**LOAN STATUS APP DEPLOYMENT GUIDE**

**BY**

**HEDWIG ORIEBA**

**(**oriehedwig@gmail.com**)**

Contents

[**Overview** 3](#_Toc132114921)

[**High Level Integration Component-tiers & Process Flow** 3](#_Toc132114922)

[**Process Flow Description** 4](#_Toc132114923)

[**Integration Components** 4](#_Toc132114924)

[**Integration Component Specifications** 5](#_Toc132114925)

[Windows Client Machine 5](#_Toc132114926)

[Hardware: 5](#_Toc132114927)

[Software: 5](#_Toc132114928)

[Mongo DB 5](#_Toc132114929)

[Hardware: 5](#_Toc132114930)

[Software: 5](#_Toc132114931)

[Mongo Compass 5](#_Toc132114932)

[Hardware: 5](#_Toc132114933)

[Software: 5](#_Toc132114934)

[**DEPLOYMENT WORK BREAK DOWN STRUCTURE (WBS)** 6](#_Toc132114935)

[Nodejs & mongodb 6](#_Toc132114936)

[IIS Service Setup 7](#_Toc132114937)

[IIS Manager 8](#_Toc132114938)

[RUNNNING & CLONING NODEJS 10](#_Toc132114939)

[DIRECT APPLICATION ACCESS 11](#_Toc132114940)

[Access the application as an admin: 11](#_Toc132114941)

[Admin view total request 12](#_Toc132114942)

[Admin create a user 13](#_Toc132114943)

[Access the application as a common user: 16](#_Toc132114944)

[**INTEGRATED APPLICATION ACCESS** 19](#_Toc132114945)

[END 21](#_Toc132114946)

# **Overview**

This document provides the technical synopsis of the deployment steps and activities needed to successfully deploy the DFCU Loan Status tracking API service. Once successfully deployed, the service shall expose the following set of functionalities:

1. Realtime system to system exchange of data through invocation of respective service resources.
2. Authenticated (Right to identity) access service resources.
3. Authorized (Right to action) access to system resources.
4. Logging/persistence of resource requests to a NoSQL document type database (Mongo DB).
5. Highly decoupled process peering to achieve fast, reliable inter-process exchange of data. The service to this effect is peered with an ASP. Net request logging process/application.

# **High Level Integration Component-tiers & Process Flow**

The API service is implemented on a two-layered design (logical divisions of responsibility) scheme that manifests as two tiers (physical divisions) of physical hardware and software deployment. The logical deployment is as follows:

Diagram

Description automatically generated

# **Process Flow Description**

A client agent/application (google chrome / an ASP. Net app.) makes an HTTP GET or POST Request to the integration/data tier.

1. The integration/data tier uses its controller logic in accordance with the application controller design pattern to route the request to a respective handler logic in alignment to the business delegate design pattern.
2. The route handler within the integration layer business section actions on the request immediately with a response or makes a call to the data persistence layer if required. This returns a transfer object known as a response.
3. The integration tier returns the transfer object ack to the client as a request response.
4. The client then either consumes the result set or logs it as per the process.

# **Integration Components**

1. Windows Client machine with the IIS system feature enabled. This will be used to host the Nodejs RESTful service & the ASP. Net application.
2. Mongo DB
3. Mongo Compass

# **Integration Component Specifications**

## Windows Client Machine

### Hardware:

Windows 10 64bit PC, 8GB RAM, 500 GB Hard Disk space with working network adapter for Internet Access.

### Software:

Nodejs.

IIS Server. The server should have Windows Authentication capability enabled & should transact over the default http port: 80

## Mongo DB

### Hardware:

Windows 10 64bit PC, 8GB RAM, 500 GB Hard Disk space with working network adapter for Internet Access.

### Software:

Mongodb database version 4.4.4

## Mongo Compass

### Hardware:

Windows 10 64bit PC, 8GB RAM, 500 GB Hard Disk space with working network adapter for Internet Access.

### Software:

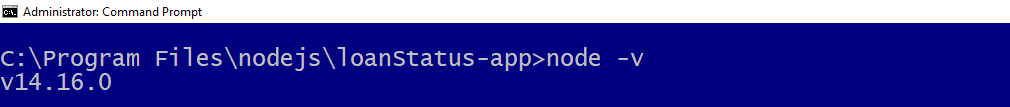
Deployed & running Mongodb database version 4.4.4

Mongo compass version 1.35.0 or higher.

# **DEPLOYMENT WORK BREAK DOWN STRUCTURE (WBS)**

## Nodejs & mongodb

1. Download and install Nodejs for windows from <https://nodejs.org/en/download>.
2. Verify successful node installation executing the following command at the CLI.



