

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ДГТУ)

Факультет Информатика и вычислительная техника

(наименование факультета)

Кафедра Программное обеспечение вычислительной техники и автоматизированных систем

(наименование кафедры)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | «ПОВТиАС» |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | В.В. Долгов |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. | |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине Программирование мобильных устройств

(наименование учебной дисциплины (модуля))

на тему: Мобильное приложение «Управление строительными проектами»

Автор проекта (работы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) И.О.Ф.

Направление/специальность, профиль/специализация:

09.03.04 Программная инженерия

код направления наименование направления (специальности)

наименование профиля (специализации)

Обозначение курсовой работы КР.530000.000 Группа ВПР41

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Н. Литвинов\_\_\_\_\_\_\_

подпись (должность, И.О.Ф.)

Проект (работа) защищен(а) \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата оценка подпись

Ростов-на-Дону  
2023



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ДГТУ)

Факультет Информатика и вычислительная техника

(наименование факультета)

Кафедра Программное обеспечение вычислительной техники и автоматизированных систем

(наименование кафедры)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | «ПОВТиАС» |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | В.В. Долгов |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. | |

ЗАДАНИЕ

на курсовой проект (работу)

Студент Абдуллах Ахмед Код 09.03.04 Группа ВПР41

тему: Мобильное приложение «Управление строительными проектами»

Срок представления проекта (работы) к защите «\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_ г.

Исходные данные для курсовой работы

1. Задание к выполнению курсовой работы

2. Основная литература: лекционный материал по дисциплине «Программирование мобильных устройств».

Содержание пояснительной записки

ВВЕДЕНИЕ:

В разделе «Введение» рассматривается актуальность приложений для аквариумистов.

Наименование и содержание разделов:

1. В разделе «Аналитический обзор» на основании сравнения существующих аналогов приведен обзор предметной области поставленной задачи, а также сформулировано ее условие.

2. В разделе «Алгоритмическое конструирование» рассматриваются алгоритм проверки входа в приложение и алгоритм добавления заметки в календарь.

3. В разделе «Программное конструирование» приводится обоснование выбора языка и среды программирования для разработки приложения «Управление строительными проектами». Также описаны основные классы, используемые для реализации программного средства.

4. В разделе «Тестирование программного средства» произведено тестирование приложения «Управление строительными проектами».

ЗАКЛЮЧЕНИЕ:

Выполнены все задачи для реализации приложения «Управление строительными проектами», создано рабочее программное средство.

Руководитель проекта (работы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. Н. Литвинов

подпись, дата

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_ А. Абдуллах

подпись, дата

Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

4

КР.530000.000

Разраб.

Абдуллах.А

Провер.

Литвинов В.Н.

.

*Мобильное приложение «Управление строительными проектами»*

Лист.

Листов

141

ДГТУ

Кафедра «ПОВТиАС»

[Введение 5](#_Toc130297564)

[1.1 Описание содержания приложения «Управление строительными проектами» 6](#_Toc130297565)

[1.2 Обзор существующих аналогов 6](#_Toc130297566)

[1.2.1 Обзор программного средства «Procore» 6](#_Toc130297567)

[1.2.2 Обзор программного средства «PlanGrid» 7](#_Toc130297568)

[1.3 Постановка задачи 7](#_Toc130297569)

[1.4 Выводы по главе 8](#_Toc130297570)

[2.1 Алгоритм проверки входа в приложение 9](#_Toc130297571)

[2.2 Алгоритм добавления новый проект 10](#_Toc130297572)

[2.3 Выводы по главе 11](#_Toc130297573)

[3 Программное конструирование 12](#_Toc130297574)

[3.1 Обоснование выбора средств разработки для реализации мобильного приложения «Управление строительными проектами» 12](#_Toc130297575)

[3.2 Объектно-ориентированная модель программного средства 12](#_Toc130297576)

[3.3 Выбор среды программирования 13](#_Toc130297577)

[3.4 Описание основных используемых классов 13](#_Toc130297578)

[3.5 Описание структуры базы данных 14](#_Toc130297579)

[3.6 Выводы по главе 14](#_Toc130297580)

[4 Тестирование программного средства 15](#_Toc130297581)

[4.1 Описание процесса тестирования 15](#_Toc130297582)

[4.2 Тестирования Приложение Управление строительными проектами 16](#_Toc130297583)

[4.2 Выводы по главе 20](#_Toc130297584)

[Заключение 21](#_Toc130297585)

[Перечень использованных информационных ресурсов 22](#_Toc130297586)

[Приложение А 23](#_Toc130297587)

[ПРИЛОЖЕНИЕ Б. UML-ДИАГРАММА КЛАССОВ. 28](#_Toc130297588)

[Приложение Г 33](#_Toc130297589)

Введение

Управление строительными проектами - это сложный процесс, который требует многих ресурсов и постоянной координации между различными участниками проекта. Современные мобильные приложения для Android помогают строительным компаниям оптимизировать управление проектами, улучшить эффективность работы и сократить время на выполнение задач.s

Мобильные приложения для Android могут использоваться для различных задач в управлении строительными проектами. Они позволяют отслеживать прогресс строительства, планировать задачи, управлять расписанием и бюджетом, контролировать качество работ и многое другое.

Одна из важных функций мобильных приложений для управления строительными проектами - это возможность создавать задачи и расписания, назначать ответственных за их выполнение и отслеживать прогресс выполнения. Это помогает командам быстро реагировать на изменения и оптимизировать рабочий процесс.

Мобильные приложения также позволяют управлять бюджетом и расписанием проекта. С их помощью можно создавать бюджетные планы, контролировать затраты, устанавливать сроки и отслеживать прогресс выполнения задач. Это позволяет компаниям более эффективно использовать ресурсы и сократить расходы.

Еще одна важная функция мобильных приложений для управления строительными проектами - это возможность контроля качества работ. Они позволяют отслеживать выполнение задач и проверять соответствие качества работ требованиям заказчика.

1 Аналитический обзор предметной области

В данной главе рассматриваются основные стандарты и принципы написания мобильного приложения «Управление строительными проектами»

* 1. Описание содержания приложения «Управление строительными проектами»

Домашняя страница, клиенты, трейдеры, проекты, работники, расходы, доходы, добавление новых проектов, заработная плата, покупки, счета, добавление новых счетов Приложение также содержит страницу входа в приложение, которая является стартовой страницей и позволяет пользователю зарегистрироваться или войти в их учетную запись.

* 1. Обзор существующих аналогов

приложение Bitx имеет много преимуществ из других приложений, которые он предназначен для небольших проектов, а не для больших, и это упрощает использование для обычного пользователя, вам не нужно иметь много времени на управление проектами, чтобы использовать его

* + 1. Обзор программного средства «Procore»

Одним из популярных приложений для управления строительными проектами является Procore. Это облачное приложение позволяет в режиме реального времени управлять проектами, составлять бюджет, отслеживать задачи, управлять документами и общаться. Он также предоставляет инструменты для управления подрядчиками и владельцами. С Procore строительные бригады могут сотрудничать более эффективно и результативно, гарантируя, что проекты будут завершены вовремя и в рамках бюджета.

* + 1. Обзор программного средства «PlanGrid»

Еще одно популярное приложение для управления строительными проектами — PlanGrid. Это приложение для управления строительными чертежами позволяет обмениваться и обновлять планы, документы и заметки между строительными бригадами в режиме реального времени. Это гарантирует, что каждый в команде имеет доступ к самой последней информации, уменьшая количество ошибок и задержек.

* 1. Постановка задачи

Когда мы говорим о приложении для управления строительными проектами, мы представляем инструмент, который позволяет строительной компании управлять всеми своими проектами и контрагентами в одном месте. Наша цель - создать приложение, которое позволит пользователям эффективно управлять своими проектами и сократить время и затраты на управление проектами.

Наше приложение будет состоять из нескольких страниц:

* Домашняя страница: это страница, на которую приходится иметь дело с информацией о своих проектах, доходах и расходах.
* Клиенты: эта страница содержит список всех клиентов, которые работают с компанией. Здесь можно добавлять новых клиентов, зачастую их данные и их историю.
* Трейдеры: эта страница содержит список всех трейдеров, с которыми постоянно работает компания. Сюда можно добавить новых трейдеров, как правило, их данные и их историю.
* Проекты: на этой странице пользователи могут управлять своими проектами, включать новые проекты, составлять список всех проектов, значительно сообщать о проектах, добавлять и расходовать, а также управлять выполнением проектов со счетами.
* Работники: на этой странице пользователи могут управлять своими работниками, добавлять новых работников, управлять их данными и управлять их заработной платой.
* Расходы и настройки: на этих приложениях добавляются новые счета, устанавливаются наборы и расходные компании и управляются своими банковскими счетами.
* Добавление новых проектов: на этой странице приложения создаются новые проекты и добавляется информация о них.
* Заработная плата: на этих пользователях
* Покупки: на этой странице пользователи могут покупать все покупки, сделанные компанией, и управлять своими закупками.
* Счета: на этой странице пользователи могут управлять своими банковскими счетами и добавлять новые счета.
  1. **Выводы по главе**

В данном обзоре был рассмотрен обзор приложений для управления строительными проектами, а также функциональных возможностей, которые имели место быть.

Выяснилось, что приложение должно состояться из нескольких страниц, как домашняя страница, страницы клиентов, трейдеров, проектов, работников, расходов, доходов, добавлений, управления новыми проектами с оплатой, покупками и счетами. Также была предложена страница входа, которая является стартовой страницей и позволяет пользователям зарегистрироваться или войти в свою учетную запись.

Основная цель приложения - выбрать компанию, занимающуюся строительством, инструмент для оптимизации управления проектами и сокращением времени и затрат на управление проектами.

**2 Алгоритмическое конструирование**

В данном разделе рассматриваются основные алгоритмы работы мобильного приложения «Управление строительными проектами»: алгоритм проверки входа в приложение и алгоритм добавления новый проект. Приведено их описание и схема работы.

**2.1 Алгоритм проверки входа в приложение**

Алгоритм проверки входа в приложение с использованием учетных записей Google включает следующие шаги:

1. Пользователь открывает приложение и выбирает опцию "Вход через Google".
2. Приложение отправляет запрос на проверку подлинности пользовательских учетных данных в систему Google.
3. Система Google проверяет, соответствуют ли предоставленные учетные данные сохраненным в ее базе данных записям.
4. Если учетные данные действительны, система Google создает токен авторизации и отправляет его обратно в приложение.
5. Приложение получает токен авторизации и использует его для получения доступа к пользовательской учетной записи Google и связанным с ней данным, таким как адрес электронной почты и фотография профиля.
6. Пользователь получает доступ к приложению и его функциональным возможностям.
7. В случае неверных учетных данных или других проблем с проверкой подлинности, система Google сообщает об этом приложению, и пользователю предлагается повторить процесс входа или использовать другой метод аутентификации.

**2.2 Алгоритм добавления новый проект**

1. Пользователь нажимает на кнопку "Добавить новый проект" на странице "Мои проекты".
2. Приложение перенаправляет пользователя на страницу "Новый проект", где пользователю предлагается ввести название проекта и выбрать клиента из списка уже существующих клиентов, которых пользователь добавил ранее.
3. Если пользователь не добавил клиента ранее, приложение перенаправляет пользователя на страницу "Новый клиент", где пользователю предлагается ввести данные о клиенте, такие как имя, адрес, контактную информацию и т.д. После того как пользователь вводит данные о клиенте, приложение сохраняет их в базе данных клиентов.
4. Пользователь вводит название проекта и выбирает клиента из списка уже существующих клиентов. Если клиента нет в списке, пользователь может добавить нового клиента, следуя шагам 2-3.
5. После того как пользователь вводит данные о проекте и клиенте, приложение сохраняет их в базе данных проектов.
6. Приложение перенаправляет пользователя на страницу "Мои проекты", где новый проект отображается в списке проектов.

**2.3 Выводы по главе**

В данной главе были представлены два алгоритма, которые значительно упрощают взаимодействие пользователя с приложением.

Первый алгоритм - это алгоритм проверки входа в приложение, который позволяет пользователям авторизоваться с использованием своего Google-аккаунта. Это удобный и безопасный способ входа в приложение, который избавляет пользователя от необходимости запоминать и вводить свой логин и пароль для каждого приложения. Данный алгоритм предусматривает проверку подлинности данных пользователя и в случае несоответствия, пользователю будет предложено повторить попытку входа.

Второй алгоритм - это алгоритм добавления нового проекта на странице "Мои проекты". Этот алгоритм предоставляет пользователю возможность добавить новый проект, выбрать уже существующего клиента из базы данных и указать название проекта. Если пользователь вводит некорректные данные, то приложение информирует его об ошибке и предлагает исправить ее. При успешном добавлении проекта, он будет сохранен в базе данных и будет доступен для просмотра на странице "Мои проекты".

Оба алгоритма были разработаны с учетом использования базы данных для хранения информации о проектах, клиентах и пользователях. Это позволяет приложению быстро и точно обрабатывать запросы и упрощает процесс добавления, редактирования и удаления информации.

Кроме того, при разработке алгоритмов уделено внимание возможным ошибкам пользователей. Если пользователь вводит некорректные данные, приложение информирует его об ошибке и предлагает исправить ее. Это позволяет сократить время на решение проблем и повышает удобство использования приложения.

**3 Программное конструирование**

В данном разделе будут обоснованы выбор языка программирования, используемый для реализации программного средства и основания выбора среды программирования. Будут определены и описаны основные классы разрабатываемого приложения, структура проекта и структура базы данных.

**3.1 Обоснование выбора средств разработки для реализации мобильного приложения «****Управление строительными проектами»**

Для реализации программного средства был взят за основу язык программирования Java и система управления базами данных firebase.

**3.2 Объектно-ориентированная модель программного средства**

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, где классы образуют иерархию наследования.

Объектно-ориентированные языки программирования пользуются большой популярностью среди программистов, так как они позволяют использовать преимущества объектно-ориентированного подхода не только на этапах проектирования и конструирования программных систем, но и на этапах их реализации, тестирования и сопровождения.

**3.3 Выбор среды программирования**

Для реализации мобильного приложения «Управление строительными проектами» была выбрана среда разработки Android Studio.

**3.4 Описание основных используемых классов**

В приложении используется несколько классов для хранения и управления данными. Они описывают различные объекты, которые используются в приложении:

* Класс ClientModel - представляет клиента, с которым связан проект. Содержит информацию о имени клиента и его номере телефона.
* Класс ExpenseModel - представляет расходы, связанные с проектом. Содержит информацию о поставщике услуг или материалов, описании расхода, его стоимости и оплаченной сумме.
* Класс IncomeModel - представляет доходы, связанные с проектом. Содержит информацию о названии дохода, его балансе и дате.
* Класс ProjectModel - представляет проект. Содержит информацию о названии проекта, доходах и расходах проекта, клиенте, связанном с проектом, списке трейдеров, связанных с проектом, и списке рабочих, задействованных в проекте.
* Класс TraderModel - представляет трейдера, поставщика услуг или материалов для проекта. Содержит информацию о названии трейдера, его номере телефона и балансе.

Класс WagesModel - представляет заработную плату рабочих, задействованных в проекте. Содержит информацию о рабочем, описании выплаты, ее стоимости, оплаченной сумме и долгах.

**3.5 Описание структуры базы данных**

Описание структуры базы данных для приложения по управлению строительством будет зависеть от используемой технологии хранения данных. В данном случае, автор приложения использует Firebase, которая предоставляет NoSQL базу данных в режиме реального времени.

Структура базы данных в Firebase будет иметь следующий вид: каждый пользователь будет иметь уникальный идентификатор, который будет использоваться в качестве ключа для создания базы данных конкретного пользователя. Внутри каждой базы данных будут храниться объекты классов, созданных для приложения. Например, все объекты класса ProjectModel будут храниться в пути /users/user\_id/projects.

Каждый объект будет содержать свой уникальный идентификатор в базе данных, который будет генерироваться автоматически Firebase. Это позволяет быстро получать доступ к нужному объекту и производить с ним дальнейшие операции. Кроме того, используя Firebase, можно легко синхронизировать данные между устройствами и обеспечить доступ к ним в режиме реального времени.

3.6 Выводы по главе

В данной главе был обоснован выбор языка и среды программирования для программного средства. Были описаны основные классы программного средства. В таблицах было приведено описание каждого метода класса. Также была описана структура базы данных.

4 Тестирование программного средства

В данном разделе продемонстрированы основные этапы тестирования мобильного приложения «Управление строительными проектами» на основе различных входных данных.

4.1 Описание процесса тестирования

В играх очень важную роль играет тестирование. Я тестировал Приложение на разных уровнях процесса разработки. Как правило, я использовал тестирование черного ящика для тестирования Приложение. Здесь я протестировал небольшие фрагменты кода, прежде чем отправить его для интеграции с остальной частью Приложение. На диаграмме ниже показан основной цикл тестирования Приложение

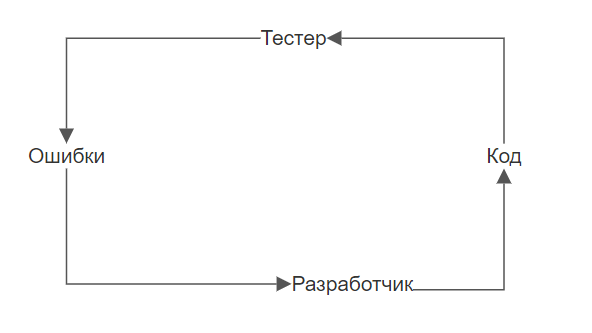
****

Рисунок 1 – Жизненный цикл тестирования

Этот цикл выполняется в следующие шесть шагов по порядку :-

Спланируйте и разработайте тест: здесь мы составляем документ плана тестирования с каждым новым прототип Приложение, вернитесь к коду и обновите его, если в Технические характеристики. Мы гарантируем, что не было внесено никаких новых проблем.

