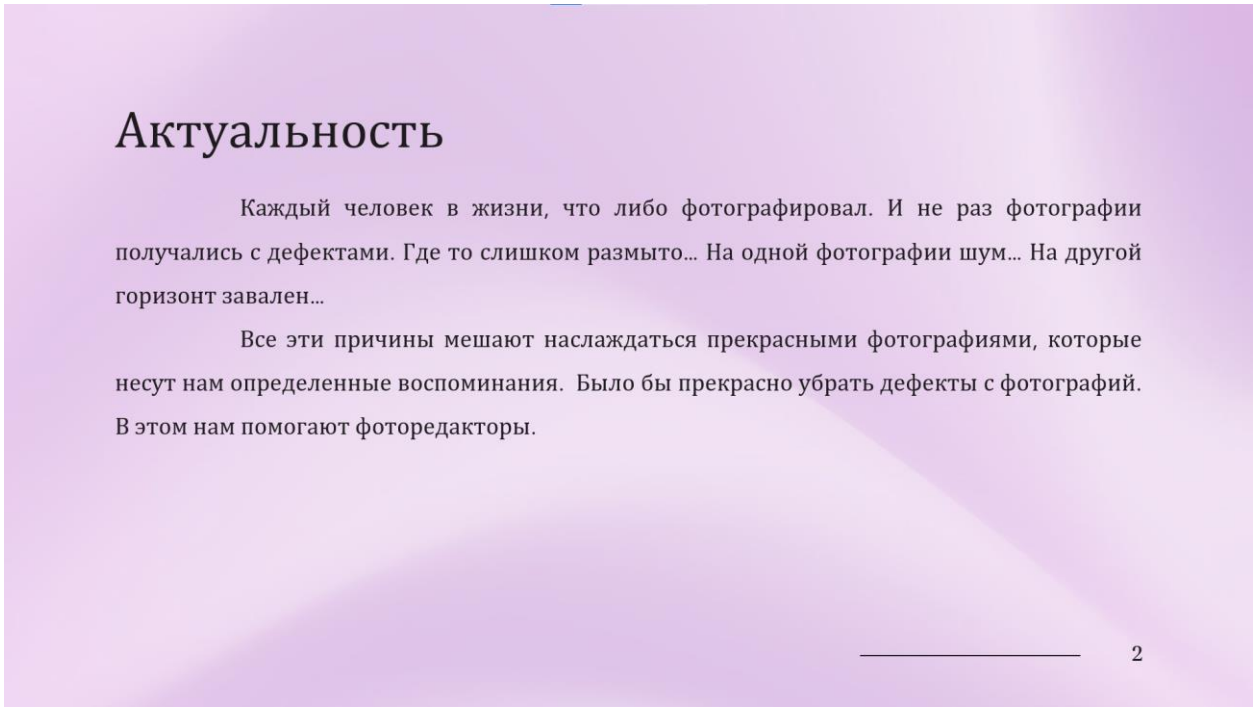

Рисунок 4 - Актуальность программного средства

На рисунке 5 предоставлена новизна нашего программного средства и чем он будет отличаться от существующих ПО.

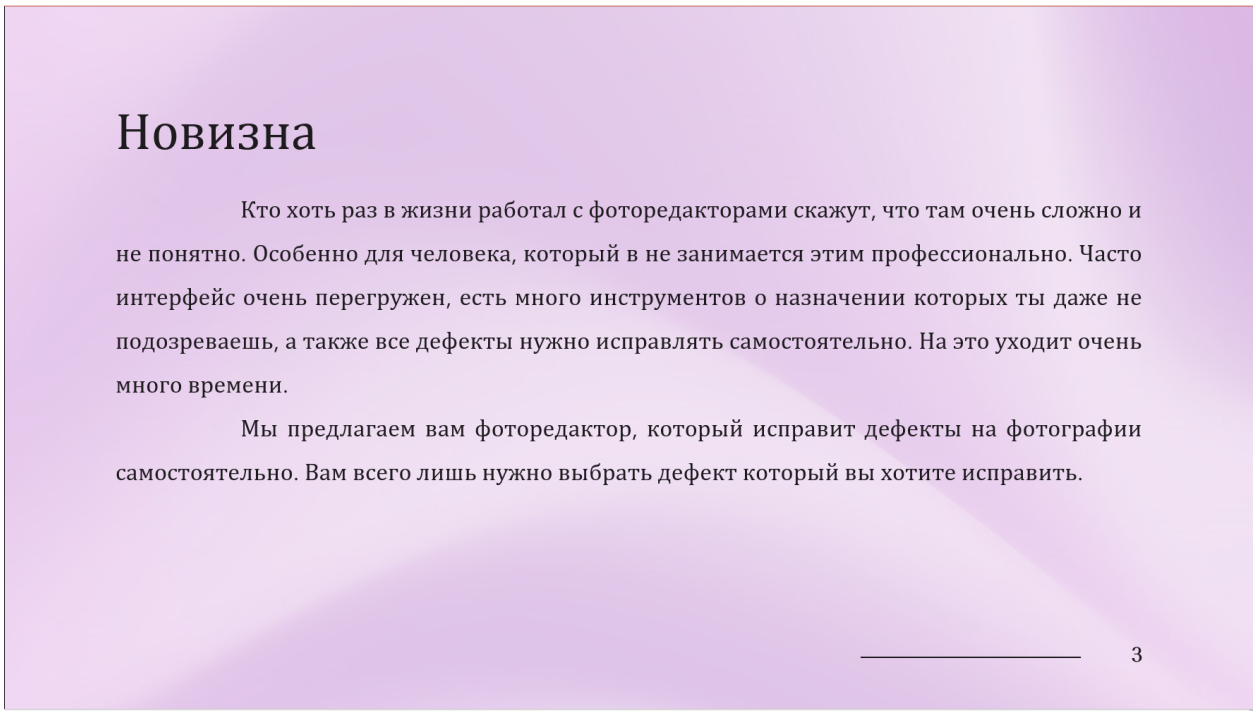

Рисунок 5 – Новизна

На рисунках ниже представлены дефекты, которые можно будет выбрать для устранения с изображения.

