CODE FORGE

Login/Sign up Module

Key Responsibility

- Designed secure user authentication system
- Implemented login and sign up pages using Angular
- Used form validation for email, password, and confirmation
- Encrypted passwords before saving to database

Features

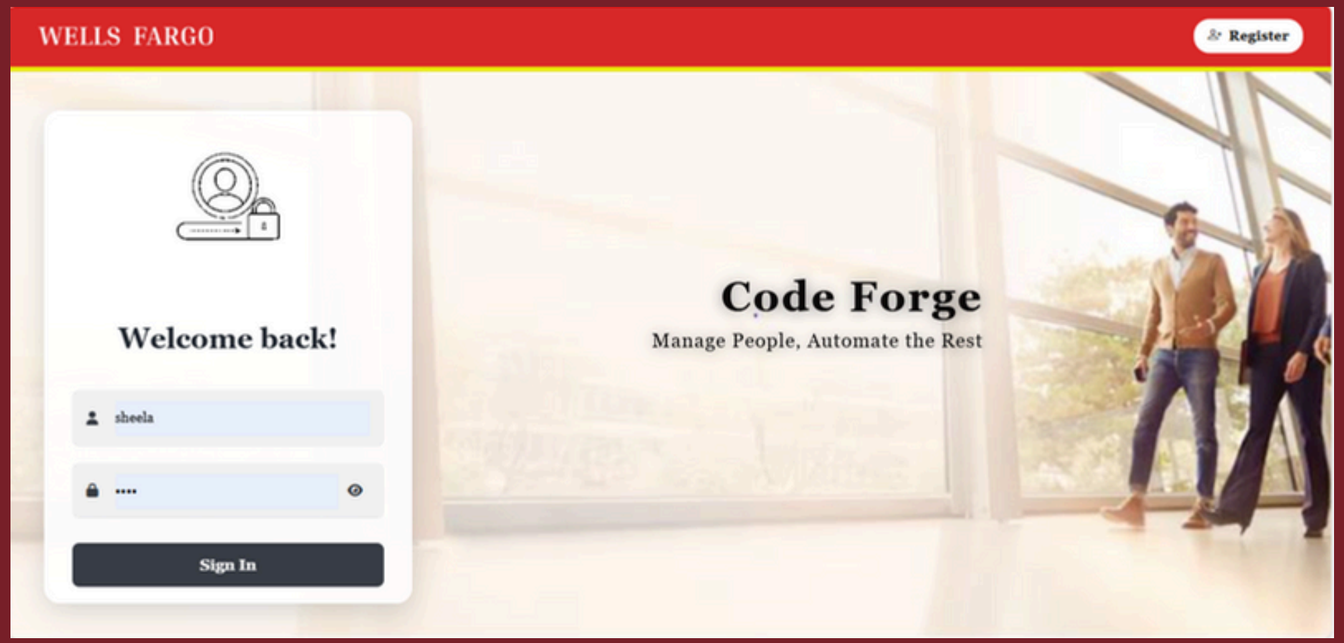
Instant feedback for input errors

Error handling (e.g., invalid credentials, duplicate user)

Success redirects and session setup

Proper Authentication and security using bcrypt and jwt

UI Component



UI Component

DATABASE

MySQL was selected for its scalability, ACID compliance, and high performance with transactional workloads. Its robust indexing and seamless integration with frameworks like Node.js make it an ideal choice for managing structured data efficiently.

Scalable

Handles large amounts of data efficiently

Reliable Data Integrity

Strong consistency and reliability

Robust Indexing

Fast search and query operations

Cross-platform

Available on various operating systems

Tables Required

1. **Employee**

Field	Type	Null	Key	Default	Extra
Emp_Id	int	NO	PRI	NULL	
First_Name	varchar(200)	NO		NULL	
Last_Name	varchar(200)	NO		NULL	
Dept_Id	int	YES	MUL	NULL	
Email_Id	varchar(200)	NO	UNI	NULL	
DOB	date	YES		NULL	
Role_Id	int	YES	MUL	NULL	

2. **Department**

Field	Type	Null	Key	Default	Extra
Dept_Id	int	NO	PRI	NULL	
Dept_Name	varchar(200)	NO		NULL	

3 Role

Field	Type	Null	Key	Default	Extra
Role_Id	int	NO	PRI	NULL	
Role_Name	varchar(200)	NO		NULL	
Access_Level	varchar(100)	YES		NULL	

