

**COMPARATIVE ANALYSIS OF API INTEGRATION**  
**USING CODE VS NO-CODE/LOW-CODE**  
**PLATFORMS**

Project report submitted in partial fulfillment of the  
requirement for the degree of Bachelor of Technology in

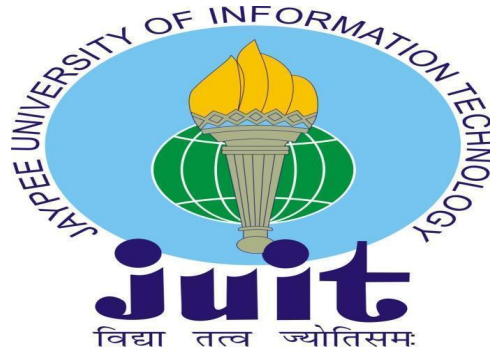
**Computer Science and Engineering**

By

Navya Yadav (191350)

Under the supervision of

Dr. Anupriya Kaur, Dr. Rakesh Kanji to



Department of Computer Science & Engineering and  
Information Technology

**Jaypee University of Information Technology Waknaghat,  
Solan-173234, Himachal Pradesh**

## DECLARATION

I hereby declare that the work presented in this report entitled “**COMPARATIVE ANALYSIS OF API INTEGRATION USING CODE VS NO-CODE/LOW-CODE PLATFORMS**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my work carried out over a period from August 2015 to December 2015 under the supervision of **Dr. Anupriya Kaur** ( post, HSS).

I also authenticate that I have carried out the above-mentioned project work under the proficiency stream **Data Science**. The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Navya Yadav, 191350

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Anupriya Kaur

Associate Professor

HSS Department

Dated: 10/5/2023

Dr. Rakesh Kanji

Assistant Professor (SG)

CSE Department

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT**  
**PLAGIARISM VERIFICATION REPORT**

Date: .....

Type of Document (Tick):  PhD Thesis  M.Tech Dissertation/ Report  B.Tech Project Report  Paper

Name: \_\_\_\_\_ Department: \_\_\_\_\_ Enrolment No \_\_\_\_\_

Contact No. \_\_\_\_\_ E-mail. \_\_\_\_\_

Name of the Supervisor: \_\_\_\_\_

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): \_\_\_\_\_

\_\_\_\_\_

**UNDERTAKING**

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

**Complete Thesis/Report Pages Detail:**

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

**FOR DEPARTMENT USE**

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

**FOR LRC USE**

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> <li>• All Preliminary Pages</li> <li>• Bibliography/Images/Quotes</li> <li>• 14 Words String</li> </ul>		Word Counts	
<b>Report Generated on</b>			Character Counts	
		<b>Submission ID</b>	Total Pages Scanned	
			File Size	

Checked by  
Name & Signature

Librarian

.....

**Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at [plagcheck.juit@gmail.com](mailto:plagcheck.juit@gmail.com)**

## **ACKNOWLEDGMENT**

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing that makes it possible to complete the project work successfully.

I am grateful and wish my profound indebtedness to Supervisor **DR. ANUPRIYA KAUR**, ASSOCIATE PROFESSOR, Department of Humanities and Social Science,s and co-supervisor **DR. RAKESH KANJI**, ASSISTANT PROFESSOR (Senior Grade), Department of CSE, Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisors in the field of “Research Area” to carry out this project. Their endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, established criticism, valuable advice, and reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

**NAVYA YADAV (191350)**

Department of Computer Science & Engineering

Jaypee University of Information Technology

## TABLE OF CONTENT

<b>S.No</b>	<b>Title</b>	<b>Page No.</b>
1	Abstract	VI
2	Chapter-1 [Introduction]	1-20
3	Chapter-2 [Literature Survey]	21-26
4	Chapter-3 [System Developement ]	27-33
5	Chapter-4 [Performance Analysis ]	34-49
6	Chapter-5 [Conclusion ]	50
7	References	51

## LIST OF FIGURES

<b>Figure No.</b>	<b>Description</b>	<b>Page No.</b>
1	Axios report	6
2	Importing ReactDOM and react	7
3	using hooks	9
4	using css	11
5	Mongoose schema	13
6	Using cors and connecting api s	16
7	App.js	28
8	Using axios	33
9	Interface of dynamic form	34
10	Structuring of project	36
11	Database record display with pagination	37
12	Data stored in MongoDB	38
13	Frontend with Appsmith	40
14	Api creation on Appsmith	41
15	Displaying data on appsmith	42
16	Pipelines in hop	43
17	Webservices in hop	44
18	Database connected via Apache hop	46
19	Connecting api with frontend	48
20	Example of how hop transforms are made	49

## **ABSTRACT**

This comparison investigates the distinctions between traditional coding methodologies and no-code/low-code platforms for API integration. The study looks at the advantages and disadvantages of each technique, such as development speed, simplicity of use, scalability, and adaptability. The findings indicate that no-code/low-code platforms provide considerable benefits in terms of speedier development timelines and ease of use, particularly for non-technical users. Traditional coding methods, on the other hand, are more adaptable and can manage more complex integrations. Finally, whether to use code or no-code/low-code platforms will be determined by the project's specific demands and the development team's skill set.

APIs (Application Programming Interfaces) are essential in modern software development. APIs enable separate programmes to connect with one another, allowing developers to build complex systems with a wide range of functionality. APIs have traditionally been connected into applications through coding approaches, which require developers to write bespoke code to connect the various platforms. The rise of no-code/low-code platforms, on the other hand, has created an alternate approach to API integration, allowing non-technical users to link disparate systems without writing any code.

This comparison investigates the distinctions between traditional coding methodologies and no-code/low-code platforms for API integration. The study looks at the advantages and disadvantages of each technique, such as development speed, simplicity of use, scalability, and adaptability.

# **Chapter 1 - INTRODUCTION**

## **1.1 COMPARATIVE ANALYSIS OF API INTEGRATION USING CODE VS NO-CODE/LOW-CODE PLATFORMS**

Integration of APIs is crucial to the creation of contemporary software. It enables the interaction of many programmes, giving consumers access to a wide range of functions. Custom software built by experienced software developers has traditionally been used to integrate APIs. Thanks to the growth of no-code/low-code platforms, non-technical people can now integrate APIs without writing code. The differences between conventional coding practises and no-code/low-code platforms for API integration are examined in this comparison. The study examines the benefits and drawbacks of each technique, including its rate of development, ease of use, scalability, and flexibility.

### Background

API integration is the process of linking various systems and apps to enable data and functionality exchange. An array of programming interfaces that specify how various systems can connect with one another are used to achieve this. API integration has traditionally been carried out utilising specialised software developers' proprietary code. This method necessitates the writing of code by developers to link many systems, which can be time-consuming and complicated.

However, with the emergence of no-code/low-code platforms, non-technical users can now integrate APIs without writing any code. These platforms provide pre-built connectors and drag-and-drop interfaces that make it easy for



non-technical users to connect different systems. This approach is ideal for simple integrations, where speed and ease of use are more important than flexibility and customization.

API integration is the process of integrating disparate systems and applications so they may exchange functionality and communicate. This is accomplished via a set of programming interfaces that define how various systems can communicate with one another. APIs are typically integrated via bespoke software created by skilled software engineers. In order to connect many systems in this manner, developers must build code, which can be time-consuming and need a high level of technical expertise.

Non-technical users can now integrate APIs without writing code, thanks to the rise of no-code/low-code platforms. These platforms feature pre-built connectors and drag-and-drop interfaces that make connecting disparate systems simple for non-technical users.

## **1.2 Problem Statement**

API integration is an important part of modern software development. API integration has traditionally been performed through the use of custom code written by knowledgeable software engineers. Non-technical users can now integrate APIs without writing code, thanks to the rise of no-code/low-code platforms. The issue is that it is not clear which strategy is preferable for API integration. While traditional coding methods provide greater flexibility and scalability, they are time-consuming and need highly competent workers. No-code/low-code platforms provide faster development timeframes and ease of use, but they may not be appropriate for complex integrations or users that need a high level of control over the integration process. As a result, a comparison

analysis is required to establish the advantages and disadvantages of each technique for API integration.

### **1.3 Objectives**

The goal of this comparative study is to:

1. Assess the advantages and disadvantages of traditional coding methods and no-code/low-code platforms for API integration.
2. Compare the development speed and simplicity of use of each strategy.
3. Evaluate each approach's scalability and adaptability.
4. Compare and contrast case studies of API integrations using standard coding approaches and no-code/low-code platforms.
5. Make recommendations for when each strategy is most appropriate for API integration based on the project's specific needs and the development team's skill sets.
6. Add to the current body of knowledge on API integration by providing insights for future research.

### **1.4 Methodology**

The goal of a comparative analysis of API integration using code versus no-code/low-code platforms is to weigh the advantages and disadvantages of each option in order to determine which is the best fit for a certain use case. The study should provide insights into the advantages and disadvantages of each alternative,

allowing decision-makers to make an informed decision based on the specific needs of the organization.

The research can also assist firms in determining the most efficient and cost-effective API integration strategy, taking into account criteria such as simplicity of use, flexibility, time to market, scalability, and maintenance. Businesses can assess which solution will produce the intended benefits while minimizing costs, saving development time, and boosting overall efficiency by evaluating these aspects.

### **Traditional coding:**

Web development is made up of various components that work together to form a functional website. Here are some of the most important aspects of web development covered in depth:

**Client-Side Scripting:** The code that runs on the user's web browser to give dynamic functionality on the website. Client-side scripting is commonly written in languages like as JavaScript, which may interact with the web page's Document Object Model (DOM) to alter components, respond to user events, and connect with server-side scripts.

**Server-Side Scripting:** This is the code that runs on the web server in order to generate dynamic content for the website. Server-side scripting is often written in languages such as PHP, Python, or Ruby, which communicate with the web server to access databases, process user inputs, and create HTML/CSS/JavaScript for the user's browser.

Markup languages are used to specify the structure and content of online pages. While CSS (Cascading Style Sheets) controls the presentation and style of web pages, HTML (Hypertext Markup Language) is the primary markup language used in web development. XML (Extensible Markup Language) is another markup language used for data exchange between systems.