Рисунок 6 - Перечень устранимых дефектов

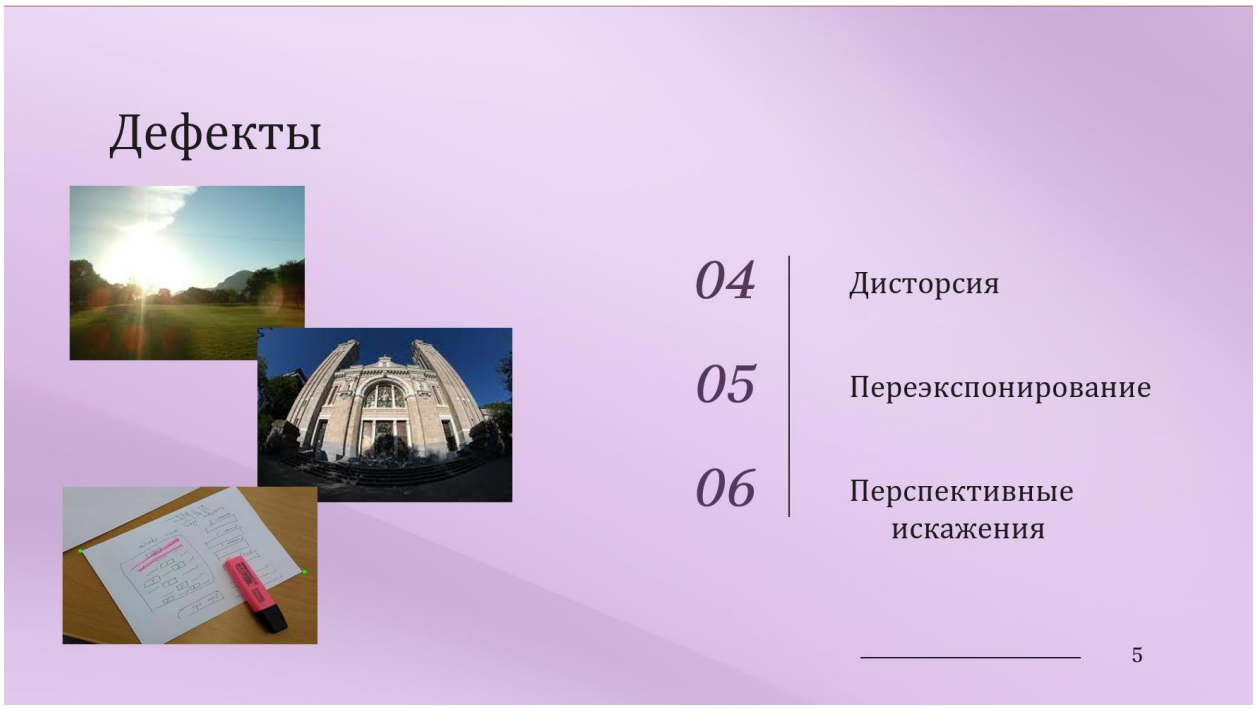

Рисунок 7 - Перечень устранимых дефектов

Рисунок 8 - Перечень устранимых дефектов

Также были добавлены дополнительные опции. Их можно увидеть на рисунке 9.

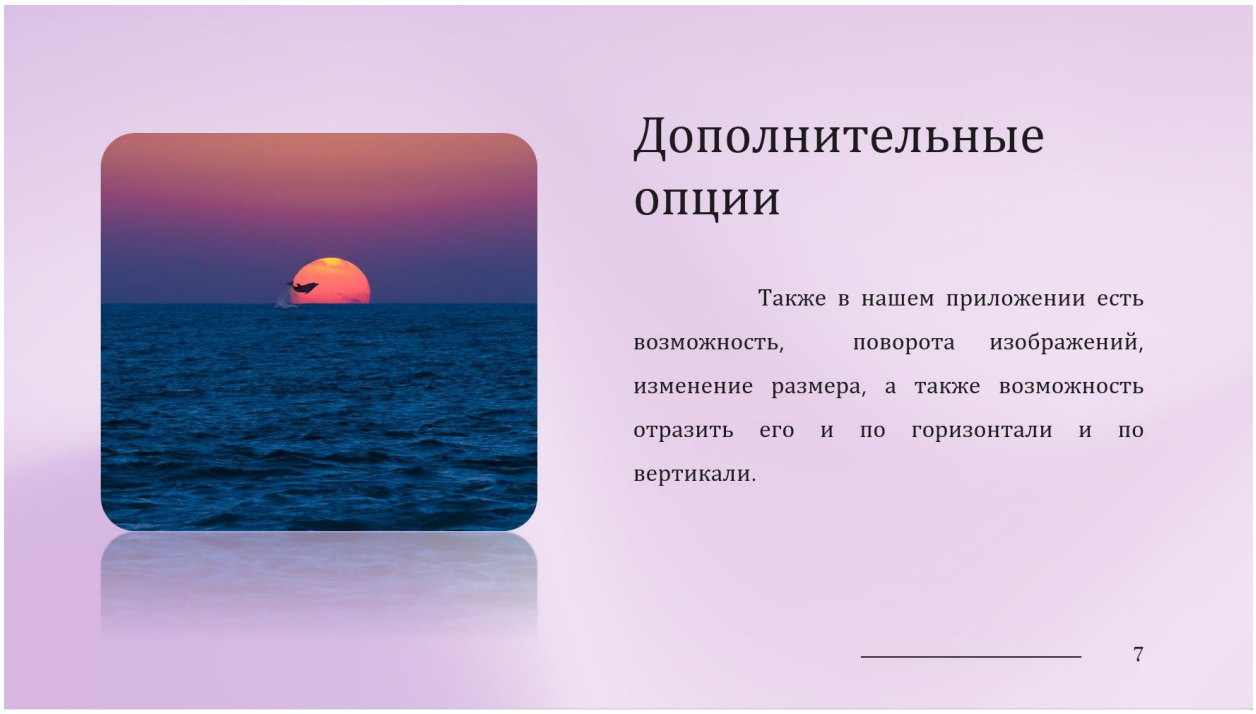

Рисунок 9 - Дополнительные опции ПО

5 Требования к проекту

Программный продукт «Программная система по удалению дефектов на изображении» предназначен для удаления выбранных дефектов пользователем с изображения.