**Input:** node -v

**Output:** v14.16.0

1. Download the mongo database community server for windows from <https://www.mongodb.com/try/download/community>.
2. Add the directory path of the installed mongodb to your machine’s path system environment variable. The path should look like: C:\Program Files\MongoDB\Server\4.4\bin.
3. Verify the success of server installation using the command: **mongod** at CLI.

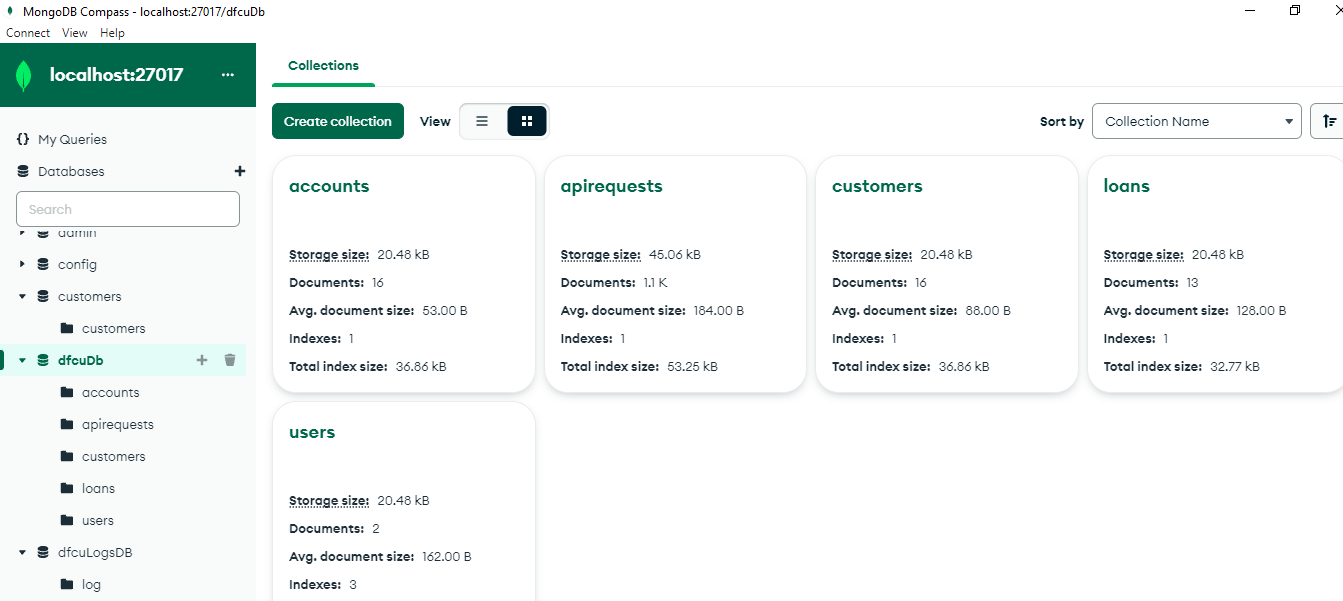
Text

Description automatically generated

**Input:** mongod

**Output**: A Json object.

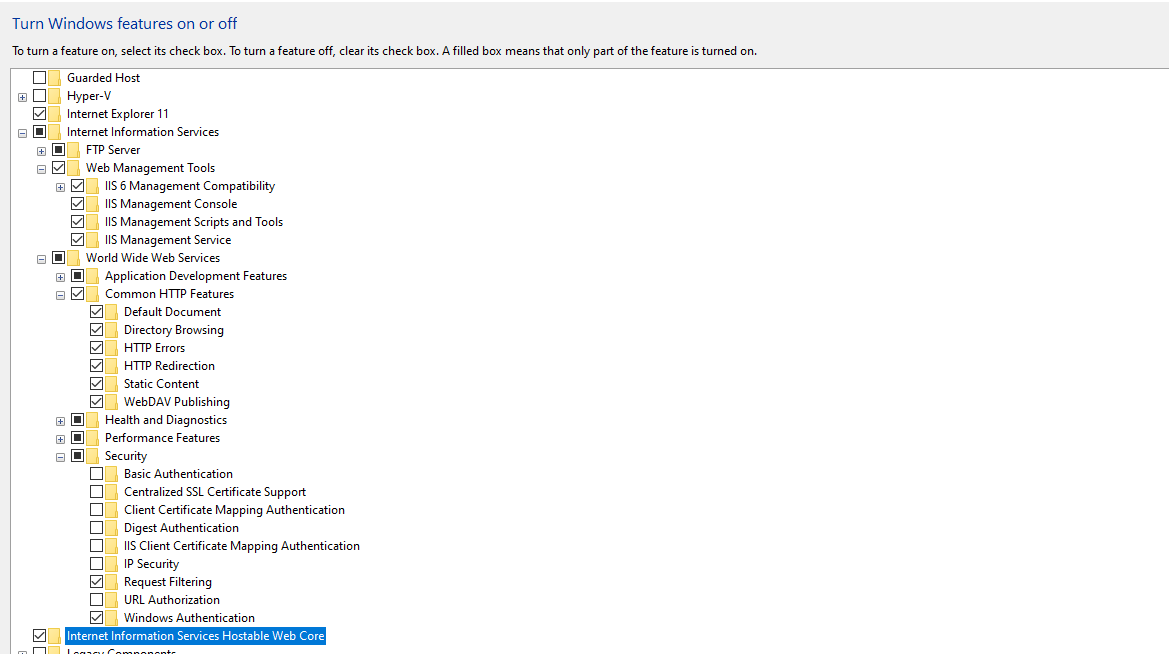
This also installs the mongo compass which is the client application that will be used to connect to the mongodb server. Use compass to import all collections shared.



