# Survey Form Web Application

## Introduction

The **Survey Form Web Application** is a user-friendly platform developed using the MERN stack—MongoDB, Express.js, React.js, and Node.js. It enables users to create, distribute, and analyze surveys efficiently, catering to a wide range of users including researchers, educators, marketers, and business owners.

**Key Features:**

1. **Intuitive Survey Creation:**
    a. **Diverse Question Types:** Users can design surveys incorporating various question formats such as multiple-choice, short answer, rating scales, and dropdowns, allowing for comprehensive data collection.
    b. **Dynamic Form Builder:** The application provides a flexible interface to add, edit, or remove questions and options, adapting to evolving survey requirements.
2. **Seamless Distribution:**
    a. **Multi-Channel Sharing:** Surveys can be distributed through unique links, email invitations, or embedded directly on websites, ensuring broad reach and engagement.
    b. **Responsive Design:** The surveys are optimized for all devices, offering a smooth experience on desktops, tablets, and smartphones.

3. **Real-Time Analytics and Reporting:**
    a. **Instant Data Access:** Users receive immediate insights through an intuitive dashboard displaying response rates, demographic breakdowns, and visual reports like charts and graphs.
4. **Robust Security Measures:**
    a. **Data Protection:** The application employs encryption and access controls to safeguard respondent data, ensuring that only authorized individuals can view or manage survey results.
    b.

**Scenario-Based Case Study:**

*ABC Electronics*, a mid-sized consumer electronics company, utilizes the Survey Form Web Application to enhance customer satisfaction. By designing a comprehensive survey with various question types, customizing it to reflect the company's branding, and distributing it through multiple channels, ABC Electronics gathers valuable feedback. Real-time analytics reveal strengths in product quality but highlight concerns in after-sales support. This insight leads to targeted improvements in support processes, staff training, and the introduction of a more efficient ticketing system, ultimately enhancing overall customer satisfaction.

**Technical Architecture:**

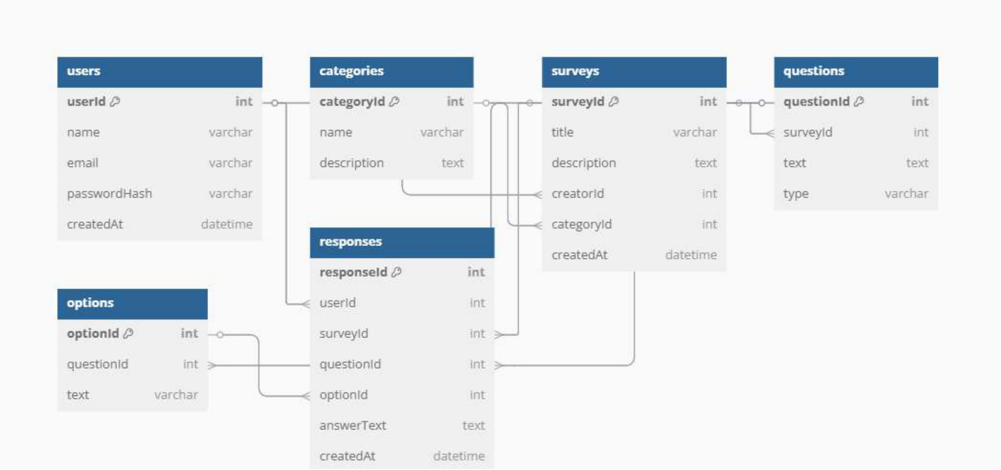
The application comprises several key components:

- **User Interface:** Developed with React.js, providing an intuitive platform for users to interact with the application, including creating, managing, and filling out surveys.

- **API Gateway:** Routes client requests to appropriate services, ensuring security and scalability.
- **Authentication Service:** Manages user authentication, verifying credentials, and handling user sessions to secure access.
- **Database:** MongoDB stores all user data, survey forms, responses, and metadata securely, ensuring data integrity and accessibility.
- **Web Server:** Hosts the application, handling client requests, serving web pages, and managing communication between the frontend and backend.
- **Survey Management Features:** Allow users to create and customize surveys, view and manage responses, and distribute surveys through various channels.

This architecture ensures a seamless, secure, and efficient experience for users, from survey creation to data analysis.

# ER Diagram



| users | |
|---|---|
| userId | int |
| name | varchar |
| email | varchar |
| passwordHash | varchar |
| createdAt | datetime |

| categories | |
|---|---|
| categoryId | int |
| name | varchar |
| description | text |

| surveys | |
|---|---|
| surveyId | int |
| title | varchar |
| description | text |
| creatorId | int |
| categoryId | int |
| createdAt | datetime |

| questions | |
|---|---|
| questionId | int |
| surveyId | int |
| text | text |
| type | varchar |

| responses | |
|---|---|
| responseId | int |
| userId | int |
| surveyId | int |
| questionId | int |
| optionId | int |
| answerText | text |
| createdAt | datetime |

| options | |
|---|---|
| optionId | int |
| questionId | int |
| text | varchar |

## ENTITIES & ATTRIBUTES

1. **`users`**

a. Represents the users of the application (both survey creators and respondents).

b. **Attributes**:

i. `userId`: Primary key (auto-incremented).

ii. `name`, `email`: User details.

iii. `passwordHash`: Hashed password for authentication.

iv. `createdAt`: Timestamp of account creation.

