

# BIRDI TMS DOCUMENTATION

---

<b>END USER DOCUMENTATION.....</b>	<b>1</b>
<b>Software requirements.....</b>	<b>2</b>
Requirements for Microsoft Windows OS.....	2
Hardware Requirement.....	2
Software Requirement.....	2
<b>Installation Guide.....</b>	<b>2</b>
<b>How to start the system.....</b>	<b>3</b>
<b>Screenshot of features.....</b>	<b>4</b>
Register new user.....	4
Login.....	4
Dashboard.....	5
Create new task.....	5
Edit task.....	6
Expand Task.....	7
Delete Task.....	8
LogOut.....	8
Error Handling.....	9
Responsive Designs.....	12
<b>TECHNICAL DOCUMENTATION.....</b>	<b>14</b>
Basic Architecture Design.....	14
BIRDITMS API.....	15
Architecture Pattern.....	15
Repository Pattern.....	15
Dependency Injection.....	15
Security.....	15
ASP.NET IDENTITY Provider.....	15
JWT Authorization.....	15
Optimization.....	16
Autmapper.....	16
Integration.....	16
EF Core - Code First Approach.....	16
Error Handling.....	16
GlobalException Middleware.....	16
NLog.....	16
BIRDITMS.REACT.....	16

Architecture Pattern.....	16
Components.....	16
Redux Saga.....	17
Security.....	17
Protected Route.....	17
SessionStorage JWT.....	17
Integration.....	17
Axios.....	17
Error Handling.....	17
TypeScript.....	17
EsLint.....	17
Optimization.....	18
Webpack.....	18
Prettier.....	18
Styling.....	18
MUI V5.....	18
Deployment.....	18
Docker.....	18
Contact.....	18

# END USER DOCUMENTATION

In this section, we will only discuss pre-requisite, installation process, commands to run the application and features for the end user.

---

## Software requirements

This software does not require anything to run apart from the browser if it's deployed on any server. However, if the user is another software developer he has to follow below steps to run the software successfully on a local machine.

## Requirements for Microsoft Windows OS

### Hardware Requirement

1. Windows 10
2. At Least 8 GB RAM
3. At Least 200 GB hard disk

### Software Requirement

#### **Mandatory**

1. Docker Desktop

#### Optional

1. Visual Studio 2022
2. SQL server Studio 2019
3. .NET framework 8
4. Node js 20
5. Npm > 10.7
6. Nvm 1.1.9

# Installation Guide

## Mandatory

Please install docker desktop from

<https://www.docker.com/products/docker-desktop/>

## Optional

Download <https://visualstudio.microsoft.com/downloads/>

Download <https://dotnet.microsoft.com/en-us/download/dotnet/8.0>

Download <https://nodejs.org/en/download/prebuilt-installer>

Download <https://github.com/coreybutler/nvm-windows>

Download <https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>

## How to start the system

1. Clone repository from GitHub  
[git@github.com:tusharborchate/BirdiTMS.git](https://github.com:tusharborchate/BirdiTMS.git)  
  
[git@github.com:tusharborchate/birditms.react.git](https://github.com:tusharborchate/birditms.react.git)
2. Please note that your folder structure should look like follow:  
Root - >  
    BirdiTMS  
    Birditms.react
3. Open command prompt and change directory to BirdiTMS  
Cd root\BirdiTMS
4. If you want to run without docker then change SQL connection string.
5. If you are running on docker then keep SQL server as container name in connection string.
6. Run following docker command