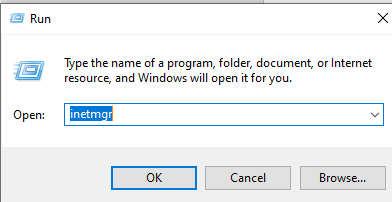
## IIS Service Setup

1. Enable IIS using the windows 10 Programs and Features of the control panel as shown below:

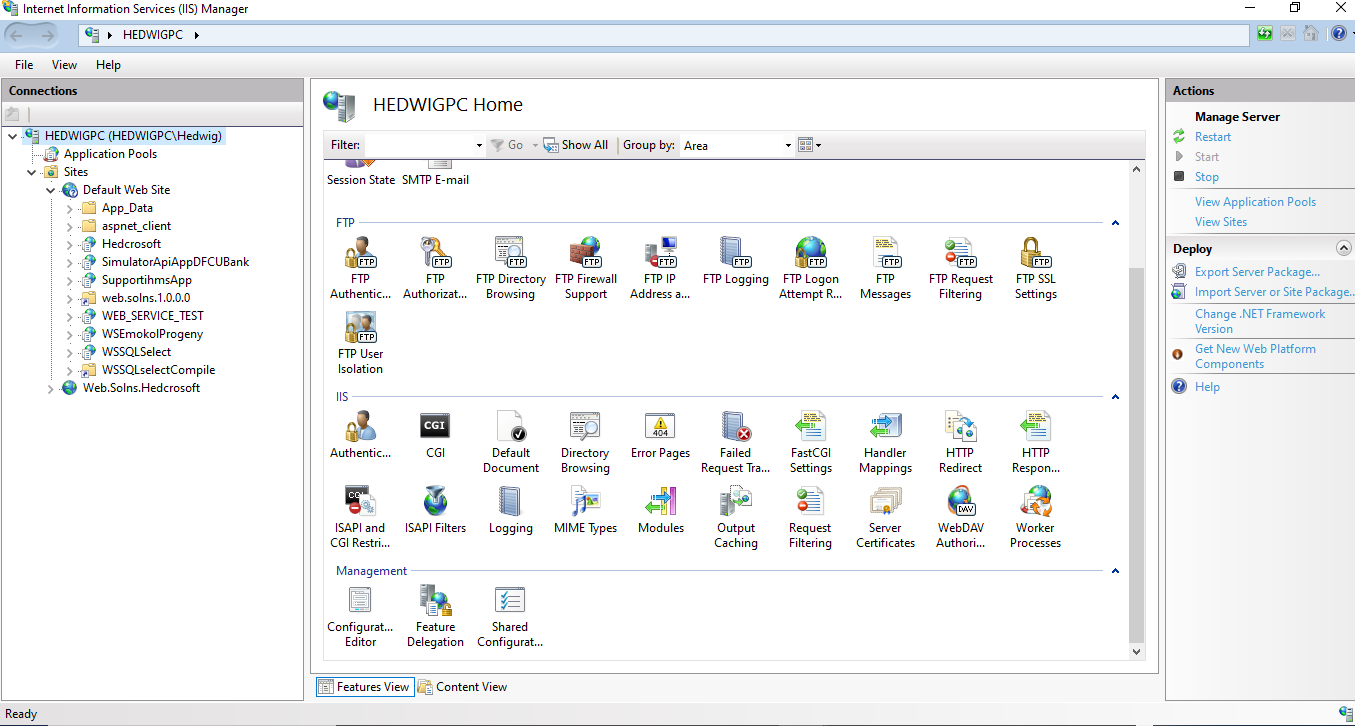
Path: (Control Panel\Programs -Turn Windows Features on or off)



Start IIS service console by typing the command inetmgr in the run window.