2. **`categories`**

a. Used to group surveys into categories (e.g., Health, Education).

b. **Attributes**:

i. `categoryId`: Primary key.

ii. `name`, `description`: Category details.

3. **`surveys`**

a. Represents individual surveys created by users.

b. **Attributes**:

i. `surveyId`: Primary key.

ii. `title`, `description`: Metadata of the survey.

iii. `creatorId`: Foreign key → `users.userId`.

iv. `categoryId`: Foreign key → `categories.categoryId`.

v. `createdAt`: Timestamp of survey creation.

4. **`questions`**

a. Questions that belong to a specific survey.

b. **Attributes**:

i. `questionId`: Primary key.

ii. `surveyId`: Foreign key → `surveys.surveyId`.

iii. `text`: The question itself.

iv. `type`: Question type (e.g., text, MCQ).

5. **`options`**

a. Options associated with multiple-choice questions.

b. **Attributes**:

i. `optionId`: Primary key.

ii. `questionId`: Foreign key → `questions.questionId`.

iii. `text`: Option text.

6. **`responses`**

a. Stores user responses to survey questions.

b. **Attributes**:

i. `responseId`: Primary key.

ii. `userId`: Foreign key → `users.userId`.
iii. `surveyId`: Foreign key → `surveys.surveyId`.
iv. `questionId`: Foreign key → `questions.questionId`.
v. `optionId`: Foreign key → `options.optionId` (nullable for text answers).
vi. `answerText`: For text-based answers.
vii. `createdAt`: Time of submission.

### 🔗 RELATIONSHIPS (Cardinalities)

- **users** ↔ **surveys**: One-to-Many

A user can create many surveys (`creatorId`).

- **surveys** ↔ **categories**: Many-to-One

Each survey belongs to a single category.

- **surveys** ↔ **questions**: One-to-Many

A survey contains multiple questions.

- **questions** ↔ **options**: One-to-Many

Each question (if multiple choice) has several options.

- **users** ↔ **responses**: One-to-Many

A user can submit many responses.

- **questions** ↔ **responses**: One-to-Many

Each question can have many responses.

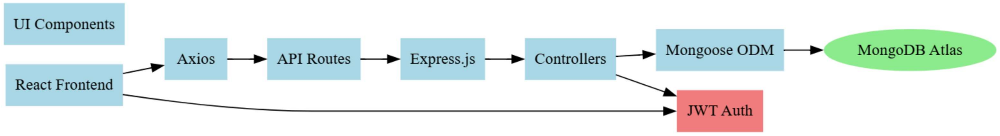
- **options** ↔ **responses**: One-to-Many

An option can appear in many responses (nullable for text responses).

### ✅ Summary

- This ER diagram supports both multiple-choice and free-text answers.
- It separates concerns using normalized tables (e.g., `questions`, `options`, `responses`).
- Easily scalable: You can add survey settings, expiry times, or analytics features.

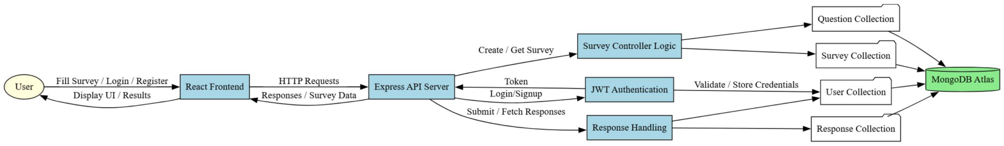
## Workflow Diagram



## 🧩 SurveyAppMERN Workflow

1. **User Accesses Survey** - A user navigates to the survey link provided by the administrator
2. **Survey Display** - The frontend (React.js) fetches the survey details from the backend (Node.js/Express.js) via API calls - The survey is presented to the user, displaying questions and answer options
3. **User Submits Responses** - The user fills out the survey and submits their responses
4. **Backend Processing** - The backend receives the responses and performs validation checks to ensure data integrity - CAPTCHA verification is performed to prevent spam submissions
5. **Data Storage** -Validated responses are stored in MongoDB for future retrieval and analysis
6. **Admin Accesses Responses** -The administrator logs into the admin panel - The frontend fetches the survey responses from the backend -The administrator can view and analyze the collected data
7. **Review Prompt** - After survey completion, users are prompted to leave an online review, enhancing feedback collection

# Dataflow Diagram



The **SurveyAppMERN** project is a full-stack survey platform developed using the MERN stack—MongoDB, Express.js, React.js, and Node.js. The platform enables administrators to create and manage surveys, collect customer feedback, and guide users to leave online reviews. It incorporates features such as authentication, input validation, CAPTCHA protection, and integration with Google Maps.

