## КАЛЕНДАРНО-ТЕМАТИЧЕСКИЙ ПЛАН

## ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ п/п** | Наименование темы и содержание работы | Количество дней | | Отметка о выполнении |
| по  плану | по  факту |
| 1. | Инструктаж, знакомство с рабочим местом и коллективом | 1 | 1 | Выполнено |
| 2. | Ознакомление с готовыми работами | 1 | 1 | Выполнено |
| 3. | Получение одобрения для начала работы  Тренировались в написании кода под присмотром руководства | 6 | 6 | Выполнено |
| 4. | Написание кода программы для ирригатора | 5 | 5 | Выполнено |
| 5. | Анализ ошибок и их исправление | 2 | 2 | Выполнено |
| 6. | Написание кода для работы с пломбами | 9 | 9 | Выполнено |
| 7. | Анализ ошибок и их исправление, закрепление сделанной работы | 3 | 3 | Выполнено |
| 8. | Активное участие в работе коллектива | 2 | 2 | Выполнено |
| 9. | Подготовка дневника отчёта, прощание с коллективом | 1 | 1 | Выполнено |
| **ИТОГО:** | | 30 | 30 |  |

1. **ДНЕВНИК-ОТЧЕТ ПО ВЫПОЛНЕННОЙ САМОСТОЯТЕЛЬНОЙ РАБОТЕ**

|  |  |  |
| --- | --- | --- |
| **Дата** | **№ темы Название темы**  **Содержание работы** | **Оценка и подпись**  **руководителя** |
| 01.11 | Как и в любом серьезном учреждении, день начался с инструктажа. Нас, практикантов, собрали в конференц-зале. Руководитель IT-отдела познакомила нас с основными аспектами работы клиники и деталями, касающимися нашего программного обеспечения. Разъяснили цели и задачи, стоящие перед нами, а также важность соблюдения стандартов безопасности и конфиденциальности данных пациентов. Участие в этом инструктаже стало для меня первым шагом в понимании специфики работы в сфере оказания медицинских услуг.  Ключевые моменты, выделенные на инструктаже:  Значение программного обеспечения в стоматологии: Обсуждали, как различные программы помогают в управлении записями пациентов, обработке рентгеновских снимков и даже в планировании лечения. Соблюдение HIPAA: Мы ознакомились с требованиями по защите информации пациентов, что особенно актуально в сфере здравоохранения. Техническая поддержка: Наша команда будет отвечать за своевременное обновление и функционирование медицинских систем, что подразумевает наличие планов на случай технических неполадок.  Знакомство с рабочим местом  После инструктажа я совершил экскурсию по клинике. Предоставили возможность ознакомиться с различными отделениями: от регистратуры до кабинетов врачей. В каждом из них используются специфические программные решения, адаптированные под нужды соответствующего отдела. Я увидел, как врачи работают с программами для диагностики и планирования лечения, а администраторы — с системами для управления записью пациентов.  Важно отметить, что кабинеты оборудованы современными технологиями, такими как цифровые рентген-аппараты и 3D-сканеры, которые оптимизируют процесс диагностики и лечения. Это вдохновляет и создает отличную атмосферу для внедрения новых IT-решений.  Знакомство с коллективом  В конце дня мне удалось пообщаться с коллегами по отделу программирования. Каждый из них имеет уникальный опыт и специализацию: одни занимаются разработкой ПО, другие — его внедрением. Атмосфера в команде была дружелюбной и открытой, поэтому вопросы и обсуждения проходили легко и непринужденно. Я почувствовал, что попал в коллектив, где ценят идеи и инициативу сотрудников.  Коллеги рассказали о своих проектах, поделились опытом и даже некоторыми трудностями, с которыми сталкивались в процессе работы. Это дало мне ясное представление о том, какую ценность я могу принести команде, а также о направлениях, в которых мне стоит развиваться.  Заключение  Первый день в клинике перевернул мои представления о том, как технологии могут интегрироваться в медицину. Инструктаж, знакомство с местом работы и коллективом стали основами для построения эффективной работы в будущих проектах. Я чувствую, что впереди меня ждут интересные задачи и новые вызовы, и я готов к ним. Внедряя технологии в стоматологию, мы можем не только повышать качество услуг, но и делать процесс лечения более комфортным для пациентов. |  |
| 02.11  03.11  04.11 | ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙ ДЕНЬ |  |

|  |  |  |
| --- | --- | --- |
| 05.11 | Введение  Вступая на путь профессиональной деятельности в рамках специальности "Программирование в стоматологии", я осознал, насколько важным является изучение уже реализованных проектов и готовых решений в данной области. Важно не только понимать теоретические аспекты программирования, но и видеть, как эти знания применяются на практике. В этом отчете я поделюсь своим опытом ознакомления с готовыми работами, которые были разработаны в нашей клинике и за ее пределами.  Цели ознакомления  Основные цели знакомства с готовыми работами:  Понимание текущих трендов в стоматологическом программировании: Я хотел понять, какие технологии и подходы наиболее часто используются, и какие новшества появляются в области. Изучение практической реализации программного обеспечения: Ознакомление с конкретными проектами позволило мне увидеть, как теоретические концепции воплощаются в жизнедеятельности клиники. Выявление проблем и решений: Понимание проблем, с которыми сталкиваются разработчики, а также изучение путей их решения может быть полезным для будущей работы и профессионального роста.  Ознакомление с проектами  В процессе ознакомления с готовыми lavori, я просматривал различные проекты, реализованные как внутри клиники, так и вне её. Вот некоторые из них:  Система управления пациентами: Одна из ключевых разработок в нашей клинике — это система, позволяющая эффективно управлять записями, а также хранить и анализировать медицинские данные пациентов. Проект охватывает интуитивно понятный интерфейс, позволяющий как медперсоналу, так и пациентам легко взаимодействовать с платформой. В рамках проекта я ознакомился с архитектурой приложения и принципами работы базы данных. Программное обеспечение для обработки рентгеновских снимков: В клинике использовали систему для автоматизированной обработки и анализа рентгеновских изображений. Эта работа особенно заинтересовала меня, так как она включает использование технологий машинного обучения для распознавания патологий на снимках. Я изучил основные алгоритмы, применяемые в этой системе, а также принципы работы с изображениями. Модуль для телемедицины.  В условиях пандемии сильно возросло значение телемедицинских услуг. Важно было разработать интерфейс для онлайн-консультаций, который бы обеспечивал безопасность и удобство. Я ознакомился с проектами, где за основу брались API для интеграции видеозвонков и чат-систем, а также изучил нюансы безопасности данных в подобных системах.  Анализ и рефлексия  Ознакомление с готовыми работами не только дало мне возможность узнать о текущих проектах, но и значительно углубило мое понимание требований, которые предъявляются к программному обеспечению в области стоматологии. Важным моментом было осознание того, как теоретические знания надстраиваются на практические задачи, а также как важно учитывать мнение пользователей при разработке систем.  Я заметил, что многие проблемы, с которыми сталкиваются разработчики, касаются юзабилити и интеграции с существующими системами. Это служит напоминанием о важности междисциплинарного подхода при разработке программного обеспечения.  Заключение  Ознакомление с готовыми работами в области программирования в стоматологии оказалось для меня бесценным опытом. Это не только расширило мой кругозор, но и дало стимул к дальнейшему изучению специфики работы в данной области. В будущем я планирую применять полученные знания и вдохновение для участия в разработке новых проектов в нашей клинике, с акцентом на инновационные решения и улучшение качества обслуживания пациентов. |  |

## 

|  |  |  |
| --- | --- | --- |
