Final Project Software Requirements Specification (SRS)-UPDATE

QuickForms AI – Smart Form & The Workflow Builder

Course: CSC 318 – Software Engineering Principles

Assignment: Final Project – QuickForms AI

Deliverable: *Software Testing Report*

Project Title: QuickForms AI – Smart Form & Workflow Builder

Student: Eric Amoh Adjei
Instructor: Professor Rawad Habib
Date: August 2025

1. Project Overview

QuickForms is an AI-assisted, intelligent, lightweight form builder for small businesses and freelancers that provides ease in designing custom forms through drag-and-drop tools or AI-assisted templates. This is an intelligent formation tool that rapidly churns out intelligent forms with automated workflows for teams. Using conversational input and predefined templates, users can create contact forms, surveys, job applications, and payment forms without needing to write code. The platform connects with Google Sheets, Slack, and Webhooks; sends professional emails; and offers smooth automation with basic business task needs.

Target Audience:

- Small business owners
- Freelancers and consultants
- Non-technical teams

2. Scope and Objectives

The system allows users to:

- Customizable form fields and templates, whereby users can create and edit form templates
- AI-assisted form generation from user prompts: whereby users Auto-generate forms using AI prompts
- Integration with Google Sheets, Slack, and webhook URLs: whereby the users can connect forms to external tools (e.g., APIs)
- Workflow builder with triggers and actions
- Form publishing and sharing options (link or embed)

Out of Scope:

- Hybrid with omnichannel support for the mobile app.
- Real-time team collaboration on forms, things like real payment processing or subscription billing.
- Advanced conditional logic for an MVP

Goals:

- Provide an intuitive and accessible form-building application
- Minimize the time that needs to be spent on setting up for small business automation
- Integrate with commonly used tools

3. Functional Requirements

1. **User Registration and Login:**
   The system shall generate form templates based on user prompts utilizing natural language processing (NLP).
2. **Create, Edit, and Delete Forms:**
   The system shall allow form fields to be manually edited and rearranged by users.
3. **Drag-and-drop Form Builder UI:**
   The system shall allow users to define workflows, setting triggers (for example, form submission) and actions (for example, send a Slack message).
4. **AI-Powered Form Generation via prompt:**
   The system shall store submitted form data and offer export options for Google Sheets.
5. **Store Form Submissions in a database:**
   The system shall provide a form preview and an embed code for live use.
6. **Enable webhook integration for form data:**
   The system shall allow users to authenticate with Google and Slack.
7. **View form submissions and export them (CSV/Google Sheets):**
   The system will save form templates under the user's dashboard.
8. **Form preview and sharing via public URL:**
   The system shall allow webhook configuration for custom integrations.
9. **Templates for common use cases (e.g., job application, contact form):**
   A validation mechanism for form input shall be in place before submission.
10. **Basic analytics (e.g., total submissions per form):**
    The system shall at least support basic user accounts with login and logout functions.

4. Non-Functional Requirements

1. **Performance:**
   Forms must load and submit within 2 seconds under normal traffic, and the system shall run without lag with 100 concurrent users.
2. **Usability:**
   The user interface shall be simple enough for even the least technically capable user to navigate or manipulate. It should have at least 99.5% uptime per month.
3. **Reliability:**
   The system shall have an uptime of 99.5% with graceful error handling, and every page shall load within 2 seconds.
4. **Security:**
   All data transmissions shall be encrypted (HTTPS), and user authentication tokens shall expire after inactivity and all form submissions are encrypted in transit and at rest

---

5. System Architecture Overview (MVC)

- The architecture follows that of the Model-View-Controller layered architecture
- **Presentation Layer (View):** Web-based user interface
- **Application Layer (Controller):** Handles logic and routes
- **Data Layer (Model):** Stores user, form, and submission data in PostgreSQL

---

6. Database Schema

Tables of key interest:

- **Users:** Stores registered users

- **Forms:** Contains metadata for forms
- **Fields:** Stores individual form fields
- **Submissions:** User responses per form
- **AI Generations:** Stores generated forms via prompt