Программный продукт «Программная система по удалению дефектов на изображении» состоит из следующих подсистем:

1. FE-1 – Пользовательский интерфейс.
2. FE-2 – База данных дефектов.
3. FE-3 – База данных изображений.
4. FE-4 – Дополнительные опции.

В таблице ниже можно увидеть матрицу требований к подсистемам.

Таблица 1 - Матрица требований к подсистемам

Матрица требований	Идентификаторы
Пользовательский интерфейс	FE-1, UI
Операции всех подсистем в интерфейсе	REQ-UI-1
Операции с базой данных дефектов	FE-2, DOD
Просмотр	REQ-DOD-1
Выбор дефекта «Шум»	REQ-DOD-2
Выбор дефекта «Размытие»	REQ-DOD-3
Выбор дефекта «Заваленный горизонт»	REQ-DOD-4
Выбор дефекта «Дисторсия»	REQ-DOD-5
Выбор дефекта «Переэкспонирование»	REQ-DOD-6
Выбор дефекта «Перспективные искажения»	REQ-DOD-7
Выбор дефекта «Хроматические аберрации»	REQ-DOD-8
Выбор дефекта «Инородные предметы на поверхности эмульсии»	REQ-DOD-9
Операции с базой данных изображений	FE-3, IDB
Просмотр изображений	REQ-IDB-1
Экспорт изображений	REQ-IDB-2

Продолжение таблицы 1

Редактирование изображений	REQ-IDB-3
Удалить изображение	REQ-IDB-4
Добавить изображение	REQ-IDB-5
Операции с дополнительными опциями	FE-4, AO
Поворот изображения	REQ-AO-1
Изменение размера изображения	REQ-AO-2
Отражение по горизонтали и по вертикали	REQ-AO-3

Расшифровка идентификаторов

– UI – User Interface (пользовательский интерфейс)

– DOD - Database Of Defects (база данных дефектов)

– IDB – Image database (база данных изображений)

– AO – Additional Options (дополнительные опции)

6 Разработка архитектуры проекта

Архитектура программного обеспечения — это основные структуры и принципы, на которых основана разработка программной системы. Включает в себя компоненты программного обеспечения, их взаимосвязи и свойства, а также способы организации и управления системой.

Архитектура программной системы – это подобие архитектуры здания, используемое в разработке программного обеспечения. Она является своего рода планом и проектом, описывающим задачи, которые должны быть выполнены командами разработчиков.

Архитектурная схема

На рисунке ниже представлена архитектурная схема системы по устранению дефектов на изображении.

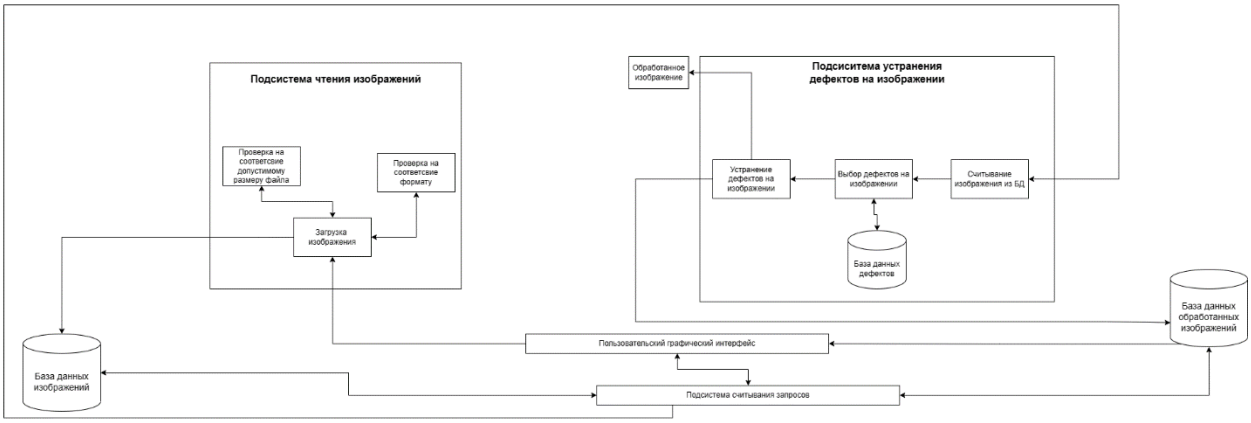

Рисунок 10 - Архитектурная схема

Данная схема показывает, что в нашей системе есть две подсистемы: подсистема чтения изображений и устранения дефектов на изображениях. Пользователь через графический интерфейс загружает изображения в подсистему чтения изображений, где идет проверка на корректность входных данных. Если проверка прошла успешно, то изображения загружаются в базу данных изображений. Далее пользователь решает, что он хочет сделать с этими изображениями. Если устранить дефекты, то данный запрос идет в подсистему считывания запросов. В данную подсистему подгружаются

изображения из базы данных. Далее для каждого изображения нужно выбрать дефекты из базы данных дефектов. После устранения выбранных дефектов, изображения загружаются в базу данных обработанных изображений. Подсистема восстановления изображений работает аналогично.

Архитектурная схема включает в себя следующие компоненты:

1. Пользовательский интерфейс.
2. Подсистема считывания запросов.
3. Подсистема чтения изображений.
   - Проверка на соответствие формату.
   - Проверка на соответствие допустимому формату файла.
4. База данных изображений.
5. Подсистема устранения дефектов на изображении.
   - Считывание изображения из базы данных.
   - Выбор дефектов на изображении.
   - База данных дефектов.
   - Устранение дефектов на изображении.
   - Обработанное изображение.