**Web servers:** A web server is a computer-based software programme that receives HTTP requests from web browsers and processes the requests before returning HTML pages or other content to the client making the request. Popular web servers include Microsoft IIS, Nginx, and Apache.

**Databases:** Websites frequently require the long-term storage of data such as user information, product specifications, and transaction records. Databases provide for the effective storage, organisation, and retrieval of this data. MySQL, PostgreSQL, and MongoDB are popular web development databases.

**APIs (Application Programming Interfaces)** are a collection of protocols and technologies used in the development of software applications. APIs can be used in web development to provide communication between multiple systems, such as retrieving data from a third-party service or integrating a payment gateway.

**CMSs** are software systems that allow users to create, manage, and publish digital material such as web pages, blog entries, and photographs without requiring technical web development experience. WordPress, Drupal, and Joomla are examples of popular CMSs.

Here's an overview of the main two components of web development via coding:

## **React**

React is a free and open-source JavaScript library for creating user interfaces. It was created by Facebook and made available to the public in 2013. React is intended to make it simple to generate reusable components that may be used to build complicated user interfaces.

React works by creating the Virtual DOM, a virtual representation of the user interface. The virtual DOM (Document Object Model), which is used to represent the structure of a web page, is a lightweight version of the real DOM (Document Object Model). React updates the actual DOM only when it is necessary, keeping track of changes to the user interface in the Virtual DOM. This technique improves productivity and makes it simpler to develop complicated user interfaces.

React components are reusable bits of code that represent certain user interface elements. To design more complicated user interfaces, components can be nested inside other components. React components are written in JavaScript and can be used in conjunction with other libraries and frameworks to construct full-featured web applications.

React's declarative programming language is one of its primary advantages. React components are declarative in nature, which means they specify what should be presented on the screen rather than how it should be displayed. This makes it easier to reason about the user interface's behaviour and helps to decrease problems and errors.

```
1 import React, { useState } from 'react';
2 import axios from 'axios';
3 import './App.css'
4 // // const ulStyle = { width: '100%', height: '100%', listStyleType: 'none' }
5 const URL = 'http://localhost:3000/users'
6
```

**Figure 1. Axios import**

## **ReactDOM**

The ReactDOM package offers an API for working with the Document Object Model (DOM). It's employed to render React components to the DOM of the browser.

Your components can be specified in React as JavaScript classes or functions. There are several ways to introduce these components into the browser using ReactDOM. For instance, a React component is rendered to a DOM node in an HTML document by the ReactDOM.render() method.

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import App from './App';
4  import DataTable from './DataTable';
5  ReactDOM.render(
6    <React.StrictMode>
7      <App />
8      <DataTable/>
9    </React.StrictMode>,
10   document.getElementById('root')
11 );
12
```

**Figure 2. Importing ReactDOM and react**

## **Express JS**

Express.js is a quick, impartial, and uncomplicated Node.js web framework. It offers a set of effective tools for creating web apps and APIs while giving developers the freedom to select their own organisational structure and design.

The following characteristics of Express.js are fundamental:

Express.js provides a straightforward and flexible approach to create routes for handling HTTP requests. As a result, linking URLs to certain controller functions

and handling different request types like GET, POST, PUT, and DELETE is made simple for developers.

**Middleware:** Middleware from Express.js can be used to increase an application's request-response cycle's functionality. The request or response objects can be changed, as well as additional logic and validation, using middleware functions.

A few of the template engines that Express.js supports are Pug, Handlebars, and EJS. Using data from the programme to display HTML templates, developers can build dynamic websites.

An Express.js middleware function makes it possible to handle failures in the application. Use this middleware feature to spot application errors and send out the right error messages.

**Serving static files from a public directory:** Express.js enables developers to serve static files including images, stylesheets, and JavaScript scripts. Therefore, serving client-side assets to the browser is simple.

**Database integration:** Using a variety of Node.js database drivers and ORMs, Express.js may be swiftly integrated with databases like MongoDB, MySQL, and PostgreSQL.

## **Hooks in React JS**

### **UseEffect() hook**

React supports the addition of side effects to functional components using the `useEffect` hook. Any actions that take place in addition to or instead of a component's normal operation are referred to as side effects. Examples include changing the title of the document or getting information from an API. The lifecycle methods `componentDidMount`, `componentDidUpdate`, and `componentWillUnmount` used by class components have been replaced with `useEffect`.

A callback function is the first argument to `useEffect`, and it will be run after each render. The second input, which is an optional array of dependents, controls how the effect is applied. The effect will be implemented if any of the dependents have changed since the last render. The effect won't be felt if the dependencies haven't changed.

```
function App() {  
  const [formFields, setFormFields] = useState({ "name": '', "entries": '', "address": '' },)  
  const [openArray, setopenArray] = useState([false])  
}
```

**Figure 3. using hooks**

### **useState() hook**

React's `useState` hook can be used to add stateful logic to functional components. The information in a component that changes over time is referred to as its "state". Using the `useState` hook, state variables can be created and modified in response to user input or other events.

After receiving an initial state value as an argument, `useState` returns an array with the current state value and a method to update the state. As the state variable, any type of data, including strings, numbers, arrays, and objects, may be utilised.

### **CORS**



CORS stands for Cross-Origin Resource Sharing. Web browsers can use this technique to send requests to domains other than the one that first served the web page.

Simply said, a web page published from one domain (origin) will often be denied access to a resource on a different domain by the browser due to security issues. You can tell the browser to accept requests from other domains by utilising CORS.

CORS employs HTTP headers to determine the domains and request types that are allowed when sending queries to a server.

## **CSS**

A styling language called Cascading Style Sheets (CSS) is used to add design, layout, and presentation to online pages. The ability to isolate a web page's presentation from its content using CSS makes it simpler to update a site's layout and design without impacting the underlying HTML code.

Certain HTML elements are given styles via CSS to operate. Styles can be seen in things like colour, typography, size, spacing, and other aesthetic elements. CSS may be used to build both straightforward and complex designs, including responsive layouts that adjust to different screen sizes and devices.

You can alter the font, colour, size, and spacing of the text on a web page with CSS. Text may become easier to read and appear better as a result.

Layouts and the placement of items on a web page can be designed using CSS. This includes setting up grids, aligning elements, and adding padding and margins.

Responsive designs, which can adjust to fit different screen sizes and devices, may be created using CSS. Building websites that function well on mobile devices requires this.

Transitions and animations: CSS can be used to add animations and transitions to web page elements. As a result, a website might become more engaging and dynamic.

```
app-frontend > src > # App.css > input
1  .App {
2    display: flex;
3    position: relative;
4    flex-direction: column;
5    justify-content: center;
6    align-items: center;
7    margin-top: 100px;
8    width: 100%;
9  }
10
11 .title{
12   position: absolute;
13   text-align: left;
14   overflow: hidden;
15   top: -100px;
16   width: 100%;
17   padding-left: 10px;
18   padding-top: 15px;
19   padding-bottom: 15px;
20   background-color: grey;
21   font-family: 'Didot ', serif;
22 }
23
24 input{
25   padding: 10px;

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

```
PS C:\Users\nyadav\Desktop\work\react project\app-frontend>
* History restored

PS C:\Users\nyadav\Desktop\work\react project>
* History restored

PS C:\Users\nyadav\Desktop\work\react project>
```

**Figure 4. Using css**

## **Schema of mongoDB**

In MongoDB, a schema explains how a database's data is organised or structured. MongoDB, a document-based NoSQL database, does not enforce a rigid structure like a traditional relational database does. Comparatively, MongoDB uses a flexible schema that permits the capture of dynamic and unstructured data.

A MongoDB schema, which is specified at the collection level, is made up of fields and the data types that go with them. Fields, which are similar to columns in a traditional relational database, represent a document's attributes. Unlike fields in traditional databases, fields in a MongoDB schema can be nested or multi-valued.

The schema in MongoDB specifies how the documents in a collection should be organised. A schema must be designed in order to guarantee consistency and validate data being added to the database. By creating a schema, developers may compel data integrity, stop data loss or corruption, and facilitate data searching.

MongoDB provides a tool called "Schema Validation" to enforce schema standards and ensure data integrity. Schema validation allows developers to specify validation constraints and rules, such as required fields, data types, and value lengths, at the collection level. When a document is added or modified, the validation criteria are used, and any violations will result in errors.

Using the Mongoose library in Node.js, you can build MongoDB schemas. A well-liked object data modelling (ODM) library called Mongoose offers a quick and effective approach to communicate with MongoDB. Here are the steps to use Mongoose to construct a schema in MongoDB:

Install Mongoose: In the directory of your Node.js project, enter the following command to install Mongoose:

Npm install mongoose copy the code

imports of monkeys The following code must be used to import Mongoose into your Node.js project:

javascript

Write the following code: `const mongoose = require("mongoose");`

Link to MongoDB: You must use Mongoose to connect to your MongoDB database before generating a schema. Utilise the following code to accomplish this:

```
1  const mongoose =require("mongoose")
2  mongoose.set('strictQuery',false);
3
4  var users = new mongoose.Schema({
5    "name":{
6      type: String,
7      required: true
8    },
9    "entries":{
10     type : String
11   },
12   "address":{
13     type : String,
14     required:true
15   }
16 }, {strict: false});
17 const userModel=mongoose.model("user",users)
18 module.exports = userModel
19
20
```

Figure 5. Mongoose schema

## **Node JS**

Node.js is a cross-platform, open-source back-end JavaScript runtime environment. It enables developers to execute JavaScript on the server side, away from the web browser environment. Node.js debuted in 2009 and has since grown to become one of the most popular technologies for developing scalable web applications.

Node.js is built on the Google Chrome V8 JavaScript engine, which offers a quick and efficient runtime environment. It employs an event-driven, non-blocking I/O strategy, making it excellent for developing high-performance, real-time, data-intensive applications.

Node.js includes a large number of built-in modules that provide a variety of capabilities, such as file system I/O, networking, and HTTP/HTTPS servers. It also features a robust ecosystem of third-party modules, which may be readily installed with the npm (Node Package Manager) tool.