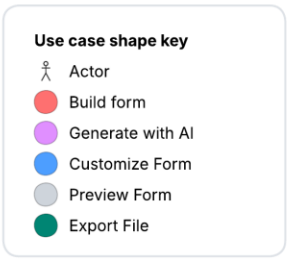
---

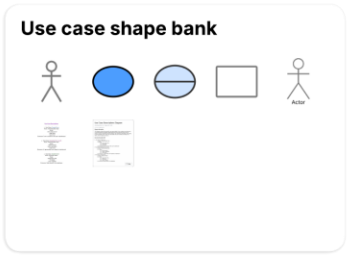
7. UML Diagrams

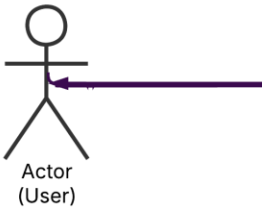
See attached diagrams:

A. A Use Case Diagram:

UML Use Case Diagram for QuickForms AI







Use Case Descriptions

- Use Case: Create Form
Actor: Authenticated User
Steps:
Click 'New Form'
Add fields
Save form
Outcome: Form is saved to the user's dashboard

- Use Case: Generate Form via AI
Actor: Authenticated User
Steps:
Enter prompt
Click 'Generate'
Edit result
Outcome: AI-generated form added to dashboard

- Use Case: Submit Form
Actor: External Visitor
Steps:
Open form URL
Fill fields
Click 'Submit'
Outcome: Data stored in the database

https://lucid.app/lucidchart/a958eb5a-2790-42cb-816e-fdb398adc52e/edit?viewport_loc=-384%2C-

509%2C2491%2C3861%2C.Q4MUjXso07N&invitationId=inv_0231e5f8-16bd-4074-8d77-3396ada87276

- Actors: User, AI Engine, Slack API, Google API
- Use Cases: Create Form, Edit Form, Define Workflow, Connect Integration, Submit Form, View Data

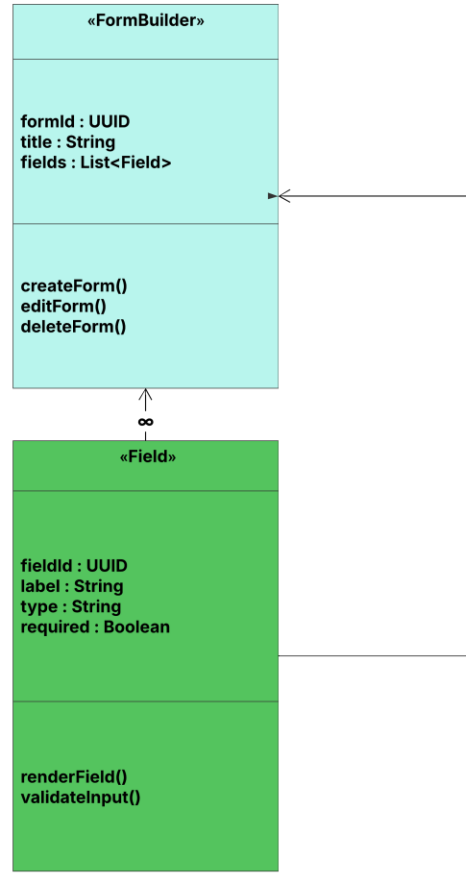
B. A_UML Class Diagram:

UML Class Diagram for QuickForms AI

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 representin attributes, methods, and relationships. This ensures adherence to object-oriented principles and cle