**Data Flow Overview:**

1. **User Interaction (Frontend):**
a. **Survey Access:** Respondents access the survey through the React.js frontend, which is designed to be user-friendly and responsive.
b. **Response Submission:** Users provide their feedback by filling out the survey forms, which may include various question types such as multiple-choice, ratings, and text inputs.
c. **Review Prompt:** After completing the survey, users are prompted to leave an online review, enhancing the feedback loop.
2. **Data Transmission (Frontend to Backend):**
a. **API Requests:** Upon submission, the React.js frontend sends HTTP requests to the Express.js backend, transmitting the

survey responses.

b. **Validation:** The backend performs validation checks on the received data to ensure completeness and correctness.

c. **Security Measures:** CAPTCHA verification is implemented to prevent spam and ensure that responses are submitted by humans.

3. **Data Storage (Backend):**

a. **Database Interaction:** Validated responses are stored in MongoDB, a NoSQL database, allowing for flexible and scalable data storage.

b. **Data Retrieval:** Administrators can retrieve and analyze survey data from MongoDB through the backend API.

4. **Administration (Frontend and Backend):**

a. **Admin Dashboard:** Administrators access a React.js-based dashboard to monitor survey responses, analyze feedback, and manage surveys.

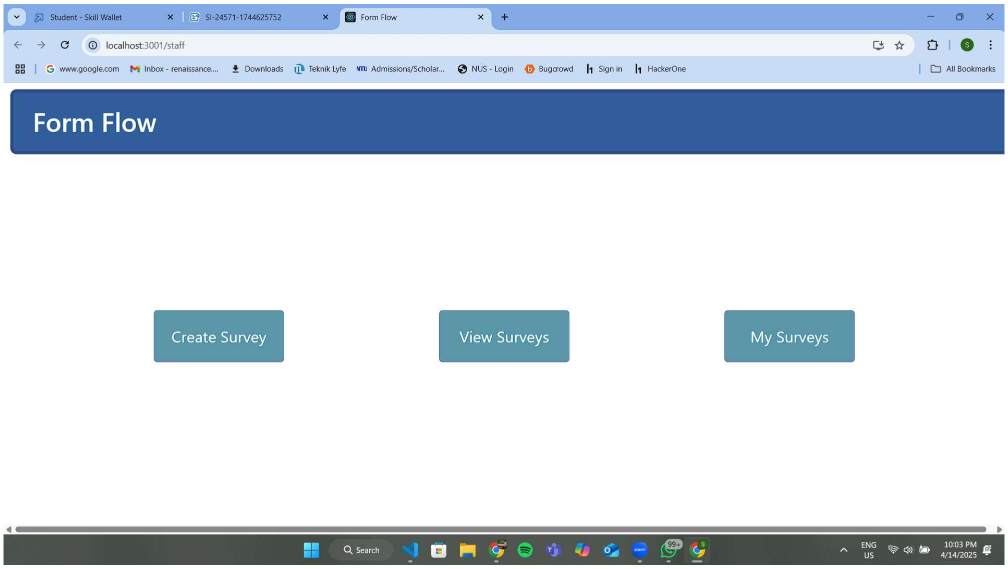
b. **Authentication:** Secure login mechanisms are in place to ensure that only authorized personnel can access administrative features.

**Additional Features:**

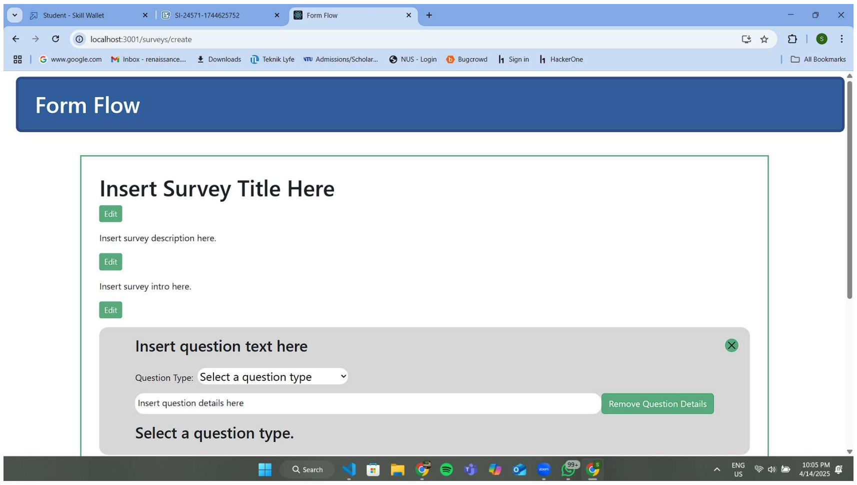
- **Google Maps Integration:** The platform integrates with Google Maps to collect location data from respondents, providing geographical insights into the survey responses.