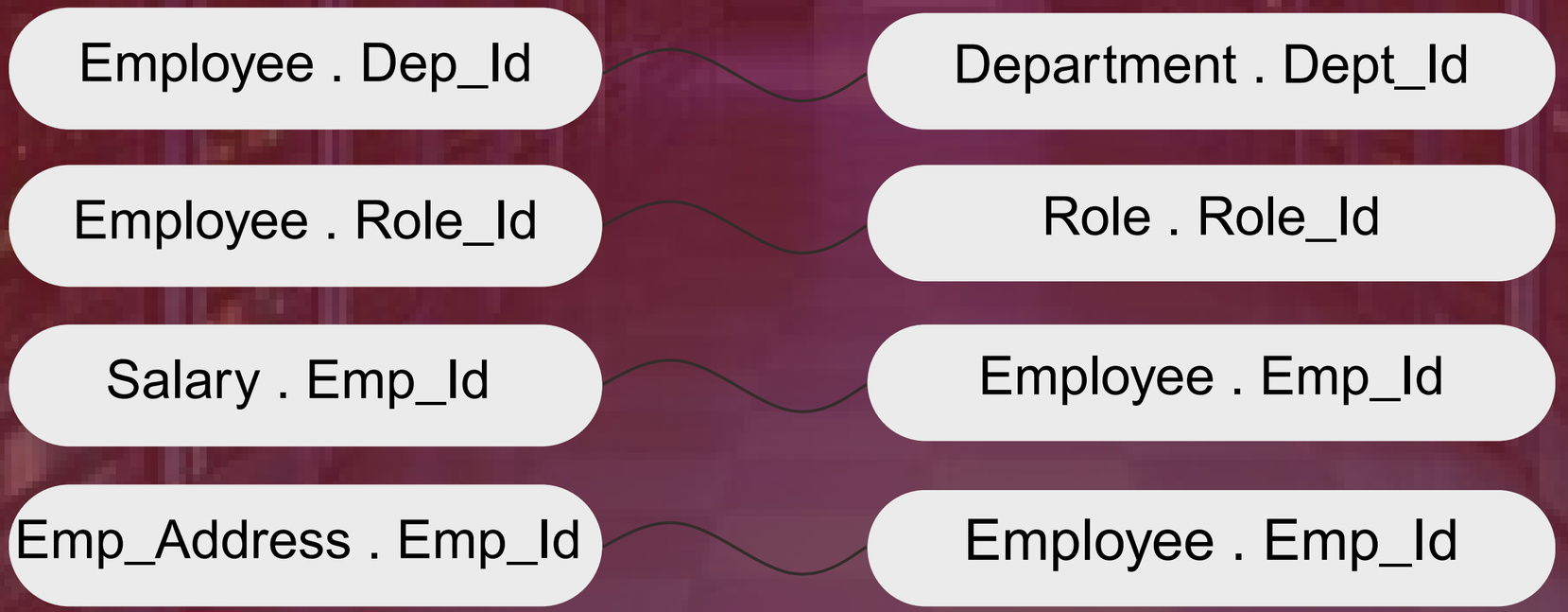
4 Salary

Field	Type	Null	Key	Default	Extra
Salary_Id	int	NO	PRI	NULL	
Emp_Id	int	YES	MUL	NULL	
Sal_Basic	decimal(10,2)	NO		NULL	
Bonus	decimal(10,2)	YES		NULL	
Total_CTC	decimal(10,2)	NO		NULL	
Last_Updated	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

5 Emp_Address

Field	Type	Null	Key	Default	Extra
Address_Id	int	NO	PRI	NULL	
Emp_Id	int	NO	MUL	NULL	
Address_Type	enum('Permanent','Current')	NO		NULL	
Address_Line1	varchar(200)	YES		NULL	
Address_Line2	varchar(200)	YES		NULL	
City	varchar(100)	YES		NULL	
State	varchar(100)	YES		NULL	
Country	varchar(100)	YES		NULL	
Postal_Code	varchar(100)	YES		NULL	

Relation among the tables



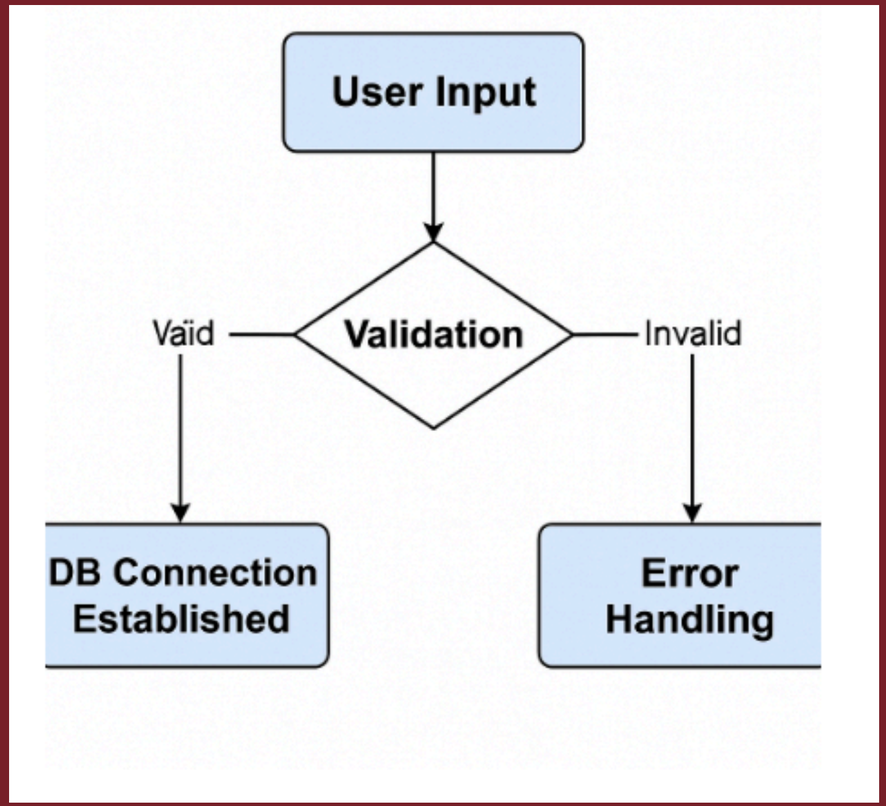
Database Connection Module

Key Responsibility

- Took user input for DB connection string
- Validated format and essential parameters
- Established secure connection to database
- Handled errors (invalid string,auth failures)