| 06.11 | В процессе моего обучения и практики в сфере программирования в стоматологии одной из ключевых задач стало получение одобрения для начала реальной разработки проектов. Это требовало не только теоретических знаний, но и практического опыта в написании кода, что возможно только под присмотром более опытных специалистов. В этом отчете я поделюсь своим опытом, связанным с получением одобрения на начало работы и с тем, как я тренировался в написании кода в такой обстановке.  Получение одобрения на начало работы  Процесс получения одобрения был важным этапом в моей практике. Он состоял из нескольких ключевых шагов:  Подготовка проекта: Я подготовил концепцию своего проекта, в которой нужно было описать его функциональность, цели, ожидаемые результаты и технологии, которые я собирался использовать. Основная задача заключалась в том, чтобы убедить руководство в реальности и целесообразности моего проекта. Презентация: Следующим шагом была презентация моего проекта перед командой руководства и коллегами. Я рассказал о своих идеях и ответил на вопросы. Это развивало мои навыки коммуникации и уверенности в себе, а также дало возможность получить конструктивную обратную связь. Получение отзывов и правок. После презентации я получил ряд замечаний и предложений, которые помогли улучшить проект. Это был важный момент, так как я понял, что обратная связь является незаменимым инструментом в процессе разработки. |  |
| 07.11 | Финальное одобрение: Завершив все корректировки, я вновь представил проект, и после обсуждения получил одобрение для начала работы. Этот момент стал для меня важным достижением в рамках практики.  Тренировки в написании кода под присмотром руководства  После получения одобрения, я приступил к практическому написанию кода, что стало еще одним значительным шагом в моем профессиональном развитии. Работа проходила под присмотром опытного наставника, что обеспечивало чувство уверенности и безопасности в процессе обучения. Вот несколько ключевых аспектов этого этапа:  Наставничество: Мой руководитель провел вводный курс по использованию технологий и инструментов, необходимых для реализации проекта. Он объяснил мне основные принципы кодирования, практики чистого кода и сразу же поделился ценными советами по отладке и тестированию. Код-ревью  После написания каждого фрагмента кода я представлял его на проверку. В процессе код-ревью мой наставник указывал на ошибки, предлагал улучшения и делился лучшими практиками. Этот опыт значимо повысил мое качество программной разработки и помог понять, как важно учитывать мнение команды.  Совместное решение задач: Мы также часто работали над сложными задачами вместе, что позволило мне быстрее находить решения и понимать подходы к решению различных проблем. |  |
| 08.11 | Совместная работа способствовала обмену опытом и развитию критического мышления. Практическое применение: Я сразу же применял полученные знания в процессе работы над проектом. Это позволило мне закрепить навыки и увидеть, как теоретические аспекты программирования влияют на реальное программное обеспечение в стоматологии.  Заключение  Получение одобрения на начало работы и тренировки в написании кода под руководством опытного наставника стали решающими шагами в моем обучении программированию в стоматологии. Эти этапы не только позволили мне начать реализацию моего проекта, но и дали возможность развить практические навыки, а также научили важности командного взаимодействия и получения обратной связи. Я уверен, что данный опыт станет важной основой для моего дальнейшего профессионального роста в области программирования.  Выбор языка программирования и технологий.  На первом этапе я выбрал язык программирования, который будет использоваться в проекте. В данном случае это был язык Python, поскольку он широко применяется для разработки приложений в области медицины и биоинформатики. Кроме того, я решил использовать фреймворк Django для создания веб-приложения, что позволило мне не только углубиться в язык программирования, но и ознакомиться с принципами работы веб-технологий.  Работа с наставником.  На протяжении написания кода я имел возможность работать под руководством опытного разработчика. Наставник помогал мне не только в решении технических задач, но и в понимании принципов разработки программного обеспечения. |  |
| 09.11  10.11 | ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙДЕНЬ |  |
| 11.11 | Мы регулярно проводили код-ревью, что позволяло выявлять ошибки и находить более эффективные решения на ранних стадиях разработки.  Практика и рефлексия.  Каждый раз, когда я завершал определенную часть кода, я проводил само-рефлексию. Это заключалось в анализе того, что у меня получилось, какие трудности возникли и каким образом я их преодолел. Я также старался документировать свой код, описывая функции, их параметры и ожидаемые результаты. Эти записи помогали мне лучше понимать свою работу и упрощали процесс редактирования кода в будущем.  Тестирование и отладка.  Одной из важных частей кодирования является тестирование. Я научился писать юнит-тесты для проверки работоспособности функций. Это оказалось полезным не только для оценки исправлений, но и для получения уверенности в том, что изменения, которые я вношу, не нарушают уже реализованный функционал. Под пристальным наблюдением своего наставника я освоил методы отладки, что сделало процесс разработки менее стрессовым и более управляемым.  Совместная работа и Agile-методы.  В рамках проекта я также познакомился с методами Agile, что способствовало лучшей организации рабочего процесса. Мы использовали подходы Scrum и Kanban, что позволяло нам эффективно делиться задачами и отслеживать прогресс. Совместная работа с другими участниками команды обогатила мой опыт и позволила увидеть, как взаимодействие в команде влияет на конечный результат.  Заключение  Получение одобрения на запуск проекта и выполнение практических заданий под руководством опытного наставника стали важными этапами в моем обучении программированию в области стоматологии. Этот опыт не только углубил мои знания в технических аспектах разработки программного обеспечения, но и развил важные навыки коммуникации и сотрудничества, которые необходимы для работы в команде в реальных условиях. |  |
| 12.11 | Я понял, что программирование — это не только написание кода, но и сложный процесс, требующий критического мышления, внимательности к деталям и способности адаптироваться к изменяющимся условиям. Этот опыт стал основой для моего дальнейшего развития как программиста и специалиста в области стоматологии, и я с нетерпением жду возможности применять свои знания и навыки в будущих проектах.  позволило более гибко управлять временем и задачами.  Я принимал участие в ежедневных стендапах, на которых мы обсуждали, какие задачи были выполнены, что планируется на день и какие препятствия возникли. Этот формат общения помогал поддерживать командный дух и обеспечивать прозрачность в работе. Мы также использовали инструменты для управления проектами, такие как Jira, что облегчало отслеживание прогресса и распределение задач.  Получение обратной связи.  Регулярная обратная связь от наставника и других членов команды ускорила мой рост как разработчика. Я получил возможность обсудить не только свои успехи, но и ошибки. Обсуждения как положительных, так и отрицательных моментов обучения позволили мне увидеть, как важно учиться на своих ошибках и не бояться их признавать. Этот опыт укрепил доверие внутри команды и создал поддерживающую среду, способствующую обучению.  Дальнейшее самообразование. В дополнение к работе над проектом я продолжал изучать дополнительные ресурсы, такие как онлайн-курсы и книги по программированию и веб-разработке. Я активно искал статьи и видеоматериалы о современных методах разработки, библиотек Python, таких как Pandas и NumPy, которые пригодились мне для работы с данными. Самообразование стало важным аспектом моего развития, так как технологии постоянно меняются, и я понимал, что необходимо быть в курсе последних трендов. |  |