# Project Implentation

- ## Landing Page (if logged in)



- ## Create Survey



Insert Survey Title Here

Insert survey description here.

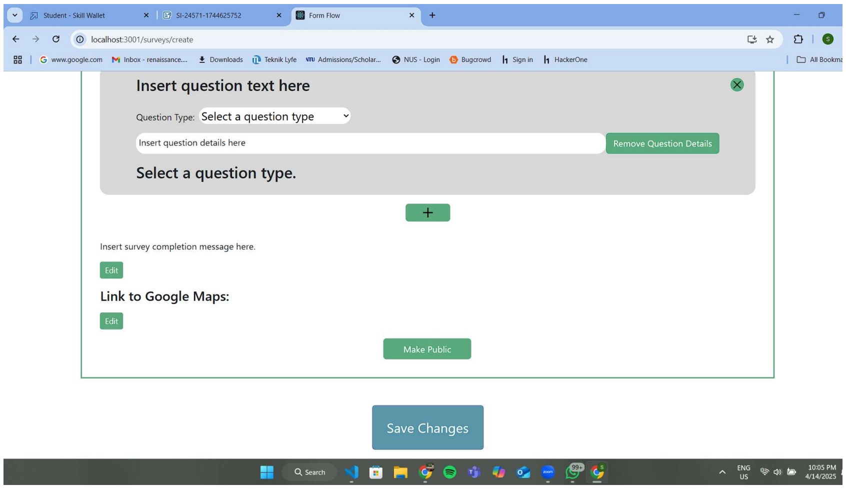
Insert survey intro here.

Insert question text here

Question Type: Select a question type

Insert question details here

Select a question type.



Insert question text here

Question Type: Select a question type

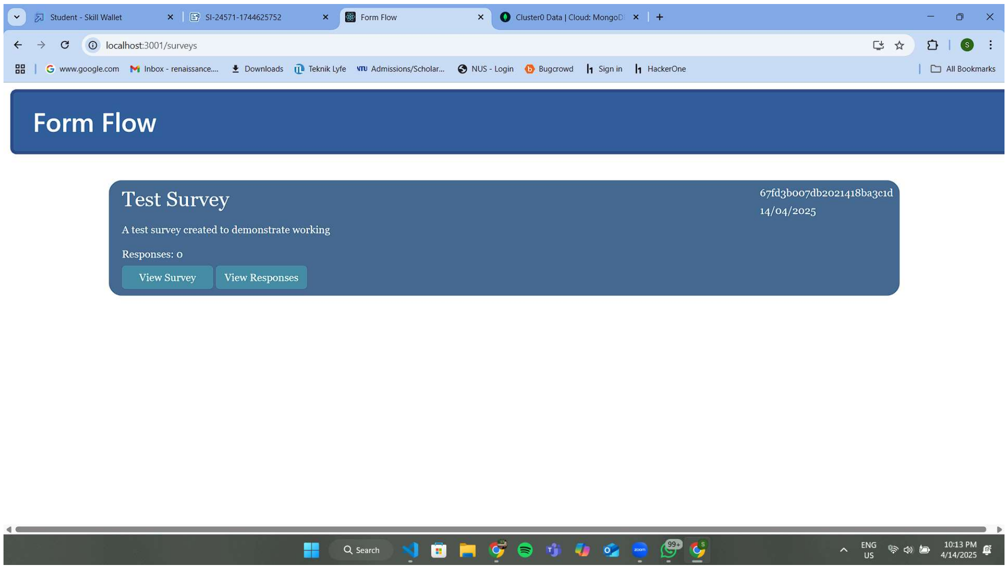
Insert question details here

Select a question type.

Insert survey completion message here.

Link to Google Maps:

- ## View Survey



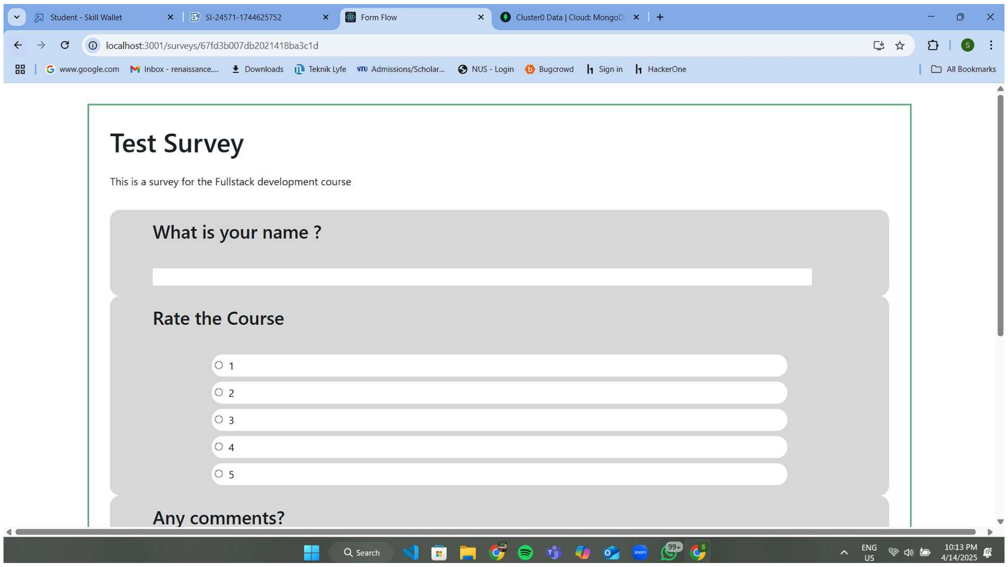
Test Survey

67fd3b007db2021418ba3c1d

14/04/2025

A test survey created to demonstrate working

Responses: 0



# Test Survey

This is a survey for the Fullstack development course

## What is your name ?

## Rate the Course

○ 1

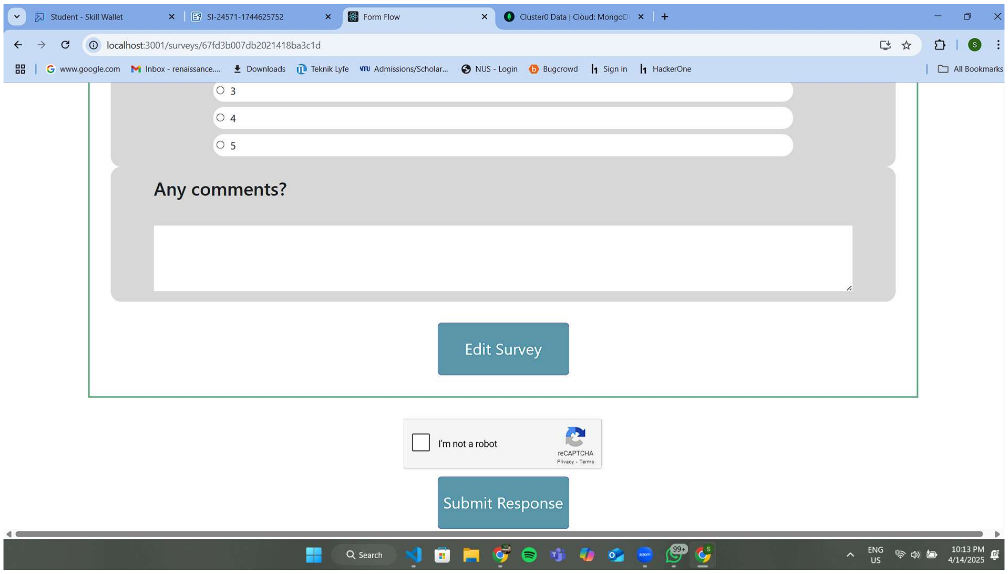
○ 2

○ 3

○ 4

○ 5

## Any comments?



○ 3

○ 4

○ 5

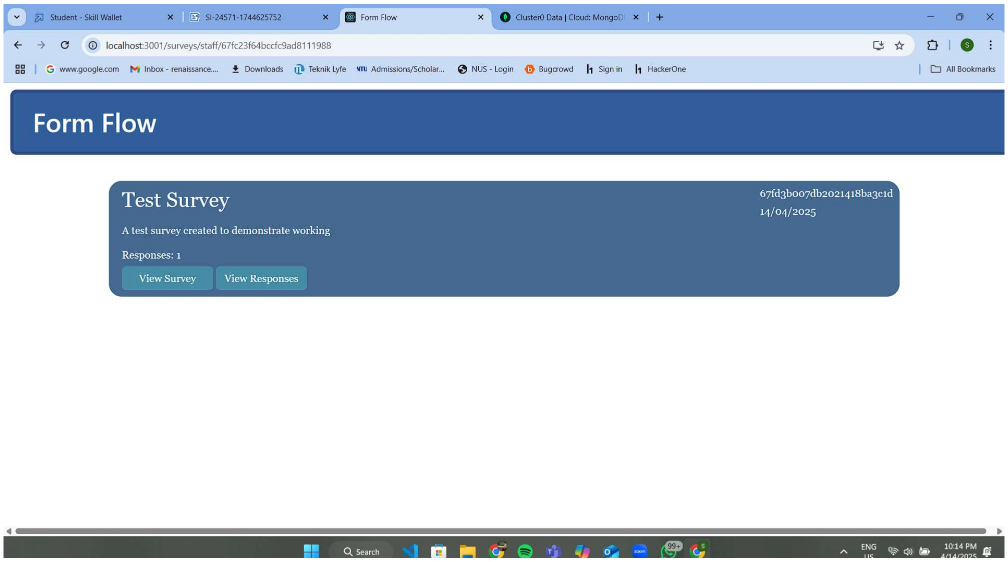
## Any comments?