Highlights

- Real-time validation feedback to users
- Modular & reusable connection logic

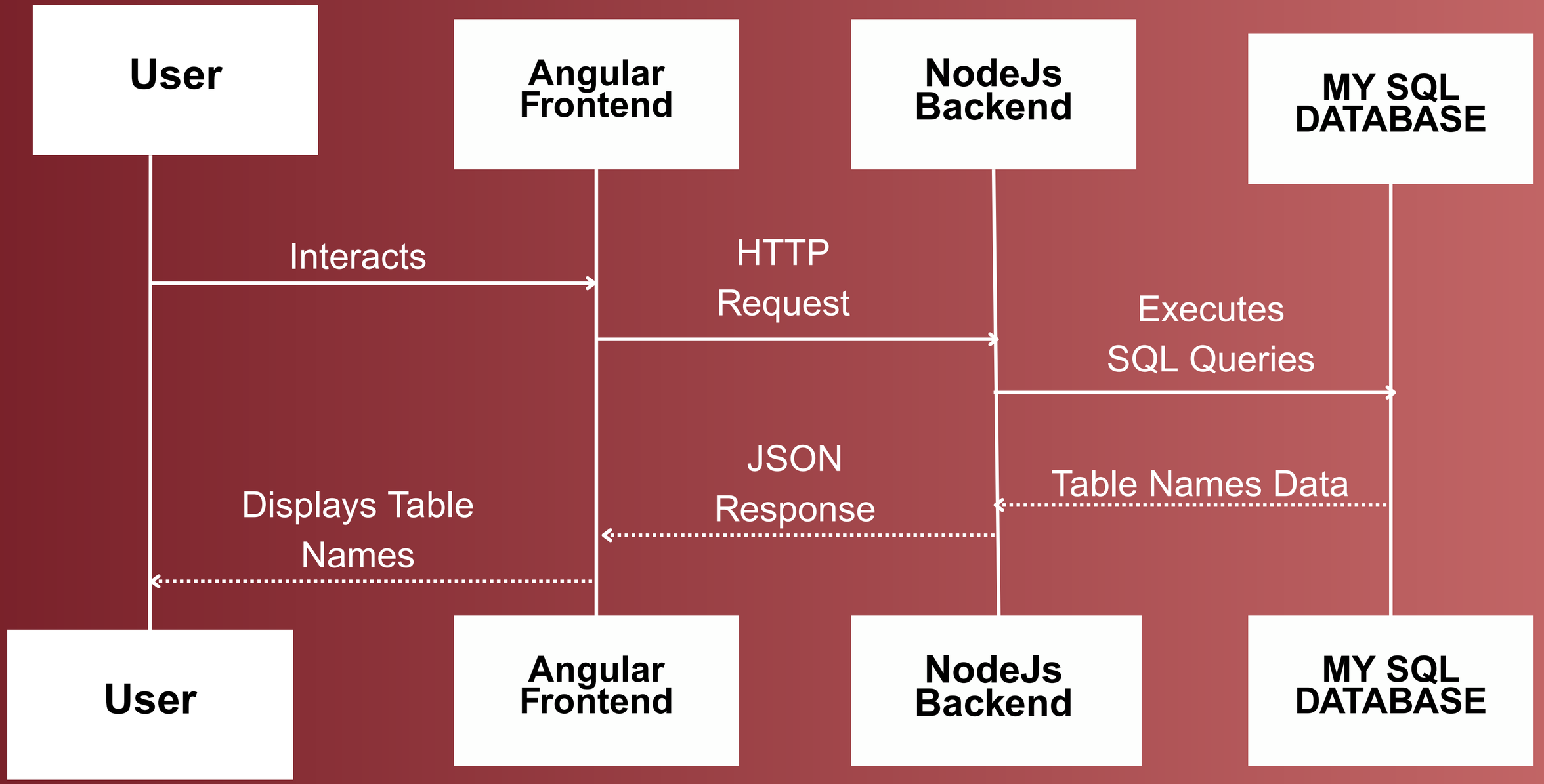

User Input
Validation
Vaïd
Invalid
DB Connection Established
Error Handling

Select Table

The select table allows to develop a dynamic web interface that fetches and displays all table names from a MySQL database using an Angular frontend and a Node.js backend

OVERVIEW

- Upon connecting to MYSQL,the system fetches all available tables dynamically.
- Users see a list of tables from the database.
- The system connects to the MySQL database.
- It retrieves all tables from the selected schema.
- Displays them in an interactive selection interface.
- The system processes the chosen table.
- It generates CRUD API endpoints (Create, Read, Update, Delete).
- Uses Node.js, Express, Sequelize for smooth database interaction.


User
Angular Frontend
NodeJs Backend
MY SQL DATABASE
Interacts
HTTP Request
Executes SQL Queries
JSON Response
Table Names Data
Displays Table Names
User
Angular Frontend
NodeJs Backend
MY SQL DATABASE

