SwipeIO

SwipeIO is a web application which helps the Administrators and Managers of an organization to know about the effective working hours of their employees. This application helps them to keep track of their employees. The effective working hours is calculated based on the “In” time and “Out” time of the employees in a day.

Screens for Admin**:**

* Login
* Dashboard
* Report
* Add/Remove
* Settings
* Import

Screens for Employee**:**

* Dashboard
* Report

Screens for Admin**:**

# **Login**:

* Invoke the SwipeIO login page. After invoking login page, It should asks to enter the Email and Password of the Employee.
* The Login can be done by Employees which is categorized as Admins and Contracts.
* After the successful login the middleware should generate JWT token and send it to the UI and the UI must store the details of logged in employee in a local storage except password.

**API:** <http://localhost:4200/Employee/authenticate>

Request payload:

{

email:

pass\_word:

}

Response:   
{

Status: 200;

Data : {

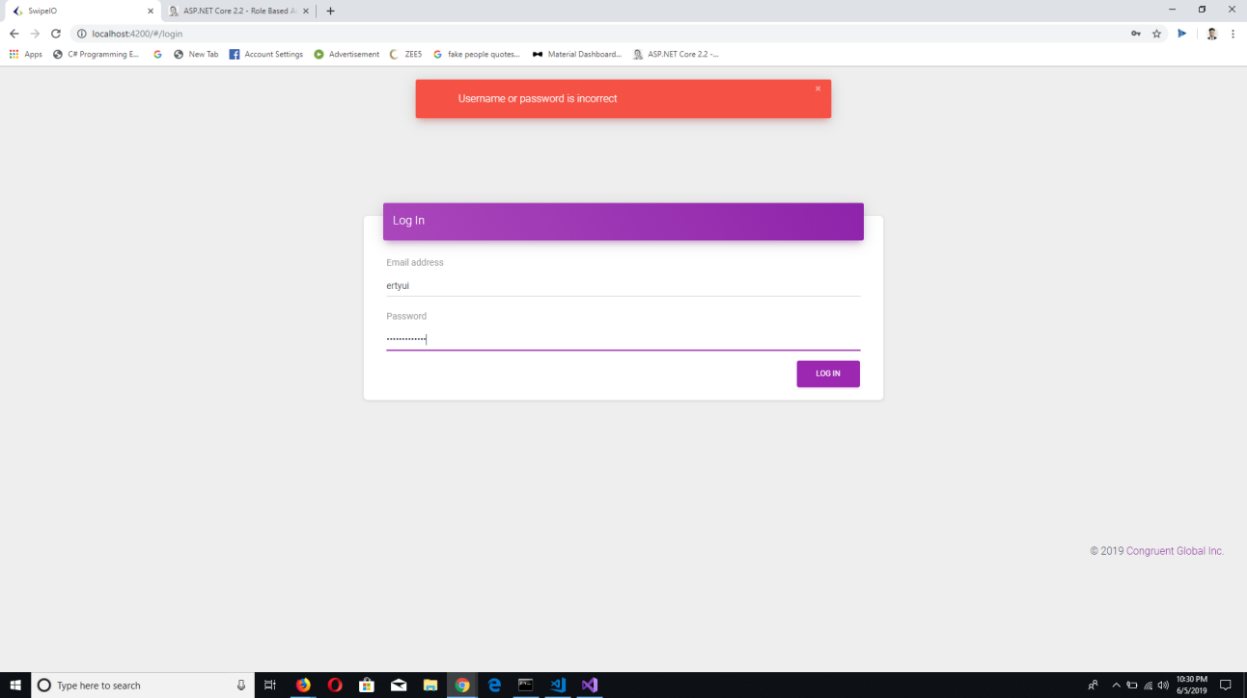
emp\_id, email, emp\_name, emp\_number, is\_admin, card\_id, Token

}

}

**Front end:**

* Use the above api to validate the email id and password provided by the user
* If the user is validated successfully then status will be ‘ok’ with data as employee Id otherwise status as ‘bad’ with error code in the data.