Архитектурно-контекстная диаграмма

Архитектурно контекстная диаграмма – это диаграмма, которая представляет собой описание системы вместе с ее взаимодействием с подсистемами. Диаграмма состоит из блоков подсистем, входов и выходов из систем.

На рисунке ниже представлена АКД системы по устранению дефектов на изображении.

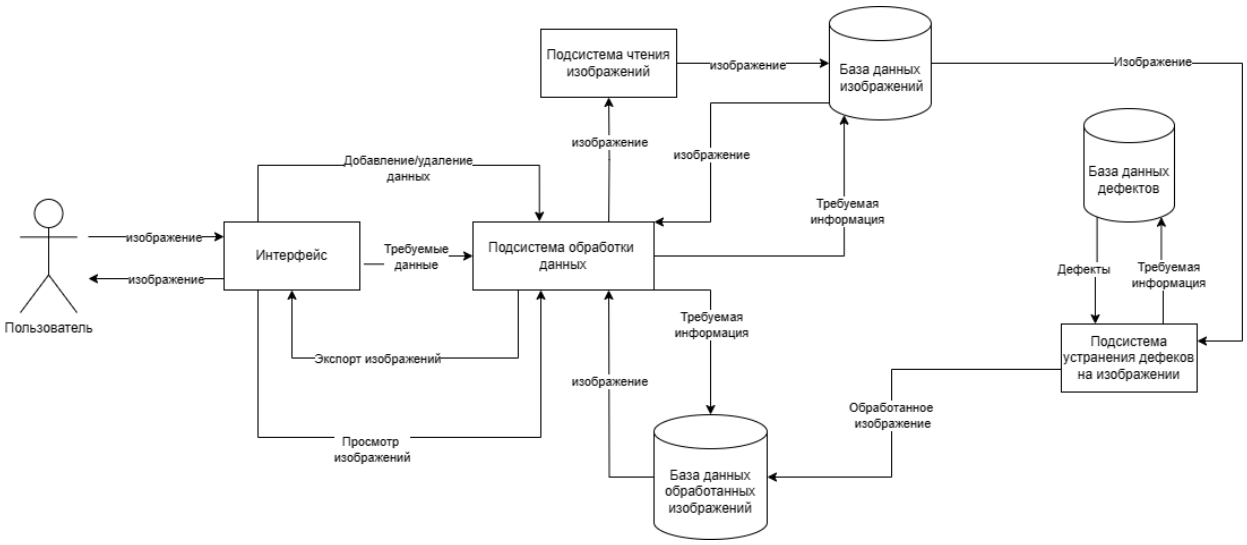

Рисунок 11 - Архитектурно-контекстная диаграмма

Система по удалению дефектов на изображении включает в себя следующие компоненты:

1. Подсистема чтения изображений.
2. База данных изображений.
3. База данных дефектов.
4. Подсистема устранения дефектов на изображении.
5. Подсистема восстановления изображений.
6. База данных обработанных изображений.
7. Подсистема обработки данных.
8. Интерфейс.

Диаграмма последовательностей состояний

Диаграмма последовательностей состояний – это подвид диаграмм взаимодействия, который позволяет описать взаимодействие между объектами в системе в виде последовательности сообщений, действий и операций, отображая порядок выполнения действий и обмена информацией между объектами.

Ниже представлена диаграмма последовательностей состояний.

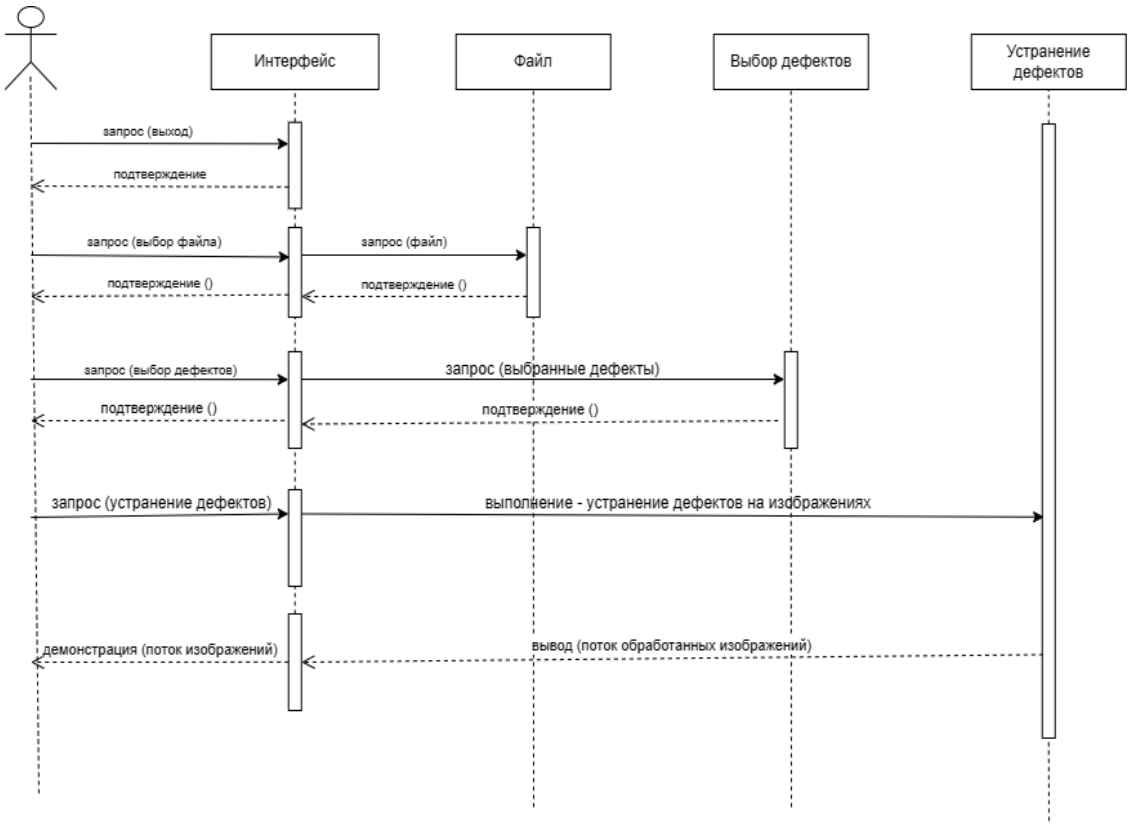

Рисунок 12 - Диаграмма последовательностей состояний

Диаграмма потоков данных

Диаграмма потоков данных — это графическое представление потока данных в информационной системе. С его помощью можно описывать входящие и выходящие потоки данных и хранилища этих данных.

