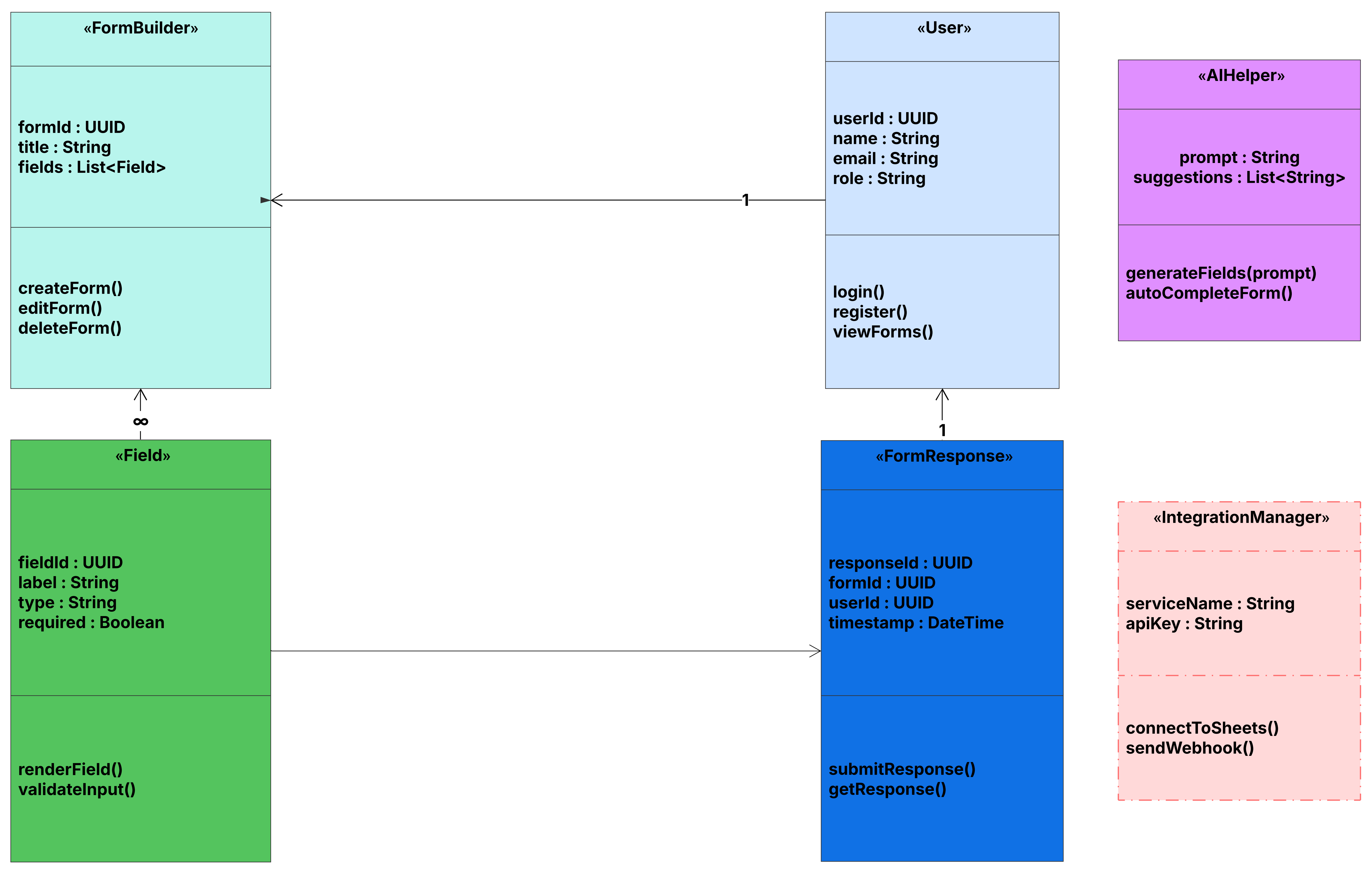

UML class diagrams

🧩 **Relationships**

- **FormBuilder** has a one-to-many relationship with **Field**
- **FormBuilder** has a one-to-many relationship with **FormResponse**
- **User** has a one-to-many relationship with **FormResponse**
- **AIHelper** and **IntegrationManager** are helper/utility classes
- Possible inheritance: if needed, AdminUser and StandardUser could inherit from User

UML Class Diagram – QuickForms AI

The **UML Class Diagram** provides a visual blueprint of the system's internal structure by representing the key classes, their attributes, methods, and relationships. This ensures adherence to object-oriented principles and clean architectural design.

🔹 Key Classes and Responsibilities

Class Name	Purpose
FormBuilder	Core class that handles creation, modification, and validation of form elements.
FormElement	Abstract class representing any form field (text, checkbox, dropdown, etc.).
TextField, CheckBox, Dropdown	Subclasses of FormElement that implement specific UI elements.
FormTemplate	Stores reusable form templates created by users.
User	Represents an authenticated user; includes user role, settings, and history.
Submission	Captures form responses and links them to users and form instances.
AIHelper	Handles AI suggestions for field types, layout, or pre-filled data using NLP.
DatabaseManager	Manages persistence layer (e.g., save/load forms, users, submissions).

🔹 Relationships & Design

- FormBuilder uses FormElement instances to dynamically build forms.
- FormElement subclasses inherit common properties like label, required, and defaultValue.
- A User can create multiple FormTemplate instances.
- Each Submission is linked to a FormTemplate and a User.
- AIHelper communicates with the backend NLP model and is used by the FormBuilder for intelligent field prediction.
- DatabaseManager is a service class that interacts with the persistence layer (e.g., SQLite, Firebase).

🔹 Design Patterns & Best Practices

- **Factory Pattern** may be used by FormBuilder to instantiate different types of FormElement.
- **Single Responsibility Principle (SRP)** is followed to separate concerns across classes.
- **Encapsulation** is applied by exposing only relevant attributes/methods via getters/setters.

New

Share