One of Node.js' primary advantages is its capacity to handle enormous amounts of data and connections at the same time. Node.js applications are built to be very scalable, and they can easily accommodate thousands of concurrent connections. This makes it excellent for developing real-time web applications such as chat, gaming, and collaboration tools.

## **Mongoose**

The well-known Object-Document Mapping (ODM) Mongoose module for Node.js with MongoDB offers a more complicated abstraction than the built-in MongoDB driver. By enabling developers to specify schemas for their data and interact with the database naturally using JavaScript, it streamlines the process of working with MongoDB.

Some distinctive qualities of Mongoose include the following:

**Schemas:** Using a straightforward declarative syntax, Mongoose enables developers to design schemas for their data. Schemas offer a mechanism to enforce data validation and guarantee uniform data preservation.

**Querying:** Mongoose offers a querying API that enables programmers to ask questions of the database in a clear, concise manner.

**Models:** Using schemas, the Mongoose framework offers a means to build models from scratch. Models offer a high-level API for data management and querying and represent a collection of documents in the MongoDB database.

**Validation:** Using a built-in validation mechanism offered by Mongoose, developers can provide validation criteria for their data. This ensures that the data is saved in the correct format and complies with all specifications.

Programmers can create actions that can be executed either before or after specific actions, such saving or deleting a document, with the use of middleware like Mongoose. This makes it possible to provide an application certain privileges and edit data before it is recorded in the database.

```

1  const express = require('express')
2  const app=express()
3  const bodyParser = require('body-parser')
4
5  const cors=require('cors');
6  const PORT = process.env.port||3000;
7  app.use(cors({credentials: true, origin: "http://localhost:3001"}));
8
9  app.use(bodyParser.json())
10
11  const db = require('./models')
12
13
14  function success(res,data){
15  |   return res.status(200).json(data)
16  | }
17
18  app.post("/users", async(req,res)=>{
19  |   try{
20  |     console.log(req.body)
21  |     const user = await db.users.create(req.body)
22  |     return success(res,user)
23  |   }catch(err){
24  |     console.log(err);
25  |   }

```

**Figure 6. Using cors and connecting api s**

## No-code/Low-code platforms

### Apache Hop

Apache Hop is a free and open-source data integration tool with a graphical user interface (GUI) for creating and running ETL (Extract, Transform, Load) workflows. It was formerly known as Apache Beam, and it is a data processing framework that can be used with a variety of computer languages including Java, Python, and Go. To avoid conflict with another Apache project, Apache Beam, the project's name was changed to Apache Hop in 2021.

Apache Hop offers a drag-and-drop interface for building ETL pipelines, making it easier for non-technical users to develop data integration procedures. It is compatible with a wide range of data sources and objectives, including relational databases, flat files, NoSQL databases, cloud storage, and others. It also contains

a number of transformation and validation steps that may be used to modify and validate data as it passes through the pipeline.

One of Apache Hop's distinguishing features is its metadata-driven approach to data integration. The tool maintains a centralised metadata repository that includes information about the many data sources and targets used in the ETL pipeline, as well as information about the data transformations and validation criteria. This metadata-driven method adds a layer of abstraction to the process of building and managing ETL pipelines.

Apache Hop also contains a number of features aimed at increasing the efficiency and dependability of ETL procedures. It, for example, supports parallel processing, which allows enormous amounts of data to be processed more quickly. Error management and retry mechanisms are also included to help ensure that ETL procedures continue to run even when errors occur.

Overall, Apache Hop is a robust and adaptable data integration tool for building sophisticated ETL operations. Its metadata-driven methodology, drag-and-drop GUI, and support for a diverse set of data sources and targets make it a compelling choice for data integration projects of all sizes.

## **Appsmith**

Appsmith is a free and open-source platform for creating internal tools and applications without the need for coding. It has a drag-and-drop interface for creating user interfaces and workflows, as well as a visual editor for creating and managing database queries and API connections.



Users can connect to a variety of data sources, including databases, APIs, and third-party services, using Appsmith. By connecting these data sources and using them to trigger actions and updates within the programme, users can develop custom workflows and automation activities.

Appsmith's ability to build real-time data visualisations and dashboards based on data from connected sources is one of its primary advantages. Users can design their own charts, graphs, and tables to display data in an easy-to-understand and analyse format.

Appsmith also provides a collaboration and deployment platform that enables several people to work on the same application at the same time. Version control, role-based access control, and a range of other capabilities make it simple to manage and distribute applications across teams.

Appsmith is designed to be extremely extendable, with custom widgets and plugins that can be added to the platform to improve its usefulness. The platform is built on a strong API and is readily linked with other tools and services.

Overall, Appsmith is a strong and adaptable framework for developing internal tools and applications without the need for coding. Its drag-and-drop interface, support for many data sources, and real-time data visualisation features make it an appealing alternative for enterprises and organisations trying to optimise workflows and automate operations.

## **Ngrok**

Developers can make their local development environment accessible via the internet by utilising the programme Ngrok to build secure tunnels to localhost. To

test and share their work with others in real-time, it offers a public URL that may be used to access a local web server running on a developer's computer.

The following are some of the main attributes and advantages of Ngrok:

**Secure tunnels:** To ensure that data is delivered surreptitiously and securely, Ngrok builds secure tunnels between a local web server and the public internet.

**Public URLs:** Ngrok offers a public URL that may be used to access a regional web server from any location in the world, making it simple to test and share work with colleagues and clients.

**Support for HTTP and HTTPS:** Ngrok enables developers to test and debug secure online applications by supporting both HTTP and HTTPS protocols.

straightforward command-line interface makes it straightforward to set up Ngrok, which makes it a perfect solution for developers who want to concentrate on creating applications rather than managing complicated infrastructure.

**Testing mobile apps:** By offering a public URL that can be used to visit a local web server running on a developer's machine, Ngrok may be used to test and debug mobile applications.

Ngrok offers a free plan with basic features as well as paid options with more features and higher usage limitations, all with flexible pricing.

## **1.5 Organisation**

The document contains a total of five chapters.

**Chapter-1** provides a brief introduction about COMPARATIVE ANALYSIS OF API INTEGRATION USING CODE VS NO-CODE/LOW-CODE PLATFORMS and various models utilised in the whole project. It also describes the problem statement and objectives of the project.

**Chapter-2** addresses the literature review for the proposed work, that is, results of the related work along with the limitations of the existing solutions.

**Chapter-3** describes the system development part that explains the detailed methodology for the proposed work.

**Chapter-4** analyses the performance of the entire project line that includes the evaluation and results at different stages.

**Chapter-5** discusses the conclusions and future scope for the proposed work.

## **Chapter-2 LITERATURE SURVEY**

### **1. [1] Low-Code Platform**

**Authors:** Alexander C. Bock & Ulrich Frank ,Business & Information Systems Engineering

**Journal:** *Business & Information Systems Engineering*

#### **Methodology:**

Alexander C. Bock and Ulrich Frank's research article "A Review of Low-Code Platforms" presents an informative overview of low-code development platforms, their properties, and how they are employed in the software development industry.

The authors start by defining low-code development and describing how it differs from traditional software development. They also provide a historical context for low-code systems, emphasising their evolution over time.

Following that, the study examines the features of low-code platforms, such as visual modelling tools, pre-built components, and code generation capabilities. The authors also discuss the advantages and disadvantages of employing low-code platforms, such as enhanced productivity, faster time-to-market, and potential customisation and scalability limits.

Furthermore, the authors explore the low-code platform market landscape and present an outline of the industry's leading competitors. They also give a case study of a company that used a low-code platform to construct a commercial application and was successful.

Concisely ,the article presents a thorough examination of low-code platforms and their potential for speeding up software development. It provides useful information about the qualities, benefits, and challenges of using low-code platforms, as well as the current market scenario.

## 2. [2] Study of deployment of “low code no code” applications toward improving digitization of supply chain management

**Authors:** Som Sekhar Bhattacharyya, Saurabh Kumar

**Journal:** Journal of Science and Technology Policy Management

### **Methodology:**

Som Sekhar Bhattacharyya and Saurabh Kumar's research paper "Study of deployment of "low code no code" applications towards improving digitization of supply chain management" is an intriguing study that investigates the potential of low-code/no-code (LCNC) platforms for improving supply chain management digitization.

The authors begin by outlining the shortcomings of traditional supply chain management methods as well as the potential benefits of digitization. They then present the notion of LCNC platforms and demonstrate how they may be used to create custom apps without substantial coding skills or expertise.

The report describes a case study of a corporation that used an LCNC platform to digitise its supply chain management operations. The authors discuss the platform's capabilities, such as visual drag-and-drop interfaces and pre-built components, which enabled the organisation to swiftly design and launch unique applications tailored to their specific requirements.

According to the findings, the LCNC platform considerably improved the efficiency of supply chain management procedures by lowering the time and effort necessary for data entry and analysis. The authors also emphasise the potential of LCNC systems to reduce the need for manual involvement in supply chain management, improve data accuracy, and reduce errors.

### 3. [3] **Characteristics and Challenges of Low-Code Development: The Practitioners' Perspective**

**Authors:** Som Sekhar Bhattacharyya, Saurabh Kumar

**Journal:** Journal of Science and Technology Policy Management

#### **Methodology:**

The above mentioned research paper by Yajing Luo and Peng Liang is a fascinating study that examines the perspectives of practitioners with experience with low-code development.

The authors begin by providing an introduction of the characteristics of low-code development, such as visual modelling tools, pre-built components, and code generation capabilities. They also discuss the potential benefits of using low-code platforms, such as faster development times and increased productivity.

The study presents the findings of a survey of low-code development practitioners. The authors highlight the challenges that practitioners face when adopting low-code platforms, such as customization limits, limited support for large applications, and worries about vendor lock-in.

The paper also digs into the factors that influence low-code platform adoption, such as ease of use, support and training availability, and the ability to integrate with existing systems..