В нашем случае на диаграмме можно увидеть, как протекают данные при загрузке изображений пользователем, при добавлении/удалении изображений, а также при устранении дефектов на изображениях.

Ниже представлена диаграмма потоков данных.

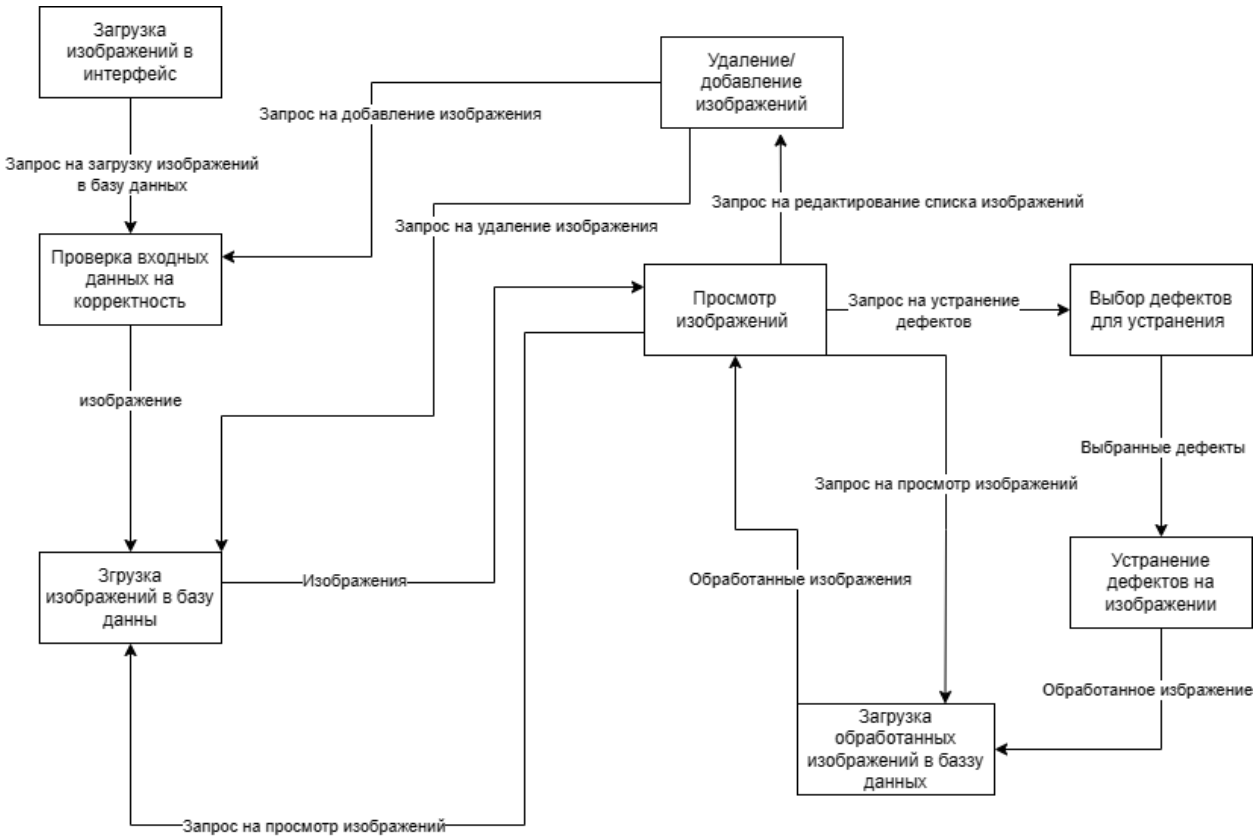

Рисунок 13 - Диаграмма потоков данных

Функциональная диаграмма

Функциональная диаграмма — это схема, отражающий взаимосвязи функций разрабатываемого программного обеспечения.

Ниже представлена функциональная диаграмма.

