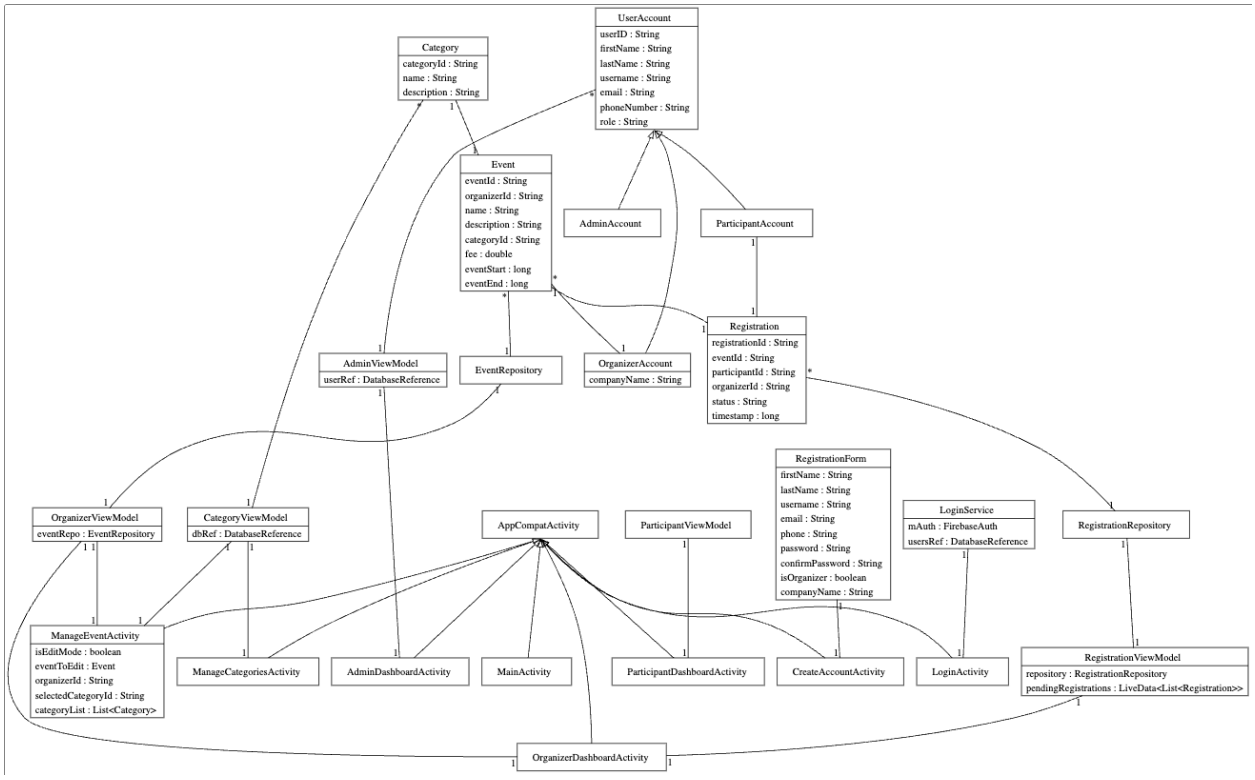

UMPLE CODE (see in project wiki)

```
/// ==================
// BEHAVIOR COMMENTS
// ==================

// ManageEventActivity handles both event creation and editing.
// In edit mode, it uses a temporary reference to the existing Event object (`eventToEdit`).
// This is not a true 1–1 ownership — it's just a transient link for editing, so we do NOT model it as a strong UML association.

// The activity depends on OrganizerViewModel and CategoryViewModel to fetch and update event and category data,
// but the ViewModels do not store a reference to the activity, so the relationships are one-sided.

// OrganizerDashboardActivity launches ManageEventActivity for editing or creating events.
// This is not a persistent relationship, just a usage dependency.

// We only include meaningful fields like `organizerId`, `selectedCategoryId`, and `isEditMode` that represent
//persistent state or logic. UI-specific components (EditTexts, Buttons, etc.) are intentionally excluded to keep
//the diagram focused on structure and behavior, not layout.

// ========== MODELS ==========
class Event {
  String eventId;
  String organizerId;
  String name;
  String description;
  String categoryId;
  double fee;
  long eventStart;
  long eventEnd;

  // Each Event refers to exactly one Category
  1 -- 1 Category;
}
```

```
class Category {
  String categoryId;
  String name;
  String description;

}

class Registration {
  String registrationId;
  String eventId;
  String participantId;
  String organizerId;
  String status; // "pending", "approved", "rejected"
  long timestamp;
  // Each Registration refers to one Event and one ParticipantAccount
  1 -- 1 Event;
  1 -- 1 ParticipantAccount;
}

class RegistrationForm {
  String firstName;
  String lastName;
  String username;
  String email;
  String phone;
  String password;
  String confirmPassword;
  boolean isOrganizer;
  String companyName;
}

// ========== ACCOUNTS ==========
class UserAccount {
  String userID;
  String firstName;
  String lastName;
  String username;
  String email;
  String phoneNumber;
  String role;
}

class AdminAccount {
  isA UserAccount;
}

class OrganizerAccount {
  isA UserAccount;
  String companyName;

  // Each OrganizerAccount can own many Events
  1 -- * Event;
}

class ParticipantAccount {
  isA UserAccount;
}

// ========== REPOSITORIES ==========
class EventRepository {
  // Repository manages many Events
  1 -- * Event;
}

class RegistrationRepository {
  // Repository manages many Registrations
  1 -- * Registration;
}

// ========== VIEWMODELS ==========
class AdminViewModel {
  DatabaseReference userRef;
  1 -- * UserAccount;
}

class OrganizerViewModel {
  EventRepository eventRepo;
  1 -- 1 EventRepository;
}

class RegistrationViewModel {
  RegistrationRepository repository;
  LiveData<List<Registration>> pendingRegistrations;
  void loadPendingRegistrations(String organizerId);
  LiveData<List<Registration>> getPendingRegistrations();
  1 -- 1 RegistrationRepository;
}

class ParticipantViewModel {
}

class CategoryViewModel {
  DatabaseReference dbRef;
  1 -- * Category;
}

// ========== SERVICES ==========
class LoginService {
  FirebaseAuth mAuth;
  DatabaseReference usersRef;
}

// ========== ACTIVITIES ==========

class AppCompatActivity {}

class AdminDashboardActivity {
  isA AppCompatActivity;
  1 -- 1 AdminViewModel;
}

class OrganizerDashboardActivity {
  isA AppCompatActivity;
  1 -- 1 OrganizerViewModel;
  1 -- 1 RegistrationViewModel;
}

class ParticipantDashboardActivity {
  isA AppCompatActivity;
  1 -- 1 ParticipantViewModel;
}

class ManageCategoriesActivity {
  isA AppCompatActivity;
  1 -- 1 CategoryViewModel;
}

class ManageEventActivity {
  isA AppCompatActivity;
  boolean isEditMode;
  Event eventToEdit;
  String organizerId;
  String selectedCategoryId;
  List<Category> categoryList;
  1 -- 1 OrganizerViewModel;
  1 -- 1 CategoryViewModel;
}

class CreateAccountActivity {
  isA AppCompatActivity;
  1 -- 1 RegistrationForm;
}

class LoginActivity {
  isA AppCompatActivity;
  1 -- 1 LoginService;
}

class MainActivity {
  isA AppCompatActivity;
}
```