### IIS Manager



1. create a new directory on your C:\ drive called gitLoanSimulator using:

**mkdir** gitLoanSimulator

1. Get into the directory using **cd gitLoanSimulator**
2. Clone the Clone the ASP. Net application from GitHub repository to your machine:

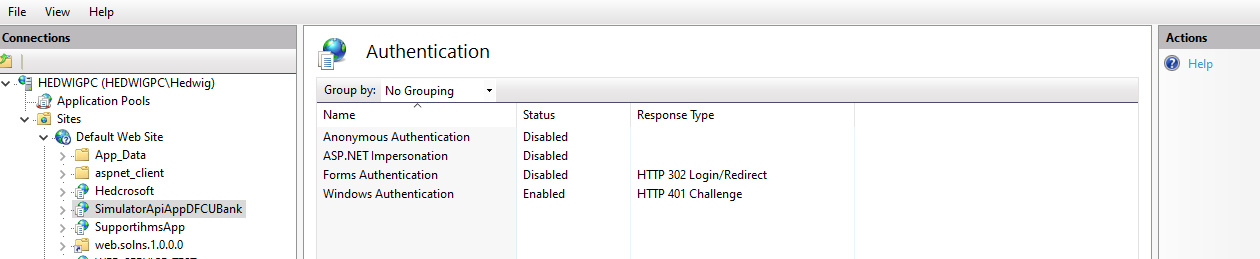
use: **git clone** <https://github.com/HedwigOrieba/loanStatusSimulator.git>

1. Move the cloned solution package to your IIS wwwroot directory for IIS hosting.

Graphical user interface, text

Description automatically generated with medium confidence

1. In IIS convert the solution to an application under the default application pool.
2. Disable anonymous access & enable windows authentication for the application.



1. Enable directory browsing for the application.

Graphical user interface, text, application

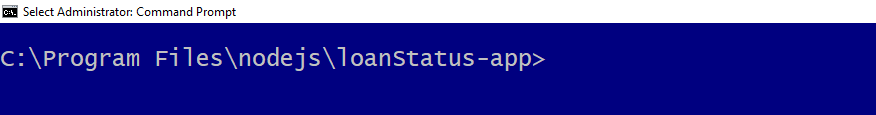
Description automatically generated

## RUNNNING & CLONING NODEJS

Clone the RESTful Nodejs application from GitHub using:

Git clone <https://github.com/HedwigOrieba/LoanStatusAppRepos.git>

Start CLI as an admin and navigate to your cloned directory.