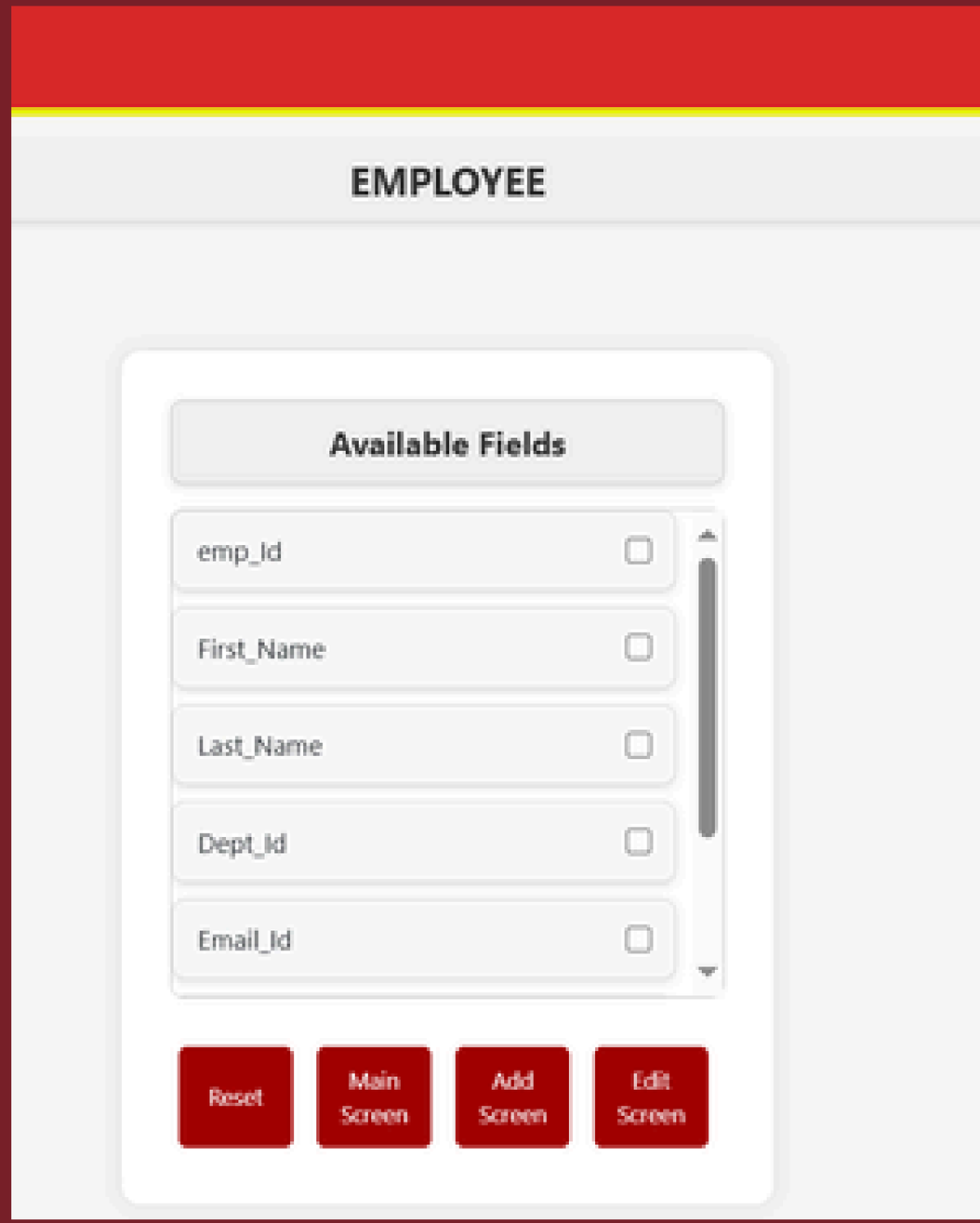
Key Features

Real-Time Data Retrieval
Interactive User Selection
Optimized for Efficiency
Error Handling & Validation