| 13.11 | Финальная интеграция и развёртывание.  На завершающем этапе проекта я принял участие в интеграции всех компонентов системы. Это требовало тщательного тестирования и настройки взаимодействия между различными частями приложения. Я не только проверял функциональность, но и осуществлял работу по развертыванию приложения на удаленном сервере. Процесс развертывания включал в себя работу с контейнерами Docker, что стало отличным опытом в управлении и настройке сред для разработки и эксплуатации.  Подведение итогов и рефлексия.  По завершении проекта мы провели оценку результатов. Обсудили, что получилось хорошо, а что можно улучшить в будущем. Эта встреча позволила всем членам команды поделиться своим опытом и дать рекомендации по улучшению рабочего процесса. Я осознал важность этой практики для дальнейшего роста и развития как команды, так и каждого из ее участников.  Перспективы и планы на будущее. Пройденный путь и все полученные навыки вдохновили меня на дальнейшее изучение и повышение квалификации. Я планирую погрузиться в более сложные аспекты веб-разработки, такие как создание приложений с использованием микро-сервисной архитектуры и изучение различных методов работы с базами данных. Моей цель является стать не только разработчиком, но и экспертом в области разработки масштабируемых и эффективных веб-приложений.  Таким образом, мой опыт в проекте стал не только возможностью применить на практике изученные знания, но и ценным этапом в понимании сложности и увлекательности работы в команде разработчиков. Этот процесс научил меня, как совмещать теорию и практику, что крайне важно для дальнейшей карьеры в области информационных технологий.  этапом моего обучения и профессионального развития.  Выбор языка программирования и технологий.  На первом этапе я выбрал язык программирования, который будет использоваться в проекте. В данном случае это был язык Python, поскольку он широко применяется для разработки приложений в области медицины и биоинформатики. Кроме того, я решил использовать фреймворк Django для создания веб-приложения, что позволило мне не только углубиться в язык программирования, но и ознакомиться с принципами работы веб-технологий.  Работа с наставником.  На протяжении написания кода я имел возможность работать под руководством опытного разработчика. Наставник помогал мне не только в решении технических задач, но и в понимании принципов разработки программного обеспечения. Мы регулярно проводили код-ревью, что позволяло выявлять ошибки и находить более эффективные решения на ранних стадиях разработки.  Практика и рефлексия.  Каждый раз, когда я завершал определенную часть кода, я проводил саморефлексию. Это заключалось в анализе того, что у меня получилось, какие трудности возникли и каким образом я их преодолел. Я также старался документировать свой код, описывая функции, их параметры и ожидаемые результаты. Эти записи помогали мне лучше понимать свою работу и упрощали процесс редактирования кода в будущем.  Тестирование и отладка.  Одной из важных частей кодирования является тестирование. Я научился писать юнит-тесты для проверки работоспособности функций. Это оказалось полезным не только для оценки исправлений, но и для получения уверенности в том, что изменения, которые я вношу, не нарушают уже реализованный функционал. Под пристальным наблюдением своего наставника я освоил методы отладки, что сделало процесс разработки менее стрессовым и более управляемым.  Совместная работа и Agile-методы В рамках проекта я также познакомился с методами Agile, что способствовало лучшей организации рабочего процесса. Мы использовали подходы Scrum и Kanban, что позволяло нам эффективно делиться задачами и отслеживать прогресс. Совместная работа с другими участниками команды обогатила мой опыт и позволила увидеть, как взаимодействие в команде влияет на конечный результат.  Заключение  Получение одобрения на запуск проекта и выполнение практических заданий под руководством опытного наставника стали важными этапами в моем обучении программированию в области стоматологии. Этот опыт не только углубил мои знания в технических аспектах разработки программного обеспечения, но и развил важные навыки коммуникации и сотрудничества, которые необходимы для работы в команде в реальных условиях.  Я понял, что программирование — это не только написание кода, но и сложный процесс, требующий критического мышления, внимательности к деталям и способности адаптироваться к изменяющимся условиям. Этот опыт стал основой для моего дальнейшего развития как программиста и специалиста в области стоматологии, и я с нетерпением жду возможности применять свои знания и навыки в будущих проектах.  позволило более гибко управлять временем и задачами. Я принимал участие в ежедневных стендапах, на которых мы обсуждали, какие задачи были выполнены, что планируется на день и какие препятствия возникли. Этот формат общения помогал поддерживать командный дух и обеспечивать прозрачность в работе. Мы также использовали инструменты для управления проектами, такие как Jira, что облегчало отслеживание прогресса и распределение задач.  Получение обратной связи. Регулярная обратная связь от наставника и других членов команды ускорила мой рост как разработчика. Я получил возможность обсудить не только свои успехи, но и ошибки. Обсуждения как положительных, так и отрицательных моментов обучения позволили мне увидеть, как важно учиться на своих ошибках и не бояться их признавать. Этот опыт укрепил доверие внутри команды и создал поддерживающую среду, способствующую обучению.  Дальнейшее самообразование.  В дополнение к работе над проектом я продолжал изучать дополнительные ресурсы, такие как онлайн-курсы и книги по программированию и веб-разработке. Я активно искал статьи и видеоматериалы о современных методах разработки, библиотек Python, таких как Pandas и NumPy, которые пригодились мне для работы с данными. Самообразование стало важным аспектом моего развития, так как технологии постоянно меняются, и я понимал, что необходимо быть в курсе последних трендов.  Финальная интеграция и развёртывание.  На завершающем этапе проекта я принял участие в интеграции всех компонентов системы. Это требовало тщательного тестирования и настройки взаимодействия между различными частями приложения. Я не только проверял функциональность, но и осуществлял работу по развертыванию приложения на удаленном сервере. Процесс развертывания включал в себя работу с контейнерами Docker, что стало отличным опытом в управлении и настройке сред для разработки и эксплуатации.  Подведение итогов и рефлексия. По завершении проекта мы провели оценку результатов. Обсудили, что получилось хорошо, а что можно улучшить в будущем. Эта встреча позволила всем членам команды поделиться своим опытом и дать рекомендации по улучшению рабочего процесса. Я осознал важность этой практики для дальнейшего роста и развития как команды, так и каждого из ее участников.  Перспективы и планы на будущее.  Пройденный путь и все полученные навыки вдохновили меня на дальнейшее изучение и повышение квалификации. Я планирую погрузиться в более сложные аспекты веб-разработки, такие как создание приложений с использованием микросервисной архитектуры и изучение различных методов работы с базами данных. Моей цель является стать не только разработчиком, но и экспертом в области разработки масштабируемых и эффективных веб-приложений.  Таким образом, мой опыт в проекте стал не только возможностью применить на практике изученные знания, но и ценным этапом в понимании сложности и увлекательности работы в команде разработчиков. Этот процесс научил меня, как совмещать теорию и практику, что крайне важно для дальнейшей карьеры в области информационных технологий. |  |
| 14.11 | Ирригаторы стали неотъемлемой частью современного ухода за полостью рта, предоставляя пользователям удобный и эффективный способ очистки межзубных промежутков. Моя задача заключалась в разработке программного обеспечения для управления работой ирригатора, что включало создание интерфейса, реализацию алгоритмов управления подачей жидкости и интеграцию сенсоров для обратной связи.  1. Определение требований к программному обеспечению  Перед началом разработки программы я встретился с стоматологами и пользователями для обсуждения функциональных возможностей ирригатора. Мы определили следующие ключевые функции:  Управление режимами работы: возможность переключения между различными режимами работы (например, чистка, массаж и промывание). Настройка мощности: пользователь мог выбирать интенсивность подачи воды. Таймер: автоматическое отключение после заданного времени работы. Индикация уровня воды: использование датчика уровня для предупреждения о необходимости дозаправки.  Эти функции стали основой для дальнейшей разработки.  2. Проектирование архитектуры программы  В процессе проектирования я выбрал модульную архитектуру для облегчения повторного использования кода и его тестирования. Основные модули включали:  Интерфейс пользователя: отвечал за взаимодействие с пользователем, включая кнопки для изменения режимов и настройки. Управляющий модуль: обрабатывал команды от интерфейса и определял действия, которые нужно выполнить. Модуль датчиков: собирал данные от датчиков уровня воды и других сенсоров. Коммуникационный модуль: отвечал за связь между модулями и выполнением команд управления насосом и другими механическими частями ирригатора.  3. Выбор технологий  Для разработки было решено использовать язык программирования Python из-за его простоты и обширной библиотеки для работы с аппаратным обеспечением. Библиотека RPi.GPIO была выбрана для работы с Raspberry Pi, которая использовалась в проекте для управления оборудованием. |  |
