Dokumentacja projektu pt.

Program do harmonogramowania pracy zespołu nad projektem inżynierskim

Julia Trzeciakiewicz | 100735

Cel projektu i funkconalności

Aplikacja mobilna do harmonogramowania pracy zespołu nad projektem inżynierskim została stworzona z myślą o studentach realizujących projekty grupowe. Umożliwia ona sprawne zarządzanie zadaniami, terminami oraz członkami zespołu, zwiększając przejrzystość i efektywność pracy.

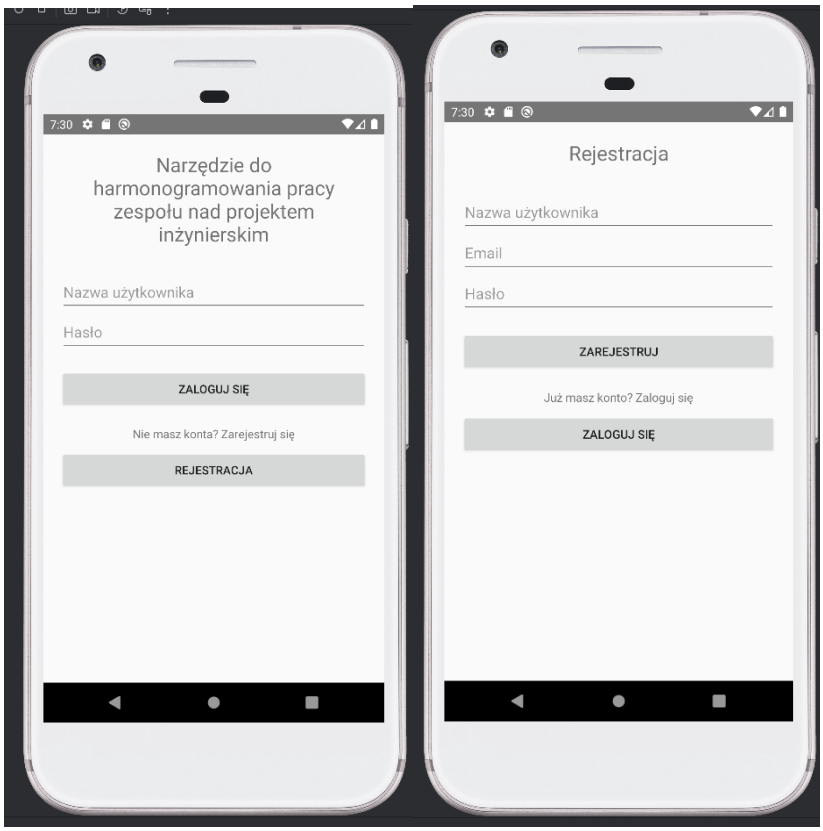
Każdy użytkownik posiada swoje konto. Aplikacja oferuje funkcję rejestracji i logowania, dzięki czemu dane użytkowników są odseparowane, a dostęp do projektów ograniczony tylko do osób do nich dodanych. Każdy projekt mogą zobaczyć i edytować wyłącznie jego uczestnicy, co zapewnia prywatność i porządek.

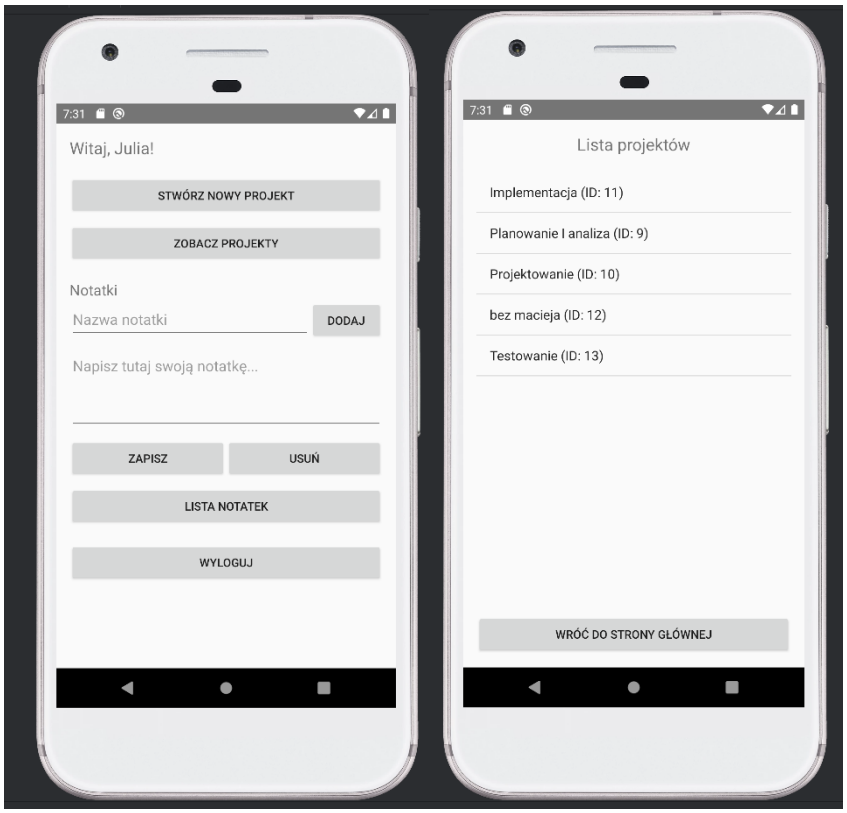
Projekty można tworzyć, edytować oraz usuwać. Podczas tworzenia określa się datę rozpoczęcia i zakończenia, nazwę oraz opis. Projekty dzielone są na zadania, które można przypisywać do konkretnych osób. Istnieje także możliwość losowego przydziału członków zespołu do zadań, co pozwala równomiernie rozłożyć obowiązki bez ręcznego wyboru.

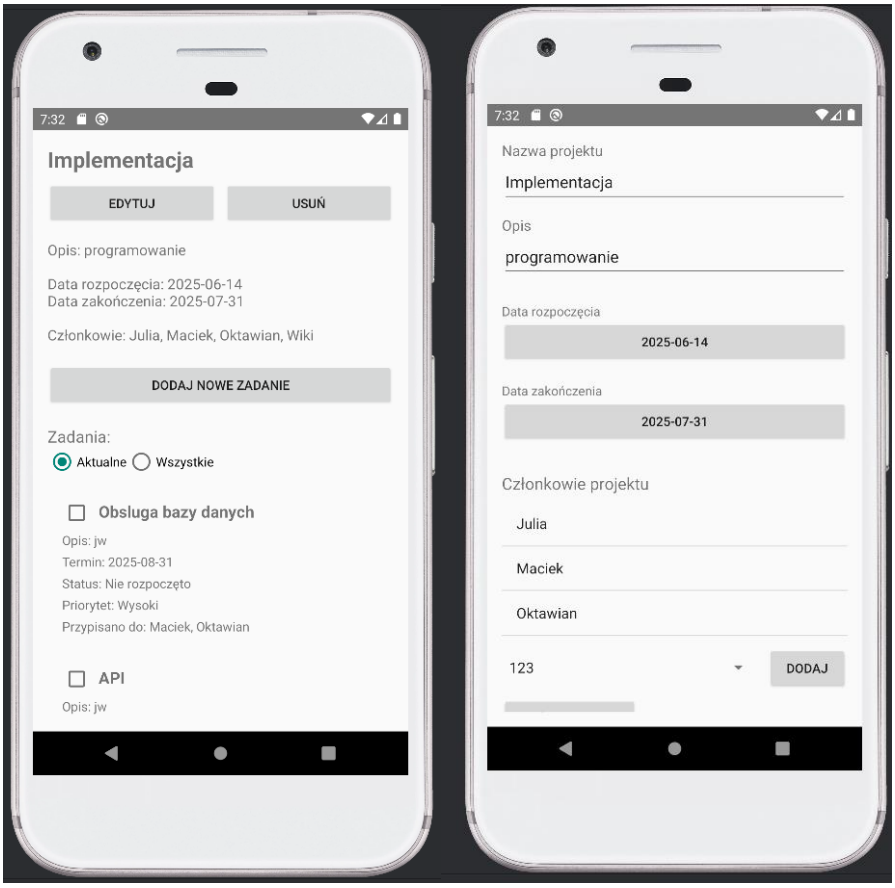
Zadania można edytować – użytkownik może zmienić ich nazwę, opis, status (np. „Nie rozpoczęto”, „W trakcie”, „Gotowe”), priorytet, termin oraz przypisaną osobę. Zadania mogą być również filtrowane – dostępne są widoki pokazujące wszystkie zadania (również zakończone) oraz tylko aktualne, co pomaga skupić się na bieżących działaniach.

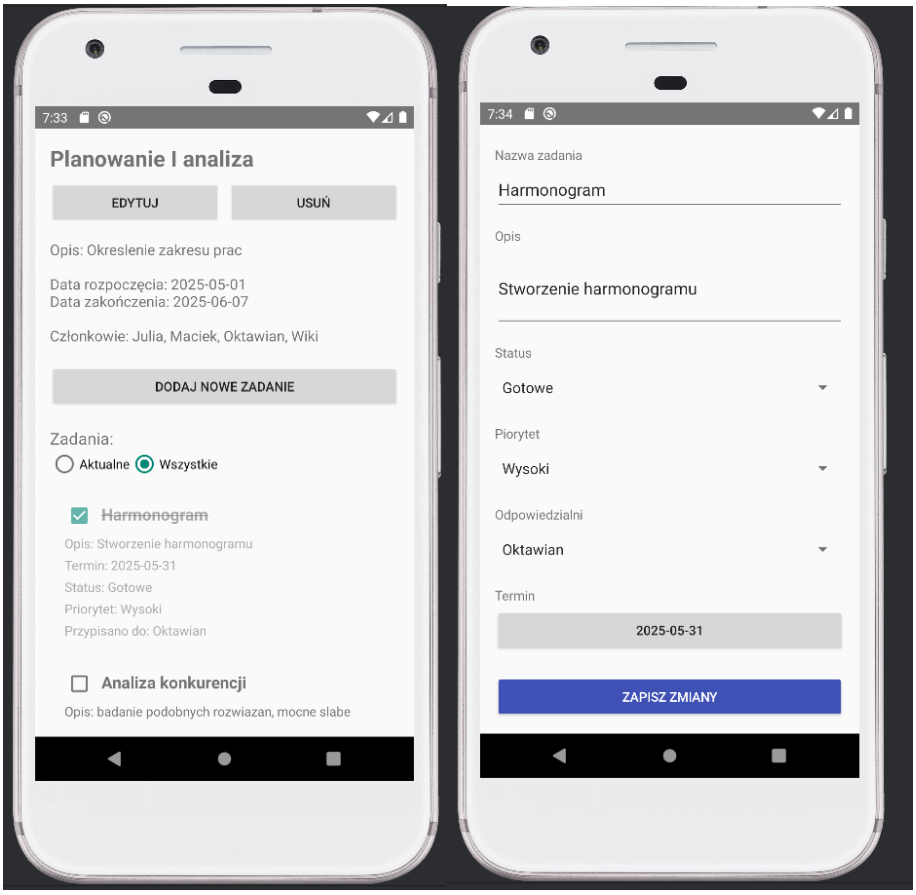
W każdym projekcie wyświetlana jest lista jego uczestników wraz z przypisaniem do konkretnych zadań, co ułatwia komunikację i współpracę w zespole. Dodatkowo aplikacja oferuje możliwość dodawania i zarządzania notatkami związanymi z danym projektem – mogą to być ustalenia, pomysły czy inne istotne informacje.

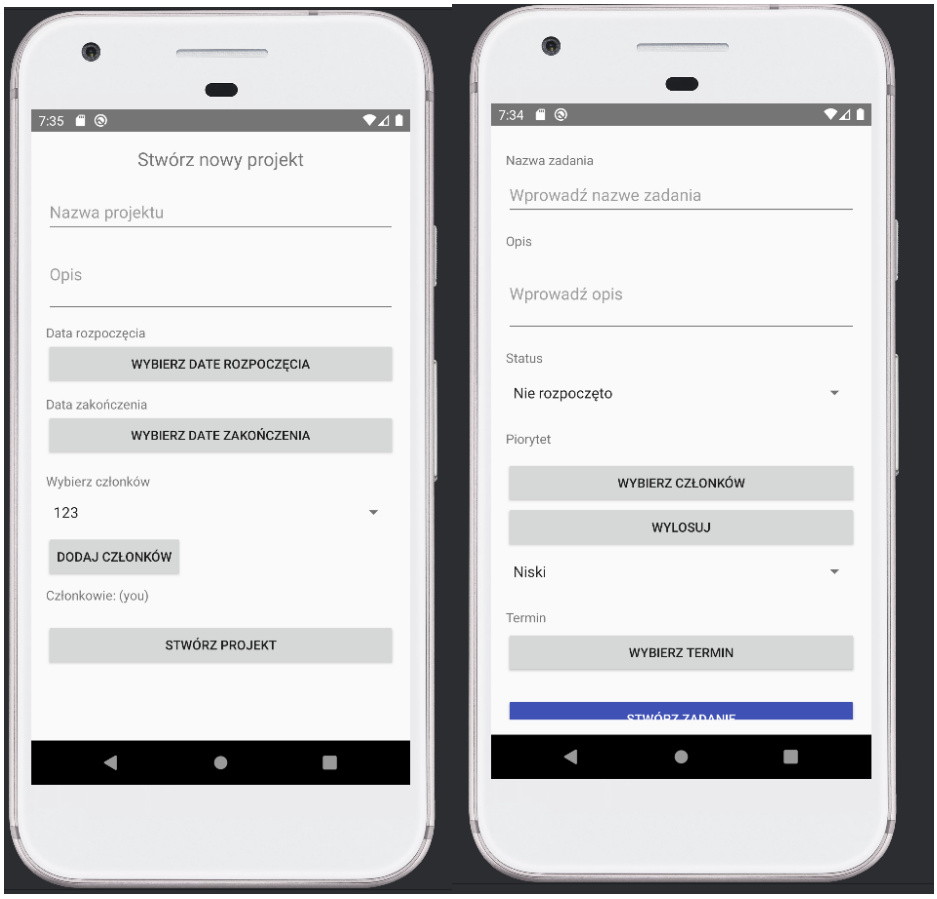
Prezentacja aplikacji



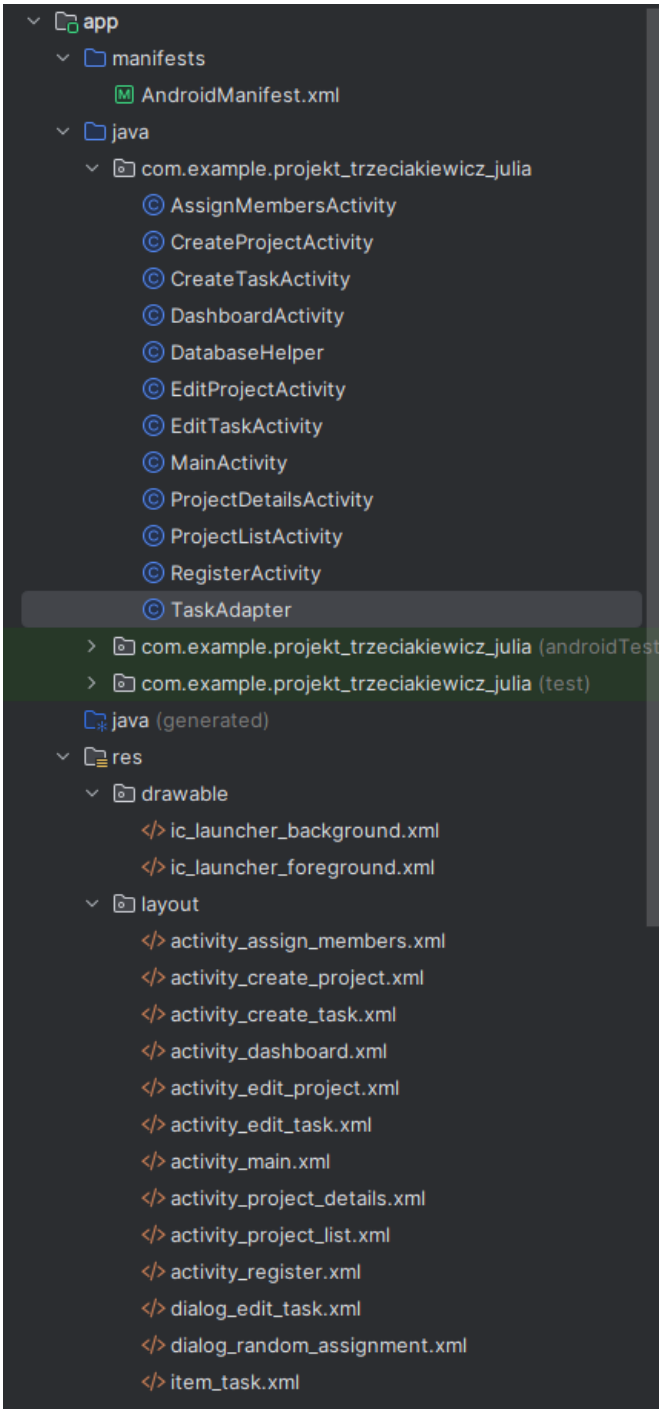



7:32
Implementacja
EDYTUJ
USUŃ
Opis: programowanie
Data rozpoczęcia: 2025-06-14
Data zakończenia: 2025-07-31
Członkowie: Julia, Maciek, Oktawian, Wiki
DODAJ NOWE ZADANIE
Zadania:
Aktualne
Wszystkie
Obsluga bazy danych
Opis: jw
Termin: 2025-08-31
Status: Nie rozpoczęto
Priorytet: Wysoki
Przypisano do: Maciek, Oktawian
API
Opis: jw
7:32
Nazwa projektu
Implementacja
Opis
programowanie
Data rozpoczęcia
2025-06-14
Data zakończenia
2025-07-31
Członkowie projektu
Julia
Maciek
Oktawian
123
DODAJ


7:33
Planowanie I analiza
EDYTUJ
USUŃ
Opis: Okreslenie zakresu prac
Data rozpoczęcia: 2025-05-01
Data zakończenia: 2025-06-07
Członkowie: Julia, Maciek, Oktawian, Wiki
DODAJ NOWE ZADANIE
Zadania:
Aktualne
Wszystkie
Harmonogram
Opis: Stworzenie harmonogramu
Termin: 2025-05-31
Status: Gotowe
Priorytet: Wysoki
Przypisano do: Oktawian
Analiza konkurencji
Opis: badanie podobnych rozwiazan, mocne slabe
7:34
Nazwa zadania
Harmonogram
Opis
Stworzenie harmonogramu
Status
Gotowe
Piorytet
Wysoki
Odpowiedzialni
Oktawian
Termin
2025-05-31
ZAPISZ ZMIANY


7:35
Stwórz nowy projekt
Nazwa projektu
Opis
Data rozpoczęcia
WYBIERZ DATE ROZPOCZĘCIA
Data zakończenia
WYBIERZ DATE ZAKOŃCZENIA
Wybierz członków
123
DODAJ CZŁONKÓW
Członkowie: (you)
STWÓRZ PROJEKT
7:34
Nazwa zadania
Wprowadź nazwe zadania
Opis
Wprowadź opis
Status
Nie rozpoczęto
Piorytet
WYBIERZ CZŁONKÓW
WYLOSUJ
Niski
Termin
WYBIERZ TERMIN

Struktura projektu



Plik	Funkcje i odpowiedzialność
MainActivity.java	- Obsługa logowania użytkownika (sprawdzenie danych w bazie). - Przekierowanie do DashboardActivity po pomyślnym zalogowaniu.

	- Wyświetlanie komunikatów błędów (np. niepoprawne dane).
RegisterActivity.java	- Rejestracja nowego użytkownika (walidacja emaila i hasła). - Sprawdzanie unikalności nazwy użytkownika i emaila. - Zapis danych do bazy (DatabaseHelper). - Przekierowanie do logowania po udanej rejestracji.
DashboardActivity.java	- Wyświetlanie powitania użytkownika. - Przejście do ProjectListActivity (lista projektów). - Zarządzanie notatkami: - Tworzenie, zapisywanie, usuwanie. - Ładowanie istniejących notatek z pamięci urządzenia. - Wyświetlanie listy notatek w oknie dialogowym.
ProjectListActivity.java	- Pobieranie listy projektów przypisanych do użytkownika (DatabaseHelper.getUserProjects()). - Wyświetlanie projektów w ListView. - Przekierowanie do ProjectDetailsActivity po wybraniu projektu. - Sprawdzanie, czy użytkownik jest członkiem projektu (isUserProjectMember()).
ProjectDetailsActivity.java	- Wyświetlanie szczegółów projektu (nazwa, opis, daty, członkowie). - Lista zadań w RecyclerView (z użyciem TaskAdapter). - Filtrowanie zadań (np. tylko aktywne). - Przejście do: - CreateTaskActivity (dodawanie zadania). - EditTaskActivity (edycja zadania). - EditProjectActivity (modyfikacja projektu). - Usuwanie projektu (DatabaseHelper.deleteProject()).
CreateProjectActivity.java	- Tworzenie nowego projektu (nazwa, opis, daty). - Dodawanie członków zespołu poprzez Spinner. - Zapisywanie projektu w bazie (DatabaseHelper.addProject()). - Przypisywanie członków do projektu (DatabaseHelper.addProjectMember()).
EditProjectActivity.java	- Edycja istniejącego projektu (nazwa, opis, daty). - Zarządzanie członkami projektu:

	- Dodawanie nowych członków. - Usuwanie obecnych członków. - Aktualizacja danych w bazie (DatabaseHelper.updateProject()).
CreateTaskActivity.java	- Tworzenie nowego zadania (nazwa, opis, status, priorytet, termin). - Przypisywanie zadania do członków zespołu (ręcznie lub losowo). - Zapisywanie zadania w bazie (DatabaseHelper.addTask()). - Przypisywanie użytkowników do zadania (DatabaseHelper.assignTaskToUser()).
EditTaskActivity.java	- Modyfikacja istniejącego zadania (nazwa, opis, status, priorytet, termin). - Zmiana przypisanych członków. - Aktualizacja danych w bazie (DatabaseHelper.updateTask()).
AssignMembersActivity.java	- Wyświetlanie listy członków projektu. - Losowe przydzielanie członków do zadań. - Wyświetlanie wyników losowania w Toast lub AlertDialog.
DatabaseHelper.java	- **Tworzenie bazy danych SQLite** (tabele: users, projects, tasks, project_members, task_assignments). - **Operacje na użytkownikach**: - Rejestracja (registerUser()). - Logowanie (checkUser()). - Pobieranie ID użytkownika (getUserId()). - **Operacje na projektach**: - Dodawanie (addProject()), edycja (updateProject()), usuwanie (deleteProject()). - Pobieranie listy projektów (getUserProjects()). - Zarządzanie członkami (addProjectMember(), deleteProjectMembers()). - **Operacje na zadaniach**: - Dodawanie (addTask()), edycja (updateTask()). - Przypisywanie do użytkowników (assignTaskToUser()). - Pobieranie zadań dla projektu (getTasksForProject()).

TaskAdapter.java	- Wyświetlanie listy zadań w RecyclerView. - Oznaczanie zadań jako wykonane/niewykonane (CheckBox). - Filtrowanie zadań (np. tylko aktywne). - Przekazywanie kliknięć do ProjectDetailsActivity (edycja zadania).

Funkcjonalności bazodanowe

Plik DatabaseHelper.java jest sercem warstwy dostępu do danych w aplikacji do zarządzania projektami na Androida. Zawiera on zarówno definicję struktury bazy danych, jak i zestaw funkcji umożliwiających operacje na użytkownikach, projektach, zadaniach oraz relacjach między nimi.

Pierwszą i podstawową tabelą jest users, która przechowuje dane wszystkich użytkowników systemu. Każdy użytkownik posiada unikalny identyfikator, nazwę użytkownika, adres e-mail oraz hasło. Zarówno nazwa użytkownika, jak i adres e-mail muszą być unikalne, co zapobiega powielaniu kont w systemie i pozwala na jednoznaczną identyfikację każdej osoby korzystającej z aplikacji.

Drugą istotną tabelą jest projects, w której gromadzone są informacje o poszczególnych projektach. Każdy projekt posiada własny identyfikator, nazwę, opis oraz daty rozpoczęcia i zakończenia. Dzięki temu możliwe jest śledzenie wielu projektów jednocześnie oraz zarządzanie ich cyklem życia.

Aby umożliwić przypisywanie użytkowników do projektów, wprowadzono tabelę project_members. Przechowuje ona powiązania pomiędzy identyfikatorami projektów a identyfikatorami użytkowników. Każdy rekord w tej tabeli reprezentuje jednego członka danego projektu. Tabela ta korzysta z kluczy obcych, które odwołują się do tabel projects oraz users, co zapewnia integralność danych i uniemożliwia przypisanie nieistniejącego użytkownika do nieistniejącego projektu.

Kolejną kluczową tabelą jest tasks, która odpowiada za przechowywanie zadań przypisanych do projektów. Każde zadanie posiada własny identyfikator, należy do konkretnego projektu, ma nazwę, opis, status, priorytet oraz termin realizacji. Relacja z projektami realizowana jest za pomocą klucza obcego, a dodatkowo wprowadzono ograniczenie unikalności, które nie pozwala na powielenie nazw zadań w obrębie jednego projektu. Domyślnie status zadania ustawiony jest na "Nie rozpoczęto", a priorytet na "Średni".

Ostatnią tabelą jest task_assignments, która umożliwia przypisywanie zadań do konkretnych użytkowników. Każdy wpis w tej tabeli zawiera identyfikator zadania oraz identyfikator użytkownika, a relacje te są zabezpieczone kluczami obcymi odwołującymi się do

odpowiednich tabel. Dzięki temu możliwe jest śledzenie, kto jest odpowiedzialny za realizację poszczególnych zadań w ramach projektu.

Konstruktor i inicjalizacja bazy

Na początku klasy zdefiniowane są stałe określające nazwy tabel i kolumn, co ułatwia późniejsze odwoływanie się do nich w zapytaniach SQL. Konstruktor DatabaseHelper(Context context) wywołuje konstruktor klasy nadrzędnej, inicjując bazę o nazwie ProjectManagement.db i wersji 2.

```
public class DatabaseHelper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "ProjectManagement.db";
    public static final int DATABASE_VERSION = 2;
    public static final String TABLE_USERS = "users";
    public static final String TABLE_PROJECTS = "projects";
    public static final String TABLE_TASKS = "tasks";
    public static final String TABLE_PROJECT_MEMBERS = "project_members";
    public static final String TABLE_TASK_ASSIGNMENTS = "task_assignments";

    public static final String COL_ID = "id";
    public static final String COL_NAME = "name";
    public static final String COL_DESCRIPTION = "description";

    public static final String COL_USERNAME = "username";
    public static final String COL_PASSWORD = "password";
    public static final String COL_EMAIL = "email";


    public static final String COL_PROJECT_ID = "project_id";
    public static final String COL_START_DATE = "start_date";
    public static final String COL_END_DATE = "end_date";


    public static final String COL_TASK_ID = "task_id";
    public static final String COL_STATUS = "status";
    public static final String COL_PRIORITY = "priority";
    public static final String COL_DUE_DATE = "due_date";


    public static final String COL_USER_ID = "user_id";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
```

Tworzenie i aktualizacja bazy danych

Metoda onCreate(SQLiteDatabase db) odpowiada za utworzenie wszystkich tabel w bazie. Tworzone są tabele: users, projects, project_members, tasks oraz task_assignments. Każda z nich zawiera odpowiednie klucze główne i obce, zapewniające spójność relacji. W przypadku zmiany wersji bazy, metoda onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) usuwa istniejące tabele i tworzy je na nowo.

```
@Override
public void onCreate(SQLiteDatabase db) {

    String CREATE_USERS_TABLE = "CREATE TABLE " + TABLE_USERS + "("
            + COL_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
            + COL_USERNAME + " TEXT UNIQUE NOT NULL,"
            + COL_EMAIL + " TEXT UNIQUE NOT NULL,"
            + COL_PASSWORD + " TEXT NOT NULL)";
    db.execSQL(CREATE_USERS_TABLE);



    String CREATE_PROJECTS_TABLE = "CREATE TABLE " + TABLE_PROJECTS + "("
            + COL_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
            + COL_NAME + " TEXT NOT NULL,"
            + COL_DESCRIPTION + " TEXT,"
            + COL_START_DATE + " TEXT,"
            + COL_END_DATE + " TEXT)";
    db.execSQL(CREATE_PROJECTS_TABLE);



    String CREATE_PROJECT_MEMBERS_TABLE = "CREATE TABLE " +
TABLE_PROJECT_MEMBERS + "("
            + COL_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
            + COL_PROJECT_ID + " INTEGER NOT NULL,"
            + COL_USER_ID + " INTEGER NOT NULL,"
            + "FOREIGN KEY(" + COL_PROJECT_ID + ") REFERENCES " +
TABLE_PROJECTS + "(" + COL_ID + "),"
            + "FOREIGN KEY(" + COL_USER_ID + ") REFERENCES " + TABLE_USERS +
"(" + COL_ID + "))";
    db.execSQL(CREATE_PROJECT_MEMBERS_TABLE);



    String CREATE_TASKS_TABLE = "CREATE TABLE " + TABLE_TASKS + "("
            + COL_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
            + COL_PROJECT_ID + " INTEGER NOT NULL,"
```

```
        + COL_NAME + " TEXT NOT NULL,"
        + COL_DESCRIPTION + " TEXT,"
        + COL_STATUS + " TEXT DEFAULT 'Nie rozpoczęto',"
        + COL_PRIORITY + " TEXT DEFAULT 'Średni',"
        + COL_DUE_DATE + " TEXT,"
        + "FOREIGN KEY(" + COL_PROJECT_ID + ") REFERENCES " +
TABLE_PROJECTS + "(" + COL_ID + "),"
        + "UNIQUE(" + COL_PROJECT_ID + ", " + COL_NAME + ") ON CONFLICT
REPLACE)";

    db.execSQL(CREATE_TASKS_TABLE);



    String CREATE_TASK_ASSIGNMENTS_TABLE = "CREATE TABLE " +
TABLE_TASK_ASSIGNMENTS + "("
        + COL_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COL_TASK_ID + " INTEGER NOT NULL,"
        + COL_USER_ID + " INTEGER NOT NULL,"
        + "FOREIGN KEY(" + COL_TASK_ID + ") REFERENCES " + TABLE_TASKS + "("
+ COL_ID + "),"
        + "FOREIGN KEY(" + COL_USER_ID + ") REFERENCES " + TABLE_USERS +
"(" + COL_ID + "))";
    db.execSQL(CREATE_TASK_ASSIGNMENTS_TABLE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_TASK_ASSIGNMENTS);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_TASKS);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_PROJECT_MEMBERS);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_PROJECTS);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_USERS);
    onCreate(db);
}
```

Rejestracja i logowanie użytkownika
Metoda registerUser(String username, String email, String password) umożliwia rejestrację nowego użytkownika, sprawdzając poprawność danych wejściowych, format e-maila oraz długość hasła. Jeśli dane są poprawne, użytkownik jest dodawany do tabeli users. Logowanie realizowane jest przez funkcję checkUser(String username, String password), która sprawdza, czy w bazie istnieje użytkownik o podanych danych. Jeśli tak, zwraca wartość true.

```
public boolean registerUser(String username, String email, String password) {
    if (username.isEmpty() || email.isEmpty() || password.isEmpty()) {
```

```
        return false;
    }

    if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        return false;
    }

    if (password.length() < 6) {
        return false;
    }

    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COL_USERNAME, username);
    values.put(COL_EMAIL, email);
    values.put(COL_PASSWORD, password);

    long result = db.insert(TABLE_USERS, null, values);
    return result != -1;
}

public boolean checkUser(String username, String password) {
    SQLiteDatabase db = this.getReadableDatabase();
    String[] columns = {COL_ID};
    String selection = COL_USERNAME + " = ?" + " AND " + COL_PASSWORD + " = ?";
    String[] selectionArgs = {username, password};
    Cursor cursor = db.query(TABLE_USERS, columns, selection, selectionArgs, null, null,
null);
    int count = cursor.getCount();
    cursor.close();
    return count > 0;
}
```

Pobieranie identyfikatora użytkownika
Funkcja getUserId(String username) pozwala uzyskać identyfikator użytkownika na podstawie jego nazwy. Jest to przydatne podczas przypisywania użytkowników do projektów czy zadań.

```
public int getUserId(String username) {
    SQLiteDatabase db = this.getReadableDatabase();
    String[] columns = {COL_ID};
    String selection = COL_USERNAME + " = ?";
    String[] selectionArgs = {username};
    Cursor cursor = db.query(TABLE_USERS, columns, selection, selectionArgs, null, null,
```

```
null);
    if (cursor.moveToFirst()) {
        @SuppressLint("Range") int id = cursor.getInt(cursor.getColumnIndex(COL_ID));
        cursor.close();
        return id;
    }
    cursor.close();
    return -1;
}
```

Dodawanie i zarządzanie projektami

Metoda addProject(String name, String description, String startDate, String endDate) umożliwia dodanie nowego projektu do bazy.

```
public long addProject(String name, String description, String startDate, String endDate) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COL_NAME, name);
    values.put(COL_DESCRIPTION, description);
    values.put(COL_START_DATE, startDate);
    values.put(COL_END_DATE, endDate);
    return db.insert(TABLE_PROJECTS, null, values);
}
public Cursor getProjectDetails(int projectId) {
    SQLiteDatabase db = this.getReadableDatabase();
    return db.query(
            TABLE_PROJECTS,
            new String[]{COL_NAME, COL_DESCRIPTION, COL_START_DATE,
COL_END_DATE},
            COL_ID + " = ?",
            new String[]{String.valueOf(projectId)},
            null, null, null);
}
```

Aktualizacja danych projektu realizowana jest przez updateProject(int projectId, String name, String description, String startDate, String endDate), natomiast usuwanie projektu i powiązanych z nim danych – przez deleteProject(int projectId). Ta ostatnia funkcja dba o kaskadowe usunięcie powiązanych rekordów z innych tabel.

```
public boolean updateProject(int projectId, String name, String description, String startDate,
String endDate) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
```

```
    values.put(COL_NAME, name);
    values.put(COL_DESCRIPTION, description);
    values.put(COL_START_DATE, startDate);
    values.put(COL_END_DATE, endDate);

    int rowsAffected = db.update(TABLE_PROJECTS, values,
            COL_ID + " = ?",
            new String[]{String.valueOf(projectId)});
    return rowsAffected > 0;
}
public boolean deleteProject(int projectId) {
    SQLiteDatabase db = this.getWritableDatabase();

    db.delete(TABLE_TASK_ASSIGNMENTS,
            COL_TASK_ID + " IN (SELECT " + COL_ID + " FROM " + TABLE_TASKS +
                    " WHERE " + COL_PROJECT_ID + " = ?)",
            new String[]{String.valueOf(projectId)});
    db.delete(TABLE_TASKS, COL_PROJECT_ID + " = ?",
            new String[]{String.valueOf(projectId)});

    db.delete(TABLE_PROJECT_MEMBERS, COL_PROJECT_ID + " = ?",
            new String[]{String.valueOf(projectId)});

    int rowsAffected = db.delete(TABLE_PROJECTS, COL_ID + " = ?",
            new String[]{String.valueOf(projectId)});
    return rowsAffected > 0;
}
```

Dodawanie i zarządzanie zadaniami

Dodawanie zadania do projektu odbywa się poprzez funkcję addTask(int projectId, String name, String description, String status, String priority, String dueDate). Aktualizacja zadania jest możliwa dzięki metodzie updateTask(long taskId, String name, String description, String status, String priority, String dueDate), a szczegóły zadania można pobrać funkcją getTaskDetails(long taskId).

```
public long addTask(int projectId, String name, String description,
                    String status, String priority, String dueDate) {
    SQLiteDatabase db = this.getWritableDatabase();
    try {
        ContentValues values = new ContentValues();
        values.put(COL_PROJECT_ID, projectId);
        values.put(COL_NAME, name);
```

```
        values.put(COL_DESCRIPTION, description);
        values.put(COL_STATUS, status);
        values.put(COL_PRIORITY, priority);
        values.put(COL_DUE_DATE, dueDate);

        Log.d("DatabaseHelper", "Dodano zadanie: " + name + " do projektu " + projectId);

        return db.insertWithOnConflict(TABLE_TASKS, null, values,
                SQLiteDatabase.CONFLICT_REPLACE);
    } finally {
        db.close();
    }
}
public Cursor getTaskDetails(long taskId) {
    SQLiteDatabase db = this.getReadableDatabase();
    return db.query(
            TABLE_TASKS,
            new String[]{COL_ID, COL_NAME, COL_DESCRIPTION, COL_STATUS,
COL_PRIORITY, COL_DUE_DATE},
            COL_ID + " = ?",
            new String[]{String.valueOf(taskId)},
            null, null, null);
}

public Cursor getTasksForProject(int projectId) {
    SQLiteDatabase db = this.getReadableDatabase();
    return db.query(
            TABLE_TASKS,
            new String[]{COL_ID, COL_NAME, COL_DESCRIPTION, COL_STATUS,
COL_PRIORITY, COL_DUE_DATE},
            COL_PROJECT_ID + " = ?",
            new String[]{String.valueOf(projectId)},
            null, null, COL_DUE_DATE + " ASC");
}

public boolean updateTask(long taskId, String name, String description,
                          String status, String priority, String dueDate) {
    SQLiteDatabase db = this.getWritableDatabase();
    try {
        ContentValues values = new ContentValues();
        values.put(COL_NAME, name);
        values.put(COL_DESCRIPTION, description);
        values.put(COL_STATUS, status);
        values.put(COL_PRIORITY, priority);
```

```
        values.put(COL_DUE_DATE, dueDate);

        int rowsAffected = db.update(TABLE_TASKS, values,
                COL_ID + " = ?",
                new String[]{String.valueOf(taskId)});

        return rowsAffected > 0;
    } finally {
        db.close();
    }
}
```

Przypisywanie zadań użytkownikom

Przypisanie zadania do użytkownika realizowane jest przez funkcję assignTaskToUser(long taskId, long userId).

```
public boolean assignTaskToUser(long taskId, long userId) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COL_TASK_ID, taskId);
    values.put(COL_USER_ID, userId);
    long result = db.insert(TABLE_TASK_ASSIGNMENTS, null, values);
    return result != -1;
}
```

Zarządzanie członkami projektu

Dodawanie użytkownika do projektu odbywa się przez funkcję addProjectMember(long projectId, long userId). Usuwanie wszystkich członków projektu umożliwia deleteProjectMembers(long projectId), natomiast lista członków projektu pobierana jest przez getProjectMembers(long projectId).

```
public boolean addProjectMember(long projectId, long userId) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COL_PROJECT_ID, projectId);
    values.put(COL_USER_ID, userId);
    long result = db.insert(TABLE_PROJECT_MEMBERS, null, values);
    db.close();
    return result != -1;
}
public boolean deleteProjectMembers(long projectId) {
    SQLiteDatabase db = this.getWritableDatabase();
    int rowsDeleted = db.delete(TABLE_PROJECT_MEMBERS,
```

```
            COL_PROJECT_ID + " = ?",
            new String[]{String.valueOf(projectId)});
    return rowsDeleted > 0;
}
public List<String> getProjectMembers(long projectId) {
    List<String> members = new ArrayList<>();
    String query = "SELECT u." + COL_USERNAME + " FROM " + TABLE_USERS + " u " +
            "JOIN " + TABLE_PROJECT_MEMBERS + " pm ON u." + COL_ID + " = pm." +
COL_USER_ID + " " +
            "WHERE pm." + COL_PROJECT_ID + " = ? " +
            "ORDER BY u." + COL_USERNAME + " ASC";

    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery(query, new String[]{String.valueOf(projectId)});

    while (cursor.moveToNext()) {
        members.add(cursor.getString(0));
    }
    cursor.close();
    return members;
}
```

Pobieranie danych i sprawdzanie relacji

Funkcja getAllUsers() zwraca listę wszystkich użytkowników, a getUserProjects(String username) pozwala pobrać projekty, w których dany użytkownik uczestniczy.

```
public List<String> getAllUsers() {
    List<String> users = new ArrayList<>();
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.query(TABLE_USERS,
            new String[]{COL_USERNAME},
            null, null, null, null, COL_USERNAME + " ASC");

    while (cursor.moveToNext()) {
        users.add(cursor.getString(0));
    }
    cursor.close();
    return users;
}
public List<String> getUserProjects(String username) {
    List<String> projects = new ArrayList<>();
    int userId = getUserId(username);
```

```
    if (userId == -1) {
        return projects;
    }

    String query = "SELECT p." + COL_ID + ", p." + COL_NAME +
            " FROM " + TABLE_PROJECTS + " p" +
            " JOIN " + TABLE_PROJECT_MEMBERS + " pm ON p." + COL_ID + " = pm." +
COL_PROJECT_ID +
            " WHERE pm." + COL_USER_ID + " = ?";

    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery(query, new String[]{String.valueOf(userId)});

    while (cursor.moveToNext()) {
        @SuppressLint("Range") int id = cursor.getInt(cursor.getColumnIndex(COL_ID));
        @SuppressLint("Range") String name =
cursor.getString(cursor.getColumnIndex(COL_NAME));
        projects.add(name + " (ID: " + id + ")");
    }
    cursor.close();
    return projects;
}
```

Podsumowanie kluczowych funkcji w poszczególnych plikach

1. AssignMembersActivity.java

loadCurrentMembers() - Ładuje i wyświetla listę członków projektu

assignRandomMembers() - Przypisuje losowych członków do zadania na podstawie wprowadzonej liczby

2. CreateProjectActivity.java

updateStartDateButton(), updateEndDateButton() - Aktualizują przyciski z datami

updateSelectedMembersList() - Aktualizuje listę wybranych członków projektu

showRemoveMemberDialog() - Pokazuje dialog do usuwania członków

createProject() - Tworzy nowy projekt i przypisuje członków

3. CreateTaskActivity.java

showMemberSelectionDialog() - Pokazuje dialog wyboru członków do zadania

assignRandomMember() - Losowo przypisuje członka do zadania

createTask() - Tworzy nowe zadanie i przypisuje członków

updateDueDateButton() - Aktualizuje przycisk z terminem wykonania

4. DashboardActivity.java

saveNote() - Zapisuje notatkę do pliku

loadNote() - Wczytuje notatkę z pliku

deleteNote() - Usuwa notatkę

showNotesList() - Pokazuje listę dostępnych notatek

5. EditProjectActivity.java

loadProjectData() - Ładuje dane projektu do edycji

loadMembersData() - Ładuje listę członków projektu

setupMemberButtons() - Konfiguruje przyciski zarządzania członkami

setupSaveButton() - Obsługuje zapis zmian w projekcie

6. EditTaskActivity.java

loadTaskData() - Ładuje dane zadania do edycji

getCurrentAssignedMember() - Pobiera aktualnie przypisanego członka

setupSaveButton() - Obsługuje zapis zmian w zadaniu

updateAssignedMember() - Aktualizuje przypisanie zadania do członka

7. MainActivity.java

onCreate() - Obsługuje logowanie użytkownika i przejście do rejestracji

8. ProjectDetailsActivity.java

loadProjectDetails() - Ładuje szczegóły projektu

loadTasks() - Ładuje listę zadań projektu

getAssignedMembers() - Pobiera członków przypisanych do zadania

setupButtonListeners() - Konfiguruje przyciski (dodawanie zadania, edycja projektu itp.)

9. ProjectListActivity.java

onCreate() - Ładuje listę projektów użytkownika

isUserProjectMember() - Sprawdza, czy użytkownik jest członkiem projektu

10. RegisterActivity.java

onCreate() - Obsługuje rejestrację nowego użytkownika

11. TaskAdapter.java

filterTasks() - Filtruje zadania (np. pokazuje/ukrywa ukończone)

updateTaskInDatabase() - Aktualizuje status zadania w bazie danych

updateTasks() - Aktualizuje listę zadań

TaskViewHolder.bind() - Wiąże dane zadania z widokiem

Struktura interfejsu użytkownika – opis komponentów

1. assign_members.xml

Przeznaczenie: Przypisywanie członków do zadania
Główne komponenty:

ScrollView - umożliwia przewijanie zawartości.

TextView (tvCurrentMembers) - wyświetla aktualnych członków przypisanych do zadania.

Button (btnAssignRandom) - losuje członków do zadania.

Button (btnBack) - powrót do poprzedniego ekranu (listy zadań).

2. create_project.xml

Przeznaczenie: Tworzenie nowego projektu
Główne komponenty:

EditText (etProjectName, etProjectDescription) - wprowadzenie nazwy i opisu projektu.

Button (btnStartDate, btnEndDate) - wybór dat rozpoczęcia i zakończenia projektu.

Spinner (spinnerMembers) - lista dostępnych użytkowników do dodania.

Button (btnAddMember) - dodaje wybranego członka z spinnera.

TextView (tvSelectedMembers) - wyświetla aktualnych członków projektu.

Button (btnCreateProject) - zapisuje projekt.

3. create_task.xml

Przeznaczenie: Tworzenie nowego zadania
Główne komponenty:

EditText (etTaskName, etTaskDescription) - nazwa i opis zadania.

Spinner (spinnerStatus, spinnerPriority) - wybór statusu i priorytetu zadania.

Button (btnAssignMembers) - przejście do przypisywania członków.

Button (btnAssignRandom) - losowe przypisanie członków.

Button (btnDueDate) - wybór terminu wykonania zadania.

Button (btnCreateTask) - zapisuje zadanie.

4. dashboard.xml

Przeznaczenie: Ekran główny użytkownika
Główne komponenty:

TextView (tvWelcome) - powitanie użytkownika.

Button (btnCreateProject, btnViewProjects) - tworzenie/przegląd projektów.

EditText (etNoteName, etNotes) - tworzenie notatek.

Button (btnSaveNote, btnDeleteNote, btnListNotes) - zarządzanie notatkami.

Button (btnLogout) - wylogowanie.

5. edit_project.xml

Przeznaczenie: Edycja istniejącego projektu
Główne komponenty:

EditText (etProjectName, etProjectDescription) - edycja nazwy i opisu.

Button (btnStartDate, btnEndDate) - zmiana dat projektu.

ListView (lvCurrentMembers) - lista aktualnych członków.

Spinner (spinnerAvailableUsers) - dostępni użytkownicy do dodania.

Button (btnAddMember, btnRemoveMember) - zarządzanie członkami.

Button (btnSave) - zapis zmian.

6. edit_task.xml

Przeznaczenie: Edycja istniejącego zadania
Główne komponenty:

EditText (etTaskName, etTaskDescription) - edycja nazwy i opisu.

Spinner (spinnerStatus, spinnerPriority, spinnerAssignedTo) - zmiana statusu, priorytetu i przypisanych osób.

Button (btnDueDate) - zmiana terminu.

Button (btnSave) - zapis zmian.

7. activity_main.xml

Przeznaczenie: Ekran logowania
Główne komponenty:

EditText (etUsername, etPassword) - wprowadzenie danych logowania.

Button (btnLogin) - logowanie.

Button (btnGoToRegister) - przejście do rejestracji.

8. project_details.xml

Przeznaczenie: Szczegóły projektu
Główne komponenty:

TextView (tvProjectName, tvProjectDetails) - nazwa i szczegóły projektu.

Button (btnEditProject, btnDeleteProject) - edycja/usuwanie projektu.

Button (btnAddTask) - dodanie nowego zadania.

RadioGroup (rgTaskFilter) - filtrowanie zadań (aktywne/wszystkie).

RecyclerView - lista zadań w projekcie.

Button (btnBackToProjects) - powrót do listy projektów.

9. project_list.xml

Przeznaczenie: Lista projektów
Główne komponenty:

ListView (lvProjects) - wyświetla projekty użytkownika.

Button (btnBackToDashboard) - powrót do dashboardu.

10. register.xml

Przeznaczenie: Rejestracja nowego użytkownika
Główne komponenty:

EditText (etUsername, etEmail, etPassword) - dane rejestracji.

Button (btnRegister) - rejestracja.

Button (btnGoToLogin) - przejście do logowania.

11. task_item.xml

Przeznaczenie: Pojedynczy element listy zadań (używany w RecyclerView)
Główne komponenty:

CheckBox (cbTaskCompleted) - oznaczenie zadania jako ukończone.

TextView (tvName) - nazwa zadania.

TextView (tvDescription, tvDueDate, tvStatus, tvPriority, tvAssignedTo) - szczegóły zadania.