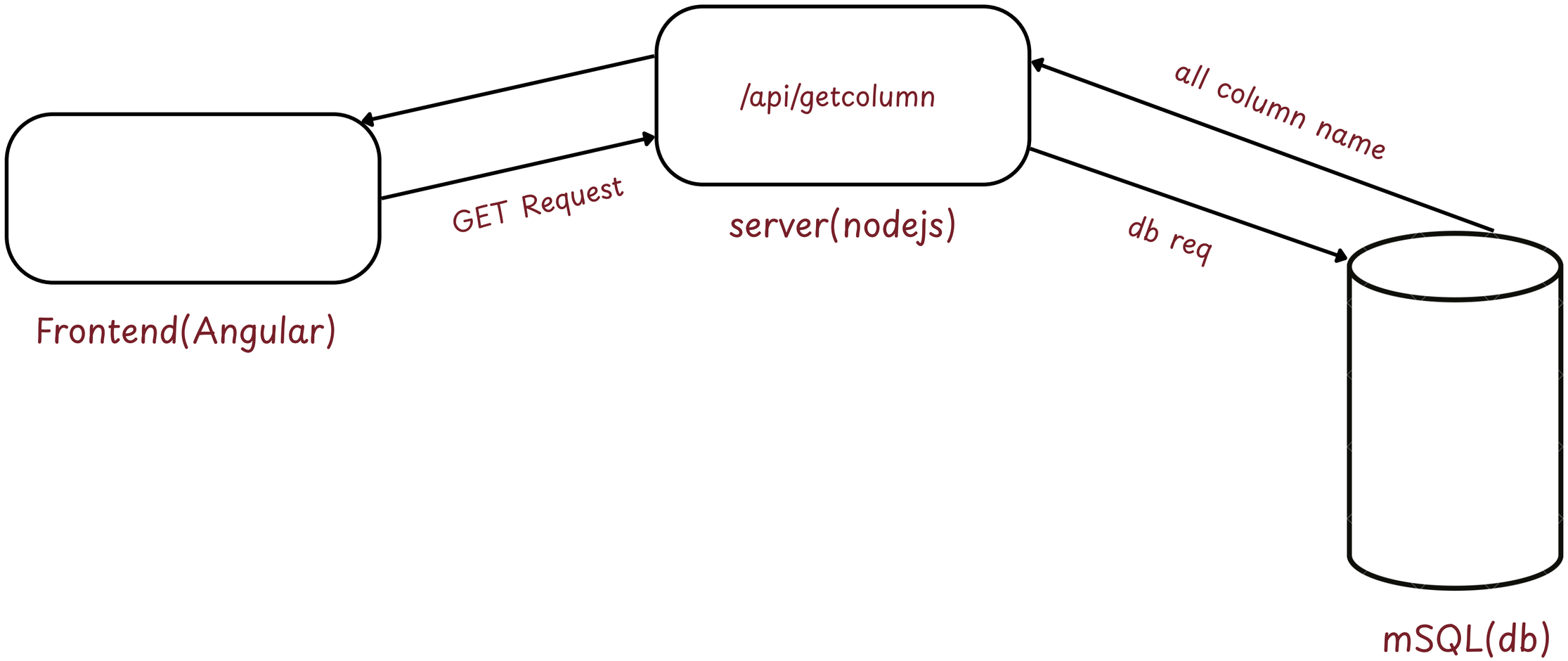
Select Column

Screen Implementation: Available Fields

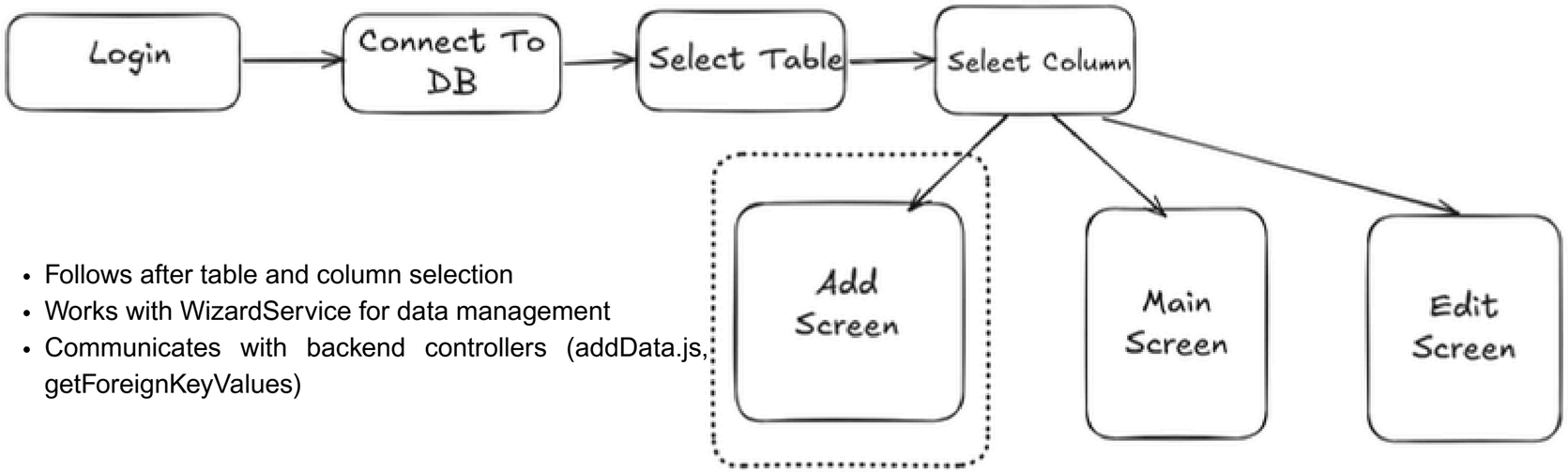
- Frontend built with Angular.
- Sends a GET request to /api/getColumn with the selected table name.
- Node js server receives request, fetches column metadata from MySQL.
- Response: JSON containing column names and types.
- UI dynamically renders available fields as selectable options.
- Buttons provided to Add, Edit, and Reset selections.



Flow


/api/getcolumn
all column name
GET Request
db req
Frontend(Angular)
server(nodejs)
mSQL(db)

ADD SCREEN



- Follows after table and column selection
- Works with WizardService for data management
- Communicates with backend controllers (addData.js, getForeignKeyValues)

Key Features

1. SMART INPUT TYPES

- Excludes Primary key.
- Handles Foreign Key.
- Assigns Type using getInputType.
- Email Validation.
- Options for ENUM datatype.
- Maximum attribute for date fields.
- Current Date and Time for TIMESTAMP.

Key Features

2. Data Validation

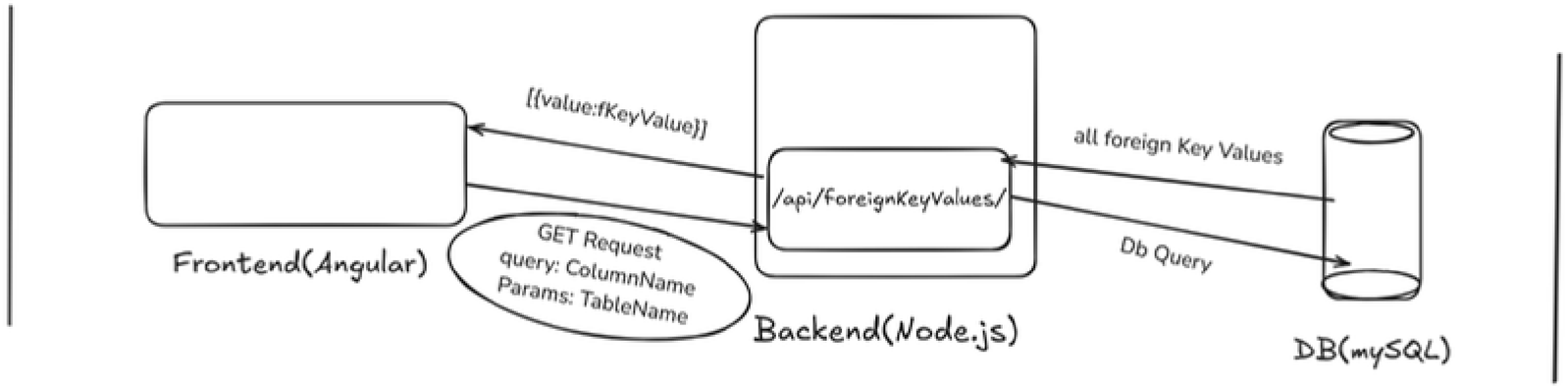
- Required field validation
- Email format validation