| 15.11 | 4. Написание кода  На этом этапе я приступил к написанию кода. Первый шаг заключался в создании интерфейса пользователя с использованием библиотеки Tkinter, что позволило сделать наглядное и удобное приложение. Ниже представлен пример кода, реализующего одну из функций:  python import tkinter as tk from time  import sleep  class IrrigatorApp: def \_\_init\_\_(self, master):  self.master = master master.title("Ирригатор")  self.mode\_label = tk.Label(master, text="Выберите режим:") self.mode\_label.pack() self.mode\_var = tk.StringVar(value="Cleaning") self.mode\_menu = tk.OptionMenu(master, self.mode\_var, "Cleaning", "Massaging", "Rinsing")  self.mode\_menu.pack()  self.power\_label = tk.Label(master, text="Сила потока:") self.power\_label.pack() self.power\_scale = tk.Scale(master, from\_=1, to=10, orient=tk.HORIZONTAL) self.power\_scale.pack()  self.start\_button = tk.Button(master, text="Начать", command=self.start\_irrigator)  self.start\_button.pack()  def start\_irrigator(self):  mode = self.mode\_var.get()  power = self.power\_scale.get()  print(f"Запуск ирригатора: Режим - {mode}, Сила потока - {power}")  if name == "main":  root = tk.Tk()  app = IrrigatorApp(root)  root.mainloop()  5. Тестирование программы  После завершения разработки был выполнен тест всех функциональных возможностей программы. Я проверил работу каждого из режимов, работу датчиков уровня и кнопку отключения. Обратная связь от команды стоматологов была негативной — на этапе тестирования обнаружилось, что автоматически отключающийся таймер работает некорректно. Я внес изменения в алгоритм, и после доработки программа стала работать стабильно.  6. Итоговая интеграция и развертывание  После успешного тестирования была проведена интеграция с аппаратной частью ирригатора. Все модули связались, и программа успешно управляла работой устройства. Установка прошла гладко, и по результатам тестирования на пользователях была получена положительная обратная связь.  Заключение  Данная практика по специальности "Программирование в стоматологии" позволила мне не только закрепить знания о программировании, но и научиться работать с реальным оборудованием. Я понял, как важно учитывать пожелания пользователей при написании ПО и как сильно программное обеспечение влияет на эффективность применения медицинских устройств. Этот опыт вдохновил меня продолжать развиваться в области разработки программного обеспечения для медицины. |  |
| 16.11  17.11 | ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙ ДЕНЬ |  |
| 18.11 | 5. Тестирование программы  После написания кода я приступил к тестированию. Важно было убедиться, что все функции работают корректно и отвечают требованиям. Я провел следующие виды тестов:  Функциональные тесты: проверил, что все кнопки и функции интерфейса работают так, как задумано. Это включало переключение режимов, изменение интенсивности и запуск ирригатора. Стресс-тесты: протестировал программу в различных режимах работы, чтобы убедиться в ее надежности и эффективности под нагрузкой. Тестирование сенсоров: проверил правильность работы датчиков уровня воды, запуская программу с различными значениями, чтобы убедиться, что она корректно идентифицирует уровень жидкости и отключает насос при необходимости.  6. Заключение и выводы  Разработка программного обеспечения для ирригатора дала мне бесценный опыт в области программирования и взаимодействия с аппаратными компонентами. Я научился обращаться с различными библиотеками и следующий шаг в моей практике будет состоять в улучшении интерфейса, добавлении новых функций и оптимизации алгоритмов управления.  Следующий шаг  Планирую продолжить работу над проектом, добавив возможность сохранения пользовательских профилей, что позволит пользователям настраивать ирригатор под свои индивидуальные предпочтения. Также было бы полезно интегрировать функцию мониторинга состояния здоровья полости рта, собирая данные о частоте использования ирригатора и его влиянии на здоровье зубов.  Эта практика значительно расширила мой кругозор в области программирования в медицине, и я надеюсь использовать полученные знания в будущем для разработки более сложных и полезных приложений. |  |
| 19.11 | 4. Написание кода  После определения требований и выбора технологий началась непосредственная реализация кода. Основные этапы написания кода включали:  Создание интерфейса: я разработал главный экран программы с кнопками для переключения между режимами работы и настройки интенсивности.  python import tkinter as tk class IrrigatorApp:  def \_\_init\_\_(self, master):  self.master = master master.title("Управление ирригатором") self.mode\_label = tk.Label(master, text="Выберите режим:") self.mode\_label.pack()  self.mode\_var = tk.StringVar(value="Обычный")  self.modes = ["Обычный", "Массаж", "Промывание"]  for mode in self.modes:  btn = tk.Radiobutton(master, text=mode, variable=self.mode\_var, value=mode)  btn.pack() self.intensity\_label = tk.Label(master, text="Настройте мощность подачи:")  self.intensity\_label.pack() self.intensity\_scale = tk.Scale(master, from=1, to=10, orient=tk.HORIZONTAL)  self.intensity\_scale.pack()  self.start\_button = tk.Button(master, text="Запустить", command=self.start\_irrigator)  self.start\_button.pack()  def start\_irrigator(self):  mode = self.mode\_var.get()  intensity = self.intensity\_scale.get()  print(f"Запуск ирригатора в режиме {mode} с мощностью {intensity}")  root = tk.Tk()  app = IrrigatorApp(root)  root.mainloop()  Разработка управляющего модуля: этот модуль обрабатывает команды, поступающие от интерфейса, и управляет насосом в зависимости от режима и настроек мощности.  Python  class IrrigatorController:  def init (self):  self.mode = "Обычный"  self.intensity = 5  def start\_irrigator(self, mode, intensity):  self.mode = mode  self.intensity = intensity  if self.mode == "Обычный":  self.activate\_pump(intensity)  def activate  print(self, intensity):  print(f"Насос активирован с интенсивностью {intensity}") |  |
| 20.11 | Ирригаторы стали неотъемлемой частью современного ухода за полостью рта, предоставляя пользователям удобный и эффективный способ очистки межзубных промежутков. Моя задача заключалась в разработке программного обеспечения для управления работой ирригатора, что включало создание интерфейса, реализацию алгоритмов управления подачей жидкости и интеграцию сенсоров для обратной связи.  1. Определение требований к программному обеспечению  Перед началом разработки программы я встретился с стоматологами и пользователями для обсуждения функциональных возможностей ирригатора. Мы определили следующие ключевые функции:  Управление режимами работы: возможность переключения между различными режимами работы (например, чистка, массаж и промывание). Настройка мощности: пользователь мог выбирать интенсивность подачи воды. Таймер: автоматическое отключение после заданного времени работы. Индикация уровня воды: использование датчика уровня для предупреждения о необходимости дозаправки.  Эти функции стали основой для дальнейшей разработки.  2. Проектирование архитектуры программы  В процессе проектирования я выбрал модульную архитектуру для облегчения повторного использования кода и его тестирования. Основные модули включали:  Интерфейс пользователя: отвечал за взаимодействие с пользователем, включая кнопки для изменения режимов и настройки. Управляющий модуль: обрабатывал команды от интерфейса и определял действия, которые нужно выполнить. Модуль датчиков: собирал данные от датчиков уровня воды и других сенсоров. Коммуникационный модуль: отвечал за связь между модулями и выполнением команд управления насосом и другими механическими частями ирригатора.  3. Выбор технологий  Для разработки было решено использовать язык программирования Python из-за его простоты и обширной библиотеки для работы с аппаратным обеспечением. Библиотека RPi.GPIO была выбрана для работы с Raspberry Pi, которая использовалась в проекте для управления оборудованием.  Кроме того, я использовал следующие технологии и инструменты:  Tkinter: для создания графического интерфейса пользователя (GUI). Этот модуль предоставляет инструменты для создания интерактивных окон и кнопок. RPi.GPIO или gpiozero: для работы с Raspberry Pi, чтобы управлять портами ввода-вывода и датчиками уровня воды. SQLite: для хранения пользовательских настроек и режима работы ирригатора. PySerial: для связи с внешними устройствами, которые могли быть подключены к ирригатору (например, дополнительные датчики или устройства для анализа). |  |