Key Classes and Responsibilities

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

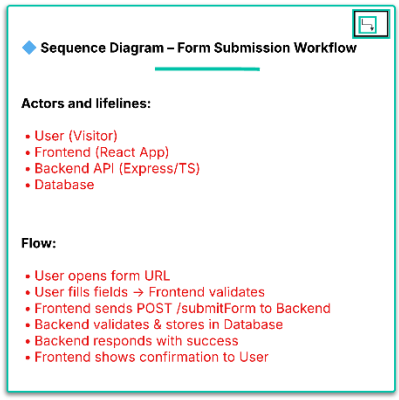
Relationships & Design

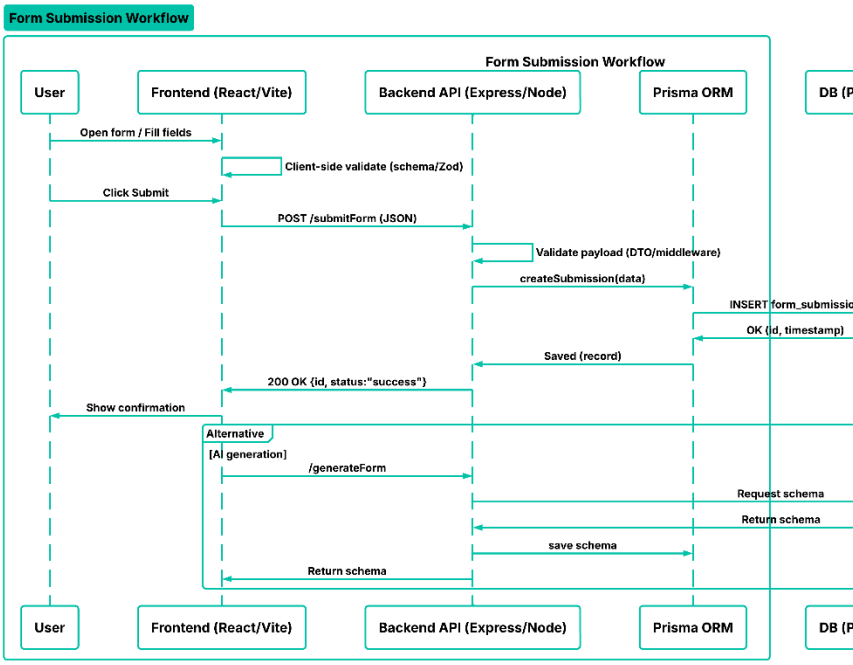
- FormBuilder uses FormElement instances to dynamically build forms.
- FormElement subclasses inherit common properties like label, required, and defaultValue.
- A User can create multiple FormTemplate instances.
- Each Submission is linked to a FormTemplate and a User.
- AIHelper communicates with the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

https://lucid.app/lucidspark/55b5dab0-e8ed-4ffc-944c-bedf769bfb33/edit?viewport_loc=-87277%2C-4612%2C4575%2C3006%2C0_0&invitationId=inv_edfc7d03-f8fb-49f1-903b-395032d01eaf

- Components: UI Frontend (React), AI Form Generator (Python), Workflow Engine, Integration Module, Database (PostgreSQL), Auth Service

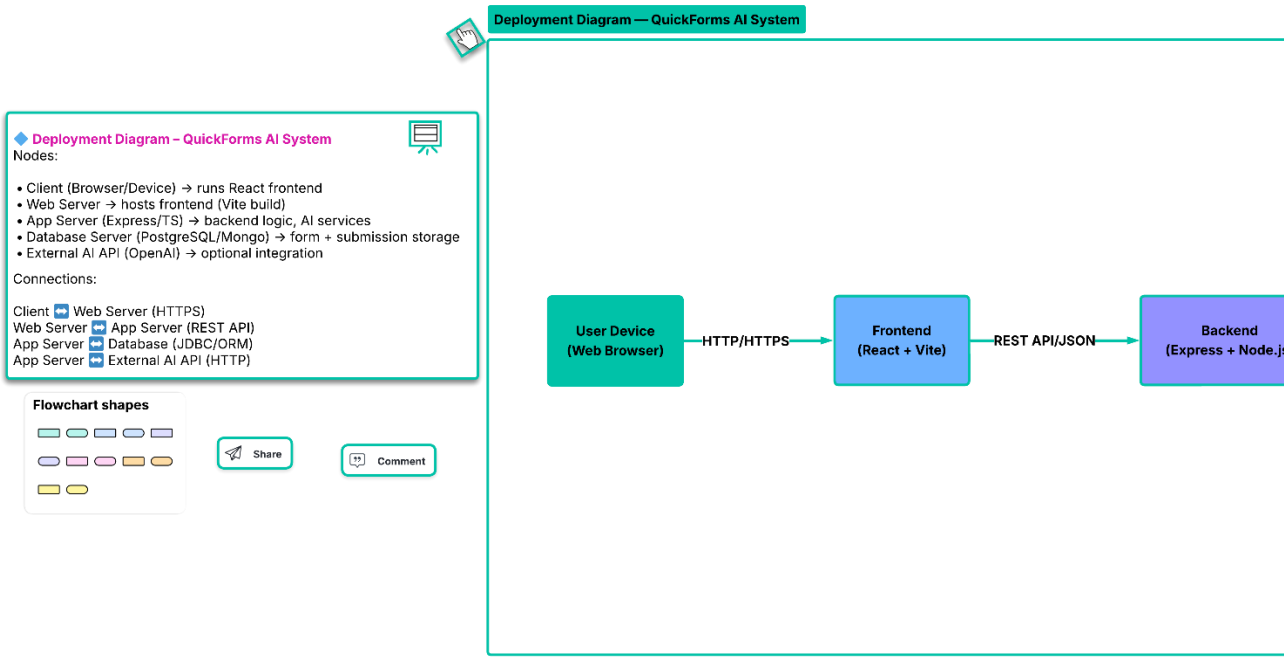
C. Sequence Diagram – Demonstrates the workflow of a form submission from frontend to backend.