#### 4. [4] **Analysis of Low Code-No Code Development Platforms in comparison with Traditional Development Methodologies**

**Authors:** Shreyas Shridhar, Siddharth Bose

**Journal:** International Journal for Research in Applied Science & Engineering Technology (IJRASET)

##### **Methodology:**

The research paper "Analysis of Low Code-No Code Development Platforms in Comparison with Traditional Development Methodologies" by Shreyas Shridhar and Siddharth Bose is a detailed study that compares low-code/no-code (LCNC) development platforms with traditional development methodologies.

The authors begin by describing the shortcomings of traditional development methodologies, such as lengthy development times, exorbitant expenses, and a scarcity of experienced developers. They then talk about LCNC development platforms and how they give a faster, more cost-effective alternative to traditional development techniques.

The study investigates the features and advantages of LCNC development platforms, including as visual drag-and-drop interfaces, pre-built components, and code generation capabilities. The authors also examine the disadvantages of LCNC platforms, such as customisation and scalability limitations.

The research compares LCNC systems to traditional development methodologies on a variety of variables, including development time, cost, ease of use, and scalability. While LCNC systems have significant advantages over traditional development approaches in terms of speed, price, and ease of use, the authors believe that they may have limitations in terms of scalability and customization.

## 5. [5] **Low-Code Is Often High-Code, So We Must Design Low-Code Platforms to Enable Proper Software Engineering**

**Authors:** Timothy C. Lethbridge

**Journal:** Part of the [Lecture Notes in Computer Science](#) book series (LNTCS,volume 13036)

### **Methodology:**

The study "Low-Code Is Often High-Code, So We Must Design Low-Code Platforms to Enable Proper Software Engineering" by Timothy C. Lethbridge is an insightful study that highlights the challenges associated with low-code development and emphasises the importance of proper software engineering practises in low-code platforms.

The paper begins by emphasising the benefits of low-code development, such as faster development times and increased productivity. However, according to the author, many low-code systems lack fundamental software engineering capabilities and best practises, resulting in issues such as poor code quality, limited flexibility, and scalability.

In low-code development, the study underlines the importance of software engineering practises such as modular design, abstraction, and testing. The author offers a set of recommendations for developing low-code platforms that enable good software engineering, such as support for common programming languages, debugging and testing tools, and customization.

OutSystems and Mendix are two low-code systems that have successfully integrated software engineering practises. The author also investigates the possible benefits of incorporating AI and machine learning (ML) technology into low-code platforms in order to improve code quality and scalability.



## 6. [6] **The Rise of No/Low Code Software Development—No Experience Needed?**

**Authors:** Marcus Woo

**Journal:** Part of the [Lecture Notes in Computer Science](#) book series (LNTCS,volume 13036)

**Publisher:** IEEE

### **Methodology:**

Marcus Woo's article "The Rise of No/Low Code Software Development—No Experience Needed?" is well-written and comprehensive, providing an overview of the no/low-code software development trend and its possible impact on the software development industry.

The article begins by explaining no/low-code development and then goes on to discuss prominent platforms such as Appian and Mendix. It then goes into the advantages of no/low-code development, such as faster development times and more productivity. The author also mentions some of the disadvantages of no/low-code development, such as limited flexibility and scalability.

The paper underlines how non-technical people may use no/low-code development and how it has the potential to democratise the software development process. The author does, however, emphasise that even with no/low-code development, some technical competence is required, and understanding the underlying code and architecture is vital.

The essay investigates the potential consequences of no/low-code development on the software development industry, such as job position shifts and greater competitiveness. The author also discusses some of the potential challenges associated with no/low-code development, such as the need for strong governance and security.

## **Chapter-3 SYSTEM DEVELOPMENT**

Setting up a development environment:

Install Node.js and create a new React project using a tool like Create React App.

To execute any React application, we must first install NodeJS on our machine. As a result, the first step will be to get NodeJS and install it.

Step 1: First, install NodeJS. Go to the official NodeJS download page to download and install the most recent version of NodeJS before installing React. We'll need to install React Boilerplate after we've installed NodeJS on our machine.

Step 2: Set up the react environment for previous and subsequent versions, and then follow anyone who is compatible with your node version.

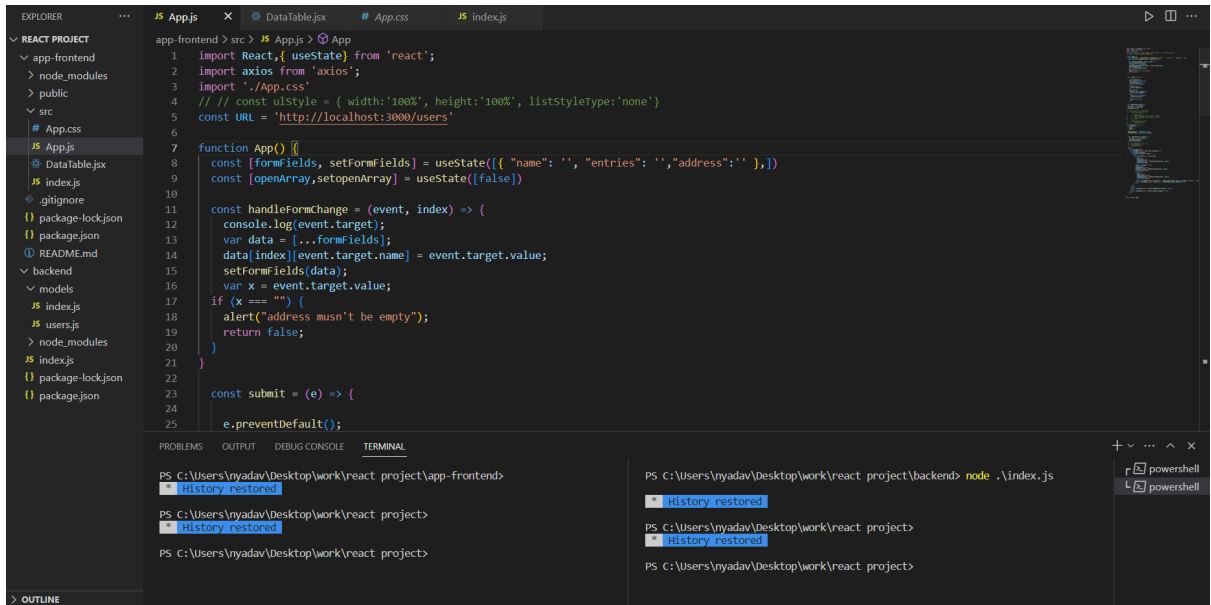
Step 3: Create a React app. We'll now create an app using the boilerplate we downloaded. The following command will create a new app called myapp.

The command statement will create a new directory inside your current directory called myapp that will contain all of the files required to launch a React app.

The directory described above has a large number of files. The beginning course will mostly focus on the index.html and index.js files. The index.html file will have a div element with the id "root" within which everything will be rendered, whilst the index.js file will contain all of our React code.

Step 4: Install and configure the development server in step four. To start the development server, navigate to "myapp" in your current directory and enter the following command:

The programming language is called JavaScript (javascript). When you run the above command in cmd successfully, your compiler will ask you to access localhost in your usual browser.



**Figure 7.App.js**

To reflect the changes you're making in your App, navigate to the URL provided in the browser. The page that appears when you enter the aforementioned URL into your default browser is as follows.

Setting up React Boilerplate on older versions of Node, such as 8.10 and 5.6. The boilerplate will be installed worldwide. To install the React js Boilerplate on your PC, type the following line into your terminal or command prompt (cmd).

Design your form: Determine the fields you want in your form and their types. You can create a form using HTML, CSS, and JavaScript in the React component.

Creating a form requires determining what types of fields you want to add, such as text inputs, radio buttons, checkboxes, dropdowns, and more. A form can be built using a variety of web technologies such as HTML, CSS, and JavaScript.

Here's how to make a form in a React component using these technologies, step by step:

Determine whatever fields you wish to include in your form: The first step in creating a form is determining which fields to include. The goal of the form and the information you need to collect from the user will determine this.

Create a server: In Node.js, create a server using a framework like Express.js. The server will handle the HTTP requests and responses between the client and the server.

Check to see if Node.js is installed on your development system. This tutorial makes use of Node.js 10.19.0. Follow the steps in [Installing Node.js and Creating a Local Development Environment on macOS](#) or [Installing Node.js on Windows](#). To do so, see the section [How to Install Node.js on Ubuntu 18.04](#).

Right out of the box, the Node.js framework simplifies the building of web servers. To begin, make sure you understand the fundamentals of Node.js. To begin, see [How to Write and Run Your First Node.js Programme](#).

If you're unfamiliar with Node.js asynchronous programming or the fs module for file communication, we use it in one of our components as well.

Set up a database: Choose a database to store the form data. You can use a SQL or NoSQL database like MySQL or MongoDB.

Let's start by installing MongoDB if you haven't previously. Install the MongoDB package appropriate for your operating system. Download the right msi/zip file for your operating system to install the application. (I won't go into depth about the installation procedure because it's simple.)

I'm assuming you've finished deploying the MongoDB database. To test your installation, launch CMD and enter `mongod --version`. Everything is fine if you get a result like the one in the image below.

Create an API: Using Express.js, create an API that will handle the form data from the client-side and store it in the database.

A web service is a collection of open protocols and standards that enable data to be shared between applications or systems. Web services are used by software programmes written in a variety of programming languages and running on a variety of platforms to exchange data via computer networks such as the Internet in a manner similar to inter-process communication on a single machine. This interoperability (communication between Java and Python or Windows and Linux apps) is enabled by the use of open standards.

RESTful web services are web services based on the REST Architecture. To implement the REST architecture concept, these web services use HTTP methods. A RESTful web service will frequently establish a URI, or Uniform Resource Identifier, that will give resource representations such as JSON and a set of HTTP-Methods.

Connect the API to the form: In the React component, use a package like Axios to make HTTP requests to the API and submit the form data.