| 21.11 | В процессе разработки программного обеспечения для контроля ирригатора, используемого в стоматологии, я столкнулся с различными ошибками и проблемами. Цель данного отчёта — проанализировать типы ошибок, возникшие в коде, методы их устранения, а также обобщить полученные уроки для будущих проектов.  Типы ошибок  Синтаксические ошибки  На начальном этапе разработки я столкнулся с несколькими синтаксическими ошибками, связанными с неправильным написанием функций и несоответствием скобок. Например, отсутствие закрывающей скобки в функциях приводило к сбоям при запуске программы.  Исправление: Обратил внимание на использование редакторов кода с подсветкой синтаксиса, что помогло быстро идентифицировать ошибки.  Логические ошибки  Логические ошибки имели место в алгоритме управления ирригатором. Например, неправильная обработка переменной, отвечающей за режим работы устройства, приводила к тому, что ирригатор начинал работать в неправильном режиме (например, вместо режима "Промывание" он устанавливался на "Массаж").  Исправление: Для выявления логических ошибок я использовал отладчик, который позволил поэтапно прослеживать выполнение программы и состояние переменных.  Ошибки ввода-вывода  Поскольку программа взаимодействует с аппаратным обеспечением, были случаи, когда данные не считывались с датчиков уровня воды. Это могло происходить из-за неправильной конфигурации GPIO-выводов на Raspberry Pi или из-за временных задержек в считывании. |  |
| 22.11 | Исправление: Я добавил задержки в код, а также реализовал повторное считывание данных в случае их отсутствия. Также изучил документацию к библиотекам для работы с GPIO, чтобы убедиться в правильной настройке.  Исключения и обработка ошибок  Обнаружил недостаточную обработку ошибок, что приводило к сбоевым ситуациям, когда пользователи, например, пытались запустить программу без подключения устройства.  Исправление: Я добавил блоки try-except для обработки потенциальных исключений, а также создал сообщения для пользователя, информирующие о необходимых действиях.  Учебные моменты  Тестирование: Я понял, что тестирование на ранней стадии разработки помогает избежать множества проблем в будущем. Я начал использовать модульные тесты для проверки работы отдельных функций. Документация: Важность документации стала очевидной, так как корректные комментарии помогали не только мне, но и другим членам команды (в последующих проектах) быстро ориентироваться в коде. Обратная связь: Регулярные обсуждения с наставником и коллегами по коду позволили выявить ошибки, которые я сам не заметил, и улучшить общую структуру и читаемость программы.  Заключение  Работа над проектом по разработке программы для управления ирригатором в стоматологии оказалась насыщенной и плодотворной. Анализ ошибок и их исправление не только способствовали улучшению текущего проекта, но и повышению моих навыков программирования. Эти уроки, полученные в ходе работы, я буду применять в будущих разработках, чтобы минимизировать количество ошибок и повысить качество итогового продукта. |  |
| 23.11  24.11 | ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙ ДЕНЬ |  |
| 25.11 | Создание программы для работы с пломбами в стоматологической практике может включать множество аспектов, таких как управление информацией о пациентах, записи о применяемых пломбах, время их установки, типы пломб и прочие сведения. Ниже приведён базовый пример на языке Python, который демонстрирует, как можно реализовать простую систему управления данными о пломбах:  python class Filling:  def init(self, filling\_type, date, patient\_name):  self.filling\_type = filling\_type  self.date = date  self.patient\_name = patient\_name  def str(self):  return f"Пломба: {self.filling\_type}, Установлена: {self.date}, Пациент: {self.patient\_name}"  class FillingManager: def init(self):  self.fillings = []  def add\_filling(self, filling):  self.fillings.append(filling)  def list\_fillings(self):  for filling in self.fillings:  print(filling)  def search\_fillings(self, patient\_name):  found\_fillings = [filling for filling in self.fillings  if filling.patient\_name == patient\_name]  return found\_fillings  def remove\_filling(self, date, patient\_name):  self.fillings = [filling for filling in self.fillings  if not (filling.date == date and filling.patient\_name == patient\_name)]  def main():  manager = FillingManager() while True: print("\n--- Меню ---")  print("1. Добавить пломбу")  print("2. Просмотреть пломбы")  print("3. Найти пломбы по пациенту")  print("4. Удалить пломбу")  print("5. Выход")  choice = input("Выберите действие (1-5): ")  if choice == '1': filling\_type = input("Введите тип пломбы: ")  date = input("Введите дату установки: ")  patient\_name = input("Введите имя пациента: ")  new\_filling = Filling(filling\_type, date, patient\_name) manager.add\_filling(new\_filling)  print("Пломба добавлена.")  elif choice == '2': print("\nСписок пломб:")  manager.list\_fillings()  elif choice == '3':  patient\_name = input("Введите имя пациента: ")  found\_fillings = manager.search\_fillings(patient\_name)  if found\_fillings:  print(f"\nПломбы пациента {patient\_name}:")  for filling in found\_fillings:  print(filling)  else: print("Пломбы не найдены.")  elif choice == '4':  date = input("Введите дату установки: ")  patient\_name = input("Введите имя пациента: ") manager.remove\_filling(date, patient\_name)  print("Пломба удалена, если она существовала.")  elif choice == '5':  print("Выход из программы.")  break else: print("Некорректный выбор. Пожалуйста, попробуйте еще раз.") if name == "main":  main()  Объяснение кода:  Классы:  Filling: Этот класс представляет отдельную пломбу с её типом, датой установки и именем пациента. Он содержит метод \_\_str\_\_, который позволяет удобно выводить информацию о пломбе. FillingManager: Этот класс управляет списком пломб. Он позволяет добавлять, просматривать, искать и удалять пломбы.  Интерфейс:  Простое текстовое меню, которое предлагает пользователю различные действия (добавление, просмотр, поиск и удаление пломб).  Главная функция:  main: В ней реализована логика работы с классами, обработка ввода пользователя и выполнение соответствующих действий.  Заключение  Данный код представляет собой простую систему для работы с информацией о пломбах. Его можно адаптировать и расширять, добавляя такие функции, как сохранение и загрузка данных из файлов, интеграция с базами данных, использование графического интерфейса пользователя и других возможностей для более интуитивного взаимодействия. |  |
