

# Mizuguna 01

lot4Aquaculture

**Guide: Prof Shiv Govind Singh**

Supported by: Mr. Chinmay Panda,  
Technical Officer, Department of Electrical Engineering,  
Indian Institute of Technology Hyderabad

Presentation By Vibhanshu Jain,  
B.tech Computer Science & Engineering Department,  
Indian Institute of Information Technology Raichur.



# Introduction

## Application for IoT infrastructure for inland fresh water fish farming

A network of Sensor Nodes spread across fish farms, which continuously monitor the water parameters and concentration of toxic substances and report to the farms on time to take the required action to improve productivity, improve the health of fish. Also to detect or avoid water poisoning, fish mortality like activities.

# Motivation

This project is developed under the Rural Development Center at IIT Hyderabad.

RDC was established with a vision to support rural development initiatives of the Government of India through innovative technologies being developed at IIT Hyderabad. One of the objective of RDC is to improve the productivity of Fish farmers by providing technology to monitor water quality and and scientific knowledge on fish farming to avoid fish mortality.

This project aims to develop an efficient web interface for visualization of the data. The user can very easily access the data of the sensor at particular timestamp to observe the changes in the sensor values.



# Market Study

SI No	Product Name	Country	Aim of Product	Product Type	Targeted Fish	Water quality Parameters
1	Future EU Aqua	Norway	Organic and conventional aquaculture of major fish species and low trophic level organisms in Europe	Hand held device with manual entry (WP5)	Salmon, Seabass, Sea Bream, Rainbow trout	Temperature, Oxygen level
2	AQUADAPT	Thailand, Cambodia, Myanmar, Vietnam	How the aquaculture sector, could adapt to climate change and at the same time be sustainable.	IoT system for DO, Tempt, PH, Ammonia, Turbidity measured and monitored in mobile app, Farmer can mange aerator & feed	Tilapia	Oxygen level,
3	Quadlink Technology	Taiwan	Smart Water Quality Monitoring System, environmental monitoring of aquaculture waters. Aquaculture feeding system. Remote monitoring of disease control. Monitoring of aquatic breeding. Feeding aquatic fingerlings. Automation and wisdom breeding facilities. High-density culture and history of big data.	Aquadlink: Smart Aquaculture Application System, Farmer can mange aerator, pump & feed	NA	Temp, DO(0~20 mg/L(water)), PH, ORP(- 1000 mV ~ + 1000 mV), Salinity(O~ 50 ppt)

# Project Objectives

The project objective is to develop an efficient and scalable application which store and visualize the data send by the sensors in the google sheet. The application should be easily accessible and will be used to visualize the data in the form of the plots and tables.

The application should also show the suggestions and message given by the AI in order to improve the water quality if it's getting polluted.

## Home Page

Title

Content

Details

## Login

Login

## Sensor Page

Sensor Information Page

Sensor 1

Plot

Table

Message

Prediction

Sensor 2

Plot

Table

Message

Prediction

Sensor 3

Plot

Table

Message

Prediction

Sensor 4

Plot

Table

Message

Prediction

Sensor 5

Plot

Table

Message

Prediction

Sensor 6

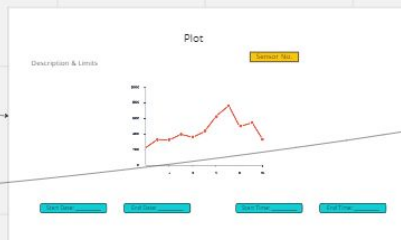
Plot

Table

Message

Prediction

## Plot



## Table

Table

Description

Height (m)	Date	Time	Sensor 1	Sensor 2
1	2023-10-27	10:00:00	10	10

## Prediction

Prediction

Sensor No.

Prediction No. / Text

Prediction No. / Text

Prediction No. / Text

Prediction No. / Text

## Message

Message

Description

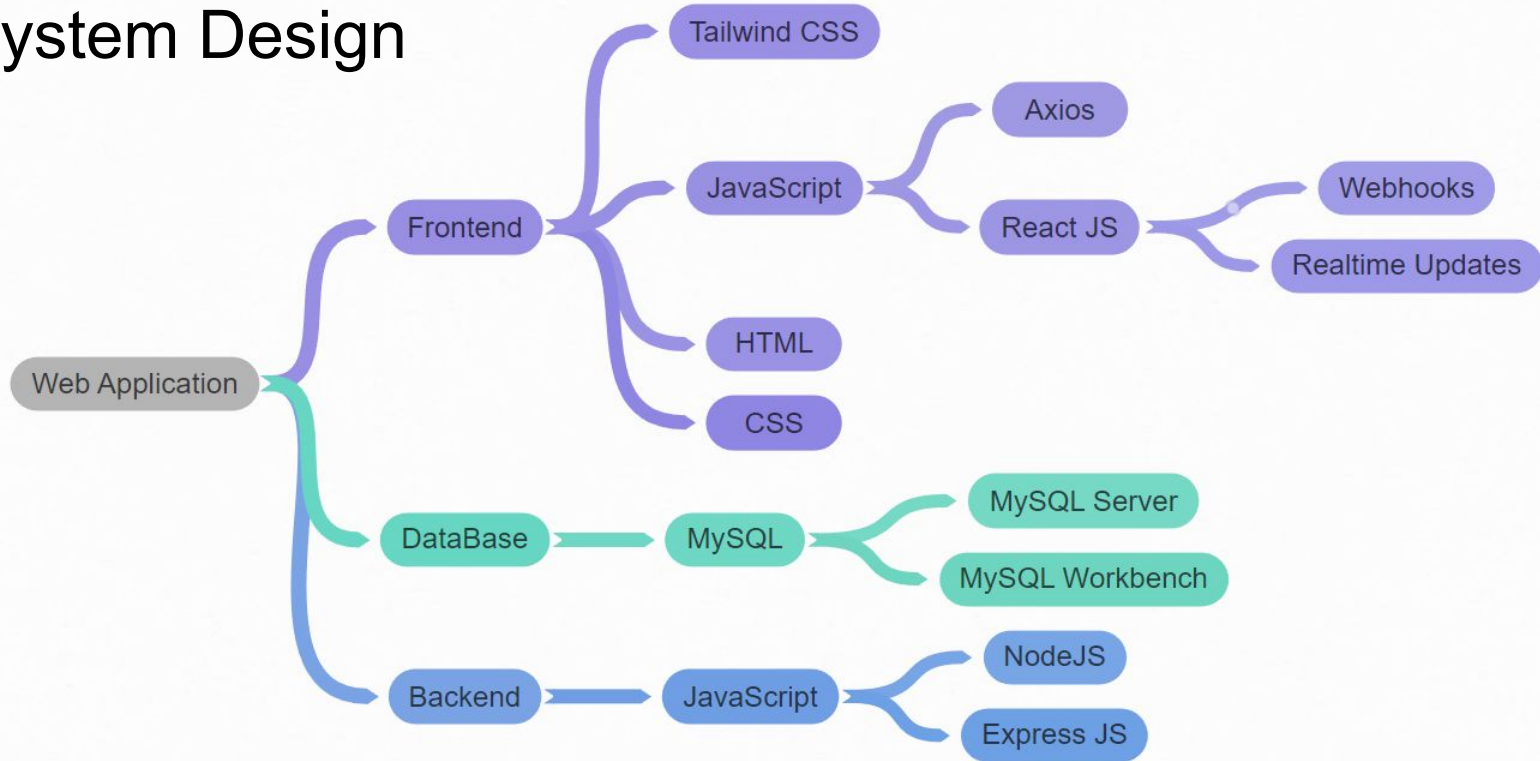
Message No. / Text

Message No. / Text

Message No. / Text

Message No. / Text

# System Design



# Frontend Application



# Database Architecture - MySQL

```
CREATE DATABASE sensor;  
CREATE TABLE `sensor`.`1` (  
  `dateTime` DATETIME NULL,  
  `temperature` DECIMAL(5,2) NULL,  
  `ph` DECIMAL(5,2) NULL,  
  `tds` DECIMAL(5,2) NULL,  
  `ammonia` DECIMAL(5,2) NULL,  
  `nitrate` DECIMAL(5,2) NULL,  
  `nitrite` DECIMAL(5,2) NULL,  
  `chlorine` DECIMAL(5,2) NULL,  
  `dissolvedOxygen` DECIMAL(5,2) NULL,  
  `orp` DECIMAL(5,2) NULL,  
  `label1` VARCHAR(255), `label2` VARCHAR(255), `label3` VARCHAR(255), `label4` VARCHAR(255),  
  `message1` VARCHAR(255), `message2` VARCHAR(255), `message3` VARCHAR(255), `message4` VARCHAR(255),  
  `prediction1` VARCHAR(255), `prediction2` VARCHAR(255), `prediction3` VARCHAR(255), `prediction4` VARCHAR(255))
```

# Backend Architecture

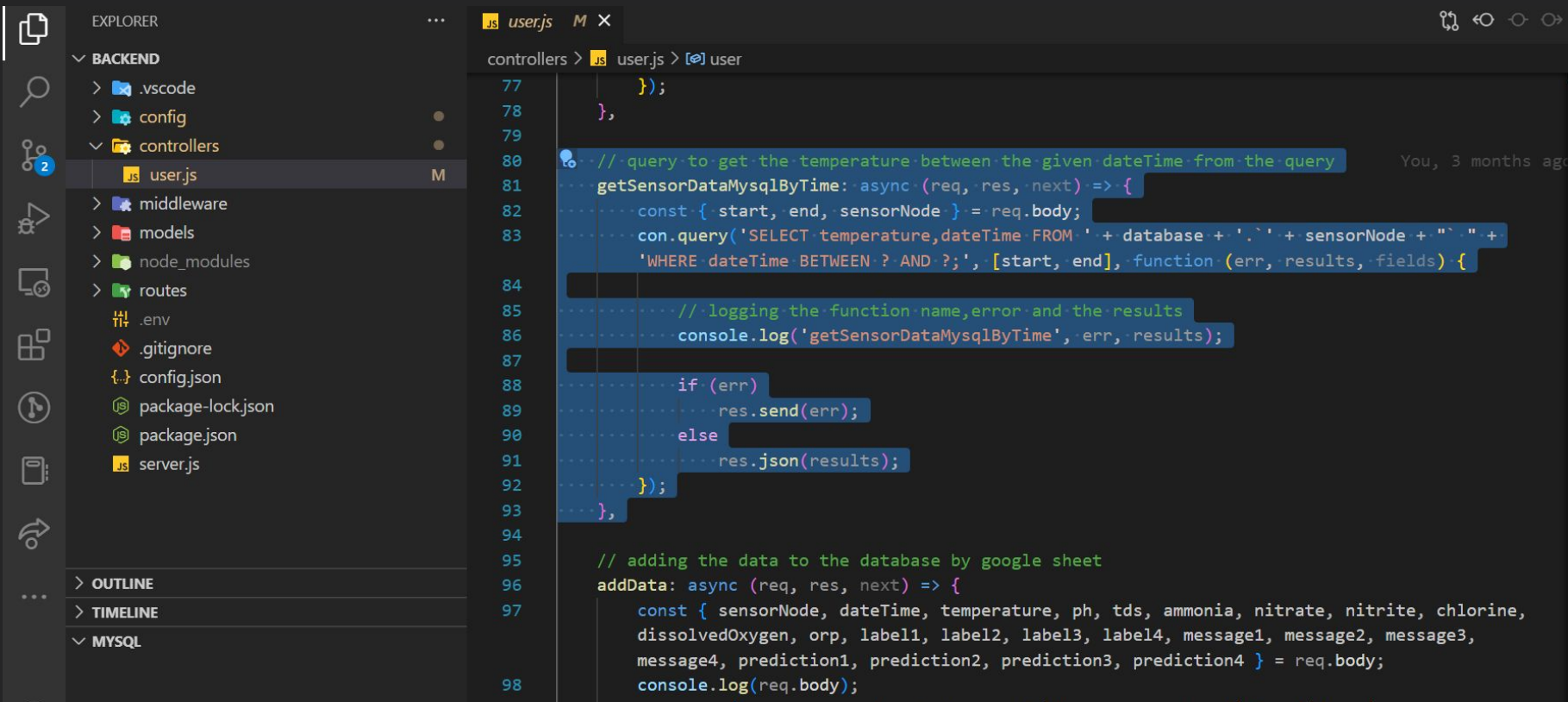
## Connecting with database

The image shows a Visual Studio Code editor interface. On the left, the Explorer sidebar displays the project structure under the 'BACKEND' folder. The files listed are .vscode, config, database.js (marked with an 'M'), controllers, middleware, models, node\_modules, routes, .env, .gitignore, config.json, package-lock.json, package.json, and server.js. Below the Explorer, the Outline and Timeline panels are visible, along with a 'MYSQL' section. The main editor area shows the 'database.js' file. The code defines a MySQL pool using 'mysql.createPool' with the following configuration: host: 'localhost', port: '3306', user: 'root', database: 'sensor', password: 'root', waitForConnections: true, connectionLimit: 10, and queueLimit: 0. A test function 'pool.getConnection' is used to verify the connection, logging 'Error connecting to Db' or 'Connection established'. The pool is then exported as 'module.exports'.

```
config > JS database.js > ...  
You, 9 minutes ago | 1 author (You)  
1 var mysql = require('mysql2');  
2 var pool = mysql.createPool({  
3   host: "localhost",  
4   port: "3306",  
5   user: "root",  
6   database: "sensor",  
7   password: "root",  
8   waitForConnections: true,  
9   connectionLimit: 10,  
10  queueLimit: 0  
11 });  
12  
13 // test the connection and log the results  
14 pool.getConnection(function (err, connection) {  
15   if (err) {  
16     console.log('Error connecting to Db');  
17     return;  
18   }  
19   console.log('Connection established');  
20   connection.release();  
21 });  
22  
23 module.exports = pool;
```

On the right side of the editor, a snippet bar shows a snippet titled 'You, last month • Mysql Pool Connection Added'.

## Getting data between given start and end time



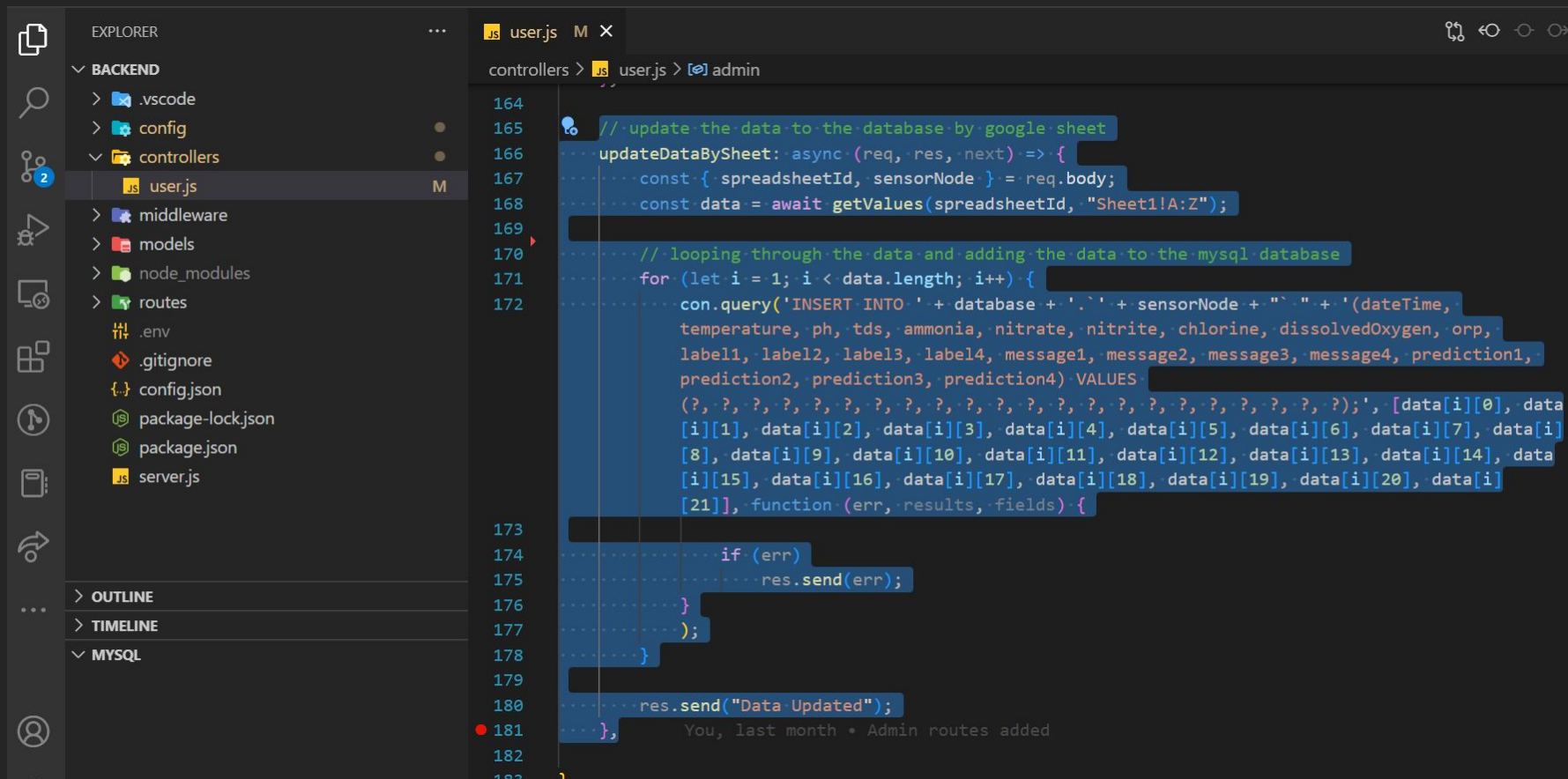
EXPLORER

- BACKEND
  - .vscode
  - config
  - controllers
    - user.js
  - middleware
  - models
  - node\_modules
  - routes
  - .env
  - .gitignore
  - config.json
  - package-lock.json
  - package.json
  - server.js
- OUTLINE
- TIMELINE
- MYSQL