REST APIs are used for obtaining and modifying data using a standard set of stateless activities. These operations are critical to the HTTP protocol and represent required create, read, update, and delete (CRUD) capabilities, but not in a neat one-to-one fashion:

POST (create a resource or provide data in general)

GET (retrieve an index of resources or a single resource)

PUT (generation or replacement of resources)

PATCH (resource modification/update)

DELETE (removing a resource)

Add dynamic form fields: To make the form dynamic, you can use React's state to add or remove form fields based on user input.

after creating a simple form in React. This method, however, is appropriate and effective only when we know how many input fields our programme requires. In other cases, where we don't know what will happen ahead of time, we must build our form in a new and more dynamic way.

To make the form dynamic, we want to express the form state in an array of objects rather than a single object that retains the form values. Each object will have information about the displayed input field, such as the label, type, and value. Then we can loop through the array for each object and render an input field.

Display the form data: Once the form data is submitted and stored in the database, you can retrieve and display it in a separate React component. This is done using axios.

## **Axios**

Axios is a popular JavaScript tool that lets you make HTTP requests from a web browser or a Node.js server. It offers a straightforward and user-friendly API for making RESTful API and other web service queries. Axios is a popular web development tool that is widely used to handle HTTP requests.

Axios has the following key characteristics:

Axios is built on JavaScript Promises, which give a simple and straightforward way to handle asynchronous operations. This means you can use `async/await` syntax to make HTTP requests and process responses, making your code more legible and manageable.

Simple to use: Axios offers a simple API for making HTTP requests. You can do GET, POST, PUT, DELETE, and other HTTP queries with just a few lines of code.

Axios is a cross-platform framework that can be used in both browser and Node.js contexts. This means you can develop code for your application's client and server sides, making it easier to maintain and scale.

Interceptors: Axios gives you the ability to intercept requests or responses before they are transmitted or received. This function is useful for adding headers, logging data, and changing request or response data.

Axios can translate request and response data between JSON, XML, and form data automatically. This simplifies working with APIs that need a specified data format.

In this example, we import the Axios library and use the axios function to make a GET request to the URL 'https://api.example.com/posts'. The response is then processed using the Promise-based syntax, and the response data is logged to the console. If an error happens, it is detected and recorded in the console.

Finally, Axios is a powerful and straightforward JavaScript library for making HTTP requests. It features Promise-based syntax, a simple API, cross-platform interoperability, interceptors, automatic transformations, and request cancellation. Axios can let you make HTTP queries and manage answers in modern web apps.

```
e.preventDefault();
axios.post(URL, formFields)
  .then((response) => {
    console.log(response);
  })
  .catch((error) => {
    console.log(error);
  })
axios.get(URL)
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  })
  .then(function () {
  });
}
```

**Figure 8. Using axios**



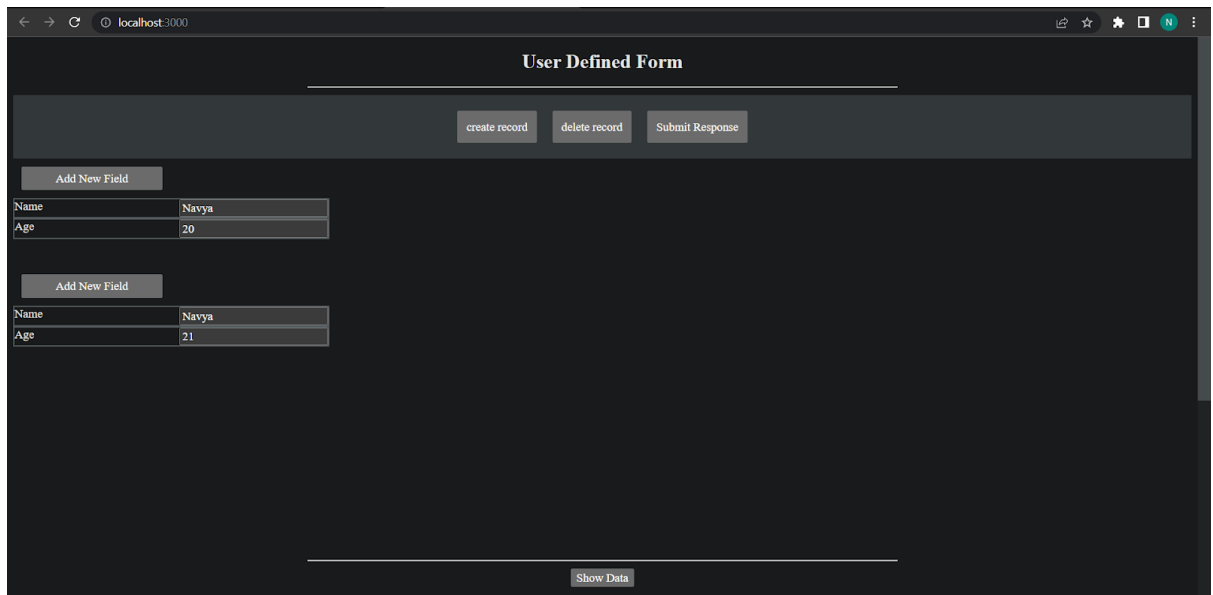
## Chapter-4 PERFORMANCE ANALYSIS

OUTPUT VIA TRADITIONAL CODING DEVELOPMENT:

Created with an aim to ask the user for name of input field along with entry to be also done by user this dynamic form has been created by the use of react and node js only.

This page contains entry for user with entry field name as given by user along with entry of value .

This is an overview of how the front page looks



**Figure 9. Interface of dynamic form**

Web development requires the ability to work with a wide range of files, including HTML, CSS, JavaScript, images, and more. Developers can more easily

locate and modify certain files by logically and effectively structuring these files with the aid of good file management.

**Collaboration:** Several developers usually work on the same project at once when creating a website. Everyone has access to the files they need, and because of effective file management, changes made by one developer are immediately visible to all other developers.

The act of tracking changes to files over time is known as version control, and file management is essential to this procedure. It is essential for making sure that the most recent version of the code is always used and for going back to prior versions when necessary.

**Security:** For security reasons, efficient file management is essential. This includes safeguarding confidential files, restricting access to certain directories or files, and frequently backing up important files.

**Performance:** How files are handled can have an impact on how well a website functions. This includes methods like cache optimisation, properly compressing images to minimise file size, and minifying JavaScript and CSS to speed up page loading.

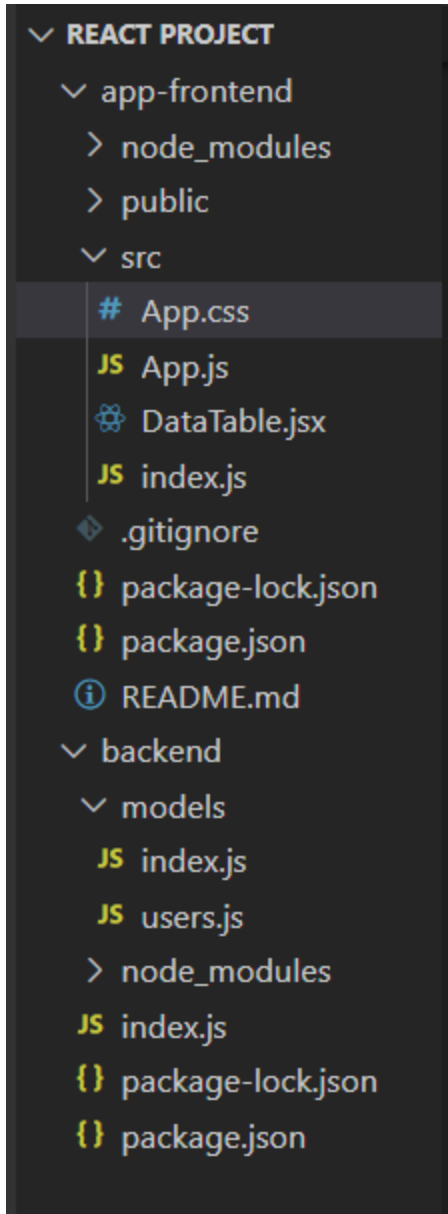
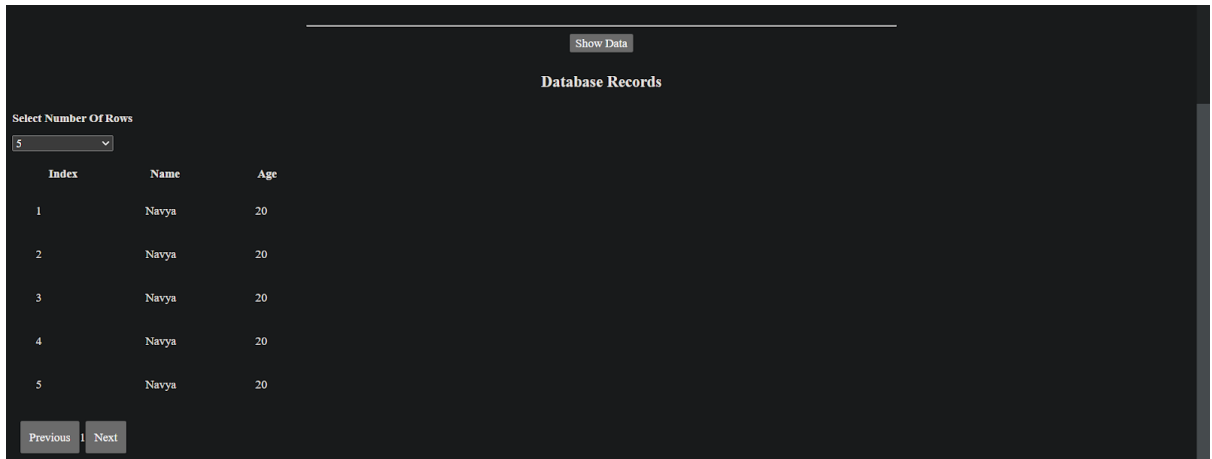


Figure 10. Structuring of project

This is how the part where the data is being displayed looks. Server side pagination has been done on the data and hence the data is displayed in a complete user friendly format.