| 26.11 | patient\_name = input("Введите имя пациента: ")  filling = Filling(filling\_type, date, patient\_name)  manager.add\_filling(filling)  print("Пломба успешно добавлена.")  elif choice == '2': print("\nСписок всех пломб:")  manager.list\_fillings()  elif choice == '3':  patient\_name = input("Введите имя пациента для поиска: ")  found\_fillings = manager.search\_fillings(patient\_name)  if found\_fillings:  print("\nНайденные пломбы:")  for filling in found\_fillings:  print(filling)  else: print("Пломбы не найдены для данного пациента.")  elif choice == '4':  date = input("Введите дату установки для удаления: ")  patient\_name = input("Введите имя пациента для удаления: ") manager.remove\_filling(date, patient\_name)  print("Пломба успешно удалена (если она существует).")  elif choice == '5':  print("Выход из программы.")  break else:  print("Некорректный выбор. Пожалуйста, попробуйте снова.")  if name == "main":  main()  Пояснение кода:  Класс Filling: Этот класс хранит информацию о пломбе, включая её тип, дату установки и имя пациента. Метод \_\_str\_\_ позволяет выводить эту информацию в удобочитаемом формате. Класс FillingManager: Этот класс управляет списком пломб. Он предоставляет методы для добавления, удаления и поиска пломб, а также для их отображения. Методы:  add\_filling: добавляет новый объект пломбы в список. list\_fillings: выводит информацию обо всех пломбах. search\_fillings: ищет пломбы по имени пациента и возвращает соответствующий список. remove\_filling: удаляет пломбу по дате и имени пациента.  Основная функция: В main() реализовано пользовательское меню, которое предоставляет возможность взаимодействия с системой. Пользователь может добавлять, просматривать, искать или удалять пломбы. |  |
| 27.11 | В функции main создается объект класса FillingManager, который будет управлять всеми операциями с пломбами. В бесконечном цикле (while True) пользователю предлагают выбрать одно из действий: добавить пломбу, просмотреть список пломб, найти пломбы по имени пациента, удалить пломбу или выйти из программы. В зависимости от выбора пользователя выполняются соответствующие действия.  Обработка ввода пользователя:  При выборе варианта "1" пользователю предлагают ввести тип пломбы, дату её установки и имя пациента. После этого создается объект Filling и добавляется в FillingManager. При выборе варианта "2" выводится полный список всех пломб. При выборе варианта "3" запрашивается имя пациента, затем программа ищет и выводит все пломбы, связанные с этим именем. При выборе варианта "4" пользователь вводит дату и имя пациента, после чего программа пытается удалить соответствующую пломбу. При выборе варианта "5" программа завершает работу, выходя из бесконечного цикла.  Как улучшить программное обеспечение  Постоянное хранение: Рассмотрите возможность добавления функции для сохранения пломб на диск, например, используя файлы (в формате CSV, JSON или другого) или базу данных. Валидация данных: Добавьте проверку вводимых данных (например, для проверки, что дата заполнена в правильном формате). Графический интерфейс: Рассмотрите возможность создания пользовательского интерфейса с использованием библиотек, таких как Tkinter или PyQt, для упрощения взаимодействия с пользователем. Расширение функционала: Рассмотрите возможность добавления функций, которые позволят редактировать информацию о пломбах, а также генерировать отчеты. Обработка ошибок: Улучшите обработку исключительных ситуаций, чтобы программа не завершалась аварийно, и предоставляла пользователю более понятные сообщения об ошибках. |  |
| 28.11 | Постоянное хранение:  Рассмотрите возможность добавления функции для сохранения пломб на диск, например, используя файлы (в формате CSV, JSON или другого) или базу данных (SQLite, PostgreSQL и т. д.). Это позволит сохранять данные между запусками программы и улучшит её функциональность. Для сериализации данных можно использовать стандартные библиотеки Python, такие как json или pickle.  Улучшенный интерфейс:  Вместо текстового интерфейса можно реализовать графический интерфейс пользователя (GUI) с использованием библиотек, таких как Tkinter или PyQt. Это сделает программу более интуитивно понятной и удобной для пользователей. Обеспечьте более детальное отображение информации и возможность редактирования данных, если это необходимо.  Валидация ввода пользователя:  Добавьте валидацию ввода для значений, таких как дата установки, имя пациента и тип пломбы. Это поможет предотвратить ошибки и введение некорректных данных. Например, можно использовать регулярные выражения для проверки формата ввода имени или даты.  Поиск и фильтрация:  Реализуйте дополнительные возможности поиска и фильтрации записей по различным критериям (например, по дате, типу пломбы и т. д.). Это сделает управление записями более удобным. Вы можете использовать модули, такие как pandas, для более продвинутого анализа и работы с данными.  Улучшение структуры кода:  Рассмотрите возможность применения принципов ООП (Объектно-Ориентированного Программирования) для создания более модульного и расширяемого кода. Например, вы можете создать классы для различных типов пломб или пациентов.  Логирование и отладка:  Добавьте возможность ведения журнала событий (логирования), чтобы отслеживать все действия в программе. Это может помочь при поиске ошибок и улучшении функции обслуживания. Использование библиотеки logging позволит хранить информацию о выполнении программы, а также об ошибках и предупреждениях.  Тестирование:  Разработайте модульные тесты для проверки функций программы. Это обеспечит надежность и упростит дальнейшую разработку. Используйте библиотеки, такие как unittest или pytest, для автоматического тестирования кода.  Заключение  Создание программы для управления пломбами — это отличная возможность для практики программирования и разработки программного обеспечения. С помощью вышеописанных улучшений, ваша программа может стать более функциональной, удобной и надежной. Недостаточные функции могут быть улучшены по мере необходимости, и со временем программа станет более мощным инструментом для ведения учета пломб. |  |
| 29.11 | Логирование и отладка (продолжение):  Используйте различные уровни логирования (например, DEBUG, INFO, WARNING, ERROR), чтобы упростить мониторинг работы программы. Это позволит пользователям и разработчикам видеть важную информацию и предупреждения, не загромождая вывод менее значимой информации. Рассмотрите возможность добавления функции сохранения логов в файл, чтобы они были доступны для последующего анализа.  Обратная связь и поддержка пользователей:  Внедрите систему обратной связи для пользователей, чтобы они могли легко сообщать о проблемах или предлагать улучшения. Это может быть формой обратной связи в приложении или специальным адресом электронной почты. Обратите внимание на отзывы пользователей и периодически обновляйте приложение на основе их предложений. Это поможет улучшить взаимодействие и удовлетворенность пользователей.  Документация и обучающие материалы:  Разработайте полную документацию, включая инструкции по установке, началу работы и возможных сценариев использования. Это поможет новым пользователям быстрее освоить программу. Подумайте о создании видеоруководств или пошаговых инструкций, чтобы визуально продемонстрировать основные функции и возможности программы.  Тестирование и отладка:  Введите тестирование для вашего кода, чтобы гарантировать его стабильность и правильную работу. Используйте фреймы тестирования, такие как unittest или pytest, для проверки различных компонентов программы. Проводите систематические тесты перед выпуском использованных функциональных улучшений, чтобы избежать возникновения новых ошибок.  Производительность:  Проверьте, как ваша программа работает с большими объемами данных, и оптимизируйте бесконечные циклы или повторяющиеся вычисления. Используйте профилирование (например, с помощью библиотеки cProfile), чтобы выявить узкие места в производительности. Оптимизируйте структуру данных, чтобы повысить эффективность поиска и обработки информации.  Обновления и расширения функциональности:  Регулярно обновляйте программу, исправляя обнаруженные ошибки и добавляя новые функции. Это не только повысит стабильность вашей программы, но и добавит новые возможности для пользователей. Рассмотрите возможность реализации функционала, который предоставляет пользователю возможность гибко настраивать интерфейс или функциональность программы под собственные требования. |  |
| 30.11  01.12 | ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙ ДЕНЬ |  |