Docker-compose build --no-cache  
Docker-compose up -d

7. Please make sure that the following ports are unused.

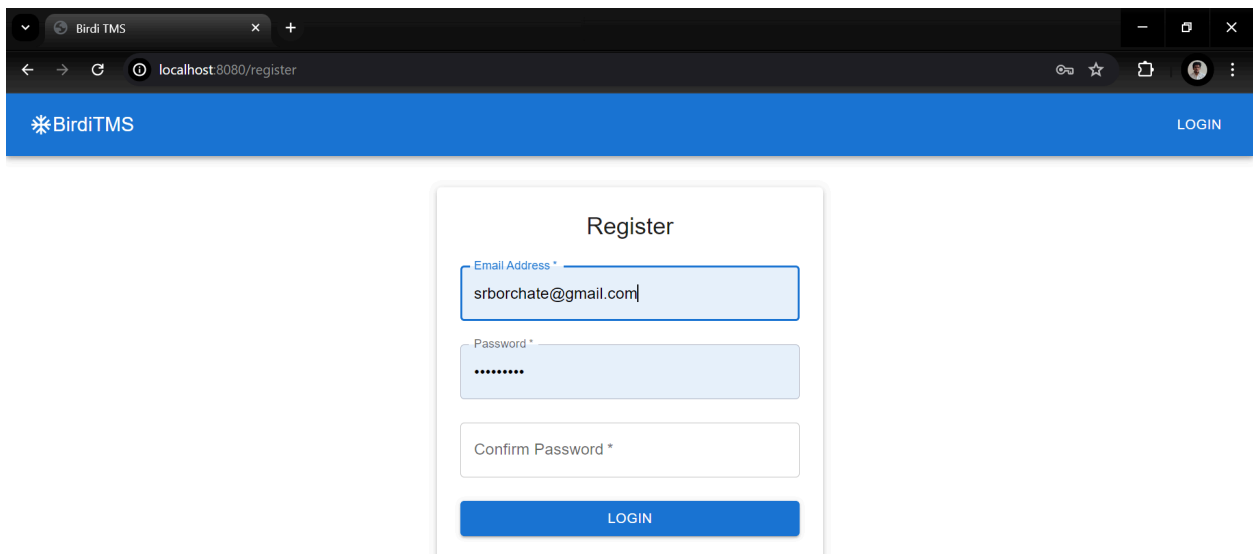
8080 for react app  
8001 for .net core api  
8002 for sql server

8. Open your browser and navigate to <http://localhost:8080/>

## Screenshot of features

### Register new user

Create new user



The screenshot shows a web browser window with the title 'Birdi TMS'. The address bar shows 'localhost:8080/register'. The page has a blue header with the 'BirdiTMS' logo on the left and a 'LOGIN' link on the right. Below the header is a white registration form titled 'Register'. The form contains three input fields: 'Email Address \*' with the value 'srborchate@gmail.com', 'Password \*' with masked characters, and 'Confirm Password \*'. At the bottom of the form is a blue button labeled 'LOGIN'.