****

**Back end:**

* Implement the API
* Validate the login by passing details to the stored procedure
* Gets the date from the stored procedure and check against the email and password match
* Based on the response of the SP, check if both the email and password match with response of SP, send the status as ‘ok’ and data as emp\_id.
* Validate for the email and password and send the respective error code , refer Error code and error messages and send status as ‘bad’

**Database end:**

* Create a stored procedure with the following details

**SP Name**: Validate

**Params**:

email1– string

pass\_word1 – string

**Output**:

Rows of matches

# Error code and error messages

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Sub category** | **Error Code** | **Error Message** |
| Login |  | L01 | Email or password is incorrect |

# Dashboard:

* In the Home page of Admin it shows details of all Employees:
  + Number of Employees
  + Filter
  + The last five days reports of all employees

**API:** http://localhost:4200/Employee

Request payload:

{

}

Response:   
{

Status: 200;

Data : {

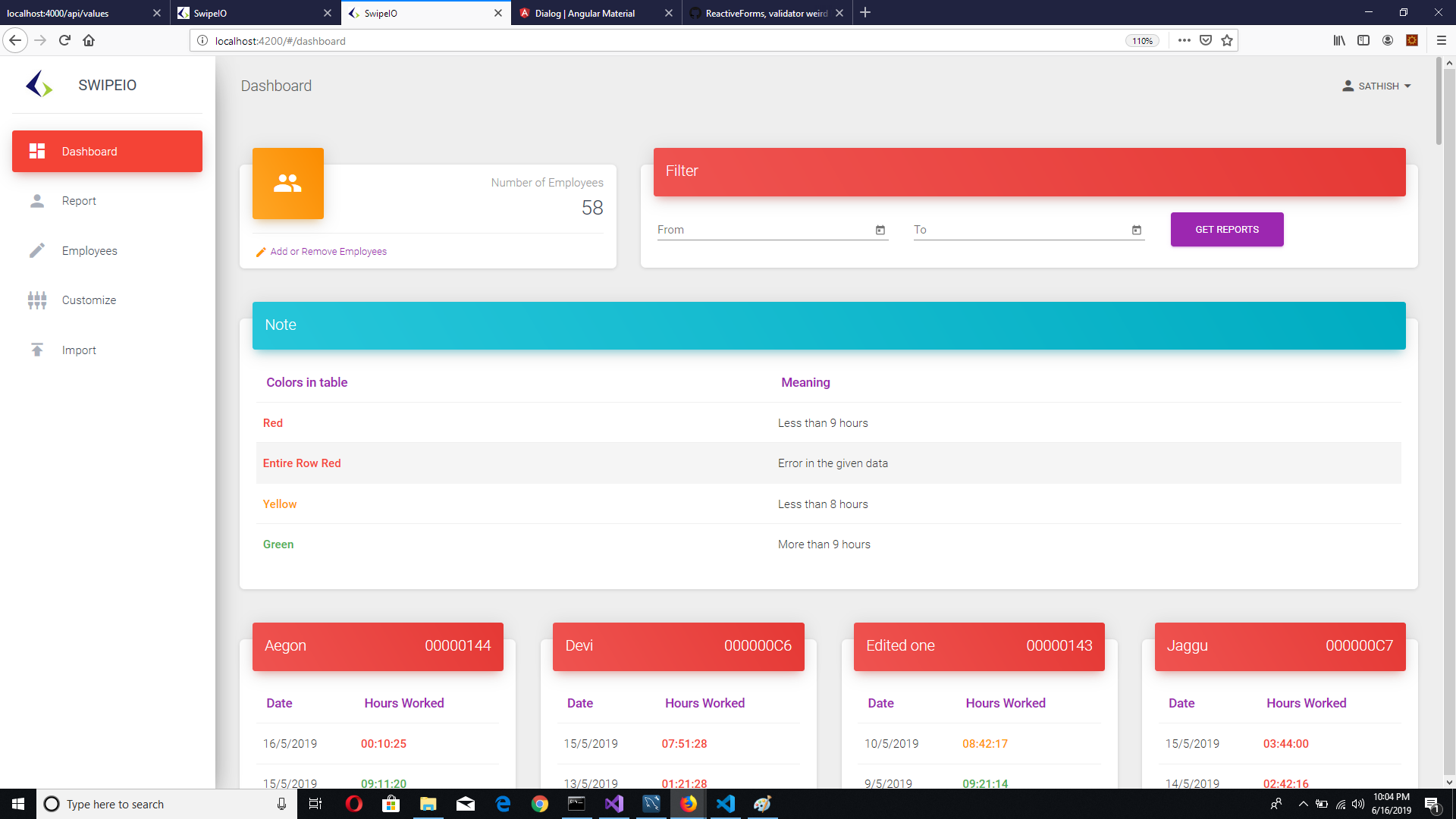
Emp\_id, report[]