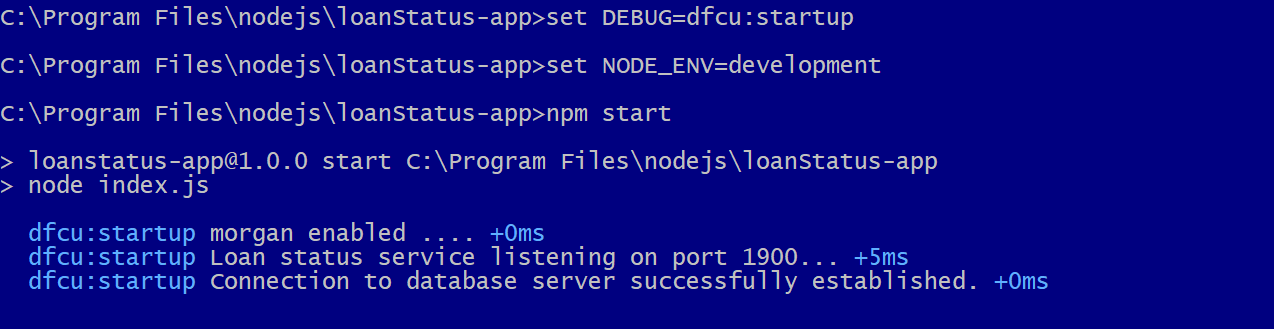


**Set the environment api variables:**

cmd: set DEBUG=dfcu:startup

cmd: set NODE\_ENV=development

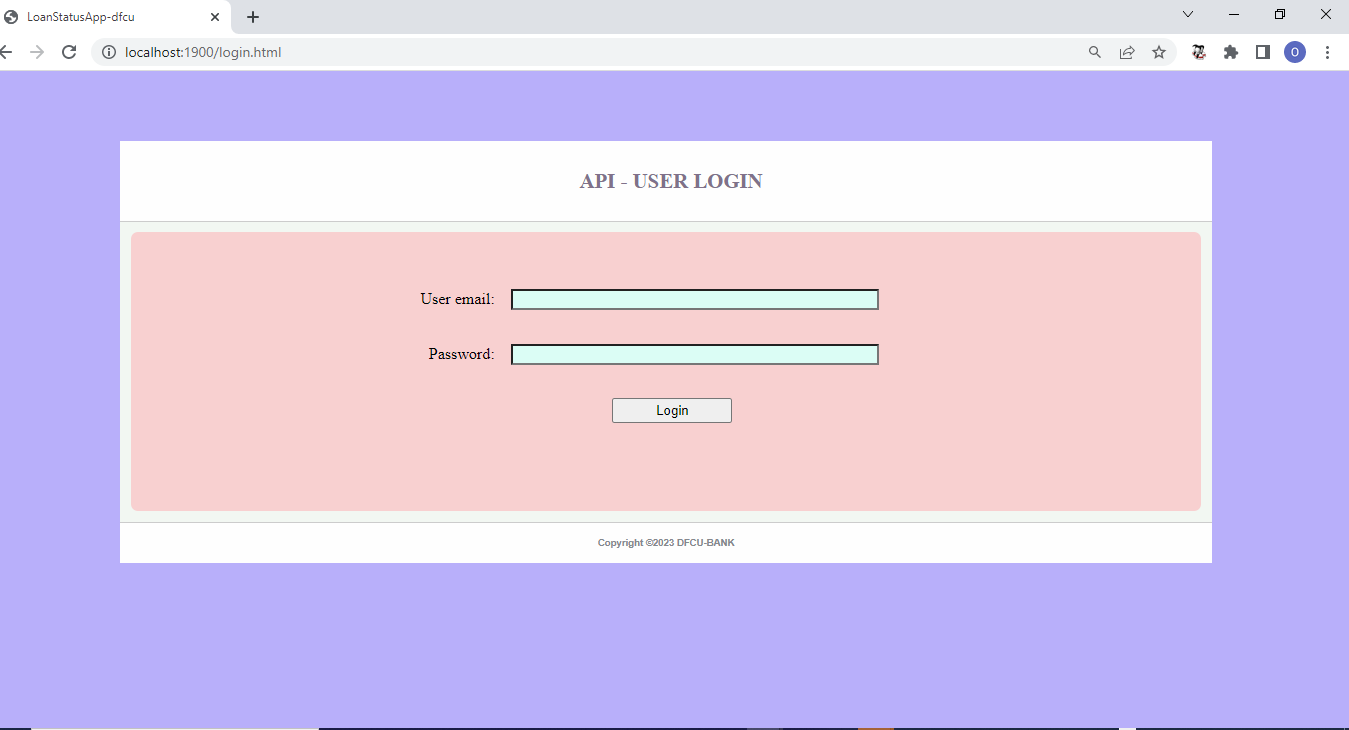
cmd: npm start



## DIRECT APPLICATION ACCESS

### Access the application as an admin:

Call the login view through the url: http://localhost:1900/login.html

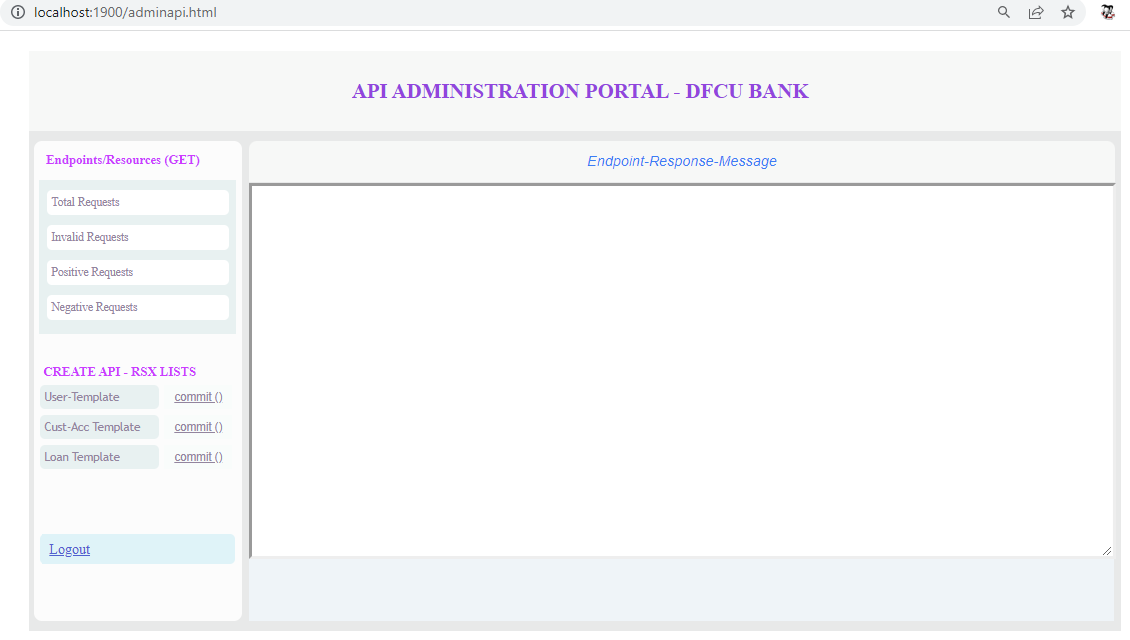


Provide the admi**n email and password as :**

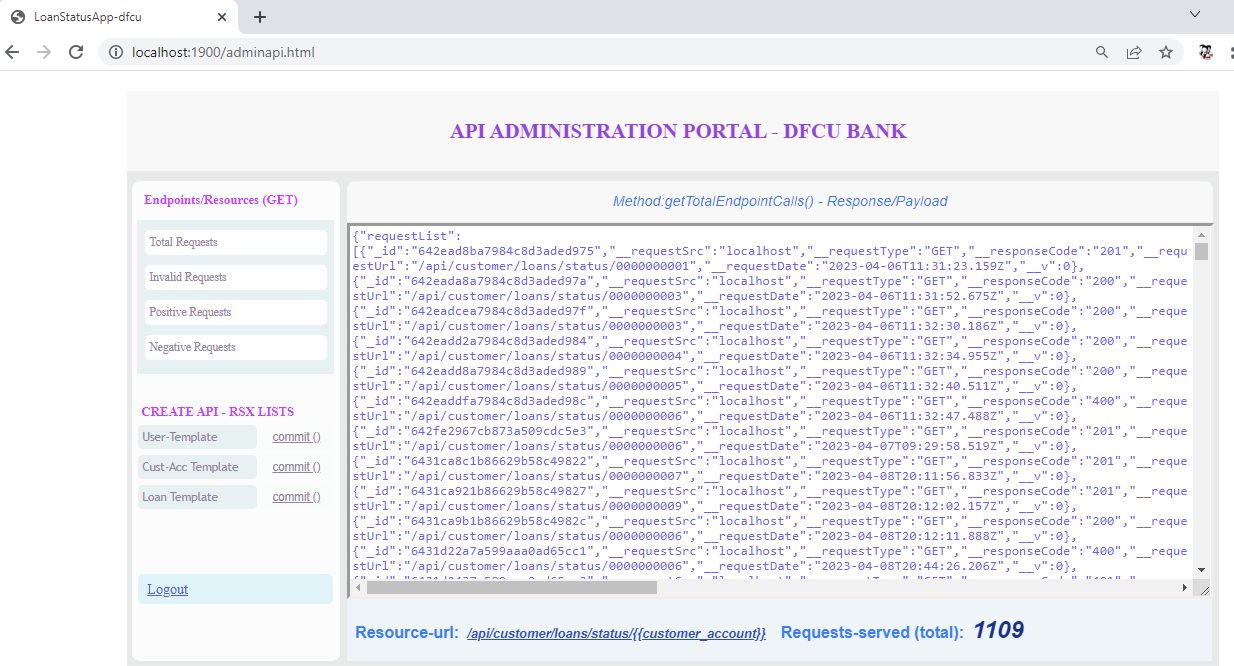
**email –** [**admin@dfcu.com**](mailto:admin@dfcu.com)

**password – 12345**

Once authenticated & authorized you will gain admin access to api where you can view total, successful, failed & invalid ones.



### Admin view total request



### Admin create a user

Step1: load the user template first.

Step2: Provide appropriate values for the template fields.

Step3: commit the template.

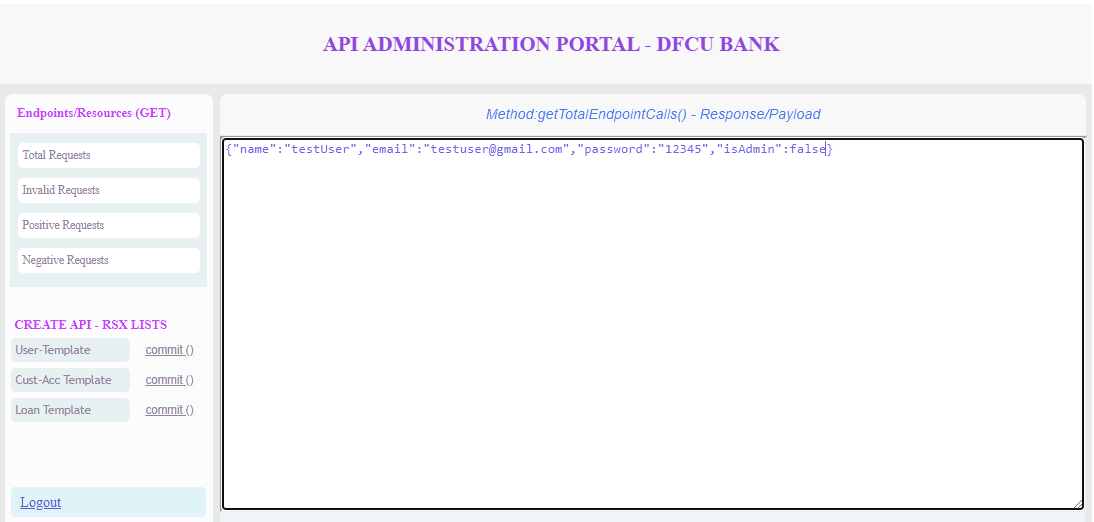
These are respectively demonstrated:

**Step1**

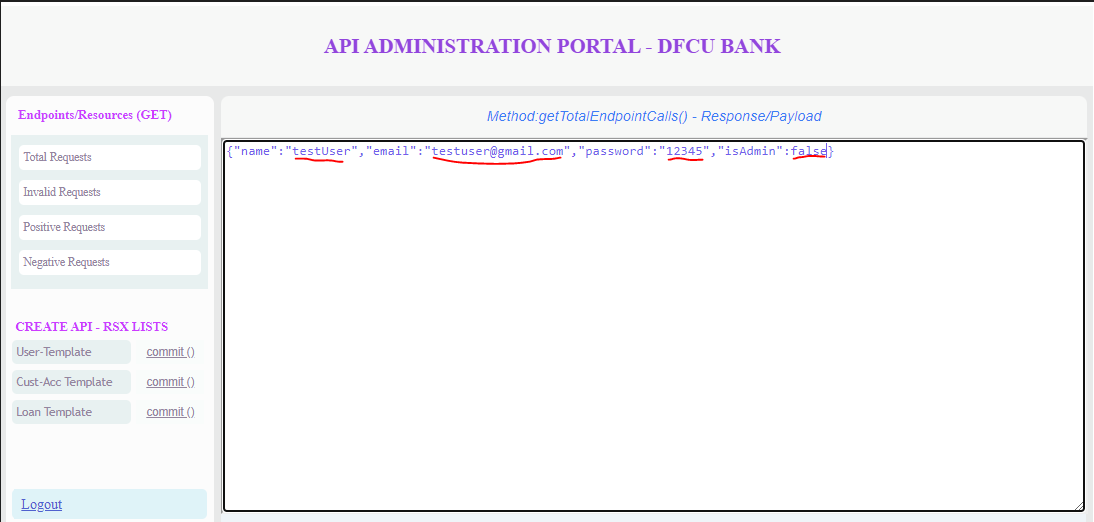
Graphical user interface, text, application

Description automatically generated

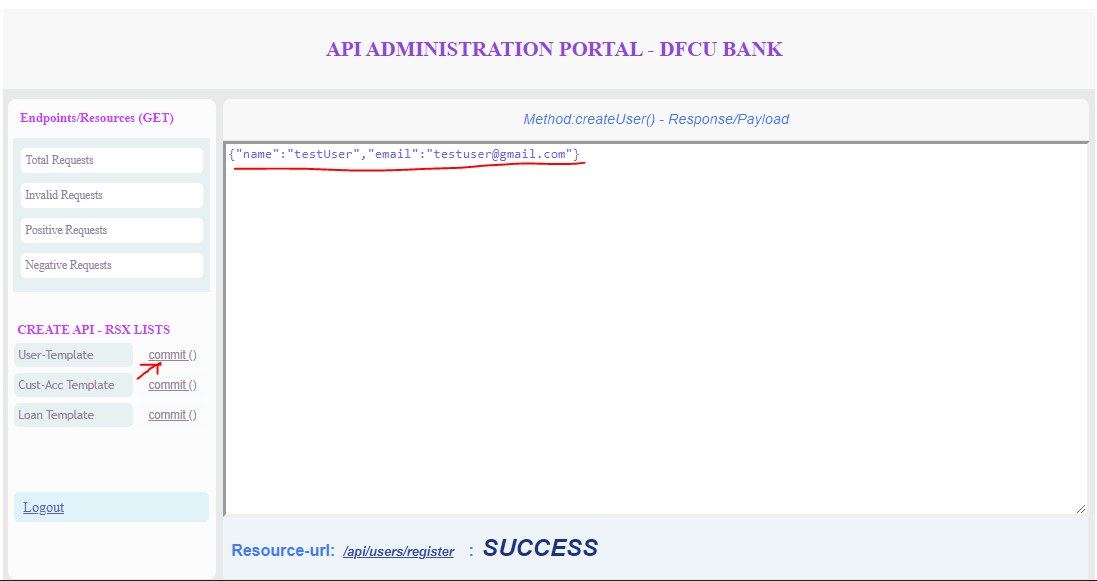
Step2:



Step3:



Step4:



This is how API -Lists are auto created .