3. UI/UX Considerations

Responsive design

Scrollable form fields

Data Flow



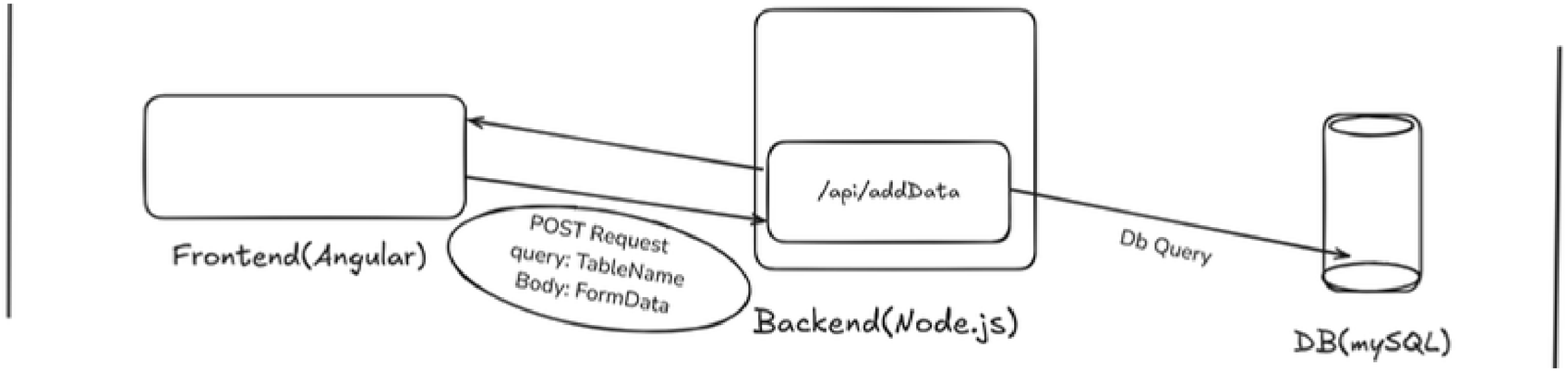
foreignKeyValues

Foreign Key Values

- Table Name - parameter, Column Name - Query.
- Selecting Referenced Table Name and Referenced Column Name.
- Fetching distinct values from a specific column in a specific table.

Data Flow

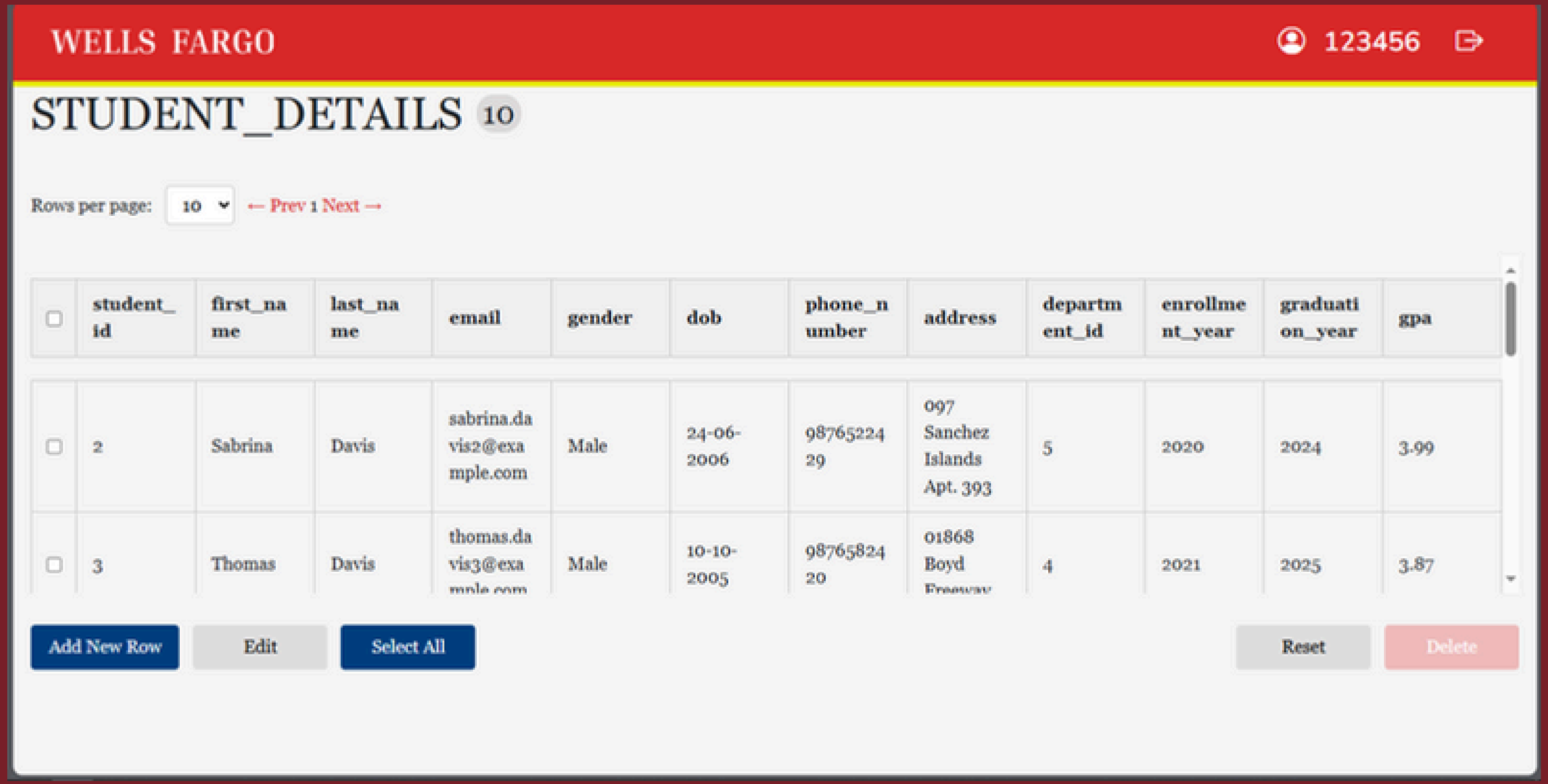
Add Data Flow



Screen Implementation: Main Screen

- Frontend built with Angular for dynamic data management.
- Fetches table data from backend using GET requests with pagination and sorting.
- Displays data in a responsive table with selectable columns and rows.
- Supports Add, Edit, Delete, and Reset actions for table rows.
- Implements row selection with checkboxes for bulk operations.
- Integrates with Node.js backend and MySQL for real-time data updates.