# Login

Login with credentials

Birdi TMS

SIGN UP

### Login

Email Address \*

srborchate@gmail.com

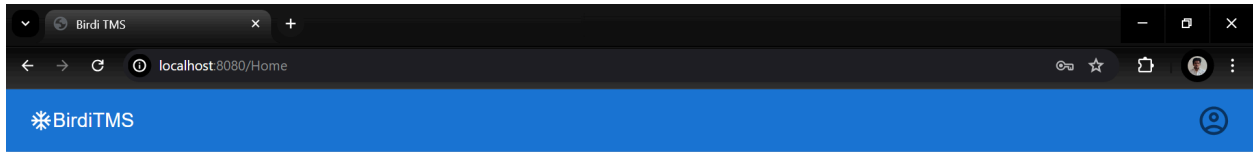
Password \*

\*\*\*\*\*

LOGIN

# Dashboard

If there is no task



Tasks

CREATE TASK

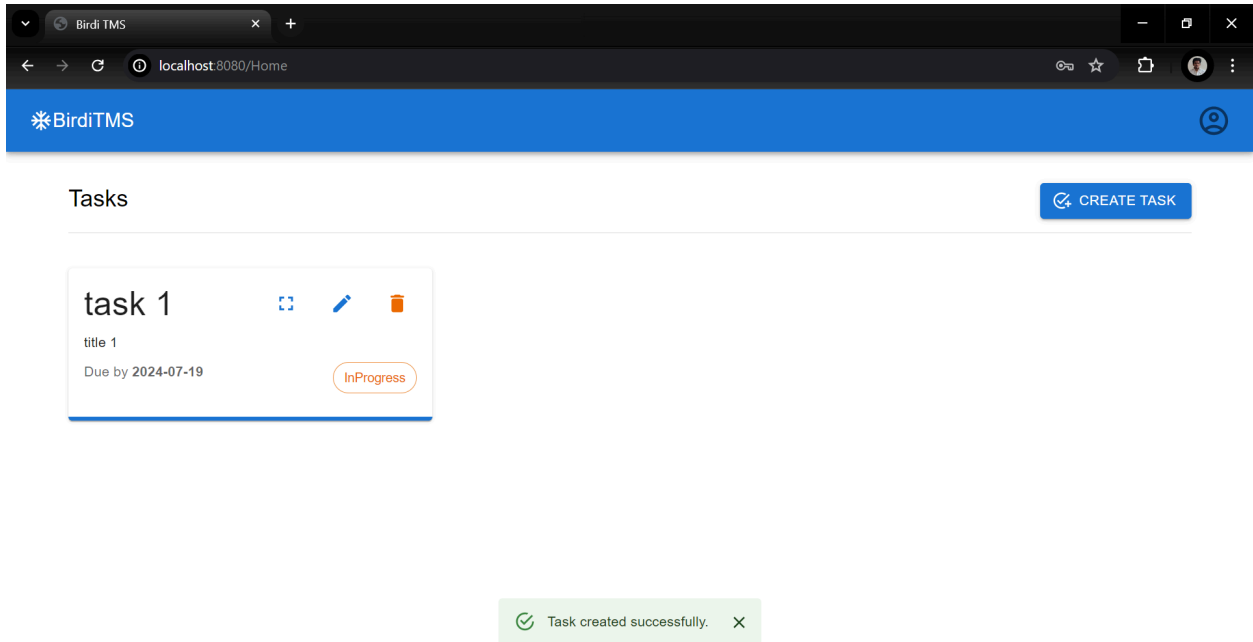


You have not created any task.