| 02.12 | Безопасность:  Обеспечьте безопасность ваших данных, добавив механизмы аутентификации и авторизации для пользователей, если программа хранит конфиденциальные данные. При работе с данными пользователей убедитесь, что соблюдаются требования законодательства (например, GDPR для европейских пользователей).  Интеграция с другими сервисами:  Подумайте о возможности интеграции с другими программами или API, чтобы расширить функциональность вашей программы. Например, может быть полезно интегрировать программу с системами управления данными пациентов или стоматологическими CRM. Создание API позволит другим разработчикам интегрировать вашу программу в свои решения.  Кроссплатформенность:  Если ваша программа изначально ориентирована на одну платформу, рассмотрите возможность ее адаптации для других ОС (Windows, macOS, Linux). Это откроет доступ к большему числу пользователей. Используйте кроссплатформенные фреймы и библиотеки для обеспечения совместимости.  Международализация:  Если ваша программа может использоваться пользователями из разных стран, добавьте возможности для локализации. Программа должна уметь поддерживать несколько языков, учитывая различия в форматах даты и времени, валюты и других культурных особенностей.  Эти рекомендации помогут сделать вашу программу более функциональной, удобной и уверенной в использовании. Успех зависит от того, насколько программа соответствует потребностям пользователей, и насколько легко вы сможете вносить в неё изменения по мере появления новых требований и технологий. Каждое из предложений может существенно повысить общую ценность вашего программного обеспечения и удовлетворенность пользователей. |  |
| 03.12 | Производительность (продолжение):  Оптимизируйте структуру данных, используя более эффективные типы данных для хранения информации. Например, если вам нужно хранить уникальные значения, рассмотрите возможность использования множества (set) вместо списка. Используйте асинхронное программирование, если ваша программа выполняет длительные операции ввода-вывода, например, сетевые запросы или операции с файловой системой. Это поможет обеспечить отзывчивость интерфейса и снизить время ожидания для пользователей.  Безопасность:  Обратите внимание на аспекты безопасности, особенно если ваша программа обрабатывает конфиденциальные данные. Используйте шифрование для защиты данных и следите за уязвимостями в используемых библиотеках. Реализуйте аутентификацию и авторизацию, если ваша программа включает многопользовательский доступ или личные учетные записи. Это поможет предотвратить несанкционированный доступ к данным.  Кроссплатформенность:  Если это применимо, протестируйте вашу программу на разных операционных системах (Windows, macOS, Linux), чтобы убедиться, что она работает корректно везде. Рассмотрите использование кроссплатформенных инструментов, если требуется. Создайте инструкции по установке для различных систем, чтобы пользователям было удобно следовать им независимо от используемой платформы. |  |
| 04.12 | Обновления и поддержка:  Регулярно обновляйте ваше программное обеспечение, добавляя новые функции, исправления и улучшая производительность. Уведомляйте пользователей о доступных обновлениях и об их ходе. Организуйте службу поддержки, чтобы пользователи могли получить помощь по возникающим вопросам или проблемам. Это может быть через форумы, электронную почту или систему тикетов.  Интеграция с другими сервисами:  Подумайте о добавлении возможностей интеграции с другими сервисами и приложениями, которые могут быть полезны вашим пользователям. Это может быть API для взаимодействия с другими программами или возможность импорта/экспорта данных. Напишите инструкции по интеграции, чтобы пользователи могли легко использовать эти функции.  Использование методологий разработки:  Подумайте о внедрении методологий Agile или Scrum, чтобы улучшить процесс разработки и обеспечить более гибкий подход к изменениям и улучшениям. Периодически пересматривайте и планируйте задачи, основываясь на полученной обратной связи и изменениях в приоритетах проекта.  Тестирование пользовательского интерфейса:  Проведите пользовательское тестирование, чтобы определить, насколько интуитивен интерфейс вашей программы. Запросите отзывы и внесите изменения на основе комментариев пользователей. Убедитесь, что все элементы интерфейса понятны и легко доступны. Следуйте принципам дизайна, чтобы создать удобный и доступный интерфейс.  Заключение  Улучшение программного обеспечения — это непрерывный процесс, который требует постоянного внимания и адаптации к изменениям в требованиях пользователей и технологиях. Следуя перечисленным рекомендациям, вы сможете создать более стабильное, безопасное и удобное в использовании приложение, что, в свою очередь, повысит его ценность для пользователей и увеличит их удовлетворенность. |  |
| 05.12 | Работа с пломбами - одна из самых важных процедур в стоматологии, так как пломбы помогают восстановить поврежденные зубы и предотвратить дальнейшее разрушение зубной эмали.  Для улучшения качества работы и повышения эффективности процесса я начал использовать специальные программы для моделирования пломб. Благодаря этим программам я могу создавать точные 3D модели пломб, учитывая все особенности каждого пациента и каждого зуба.  Также программирование позволяет мне быстро рассчитывать необходимое количество материала для пломбирования, что помогает сэкономить время и ресурсы. Я могу оптимизировать процесс работы, улучшить точность и качество работы, что в конечном итоге положительно сказывается на здоровье пациентов.  Работа с пломбами с использованием программирования  - это не только удобный и эффективный инструмент, но и возможность постоянно совершенствоваться в своей профессии и предоставлять пациентам качественное лечение. |  |
| 06.12 | Провел анализ ошибок, которые часто возникают при их установке. Наиболее распространенные проблемы - неправильная подготовка полости зуба перед установкой пломбы и недостаточное плотное прилегание материала к стенкам полости. Начал исправлять ошибки, изучив литературу и проконсультировавшись с опытными коллегами.  Продолжил работу над исправлением ошибок в установке пломб. Внимательно изучил технику правильной подготовки и установки пломбы, провел несколько тренировочных упражнений на специальных моделях зубов. Ощутил уверенность в своих навыках и готов приступить к закреплению сделанной работы.  Сегодня закрепил свою работу по установке пломб. Провел дополнительные проверки и корректировки, чтобы убедиться, что пломба надежно закрывает полость зуба и не оставляет никаких пробелов. Радует успех - все ошибки исправлены, работа выполнена качественно.  Провел анализ своей работы и проработал план дальнейших действий. Понял, что для достижения высоких результатов в установке пломб необходимо постоянное обучение и практика. Запланировал провести дополнительные тренировочные сессии и пройти курсы повышения квалификации.  Получил ценный опыт и знания, которые помогут мне стать более профессиональным специалистом в области программирования. Готов продолжать развиваться и совершенствовать свои навыки |  |
| 07.12  08.12 | ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙ ДЕНЬ |  |
| 09.12 | В рамках моей работы программиста в стоматологической клинике, я занимаюсь созданием и поддержкой специализированного программного обеспечения для работы с пломбами. Это важное направление в стоматологии, которое требует точности и высокого качества работы. В процессе моей работы я столкнулся с различными ошибками и проблемами, которые требовали моего внимания и усилий для их исправления.  Анализ ошибок: Одной из основных проблем, с которой я столкнулся, были ошибки при заполнении данных о пациентах и проведенных процедурах. Это приводило к ошибкам в документации и затрудняло последующую работу стоматологов. Я также обнаружил ошибки в алгоритмах расчета объема материала для пломбирования, что приводило к неправильным результатам и дополнительным расходам для клиники.  Исправление ошибок: Для исправления выявленных ошибок я провёл дополнительное тестирование программного обеспечения и внес необходимые коррективы в код. Я также провел обучение сотрудников клиники по правильному заполнению данных и использованию программы. Это позволило уменьшить количество ошибок и повысить эффективность работы всей клиники.  Закрепление сделанной работы: Для закрепления сделанных изменений я провел обучающие семинары для сотрудников клиники по использованию обновленного программного обеспечения. Я также создал подробное руководство по работе с программой, которое было распечатано и выдано каждому сотруднику. Это позволило всем сотрудникам клиники быть в курсе всех изменений и использовать программу более эффективно.  Итог: В результате моей работы по исправлению ошибок и улучшению программного обеспечения для работы с пломбами, удалось улучшить работу стоматологической клиники и уменьшить количество ошибок. Это позволило клиентам получать более качественное и точное лечение, а сотрудникам клиники работать более эффективно и без лишних сложностей. |  |