Edit Survey

□ I'm not a robot

Submit Response

- ## My Survey



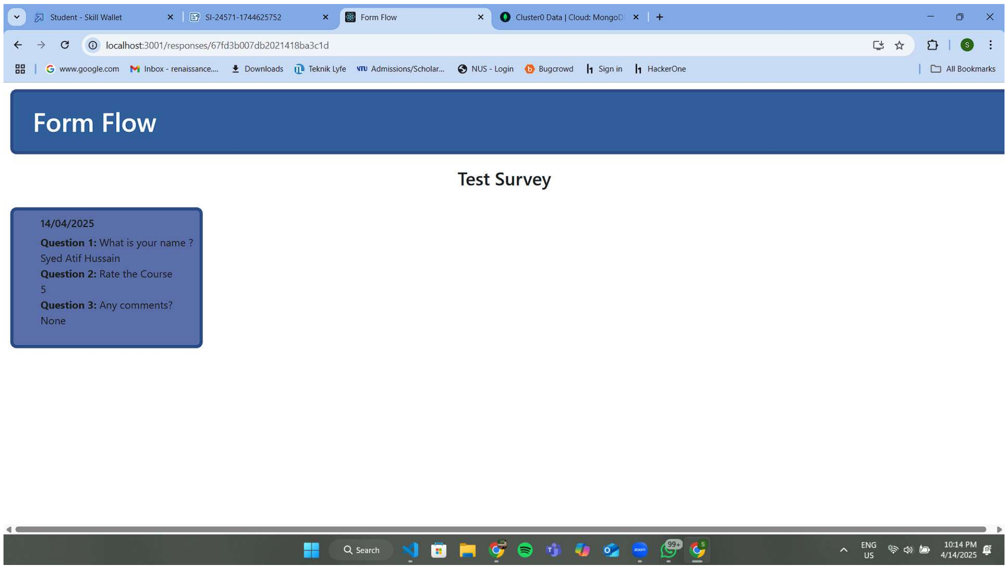
Form Flow

Test Survey

67fd3b007db2021418ba3c1d
14/04/2025

A test survey created to demonstrate working

Responses: 1

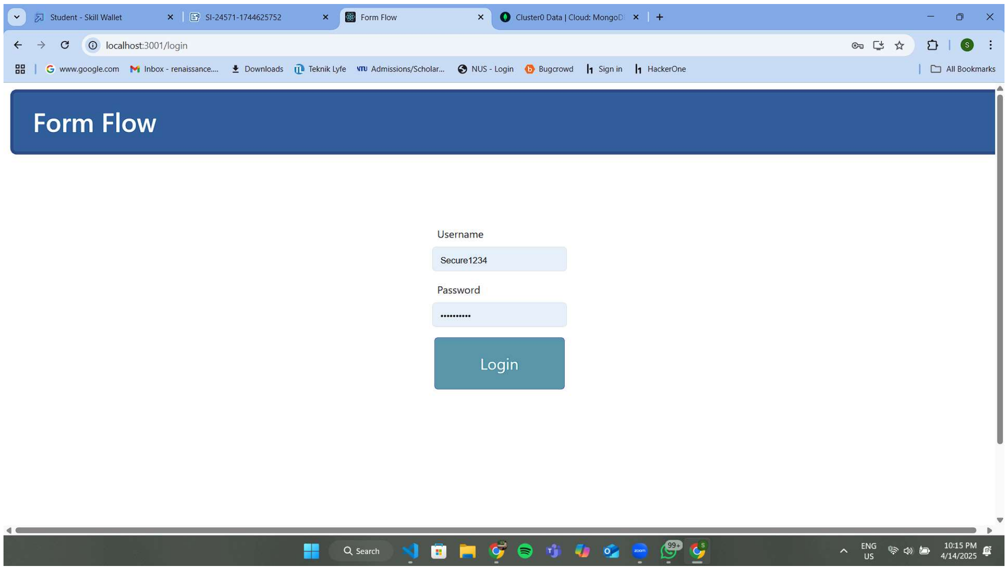


Form Flow

Test Survey

14/04/2025
**Question 1:** What is your name ?
Syed Atif Hussain
**Question 2:** Rate the Course
5
**Question 3:** Any comments?
None