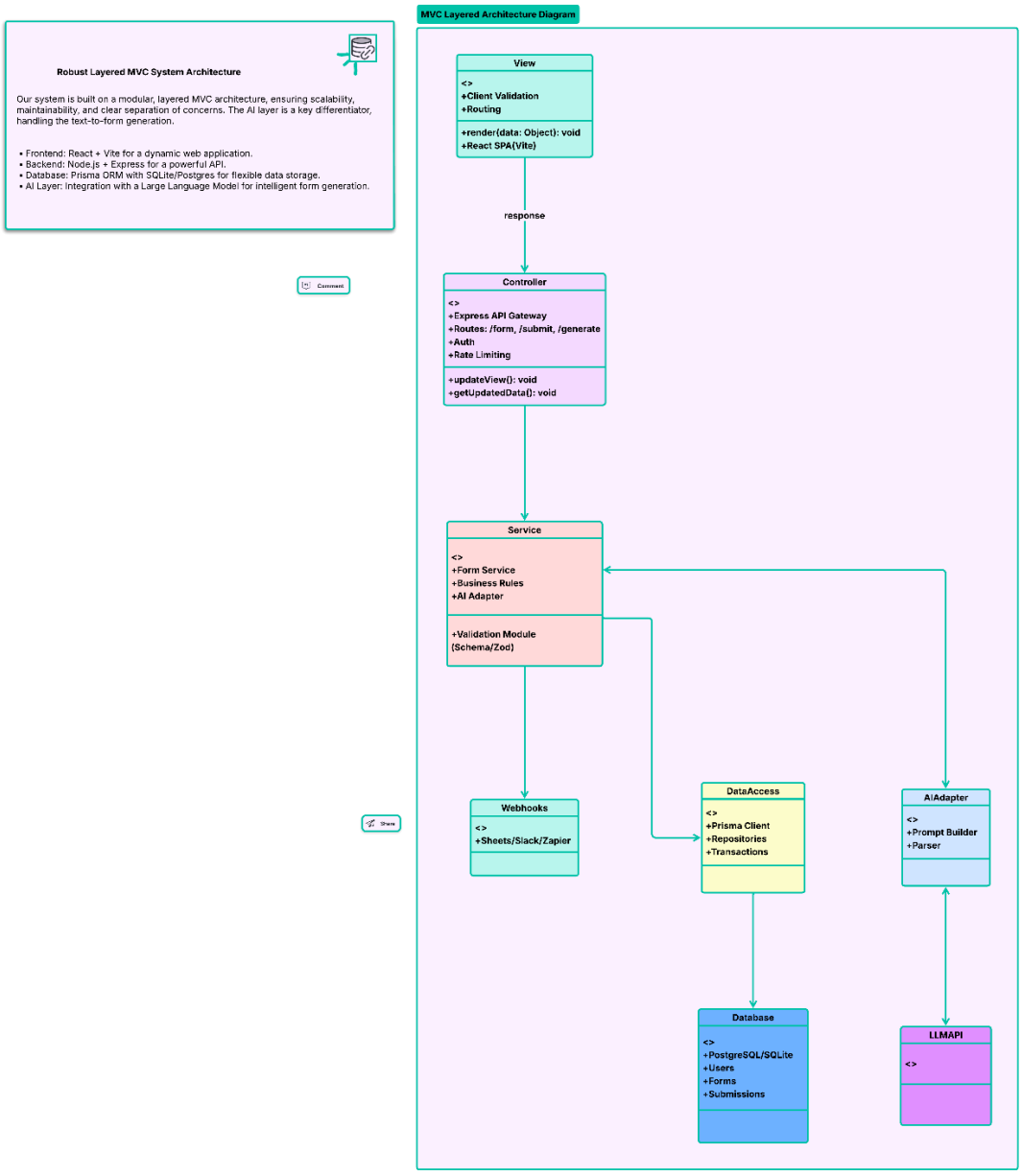
https://lucid.app/lucidchart/9ebac5f8-2750-4c49-a818-a68a9c01148f/edit?invitationId=inv_a858d1aa-8893-44db-b14a-b05e9f610924

D. Deployment Diagram – Visualizes how QuickForms AI is deployed across client, server, and database environments.

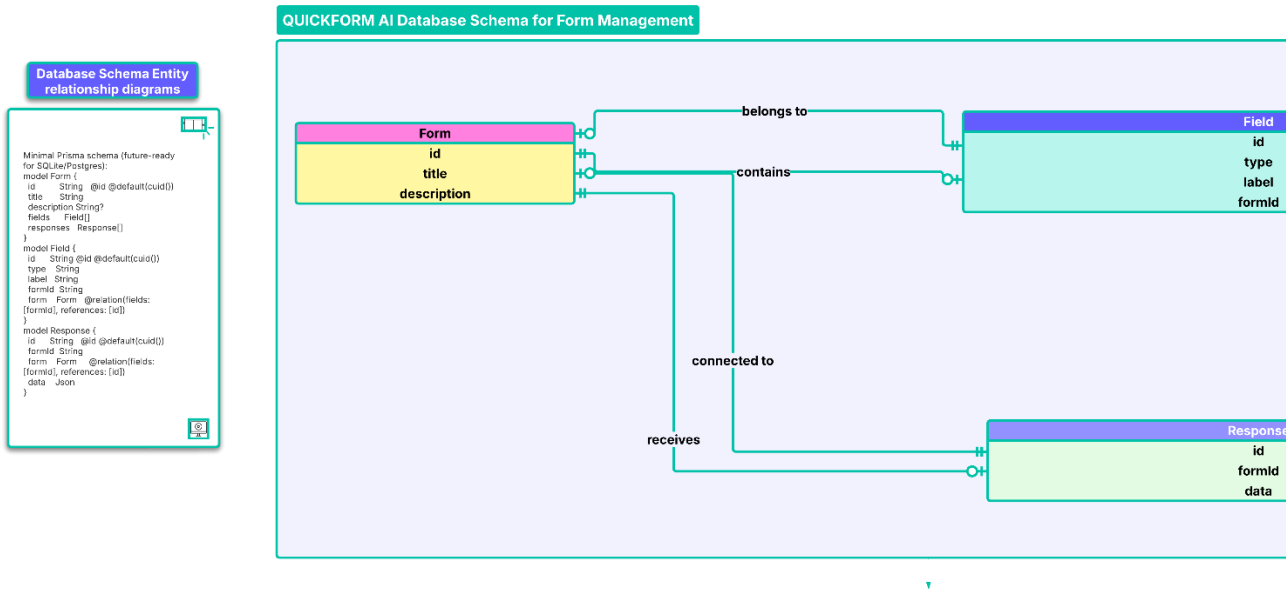


https://lucid.app/lucidchart/1c700ff5-06c7-4310-b4ef-26c5ae98bbab/edit?invitationId=inv_aa12924c-587a-43bd-adfd-b25f6b2aed0f

