Здравствуйте, уважаемые председатель и члены государственной аттестационной комиссии.

**Слайд 1.** **Титульный лист**.

Вашему вниманию представляется работа на тему «Разработка мобильного приложения для просмотра расписания МИЭТ»

Целью работы является ускорение просмотра расписания студенческих групп и преподавателей с возможностью поиска окна для переноса занятия.

Для достижения данной цели были поставлены задачи, которые перечислены на слайде.

**Слайд 2 Исследование предметной области.**

Интенсивный ритм жизни современного студента и использование в методиках обучения коллективных проектов предполагает не только тесное взаимодействие с одногруппниками, но и постоянный контакт с преподавателями. Поэтому разработка приложения, обеспечивающего оперативный доступ к расписанию МИЭТ, весьма актуально.

**Слайд 3. Анализ аналогичных программных решений.**

На данный момент не существует работающих приложений, решающих данную проблему.

Доступ к расписанию через сайт miet.ru не является оперативным, а также не дает возможности просмотреть свое расписание без доступа в интернет и расписание преподавателя.

Приложение «Orioks», как и «SKED» на данный момент не работают.

«Кампус» работает с расписанием различных вузов, но МИЭТа в этом списке нет.

Приложение «СтудЖурнал» позволяет составить расписание вручную, что занимает много времени, а также в приложении нет доступа к расписанию преподавателей и других групп.

Так как ни одно из существующих решений полностью не удовлетворяет описанным требованиям, то актуальной является задача создания нового мобильного приложения – МП ПР

**Слайд 4. Выбор языка программирования.**

Рассматривалось пять языков для разработки мобильного приложения на Android: С++, С#, Python, Java и Kotlin.

В результате сравнения языков программирования по представленным на слайде критериям для разработки МП ПР выбран язык программирования Kotlin, так как в нем присутствует транслятор под платформу Android и существует декларативный фреймворк Jetpack Compose.

**Слайд 5. Выбор среды программирования**.

Для языка Kotlin существует шесть сред разработки: IntelliJ IDEA, Eclipse, VS Code, Atom, Vim и Android Studio.

В результате сравнения, выбрана интегрированная среда разработки Android Studio, которая является официальной и рекомендованной средой разработки под операционную систему Android, так как в ней, в отличие от остальных, имеется эмулятор для запуска Android-приложения и инструменты для работы с UI.

**Слайд 6.** **Схема данных**.

В течение работы приложения происходит постоянное взаимодействие пользователя с базой данных.

На схеме видно, что состояния системы при взаимодействии с пользователем это в основном демонстрация окон с необходимым ему расписанием. Также при выборе окна задач пользователю дается возможность добавить задачу.

**Слайд 7. Схема алгоритма МП ПР.**

На слайде представлена схема алгоритма работы приложения.

Перед началом работы необходимо авторизоваться через систему ОРИОКС. Далее пользователь может взаимодействовать с функционалом приложения, просматривая свое расписание на сегодня и другие дни. При нажатии на кнопку «Поиск» появится окно для поиска расписания групп, преподавателей и поиска окон для переноса занятия, где при вводе номеров групп, фамилии преподавателя, приложение отправит запрос в базу данных и представит необходимые данные пользователю. При нажатии кнопки «Задачи» пользователь может просмотреть существующие задачи, а также добавить новую, указав ее название, текст, предмет и срок выполнения. Чтобы вернуться к просмотру своего расписания пользователю необходимо нажать кнопку «Расписание».

**Слайд 8. Экранная форма пользовательского интерфейса.**

В приложении три основных окна:

Расписание, на котором отображается расписание пользователя.

Поиск для поиска расписания студенческих групп, преподавателей и окна для переноса занятия. На слайде показан поиск окна для трех групп и преподавателя. Пары, выделенные зеленым, являются доступными для переноса занятия.

И окно задач для добавления и просмотра существующих задач.

**Слайд 9. Отладка и тестирование.**

Отладка выполнялась встроенными средствами Android Studio. На слайде показан пример использования точки остановы. Тестирование проводилось методом черного ящика. Тестировались авторизация, правильность отображение расписания и задач, а также работа алгоритма поиска. Результат выполнения этих тестов вы можете видеть на слайде.

**Слайд 10. Апробация и публикация**

**Слайд 9.** Результаты работы представлены на слайде. Доклад закончен. Спасибо за внимание!

**Благ. слово:** Хотелось бы поблагодарить директора института СПИНТех, Гагарину Ларису Геннадьевну. Также хотелось бы поблагодарить Федотову Елену Леонидовну и Федорова Алексея Роальдовича за помощь с подготовкой выпускной квалификационной работы

**Программа** - данные, предназначенные для управления конкретными компонентами системы обработки информации в целях реализации определенного алгоритма.

**Программное обеспечение** - совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ.

**Программный модуль** - программа или функционально завершенный фрагмент программы, предназначенный для хранения, трансляции, объединения с другими программными модулями и загрузки в оперативную память.

**Программное средство** - объект, состоящий из программ, процедур, правил, а также, если предусмотрено, сопутствующих им документации и данных, относящихся к функционированию системы обработки информации.

**Программный продукт** - программное средство, предназначенное для поставки, передачи, продажипользователю.

**Программное изделие** - изделие межотраслевого применения вида "программной продукции", прошедшее испытание, имеющее соответствующий комплект программных документов и готовое к серийному производству.

**Черный ящик** – метод тестирования функционального поведения объекта (программы, системы) с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве (коде) тестируемого объекта

**Белый ящик** – тестирование, которое учитывает внутренние механизмы системы или компонента.

**Концептуальная модель** – модель, в которой указываются сущности и связи с точки зрения смысловой нагрузки рассматриваемой предметной области.

**Автоматический** - без участия человека, **автоматизированный** - с его участием.

**Алгоритм** - совокупность точно заданных правил решения некоторого класса задач или набор инструкций, описывающих порядок действий исполнителя для решения определённой задачи.

**Нагрузочное тестирование** - это автоматизированное тестирование, имитирующее работу определенного количества пользователей на каком-либо общем ресурсе.

Тестирование использования ресурсов.

**Отладка** - обнаружение, локализация и устранение ошибок в программе.

**Тестирование** - процесс испытания программного продукта, цель которого проверка соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов.