Software Engineering Lab
(MA 616)

Mini Project

Project Title:

InsightForm

Team Members:
Abhishek (244CA002)
Anubhav Srivastava (244CA007)
Kushagra Singh (244CA024)
Naveen Vashistha (244CA032)

Department of Mathematical & Computational Sciences
National Institute of Technology Karnataka, Surathkal, India

Contents

1 Introduction

InsightForm is an intelligent and intuitive form-building platform that transforms the traditional way of collecting and analyzing data. Unlike existing tools that simply allow form creation, InsightForm integrates the power of AI to help users formulate surveys based on their business objectives, automate analysis, and receive actionable insights. This report outlines the core goals, technical details, timelines, and motivations behind the development of InsightForm.

2 Team Members

- **Abhishek** – Full Stack Developer
- **Anubhav Srivastava** – API Engineer & Frontend Developer
- **Kushagra Singh** – UI/UX Designer & Frontend Developer
- **Naveen Vashistha** – Full Stack Developer

3 Problem Statement

Businesses often rely on surveys to understand customer behavior, employee satisfaction, and market needs. However, most form-building tools provide a generic interface without context-sensitive support. This leads to the following.

- Poorly formulated questions unrelated to specific objectives
- Difficulty in interpreting collected responses
- Manual processes for insight generation
- Delayed decisions and lack of clarity in outcome-driven forms

3.1 The Gap in Existing Solutions

Popular form tools like Google Forms, Typeform, and Jotform offer flexible builders, but lack intelligent assistance, objectives-based customization, and real-time insights that users can immediately act on. These platforms are not built for users seeking immediate, context-driven recommendations.

3.1.1 Need for AI-Assisted Feedback Loops

Stakeholders want to:

- Define business problems in natural language
- Let AI translate these into structured questionnaires
- Collect data, and instantly receive data-driven suggestions

3.2 Project Objectives

1. Enable users to articulate business goals in plain English
2. Use AI to suggest and refine question sets
3. Provide several types of question components to build form easily.
4. Automate report generation based on responses and initial intent
5. Ensure real-time updates, feedback notifications, and data export options

4 Scope of the Project

The project aims to deliver a Minimum Viable Product (MVP) that demonstrates the full "objective-to-insight" cycle. It includes:

- Authenticated user access
- AI-assisted form generation and editing
- Data collection from public form links
- AI-powered single-page reports
- Tabular raw data visualization
- Export functionalities (PDF, Excel)
- Real-time notification system

Out-of-scope for MVP:

- Advanced branching logic in forms
- Multi-user collaboration on forms
- Integration with third-party tools (e.g., Slack, Google Sheets)

5 Target Users / Stakeholders

InsightForm is designed for users who need fast, structured insights:

- Marketing teams conducting customer feedback
- Product managers gathering UX insights
- HR teams surveying employee satisfaction
- Educators running class feedback and research forms
- Startup founders assessing market fit

These users need more than just forms—they need AI-supported, business-aligned data interpretation tools.

6 Technology Stack

InsightForm leverages modern tools to deliver a responsive, scalable, and secure platform:

- **Frontend:** React.js, Tailwind CSS
- **Backend:** Node.js, Express.js, Passport.js
- **Database:** MongoDB
- **Authentication:** JWT, Google OAuth
- **Real-Time System:** WebSockets (Socket.io)
- **AI Layer:** Google Gemini or OpenAI
- **PDF Export:** jsPDF
- **Excel Export:** SheetJS (xlsx)
- **Deployment:** Render/Heroku or Docker + CI/CD pipelines

7 Functional Requirements

Functional requirements define what the system must do. These are the specific features and functions of application.

- **User Authentication:**
  - 1: The system shall allow users to register for a new account using an email address and password.
  - 2: The system shall verify that a user's email is valid by sending a confirmation link.
  - 3: The system shall allow users to register and log in using their Google account via OAuth 2.0.
  - 4: The system shall authenticate users and provide a JSON Web Token (JWT) for session management.
  - 5: The system shall have a protected route (/api/auth/me) to verify a user's token and return their profile information.
  - 6: The system shall prevent authenticated users from accessing the login and signup pages.
- **Form Management:**
  - 1: Authenticated users shall be able to create a new form by providing a title and a business objective.
  - 2: The system shall allow users to add, edit, and delete questions within a form. Supported question types include text, MCQ, and rating.
  - 3: The system shall allow users to reorder questions using a drag-and-drop interface.
  - 4: The system shall automatically save any changes made to a form's structure using a debounced PATCH request.
  - 5: Authenticated users shall be able to view a list of all forms they have created on a dashboard.
  - 6: Authenticated users shall be able to delete their own forms.
  - 7: An user can change their form's visibility, decide if the form can be anonymously filled and if only authenticated users can fill the form or not.
- **Response Collection:**
  - 1: Each form shall have a unique, shareable public URL for collecting responses.
  - 2: The system shall validate all incoming responses on the backend to ensure required questions are answered and data types are correct.
  - 3: The system shall allow form creators to set an option to limit submissions to one per authenticated user.
  - 4: The system shall store each response with a "snapshot" of the questions as they existed at the time of submission to maintain data integrity.
- **Reporting and analytics:**
  - 1: The system shall allow form creators to request an AI-generated report for their form.
  - 2: The system shall pre-process and summarize quantitative and qualitative data before sending it to an external LLM API.
  - 3: The AI-generated report shall include a high-level summary and a list of structured, actionable suggestions.

- 4: The system shall provide a rate-limiting feature (e.g., 3 tries per form) for generating new AI reports to control costs.
- 5: The system shall cache the latest report and only allow a new one to be generated if new responses have been collected.
- 6: The report page shall display a raw data view of all responses and a summary view with data visualizations (charts, graphs).
- 7: The system shall preserve a history of all generated reports for a form, allowing for trend analysis.

8 Non Functional Requirements

Non-functional requirements define how the system should operate. They are the quality attributes, constraints, and standards the system must meet.

- **Security:**
  - 1: All user passwords shall be securely hashed and salted using bcrypt before being stored in the database.
  - 2: All protected API routes shall be secured using JWT verification.
  - 3: The system shall prevent Insecure Direct Object Reference (IDOR) by ensuring users can only access or modify their own data (forms, reports, etc.).
  - 4: All user-provided data shall be validated on the backend to prevent injection attacks and ensure data integrity.
  - 5: The API shall be configured with a CORS policy that only allows requests from the designated frontend domain in production.
- **Performance:**
  - 1: All API responses for non-AI-related requests shall have a response time of less than 500ms under normal load.
  - 2: The frontend application's initial load time shall be under 3 seconds on a standard internet connection.
  - 3: The debounced autosave feature shall have a delay of 1.5-2 seconds to balance responsiveness and server load.
- **Usability:**
  - 1: The web application's user interface shall be responsive and fully functional on modern desktop and mobile browsers.
  - 2: The application shall provide clear visual feedback for background operations, such as "Saving..." and "All changes saved" indicators.
  - 3: The system shall provide clear and user-friendly error messages for both client-side and server-side validation failures.
- **Reliability:**
  - 1: The system shall have robust error handling on the backend to prevent crashes from unexpected user input or system failures.
  - 2: The production application should aim for an uptime of 99.9
- **Maintainability:**
  - 1: The backend code shall be organized by resource (e.g., authRoutes, formRoutes) and use a modular structure (controllers, models, middleware).
  - 2: The frontend code shall be component-based and use a centralized state management solution (e.g., React Context) for authentication.