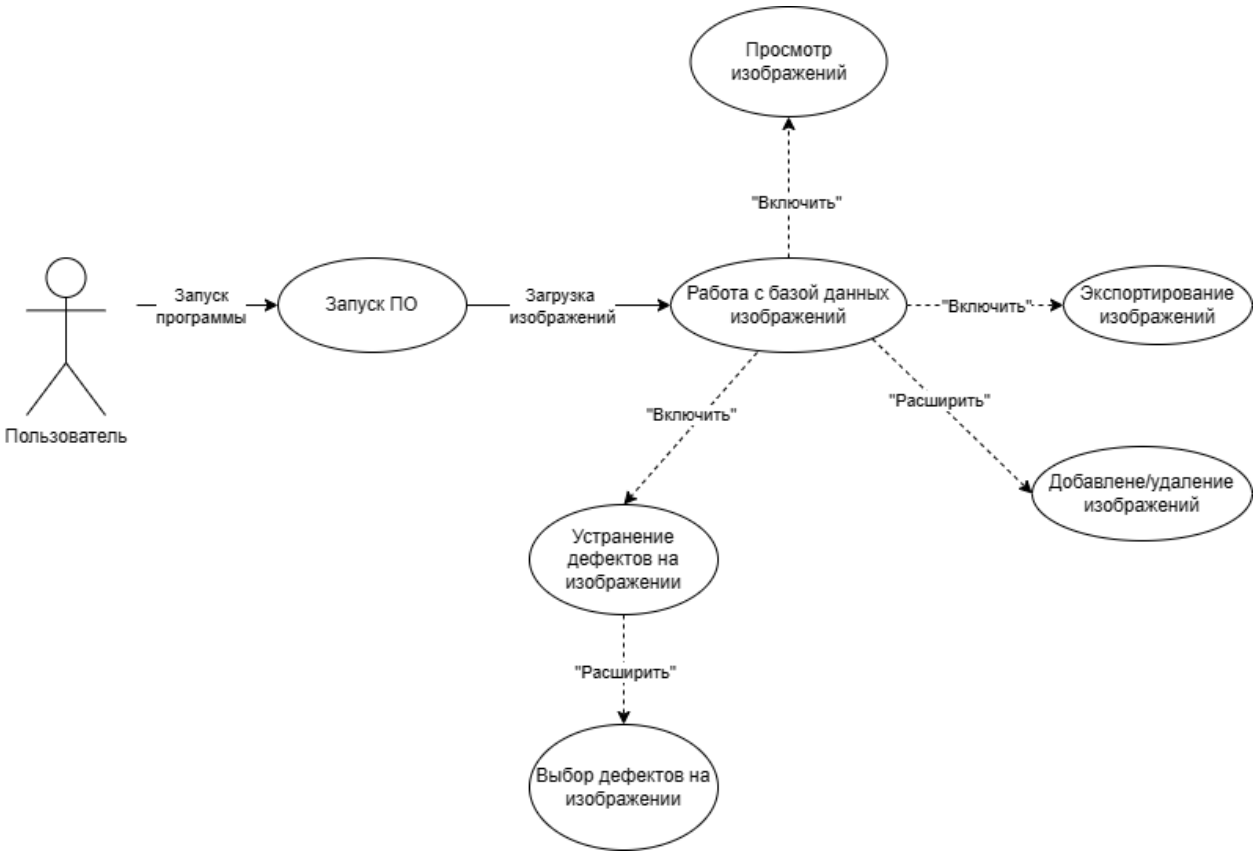

Рисунок 14 - Функциональная диаграмма

На этой диаграмме можно увидеть диалог с пользователем и функции нашей программы. После запуска программы пользователь загружает изображения в базу данных. Далее идет работа с базой данных изображений к ней же относится такая функция как удаление/добавление изображений. У данной функции есть свои подфункции, к которым относятся: просмотр изображений, экспортирование изображений, устранение дефектов на изображениях к ней же относится выбор дефектов на изображении.

7 Измерения проекта

Контроль над производственным процессом и его результатами является ключевым видом деятельности на современном предприятии, производящем программное обеспечение на заказ. В силу специфики такого продукта, как программное обеспечение, для оценки эффективности процесса и качества конечного продукта применяются особые методы.

Комплекс мероприятий, направленных на количественную оценку эффективности работы, называется программой измерений.

Метрика эффективности процесса производства

1. Inspection Fault Density (IFD)

IFD = (Количество найденных ошибок / Размер рабочего продукта)

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – инспекция, измеряемый атрибут – плотность найденных в ходе инспекции ошибок.

Единица измерения – ошибка / <страница, требование, LOC, тест>.

2. Faults Screening (FS)

FS = (Общее количество ошибок – Число не найденных ошибок)/ Общее количество ошибок * 100%;

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – проект, измеряемый атрибут – эффективность обнаружения дефектов.

Единица измерения – %

Метрика качества продукта

3. In Process Faults (IPF)

IPF = (Число обнаруженных ошибок до выпуска его релиза) / LOC;

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – продукт, измеряемый атрибут – плотность неполадок. Единица измерения – неполадка / LOC.

8 Перечень задач проекта

Подсистема FE-1

TASK-1 UI

Пользовательский интерфейс

До 15.01.2024 реализовать пользовательский интерфейс для работы с системой. Интерфейс должен включать в себя элементы всех подсистем системы (база данных дефектов, база данных изображений, дополнительные опции) – REQ-UI-1.

Подсистема FE-2

TASK-1 DOD

База данных дефектов

До 24.12.2023 реализовать алгоритмы устранения дефектов, предоставление выбора дефектов пользователю и просмотр перечня всех возможных дефектов, которые пользователь может выбрать для устранения с изображения - REQ-DOD-1, REQ-DOD-2, REQ-DOD-3, REQ-DOD-4, REQ-DOD-5, REQ-DOD-6, REQ-DOD-7, REQ-DOD-8, REQ-DOD-9.

Подсистема FE-3

TASK-1 IDB

База данных изображений

До 24.12.2023 реализовать возможность работы с изображением, то есть просмотр изображений, экспорт изображений, редактирование изображений (применение дефектов для их устранения с изображения), добавление и удаление из базы данных - REQ-IDB-1, REQ-IDB-2, REQ-IDB-3, REQ-IDB-4, REQ-IDB-5, REQ-IDB-6.

Подсистема FE-4

TASK-1 AO

Дополнительные опции

До 24.12.2023 реализовать дополнительные опции для работы с изображением (изменение размера, поворот изображение и отражение по горизонтали и по вертикали) – REQ-AO-1, REQ-AO-2, REQ-AO-3.

9 Правила по кодированию

Для создания качественного кода на любом языке программирования, обладающего таким свойствами, как удобочитаемость (readability) и понятность (understandability), необходимо следовать хорошо определённым стандартам и руководствам. Особенно это актуально при коллективной разработке программ.

Любой стандарт кодирования призван определить набор правил, которые способствуют разработке более единообразного кода и минимизации числа общераспространенных ошибок в нем, не ущемляя при этом права разработчика на творчество.

Рекомендации

1. Комментарии должны быть законченными предложениями.