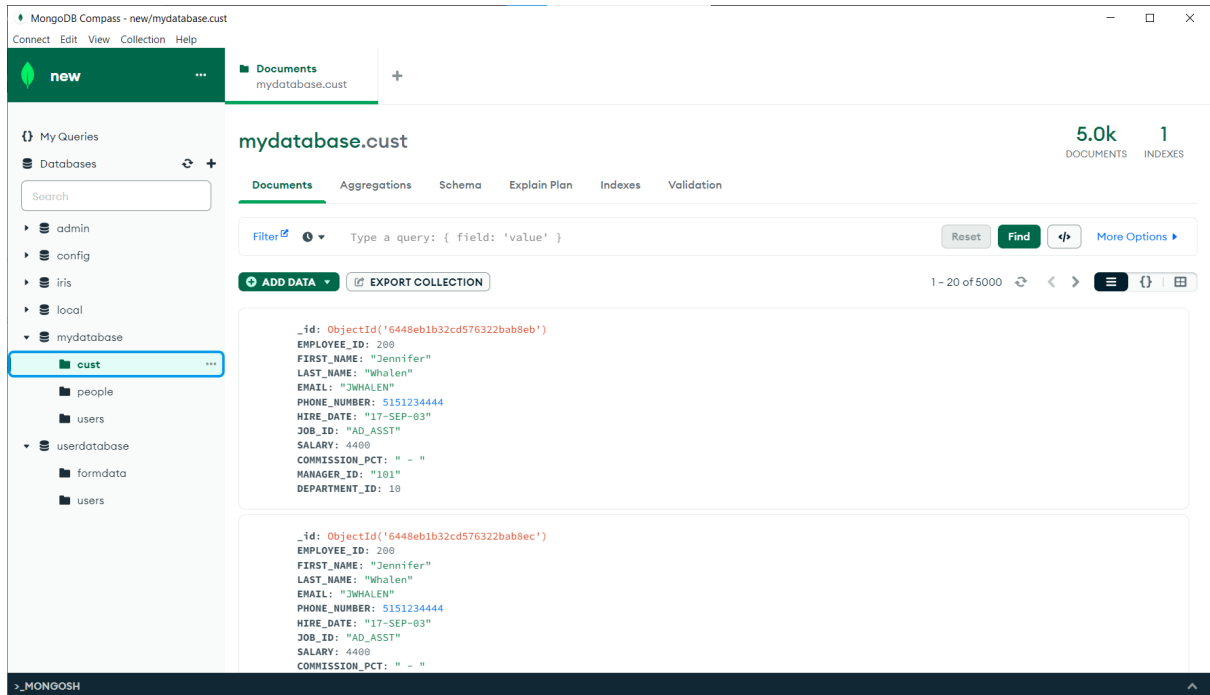


The screenshot shows a web interface for displaying database records. At the top right, there is a 'Show Data' button. Below it, the title 'Database Records' is centered. On the left, there is a 'Select Number Of Rows' dropdown menu currently set to '5'. The main content is a table with three columns: 'Index', 'Name', and 'Age'. The table contains five rows of data, each with an index from 1 to 5, the name 'Navya', and the age '20'. At the bottom left, there are 'Previous' and 'Next' buttons for navigation.

Index	Name	Age
1	Navya	20
2	Navya	20
3	Navya	20
4	Navya	20
5	Navya	20

**Figure 11 Database record display with pagination**

This is how the data saved in the mongoDB looks. All of the data is custom as field name and size of json object both vary according to users input:



**Figure 12. Data stored in MongoDB**

## OUTPUT VIA NO-CODE/LOW-CODE PLATFORMS:

Using pre-made user interface components known as Appsmith widgets, developers are able to quickly and effectively design complicated web apps without writing any code. Widgets are drag-and-drop elements that may be added to a canvas and set in a variety of ways to display or manipulate data.

The following are some of the main characteristics of Appsmith widgets:

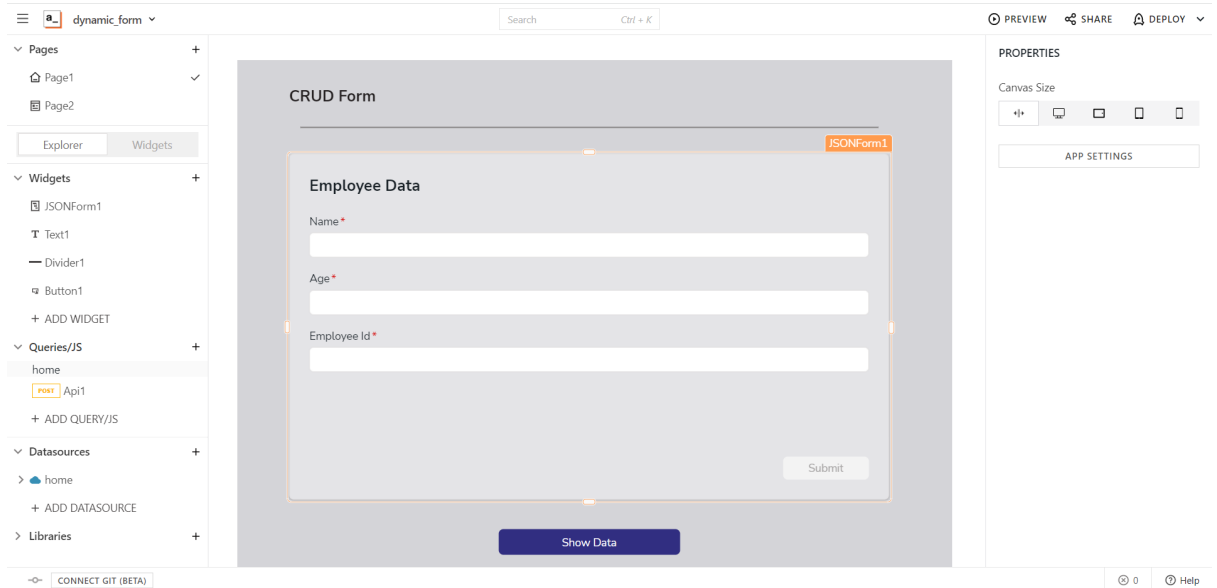
Pre-built functionality: With the help of Appsmith widgets, programmers can easily and rapidly add features to their apps, like data tables, forms, charts, and more.

A number of programmable properties can be changed to change the widget's appearance, behaviour, and functionality. Developers might be given the option to manage the data source, column titles, and sorting options for a data table widget, for instance.

Widgets are created with what would be called much of an interactive design, enabling programmers to preview and make changes to their applications in real-time. This makes it easy to experiment with different layouts and configurations until the desired effect is reached.

Integration with data sources: Developers can comparatively easily display and manipulate data in real-time by integrating widgets with a variety of data sources, such as databases, APIs, and external services.

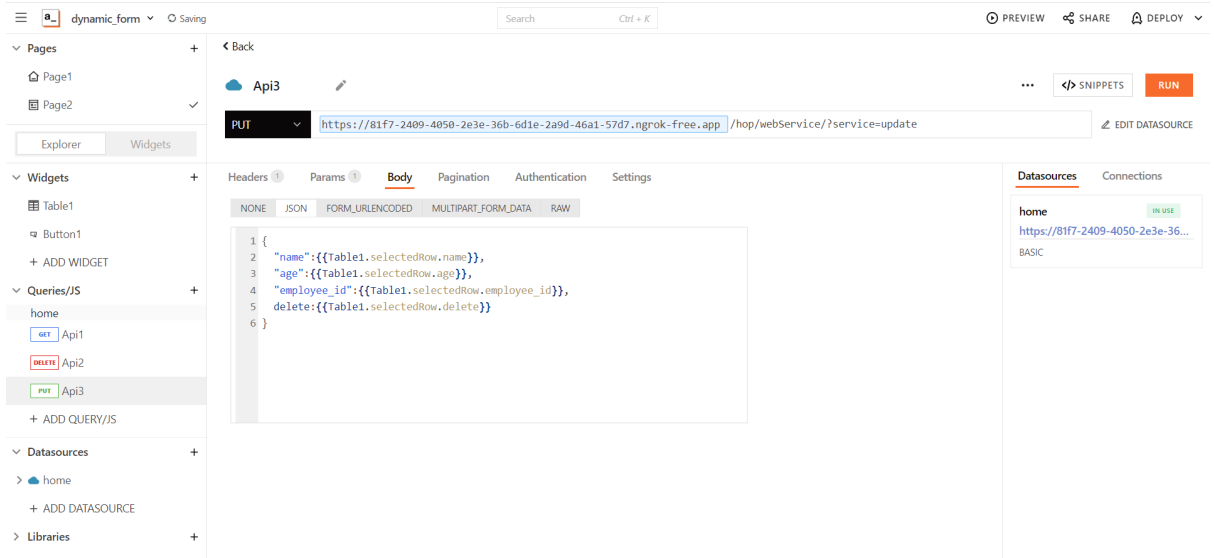
This is the front end form created using Appsmith widgets and api integration from ngrok connected to Apache Hop pipelines.



**Figure 13. Frontend with Appsmith**

In Appsmith, the routes and endpoints that handle incoming requests and generate results are referred to as APIs. The functionality and data of an application are made available through APIs so that other applications and services can access it programmatically.

With Appsmith's visual API definition interface, you can construct API endpoints and specify the logic for handling incoming requests and generating responses. Using the built-in widgets of Appsmith, you may specify the functionality of your API endpoints, including database searches, external API calls, and data manipulation activities.



**Figure 14. Api creation on Appsmith**

In Appsmith, APIs are used to make an application's logic and data available to other apps and services for programmatic access. APIs offer easy system integration by acting as a channel of communication between various software components.

In Appsmith, there are numerous API use cases, including:

Integrating with external systems: Using APIs, transferring data across systems and interacting with external systems like other web apps or services is a breeze.