| 10.12 | Сегодня я продолжил работу над пломбами, а также приступил к закреплению сделанной работы. Для этого были использованы специальные материалы и инструменты, чтобы обеспечить надежное крепление и защиту зубов от повторного повреждения. Также провел анализ сделанной работы, чтобы убедиться в ее качестве и долговечности.  Я провел осмотр и анализ своей работы. Было обнаружено несколько небольших ошибок, которые я немедленно исправил. Это позволило улучшить качество и долговечность пломб, а также повысить уровень моего профессионализма. Я уделил особое внимание деталям и заботе о здоровье пациентов, что помогло достигнуть отличных результатов.  Я завершил работу над пломбами в рамках специальности программирование в стоматологии. Я провел окончательный осмотр и анализ сделанной работы, убедившись в ее качестве и безупречности. Благодаря своему усердию и профессионализму, я достиг высокого уровня навыков и умений в данной области. Я готов продолжить развиваться и совершенствоваться, чтобы стать настоящим экспертом в работе с пломбами в стоматологии. |  |
| 11.12 | Сегодня я провел анализ текущих процессов в клинике и выявил несколько узких мест, которые можно оптимизировать с помощью программного обеспечения. Поделился своими идеями с коллегами и вместе мы начали разрабатывать план действий.  Я приступил к разработке программного обеспечения, которое поможет улучшить процессы приема и записи пациентов. У меня было небольшое затруднение с одним из блоков кода, но благодаря помощи коллег я смог его успешно решить.  Сегодня я провел обучение сотрудников клиники работе с новым программным обеспечением. Все прошло успешно, и теперь мы можем приступить к его использованию в повседневной деятельности.  Для успешного выполнения поставленных задач я принимал участие в обучающих курсах и семинарах по программированию и стоматологии. Это позволило мне углубить знания в обоих областях и обеспечить эффективное взаимодействие с коллегами.  Разработка программного обеспечения  Моя основная задача заключалась в разработке программы для учета пациентов, записи на приемы, контроля за лечением и хранения медицинских данных. В процессе работы я активно взаимодействовал с врачами и медперсоналом клиники, чтобы учесть все их потребности и требования.  Тестирование и внедрение  После завершения разработки программы я активно участвовал в ее тестировании и внедрении. Благодаря этому удалось выявить и устранить возможные ошибки и недоработки, а также обучить персонал клиники работе с новым ПО.  Оценка результатов  Результатом моей работы стало значительное повышение эффективности работы стоматологической клиники и улучшение качества обслуживания пациентов. Коллектив оценил мои усилия и вклад в достижение общей цели. |  |
| 12.12 | Заключение  Моя активная роль в работе коллектива по направлению программирование в стоматологии не только позволила мне расширить знания и навыки в этой области, но и принести значительную пользу клинике и ее пациентам. Я горжусь своей работой и готов продолжать развиваться и совершенствовать свои навыки в этой интересной и перспективной сфере.  Я начал работу над проектом программирования в стоматологии. Мы обсудили с коллегами основные задачи, которые нужно выполнить, и распределили ответственности между членами команды. Я взял на себя разработку алгоритма для управления стоматологическим оборудованием через программное обеспечение.  Продолжил свою работу над проектом. Сегодня прошел встречу с врачами и специалистами по стоматологии, чтобы уточнить требования к программе. Было очень интересно общаться с профессионалами из другой отрасли и понять, какие функциональности им действительно нужны для удобной работы.  Провели рабочее совещание с командой, чтобы обсудить промежуточные результаты работы. Вместе мы выявили несколько проблемных моментов и нашли пути их решения. Было приятно видеть, как все члены команды активно включаются в обсуждение и предлагают свои идеи.  Я выступил на конференции по программированию в стоматологии с презентацией нашего проекта. Получил много полезных отзывов и предложений от коллег из других компаний, что поможет нам улучшить нашу программу.  Завершил работу над проектом и представил его врачам и специалистам по стоматологии. Получил много положительных отзывов и благодарностей за проделанную работу. Был рад видеть, что наша программа действительно поможет им улучшить качество оказания медицинских услуг.  Общий вывод: Активное участие в работе коллектива позволило нам успешно реализовать проект по программированию в стоматологии. Благодаря нашей взаимодействию и обмену идеями, мы смогли создать программу, которая действительно пригодится врачам и поможет им улучшить качество своей работы. |  |
| 13.12 | Сегодня я хочу поделиться своими мыслями и эмоциями о том, как прошла моя практика в области программирования в стоматологии. Это было невероятное время, полное увлекательных задач и интересных проектов. Опыт работы в этой сфере оказался не только увлекательным, но и очень полезным для моего профессионального роста.  Работа в коллективе:  Я хочу поблагодарить своих коллег за поддержку и помощь во время моей практики. Благодаря им я смог освоить новые навыки программирования, изучить специфику работы в стоматологии и применить свои знания на практике. С каждым из них было приятно работать, и я надеюсь на дальнейшее сотрудничество в будущем.  Опыт и достижения:  За время работы в области программирования в стоматологии я смог реализовать несколько уникальных проектов, которые помогли оптимизировать рабочий процесс стоматологической клиники. Благодаря этому опыту я улучшил свои навыки программирования и получил ценный опыт работы в команде.  Прощание с коллективом:  Сегодня для меня настало время прощания с коллективом, который стал для меня второй семьей. Я буду скучать по каждому из вас и надеюсь на дальнейшее взаимодействие. Благодаря вам я стал лучше, и я никогда не забуду это время, проведенное в вашей компании.  Заключение:  Моя практика в области программирования в стоматологии была незабываемым опытом, который я никогда не забуду. Я благодарен всем за поддержку, опыт и ценные уроки, которые я получил за это время. Я уверен, что этот опыт поможет мне дальше развиваться и расти как специалист. Спасибо всем за это незабываемое время! |  |
| 14.12 | В рамках своего профессионального роста, я принял участие в реализации проекта по внедрению программного обеспечения в стоматологическую клинику. Моя задача заключалась в разработке и адаптации программного продукта для улучшения эффективности работы стоматологов и улучшения качества обслуживания пациентов.  Я начал разработку программного обеспечения с учетом специфики стоматологической практики. Я провел анализ потребностей клиники, выявил основные задачи, которые необходимо автоматизировать, и начал создание прототипа программы.  Продолжил работу над программой. Определены основные функциональные возможности, добавлены новые модули для управления записями пациентов, ведения учета услуг и финансовой отчетности. Программа начала принимать окончательный вид.  Проведено тестирование программы на работе с реальными пациентами. Выявлены некоторые недочеты, которые были исправлены в кратчайшие сроки. Пользователи оценили удобство и функционал программы.  Заключение: Работа над проектом по программированию в стоматологии была для меня интересным и познавательным опытом. Я убедился в эффективности использования современных технологий в медицине и убедился, что программное обеспечение способно значительно упростить рабочий процесс в стоматологической клинике. Подготовив дневник отчета, я чувствую, что завершаю этот этап своей профессиональной деятельности и готов к новым вызовам в области программирования. |  |

## Подпись руководителя и печать профильной

**организации**

**(ставится в последний день практики)**