Первое слово комментария должно быть написано с большой буквы, за исключением имен переменных, которые начинаются с маленькой буквы. После точки в конце предложения следует ставить два пробела. Комментарии могут быть написаны на русском языке.

2. Использование метода range.

Метод range предпочтительнее при использовании цикла для прохода по ряду чисел. Он потребляет меньше памяти по сравнению с использованием списка чисел в цикле.

Неправильно:

```
for i in [0, 1, 2, 3, 4, 5]:
  print(i)
```

Правильно:

```
for i in range(6):
  print(i)
```

3. Каждый импорт должен быть на отдельной строке.

Неправильно:

```
import cv2, numpy
```

Правильно:

```
import cv2
```

```
import numpy
```

4. Нежелательно использовать составные инструкции.

Не следует размещать несколько команд в одной строке.

Неправильно:

```
if foo == 'blah': do-blah-thing()
do-one(); do-two(); do-three()
```

Правильно:

```
if foo == 'blah':
  do-blah-thing()
do-one()
do-two()
do-three()
```

Запреты

1. Запрещено использование разных отступов внутри блока для вложенных инструкций.

Неправильно:

```
def long-function-name(var-one, var-two):
 print(var-one)
 print(var-two)
```

Правильно:

```
def long-function-name(var-one, var-two):
  print(var-one)
  print(var-two)
```

2. При программировании на Python запрещается использовать пробелы.

– Непосредственно внутри круглых, квадратных или фигурных скобок.

Неправильно:

```
spam( ham[ 1 ], { eggs: 2 }
```

Правильно:

```
spam(ham[1], {eggs: 2})
```

– Непосредственно перед запятой, точкой с запятой или двоеточием.

Неправильно:

```
if x == 4 : print(x , y) ; x , y = y , x
```

Правильно:

```
if x == 4: print(x, y); x, y = y, x
```

– Сразу перед открывающей скобкой, после которой начинается список аргументов при вызове функции.

Неправильно:

```
spam (1)
```

Правильно:

```
spam(1)
```

– Сразу перед открывающей скобкой, после которой следует индекс или срез.

Неправильно:

```
dict ['key'] = list [index]
```

Правильно:

```
dict['key'] = list[index]
```

3. Запрещается использовать объявления импорта где-то кроме верхней части файла.

Объявления импорта должны находиться только в верхней части файла, после комментариев модуля и символов документации, перед объявлениями глобальных переменных и констант.

Требования

1. Наследуйте свой класс исключения от Exception, а не от BaseException.

Прямое наследование от BaseException зарезервировано для исключений, которые не следует перехватывать.

2. Имена классов дополняются соответствующим префиксом либо постфиксом

– Абстрактные классы (интерфейсы) — префикс *I.*

– Классы реализации — постфикс *Imp* или *Impl*.

– Класс в составе иерархии — префикс *C.*

– Класс исключений – постфикс *Exception.*

Есть множество других вариантов, которые часто противоречат друг другу.

3. Имена функций дополняются префиксами.

– *is-* – выполняет проверку и возвращает логический тип – is-digit.

– *has-* – выполняет поиск какого-либо значения в контейнере – has-primeNumber.

– *get-* и *set-* – метод возвращает или устанавливает значение какого-либо поля – set-width, get-height.

4. Придерживайтесь при выборе наименований использования нижнего подчёркивания (snake_case) – some_variable = 5

5. Используйте схемы именования, позволяющие различать объекты и типы данных.

Имена пользовательских типов нужно начинать с заглавной буквы, а имена объектов – со строчной.

6. Для названий требуется использовать слова из предметной области решаемой задачи.

Имена классов должны выражаться существительными, функций и методов – глаголами. Имена классов должны быть хорошо различимы.

Неправильно:

```
x = 'Иван Петров'
y, z = x.split()
```

Правильно:

```
name = 'Иван Петров'
first-name, last-name = name.split()
```

10 Разработка плана тестирования проекта

Тестирование для подтверждения программного средства

В таблице 2 представлено разделение входных и выходных данных на классы эквивалентности.

Таблица 2 - Разделение входных и выходных данных на классы эквивалентности модуля "Пользовательский интерфейс"

Функция	Входные/выходные данные	Класс эквивалентности	Тестовая ситуация
Загрузка изображений	изображение формата JPEG	1	Изображения успешно загружены
		2	Изображения не были загружены
		3	Изображения были неверно названы
		4	Изображения не подходят по формату/размеру/весу
Добавление и удаление изображений	Изображение формата JPEG	5	Изображения были удалены/добавлены
Выбор дефектов для устранения	Перечень дефектов	6	Выбраны дефекты для устранения
		7	Выбрано ни одного дефекта для устранения
Экспорт изображения	изображение формата JPEG	8	Экспортировано изображение
		9	Ничего не экспортировано

На основе разделения входных и выходных данных на классы эквивалентности были созданы тесты для тестирования системы по методу черного ящика.

Тестирование чёрного ящика – стратегия тестирования функционального поведения объекта (программы, системы) с точки зрения внешнего мира, при котором не используются знания о структуре программы,

без привлечения сведений о логике программы (неизвестна программа, но известна постановка задачи).

Данные тесты представлены в таблице 3.

Таблица 3 -Тестирование по методу чёрного ящика модуля "Пользовательский интерфейс"

Функция	Тестовые ситуации	Входные данные	Выходные данные
Загрузка изображений	1	Изображение формата JPEG	Изображение формата JPEG
	3,4	Изображение формата JPEG	Окно ошибки «В наименовании изображения используются запрещенные символы, переименуйте файлы»
	2,3	Поток изображений формата JPEG	Окно ошибки «Такой формат не может быть обработан»
	2,3	Поток изображений формата JPEG	Окно ошибки «Файл «название» превышает 4 Мбайт»
	2,3	Поток изображений формата JPEG	Окно ошибки «Файл «название» превышает разрешения 1024*768»
Добавление и удаление изображений	5	Изображение формата JPEG	Поток изображений формата JPEG с добавленным изображением
		Поток изображений формата JPEG	Поток изображений формата JPEG с удаленным изображением
		Отсутствуют файлы	Окно ошибки «Невозможно удалить файл»

Выбор дефектов для устранения	7	Перечень дефектов	Окно ошибки «Дефекты не выбраны»
Выбор дефектов для устранения	6	Перечень дефектов	Перечень дефектов
Экспорт изображения	8	Поток изображений формата JPEG	Поток изображений формата JPEG
Экспорт изображения	9	Поток изображений формата JPEG	Файл ничего не содержит

Система устранения дефектов на изображениях

В таблице 4 представлено разделение входных и выходных данных на классы эквивалентности.