user.js

```
77     });
78   },
79
80   // query to get the temperature between the given dateTime from the query
81   getSensorDataMysqlByTime: async (req, res, next) => {
82     const { start, end, sensorNode } = req.body;
83     con.query('SELECT temperature,dateTime FROM ' + database + '.' + sensorNode + " " +
84       'WHERE dateTime BETWEEN ? AND ?;', [start, end], function (err, results, fields) {
85       // logging the function name, error and the results
86       console.log('getSensorDataMysqlByTime', err, results);
87
88       if (err)
89         res.send(err);
90       else
91         res.json(results);
92     });
93   },
94
95   // adding the data to the database by google sheet
96   addData: async (req, res, next) => {
97     const { sensorNode, dateTime, temperature, ph, tds, ammonia, nitrate, nitrite, chlorine,
98       dissolvedOxygen, orp, label1, label2, label3, label4, message1, message2, message3,
99       message4, prediction1, prediction2, prediction3, prediction4 } = req.body;
100     console.log(req.body);
```

# Updating the data by Google sheet to MySQL



The image shows a VS Code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'BACKEND' folder containing 'controllers', 'middleware', 'models', 'node\_modules', 'routes', and 'server.js'. The 'controllers' folder is expanded, showing 'user.js'. The code editor displays the 'user.js' file, which contains a function 'updateDataBySheet' that updates data from a Google Sheet to a MySQL database. The code is as follows:

```
164
165 //update the data to the database by google sheet
166 updateDataBySheet: async (req, res, next) => {
167     const { spreadsheetId, sensorNode } = req.body;
168     const data = await getValues(spreadsheetId, "Sheet1!A:Z");
169
170     //looping through the data and adding the data to the mysql database
171     for (let i = 1; i < data.length; i++) {
172         con.query('INSERT INTO ' + database + '.' + sensorNode + " (" + '(dateTime,
173             temperature, ph, tds, ammonia, nitrate, nitrite, chlorine, dissolvedOxygen, orp,
174             label1, label2, label3, label4, message1, message2, message3, message4, prediction1,
175             prediction2, prediction3, prediction4) VALUES
176             (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);', [data[i][0], data
177             [i][1], data[i][2], data[i][3], data[i][4], data[i][5], data[i][6], data[i][7], data[i]
178             [8], data[i][9], data[i][10], data[i][11], data[i][12], data[i][13], data[i][14], data
179             [i][15], data[i][16], data[i][17], data[i][18], data[i][19], data[i][20], data[i]
180             [21]], function (err, results, fields) {
181
182             if (err)
183                 res.send(err);
184             }
185         );
186     }
187     res.send("Data Updated");
188 }, You, last month • Admin routes added
189
```

## Adding the data to MySQL

EXPLORER

BACKEND

- .vscode
- config
- controllers
  - user.js
- middleware
- models
- node\_modules
- routes
- .env
- .gitignore
- config.json
- package-lock.json
- package.json
- server.js

OUTLINE

TIMELINE

MYSQL

```
93 },
94
95 // adding the data to the database by google sheet
96 addData: async (req, res, next) => {
97     const { sensorNode, dateTime, temperature, ph, tds, ammonia, nitrate, nitrite, chlorine,
98         dissolvedOxygen, orp, label1, label2, label3, label4, message1, message2, message3,
99         message4, prediction1, prediction2, prediction3, prediction4 } = req.body;
100     console.log(req.body);
101     con.query('INSERT INTO ' + database + '.' + sensorNode + ' (' + dateTime, temperature,
102         ph, tds, ammonia, nitrate, nitrite, chlorine, dissolvedOxygen, orp, label1, label2,
103         label3, label4, message1, message2, message3, message4, prediction1, prediction2,
104         prediction3, prediction4) VALUES
105         (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);', [dateTime,
106         temperature, ph, tds, ammonia, nitrate, nitrite, chlorine, dissolvedOxygen, orp, label1,
107         label2, label3, label4, message1, message2, message3, message4, prediction1, prediction2,
108         prediction3, prediction4], function (err, results, fields) {
109             // logging the function name, error and the results
110             console.log('addData', err, results);
111             if (err) {
112                 res.send(err);
113             } else {
114                 res.json(results);
115             }
116         });
117     },
```

You, 3 months ago • data adding query added ...



# Conclusion

The web-application is developed which contains all the necessary information and feature along with the scalable backend and MySQL database.



# Future Scope

The application can be deployed on highly scalable platforms using the different parallel processing algorithms. A good and efficient AI model can be used for the data analytics and predictive modeling.