Подготовьте тест: здесь мы обновили наш код, тесты, документ и согласовали его с друг Приложение.

Выполните тест: в этом мы снова запустили тестовый костюм. Если обнаружен какой-либо дефект, мы проверяем вокруг дефекта, чтобы убедиться, что ошибка определена.

Сообщите о результатах: в этом мы сообщили полную информацию об ошибках.

Устранение ошибок: здесь мы проводим прямое тестирование, чтобы отследить ошибку и исправить ее.

Вернитесь к шагу 1 и повторите тестирование: в этом мы возвращаемся к шагу 1 и повторно тестируем другой. прототип игры таким же образом.

Тестирование Приложение проводится в структурированном виде. Независимо от размера Приложение и время, необходимое для производства Приложение

* 1. Тестирования Приложение Управление строительными проектами

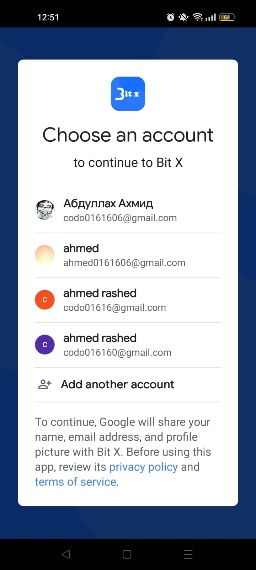
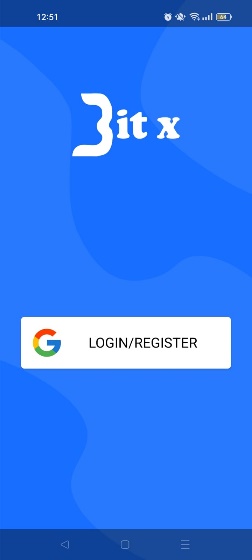


Рисунок 2 – страница регистрации



Рисунок 3 – домашняя страница

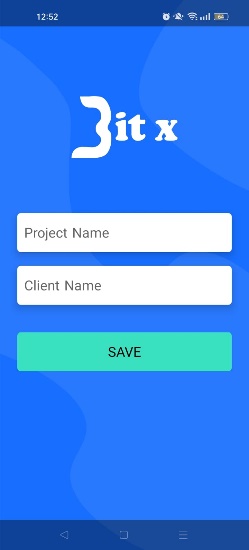
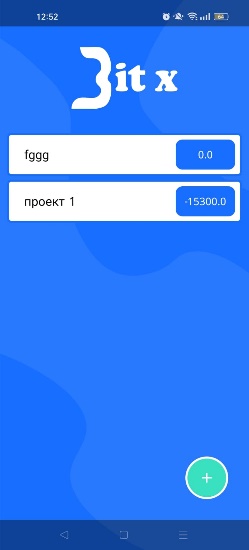


Рисунок 4 – страница проектов и добавить новый проект



Рисунок 5 – страница проекта

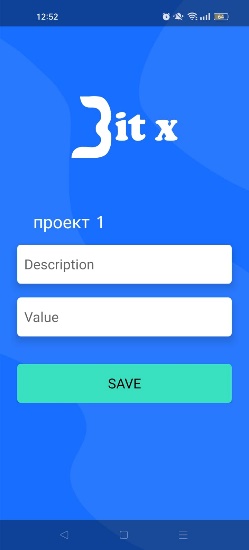
****

Рисунок 6 – страница доходов и добавить новую дохода

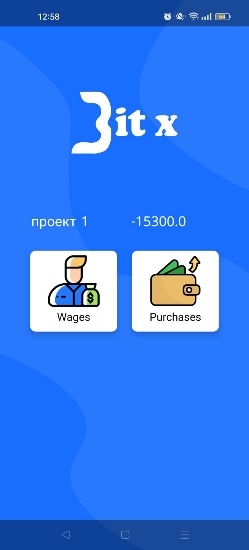


Рисунок 7 – страница расходов

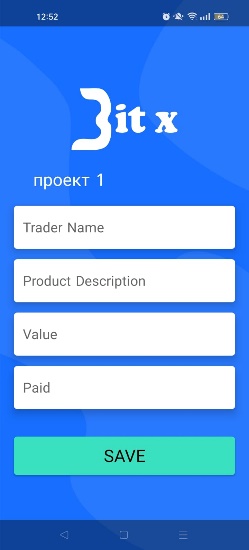


Рисунок 8 – покупоки и добавить новую страницу покупки

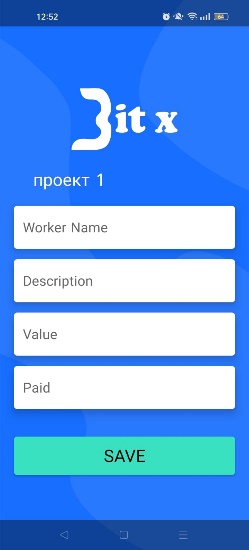


Рисунок 9 – заработной платы , добавить новую заработной платы страницы

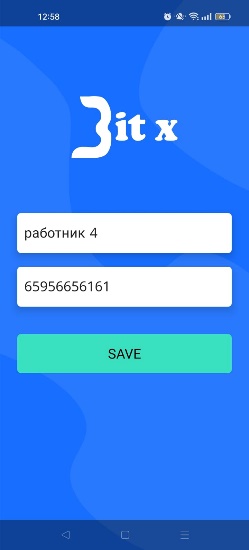
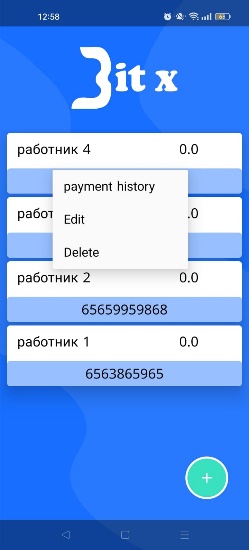


Рисунок 10 – Страницы рабочий, на которых вы можете просмотреть всех рабочов, отредактировать рабочий или удалить рабочий

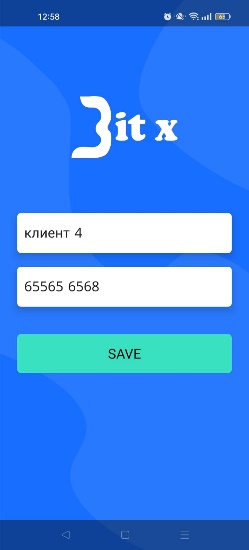
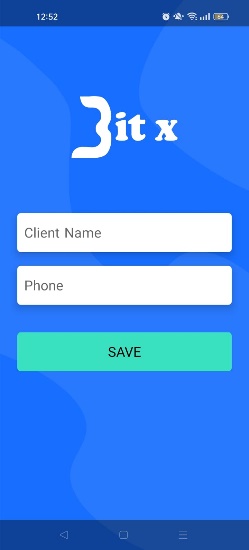


Рисунок 11 – Страницы клиента, на которых вы можете просмотреть всех клиентов, отредактировать клиента или удалить клиента

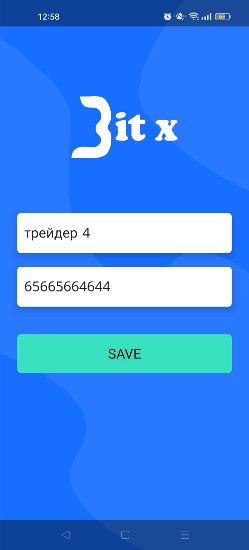
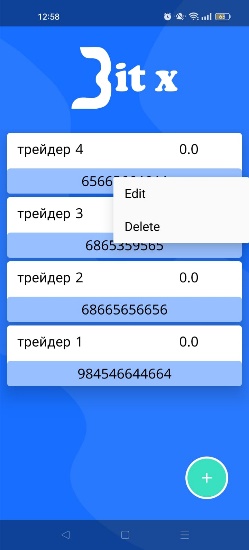
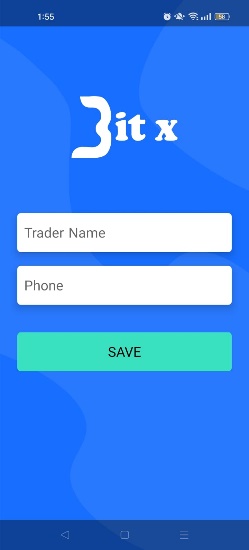


Рисунок 12 – Страницы продавца, на которых вы можете просмотреть всех продавцов, отредактировать продавца или удалить продавца

4.2 Выводы по главе

В данной главе было произведено последовательное знакомство с разработанным функционалом мобильного приложения «Управление строительными проектами» с последующим его тестированием.

Заключение

В целом, проект был успешно реализован и достиг своей основной цели - предоставление удобного и эффективного инструмента для управления небольшими строительными проектами через мобильное приложение. Команда проекта проявила высокий профессионализм и технические знания в создании приложения, которое имеет удобный и интуитивно понятный интерфейс, а также функционал, необходимый для управления проектами.

Однако, в процессе выполнения проекта возникли некоторые трудности, такие как сложности с совместимостью приложения на разных устройствах и в различных операционных системах. Были также выявлены некоторые недочеты в работе приложения, которые нуждаются в дальнейшей доработке и улучшении.

В целом, проект был выполнен с высокой степенью внимания к деталям и профессионализма. Были изучены многие аспекты управления строительными проектами, и приложение было разработано в соответствии с ними.

Перечень использованных информационных ресурсов

1. Aws [Электронный ресурс], URL: [https://aws.amazon.com/ru/what-is/java](https://aws.amazon.com/ru/what-is/java/#:~:text=Java%20%E2%80%93%20%D1%8D%D1%82%D0%BE%20%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9%2C%20%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE%2D,%D0%B1%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B) (дата обращения 20.02.2023);

2. Web-creator [Электронный ресурс], URL: <https://web-creator.ru/articles/java> (дата обращения 22.02.23);

3. Skillfactory [Электронный ресурс], URL: <https://blog.skillfactory.ru/glossary/sqlite/> (дата обращения 23.02.23);

4. SQLite [Электронный ресурс], URL: <https://younglinux.info/sqlite/sqlite> (дата обращения 24.02.23);

5. Web-Proger [Электронный ресурс], URL: [http://web.spt42.ru/index.php](http://web.spt42.ru/index.php/chto-takoe-android-studio#:~:text=Android%20Studio%20%E2%80%94%20%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F%20%D1%81%D1%80%D0%B5%D0%B4%D0%B0%20%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8,Play%20App%20Store%20%D1%81%D) (дата обращения 01.03.23);

6. Otus journal [Электронный ресурс], URL: [http://web.spt42.ru/index.php](http://web.spt42.ru/index.php/chto-takoe-android-studio#:~:text=Android%20Studio%20%E2%80%94%20%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F%20%D1%81%D1%80%D0%B5%D0%B4%D0%B0%20%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8,Play%20App%20Store%20%D1%81%D) (дата обращения 01.03.23);

Приложение А

Техническое задание

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Доцент каф. «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_\_/ В.Н. Литвинов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2023г. |  |

А.1 Введение

А.1.1 Наименование программного средства

Наименование программного средства – «Управление строительными проектами».

А.1.2 Область применения

Приложение можете использовать приложение, чтобы помочь в строительных проектах

А.2 Основание для разработки

Разработка ведется на основании документа «Учебный план для студентов ВУЗа» направление 09.03.04 «Программная инженерия» кафедры «Программное обеспечение вычислительное техники и автоматизированных систем» факультета «Информатика и вычислительная техника» Донского Государственного Технического Университета.

А.3 Назначение разработки

А.3.1 Функциональное назначение

использовать, чтобы помочь вам в строительных проектах финансовых

А.3.2 Эксплуатационное назначение

Оперативная цель - использовать программу на android.

А.4 Требования к программе

А.4.1 Требования к функциональным характеристикам

Программное средство должно осуществлять следующие функции:

• Создать новый проект

• Добавлять и редактировать клиентов

• Добавлять и редактировать работников

• Добавлять и редактировать продавцов

• Добавлять и редактировать счета

А.4.2 Требования к надежности

Для надежной работы программного средства необходимо, чтобы выполнялись следующие условия:

* бесперебойное питание технического средства, на котором находится продукт;
* регулярная проверка программного средства на наличие вирусов.

А.4.2.1 Входные данные

Данные о проектах и ​​расходах товаров и рабочих

А.4.2.2 Выходные данные

Вывод предоставляется в Финансовые счета проекта.

А.4.3 Условия эксплуатации

Для стабильного функционирования и оптимальной работы программного продукта необходимо соблюдение всех требований и правил эксплуатации вычислительной техники. Каких-либо требований к пользователю данного приложения нет.

А.4.4 Требование к составу и параметрам технических средств

В состав технических средств должна входить операционная система android:

● процессор snabdragon460;

● Оперативная память не менее 1 ГБ.

Дополнительные требования и ограничения к составу и параметрам технических средств не вводятся..

А.4.5 Требования к исходным кодам и языкам программирования

Программное средство должно быть реализовано с использованием языка программирования Java, а для пользовательского интерфейса должен использоваться Java Swing.

А.4.6 Требования к упаковке и маркировки

К упаковке и маркировки специальных требований не предъявляются.

А.4.7 Требования к транспортировке и хранению

Условия транспортирования, места хранения, условия складирования и сроки хранения в различных условиях должны соответствовать требованиям, предъявляемым к носителям информации, на которых будет содержаться данное программное изделие. Программное средство может храниться на любых цифровых носителях информации (жесткий диск, компакт – диск, флэш накопитель и т. п.).

А.5 Требование к программной документации

Программная документация состоит из следующего:

* титульный лист;
* лист задания на учебную ознакомительную практику;
* пояснительная записка к учебной ознакомительной практике;
* техническое задание по ГОСТ 19.201-78 ЕСПД;
* исходный код программного средства по ГОСТ 19.401-79 ЕСПД.

А.6 Стадии и этапы разработки

Реализация программного средства состояла из следующих этапов:

* постановка задачи (22.06.2021);
* изучение предметной области (23.06.2021 – 25.06.2021);
* алгоритмическое конструирование (26.06.2021 – 1.07.2021);
* программная реализация (2.07.2021 – 11.07.2021);
* тестирование приложения (12.07.2021 – 13.07.2021);
* разработка отчета (14.07.2021 – 16.07.2021).

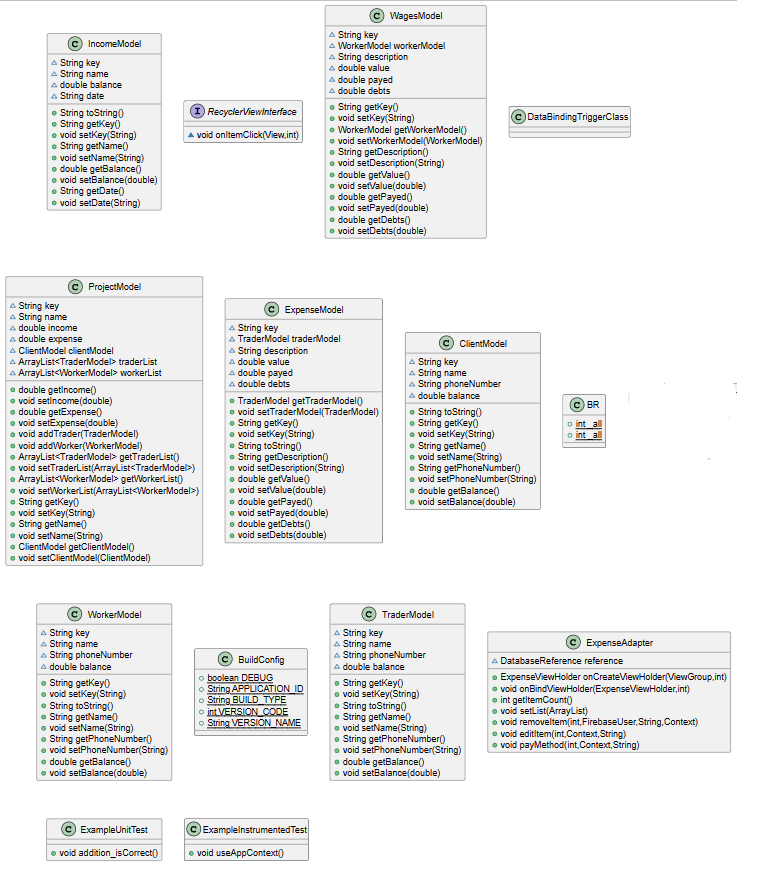
А.7 Порядок и контроль приемки

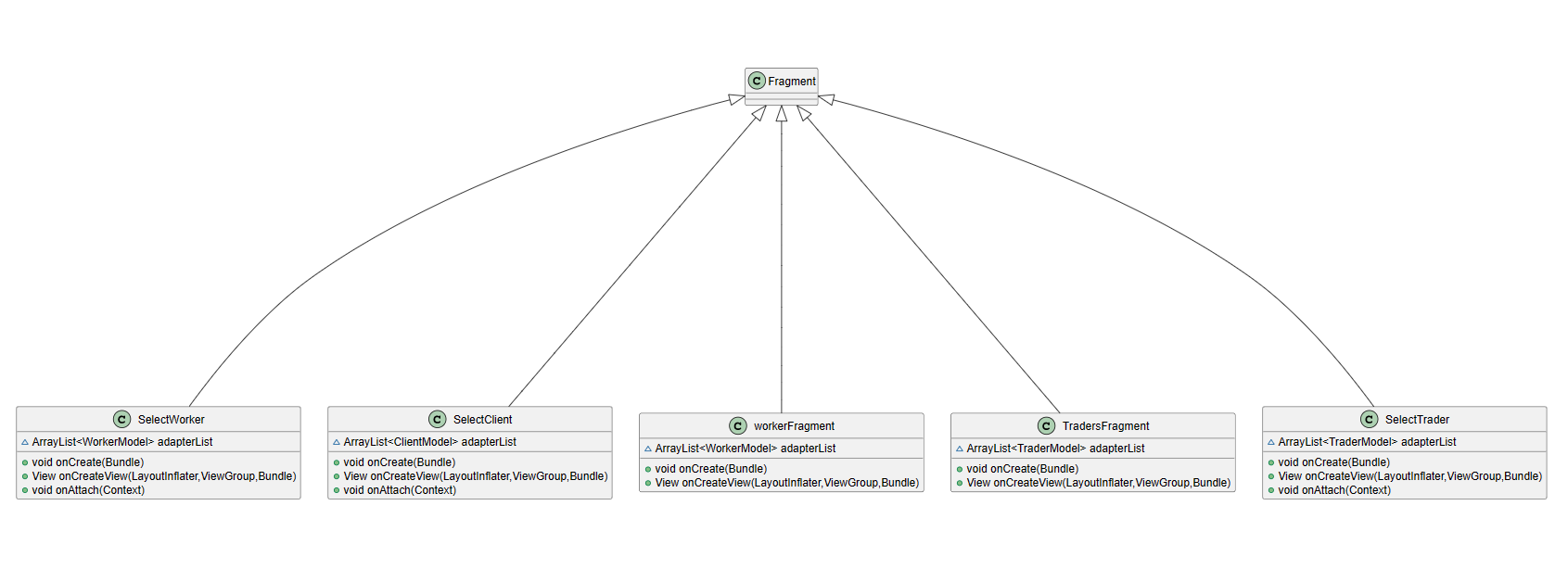
Порядок и контроль приемки определяются заведующим кафедрой «ПОВТ и АС», подразумевающие собой демонстрацию показателя владения средствами для разработки программных средств в различных направлениях.

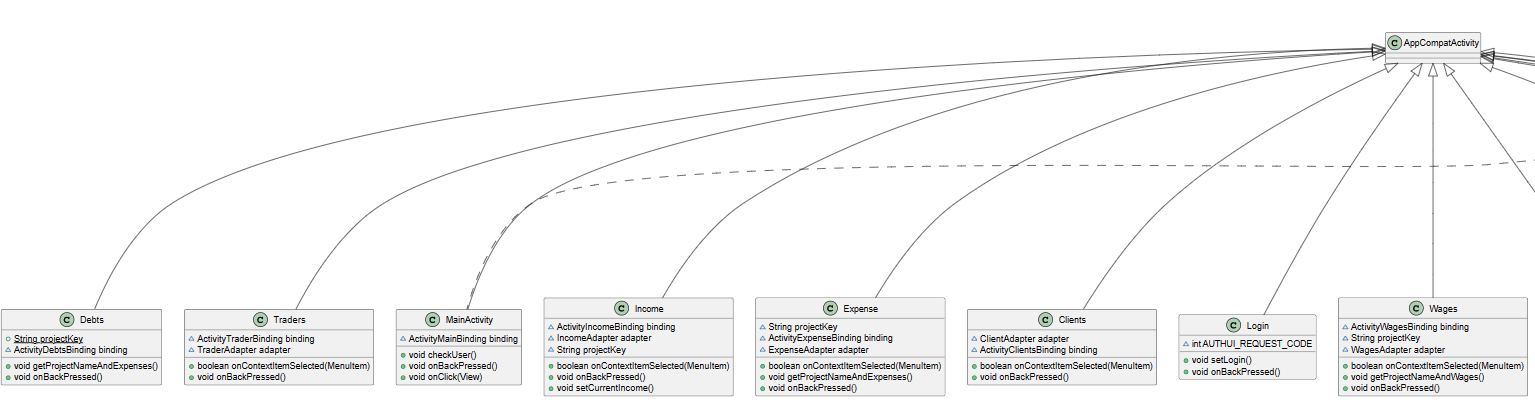
Главным требованием к приемке является наличие корректного работающего программного средства и отчета, предоставленного в печатном виде.

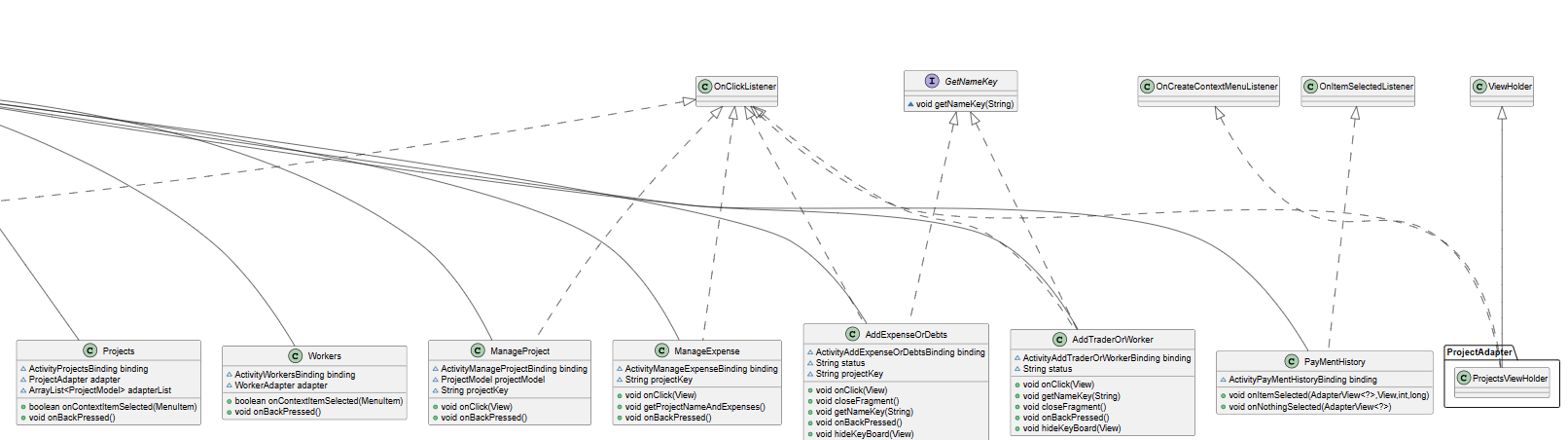
|  |  |
| --- | --- |
| Разработчик технического задания:  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. | Абдуллах Ахмид  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

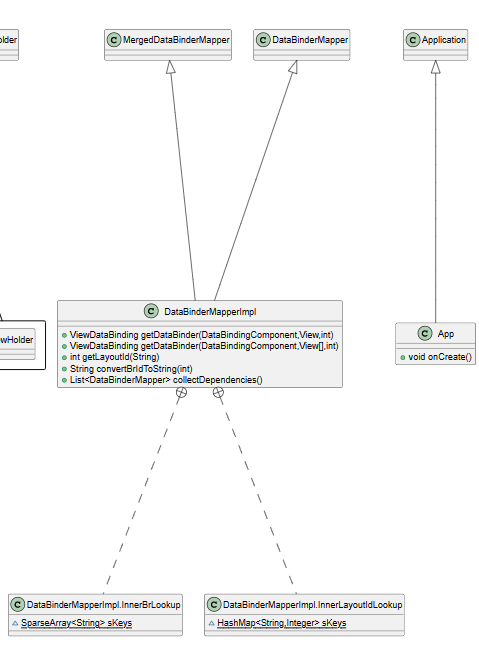
ПРИЛОЖЕНИЕ Б. UML-ДИАГРАММА КЛАССОВ.











Приложение Г

Исходный код программного средства

Полный код программного средства находится на съемном носителе, прилагаемом к данной работе.

public class ClientModel {  
 @Exclude  
 String key;  
 String name;  
 String phoneNumber;  
 double balance;  
  
 public ClientModel() {  
 }  
  
 public ClientModel(String name, String phoneNumber) {  
 this.name = name;  
 this.phoneNumber = phoneNumber;  
 }  
  
  
 public ClientModel( String name, String phoneNumber, double balance) {  
 this.name = name;  
 this.phoneNumber = phoneNumber;  
 this.balance = balance;  
 }  
  
 @Override  
 public String toString() {  
 return "ClientModel{" +  
 "key='" + key + '\'' +  
 ", name='" + name + '\'' +  
 ", phoneNumber='" + phoneNumber + '\'' +  
 ", balance=" + balance +  
 '}';  
 }  
  
 public String getKey() {  
 return key;  
 }  
  
 public void setKey(String key) {  
 this.key = key;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 public void setPhoneNumber(String phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 }  
  
 public double getBalance() {  
 return balance;  
 }  
  
 public void setBalance(double balance) {  
 this.balance = balance;  
 }  
}

public class ExpenseModel {  
  
  
 @Exclude  
 String key;  
 TraderModel traderModel;  
 String description;  
 double value;  
 double payed;  
  
  
 public ExpenseModel() {  
 }  
  
  
 public ExpenseModel(TraderModel traderModel, String description, double value, double payed) {  
 this.traderModel = traderModel;  
 this.description = description;  
 this.value = value;  
 this.payed = payed;  
 this.debts = payed-value;  
 }  
  
 double debts;  
  
  
 public TraderModel getTraderModel() {  
 return traderModel;  
 }  
  
 public void setTraderModel(TraderModel traderModel) {  
 this.traderModel = traderModel;  
 }  
  
 public String getKey() {  
  
 return key;  
 }  
  
 public void setKey(String key) {  
 this.key = key;  
 }  
  
 @Override  
 public String toString() {  
 return "ExpenseModel{" +  
 "traderName='" + traderModel + '\'' +  
 ", description='" + description + '\'' +  
 ", value=" + value +  
 ", payed=" + payed +  
 ", debts=" + debts +  
 '}';  
 }  
  
  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 public double getValue() {  
 return value;  
 }  
  
 public void setValue(double value) {  
 this.value = value;  
 }  
  
 public double getPayed() {  
 return payed;  
 }  
  
 public void setPayed(double payed) {  
 this.payed = payed;  
 }  
  
 public double getDebts() {  
 return debts;  
 }  
  
 public void setDebts(double debts) {  
 this.debts = debts;  
 }  
  
  
}

public class IncomeModel {  
  
 @Exclude  
 String key;  
 String name;  
 double balance;  
 String date;  
  
 public IncomeModel(String name, double balance, String date) {  
 this.name = name;  
 this.balance = balance;  
 this.date = date;  
 }  
  
 public IncomeModel(String name, double balance) {  
 this.name = name;  
 this.balance = balance;  
 }  
  
 public IncomeModel() {  
 }  
  
 @Override  
 public String toString() {  
 return "IncomeModel{" +  
 "name='" + name + '\'' +  
 ", balance=" + balance +  
 ", date='" + date + '\'' +  
 '}';  
 }  
  
 public String getKey() {  
 return key;  
 }  
  
 public void setKey(String key) {  
 this.key = key;  
 }  
  
  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public double getBalance() {  
 return balance;  
 }  
  
 public void setBalance(double balance) {  
 this.balance = balance;  
 }  
  
 public String getDate() {  
 return date;  
 }  
  
 public void setDate(String date) {  
 this.date = date;  
 }  
}

public class ProjectModel {  
 @Exclude  
 String key;  
 String name;//\*  
 double income;  
 double expense;  
 ClientModel clientModel;//\*  
 ArrayList<TraderModel> traderList;  
 ArrayList<WorkerModel> workerList;  
  
 public double getIncome() {  
 return income;  
 }  
  
 public void setIncome(double income) {  
 this.income = income;  
 }  
  
 public double getExpense() {  
 return expense;  
 }  
  
 public void setExpense(double expense) {  
 this.expense = expense;  
 }  
  
  
 public ProjectModel(String name, ClientModel clientModel) {  
 this.name = name;  
 this.clientModel = clientModel;  
 }  
  
 public ProjectModel(String name, double income) {  
 this.name = name;  
 this.income = income;  
 }  
  
 public ProjectModel() {  
 }  
  
 public ProjectModel(String name, double income, double expense, ClientModel clientModel) {  
 this.name = name;  
 this.income = income;  
 this.expense = expense;  
 this.clientModel = clientModel;  
 }  
  
  
  
  
 public void addTrader(TraderModel traderModel){  
 traderList.add(traderModel);  
 }  
 public void addWorker(WorkerModel workerModel){  
 workerList.add(workerModel);  
 }  
 public ArrayList<TraderModel> getTraderList() {  
 return traderList;  
 }  
  
 public void setTraderList(ArrayList<TraderModel> traderList) {  
 this.traderList = traderList;  
 }  
  
 public ArrayList<WorkerModel> getWorkerList() {  
 return workerList;  
 }  
  
 public void setWorkerList(ArrayList<WorkerModel> workerList) {  
 this.workerList = workerList;  
 }  
  
 public String getKey() {  
 return key;  
 }  
  
 public void setKey(String key) {  
 this.key = key;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
  
  
 public ClientModel getClientModel() {  
 return clientModel;  
 }  
  
 public void setClientModel(ClientModel clientModel) {  
 this.clientModel = clientModel;  
 }  
  
  
}

public class TraderModel {  
  
 @Exclude  
 String key;  
 String name;  
 String phoneNumber;  
 double balance;  
  
  
  
 public String getKey() {  
 return key;  
 }  
  
 public void setKey(String key) {  
 this.key = key;  
 }  
  
  
 public TraderModel() {  
 }  
  
 public TraderModel(String name, String phoneNumber) {  
 this.name = name;  
 this.phoneNumber = phoneNumber;  
 }  
  
  
 @Override  
 public String toString() {  
 return "Trader{" +  
 "name='" + name + '\'' +  
 ", phoneNumber='" + phoneNumber + '\'' +  
 ", balance=" + balance +  
 '}';  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 public void setPhoneNumber(String phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 }  
  
 public double getBalance() {  
 return balance;  
 }  
  
 public void setBalance(double balance) {  
 this.balance = balance;  
 }  
  
 public TraderModel(String name, String phoneNumber, double balance) {  
 this.name = name;  
 this.phoneNumber = phoneNumber;  
 this.balance = balance;  
 }  
}

public class WagesModel {  
 @Exclude  
 String key;  
 WorkerModel workerModel;  
 String description;  
 double value;  
 double payed;  
 double debts;  
  
 public WagesModel() {  
 }  
  
 public WagesModel(WorkerModel workerModel, String description, double value, double payed) {  
 this.workerModel = workerModel;  
 this.description = description;  
 this.value = value;  
 this.payed = payed;  
 this.debts = payed-value;  
 }  
 public String getKey() {  
 return key;  
 }  
  
 public void setKey(String key) {  
 this.key = key;  
 }  
  
 public WorkerModel getWorkerModel() {  
 return workerModel;  
 }  
  
 public void setWorkerModel(WorkerModel workerModel) {  
 this.workerModel = workerModel;  
 }  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 public double getValue() {  
 return value;  
 }  
  
 public void setValue(double value) {  
 this.value = value;  
 }  
  
 public double getPayed() {  
 return payed;  
 }  
  
 public void setPayed(double payed) {  
 this.payed = payed;  
 }  
  
 public double getDebts() {  
 return debts;  
 }  
  
 public void setDebts(double debts) {  
 this.debts = debts;  
 }  
  
}

public class WorkerModel {  
 @Exclude  
 String key;  
 String name;  
 String phoneNumber;  
 double balance;  
  
 public WorkerModel(String key, String name, String phoneNumber, double balance) {  
 this.key = key;  
 this.name = name;  
 this.phoneNumber = phoneNumber;  
 this.balance = balance;  
 }  
  
 public String getKey() {  
 return key;  
 }  
  
 public void setKey(String key) {  
 this.key = key;  
 }  
  
  
  
 public WorkerModel(String name, String phoneNumber) {  
 this.name = name;  
 this.phoneNumber = phoneNumber;  
 }  
  
 public WorkerModel() {  
 }  
  
  
 @Override  
 public String toString() {  
 return "WorkerModel{" +  
 "name='" + name + '\'' +  
 ", phoneNumber='" + phoneNumber + '\'' +  
 ", balance=" + balance +  
 '}';  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 public void setPhoneNumber(String phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 }  
  
 public double getBalance() {  
 return balance;  
 }  
  
 public void setBalance(double balance) {  
 this.balance = balance;  
 }  
  
 public WorkerModel(String name, String phoneNumber, double balance) {  
 this.name = name;  
 this.phoneNumber = phoneNumber;  
 this.balance = balance;  
 }  
}

public class SelectClient extends Fragment {  
 private RecyclerViewInterface listener;  
 private GetNameKey namePosition;  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 ArrayList<ClientModel> adapterList;  
 private FirebaseUser user;  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
  
 View v = inflater.inflate(R.layout.*fragment\_select\_client*, container, false);  
  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
  
 setOnClickListener();  
 getClients(v);  
 return v;  
 }  
  
 @Override  
 public void onAttach(@NonNull Context context) {  
 super.onAttach(context);  
 Activity activity = (Activity) context;  
 try {  
 namePosition = (GetNameKey) activity;  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 private void getClients(View v){  
 adapterList = new ArrayList<>();  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Clients");  
 reference.keepSynced(true);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 adapterList.clear();  
 for (DataSnapshot data: snapshot.getChildren()) {  
 ClientModel clientModel = data.getValue(ClientModel.class);  
 clientModel.setKey(data.getKey());  
 adapterList.add(clientModel);  
 }}  
 setAdapter(adapterList,v);  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
  
 private void setAdapter(ArrayList<ClientModel> clientList ,View v){  
 ClientFragmentAdapter adapter =new ClientFragmentAdapter();  
 RecyclerView recyclerView = v.findViewById(R.id.*selectClientRecycler*);  
 ImageView icon =v.findViewById(R.id.*selectClientIcon*);  
 if(adapterList.size()>0){  
 icon.setVisibility(View.*GONE*);  
 recyclerView.setVisibility(View.*VISIBLE*);  
 }else{  
 icon.setVisibility(View.*VISIBLE*);  
 recyclerView.setVisibility(View.*GONE*);  
 }  
 recyclerView.setAdapter(adapter);  
 adapter.setList(clientList,listener);  
 LinearLayoutManager linearLayoutManager = new LinearLayoutManager(v.getContext());  
 linearLayoutManager.setReverseLayout(true);  
 linearLayoutManager.setStackFromEnd(true);  
 recyclerView.setLayoutManager(linearLayoutManager);  
 }  
  
 private void setOnClickListener() {  
 listener =new RecyclerViewInterface() {  
 @Override  
 public void onItemClick(View v, int pos) {  
 try {  
 namePosition.getNameKey(adapterList.get(pos).getKey());  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 };  
 }  
  
}

public class SelectTrader extends Fragment{  
  
 private RecyclerViewInterface listener;  
 private GetNameKey namePosition;  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 ArrayList<TraderModel> adapterList;  
 private FirebaseUser user;  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 View v = inflater.inflate(R.layout.*fragment\_select\_trader*, container, false);  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
  
 setOnClickListener();  
 getTraders(v);  
 return v;  
 }  
  
 @Override  
 public void onAttach(@NonNull Context context) {  
 super.onAttach(context);  
 Activity activity = (Activity) context;  
 try {  
 namePosition = (GetNameKey) activity;  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
  
 private void getTraders(View v){  
 adapterList = new ArrayList<>();  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Traders");  
 reference.keepSynced(true);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()) {  
 adapterList.clear();  
 for (DataSnapshot data : snapshot.getChildren()) {  
 TraderModel traderModel = data.getValue(TraderModel.class);  
 traderModel.setKey(data.getKey());  
 adapterList.add(traderModel);  
 }  
 }  
 setAdapter(adapterList,v);  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
  
 private void setAdapter(ArrayList<TraderModel> traderList ,View v){  
 TraderFragmentAdapter adapter =new TraderFragmentAdapter();  
 RecyclerView recyclerView = v.findViewById(R.id.*selectTradeRecycler*);  
 ImageView icon =v.findViewById(R.id.*selectTraderIcon*);  
 if(adapterList.size()>0){  
 icon.setVisibility(View.*GONE*);  
 recyclerView.setVisibility(View.*VISIBLE*);  
 }else{  
 icon.setVisibility(View.*VISIBLE*);  
 recyclerView.setVisibility(View.*GONE*);  
 }  
 recyclerView.setAdapter(adapter);  
 adapter.setList(traderList,listener);  
 LinearLayoutManager linearLayoutManager = new LinearLayoutManager(v.getContext());  
 linearLayoutManager.setReverseLayout(true);  
 linearLayoutManager.setStackFromEnd(true);  
 recyclerView.setLayoutManager(linearLayoutManager);  
 }  
  
  
 private void setOnClickListener() {  
 listener =new RecyclerViewInterface() {  
 @Override  
 public void onItemClick(View v, int pos) {  
 try {  
 namePosition.getNameKey(adapterList.get(pos).getKey());  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
  
 };  
 }  
}

public class SelectWorker extends Fragment {  
  
 private RecyclerViewInterface listener;  
 private GetNameKey namePosition;  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 ArrayList<WorkerModel> adapterList;  
 private FirebaseUser user;  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View v = inflater.inflate(R.layout.*fragment\_select\_worker*, container, false);  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
  
 setOnClickListener();  
 getWorkers(v);  
 return v;  
 }  
  
 @Override  
 public void onAttach(@NonNull Context context) {  
 super.onAttach(context);  
 Activity activity = (Activity) context;  
 try {  
 namePosition = (GetNameKey) activity;  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 private void getWorkers(View v){  
 adapterList = new ArrayList<>();  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Workers");  
 reference.keepSynced(true);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()) {  
 adapterList.clear();  
 for (DataSnapshot data : snapshot.getChildren()) {  
 WorkerModel workerModel = data.getValue(WorkerModel.class);  
 workerModel.setKey(data.getKey());  
 adapterList.add(workerModel);  
 }  
 }  
 setAdapter(adapterList,v);  
  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
  
 private void setAdapter(ArrayList<WorkerModel> workerList ,View v){  
 WorkerFragmentAdapter adapter =new WorkerFragmentAdapter();  
 RecyclerView recyclerView = v.findViewById(R.id.*selectWorkerRecycler*);  
 ImageView icon =v.findViewById(R.id.*selectWorkerIcon*);  
 if(adapterList.size()>0){  
 icon.setVisibility(View.*GONE*);  
 recyclerView.setVisibility(View.*VISIBLE*);  
 }else{  
 icon.setVisibility(View.*VISIBLE*);  
 recyclerView.setVisibility(View.*GONE*);  
 }  
 recyclerView.setAdapter(adapter);  
 adapter.setList(workerList,listener);  
 LinearLayoutManager linearLayoutManager = new LinearLayoutManager(v.getContext());  
 linearLayoutManager.setReverseLayout(true);  
 linearLayoutManager.setStackFromEnd(true);  
 recyclerView.setLayoutManager(linearLayoutManager);  
 }  
  
  
  
 private void setOnClickListener() {  
 listener =new RecyclerViewInterface() {  
 @Override  
 public void onItemClick(View v, int pos) {  
 try {  
 namePosition.getNameKey(adapterList.get(pos).getKey());  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 };  
 }  
}

public class TradersFragment extends Fragment {  
 private DatabaseReference reference;  
 private RecyclerViewInterface listener;  
 private String projectKey ="";  
 private FirebaseAuth auth ;  
 private FirebaseUser user;  
 ArrayList<TraderModel> adapterList;  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 projectKey = Debts.*projectKey*;  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View v = inflater.inflate(R.layout.*fragment\_traders*, container, false);  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
 setOnClickListener();  
 getTraders(v);  
 return v;  
 }  
 private void getTraders(View v){  
 adapterList = new ArrayList<>();  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Expenses");  
 reference.keepSynced(true);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 adapterList.clear();  
 for (DataSnapshot data: snapshot.getChildren()) {  
 ExpenseModel expenseModel = data.getValue(ExpenseModel.class);  
 if(expenseModel.getTraderModel().getBalance()<0){  
 adapterList.add(expenseModel.getTraderModel());  
 }  
 }  
 for (int i = 0; i < adapterList.size(); i++) {  
 for (int j = i; j < adapterList.size(); j++) {  
 if(i!=j&&adapterList.get(i).getKey().equals(adapterList.get(j).getKey())){  
 adapterList.get(i).setBalance(adapterList.get(i).getBalance()+adapterList.get(j).getBalance());  
 adapterList.remove(j);  
 }  
 }  
 }  
 setAdapter(adapterList,v);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
  
  
 private void setAdapter(ArrayList<TraderModel> traderList ,View v){  
 TraderFragmentAdapter adapter =new TraderFragmentAdapter();  
 RecyclerView recyclerView = v.findViewById(R.id.*tabTradeRecycler*);  
 ImageView icon =v.findViewById(R.id.*tabTraderIcon*);  
 if(adapterList.size()>0){  
 icon.setVisibility(View.*GONE*);  
 recyclerView.setVisibility(View.*VISIBLE*);  
 }else{  
 icon.setVisibility(View.*VISIBLE*);  
 recyclerView.setVisibility(View.*GONE*);  
 }  
 recyclerView.setAdapter(adapter);  
 adapter.setList(traderList,listener);  
 LinearLayoutManager linearLayoutManager = new LinearLayoutManager(v.getContext());  
 linearLayoutManager.setReverseLayout(true);  
 linearLayoutManager.setStackFromEnd(true);  
 recyclerView.setLayoutManager(linearLayoutManager);  
 }  
 private void setOnClickListener() {  
 listener =new RecyclerViewInterface() {  
 @Override  
 public void onItemClick(View v, int pos) {  
 Intent intent =new Intent(getActivity(), AddTraderOrWorker.class);  
 intent.putExtra("tag","editTraderDebts");  
 intent.putExtra("projectKey",projectKey);  
 intent.putExtra("workerKey",adapterList.get(pos).getKey());  
 startActivity(intent);  
 ((Activity) getActivity()).overridePendingTransition(0, 0);  
  
 }  
 };  
 }  
  
}

package com.example.bit\_x.ui.fragment;  
  
import android.app.Activity;  
import android.content.Context;  
import android.content.Intent;  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ImageView;  
  
import com.example.bit\_x.R;  
import com.example.bit\_x.adabter.WorkerFragmentAdapter;  
import com.example.bit\_x.interfaces.GetNameKey;  
import com.example.bit\_x.interfaces.RecyclerViewInterface;  
import com.example.bit\_x.models.WagesModel;  
import com.example.bit\_x.models.WorkerModel;  
import com.example.bit\_x.ui.AddTraderOrWorker;  
import com.example.bit\_x.ui.Debts;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.ArrayList;  
  
public class workerFragment extends Fragment {  
  
 private RecyclerViewInterface listener;  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 ArrayList<WorkerModel> adapterList;  
 private FirebaseUser user;  
 private String projectKey ="";  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 projectKey = Debts.*projectKey*;  
  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View v = inflater.inflate(R.layout.*fragment\_worker*, container, false);  
 auth = FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
  
 setOnClickListener();  
 getWorkers(v);  
 return v;  
 }  
  
 private void getWorkers(View v){  
 adapterList = new ArrayList<>();  
 System.*out*.println(Debts.*projectKey*);  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Wages");  
 reference.keepSynced(true);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 adapterList.clear();  
 System.*out*.println("Workers");  
 for (DataSnapshot data: snapshot.getChildren()) {  
 WagesModel wagesModel = data.getValue(WagesModel.class);  
 if(wagesModel.getWorkerModel().getBalance()<0){  
 adapterList.add(wagesModel.getWorkerModel());  
 }  
 }  
 for (int i = 0; i < adapterList.size(); i++) {  
 for (int j = i; j < adapterList.size(); j++) {  
 if(i!=j&&adapterList.get(i).getKey().equals(adapterList.get(j).getKey())){  
 adapterList.get(i).setBalance(adapterList.get(i).getBalance()+adapterList.get(j).getBalance());  
 adapterList.remove(j);  
 }  
 }  
 }  
 setAdapter(adapterList,v);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
  
 private void setAdapter(ArrayList<WorkerModel> workerList ,View v){  
 WorkerFragmentAdapter adapter =new WorkerFragmentAdapter();  
 RecyclerView recyclerView = v.findViewById(R.id.*tabWorkerRecycler*);  
 ImageView icon =v.findViewById(R.id.*tabWorkerIcon*);  
 if(adapterList.size()>0){  
 icon.setVisibility(View.*GONE*);  
 recyclerView.setVisibility(View.*VISIBLE*);  
 }else{  
 icon.setVisibility(View.*VISIBLE*);  
 recyclerView.setVisibility(View.*GONE*);  
 }  
 recyclerView.setAdapter(adapter);  
 adapter.setList(workerList,listener);  
 LinearLayoutManager linearLayoutManager = new LinearLayoutManager(v.getContext());  
 linearLayoutManager.setReverseLayout(true);  
 linearLayoutManager.setStackFromEnd(true);  
 recyclerView.setLayoutManager(linearLayoutManager);  
 }  
  
  
 private void setOnClickListener() {  
 listener =new RecyclerViewInterface() {  
 @Override  
 public void onItemClick(View v, int pos) {  
 Intent intent =new Intent(getActivity(), AddTraderOrWorker.class);  
 intent.putExtra("tag","editWorkerDebts");  
 intent.putExtra("projectKey",projectKey);  
 intent.putExtra("workerKey",adapterList.get(pos).getKey());  
 startActivity(intent);  
 ((Activity) getActivity()).overridePendingTransition(0, 0);  
 ((Activity) getActivity()).finish();  
 }  
 };  
 }  
}

package com.example.bit\_x.ui;  
  
import static android.content.ContentValues.*TAG*;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.fragment.app.Fragment;  
import androidx.fragment.app.FragmentManager;  
import androidx.fragment.app.FragmentTransaction;  
  
import android.content.Context;  
import android.content.Intent;  
import android.content.res.Configuration;  
import android.graphics.drawable.Drawable;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.view.inputmethod.InputMethodManager;  
  
import com.example.bit\_x.R;  
import com.example.bit\_x.databinding.ActivityAddExpenseOrDebtsBinding;  
import com.example.bit\_x.interfaces.GetNameKey;  
import com.example.bit\_x.models.ClientModel;  
import com.example.bit\_x.models.ExpenseModel;  
import com.example.bit\_x.models.ProjectModel;  
import com.example.bit\_x.models.TraderModel;  
import com.example.bit\_x.models.WagesModel;  
import com.example.bit\_x.models.WorkerModel;  
import com.example.bit\_x.ui.fragment.SelectTrader;  
import com.example.bit\_x.ui.fragment.SelectWorker;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.Locale;  
  
public class AddExpenseOrDebts extends AppCompatActivity implements View.OnClickListener , GetNameKey {  
 ActivityAddExpenseOrDebtsBinding binding;  
 String status="";  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 private FirebaseUser user;  
 private TraderModel mainTraderModel;  
 private WorkerModel mainWorkerModel;  
 private WagesModel mainWagesModel;  
 String projectKey="";  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding =ActivityAddExpenseOrDebtsBinding.*inflate*(getLayoutInflater());  
 View view = binding.getRoot();  
 this.setContentView(view);  
 // setAppLanguage();  
 binding.name.setOnClickListener(this);  
 binding.buAddEWSave.setOnClickListener(this);  
 binding.linearLayout.setOnClickListener(this);  
 binding.constraintLayout.setOnClickListener(this);  
 binding.logo.setOnClickListener(this);  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
  
 Intent intent = getIntent();  
 projectKey =intent.getStringExtra("projectKey");  
 String id = intent.getStringExtra("tag");  
 switch (id){  
 case "addExpense":  
 setExpenseView();  
 status ="addExpense";  
 break;  
 case "editExpense":  
 binding.name.setBackgroundResource(R.drawable.*layout\_design\_gray*);  
 setExpenseView();  
 getExpenseData();  
 getMainTrader();  
 status="editExpense";  
 break;  
 case "addWages":  
 setWagesView();  
 status="addWages";  
 break;  
 case "editWages":  
 binding.name.setBackgroundResource(R.drawable.*layout\_design\_gray*);  
 setWagesView();  
 getWagesData();  
 getMainWorker();  
 status="editWages";  
 break;  
 case "workerPayMethod":  
 setPayMethodView();  
 getWagesData();  
 getMainWorker();  
 status="workerPayMethod";  
 break;  
 case "traderPayMethod":  
 setPayMethodView();  
 getExpenseData();  
 getMainTrader();  
 status ="traderPayMethod";  
 break;  
 }  
 }  
  
 @Override  
 public void onClick(View view) {  
  
 try {  
 int id =view.getId();  
 if(binding.buAddEWSave.getId()==id){  
 String desc ="";  
 double value =0;  
 double payed =0;  
 if(binding.name.getText().length()==0){  
 binding.name.setHint("يجب ملئ هذا الحقل");  
 }else binding.name.setError(null);  
 if(binding.desc.getText().length()>0){  
 desc = binding.desc.getText().toString();  
 }  
 if(binding.value.getText().length()==0){  
 binding.value.setError("يجب ملئ هذا الحقل");  
 }else {  
 value = Double.valueOf(binding.value.getText().toString());  
 binding.value.setError(null);  
 }  
 if(binding.payed.getText().length()==0){  
 payed =0;  
 binding.payed.setText("0");  
 }else{  
 payed = Double.valueOf(binding.payed.getText().toString());  
 }  
  
 if(binding.name.getError()==null && binding.value.getError()==null){  
 if(status.equals("addExpense")){  
 mainTraderModel.setBalance(payed-value);  
 ExpenseModel expenseModel = new ExpenseModel(mainTraderModel,desc,value,payed);  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Expenses");  
 reference.push().setValue(expenseModel);  
  
 addProjectExpense();  
 addTraderBalance();  
 Intent intent = new Intent(AddExpenseOrDebts.this, Expense.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }else if(status.equals("editExpense")){  
 Intent intent1 =getIntent();  
 String expenseKey = intent1.getStringExtra("expenseKey");  
 editProjectExpense();  
 editTraderBalance();  
 ExpenseModel expenseModel = new ExpenseModel(mainTraderModel,desc,value,payed);  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Expenses").child(expenseKey);  
 reference.setValue(expenseModel);  
  
 Intent intent = new Intent(AddExpenseOrDebts.this, Expense.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }else if(status.equals("addWages")){  
 mainWorkerModel.setBalance(payed-value);  
 WagesModel wagesModel = new WagesModel(mainWorkerModel,desc,value,payed);  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Wages");  
 reference.push().setValue(wagesModel);  
 addProjectExpense();  
 addWorkerBalance();  
 Intent intent = new Intent(AddExpenseOrDebts.this, Wages.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }else if(status.equals("editWages")){  
 Intent intent1 =getIntent();  
 String wagesKey = intent1.getStringExtra("wagesKey");  
 editProjectWages();  
 editWorkerBalance();  
 WagesModel wagesModel = new WagesModel(mainWorkerModel,desc,value,payed);  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Wages").child(wagesKey);  
 reference.setValue(wagesModel);  
 Intent intent = new Intent(AddExpenseOrDebts.this, Wages.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }else if(status.equals("workerPayMethod")){  
 Intent intent1 = getIntent();  
 String wagesKey = intent1.getStringExtra("wagesKey");  
 payWorkerMethod();  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Wages").child(wagesKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double oldPayed = snapshot.getValue(WagesModel.class).getPayed();  
 snapshot.getRef().child("payed").setValue(Double.valueOf(binding.payed.getText().toString())+oldPayed);  
 double oldDebts =snapshot.getValue(WagesModel.class).getDebts();  
 snapshot.getRef().child("debts").setValue(Double.valueOf(binding.payed.getText().toString())+oldDebts);  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 Intent intent = new Intent(AddExpenseOrDebts.this, Wages.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }else if(status.equals("traderPayMethod")){  
 Intent intent1 = getIntent();  
 String expenseKey = intent1.getStringExtra("expenseKey");  
 payTraderMethod();  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Expenses").child(expenseKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double oldPayed = snapshot.getValue(ExpenseModel.class).getPayed();  
 snapshot.getRef().child("payed").setValue(Double.valueOf(binding.payed.getText().toString())+oldPayed);  
 double oldDebts =snapshot.getValue(ExpenseModel.class).getDebts();  
 snapshot.getRef().child("debts").setValue(Double.valueOf(binding.payed.getText().toString())+oldDebts);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 Intent intent = new Intent(AddExpenseOrDebts.this, Expense.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }  
 }  
  
 }else if(binding.name.getId()==id){  
 if(status.equals("addExpense")){  
 binding.scrollView.setVisibility(View.GONE);  
 binding.fragmentContainer.setVisibility(View.VISIBLE);  
 Fragment fragment = new SelectTrader();  
 FragmentManager fragmentManager = getSupportFragmentManager();  
 fragmentManager.beginTransaction().add(R.id.fragmentContainer, fragment).commit();  
  
 }else if(status.equals("addWages")){  
 binding.scrollView.setVisibility(View.GONE);  
 binding.fragmentContainer.setVisibility(View.VISIBLE);  
 Fragment fragment = new SelectWorker();  
 FragmentManager fragmentManager = getSupportFragmentManager();  
 fragmentManager.beginTransaction().add(R.id.fragmentContainer, fragment).commit();  
 }  
 }else if(id==binding.logo.getId()||id==binding.linearLayout.getId()||id==binding.constraintLayout.getId()){  
 hideKeyBoard(view);  
 }  
  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
  
 public void closeFragment(){  
 FragmentManager fragmentManager ;  
 fragmentManager = getSupportFragmentManager();  
 FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();  
 Fragment fragment = fragmentManager.findFragmentById(R.id.fragmentContainer);  
 if(fragment!=null){  
 fragmentTransaction.remove(fragment);  
 fragmentTransaction.commit();}  
 }  
 @Override  
 public void getNameKey(String key) {  
 if(status.equals("addExpense")){  
 closeFragment();  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Traders").child(key);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if(snapshot.exists()){  
 TraderModel traderModel = snapshot.getValue(TraderModel.class);  
 traderModel.setKey(snapshot.getKey());  
 binding.name.setText(traderModel.getName());  
 mainTraderModel = traderModel;  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
  
 }else if (status.equals("addWages")){  
 closeFragment();  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Workers").child(key);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if(snapshot.exists()){  
 WorkerModel workerModel = snapshot.getValue(WorkerModel.class);  
 workerModel.setKey(snapshot.getKey());  
 binding.name.setText(workerModel.getName());  
 mainWorkerModel = workerModel;  
 }  
  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 binding.fragmentContainer.setVisibility(View.GONE);  
 binding.scrollView.setVisibility(View.VISIBLE);  
 }  
 private void getProjectName(){  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 String name =snapshot.getValue(ProjectModel.class).getName();  
 binding.projectName.setText(name);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 }  
 private void setExpenseView(){  
 getProjectName();  
 binding.name.setHint(R.string.traderName);  
 binding.desc.setHint(R.string.productDescription);  
 binding.value.setHint(R.string.value);  
 binding.payed.setHint(R.string.payed);  
 }  
 private void getExpenseData(){  
 Intent intent =getIntent();  
 String expenseKey = intent.getStringExtra("expenseKey");  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Expenses").child(expenseKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 ExpenseModel expenseModel = snapshot.getValue(ExpenseModel.class);  
 expenseModel.setKey(snapshot.getKey());  
 mainTraderModel = expenseModel.getTraderModel();  
 binding.name.setText(expenseModel.getTraderModel().getName());  
 binding.desc.setText(expenseModel.getDescription());  
 binding.value.setText(String.valueOf(expenseModel.getValue()));  
 if(status.equals("editExpense")) binding.payed.setText(String.valueOf(expenseModel.getPayed()));  
 else if(status.equals("traderPayMethod")) {  
 if(expenseModel.getDebts()==0){  
 binding.payed.setHint(R.string.theirIsNoDebts);  
 }else if(expenseModel.getDebts()<0){  
 binding.payed.setHint( "المديونية"+ " : " + expenseModel.getDebts()\*-1 );  
 }else if(expenseModel.getDebts()>0){  
 binding.payed.setHint( "الرصيد" + " : " + expenseModel.getDebts());  
 }  
 }  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 private void addProjectExpense(){  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double expense = Double.parseDouble(binding.value.getText().toString());  
 double oldExpense = snapshot.getValue(ProjectModel.class).getExpense();  
 snapshot.getRef().child("expense").setValue(expense+oldExpense);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 private void editProjectExpense(){  
 Intent intent =getIntent();  
 String expenseKey = intent.getStringExtra("expenseKey");  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Expenses").child(expenseKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double oldExpense =snapshot.getValue(ExpenseModel.class).getValue();  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double projectOldExpense = snapshot.getValue(ProjectModel.class).getExpense();  
 double newExpense = Double.parseDouble(binding.value.getText().toString());  
 snapshot.getRef().child("expense").setValue(projectOldExpense-oldExpense+newExpense);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
  
 }  
 private void addTraderBalance(){  
 double debts = Double.parseDouble(binding.payed.getText().toString())-Double.parseDouble(binding.value.getText().toString());  
 String traderKey =mainTraderModel.getKey();  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Traders").child(traderKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double oldBalance = snapshot.getValue(TraderModel.class).getBalance();  
 snapshot.getRef().child("balance").setValue(debts+oldBalance);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 }  
 private void editTraderBalance(){  
 Intent intent =getIntent();  
 String expenseKey = intent.getStringExtra("expenseKey");  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Expenses").child(expenseKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 ExpenseModel expenseModel =snapshot.getValue(ExpenseModel.class);  
 double dept = expenseModel.getValue()-expenseModel.getPayed();  
 String traderKey = expenseModel.getTraderModel().getKey();  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Traders").child(traderKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double traderBalance =snapshot.getValue(TraderModel.class).getBalance();  
 double value = Double.valueOf(binding.value.getText().toString());  
 double payed = Double.valueOf(binding.payed.getText().toString());  
 double newDebt = payed-value;  
 snapshot.getRef().child("balance").setValue(traderBalance+dept+newDebt);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Expenses").child(expenseKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double traderBalance =snapshot.getValue(ExpenseModel.class).getTraderModel().getBalance();  
 double value = Double.valueOf(binding.value.getText().toString());  
 double payed = Double.valueOf(binding.payed.getText().toString());  
 double newDebt = payed-value;  
 snapshot.getRef().child("traderModel").child("balance").setValue(traderBalance+dept+newDebt);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
  
 }  
 private void getMainTrader(){  
 Intent intent1 =getIntent();  
 String expenseKey = intent1.getStringExtra("expenseKey");  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Expenses").child(expenseKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 mainTraderModel =snapshot.getValue(ExpenseModel.class).getTraderModel();  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 }  
  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 private void setWagesView(){  
 getProjectName();  
 binding.name.setHint(R.string.workerName);  
 binding.desc.setHint(R.string.description);  
 binding.value.setHint(R.string.value);  
 binding.payed.setHint(R.string.payed);  
 }  
 private void getMainWorker(){  
 Intent intent1 =getIntent();  
 String wagesKey = intent1.getStringExtra("wagesKey");  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Wages").child(wagesKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 mainWorkerModel =snapshot.getValue(WagesModel.class).getWorkerModel();  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 }  
 private void getWagesData(){  
 Intent intent =getIntent();  
 String wagesKey = intent.getStringExtra("wagesKey");  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Wages").child(wagesKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 WagesModel wagesModel = snapshot.getValue(WagesModel.class);  
 wagesModel.setKey(snapshot.getKey());  
 mainWorkerModel = wagesModel.getWorkerModel();  
 mainWagesModel =wagesModel;  
 binding.name.setText(wagesModel.getWorkerModel().getName());  
 binding.desc.setText(wagesModel.getDescription());  
 binding.value.setText(String.valueOf(wagesModel.getValue()));  
  
 if(status.equals("editWages")) binding.payed.setText(String.valueOf(wagesModel.getPayed()));  
 else if(status.equals("workerPayMethod")){  
 if(wagesModel.getDebts()==0){  
 binding.payed.setHint(R.string.theirIsNoDebts);  
 }else if(wagesModel.getDebts()<0){  
 binding.payed.setHint( "المديونية"+ " : " + wagesModel.getDebts()\*-1 );  
 }else if(wagesModel.getDebts()>0){  
 binding.payed.setHint( "الرصيد" + " : " + wagesModel.getDebts());  
 }  
 }  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
  
 }  
 private void addWorkerBalance(){  
 double debts = Double.parseDouble(binding.payed.getText().toString())-Double.parseDouble(binding.value.getText().toString());  
 String workerKey =mainWorkerModel.getKey();  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Workers").child(workerKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double oldBalance = snapshot.getValue(WorkerModel.class).getBalance();  
 snapshot.getRef().child("balance").setValue(debts+oldBalance);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 }  
 private void editProjectWages(){  
 Intent intent =getIntent();  
 String wagesKey = intent.getStringExtra("wagesKey");  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Wages").child(wagesKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double oldWages = snapshot.getValue(WagesModel.class).getValue();  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double projectOldExpense = snapshot.getValue(ProjectModel.class).getExpense();  
 double newExpense = Double.parseDouble(binding.value.getText().toString());  
 snapshot.getRef().child("expense").setValue(projectOldExpense-oldWages+newExpense);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
  
 }  
 private void editWorkerBalance(){  
 Intent intent =getIntent();  
 String wagesKey = intent.getStringExtra("wagesKey");  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Wages").child(wagesKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if(snapshot.exists()){  
 WagesModel wagesModel =snapshot.getValue(WagesModel.class);  
 double dept = wagesModel.getValue()-wagesModel.getPayed();  
 String workerKey = wagesModel.getWorkerModel().getKey();  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Workers").child(workerKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if(snapshot.exists()){  
 double workerBalance =snapshot.getValue(WorkerModel.class).getBalance();  
 double value = Double.valueOf(binding.value.getText().toString());  
 double payed = Double.valueOf(binding.payed.getText().toString());  
 double newDebt = payed-value;  
 snapshot.getRef().child("balance").setValue(workerBalance+dept+newDebt);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Wages").child(wagesKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double workerBalance =snapshot.getValue(WagesModel.class).getWorkerModel().getBalance();  
 double value = Double.valueOf(binding.value.getText().toString());  
 double payed = Double.valueOf(binding.payed.getText().toString());  
 double newDebt = payed-value;  
 snapshot.getRef().child("workerModel").child("balance").setValue(workerBalance+dept+newDebt);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
  
 }  
 private void payWorkerMethod(){  
 Intent intent =getIntent();  
 String wagesKey = intent.getStringExtra("wagesKey");  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Wages").child(wagesKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double newPayed =Double.parseDouble(binding.payed.getText().toString());  
 double workerOldBalance = snapshot.getValue(WagesModel.class).getWorkerModel().getBalance();  
 String workerKey = snapshot.getValue(WagesModel.class).getWorkerModel().getKey();  
 snapshot.getRef().child("workerModel").child("balance").setValue(workerOldBalance+newPayed);  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Workers").child(workerKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if(snapshot.exists()){  
 double newPayed =Double.parseDouble(binding.payed.getText().toString());  
 double workerOldBalance = snapshot.getValue(WorkerModel.class).getBalance();  
 snapshot.getRef().child("balance").setValue(workerOldBalance+newPayed);  
 }  
  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 private void payTraderMethod(){  
 Intent intent =getIntent();  
 String expenseKey = intent.getStringExtra("expenseKey");  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Expenses").child(expenseKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double newPayed =Double.parseDouble(binding.payed.getText().toString());  
 double traderOldBalance = snapshot.getValue(ExpenseModel.class).getTraderModel().getBalance();  
 String traderKey = snapshot.getValue(ExpenseModel.class).getTraderModel().getKey();  
 snapshot.getRef().child("traderModel").child("balance").setValue(traderOldBalance+newPayed);  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Traders").child(traderKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if(snapshot.exists()){  
 double newPayed =Double.parseDouble(binding.payed.getText().toString());  
 double traderOldBalance = snapshot.getValue(TraderModel.class).getBalance();  
 snapshot.getRef().child("balance").setValue(traderOldBalance+newPayed);  
 }  
  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
  
  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 try {  
 if(status.equals("addExpense")){  
 Intent intent = new Intent(AddExpenseOrDebts.this, Expense.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }else if(status.equals("editExpense")){  
 Intent intent = new Intent(AddExpenseOrDebts.this, Expense.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }else if(status.equals("addWages")){  
 Intent intent = new Intent(AddExpenseOrDebts.this, Wages.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }else if(status.equals("editWages")){  
 Intent intent = new Intent(AddExpenseOrDebts.this, Wages.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }else if(status.equals("workerPayMethod")){  
 Intent intent = new Intent(AddExpenseOrDebts.this, Wages.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
  
 }  
 private void setPayMethodView(){  
 getProjectName();  
 binding.name.setBackgroundResource(R.drawable.layout\_design\_gray);  
 binding.value.setBackgroundResource(R.drawable.layout\_design\_gray);  
 binding.desc.setBackgroundResource(R.drawable.layout\_design\_gray);  
  
 binding.name.setFocusable(false);  
 binding.name.setFocusableInTouchMode(false);  
 binding.name.setClickable(false);  
 binding.name.setCursorVisible(false);  
  
 binding.desc.setFocusable(false);  
 binding.desc.setFocusableInTouchMode(false);  
 binding.desc.setClickable(false);  
 binding.desc.setCursorVisible(false);  
  
 binding.value.setFocusable(false);  
 binding.value.setFocusableInTouchMode(false);  
 binding.value.setClickable(false);  
 binding.value.setCursorVisible(false);  
  
 }  
 public void hideKeyBoard(View v){  
 InputMethodManager imm = (InputMethodManager)getSystemService(Context.INPUT\_METHOD\_SERVICE);  
 imm.hideSoftInputFromWindow(v.getWindowToken(), 0);  
 }  
  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.setDefault(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
}

package com.example.bit\_x.ui;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.fragment.app.Fragment;  
import androidx.fragment.app.FragmentManager;  
import androidx.fragment.app.FragmentTransaction;  
  
import android.content.Context;  
import android.content.Intent;  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.view.View;  
import android.view.inputmethod.InputMethodManager;  
  
import com.example.bit\_x.R;  
import com.example.bit\_x.databinding.ActivityAddTraderOrWorkerBinding;  
import com.example.bit\_x.interfaces.GetNameKey;  
import com.example.bit\_x.models.ClientModel;  
import com.example.bit\_x.models.IncomeModel;  
import com.example.bit\_x.models.ProjectModel;  
import com.example.bit\_x.models.TraderModel;  
import com.example.bit\_x.models.WorkerModel;  
import com.example.bit\_x.ui.fragment.SelectClient;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.Locale;  
  
public class AddTraderOrWorker extends AppCompatActivity implements View.OnClickListener , GetNameKey {  
 ActivityAddTraderOrWorkerBinding binding;  
 String status="";  
 private DatabaseReference databaseReference;  
 private FirebaseAuth auth ;  
 private FirebaseUser user;  
 private ClientModel mainClientModel;  
 private String projectKey="";  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding= ActivityAddTraderOrWorkerBinding.*inflate*(getLayoutInflater());  
 View view = binding.getRoot();  
 this.setContentView(view);  
 //setAppLanguage();  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
  
  
 binding.logo.setOnClickListener(this);  
 binding.mainConstraintLayout.setOnClickListener(this);  
 binding.constraintLayout.setOnClickListener(this);  
 binding.secondField.setOnClickListener(this);  
 binding.buAddTWSave.setOnClickListener(this);  
 Intent intent = getIntent();  
 String id = intent.getStringExtra("tag");  
 projectKey =intent.getStringExtra("projectKey");  
 switch (id){  
 case "addTrader":  
 databaseReference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Traders");  
 binding.projectName.setVisibility(View.*GONE*);  
 binding.firstField.setHint(R.string.*traderName*);  
 binding.secondField.setHint(R.string.*mobilePhone*);  
 status ="addTrader";  
 break;  
 case "editTrader":  
 binding.projectName.setVisibility(View.*GONE*);  
 binding.firstField.setHint(R.string.*traderName*);  
 binding.secondField.setHint(R.string.*mobilePhone*);  
 Intent intent1 = getIntent();  
 String traderKey = intent1.getStringExtra("traderKey");  
 databaseReference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Traders").child(traderKey);  
 databaseReference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 TraderModel traderModel = snapshot.getValue(TraderModel.class);  
 binding.firstField.setText(traderModel.getName());  
 binding.secondField.setText(traderModel.getPhoneNumber());  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 status ="editTrader";  
 break;  
 case "addWorker":  
 databaseReference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Workers");  
 binding.projectName.setVisibility(View.*GONE*);  
 binding.firstField.setHint(R.string.*workerName*);  
 binding.secondField.setHint(R.string.*mobilePhone*);  
 status ="addWorker";  
 break;  
 case "editWorker":  
 binding.projectName.setVisibility(View.*GONE*);  
 binding.firstField.setHint(R.string.*workerName*);  
 binding.secondField.setHint(R.string.*mobilePhone*);  
  
 String workerKey = intent.getStringExtra("workerKey");  
 databaseReference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Workers").child(workerKey);  
 databaseReference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 WorkerModel workerModel = snapshot.getValue(WorkerModel.class);  
 binding.firstField.setText(workerModel.getName());  
 binding.secondField.setText(workerModel.getPhoneNumber());  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 status ="editWorker";  
 break;  
 case "addIncome":  
 binding.projectName.setVisibility(View.*VISIBLE*);  
 binding.firstField.setHint(R.string.*description*);  
 binding.secondField.setHint(R.string.*value*);  
 databaseReference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 ProjectModel projectModel =snapshot.getValue(ProjectModel.class);  
 binding.projectName.setText(projectModel.getName());  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 databaseReference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Incomes");  
 status ="addIncome";  
 break;  
 case "editIncome":  
 String incomeKey = intent.getStringExtra("incomeKey");  
 binding.projectName.setVisibility(View.GONE);  
 binding.firstField.setHint(R.string.description);  
 binding.secondField.setHint(R.string.value);  
 System.out.println(incomeKey);  
 System.out.println(projectKey);  
  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Incomes").child(incomeKey);  
 databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()) {  
 IncomeModel incomeModel = snapshot.getValue(IncomeModel.class);  
 binding.firstField.setText(incomeModel.getName());  
 binding.secondField.setText(String.valueOf(incomeModel.getBalance()));  
 }else System.out.println("error");  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
  
 status ="editIncome";  
 break;  
 case "addClient":  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Clients");  
 binding.projectName.setVisibility(View.GONE);  
 binding.firstField.setHint(R.string.clientName);  
 binding.secondField.setHint(R.string.mobilePhone);  
 status ="addClient";  
 break;  
 case "editClient":  
 String clientKey = intent.getStringExtra("clientKey");  
 binding.projectName.setVisibility(View.GONE);  
 binding.firstField.setHint(R.string.clientName);  
 binding.secondField.setHint(R.string.mobilePhone);  
  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Clients").child(clientKey);  
 databaseReference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 ClientModel clientModel = snapshot.getValue(ClientModel.class);  
 binding.firstField.setText(clientModel.getName());  
 binding.secondField.setText(clientModel.getPhoneNumber());  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 status ="editClient";  
 break;  
 case "addProject":  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects");  
 setProjectView();  
 status ="addProject";  
 break;  
 case "editProject":  
 setProjectView();  
 binding.secondField.setBackgroundResource(R.drawable.layout\_design\_gray);  
 getMainClientModel();  
 status ="editProject";  
 break;  
 case "editTraderDebts":  
 status="editTraderDebts";  
 break;  
 case "editWorkerDebts":  
 binding.projectName.setVisibility(View.VISIBLE);  
 binding.firstField.setFocusable(false);  
 binding.firstField.setFocusableInTouchMode(false);  
 binding.firstField.setClickable(false);  
 binding.firstField.setCursorVisible(false);  
 binding.secondField.setHint(R.string.value);  
 binding.firstField.setBackgroundResource(R.drawable.layout\_design\_gray);  
 String workerKey1 = intent.getStringExtra("workerKey");  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Workers").child(workerKey1);  
 databaseReference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 WorkerModel workerModel = snapshot.getValue(WorkerModel.class);  
 binding.firstField.setText(workerModel.getName());  
 binding.secondField.setHint(workerModel.getBalance()\*-1+" : المديونية");  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 status="editWorkerDebts";  
 break;  
 }  
 }  
 @Override  
 public void onClick(View view) {  
  
 try {  
 if (view.getId() == binding.buAddTWSave.getId()) {  
 String firstField = binding.firstField.getText().toString();  
 String secondField = binding.secondField.getText().toString();  
 if (firstField.length() == 0) {  
 binding.firstField.setError("يجب ملئ هذا الحقل");  
 } else binding.firstField.setError(null);  
 if(status.equals("addProject")||status.equals("editProject")){  
 if (secondField.length()==0){  
 binding.secondField.setError("يجب ملئ هذا الحقل");  
 }else binding.secondField.setError(null);  
 }  
  
 if (binding.firstField.getError() == null && binding.secondField.getError()==null) {  
 if (status.equals("addTrader")) {  
 TraderModel traderModel = new TraderModel(firstField, secondField);  
 databaseReference.push().setValue(traderModel);  
  
 Intent intent = new Intent(AddTraderOrWorker.this, Traders.class);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("addWorker")) {  
 WorkerModel workerModel = new WorkerModel(firstField, secondField);  
 databaseReference.push().setValue(workerModel);  
 Intent intent = new Intent(AddTraderOrWorker.this, Workers.class);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("addIncome")) {  
 String date = getCurrentTime();  
 double income =Double.valueOf(secondField);  
 IncomeModel incomeModel = new IncomeModel(firstField,income , date);  
 databaseReference.push().setValue(incomeModel);  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
  
 databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double t = snapshot.getValue(ProjectModel.class).getIncome();  
 double t1 = Double.valueOf(binding.secondField.getText().toString());  
 snapshot.getRef().child("income").setValue(t+t1);  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
  
 controlClientBalance(projectKey,Double.valueOf(secondField),"+",0);  
 controlClientBalanceInProject(projectKey,Double.valueOf(secondField),"+",0);  
  
 Intent intent = new Intent(AddTraderOrWorker.this, Income.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }else if (status.equals("editIncome")) {  
 Intent intent1 =getIntent();  
 String incomeKey =intent1.getStringExtra("incomeKey");  
 editProjectIncome(projectKey,incomeKey);  
 double newIncomeBalance = Double.parseDouble(binding.secondField.getText().toString());  
 editClientIncome(projectKey,incomeKey,newIncomeBalance);  
  
 //to save  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Incomes").child(incomeKey);  
 IncomeModel incomeModel = new IncomeModel(firstField, Double.valueOf(secondField),getCurrentTime());  
 databaseReference.setValue(incomeModel);  
  
 Intent intent = new Intent(AddTraderOrWorker.this, Income.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("editTrader")) {  
 TraderModel traderModel = new TraderModel(firstField, secondField);  
 databaseReference.setValue(traderModel);  
 Intent intent = new Intent(AddTraderOrWorker.this, Traders.class);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("editWorker")) {  
 WorkerModel workerModel = new WorkerModel(firstField, secondField);  
 databaseReference.setValue(workerModel);  
 Intent intent = new Intent(AddTraderOrWorker.this, Workers.class);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("addClient")) {  
 ClientModel clientModel = new ClientModel(firstField, secondField);  
 databaseReference.push().setValue(clientModel);  
 Intent intent = new Intent(AddTraderOrWorker.this, Clients.class);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("editClient")) {  
 ClientModel clientModel = new ClientModel(firstField, secondField);  
 databaseReference.setValue(clientModel);  
 Intent intent = new Intent(AddTraderOrWorker.this, Clients.class);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("addProject")) {  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects");  
 ProjectModel projectModel = new ProjectModel(firstField, mainClientModel);  
 databaseReference.push().setValue(projectModel);  
 Intent intent = new Intent(AddTraderOrWorker.this, Projects.class);  
 startActivity(intent);  
 finish();  
 }else if (status.equals("editProject")) {  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("clientModel");  
 databaseReference.setValue(mainClientModel);  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 snapshot.getRef().child("name").setValue(binding.firstField.getText().toString());  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
  
 Intent intent = new Intent(AddTraderOrWorker.this, Projects.class);  
 startActivity(intent);  
 finish();  
 }else if(status.equals("editWorkerDebts")){  
 Intent intent =getIntent();  
 String workerKey = intent.getStringExtra("workerKey");  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Workers").child(workerKey);  
 databaseReference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double balance = snapshot.getValue(WorkerModel.class).getBalance();  
 snapshot.getRef().child("balance").setValue(secondField+balance);  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Wages");  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
  
  
 }  
 }  
 } else if (binding.secondField.getId() == view.getId()) {  
 if (status == "addProject") {  
 binding.constraintLayout.setVisibility(View.GONE);  
 Fragment fragment = new SelectClient();  
 FragmentManager fragmentManager = getSupportFragmentManager();  
 fragmentManager.beginTransaction().add(R.id.clientFragmentContainer, fragment).commit();  
 }  
 }else if(view.getId()==binding.logo.getId()||view.getId()==binding.mainConstraintLayout.getId()||view.getId()==binding.constraintLayout.getId()){  
 hideKeyBoard(view);  
 }  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
  
  
 @Override  
 public void getNameKey(String key) {  
 closeFragment();  
 binding.constraintLayout.setVisibility(View.VISIBLE);  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Clients").child(key);  
 databaseReference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if(snapshot.exists()){  
 ClientModel clientModel = snapshot.getValue(ClientModel.class);  
 clientModel.setKey(snapshot.getKey());  
 binding.secondField.setText(clientModel.getName());  
 mainClientModel = clientModel;  
  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
  
 }  
 private String getCurrentTime(){  
 SimpleDateFormat time = new SimpleDateFormat("dd/MM/yyyy", Locale.getDefault());  
 String currentDate = time.format(new Date());  
 return currentDate;  
 }  
 public void closeFragment(){  
 FragmentManager fragmentManager ;  
 fragmentManager = getSupportFragmentManager();  
 FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();  
 Fragment fragment = fragmentManager.findFragmentById(R.id.clientFragmentContainer);  
 if(fragment!=null){  
 fragmentTransaction.remove(fragment);  
 fragmentTransaction.commit();}  
 }  
 private void controlClientBalanceInProject(String keyProject ,double incomeNewBalance ,String operation ,double incomeOldBalance){  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(keyProject).child("clientModel");  
 databaseReference .addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double clientOldBalance = snapshot.getValue(ClientModel.class).getBalance();  
 double totalNewBalance=0;  
 if(operation.equals("+")){  
 totalNewBalance = incomeNewBalance + clientOldBalance;  
 }else if(operation.equals("-")){  
 totalNewBalance = clientOldBalance-incomeNewBalance;  
 }else if(operation.equals("/")){//edit  
 totalNewBalance = clientOldBalance - incomeOldBalance + incomeNewBalance;  
 }  
 snapshot.getRef().child("balance").setValue(totalNewBalance);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
  
 }  
 private void controlClientBalance(String keyProject,double incomeNewBalance ,String operation,double incomeOldBalance) {  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(keyProject).child("clientModel");  
 databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 ClientModel clientModel = snapshot.getValue(ClientModel.class);  
 String key = clientModel.getKey();  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Clients").child(key);  
 databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 if(snapshot.exists()){  
 double clientOldBalance = snapshot.getValue(ClientModel.class).getBalance();  
 double totalNewBalance = 0;  
 if (operation == "+") {  
 totalNewBalance = incomeNewBalance + clientOldBalance;  
 } else if (operation == "-") {  
 totalNewBalance = clientOldBalance - incomeNewBalance;  
 }else if(operation.equals("/")){//edit  
 totalNewBalance = clientOldBalance - incomeOldBalance + incomeNewBalance;  
 }  
 snapshot.getRef().child("balance").setValue(totalNewBalance);  
 }  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 private void editClientIncome(String projectKey,String incomeKey,double newIncomeBalance){  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Incomes").child(incomeKey);  
 databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double incomeOldBalance = snapshot.getValue(IncomeModel.class).getBalance();  
 controlClientBalance(projectKey,newIncomeBalance,"/",incomeOldBalance);  
 controlClientBalanceInProject(projectKey,newIncomeBalance,"/",incomeOldBalance);  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 private void editProjectIncome(String projectKey ,String incomeKey){  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Incomes").child(incomeKey);  
 databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 double balance1 = snapshot.getValue(IncomeModel.class).getBalance();  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 double t = snapshot.getValue(ProjectModel.class).getIncome();  
 double newBalance = Double.parseDouble(binding.secondField.getText().toString());  
 snapshot.getRef().child("income").setValue(t-balance1+newBalance);  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
  
 }  
 private void getMainClientModel(){  
 databaseReference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 ProjectModel projectModel = snapshot.getValue(ProjectModel.class);  
 binding.firstField.setText(projectModel.getName());  
 mainClientModel = projectModel.getClientModel();  
 mainClientModel.setKey(projectModel.getClientModel().getKey());  
 binding.secondField.setText(projectModel.getClientModel().getName());  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 }  
  
  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 try {  
 if (status.equals("addTrader")) {  
 Intent intent = new Intent(AddTraderOrWorker.this, Traders.class);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("addWorker")) {  
 Intent intent = new Intent(AddTraderOrWorker.this, Workers.class);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("addIncome")) {  
 Intent intent = new Intent(AddTraderOrWorker.this, Income.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }else if (status.equals("editIncome")) {  
 Intent intent = new Intent(AddTraderOrWorker.this, Income.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("editTrader")) {  
 Intent intent = new Intent(AddTraderOrWorker.this, Traders.class);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("editWorker")) {  
 Intent intent = new Intent(AddTraderOrWorker.this, Workers.class);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("addClient")) {  
 Intent intent = new Intent(AddTraderOrWorker.this, Clients.class);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("editClient")) {  
 Intent intent = new Intent(AddTraderOrWorker.this, Clients.class);  
 startActivity(intent);  
 finish();  
 } else if (status.equals("addProject")) {  
 Intent intent = new Intent(AddTraderOrWorker.this, Projects.class);  
 startActivity(intent);  
 finish();  
 }else if (status.equals("editProject")) {  
 Intent intent = new Intent(AddTraderOrWorker.this, Projects.class);  
 startActivity(intent);  
 finish();  
 }else if (status.equals("editWorkerDebts")) {  
 Intent intent = new Intent(AddTraderOrWorker.this, Debts.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
  
 private void setProjectView(){  
 binding.projectName.setVisibility(View.GONE);  
 binding.firstField.setHint(R.string.projectName);  
 binding.secondField.setHint(R.string.clientName);  
 binding.secondField.setFocusable(false);  
 binding.secondField.setFocusableInTouchMode(false);  
 binding.secondField.setClickable(false);  
 binding.secondField.setCursorVisible(false);  
 }  
 public void hideKeyBoard(View v){  
 InputMethodManager imm = (InputMethodManager)getSystemService(Context.INPUT\_METHOD\_SERVICE);  
 imm.hideSoftInputFromWindow(v.getWindowToken(), 0);  
 }  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.setDefault(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
  
}

package com.example.bit\_x.ui;  
  
import static android.content.ContentValues.*TAG*;  
  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.content.Intent;  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.MenuItem;  
import android.view.View;  
  
import com.example.bit\_x.adabter.ClientAdapter;  
import com.example.bit\_x.databinding.ActivityClientsBinding;  
import com.example.bit\_x.interfaces.RecyclerViewInterface;  
import com.example.bit\_x.models.ClientModel;  
import com.google.android.material.snackbar.Snackbar;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.ArrayList;  
import java.util.Locale;  
  
public class Clients extends AppCompatActivity {  
 private RecyclerViewInterface listener;  
  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 private FirebaseUser user;  
 ClientAdapter adapter;  
 ActivityClientsBinding binding;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding =ActivityClientsBinding.*inflate*(getLayoutInflater());  
 View view = binding.getRoot();  
 this.setContentView(view);  
 //setAppLanguage();  
 binding.shimmer.startShimmer();  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
 //set Adapter  
 setOnClickListener();  
 getClients();  
  
  
 binding.floatingButtonClient.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 try {  
 Intent intent = new Intent(Clients.this, AddTraderOrWorker.class);  
 intent.putExtra("tag","addClient");  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 });  
 }  
  
 private void setOnClickListener() {  
 listener =new RecyclerViewInterface() {  
 @Override  
 public void onItemClick(View v, int pos) {  
 Log.*i*(*TAG*, "onItemClick: "+pos);  
 }  
  
 };  
 }  
 private void getClients(){  
 ArrayList<ClientModel> adapterList = new ArrayList<>();  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Clients");  
 reference.keepSynced(true);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 adapterList.clear();  
 for (DataSnapshot data : snapshot.getChildren()) {  
 ClientModel clientModel = data.getValue(ClientModel.class);  
 clientModel.setKey(data.getKey());  
 adapterList.add(clientModel);  
 }  
 setAdapter(adapterList);  
 binding.shimmer.setVisibility(View.*GONE*);  
 binding.shimmer.stopShimmer();  
 binding.imageView.setVisibility(View.*VISIBLE*);  
 binding.fabWrapper.setVisibility(View.*VISIBLE*);  
 if(adapterList.size()>0){  
 binding.clientRecycleView.setVisibility(View.*VISIBLE*);  
 binding.emptyIcon.setVisibility(View.*GONE*);  
 }else{  
 binding.clientRecycleView.setVisibility(View.*GONE*);  
 binding.emptyIcon.setVisibility(View.*VISIBLE*);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 private void setAdapter(ArrayList<ClientModel> clientList){  
 adapter = new ClientAdapter();  
 binding.clientRecycleView.setAdapter(adapter);  
 adapter.setList(clientList,listener);  
 LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this);  
 linearLayoutManager.setReverseLayout(true);  
 linearLayoutManager.setStackFromEnd(true);  
 binding.clientRecycleView.setLayoutManager(linearLayoutManager);  
 }  
  
 @Override  
 public boolean onContextItemSelected(@NonNull MenuItem item) {  
 switch (item.getItemId()){  
 case 120:  
 try {  
 adapter.editItem(item.getGroupId(),Clients.this);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 case 121:  
 try {  
 adapter.removeItem(item.getGroupId(),user,Clients.this);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 default:  
 return super.onContextItemSelected(item);  
 }  
 }  
  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 try {  
 Intent intent = new Intent(Clients.this, MainActivity.class);  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.*setDefault*(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
  
}

package com.example.bit\_x.ui;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.fragment.app.FragmentPagerAdapter;  
  
import android.content.Intent;  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.view.View;  
  
import com.example.bit\_x.R;  
import com.example.bit\_x.adabter.TabLayoutAdapter;  
import com.example.bit\_x.databinding.ActivityDebtsBinding;  
import com.example.bit\_x.interfaces.GetNameKey;  
import com.example.bit\_x.models.ProjectModel;  
import com.example.bit\_x.ui.fragment.TradersFragment;  
import com.example.bit\_x.ui.fragment.workerFragment;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.Locale;  
  
public class Debts extends AppCompatActivity {  
 public static String projectKey="";  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 private FirebaseUser user;  
 ActivityDebtsBinding binding;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding = ActivityDebtsBinding.*inflate*(getLayoutInflater());  
 View view = binding.getRoot();  
 this.setContentView(view);  
 // setAppLanguage();  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
 Intent intent =getIntent();  
 *projectKey* = intent.getStringExtra("projectKey");  
 binding.tabLayout.setupWithViewPager(binding.viewPager);  
 TabLayoutAdapter adapter =new TabLayoutAdapter(getSupportFragmentManager(), FragmentPagerAdapter.*BEHAVIOR\_RESUME\_ONLY\_CURRENT\_FRAGMENT*);  
 adapter.addFragment(new workerFragment(), "الصنايعية");  
 adapter.addFragment(new TradersFragment(),"التجار");  
 binding.viewPager.setAdapter(adapter);  
 getProjectNameAndExpenses();  
 }  
 public void getProjectNameAndExpenses(){  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Projects").child(*projectKey*);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 ProjectModel projectModel= snapshot.getValue(ProjectModel.class);  
 binding.proName.setText(projectModel.getName());  
 double expense = projectModel.getExpense();  
 System.*out*.println(expense);  
 binding.proBalance.setText(String.*valueOf*((expense/100\*15)+expense));  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 }  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 Intent intent = new Intent(Debts.this, ManageExpense.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.*setDefault*(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
  
}

package com.example.bit\_x.ui;  
  
import static android.content.ContentValues.*TAG*;  
  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.content.Intent;  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.MenuItem;  
import android.view.View;  
  
import com.example.bit\_x.R;  
import com.example.bit\_x.adabter.ExpenseAdapter;  
import com.example.bit\_x.adabter.TraderAdapter;  
import com.example.bit\_x.databinding.ActivityExpenseBinding;  
import com.example.bit\_x.interfaces.RecyclerViewInterface;  
import com.example.bit\_x.models.ExpenseModel;  
import com.example.bit\_x.models.ProjectModel;  
import com.example.bit\_x.models.TraderModel;  
import com.example.bit\_x.models.WagesModel;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.ArrayList;  
import java.util.Locale;  
  
public class Expense extends AppCompatActivity {  
 String projectKey="";  
 private RecyclerViewInterface listener;  
 ActivityExpenseBinding binding;  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 private FirebaseUser user;  
 ExpenseAdapter adapter;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding = ActivityExpenseBinding.*inflate*(getLayoutInflater());  
 View view = binding.getRoot();  
 this.setContentView(view);  
//setAppLanguage();  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
  
 Intent intent =getIntent();  
 projectKey = intent.getStringExtra("projectKey");  
 getProjectNameAndExpenses();  
  
 //set Adapter  
 getExpenses();  
 binding.floatingButtonExpense.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 try {  
 Intent intent = new Intent(Expense.this, AddExpenseOrDebts.class);  
 intent.putExtra("tag","addExpense");  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 });  
 }  
 private void getExpenses(){  
 ArrayList<ExpenseModel> adapterList = new ArrayList<>();  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Expenses");  
 reference.keepSynced(true);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 adapterList.clear();  
 for (DataSnapshot data : snapshot.getChildren()) {  
 ExpenseModel expenseModel = data.getValue(ExpenseModel.class);  
 expenseModel.setKey(data.getKey());  
 adapterList.add(expenseModel);  
 }  
  
 for (int i = 0; i < adapterList.size(); i++) {  
 if (adapterList.get(i).getDebts() == 0) {  
 ExpenseModel expenseModel = adapterList.get(i);  
 for (int j = i; j > 0; j--) {  
 adapterList.set(j, adapterList.get(j - 1));  
 }  
 adapterList.set(0, expenseModel);  
 }  
 }  
 setAdapter(adapterList);  
 binding.shimmer.setVisibility(View.*GONE*);  
 binding.shimmer.stopShimmer();  
 binding.imageView.setVisibility(View.VISIBLE);  
 binding.expenseRecycleView.setVisibility(View.VISIBLE);  
 binding.fabWrapper.setVisibility(View.VISIBLE);  
 if(adapterList.size()>0){  
 binding.constraintLayout.setVisibility(View.VISIBLE);  
 binding.expenseRecycleView.setVisibility(View.VISIBLE);  
 binding.emptyIcon.setVisibility(View.GONE);  
 }else{  
 binding.expenseRecycleView.setVisibility(View.GONE);  
 binding.emptyIcon.setVisibility(View.VISIBLE);  
 binding.constraintLayout.setVisibility(View.GONE);  
  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 private void setAdapter(ArrayList<ExpenseModel> expenseList){  
 adapter = new ExpenseAdapter();  
 binding.expenseRecycleView.setAdapter(adapter);  
 adapter.setList(expenseList);  
 LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this);  
 linearLayoutManager.setReverseLayout(true);  
 linearLayoutManager.setStackFromEnd(true);  
 binding.expenseRecycleView.setLayoutManager(linearLayoutManager);  
 }  
  
 @Override  
 public boolean onContextItemSelected(@NonNull MenuItem item) {  
  
 switch (item.getItemId()){  
 case 160:  
 try {  
 adapter.payMethod(item.getGroupId(),Expense.this,projectKey);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 case 161:  
 try {  
 adapter.editItem(item.getGroupId(),Expense.this,projectKey);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 case 162:  
 try {  
 adapter.removeItem(item.getGroupId(),user,projectKey,Expense.this);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 default:  
 return super.onContextItemSelected(item);  
 }  
  
 }  
 public void getProjectNameAndExpenses(){  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 ProjectModel projectModel= snapshot.getValue(ProjectModel.class);  
 binding.proName.setText(projectModel.getName());  
 double expense = projectModel.getExpense();  
 System.out.println(expense);  
 binding.incomeProBalance.setText(String.valueOf((expense/100\*15)+expense));  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 }  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 try {  
 Intent intent = new Intent(Expense.this, ManageExpense.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.setDefault(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
  
}

package com.example.bit\_x.ui;  
  
import static android.content.ContentValues.*TAG*;  
  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.content.Intent;  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.MenuItem;  
import android.view.View;  
  
import com.example.bit\_x.R;  
import com.example.bit\_x.adabter.IncomeAdapter;  
import com.example.bit\_x.databinding.ActivityIncomeBinding;  
import com.example.bit\_x.interfaces.RecyclerViewInterface;  
import com.example.bit\_x.models.IncomeModel;  
import com.example.bit\_x.models.ProjectModel;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.ArrayList;  
import java.util.Locale;  
  
public class Income extends AppCompatActivity {  
 ActivityIncomeBinding binding;  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 private FirebaseUser user;  
 IncomeAdapter adapter;  
 String projectKey;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding =ActivityIncomeBinding.*inflate*(getLayoutInflater());  
 View view = binding.getRoot();  
 this.setContentView(view);  
 // setAppLanguage();  
 Intent intent =getIntent();  
 projectKey = intent.getStringExtra("projectKey");  
  
  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
 binding.shimmer.startShimmer();  
 //set Adapter  
 getIncome(projectKey);  
  
 binding.floatingButtonIncome.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 try {  
 Intent intent = new Intent(Income.this, AddTraderOrWorker.class);  
 intent.putExtra("tag","addIncome");  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 });  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if(snapshot.exists()){  
 ProjectModel projectModel =snapshot.getValue(ProjectModel.class);  
 binding.proName.setText(projectModel.getName());  
 binding.incomeProBalance.setText(String.*valueOf*(projectModel.getIncome()));  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
  
 private void getIncome(String key){  
 ArrayList<IncomeModel> adapterList = new ArrayList<>();  
 if(key !=null){  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Projects").child(key).child("Incomes");  
 reference.keepSynced(true);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()) {  
 adapterList.clear();  
 for (DataSnapshot data : snapshot.getChildren()) {  
 IncomeModel incomeModel = data.getValue(IncomeModel.class);  
 incomeModel.setKey(data.getKey());  
 adapterList.add(incomeModel);  
 }  
 setCurrentIncome();  
 }  
 setAdapter(adapterList);  
 binding.shimmer.setVisibility(View.*GONE*);  
 binding.shimmer.stopShimmer();  
 binding.imageView.setVisibility(View.*VISIBLE*);  
 binding.incomeRecycleView.setVisibility(View.*VISIBLE*);  
 binding.fabWrapper.setVisibility(View.*VISIBLE*);  
 if(adapterList.size()>0){  
 binding.incomeRecycleView.setVisibility(View.*VISIBLE*);  
 binding.emptyIcon.setVisibility(View.*GONE*);  
 binding.constraintLayout.setVisibility(View.*VISIBLE*);  
 }else{  
 binding.incomeRecycleView.setVisibility(View.*GONE*);  
 binding.emptyIcon.setVisibility(View.*VISIBLE*);  
 binding.constraintLayout.setVisibility(View.*GONE*);  
  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 }  
 private void setAdapter(ArrayList<IncomeModel> incomeModel){  
 adapter = new IncomeAdapter();  
 binding.incomeRecycleView.setAdapter(adapter);  
 adapter.setList(incomeModel);  
 LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this);  
 linearLayoutManager.setReverseLayout(true);  
 linearLayoutManager.setStackFromEnd(true);  
 binding.incomeRecycleView.setLayoutManager(linearLayoutManager);  
 }  
  
  
 @Override  
 public boolean onContextItemSelected(@NonNull MenuItem item) {  
  
 switch (item.getItemId()){  
 case 150:  
 try {  
 adapter.editItem(item.getGroupId(),Income.this,projectKey);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 case 151:  
 try {  
 adapter.removeItem(item.getGroupId(),user,projectKey,Income.this);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 setCurrentIncome();  
 return true;  
 default:  
 return super.onContextItemSelected(item);  
 }  
  
 }  
  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 try {  
 Intent intent = new Intent(Income.this, ManageProject.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
  
 }  
 public void setCurrentIncome (){  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 ProjectModel projectModel =snapshot.getValue(ProjectModel.class);  
 binding.proName.setText(projectModel.getName());  
 binding.incomeProBalance.setText(String.*valueOf*(projectModel.getIncome()));  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.*setDefault*(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
  
}

package com.example.bit\_x.ui;  
  
  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.util.Log;  
import android.widget.Toast;  
  
import com.example.bit\_x.R;  
import com.firebase.ui.auth.AuthMethodPickerLayout;  
import com.firebase.ui.auth.AuthUI;  
import com.firebase.ui.auth.IdpResponse;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
  
import java.util.Arrays;  
import java.util.List;  
import java.util.Locale;  
  
public class Login extends AppCompatActivity {  
 private static final String *TAG* = "LoginRegisterActivity";  
 int AUTHUI\_REQUEST\_CODE = 10001;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_login*);  
 //setAppLanguage();  
 if(FirebaseAuth.*getInstance*().getCurrentUser()!=null){  
 Intent intent = new Intent(Login.this,MainActivity.class);  
 startActivity(intent);  
 finish();  
 }else{  
 setLogin();  
 }  
 }  
 public void setLogin(){  
 List<AuthUI.IdpConfig> providers = Arrays.*asList*(  
 new AuthUI.IdpConfig.GoogleBuilder().build());  
  
 //build layout  
 AuthMethodPickerLayout authUiLayout = new AuthMethodPickerLayout  
 .Builder(R.layout.*google\_layout*)  
 .setGoogleButtonId(R.id.*buLogin*)  
 .build();  
  
 //build google login  
 Intent intent = AuthUI.*getInstance*()  
 .createSignInIntentBuilder()  
 .setAvailableProviders(providers)  
 .setTheme(R.style.*Theme\_Bit\_X*)  
 .setAuthMethodPickerLayout(authUiLayout)  
 .setAlwaysShowSignInMethodScreen(true)  
 .setIsSmartLockEnabled(false)  
 .build();  
 startActivityForResult(intent, AUTHUI\_REQUEST\_CODE);  
 }  
 @Override  
 protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
 if (requestCode == AUTHUI\_REQUEST\_CODE) {  
 if (resultCode == *RESULT\_OK*) {  
 // We have signed in the user or we have a new user  
 FirebaseUser user = FirebaseAuth.*getInstance*().getCurrentUser();  
 Log.*d*(*TAG*, "onActivityResult: " + user.toString());  
 //Checking for User (New/Old)  
  
 Intent intent = new Intent(this, MainActivity.class);  
 startActivity(intent);  
 this.finish();  
  
 } else {  
 // Signing in failed  
 IdpResponse response = IdpResponse.*fromResultIntent*(data);  
 if (response == null) {  
 finish();  
 } else {  
 Toast.*makeText*(this, response.getError().getMessage(), Toast.*LENGTH\_LONG*).show();  
 FirebaseAuth.*getInstance*().signOut();  
 AuthUI.*getInstance*().signOut(this);  
 setLogin();  
 }  
 }  
 }  
 }  
  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 finish();  
 }  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.*setDefault*(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
  
}

package com.example.bit\_x.ui;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.preference.PreferenceManager;  
import android.view.View;  
  
import com.example.bit\_x.R;  
import com.example.bit\_x.databinding.ActivityMainBinding;  
import com.firebase.ui.auth.AuthUI;  
import com.google.firebase.auth.FirebaseAuth;  
  
import java.util.Locale;  
  
public class MainActivity extends AppCompatActivity implements View.OnClickListener {  
  
 ActivityMainBinding binding;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding = ActivityMainBinding.*inflate*(getLayoutInflater());  
 View view = binding.getRoot();  
 this.setContentView(view);  
 // setAppLanguage();  
 checkUser();  
 binding.mainMenuProjects.setOnClickListener(this);  
 binding.mainMenuWorkers.setOnClickListener(this);  
 binding.mainMenuTraders.setOnClickListener(this);  
 binding.mainMenuAnalise.setOnClickListener(this);  
 binding.buLogout.setOnClickListener(this);  
  
 }  
  
 public void checkUser(){  
 if(FirebaseAuth.*getInstance*().getCurrentUser()==null){  
 try {  
 Intent intent =new Intent(MainActivity.this,Login.class);  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 }  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 finish();  
 }  
  
 @Override  
 public void onClick(View view) {  
 try {  
 Intent intent;  
 switch (view.getId()) {  
 case R.id.*mainMenuProjects*:  
 intent = new Intent(MainActivity.this, Projects.class);  
 startActivity(intent);  
 finish();  
 break;  
 case R.id.*mainMenuWorkers*:  
 intent = new Intent(MainActivity.this, Workers.class);  
 startActivity(intent);  
 finish();  
 break;  
 case R.id.*mainMenuTraders*:  
 intent = new Intent(MainActivity.this, Traders.class);  
 startActivity(intent);  
 finish();  
 break;  
 case R.id.*mainMenuAnalise*:  
 intent = new Intent(MainActivity.this, Clients.class);  
 startActivity(intent);  
 finish();  
 break;  
 case R.id.*buLogout*:  
 AuthUI.*getInstance*().signOut(this);  
 intent =new Intent(MainActivity.this,Login.class);  
 startActivity(intent);  
 finish();  
 break;  
 }  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
  
 }  
  
  
}

package com.example.bit\_x.ui;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.view.View;  
  
import com.example.bit\_x.R;  
import com.example.bit\_x.databinding.ActivityManageExpenseBinding;  
import com.example.bit\_x.models.ProjectModel;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.Locale;  
  
public class ManageExpense extends AppCompatActivity implements View.OnClickListener {  
ActivityManageExpenseBinding binding;  
String projectKey="";  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 private FirebaseUser user;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding = ActivityManageExpenseBinding.*inflate*(getLayoutInflater());  
 View view = binding.getRoot();  
 //setAppLanguage();  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
 Intent intent =getIntent();  
 projectKey = intent.getStringExtra("projectKey");  
 this.setContentView(view);  
 binding.expense.setOnClickListener(this);  
 // binding.debts.setOnClickListener(this);  
 binding.wages.setOnClickListener(this);  
 getProjectNameAndExpenses();  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 Intent intent =getIntent();  
 projectKey = intent.getStringExtra("projectKey");  
 }  
  
 @Override  
 public void onClick(View view) {  
 try {  
 Intent intent;  
 switch (view.getId()) {  
 case R.id.*expense*:  
 intent = new Intent(ManageExpense.this, Expense.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 break;  
 /\* case R.id.debts:  
 intent = new Intent(ManageExpense.this, Debts.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 break;\*/  
 case R.id.*wages*:  
 intent = new Intent(ManageExpense.this, Wages.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 break;  
 }  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 public void getProjectNameAndExpenses(){  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 ProjectModel projectModel= snapshot.getValue(ProjectModel.class);  
 binding.proName.setText(projectModel.getName());  
 double expense =(projectModel.getExpense()/100\*15)+projectModel.getExpense();  
  
 binding.proBalance.setText(String.*valueOf*(projectModel.getIncome()-expense));  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 }  
  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 try {  
 Intent intent = new Intent(ManageExpense.this, ManageProject.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.*setDefault*(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
  
}

package com.example.bit\_x.ui;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.view.View;  
  
import com.example.bit\_x.R;  
import com.example.bit\_x.databinding.ActivityManageProjectBinding;  
import com.example.bit\_x.models.ProjectModel;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.Locale;  
  
public class ManageProject extends AppCompatActivity implements View.OnClickListener {  
ActivityManageProjectBinding binding;  
 ProjectModel projectModel;  
 String projectKey="";  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 private FirebaseUser user;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding = ActivityManageProjectBinding.inflate(getLayoutInflater());  
 View view = binding.getRoot();  
 this.setContentView(view);  
 // setAppLanguage();  
 Intent intent =getIntent();  
 projectKey =intent.getStringExtra("projectKey");  
 auth =FirebaseAuth.getInstance();  
 user =auth.getCurrentUser();  
 binding.manageProjectExpense.setOnClickListener(this);  
 binding.manageProjectIncome.setOnClickListener(this);  
  
 getCurrentProject();  
  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 System.out.println("testtest11111");  
 Intent intent =getIntent();  
 projectKey =intent.getStringExtra("projectKey");  
 }  
  