## Create new task

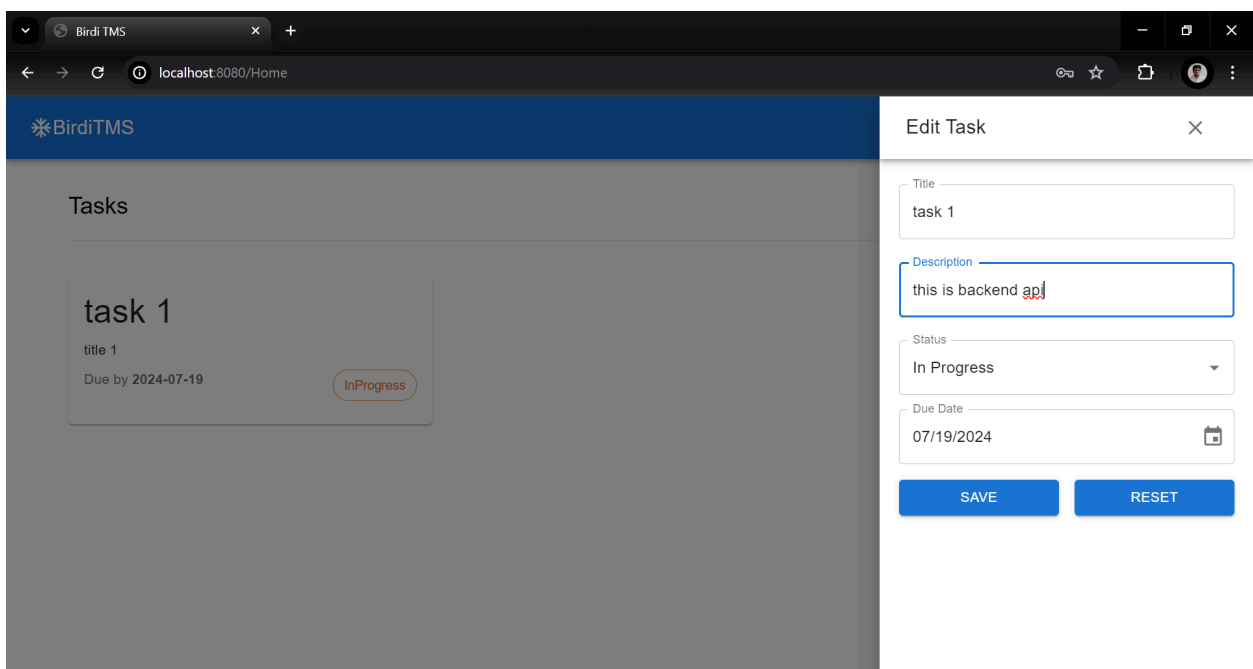
User can create new task

The screenshot shows the 'BirdiTMS' web application with a 'Create Task' modal form open on the right side. The background is dimmed. The modal form has a title 'Create Task' and a close button. It contains four input fields: 'Title' with the value 'task 1', 'Description' with the value 'this task is backend api', 'Status' with a dropdown menu showing 'Open', and 'Due Date' with the value '07/20/2024'. At the bottom of the form are two buttons: 'SAVE' and 'RESET'. The background shows the 'Tasks' section with the same clipboard icon and the text 'You have not created any task.'



## Edit task

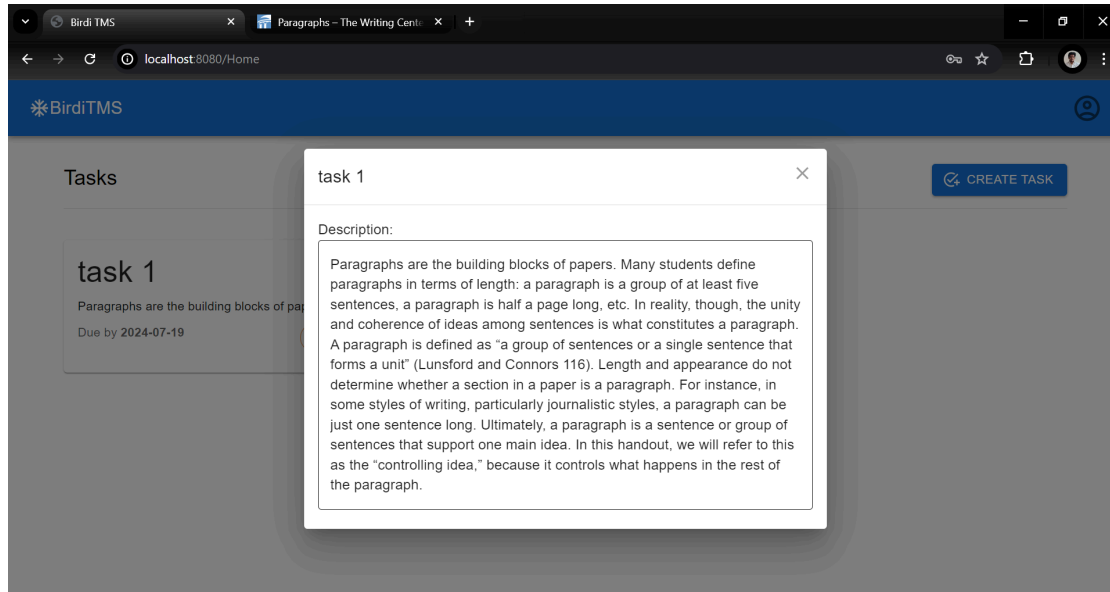
On hover on task you will see three options  
Click one edit pencil