}

}

**Front end:**

* Use the above API and display the last five days reports of all the employees



**Back end:**

* Implement the API

**Database end:**

* Create a stored procedure with the following details

**SP Name**: get\_last\_records

**Params**: emp\_id, gate\_id

**Output**: last 5 dates of employee swipe log

**SP Name**: get\_records

**Params**: emp\_id, gate\_id, date

**Output**: swipe logs in that date

# REPORT:

* In the Report Page of admin :
* Filter (From, To, Employee, Gate)
* Report

**API:** <http://localhost:4200/>Report/get\_report

Request payload:

{

Employee Id:

From:

To:

Gate\_id

}

Response:  
{

Date:

In:

Out:

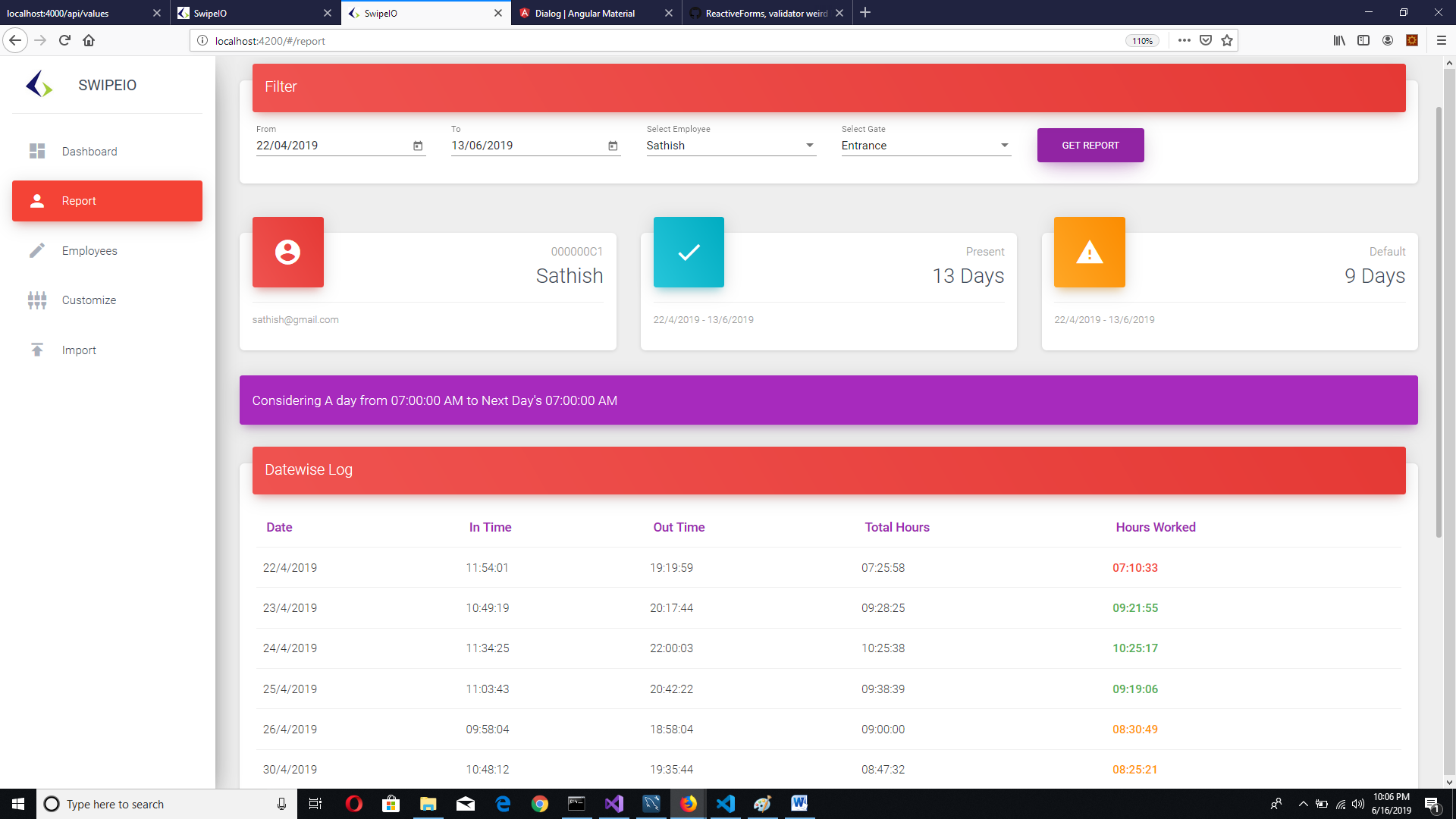
Hours inside:

Hours Worked

}

**Front end:**

* In the front-end should get the “from” and “to” dates from the admin and send request to the API.
* When it gets the response from the API it should display the report and the report should display the details of the employee with employee name, id, email, number of days present and number of default days.
* It should also display date wise log and it should contain date ,in time, out time, hours inside and hours worked.



**Back end:**

* Implement the API which gets the from date and to date as a request and it must calculate the report and it must return date, in time, out time, hours inside and hours worked as a response.

**Database end:**

* Create a stored procedure with the following details

**SP Name**: get\_date

**Params**: in emp\_id1,in from\_date ,in to\_date

**Output**: dates

**SP Name**:, get\_swipe\_log

**Params**: in emp\_id1,in date1

**Output**: logs.

# ADD/REMOVE:

* In Add/Remove the admin can add Employees and can do changes like:
* View
* Edit
* Remove
* To Add a New Employee:
* Employee Name, Employee ID, Email Address, Password, Card Id, Isadmin and iscontract should be filled.

**API:** http://localhost:4200/Employee/add

Request payload:

{

Employee Name:

Employee Id:

Card Id:

Email Address:

Password:

Isadmin:

Iscontract:

Incharge list:

}

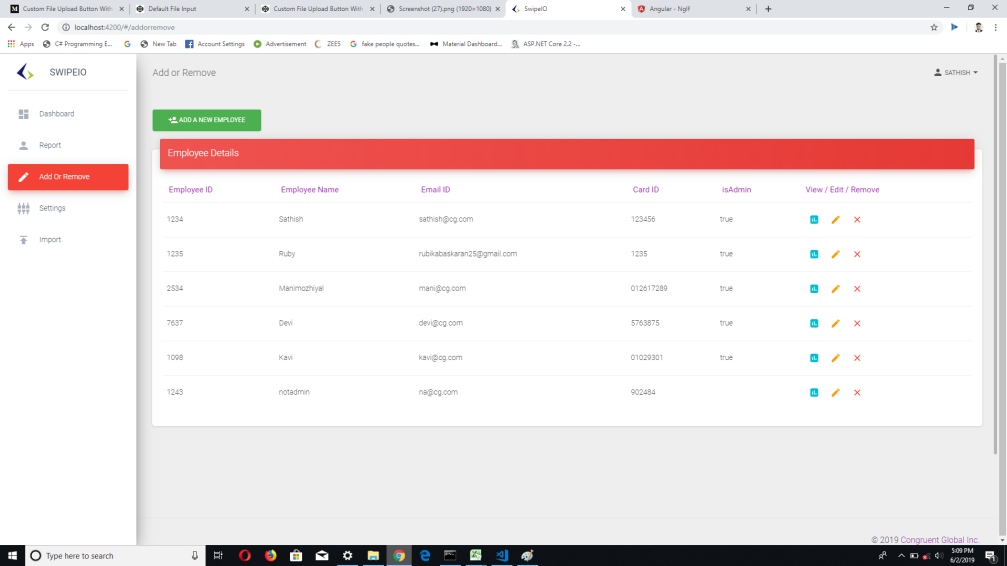
Response:  
{

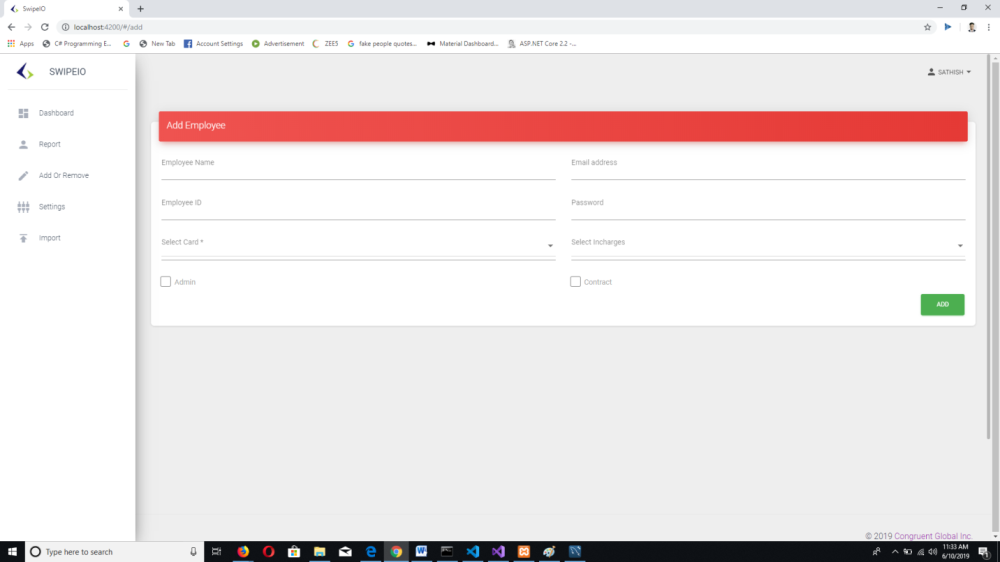
status:200

}

**Front end:**

* The front-end must get the details of the employee as an input.



****

**Back end:**

* Implement the API which gets the details of the employee as a request an call the stored procedure and return the status.

**Database end:**

* Create a stored procedure with the following details

**SP Name** : insert\_employee

**Params**: in emp\_number1 , in emp\_name in email1, in pass\_word1 ,in is\_admin1,in is\_contract1, in card\_id1

**Output**: Acknowledgement.

# Customize:

* In Settings, Gate, Card Details and leave details will be specified of employees.
* And do changes like Edit/Remove of the gate or card details.

**API:** http://localhost:4200/Settings

Request payload:

{

Get\_cards:

Get\_leaves:

Get\_gates:

}

Response:  
{

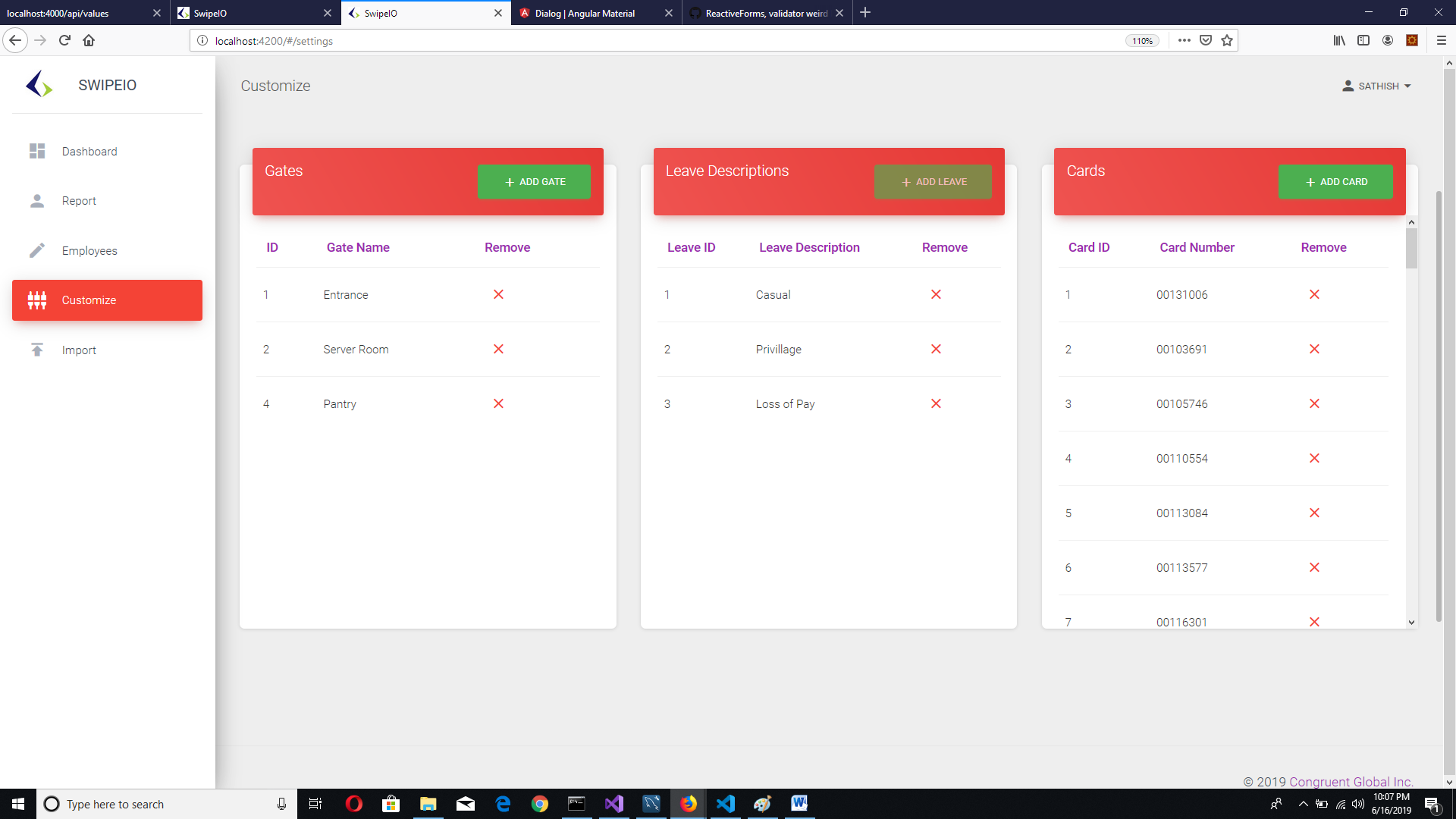
Gates,cards,leaves

status:200

}

**Front end:**

* It should displays gates ,cards, leaves, add and remove buttons



**Back end:**

* Implement the API which gets the request from the UI and send the gate, leave, card details as a response.

**Database end:**

* Create a stored procedure with the following details

**SP Name** :get\_gates,get\_leaves,get\_cards

**Params**:

**Output**: gates, leaves and cards.

# IMPORT:

* In Import screen the admin should upload XLSX File of the employees, based on that they can update the number of hours worked(calculated on IN/OUT) Timings.

**API:**<http://localhost:4200/>Log/upload

Request payload:

{

Date:

Time:

Cardid:

Empid:

EmpName:

Department:

Type:

CID:

Gate:

InOut:

Remark:

}

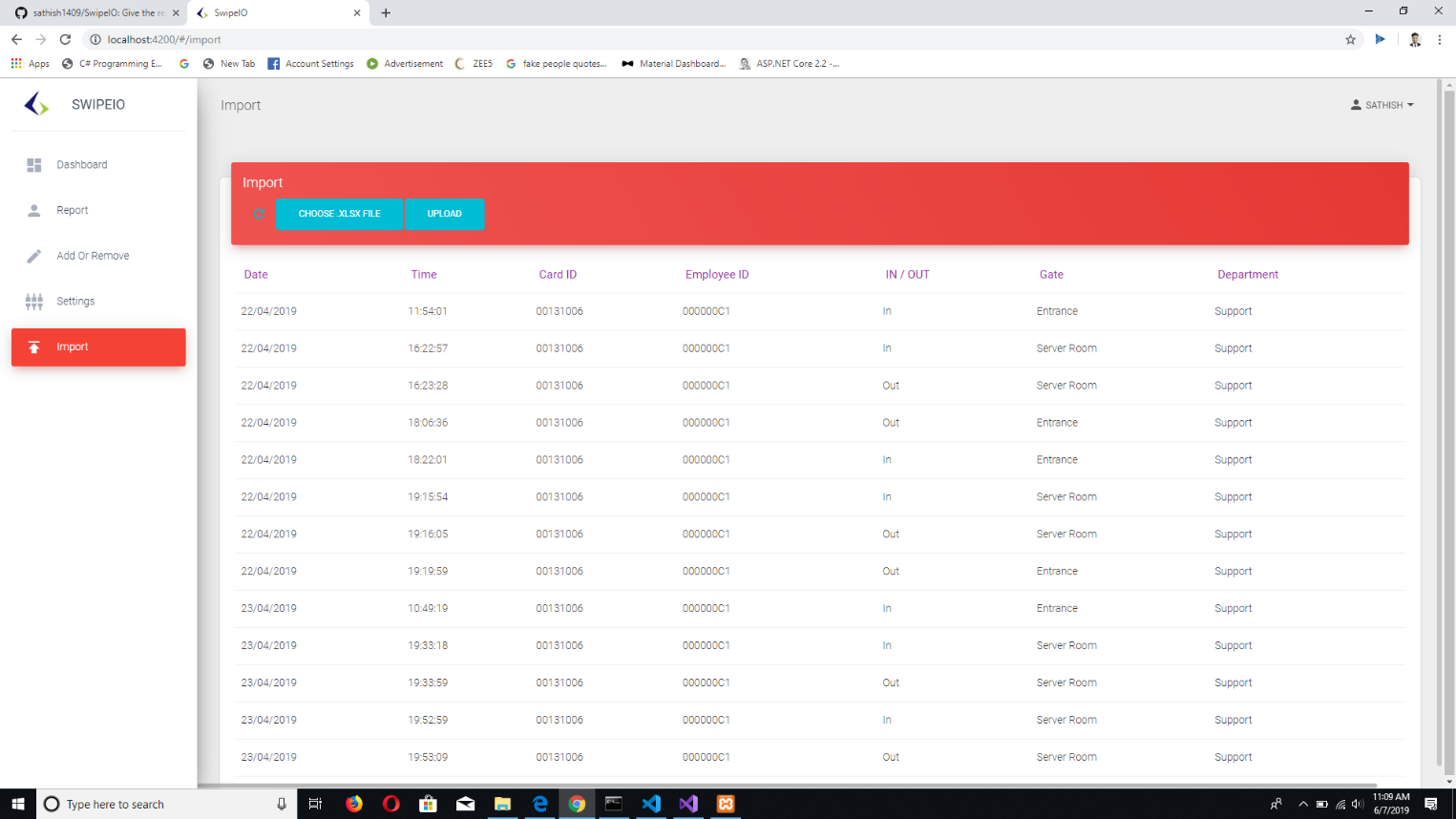
Response:  
{

status:200

}

**Front end:**

* The front-end must have XLSX file when the user clicks that button they must be able to choose the excel file which has log of swipes.
* The front -end should show the contents of the chosen excel file.



**Back end:**

* Implement the API
* Get the log details from front-end and iterate every data and for every iteration call the stored procedure import \_to\_swipe

**Database end:**

* Create a stored procedure with the following details

**SP Name** :import\_to\_swipe

**Params**:

date1, time1, card\_number1, emp\_number1, gate\_name1, inorout1, in remark1.

**Output**:

Acknowledgement.

# Screens for Employees:

# Dashboard:

* In the Home page of Admin it shows details of all reporting Employees under him:
  + Number of Reporting Employees
  + Filter
  + The last five days reports of all employees

**API:** http://localhost:4200/Employee

Request payload:

{

}

Response:   
{

Status: 200;

Data : {

Emp\_id, report[]

}

}

**Front end:**

* Use the above API and display the last five days reports of all the employees

**Back end:**

* Implement the API

**Database end:**

* Create a stored procedure with the following details

**SP Name**: get\_last\_records

**Params**: emp\_id, gate\_id

**Output**: last 5 dates of reporting employee swipe log

**SP Name**: get\_records

**Params**: emp\_id, gate\_id, date

**Output**: swipe logs in that date

# REPORT:

* In the Report Page of admin :
* Filter (From, To, Employee, Gate)
* Report

**API:** <http://localhost:4200/>Report/get\_report

Request payload:

{

Employee Id:

From:

To:

Gate\_id

}

Response:  
{

Date:

In:

Out:

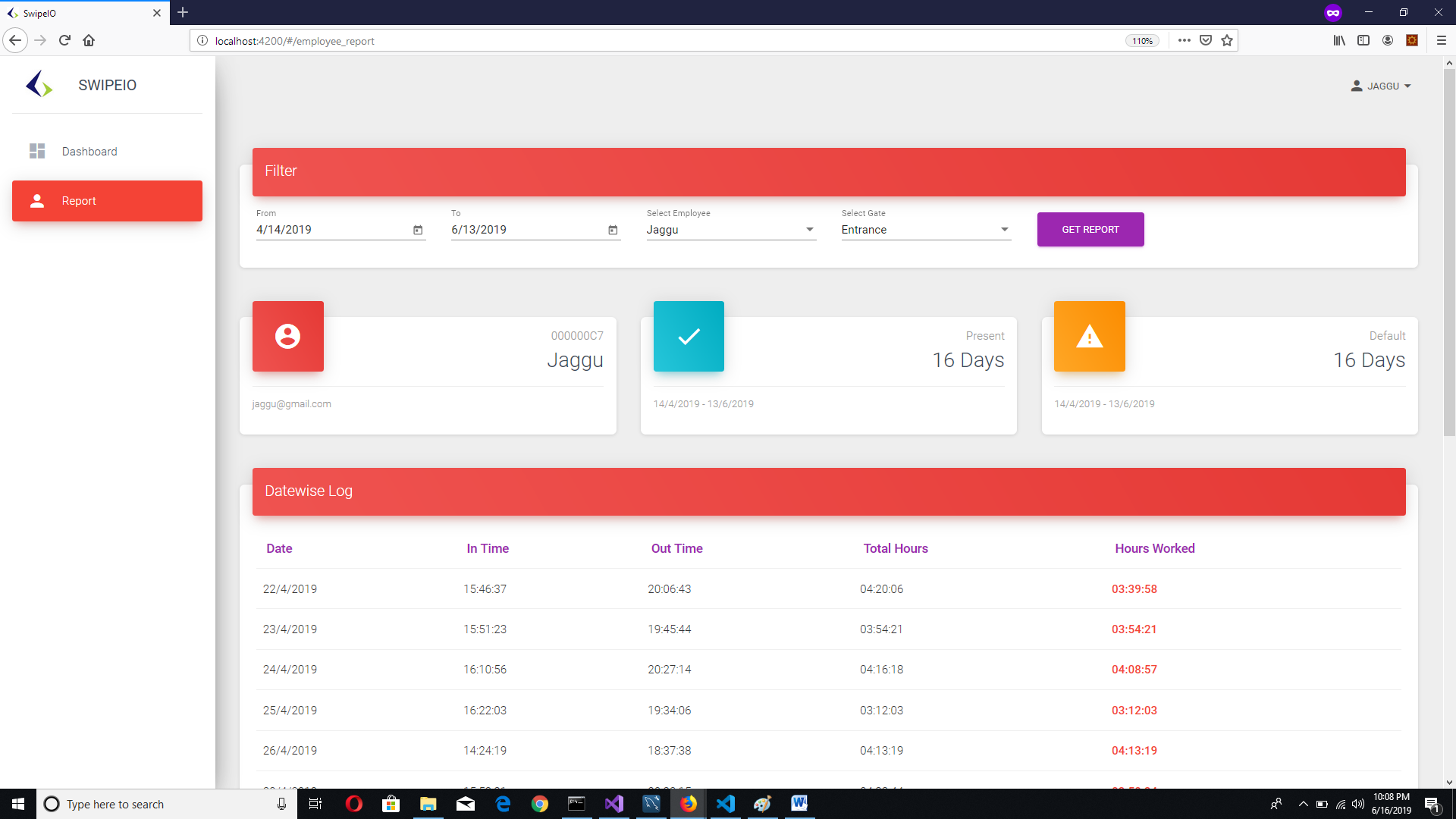
Hours inside:

Hours Worked

}

**Front end:**

* In the front-end should get the “from” and “to” dates from the admin and send request to the API.
* When it gets the response from the API it should display the report and the report should display the details of the employee with employee name, id, email, number of days present and number of default days.
* It should also display date wise log and it should contain date ,in time, out time, hours inside and hours worked.



**Back end:**

* Implement the API which gets the from date and to date as a request and it must calculate the report and it must return date, in time, out time, hours inside and hours worked as a response.

**Database end:**

* Create a stored procedure with the following details

**SP Name**: get\_date

**Params**: in emp\_id1,in from\_date ,in to\_date

**Output**: dates

**SP Name**:, get\_swipe\_log

**Params**: in emp\_id1,in date1

**Output**: logs.