9 External Interfaces

- **User Interface (UI/UX):**
  - **Responsiveness:** The UI shall be fully responsive, providing a seamless experience on standard desktop, tablet, and mobile browsers.
  - **Clarity:** The interface shall be uncluttered, with a logical flow and clear navigation to guide users through creating forms, viewing reports, and managing their accounts.
  - **Feedback:** The system shall provide immediate and clear visual feedback for user actions, including loading indicators for data fetching, success/error notifications for submissions, and a persistent status indicator for the autosave feature (e.g., "Saving...", "All changes saved").
  - **Accessibility:** The UI should strive to follow basic web accessibility standards (WCAG) to improve usability for people with disabilities. This will be an area of focus for future enhancements.
- **Hardware Interfaces:** As a web-based application, InsightForm does not directly interface with any specialized hardware. The application will be accessible through any standard user device with a modern web browser and an internet connection, including:
  - Desktop Computers
  - Laptops
  - Tablets
  - Smartphones

  The application will rely on standard input hardware such as keyboards, mice, and touchscreens.
- **Software Interfaces:** The system will interact with several internal and external software components.
  - **frontend-backend API**: The frontend (React) and backend (Node.js/Express) will communicate via a RESTful API. Data will be transmitted in JSON format over HTTP/S. The API will use JWTs in the Authorization header for securing protected endpoints.
  - **Database Interface**: The backend server will interface with a MongoDB database (for flexible response data) using the Mongoose ODM.
  - **Google OAuth 2.0**: The system will interface with the Google Identity Services API to provide a "Sign in with Google" option for users. This involves redirecting users to Google's OAuth endpoint and handling the callback to securely authenticate them.**Google OAuth 2.0**:
  - **LLM Service**: The backend will interface with an external Large Language Model API (e.g., OpenAI API) to generate report summaries and suggestions. This communication will occur over a secure HTTP/S connection, sending processed response data and receiving structured JSON.
  - **Email Service**: The backend will interface with a transactional email service (e.g., SendGrid, Mailgun) via their API or SMTP to send account verification and notification emails.
- **Communication/Network Interfaces:**
  - **HTTPS**: All communication between the client (browser) and the server will occur over the Hypertext Transfer Protocol. The HTTPS protocol shall be enforced in the production environment to ensure all data is encrypted in transit.
  - **Web Sockets**: The application will use the WebSocket protocol to provide real-time, bi-directional communication for the user notification feature (e.g., alerting a form creator instantly when a new response is submitted).

10 Use Case/Scenarios

- **Use Case 1: User Authentication**
  - **Use case name:** User Authentication
  - **Actor:** Form Creator
  - **Goal:** To securely access a personal account or create a new one.
  - **Preconditions:** The user has a device with a web browser and an internet connection.
  - **Trigger:** The user navigates to the application's login or signup page.
  - **Main Success Flow (Local Registration):**
    - 1. User selects "Sign Up" and provides their name, email, and a valid password.
    - 2. The system validates the input format
    - 3. The system creates a new user account with an isVerified: false status.
    - 4. The system sends a verification email with a unique link to the provided address.
    - 5. The user clicks the link in the email.
    - 6. The system verifies the link, updates the user's status to isVerified: true, and logs them in by providing a JWT.
  - **Main Success Flow (login):**
    - 1. User enters their credentials (email/password or clicks "Sign in with Google").
    - 2. The system validates the credentials against the database or via Google's OAuth service
    - 3. Upon successful validation, the system generates a JWT.
    - 4. The system sends the JWT to the client, establishing an authenticated session
    - 5. The user is redirected to their dashboard.
  - **Alternate flows/Exceptions:**
    - A1: User Already Exists: During signup, if the email is already registered, the system displays an error message.
    - E1: Invalid Credentials: During login, if the email/password is incorrect, the system displays an error message
    - E2: Google Auth Fails: If the Google OAuth process is canceled or fails, the user is returned to the login page with an error notification
    - E3: User is Already Logged In: If an authenticated user attempts to access the login/signup page, the system redirects them to their dashboard.
- **Use Case 2: Form Creation and Management**
  - **Use case name:** Form Creation and Management
  - **Actor:** Form Creator
  - **Goal:** To create, customize, and save a form that is ready to collect responses.
  - **Preconditions:** The Form Creator must be authenticated and on their dashboard.
  - **Trigger:** The Form Creator clicks the "Create New Form" button.
  - **Main Success Flow:**
    - 1. The system prompts the creator for a business objective
    - 2. The creator enters the objective and proceeds to the form builder
    - 3. The system displays the builder interface with a list of AI-suggested questions based on the objective
    - 4. The creator adds questions from the suggestions or creates them manually.
    - 5. The creator edits question text, changes question types, reorders questions, and sets options (e.g., "required").

    - 6. As the creator makes changes, the system automatically saves the entire form structure to the database in the background after a short, debounced delay.
    - 7. The creator receives visual feedback that their changes are being saved
  - **Alternate flows/Exceptions:**
    - A1: Change Form Status: The creator can change the form's status from "draft" to "live," making it accessible to respondents via a public URL.
    - E1: Network Failure: If the autosave fails due to a network error, the UI displays a "Failed to save" message and retries when the connection is restored.
    - E2: Validation Error: If the creator tries to save invalid data (e.g., an MCQ with no options), the system rejects the change and displays an error.

- **Use Case 3: Response Submission**
  - **Use case name:** Response Submission
  - **Actor:** Respondent
  - **Goal:** To successfully submit a complete and valid set of answers to a form
  - **Preconditions:** The form must have a "live" status.
  - **Trigger:** The Respondent accesses the form's unique public URL.
  - **Main Success Flow:**
    - 1. The system fetches and displays the public details of the form (title and questions)
    - 2. The respondent fills out the answers for each question
    - 3. The respondent clicks the "Submit" button
    - 4. The frontend sends the response data (an array of answers) to the backend API.
    - 5. The backend validates that all required questions have been answered and that the data types are correct.
    - 6. The system saves the response to the database, including a snapshot of the questions
    - 7. The system redirects the respondent to a "Submission Successful" page.
  - **Alternate flows/Exceptions:**
    - A1: Authenticated Submission: If the form requires authentication, there will be a user email field in form and that user email should be present in the database for successful submission.
    - E1: Validation Failure: If the submission fails backend validation (e.g., a required answer is missing), the system rejects it and returns an error message to the user.
    - E2: Duplicate Submission: If the form is limited to one response and the user has already submitted, the system rejects the new submission with an error.
    - E3: Form Not Found/Live: If the URL is invalid or the form is not live, the system displays a "Form Not Available" error page.

- **Use Case 4: Report Generation and Analysis**
  - **Use case name:** Report Generation and Analysis
  - **Actor:** Form Creator
  - **Goal:** To view collected data and generate AI-powered insights.
  - **Preconditions:** The Form Creator is authenticated and has a form with at least one response
  - **Trigger:** The Form Creator clicks the "View Report" button for a form on their dashboard.
  - **Main Success Flow:**

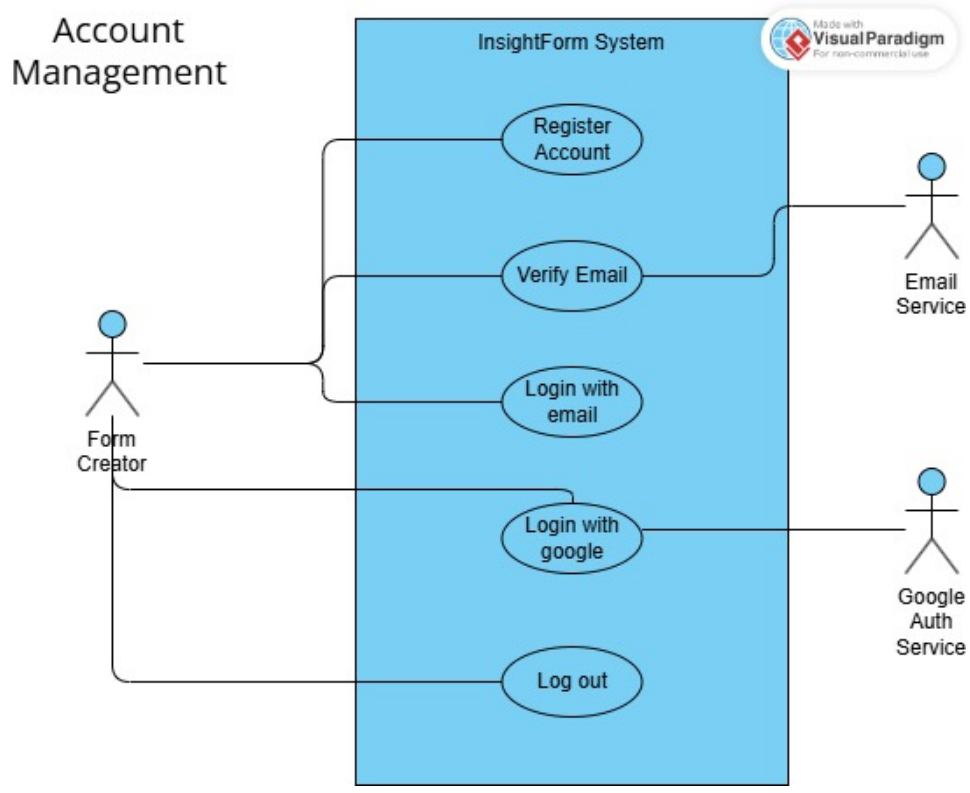
* 1. The system displays the report page for the selected form.
* 2. The creator requests to generate a new AI report
* 3. The system checks if the creator has remaining generation attempts and if new responses have been collected since the last report.
* 4. If the checks pass, the backend pre-processes all responses (summarizing quantitative data) and sends the data to the LLM API with a structured prompt
* 5. The LLM returns a JSON object containing a summary and a list of structured suggestions.
* 6. The system saves this output as a new Report document in the database.
* 7. The system updates the Form document with the new lastReportResponseCount and decrements the reportTryCount
* 8. The UI displays the new summary, suggestions, and data visualizations.

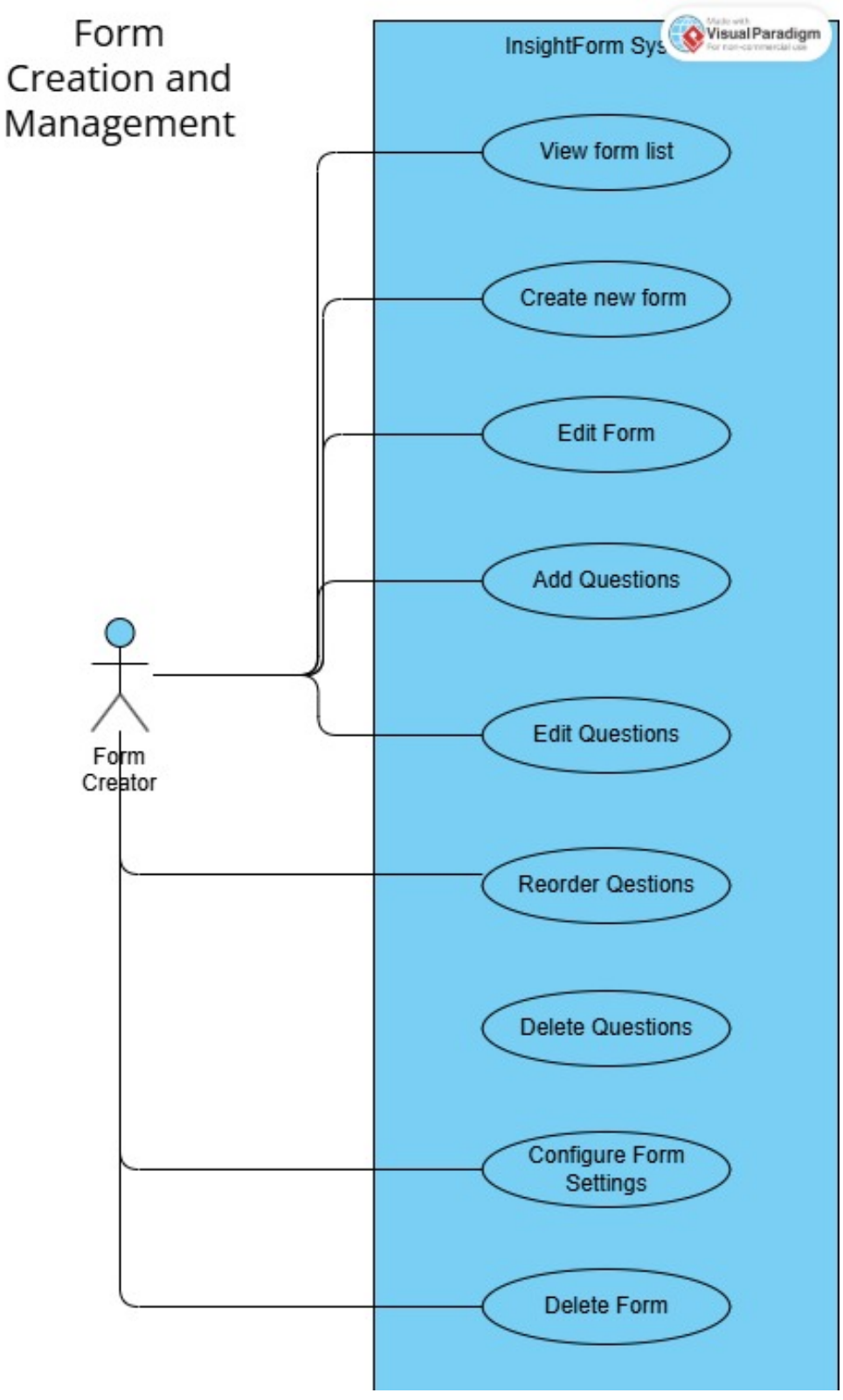
– **Alternate flows/Exceptions:**

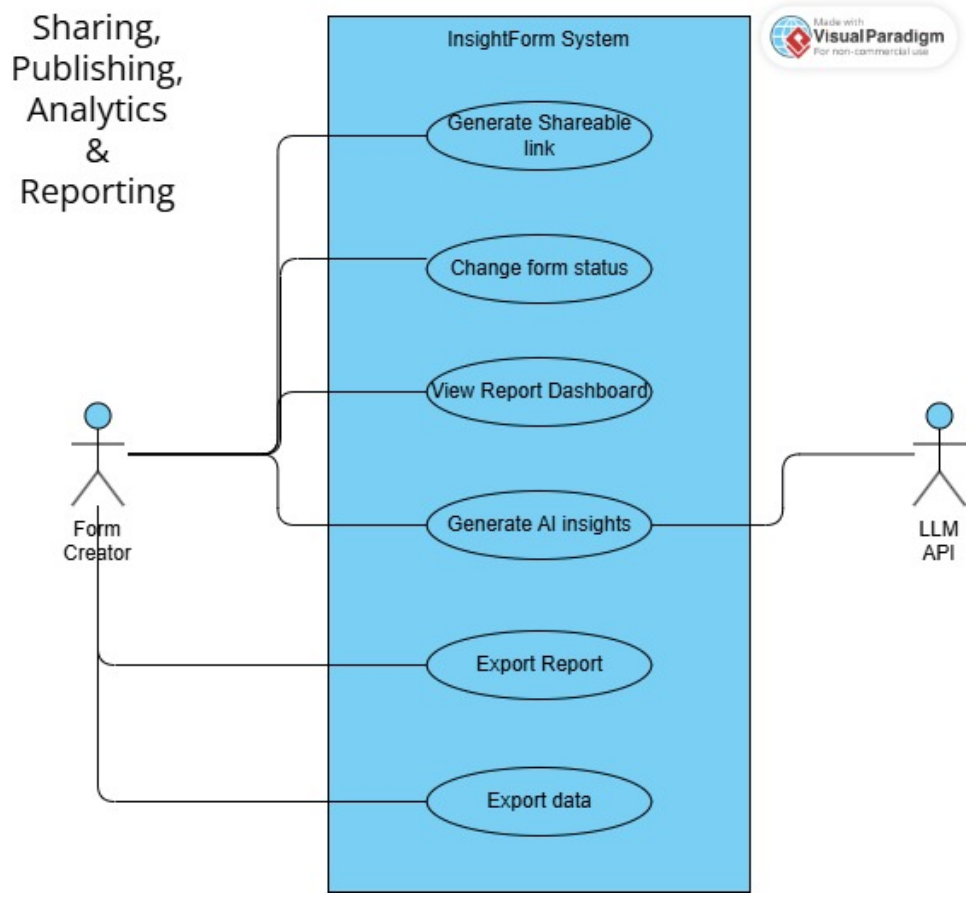
* A1: View Cached Report: If no new responses have been collected, the system skips the LLM call and displays the most recently generated report
* E1: No Attempts Remaining: If the reportTryCount is zero, the system disables the "Generate" button and informs the user they have no attempts left.
* E2: LLM API Failure: If the external API call fails, the system returns an error message to the user.

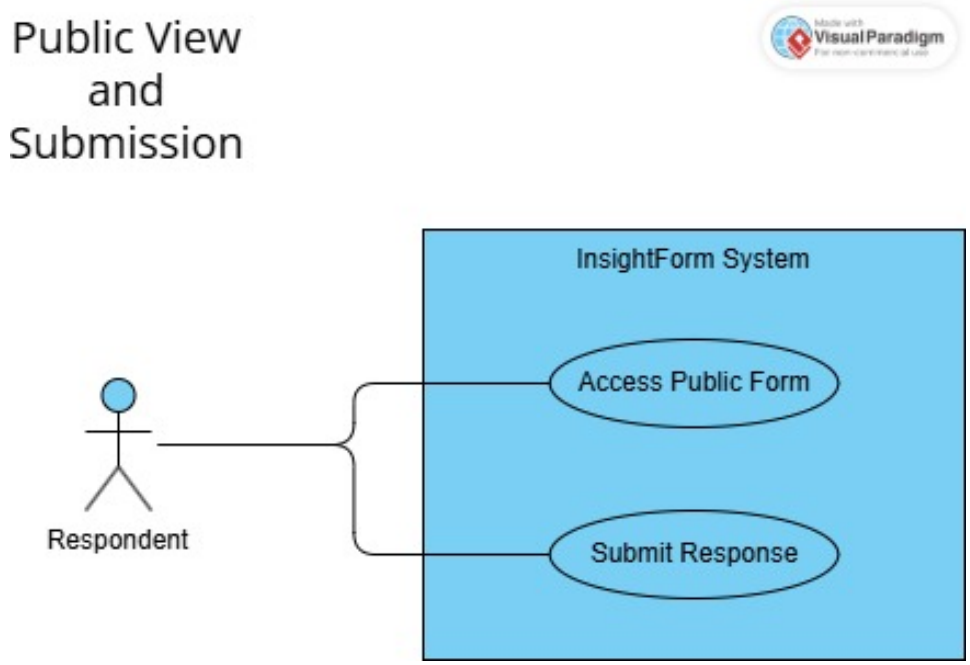
11 UML Diagrams

- **Use Case Diagrams**: A Use Case Diagram helps in understanding the functionality of a system from the perspective of end-users and how the system interacts with different actors (users or systems). It provides a visual representation of the system's functionality, showing what tasks users can perform, and the system's responses to those tasks.

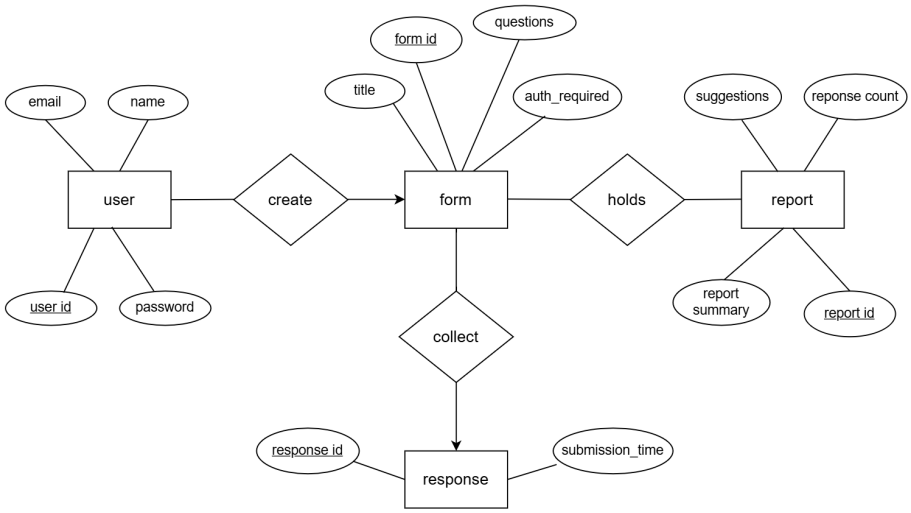



Form
Creation and
Management
InsightForm Sys
View form list
Create new form
Edit Form
Add Questions
Edit Questions
Reorder Qestions
Delete Questions
Configure Form
Settings
Delete Form
Form
Creator


Sharing, Publishing, Analytics & Reporting
InsightForm System
Generate Shareable link
Change form status
View Report Dashboard
Generate AI insights
Export Report
Export data
Form Creator
LLM API


Public View and Submission
InsightForm System
Access Public Form
Submit Response
Respondent

- **Entity Reltaionship Diagrams**:



12 Modules or Major Features

- **User Authentication:** Sign Up, Login, Google OAuth, Forgot Password
- **Dashboard:** View, edit, delete, share forms, see reports
- **Form Creation Workflow:**
  - Step 1: Define Objective
  - Step 2: AI generates context-based questions
  - Step 3: User selects or edits these questions
  - Step 4: Form is live for public submissions
- **Public Form Page:** Responsive and accessible for external users
- **AI-Powered Report Page:**
  - Summary insights with recommendations
  - Raw data in structured table
- **Exports:** PDF for summaries, Excel/CSV for raw data
- **Real-Time Notifications:** WebSocket-based alerts and dashboard badges

13 Modeling Approach

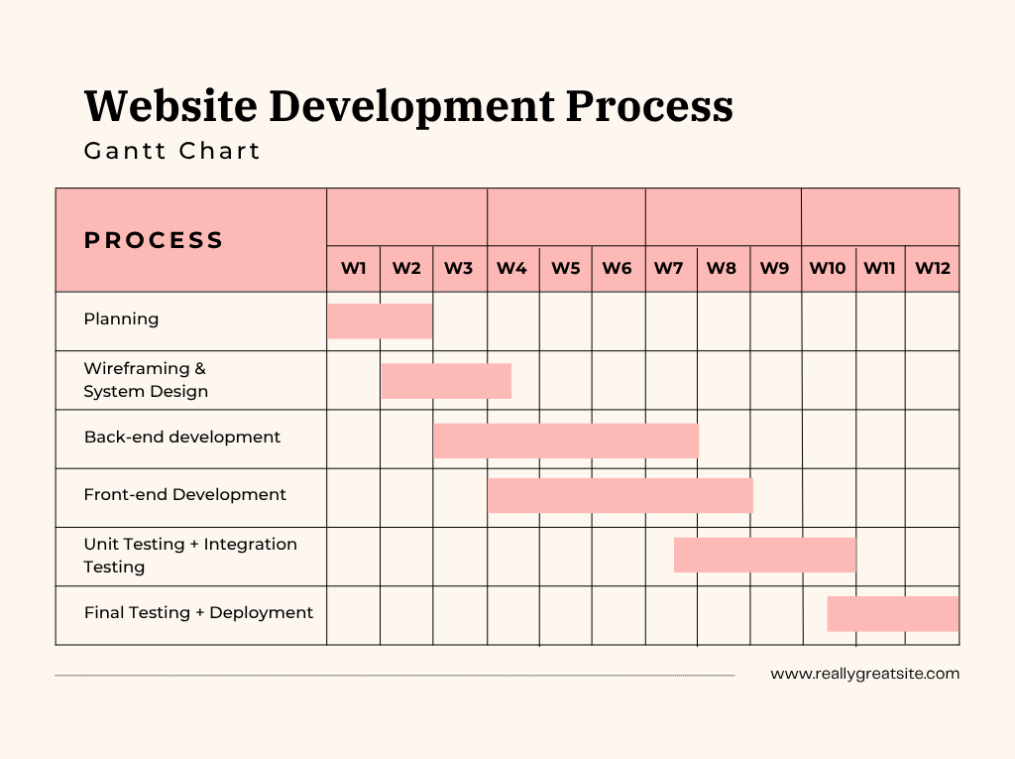
- **Chosen Approach**: For the InsightForm project, a Scenario-based modeling approach has been chosen as the primary method for defining and analyzing the system's requirements.

  This approach focuses on describing the system's functionality from the perspective of the end-users (the actors). The primary tools used within this approach are Use Cases and their corresponding diagrams and detailed descriptions, which capture the interactions between the actors and the system.

- **Justification**
  - **User-Centric Design**: The core of InsightForm is its interaction with two distinct types of users: Form Creators and Respondents. A scenario-based model, which is built around user goals and journeys, ensures that the system's design and functionality are always aligned with the needs and expectations of these users.
  - **Clear Functional Requirements**: By defining requirements as a series of use cases (e.g., "Create Form," "Submit Response," "Generate AI Report"), this approach provides a clear, comprehensive, and unambiguous list of what the system must do. It makes it easy to trace every feature back to a specific user need.
  - **Effective Communication**: Use Case descriptions are written in a semi-formal, narrative style that is easily understood by all project stakeholders, including developers, designers, and potential clients, regardless of their technical background.
  - **Complements Other Models**: While the Scenario-based approach is primary, it is effectively supplemented by other models to provide a complete picture. For instance, the Data model (represented by the ER Diagram) describes the data structures required for the scenarios, and Flow-oriented models (represented by Activity Diagrams) detail the internal logic of the more complex use cases.

14 Tentative Timeline / Work Plan

Week	Tasks
1	Requirements gathering, competitor analysis, user stories, and schema design
2	UI/UX wireframes for dashboard, form builder, and reports; finalize tech stack
3	Backend setup (Express + MongoDB); define models and test basic APIs
4	Authentication system (JWT, Google OAuth); setup user roles & protected routes
5	Frontend setup with React & Tailwind; layout, navbar, and dashboard skeleton
6	Objective input module + AI question suggestion using OpenAI
7	Form Builder (Add/Edit/Delete questions); MongoDB sync and live preview
8	Public form rendering, client-side validations, response handling
9	AI Report Generation: summary, recommendations, and charts
10	PDF and Excel export features integration
11	Real-time notification system with WebSockets; UI alerts and dashboard updates
12	Final testing, bug fixes, performance polish, deployment setup, documentation



15 Why This Project?

InsightForm stands out by blending AI with form generation and response interpretation. It meets the rising demand for smarter tools in:

- Market Research
- Internal Surveys
- Feedback loops in agile teams
- Educational research

This project also reflects the increasing shift toward automation in business analysis. With limited human bandwidth, intelligent tooling can save time and boost decision quality.

Key motivations include:

- Eliminating guesswork in question formulation
- Fast-tracking the path to actionable data
- Reducing the learning curve in data analysis

16 Expected Challenges

- **AI Accuracy:** Fine-tuning prompts for useful suggestions
- **State Management:** Preventing form data inconsistency
- **Scalability:** Supporting many users and large datasets
- **Security:** Ensuring privacy in forms and data storage
- **User Education:** Helping users utilize AI without training

17 User Interface Screenshots

This section presents the screenshots of the website user interface, showcasing each page and key feature.

Home Page

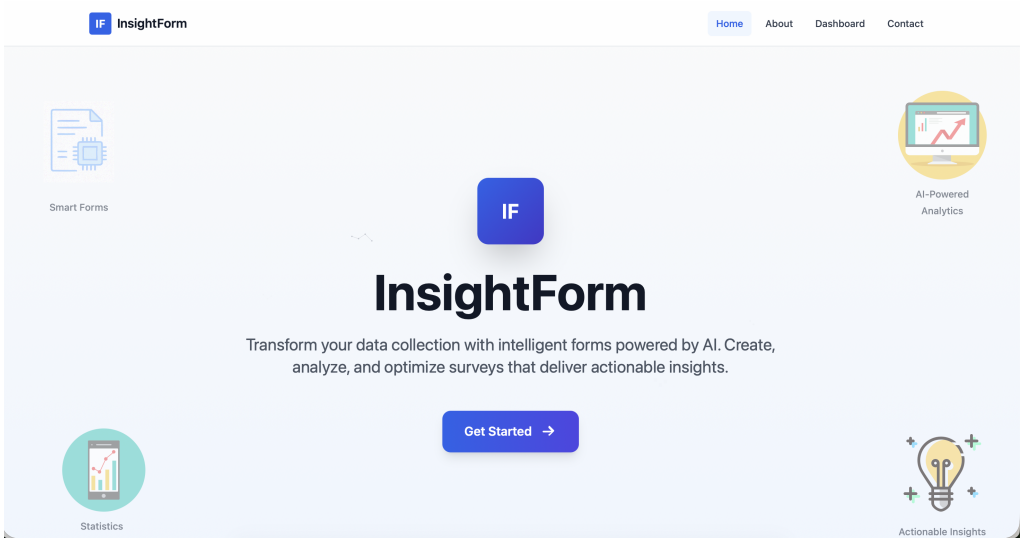


Figure 1: Homepage of the website displaying navigation and introductory content.

Login Page

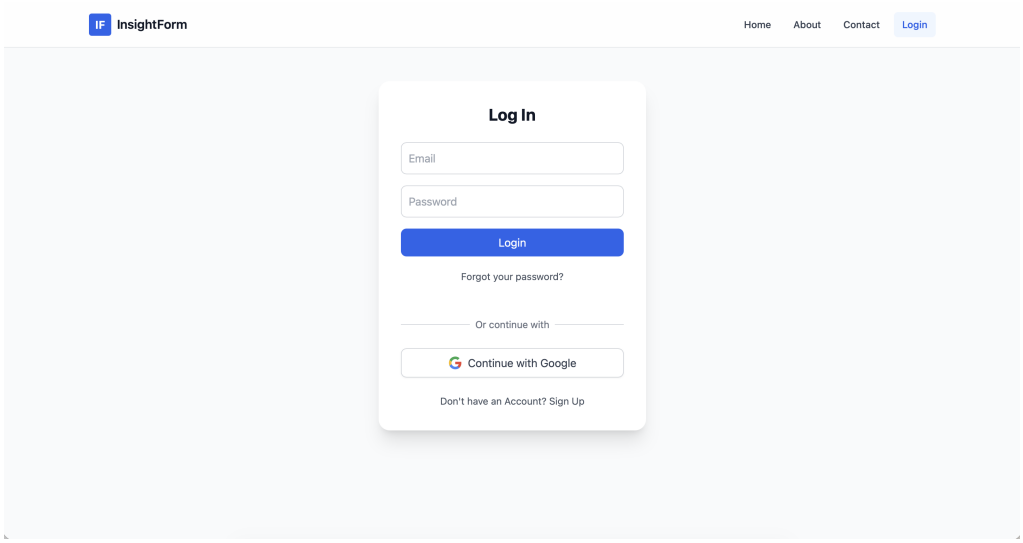


Figure 2: Login page for user authentication.

About Page

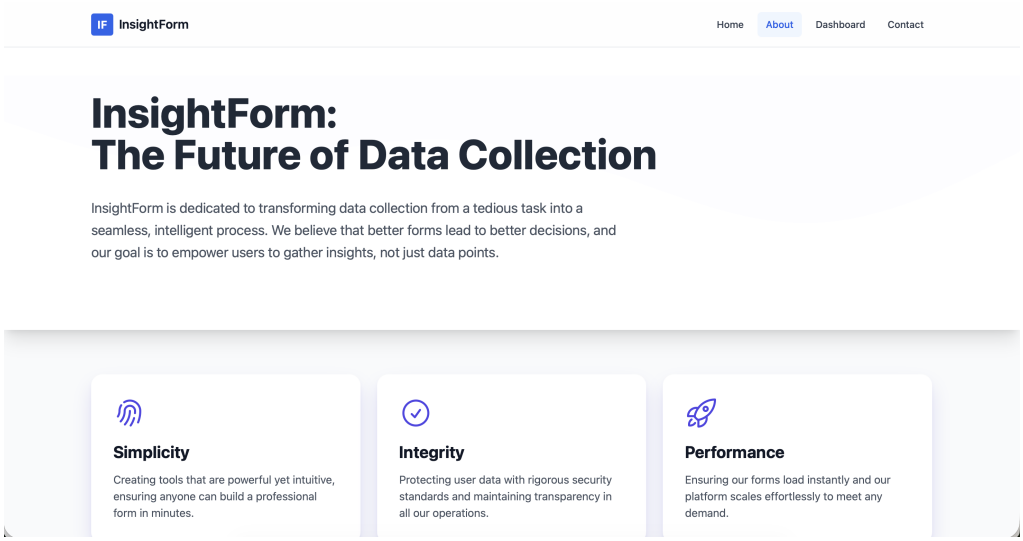


Figure 3: About page describing the purpose and background of the project.

Contact Page

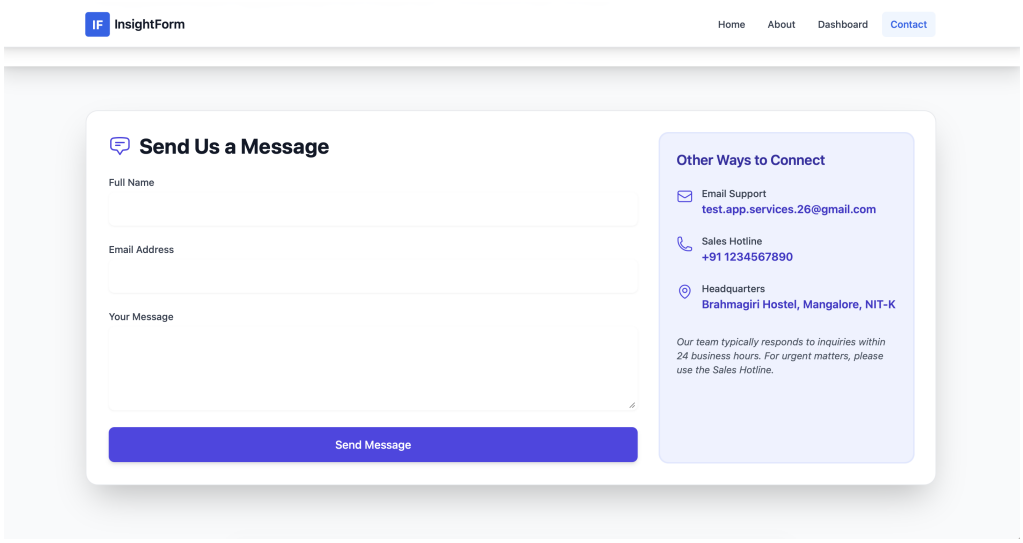


Figure 4: Contact page allowing users to reach out to the administrators.

Dashboard View

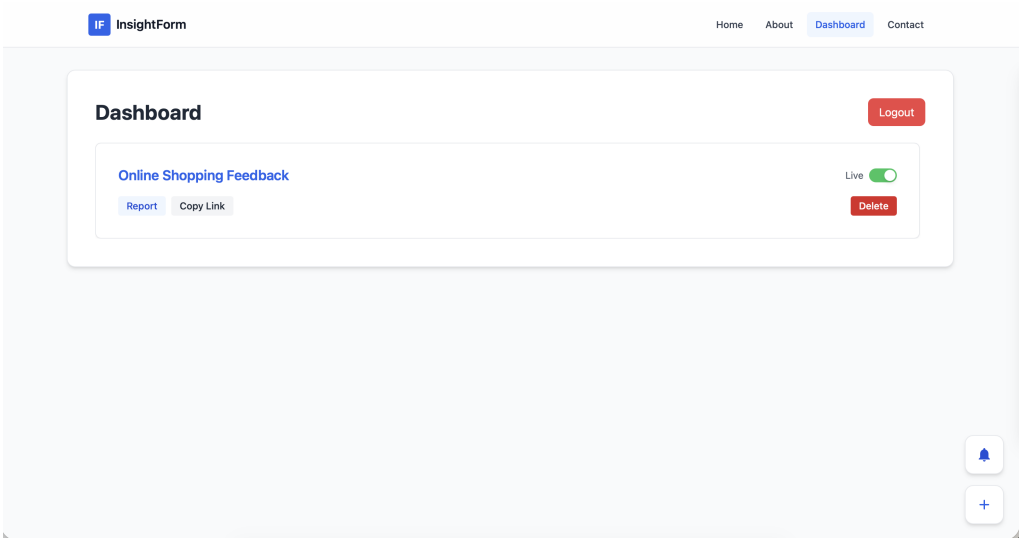


Figure 5: Dashboard displaying available forms, form controls and notification.

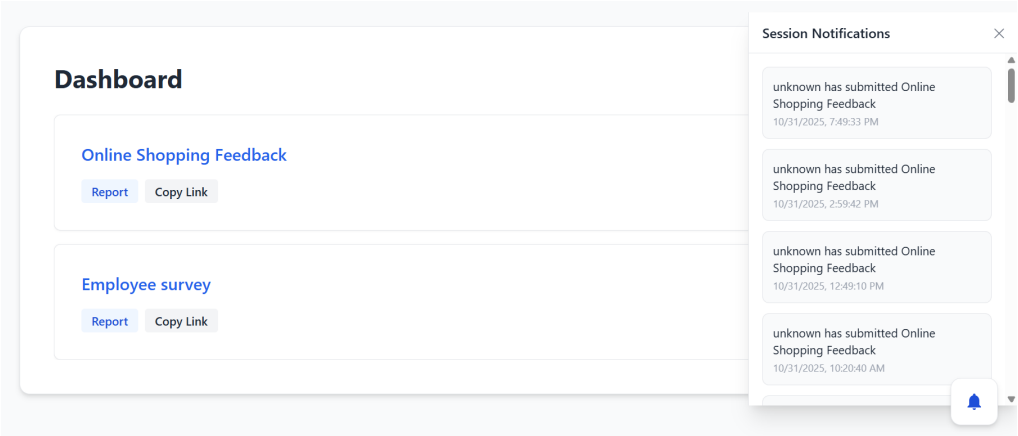


Figure 6: Real-time Notifications when user submit the form.

Form Builder/Editor Page

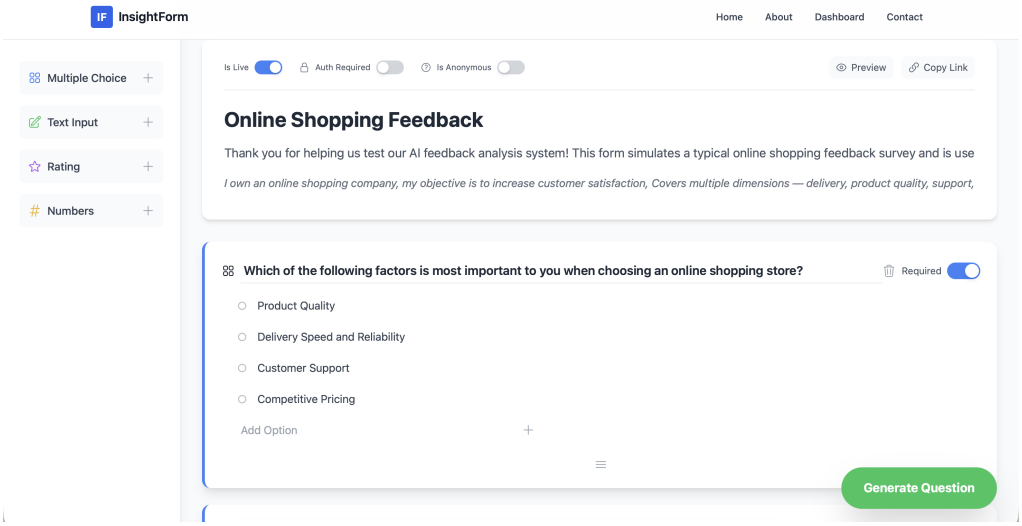


Figure 7: Form Editor to edit and change form settings.

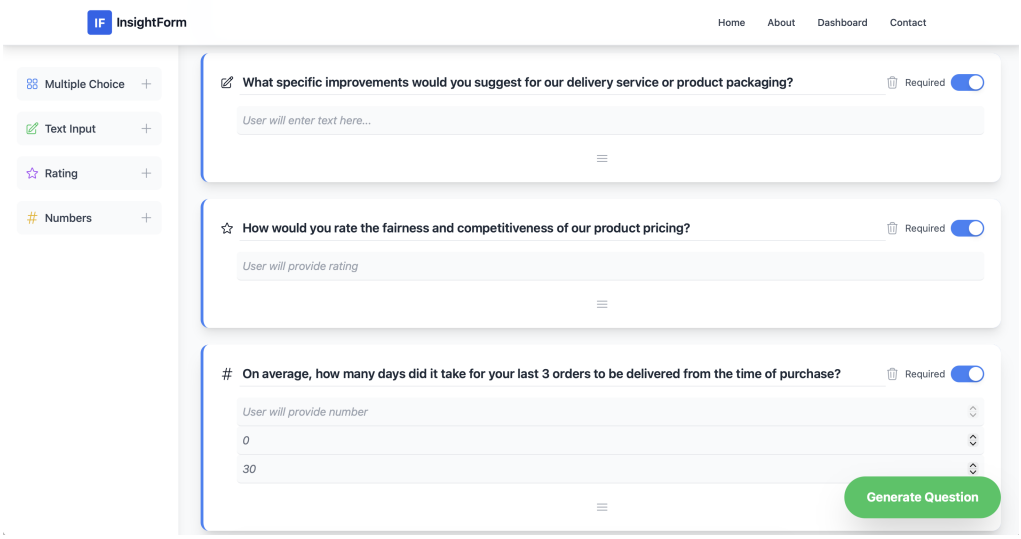


Figure 8

Form Submission Page

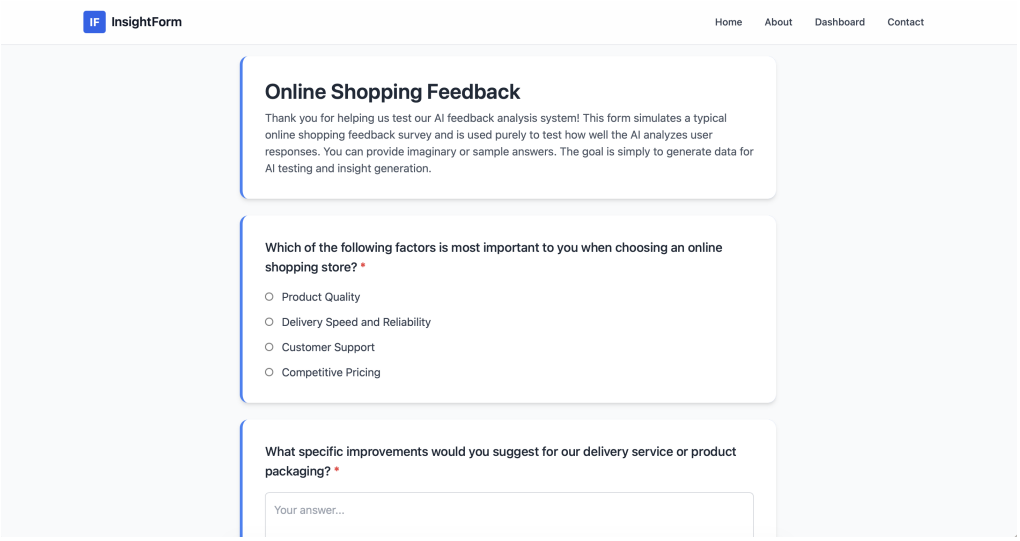


Figure 9

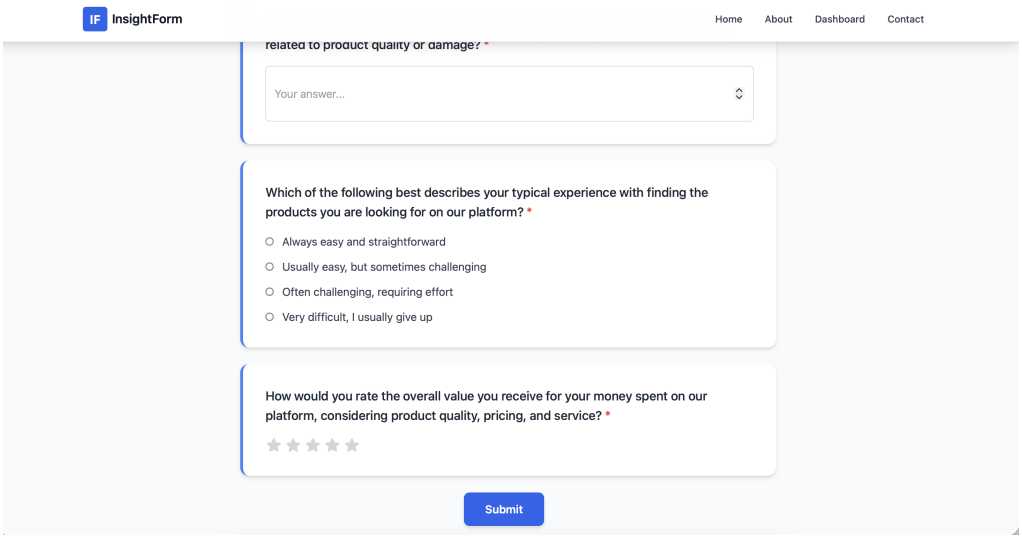


Figure 10

AI Generated Summary

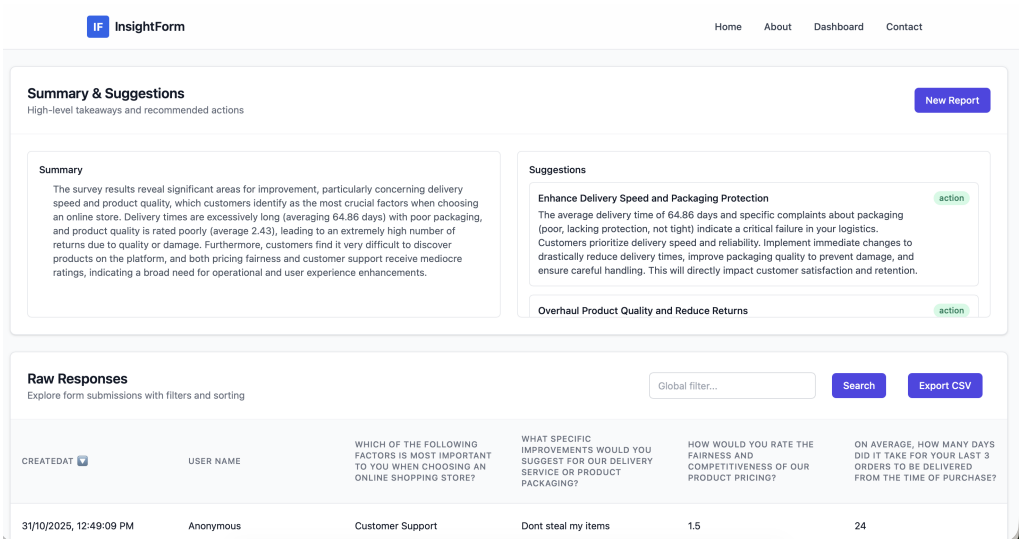


Figure 11: Summary page displaying key performance insights.

Raw Response

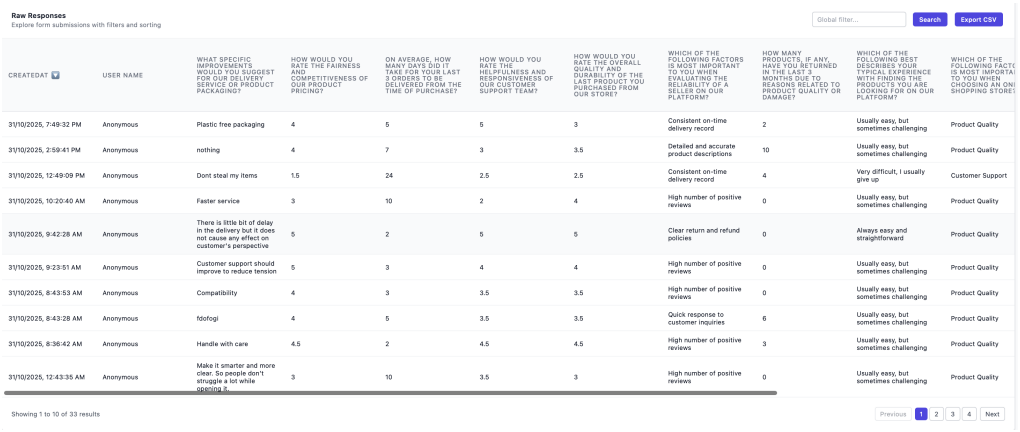


Figure 12: Responses shown in tabular format provided by the respondent.

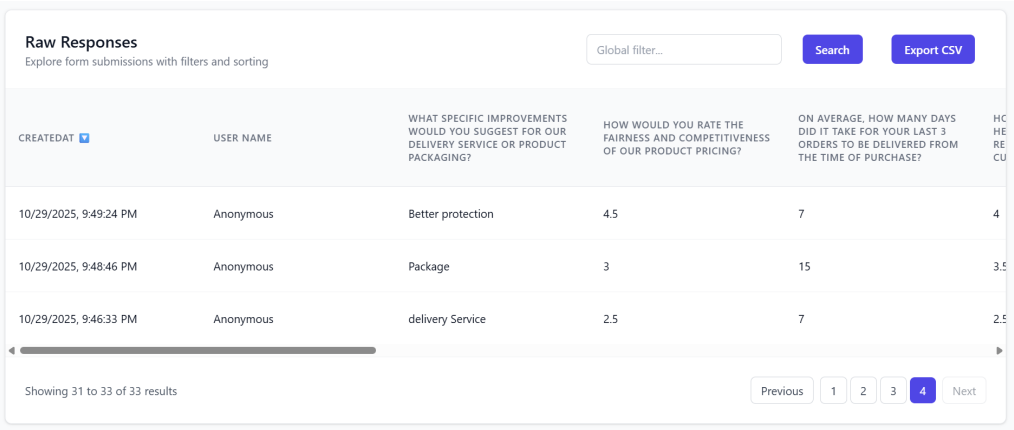


Figure 13

Graphical Representation of Data

Figure 14: Numerical Type Questions

Figure 15: Multiple Choice Questions

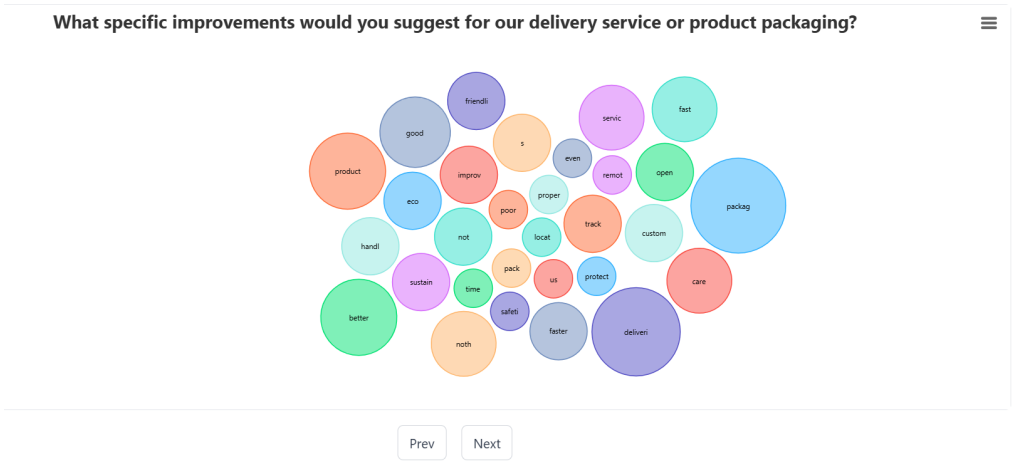


Figure 16: Text Based Questions

Figure 17: Rating Based Questions

18 References

1. Gemini API Documentation
2. MongoDB Documentation
3. React.js Documentation
4. Tailwind CSS Documentation
5. Socket.io Documentation
6. jsPDF
7. SheetJS (xlsx)
8. Google Forms, Typeform (Comparative Research)
9. UX Design Principles – NNGroup, Material UI