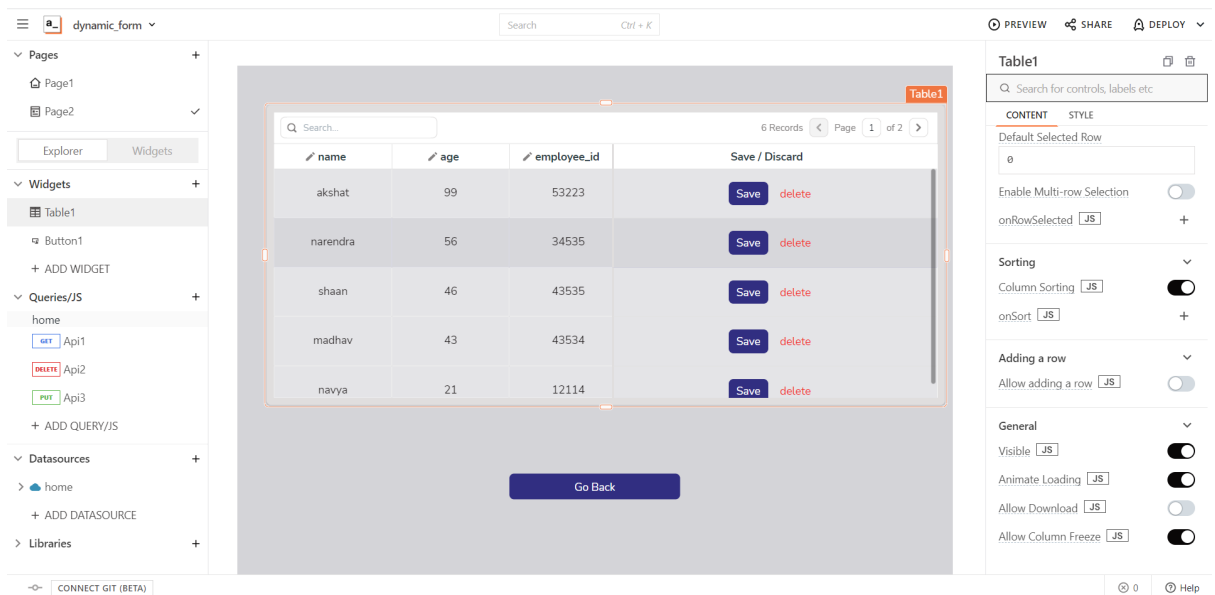
APIs can be used to build customised workflows and automate time-consuming jobs like data processing and reporting.

Microservice development: By using APIs, microservices that can be scaled and delivered individually can be built, providing more flexibility and efficiency.



By providing a backend for client-side apps, APIs can be utilised to accelerate the development of mobile and web applications.

Appsmith is a sophisticated tool for creating online services and applications that can be combined with other services and applications. Because you can create complex APIs without writing any code, Appsmith makes it easy to develop and deploy dependable online applications and services.



**Figure 15. Displaying data on appsmith**

Each transform in a Hop pipeline stands for a specific data operation, such as reading data from a file, filtering records using a set of specified criteria, or

publishing data to a database. The result of one transform serves as the input for the subsequent transform in the pipeline when transforms are combined in a specific order to build a pipeline.

The pipelines used by Hop are designed to be flexible, scalable, and able to process enormous amounts of data in a distributed computing environment. Pipelines can be operated using a variety of execution engines, including Apache Spark, Apache Flink, Google Cloud Dataflow, and others.

Hop offers a large selection of pre-built transforms and connectors, as well as the option to design your own transforms and connections to accommodate unique data sources or processing demands. Users can create intricate data processing systems with hop pipelines that can manage a variety of data processing jobs.

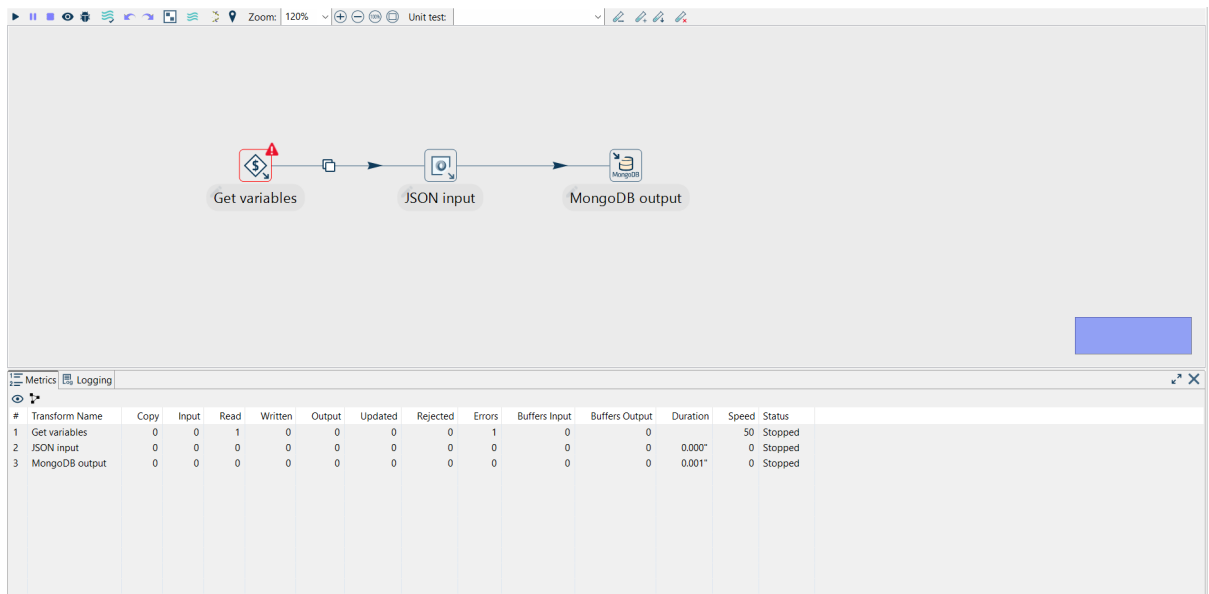


Figure 16 Pipelines in hop

There are several ways to integrate web services with Apache Hop. It provides REST APIs that may be used to programmatically manage and create workflows. Hop can also be used to create web services by creating routines that expose data through REST or SOAP endpoints. The built-in web server in Hop can be used to host the service and make it available online once it has been defined. As an alternative, you can set up a another web server, like Apache Tomcat or Jetty, to handle inbound web service requests and deploy the workflow there. Before using Apache Hop to create a web service, you can first create a workflow that does the necessary data translation or processing. The "HTTP POST" or "HTTP GET" phases can then be used to make the process accessible as a REST or SOAP service. You can define the input parameters for the service and map them to the input fields in the process by following these steps.

Web Services created on apache hop:

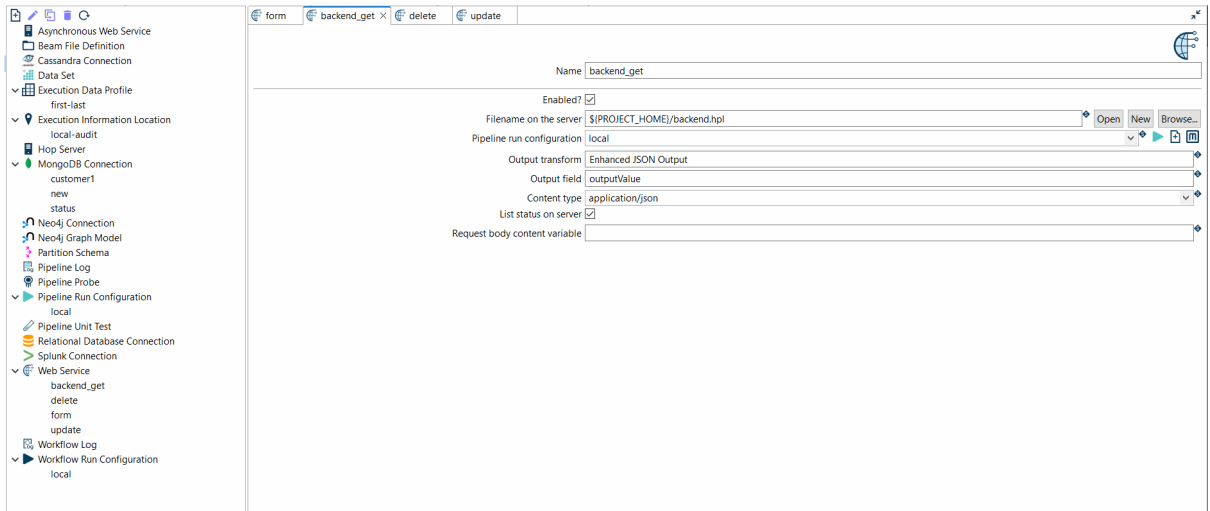


Figure 17. Webservices in hop

A collection in MongoDB is a container for organising and storing groups of documents. Collections are used to bring together related documents, just like tables are used in relational databases. In MongoDB, any number of documents, each with a potential different structure, can be found in a collection.

The following are important characteristics and advantages of collections in MongoDB:

The dynamic structure of MongoDB collections makes it possible to include documents with a wide range of formats and fields. This makes it possible to store and use unstructured data as well as model data in a flexible way.

To improve the efficiency and speed of queries, collections can be indexed on one or more fields. MongoDB supports a wide range of indexing options, such as text indexes, single field indexes, and compound indexes.

Sharding: MongoDB collections may be divided over numerous servers to improve scalability and performance. Sharding enables handling of extremely large data volumes by distributing massive data sets across multiple machines.

Atomic modifications to every document in a collection are possible with MongoDB's document-level atomicity feature. This ensures that every edit is made in a single, fluid step.

GridFS: MongoDB contains a feature called GridFS that makes it possible to save and retrieve large binary files as a part of a document in a collection. GridFS allows MongoDB to manage and store large files, such as images or videos.