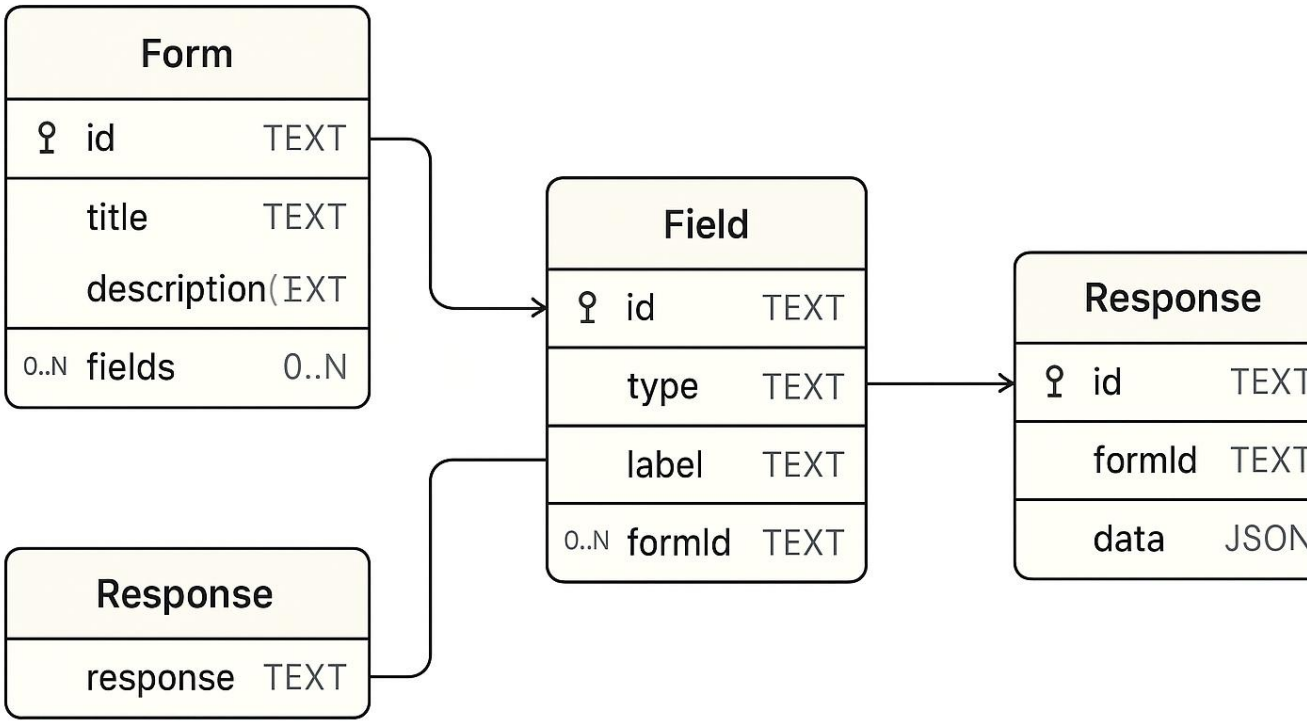
E. Layered MVC architecture Diagram



F. Database Schema



https://lucid.app/lucidchart/bb13c9f2-ef7c-4fa7-b07a-97e29cee2578/edit?invitationId=inv_6d3aa4a2-01bd-46cd-8820-9fd786dff201



---

8. Glossary and Assumptions

Glossary:

- **Form Builder (**Template)**:** The user interface to create or modify a form, a reusable layout of input fields
- **Webhook:** An HTTP callback, user-defined, to interface with third-party services; in other words, an external URL to which data is submitted
- **NLP:** Natural Language Processing used to interpret user input
- **LLM**: Large Language Model for AI-generated forms
- **Payload**: The data submitted by the user through a form
- **Workflow:** A set of automation rules based on form events
- **MVP:** Minimum Viable Product

Assumptions:

- Users have internet access with a modern web browser, while for managing forms, they must be logged in
- Given the text interpretation, AI is empowered by a pre-trained language model. An AI generator uses the OpenAI/HuggingFace API
- One form is set up for one user
- External APIs (Google, Slack) will be assumed stable and well-documented, whereas generative data will be stored securely

---

9. Use Case Descriptions

- ### Use Case: Create Form

Actor: Authenticated User

Steps:

- Click 'New Form'
- Add fields
- Save form

Outcome: Form is saved to the user's dashboard

- **Use Case: Generate Form via AI**

Actor: Authenticated User

Steps:

- Enter prompt
- Click 'Generate'
- Edit result

Outcome: AI-generated form added to dashboard

- **Use Case: Submit Form**

Actor: External Visitor

Steps:

- Open form URL
- Fill fields
- Click 'Submit'

Outcome: Data stored in the database