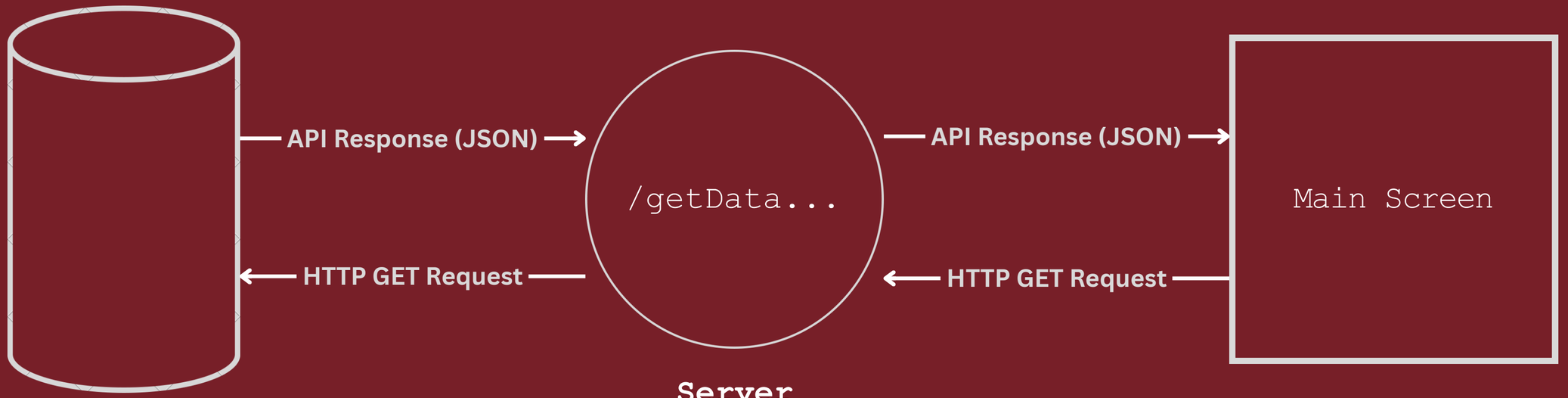
Main Screen


WELLS FARGO
123456
STUDENT_DETAILS 10
Rows per page: 10
← Prev 1 Next →
student_id
first_name
last_name
email
gender
dob
phone_number
address
department_id
enrollment_year
graduation_year
gpa
2
Sabrina
Davis
sabrina.davis2@example.com
Male
24-06-2006
9876522429
097 Sanchez Islands Apt. 393
5
2020
2024
3.99
3
Thomas
Davis
thomas.davis3@example.com
Male
10-10-2005
9876582420
01868 Boyd
4
2021
2025
3.87
Add New Row
Edit
Select All
Reset
Delete

Data Flow Diagram

API Call:

GET http://localhost:3000/getData?tableName=employees&limit=5&pageNo=1&sortBy=name&sortOrder=asc



Edit Screen

The auto-generated Edit Screen enables users to view, modify, and update the database

Code Autogeneration

The angular component for the edit screen is generated based on the predefined template in the backend

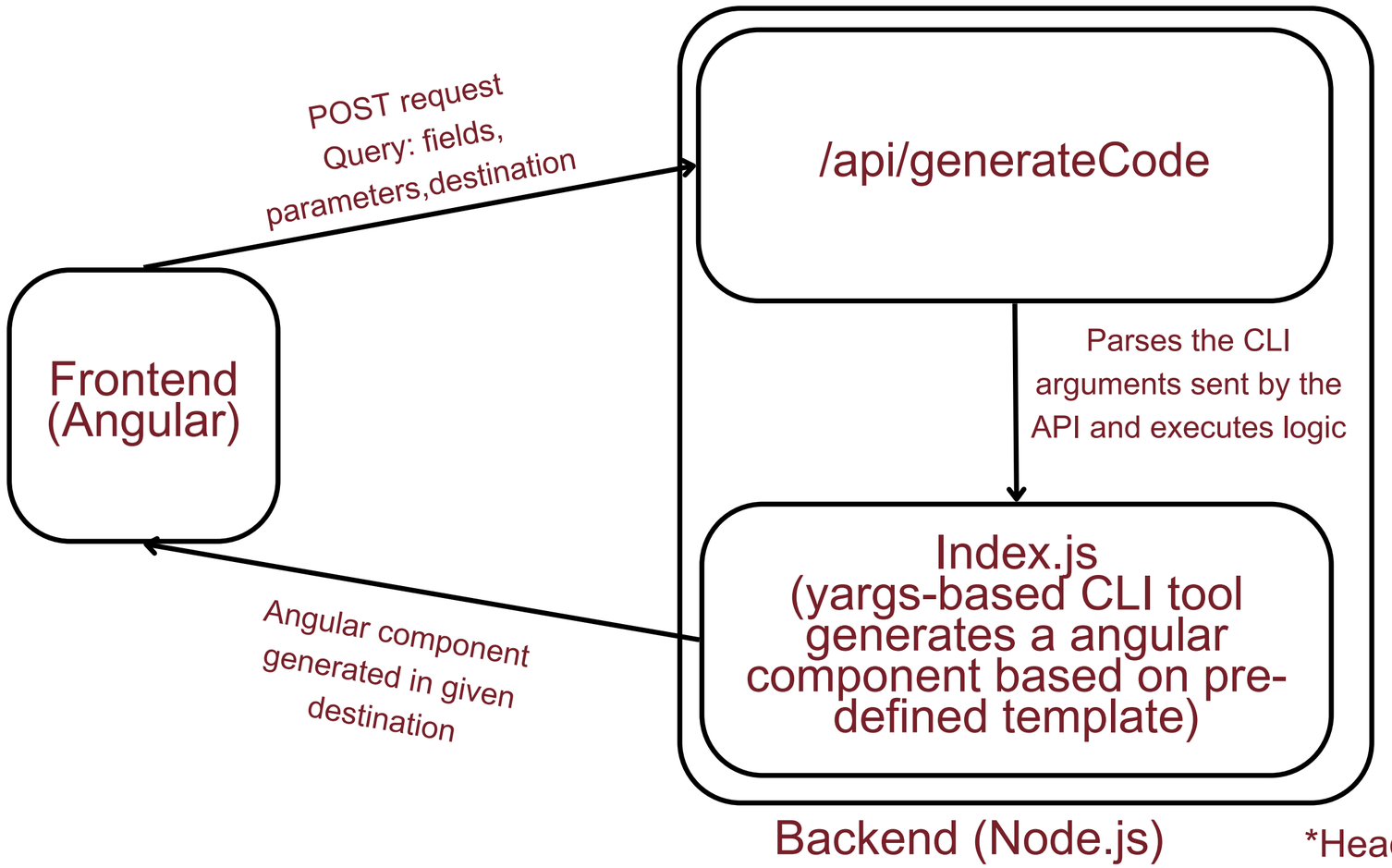
User Friendly interface

Allows the user to search through the data and organize them using sorting and pagination

Responsibilities

- Triggered frontend code generation based on predefined templates
- Fetch and display data from the selected tables
- Enabled in-place row updates with backend sync
- Implemented pagination, sorting, and search features

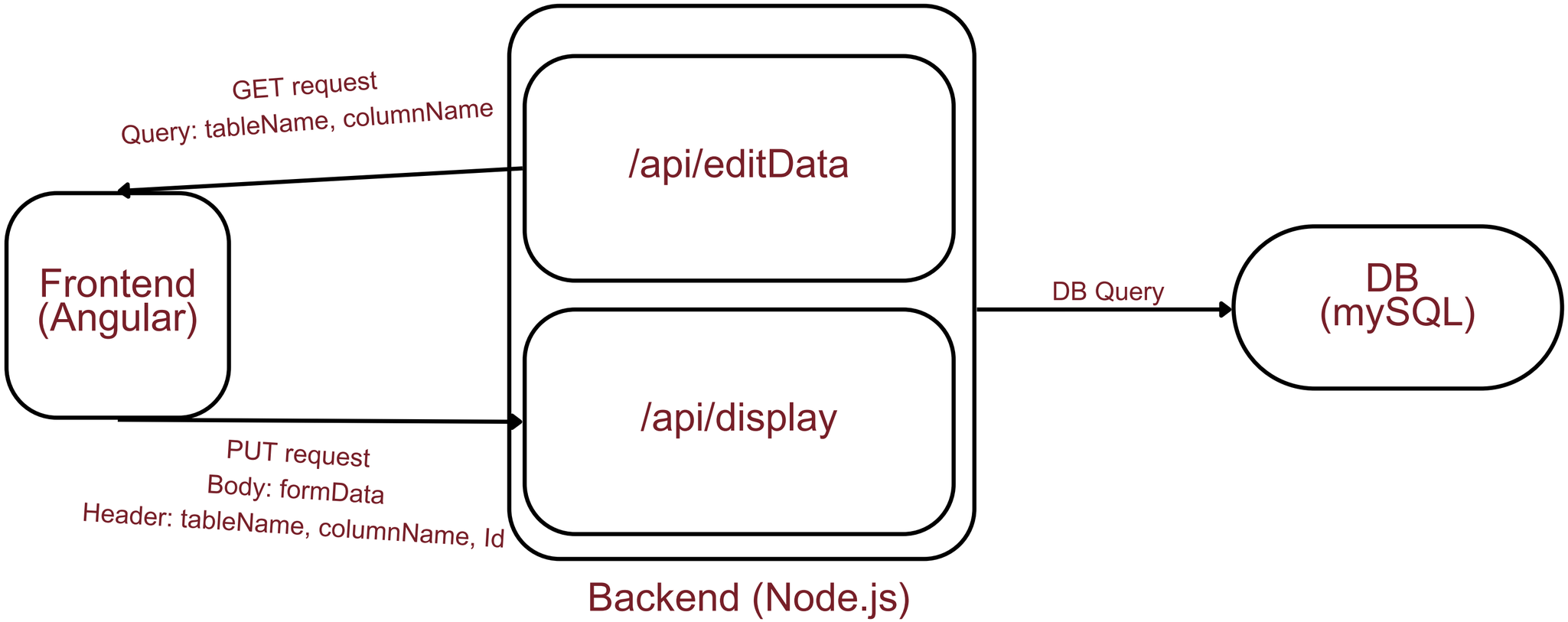
Code Autogeneration data flow



Field:
columnName:Datatype:key
Parameter:
tableName&primaryKeyName
Destination:
Location of generated code

*Header includes DB connection parameters

Data Flow Diagram



*Header includes DB connection parameters

Thank you