- **Login**



Form Flow

Username

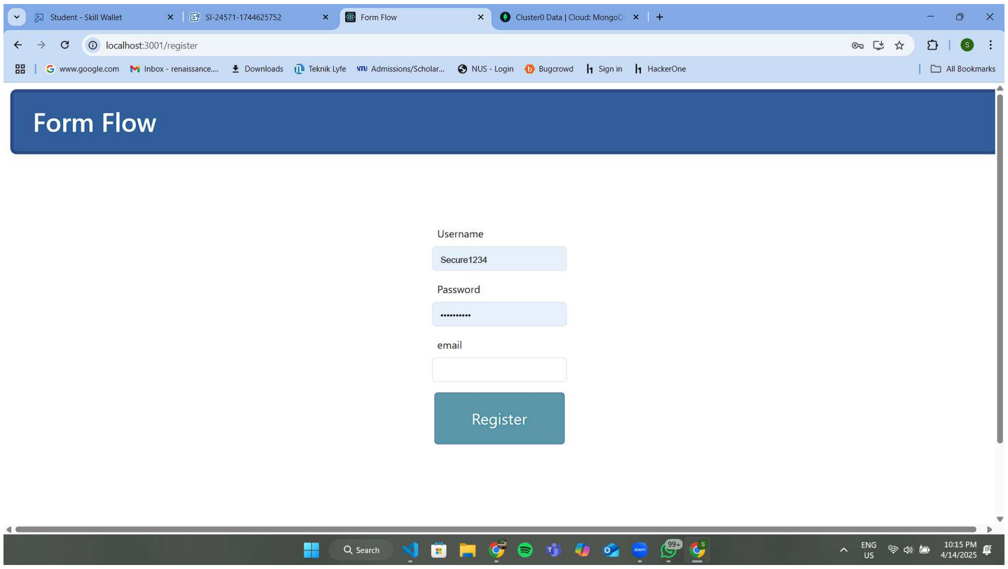
Secure1234

Password

••••••••••

Login

- **Register**



Form Flow

Username

Secure1234

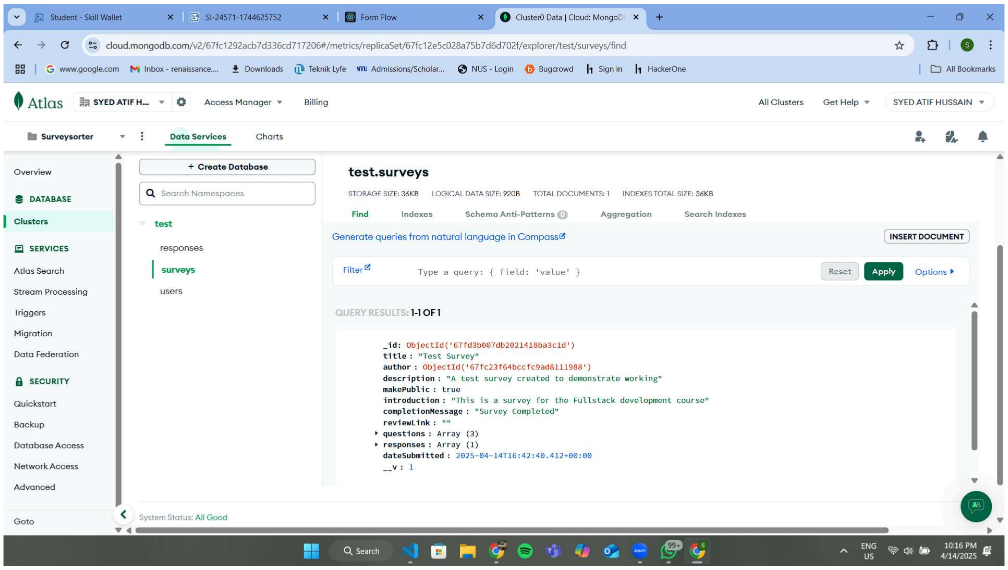
Password

••••••••••

email

Register

- **Database (MongoDB)**



cloud.mongodb.com/v2/67fc1292acb7d336cd717206#/metrics/replicaSet/67fc12e5c028a75b7d6d702f/explorer/test/surveys/find

**test.surveys**

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 920B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 36KB

Generate queries from natural language in Compass

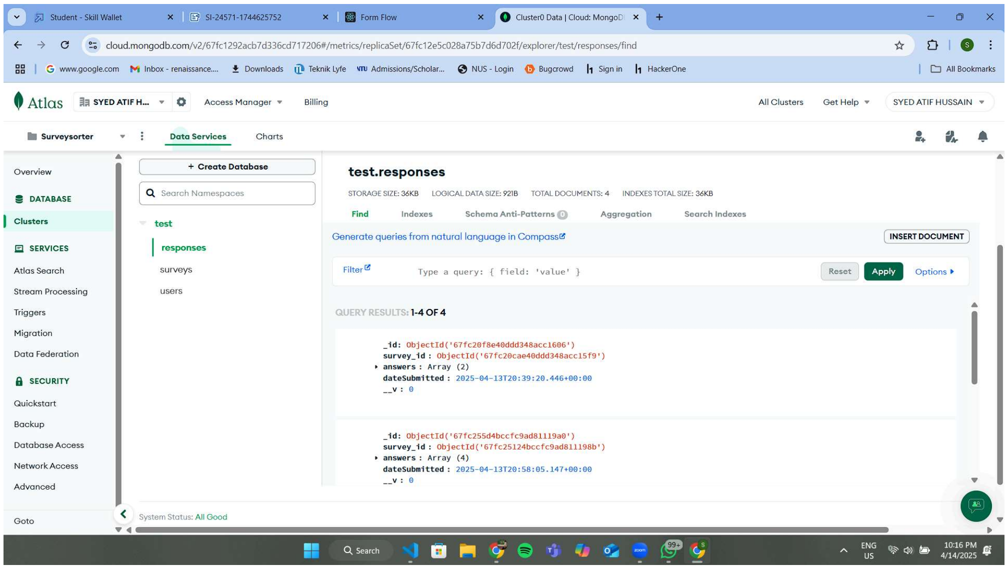
Filter: Type a query: { field: 'value' }

QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('67fd3b007db2021418ba3c1d')
title : "Test Survey"
author : ObjectId('67fc23f64bccfc9ad8111988')
description : "A test survey created to demonstrate working"
makePublic : true
introduction : "This is a survey for the Fullstack development course"
completionMessage : "Survey Completed"
reviewLink : ""
▸ questions : Array (3)
▸ responses : Array (1)
dateSubmitted : 2025-04-14T16:42:40.412+00:00
__v : 1
```

System Status: All Good

10:16 PM 4/14/2025



cloud.mongodb.com/v2/67fc1292acb7d336cd717206#/metrics/replicaSet/67fc12e5c028a75b7d6d702f/explorer/test/responses/find

**test.responses**

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 921B TOTAL DOCUMENTS: 4 INDEXES TOTAL SIZE: 36KB

Generate queries from natural language in Compass

Filter: Type a query: { field: 'value' }

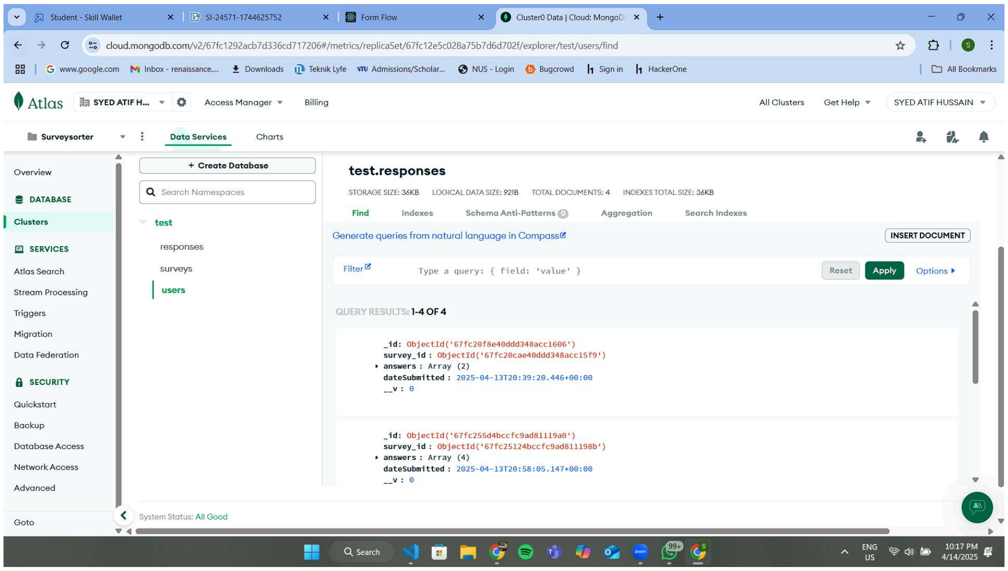
QUERY RESULTS: 1-4 OF 4

```
_id: ObjectId('67fc20f8e40ddd348acc1606')
survey_id : ObjectId('67fc20cae40ddd348acc15f9')
▸ answers : Array (2)
dateSubmitted : 2025-04-13T20:39:20.446+00:00
__v : 0
```

```
_id: ObjectId('67fc255d4bccfc9ad81119a0')
survey_id : ObjectId('67fc25124bccfc9ad811198b')
▸ answers : Array (4)
dateSubmitted : 2025-04-13T20:58:05.147+00:00
__v : 0
```

System Status: All Good

10:16 PM 4/14/2025



test.responses

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 921B TOTAL DOCUMENTS: 4 INDEXES TOTAL SIZE: 36KB

QUERY RESULTS: 1-4 OF 4

```
  _id: ObjectId('67fc20f8e40ddd348acc1606')
  survey_id : ObjectId('67fc20cae40ddd348acc15f9')
▸ answers : Array (2)
  dateSubmitted : 2025-04-13T20:39:20.446+00:00
  __v : 0
```

```
  _id: ObjectId('67fc255d4bccfc9ad81119a0')
  survey_id : ObjectId('67fc25124bccfc9ad811198b')
▸ answers : Array (4)
  dateSubmitted : 2025-04-13T20:58:05.147+00:00
  __v : 0
```