  
 @Override  
 public void onClick(View view) {  
 try {  
 Intent intent;  
 switch (view.getId()) {  
 case R.id.manageProjectIncome:  
 intent = new Intent(ManageProject.this, Income.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 break;  
 case R.id.manageProjectExpense:  
 intent = new Intent(ManageProject.this, ManageExpense.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 break;  
 }  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 private void getCurrentProject(){  
 try {  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if(snapshot.exists()){  
 projectModel = snapshot.getValue(ProjectModel.class);  
 projectModel.setKey(snapshot.getKey());  
 binding.proName.setText(projectModel.getName());  
 double expense =(projectModel.getExpense()/100\*15)+projectModel.getExpense();  
 binding.proBalance.setText(String.valueOf(projectModel.getIncome()-expense));  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
  
 }  
  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 try {  
 Intent intent = new Intent(this, Projects.class);  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.setDefault(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
  
}

package com.example.bit\_x.ui;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.ArrayAdapter;  
  
import com.example.bit\_x.R;  
import com.example.bit\_x.databinding.ActivityPayMentHistoryBinding;  
  
import java.util.Locale;  
  
public class PayMentHistory extends AppCompatActivity implements AdapterView.OnItemSelectedListener {  
  
 ActivityPayMentHistoryBinding binding;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding =ActivityPayMentHistoryBinding.*inflate*(getLayoutInflater());  
 View view = binding.getRoot();  
 this.setContentView(view);  
 // setAppLanguage();  
 setSpinner();  
  
  
 }  
  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.*setDefault*(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
  
 @Override  
 public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {  
 if(position==0){  
 System.*out*.println(position);  
 }else if(position ==1){  
 System.*out*.println(position);  
 }  
 }  
  
 @Override  
 public void onNothingSelected(AdapterView<?> parent) {  
  
 }  
 private void setSpinner(){  
 ArrayAdapter<CharSequence> adapter = ArrayAdapter.*createFromResource*(this,  
 R.array.*payment\_spinner\_array*, android.R.layout.*simple\_spinner\_item*);  
 adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
 binding.paymentSpinner.setAdapter(adapter);  
 binding.paymentSpinner.setOnItemSelectedListener(this);  
 }  
}

package com.example.bit\_x.ui;  
  
import static android.content.ContentValues.*TAG*;  
  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.content.Intent;  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.MenuItem;  
import android.view.View;  
  
import com.example.bit\_x.R;  
import com.example.bit\_x.adabter.ProjectAdapter;  
import com.example.bit\_x.adabter.TraderAdapter;  
import com.example.bit\_x.databinding.ActivityProjectsBinding;  
import com.example.bit\_x.interfaces.GetNameKey;  
import com.example.bit\_x.interfaces.RecyclerViewInterface;  
import com.example.bit\_x.models.ProjectModel;  
import com.example.bit\_x.models.TraderModel;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.ArrayList;  
import java.util.Locale;  
  
public class Projects extends AppCompatActivity {  
 ActivityProjectsBinding binding;  
 ProjectAdapter adapter;  
 ArrayList<ProjectModel> adapterList;  
  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 private FirebaseUser user;  
  
private RecyclerViewInterface listener;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding =ActivityProjectsBinding.*inflate*(getLayoutInflater());  
 View view = binding.getRoot();  
 this.setContentView(view);  
 // setAppLanguage();  
 binding.shimmer.startShimmer();  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
  
 getProjects();  
 //set Adapter  
 setOnClickListener();  
  
  
 binding.floatingButtonProject.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 try {  
 Intent intent = new Intent(Projects.this, AddTraderOrWorker.class);  
 intent.putExtra("tag","addProject");  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
  
 }  
 });  
 }  
  
  
 private void setOnClickListener() {  
 listener = new RecyclerViewInterface() {  
 @Override  
 public void onItemClick(View v, int pos) {  
 try {  
 Intent intent =new Intent(Projects.this,ManageProject.class);  
 intent.putExtra("id",pos);  
 intent.putExtra("projectKey",adapterList.get(pos).getKey());  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
  
 };  
 }  
 private void getProjects(){  
 adapterList = new ArrayList<>();  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Projects");  
 reference.keepSynced(true);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 adapterList.clear();  
 for (DataSnapshot data : snapshot.getChildren()) {  
 ProjectModel projectModel = data.getValue(ProjectModel.class);  
 projectModel.setKey(data.getKey());  
 adapterList.add(projectModel);  
 }  
 setAdapter(adapterList);  
 binding.shimmer.setVisibility(View.*GONE*);  
 binding.shimmer.stopShimmer();  
 binding.imageView.setVisibility(View.*VISIBLE*);  
 binding.projectRecycleView.setVisibility(View.*VISIBLE*);  
 binding.fabWrapper.setVisibility(View.*VISIBLE*);  
 if(adapterList.size()>0){  
 binding.projectRecycleView.setVisibility(View.*VISIBLE*);  
 binding.emptyIcon.setVisibility(View.*GONE*);  
 }else{  
 binding.projectRecycleView.setVisibility(View.*GONE*);  
 binding.emptyIcon.setVisibility(View.*VISIBLE*);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 @Override  
 public boolean onContextItemSelected(@NonNull MenuItem item) {  
  
 switch (item.getItemId()){  
 case 130:  
 try {  
 adapter.editItem(item.getGroupId(),Projects.this);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 case 131:  
 try {  
 adapter.removeItem(item.getGroupId(),user,Projects.this);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 default:  
 return super.onContextItemSelected(item);  
 }  
 }  
 private void setAdapter(ArrayList<ProjectModel> projectModel){  
 adapter = new ProjectAdapter();  
 binding.projectRecycleView.setAdapter(adapter);  
 adapter.setList(projectModel,listener);  
 LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this);  
 linearLayoutManager.setReverseLayout(true);  
 linearLayoutManager.setStackFromEnd(true);  
 binding.projectRecycleView.setLayoutManager(linearLayoutManager);  
 }  
  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 try {  
 Intent intent = new Intent(Projects.this, MainActivity.class);  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.*setDefault*(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
  
}

package com.example.bit\_x.ui;  
  
import static android.content.ContentValues.*TAG*;  
  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.content.Intent;  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.MenuItem;  
import android.view.View;  
  
import com.example.bit\_x.adabter.TraderAdapter;  
import com.example.bit\_x.databinding.ActivityTraderBinding;  
import com.example.bit\_x.interfaces.RecyclerViewInterface;  
import com.example.bit\_x.models.TraderModel;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.ArrayList;  
import java.util.Locale;  
  
public class Traders extends AppCompatActivity {  
 ActivityTraderBinding binding;  
 private RecyclerViewInterface listener;  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 private FirebaseUser user;  
 TraderAdapter adapter;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding =ActivityTraderBinding.*inflate*(getLayoutInflater());  
 View view = binding.getRoot();  
 this.setContentView(view);  
 // setAppLanguage();  
 binding.shimmer.startShimmer();  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
 //set Adapter  
 setOnClickListener();  
 getTraders();  
  
 binding.floatingButtonTrader.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 try {  
 Intent intent = new Intent(Traders.this, AddTraderOrWorker.class);  
 intent.putExtra("tag","addTrader");  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 });  
 }  
  
 private void setOnClickListener() {  
 listener =new RecyclerViewInterface() {  
 @Override  
 public void onItemClick(View v, int pos) {  
  
 }  
  
 };  
 }  
 private void getTraders(){  
 ArrayList<TraderModel> adapterList = new ArrayList<>();  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Traders");  
 reference.keepSynced(true);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 adapterList.clear();  
 for (DataSnapshot data : snapshot.getChildren()) {  
 TraderModel traderModel = data.getValue(TraderModel.class);  
 traderModel.setKey(data.getKey());  
 adapterList.add(traderModel);  
 }  
 setAdapter(adapterList);  
 binding.shimmer.setVisibility(View.GONE);  
 binding.shimmer.stopShimmer();  
 binding.imageView.setVisibility(View.VISIBLE);  
 binding.traderRecycleView.setVisibility(View.VISIBLE);  
 binding.fabWrapper.setVisibility(View.VISIBLE);  
 if(adapterList.size()>0){  
 binding.traderRecycleView.setVisibility(View.VISIBLE);  
 binding.emptyIcon.setVisibility(View.GONE);  
 }else{  
 binding.traderRecycleView.setVisibility(View.GONE);  
 binding.emptyIcon.setVisibility(View.VISIBLE);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 private void setAdapter(ArrayList<TraderModel> traderModels){  
 adapter = new TraderAdapter();  
 binding.traderRecycleView.setAdapter(adapter);  
 adapter.setList(traderModels,listener);  
 LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this);  
 linearLayoutManager.setReverseLayout(true);  
 linearLayoutManager.setStackFromEnd(true);  
 binding.traderRecycleView.setLayoutManager(linearLayoutManager);  
 }  
  
 @Override  
 public boolean onContextItemSelected(@NonNull MenuItem item) {  
  
 switch (item.getItemId()){  
 case 101:  
 try {  
 adapter.editItem(item.getGroupId(),Traders.this);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 case 102:  
 try {  
 adapter.removeItem(item.getGroupId(),user);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 default:  
 return super.onContextItemSelected(item);  
 }  
  
 }  
  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 try {  
 Intent intent = new Intent(Traders.this, MainActivity.class);  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.setDefault(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
  
}

package com.example.bit\_x.ui;  
  
  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.content.Intent;  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.view.MenuItem;  
import android.view.View;  
  
  
import com.example.bit\_x.adabter.WagesAdapter;  
  
import com.example.bit\_x.databinding.ActivityWagesBinding;  
import com.example.bit\_x.interfaces.RecyclerViewInterface;  
import com.example.bit\_x.models.ProjectModel;  
import com.example.bit\_x.models.WagesModel;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.ArrayList;  
import java.util.Locale;  
  
public class Wages extends AppCompatActivity {  
 ActivityWagesBinding binding;  
 String projectKey="";  
 private RecyclerViewInterface listener;  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 private FirebaseUser user;  
 WagesAdapter adapter;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding = ActivityWagesBinding.*inflate*(getLayoutInflater());  
 View view = binding.getRoot();  
 this.setContentView(view);  
 // setAppLanguage();  
 auth = FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
  
 Intent intent =getIntent();  
 projectKey = intent.getStringExtra("projectKey");  
 getProjectNameAndWages();  
  
 //set Adapter  
 getWages();  
 binding.floatingButtonWages.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 try {  
 Intent intent = new Intent(Wages.this, AddExpenseOrDebts.class);  
 intent.putExtra("tag","addWages");  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 });  
 }  
 private void getWages(){  
 ArrayList<WagesModel> adapterList = new ArrayList<>();  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey).child("Wages");  
 reference.keepSynced(true);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 adapterList.clear();  
 for (DataSnapshot data : snapshot.getChildren()) {  
 WagesModel wagesModel = data.getValue(WagesModel.class);  
 wagesModel.setKey(data.getKey());  
 adapterList.add(wagesModel);  
 }  
  
 for (int i = 0; i < adapterList.size(); i++) {  
 if (adapterList.get(i).getDebts() == 0) {  
 WagesModel wagesModel = adapterList.get(i);  
 for (int j = i; j > 0; j--) {  
 adapterList.set(j, adapterList.get(j - 1));  
 }  
 adapterList.set(0, wagesModel);  
 }  
 }  
 setAdapter(adapterList);  
 binding.shimmer.setVisibility(View.GONE);  
 binding.shimmer.stopShimmer();  
 binding.imageView.setVisibility(View.VISIBLE);  
 binding.wagesRecycleView.setVisibility(View.VISIBLE);  
 binding.fabWrapper.setVisibility(View.VISIBLE);  
 if(adapterList.size()>0){  
 binding.wagesRecycleView.setVisibility(View.VISIBLE);  
 binding.emptyIcon.setVisibility(View.GONE);  
 binding.constraintLayout.setVisibility(View.VISIBLE);  
  
 }else{  
 binding.wagesRecycleView.setVisibility(View.GONE);  
 binding.emptyIcon.setVisibility(View.VISIBLE);  
 binding.constraintLayout.setVisibility(View.GONE);  
  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
 private void setAdapter(ArrayList<WagesModel> wagesList){  
 adapter = new WagesAdapter();  
 binding.wagesRecycleView.setAdapter(adapter);  
 adapter.setList(wagesList);  
 LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this);  
 linearLayoutManager.setReverseLayout(true);  
 linearLayoutManager.setStackFromEnd(true);  
 binding.wagesRecycleView.setLayoutManager(linearLayoutManager);  
 }  
  
 @Override  
 public boolean onContextItemSelected(@NonNull MenuItem item) {  
  
 switch (item.getItemId()){  
 case 170:  
 try {  
 adapter.payMethod(item.getGroupId(),Wages.this,projectKey);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 case 171:  
 try {  
 adapter.editItem(item.getGroupId(),Wages.this,projectKey);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 case 172:  
 try {  
 adapter.removeItem(item.getGroupId(),user,projectKey,Wages.this);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 default:  
 return super.onContextItemSelected(item);  
 }  
  
 }  
 public void getProjectNameAndWages(){  
 reference = FirebaseDatabase.getInstance().getReference().child("users").child(user.getUid()).child("Projects").child(projectKey);  
 reference.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.exists()){  
 ProjectModel projectModel= snapshot.getValue(ProjectModel.class);  
 binding.proName.setText(projectModel.getName());  
 double expense = projectModel.getExpense();  
 binding.ProBalance.setText(String.valueOf((expense/100\*15)+expense));  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 }  
 });  
 }  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 try {  
 Intent intent = new Intent(Wages.this, ManageExpense.class);  
 intent.putExtra("projectKey",projectKey);  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.setDefault(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
  
}

package com.example.bit\_x.ui;  
  
import static android.content.ContentValues.*TAG*;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.content.Intent;  
import android.content.res.Configuration;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.MenuItem;  
import android.view.View;  
  
import com.example.bit\_x.R;  
import com.example.bit\_x.adabter.TraderAdapter;  
import com.example.bit\_x.adabter.WorkerAdapter;  
import com.example.bit\_x.databinding.ActivityWorkersBinding;  
import com.example.bit\_x.interfaces.RecyclerViewInterface;  
import com.example.bit\_x.models.TraderModel;  
import com.example.bit\_x.models.WorkerModel;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.ArrayList;  
import java.util.Locale;  
  
public class Workers extends AppCompatActivity {  
 private RecyclerViewInterface listener;  
 ActivityWorkersBinding binding;  
  
 private DatabaseReference reference;  
 private FirebaseAuth auth ;  
 private FirebaseUser user;  
 WorkerAdapter adapter;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding =ActivityWorkersBinding.*inflate*(getLayoutInflater());  
 View view = binding.getRoot();  
 this.setContentView(view);  
 // setAppLanguage();  
 binding.shimmer.startShimmer();  
 auth =FirebaseAuth.*getInstance*();  
 user =auth.getCurrentUser();  
  
 setOnClickListener();  
 getWorkers();  
  
 binding.floatingButtonWorker.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 try {  
 Intent intent = new Intent(Workers.this,AddTraderOrWorker.class);  
 intent.putExtra("tag","addWorker");  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 });  
 }  
 private void setOnClickListener() {  
 listener =new RecyclerViewInterface() {  
 @Override  
 public void onItemClick(View v, int pos) {  
  
 }  
 };  
 }  
 private void getWorkers(){  
 try{  
 ArrayList<WorkerModel> adapterList = new ArrayList<>();  
 reference = FirebaseDatabase.*getInstance*().getReference().child("users").child(user.getUid()).child("Workers");  
 reference.keepSynced(true);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 adapterList.clear();  
 for (DataSnapshot data : snapshot.getChildren()) {  
 if (data.exists()) {  
 WorkerModel workerModel = data.getValue(WorkerModel.class);  
 workerModel.setKey(data.getKey());  
 adapterList.add(workerModel);  
 }  
 }  
 setAdapter(adapterList);  
 binding.shimmer.setVisibility(View.*GONE*);  
 binding.shimmer.stopShimmer();  
 binding.imageView.setVisibility(View.*VISIBLE*);  
 binding.workerRecycleView.setVisibility(View.*VISIBLE*);  
 binding.fabWrapper.setVisibility(View.*VISIBLE*);  
 if(adapterList.size()>0){  
 binding.workerRecycleView.setVisibility(View.*VISIBLE*);  
 binding.emptyIcon.setVisibility(View.*GONE*);  
 }else{  
 binding.workerRecycleView.setVisibility(View.*GONE*);  
 binding.emptyIcon.setVisibility(View.*VISIBLE*);  
 }  
 }  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 private void setAdapter(ArrayList<WorkerModel> list){  
 adapter = new WorkerAdapter();  
 binding.workerRecycleView.setAdapter(adapter);  
 adapter.setList(list,listener);  
 LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this);  
 linearLayoutManager.setReverseLayout(true);  
 linearLayoutManager.setStackFromEnd(true);  
 binding.workerRecycleView.setLayoutManager(linearLayoutManager);  
 }  
 @Override  
 public boolean onContextItemSelected(@NonNull MenuItem item) {  
 switch (item.getItemId()){  
 case 110:  
 try {  
 adapter.getHistory(item.getGroupId(),Workers.this);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 case 111:  
 try {  
 adapter.editItem(item.getGroupId(),Workers.this);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 case 112:  
 try {  
 adapter.removeItem(item.getGroupId(),user);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return true;  
 default:  
 return super.onContextItemSelected(item);  
 }  
 }  
  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 try {  
 Intent intent = new Intent(Workers.this, MainActivity.class);  
 startActivity(intent);  
 finish();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 private void setAppLanguage(){  
 String languageToLoad = "english"; // your language  
 Locale locale = new Locale(languageToLoad);  
 Locale.*setDefault*(locale);  
 Configuration config = new Configuration();  
 config.locale = locale;  
 getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());  
 }  
  
}