Таблица 4 - Разделение входных и выходных данных на классы эквивалентности модуля "Устранение дефектов на изображениях"

Функция	**Входные/выходные данные**	**Класс эквивалентности**	**Тестовая ситуация**
Устранение дефекта «Шум»	Изображение формата JPEG	1	Дефект устранен с потока изображений полностью
		2	Дефект устранен с потока изображений частично
		3	Дефект не был устранён ни с одного изображения потока изображений

		4	Дефект устранён не на всех изображениях потока
		5	Дефект устранен на всех изображениях потока
Устранение дефекта «Переэкспонирование»	Изображение формата JPEG	6	Дефект устранен с потока изображений полностью
		7	Дефект устранен с потока изображений частично
		8	Дефект не был устранён ни с одного изображения потока изображений
		9	Дефект устранён не на всех изображениях потока
		10	Дефект устранен на всех изображениях потока
Устранение дефекта «Перспективные искажения»	Изображение формата JPEG	11	Дефект устранен с потока изображений полностью
		12	Дефект устранен с потока

			изображений частично
		13	Дефект не был устранён ни с одного изображения потока изображений
		14	Дефект устранён не на всех изображениях потока
		15	Дефект устранен на всех изображениях потока
Устранение дефекта “Инородные предметы на поверхности эмульсии”	Изображение формата JPEG	16	Дефект устранен с потока изображений полностью
		17	Дефект устранен с потока изображений частично
		18	Дефект не был устранён ни с одного изображения потока изображений
		19	Дефект устранён не на всех изображениях потока

		20	Дефект устранен на всех изображениях потока
Устранение дефекта “Размытие”	Изображение формата JPEG	21	Дефект устранен с потока изображений полностью
		22	Дефект устранен с потока изображений частично
		23	Дефект не был устранён ни с одного изображения потока изображений
		24	Дефект устранён не на всех изображениях потока
		25	Дефект устранен на всех изображениях потока
Устранение дефекта “Дисторсия”	Изображение формата JPEG	26	Дефект устранен с потока изображений полностью
		27	Дефект устранен с потока изображений частично
		28	Дефект не был устранён ни с

			одного изображения потока изображений
		29	Дефект устранён не на всех изображениях потока
		30	Дефект устранен на всех изображениях потока
Устранение дефекта “Хроматические аберрации”	Изображение формата JPEG	31	Дефект устранен с потока изображений полностью
		32	Дефект устранен с потока изображений частично
		33	Дефект не был устранён ни с одного изображения потока изображений
		34	Дефект устранён не на всех изображениях потока
		35	Дефект устранен на всех изображениях потока
	Изображение формата JPEG	36	Дефект устранен с потока

			изображений полностью
		37	Дефект устранен с потока изображений частично
Устранение дефекта “Заваленный горизонт”		38	Дефект не был устранён ни с одного изображения потока изображений
		39	Дефект устранён не на всех изображениях потока
		40	Дефект устранен на всех изображениях потока

На основе разделения входных и выходных данных на классы эквивалентности были созданы тесты для тестирования системы по методу чёрного ящика.

Данные тесты представлены в таблице 5.

Таблица 5 - Тестирование по методу чёрного ящика модуля "Устранение дефектов на изображениях"

Функция	Тестовые ситуации	Входные данные	Выходные данные
Устранение дефекта «Шум»	1,4	Изображение формата JPEG	Изображение формата JPEG
	1,5		
	2,4		
	2,5		
	3		
	6,10		

Устранение дефекта «Переэкспонирование»	6,9 7,9 7,10 8	Изображение формата JPEG	Изображение формата JPEG
Устранение дефекта «Перспективные искажения»	11,14 11,15 12,14 12,15 13	Изображение формата JPEG	Изображение формата JPEG
Устранение дефекта “Инородные предметы на поверхности эмульсии”	16,19 16,20 17,19 17,20 18	Изображение формата JPEG	Изображение формата JPEG
Устранение дефекта “Размытие”	21,24 21,25 22,24 22,25 23	Изображение формата JPEG	Изображение формата JPEG
Устранение дефекта “Дисторсия	26,29 26,30 27,29 27,30 28	Изображение формата JPEG	Изображение формата JPEG
Устранение дефекта “Хроматические абберации”	31,34 31,35 32,34 32,35 33	Изображение формата JPEG	Изображение формата JPEG
Устранение дефекта “Заваленный горизонт”	36,39 36,40 37,39 37,40 38	Изображение формата JPEG	Изображение формата JPEG

11 Тестирование проекта

Заключение

В рамках курсовой работы было разработано программное средство «Программная система по устранению дефектов на изображении» с использованием подходов коллективной промышленной разработки, для чего были решены следующие поставленные задачи:

1. Разработан план проекта.
2. Разработан регламент проведения инспекции.
3. Разработана модель состояний задач.
4. Разработана презентация проекта.
5. Разработаны требования к проекту.
6. Разработана архитектура проекта.
7. Разработаны измерения проекта.
8. Разработан перечень задач проекта.
9. Разработаны рекомендации по кодированию.
10. Разработан план тестирования проекта.
11. Протестирован проект.

Таким образом, цель данной курсовой работы была достигнута.

Список литературы

1. Гриняк В.М. Лекции по дисциплине «Технологии коллективной промышленной разработки информационных систем». Электронный вариант.