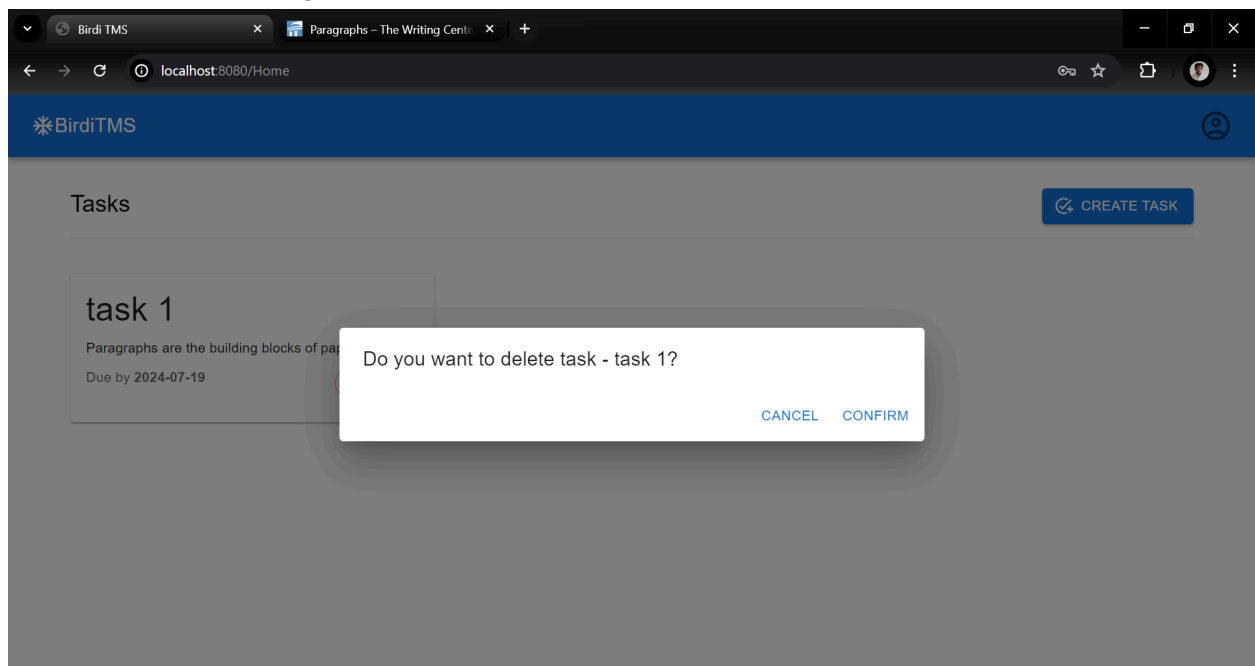
## Expand Task

If the task has big text of description then you can read by expanding it



## Delete Task

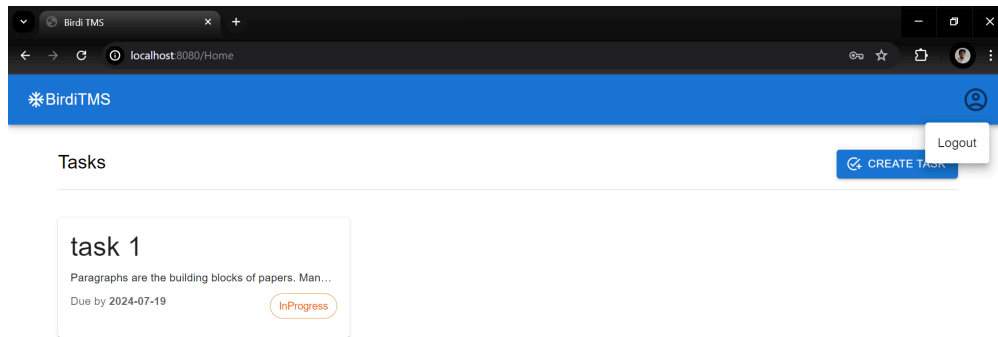
User can delete existing task



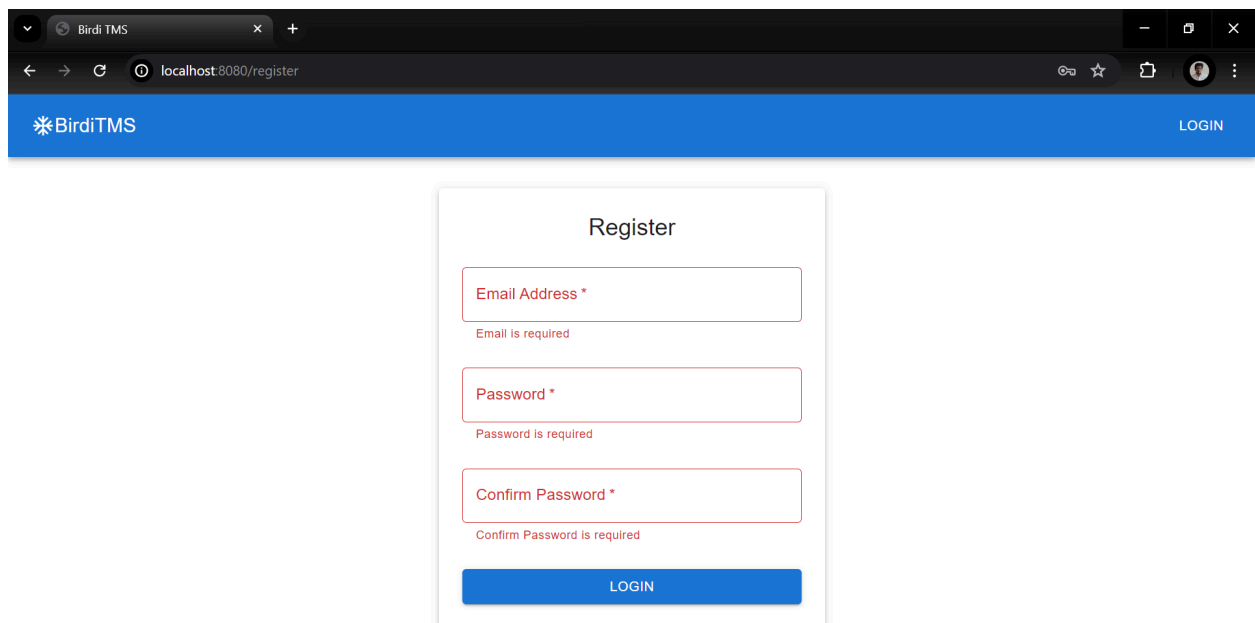
# LogOut

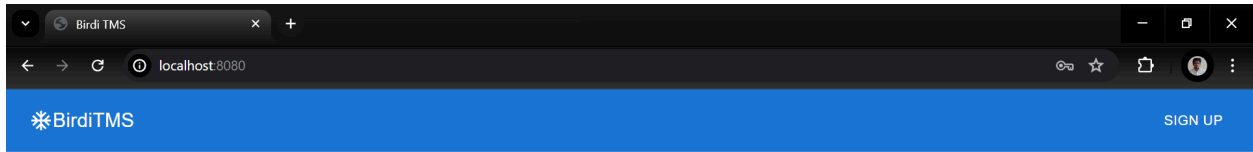
User can log out from application

Please note that you will be logged out of the system if you perform any activity after 3 minutes as the JWT token's expiration time is 3 minutes after login. However , you can change it from code.



# Error Handling



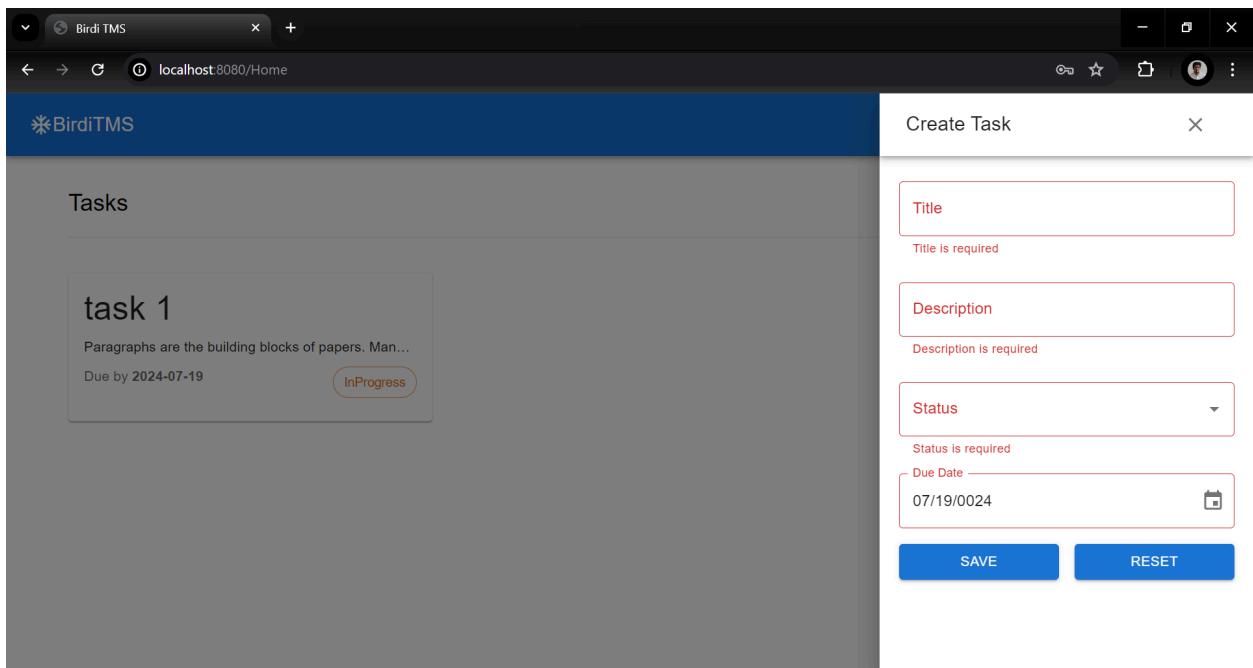


### Login

Email Address \*  
  
Email is required

Password \*  
  
Password is required

LOGIN



### Create Task

Title  
  
Title is required

Description  
  
Description is required

Status  
  
Status is required

Due Date  
  
07/19/0024

SAVE RESET



### Login

Email Address \*

Password \*

LOGIN

Incorrect username or password



### Register

Email Address \*

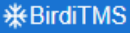
Password \*

Confirm Password \*

LOGIN

User Already Exist

## Responsive Designs

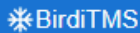

SIGN UP

### Login

Email Address \*  
srborchate1@gmail.com

Password \*  
\*\*\*\*\*

LOGIN

BirdiTMS

Tasks

CREATE TASK

backend

- create backend api - create REST API - Error handl...  
Due by 2024-07-31InProgress

front end

froontend  
Due by 2024-07-19Open

docume...

documentation  
Due by 2024-07-20InProgress

interview

round 1  
Due by 2024-07-20InProgress

Create Task

×

Title

Description

Status

▼

Due Date

07/19/2024

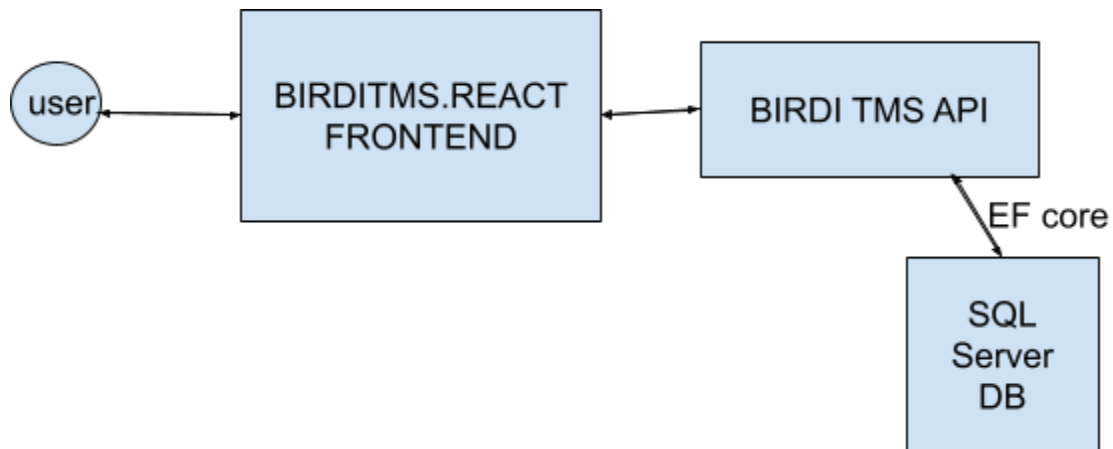
SAVE

RESET

# TECHNICAL DOCUMENTATION

In this section, we will discuss in depth about application. This software developed as microservices architecture. We will explore all parts of services. We will discuss different libraries/packages we use to enhance our application.

## Basic Architecture Design



# BIRDITMS API

This project is developed using Visual studio 2022 on the latest .NET 8 framework.

## Architecture Pattern

### Repository Pattern

We have used a repository pattern which consists of the following parts.

1. API Controllers : It contains endpoints which are consumed by clients.
2. Services : It contains business logic which is not exposed to clients and only consumed by controllers using dependency injection.
3. Repository : It contains logic to connect to a database using the EF Core framework.

### Dependency Injection

We have used dependency injection to make a loosely coupled architecture of the project. We have passed dependency through constructors and registered services in program.cs class. We have used various interfaces for services and repositories.

## Security

### ASP.NET IDENTITY Provider

We are using the inbuilt Microsoft authentication mechanism for authenticating and managing users.

### JWT Authorization

We have used the JWT Bearer package for authorization from Nuget to make our application secure.

We are also using authorization filter at the controller level.

**Please note that we have set 3 minutes of JWT token expiration time as default.**



## Optimization

### Automapper

We have used the Automapper nuget package to optimize our entity to viewmodel conversion.

## Integration

### EF Core - Code First Approach

We have integrated with SQL server using EF core.

We are using a code first approach to create Databases and entities.

## Error Handling

### GlobalException Middleware

### NLog

We are using global exception middleware to catch errors without try catch block.

We are also using the NLog package to log information about user flow.

## BIRDITMS.REACT

This project is developed using the latest Reactjs version 18 and Nodejs 20.

We are using typescript templates to make sure we are following correct variable types to avoid errors.

## Architecture Pattern

We are having different folders which include different components, styles and routing.

### Components

All components are created as react functional component.

We have different components under different folders which will make code maintainable. We have folders such as Dashboard, Common,layout.

## Redux Saga

We are using Redux Saga as middleware to dispatch actions and store information.

## Security

### Protected Route

For security we have defined a protected route which will only be accessible once the user is authenticated.

### SessionStorage JWT

We are storing JWT token in storage which will be removed from the session once the user logs out or closes the browser.

## Integration

### Axios

We are using the Axios library to call api from react client. We have also defined axios interceptors which will intercept request to check JWT Validation.

## Error Handling

### TypeScript

We are using typescript templates which make sure we are using correct types.

### ESLint

We are using the npm package to lint our reactjs project and check errors.

## Optimization

### Webpack

We are using webpack configuration to make our project optimized. We are using ts loader, babel loader.

### Prettier

We are using a prettier npm package to beautify code before starting the application. This ensures code consistency.

## Styling

### MUI V5

We are using MUI 5 package for styling to our project. This project is totally based MUI V5 layout. We are using same theme across all components to keep consistency.

## Deployment

### Docker

We are using docker-compose to deploy applications to docker.

Please check dockerfile for .net core API and reactjs application.

We advise you to run an application using docker.

## Contact

If you feel any difficulties while running the project please contact me.

Tushar Borchate  
+91 8369452655  
srborchate@gmail.com