Mongo db window of this lo-code/low-code

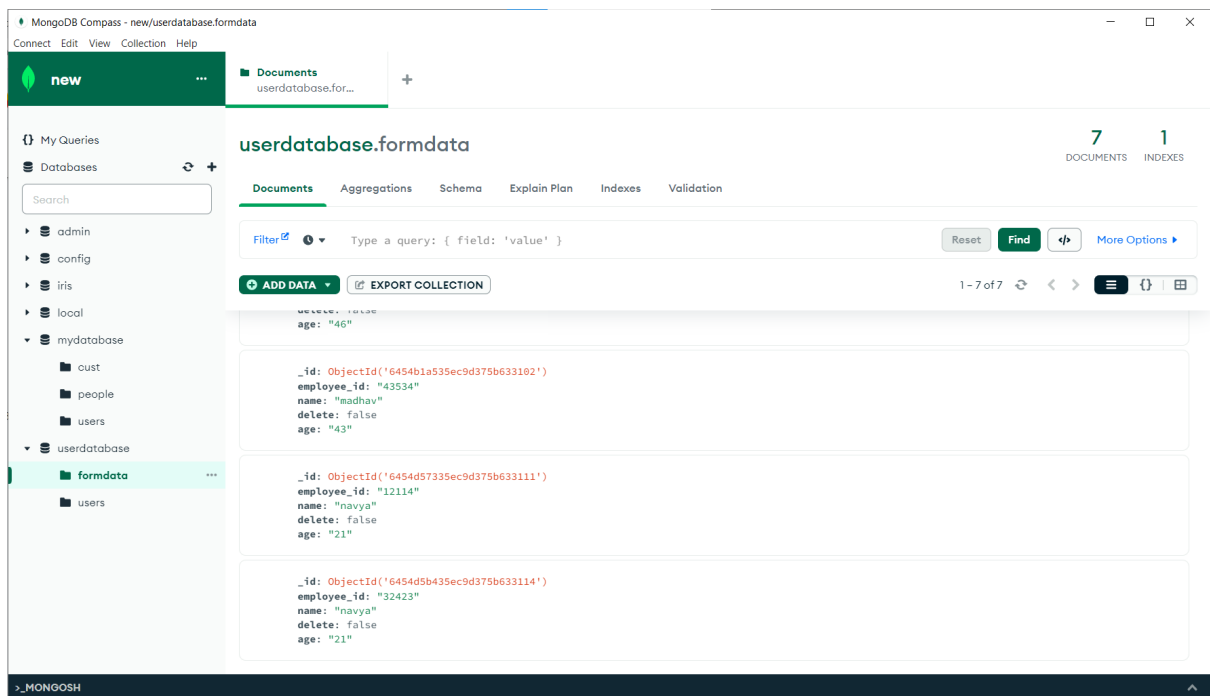


Figure 18 Database connected via Apache hop

The "API" widget should be dragged and dropped onto the page from the widgets pane.

To create a new API, select the "New API" button in the "API" widget properties panel.

Enter the API endpoint's URL along with any necessary headers, authentication information, or other parameters in the "New API" dialogue box.

Click the "Test API" button once you have entered all the necessary data to ensure the API is operational.

Click the "Save" button to save the API configuration if the API test is successful.

This API can now be used as per interest of the developer.

To use the API widget, you can:

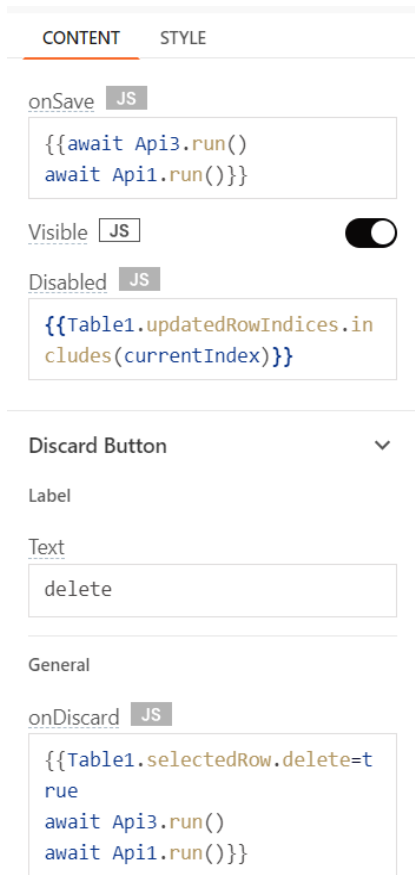
Drag and drop the widget onto the desired page to display the API results.

From the "API" drop-down menu in the widget properties panel, select the required API.

Make your request by selecting its type (such as GET, POST, PUT, or DELETE).

You should enter any essential request headers or parameters in the appropriate slots.

Click the "Send Request" button to send the API request and show the results on the page.



**Figure 19. Connecting api with frontend**

## Transforms

### JSON input data transform

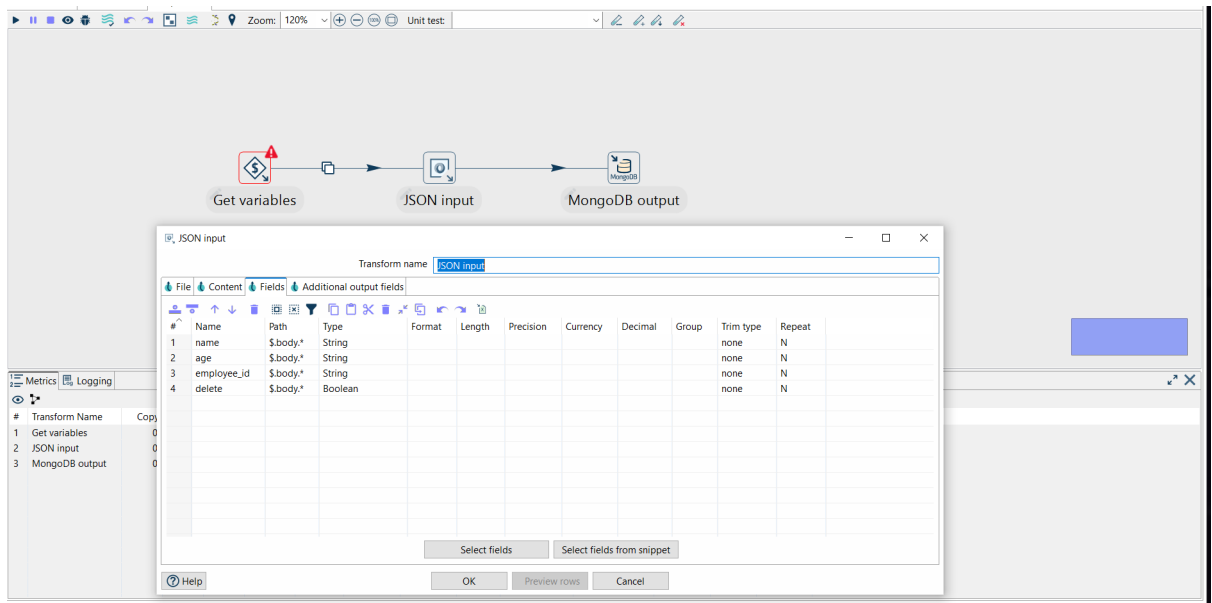
You can read and parse JSON data from many sources, such as files or internet APIs, using the JSON Input stage of the Apache Hop data integration tool (formerly known as Hop or Pentaho Data Integration).

You can specify the incoming JSON data's structure, map its fields to the transform's output fields, then, at your choice, filter or aggregate the data by applying this transform.

The JSON Input transform supports a variety of JSON types, including objects, arrays, and hierarchical structures. It also makes it possible to stream, flatten, and normalise data, among other data parsing possibilities.

When used with other transformations in Hop, the JSON Input transform can be used to do a number of data integration tasks like joining, aggregating, or changing the data. The resulting data can be exported to databases, files, and web services, among other forms.

Additionally, you can indicate how to handle incorrect or missing data, for as by giving it default values or eliminating it entirely.



**Figure 20. Example of how hop transforms are made**



## **Chapter 5 - CONCLUSION**

So, obtaining and analysing data from the entire work it is concluded that no-code/low-code platforms may offer a simpler alternative for people who are well versed with coding practices but not with actual coding may find it easier to work on these platforms given the drawbacks of less help and less flexibility ,but this might be okay to be compromised with given that people are not required to spend too much time learning coding and have more time to work on improving their overall service of the platform they are creating.

In comparison, the people who are creating their project from sheer basics and are already well-versed with coding might have a good option and these no-code/low-code platforms with a little more polishing ,documentation improvement and increased flexibility would convincingly be testing developers for their decision of traditional coding practices.

## REFERENCES

1. [1] Low-Code Platform Authors: Alexander C. Bock & Ulrich Frank ,Business & Information Systems Engineering Journal: *Business & Information Systems Engineering*
2. [2] Study of deployment of “low code no code” applications toward improving digitization of supply chain management Authors: Som Sekhar Bhattacharyya, Saurabh Kumar Journal: *Journal of Science and Technology Policy Management*
3. [3] Characteristics and Challenges of Low-Code Development: The Practitioners' Perspective Authors: Som Sekhar Bhattacharyya, Saurabh Kumar Journal: *Journal of Science and Technology Policy Management*
4. [4] Analysis of Low Code-No Code Development Platforms in comparison with Traditional Development Methodologies Authors: Shreyas Shridhar, Siddharth Bose Journal: *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*
5. [5] Low-Code Is Often High-Code, So We Must Design Low-Code Platforms to Enable Proper Software Engineering Authors: Timothy C. Lethbridge Journal: Part of the [Lecture Notes in Computer Science](#) book series (LNTCS,volume 13036)
6. [6] The Rise of No/Low Code Software Development—No Experience Needed? Authors: Marcus Woo Journal: Part of the [Lecture Notes in Computer Science](#) book series (LNTCS,volume 13036)

ORIGINALITY REPORT

---

4%

SIMILARITY INDEX

4%

INTERNET SOURCES

1%

PUBLICATIONS

3%

STUDENT PAPERS

---

PRIMARY SOURCES

---

1

Submitted to University of Stirling

Student Paper

1%

2

[www.digitalocean.com](http://www.digitalocean.com)

Internet Source

1%

3

[link.springer.com](http://link.springer.com)

Internet Source

1%

4

[www.semanticscholar.org](http://www.semanticscholar.org)

Internet Source

<1%

5

Hari Narayn. "Chapter 3 Start Reacting",  
Springer Science and Business Media LLC,  
2022

Publication

<1%

6

[scholar.archive.org](http://scholar.archive.org)

Internet Source

<1%

7

[www.unifr.ch](http://www.unifr.ch)

Internet Source

<1%

8

Submitted to Institute of Art Design and  
Technology

Student Paper

<1%

---