### **Access the application as a common user**:

Provide the non admin email and password as:

**email –** [**dfcuba**nk**@gmail.com**](mailto:dfcubank@gmail.com)

**password – 123456**

Graphical user interface

Description automatically generated

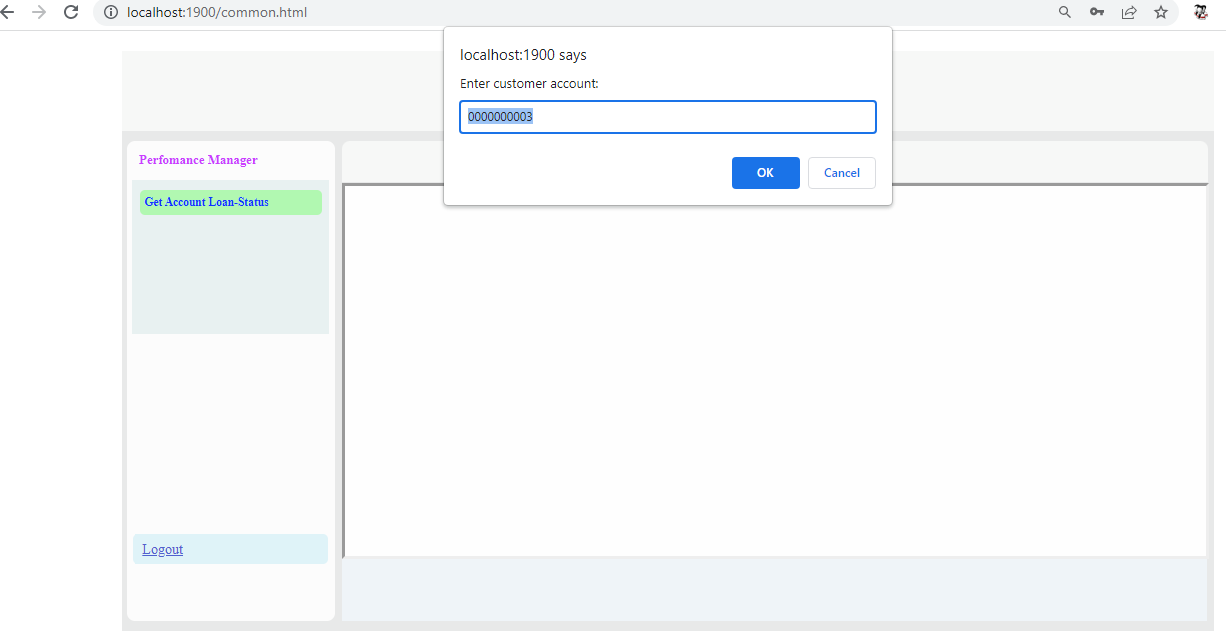
Once authenticated & authorized you will gain non-admin access to api where you can loan status

Of each customer account:

Graphical user interface, text, application

Description automatically generated

Click on Get Account Loan-Status.



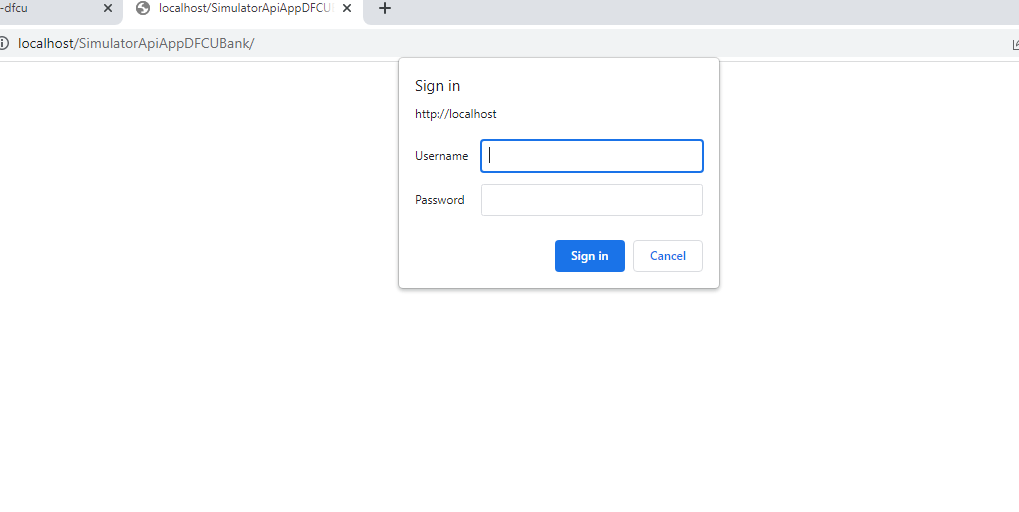
Select ok after specifying the account.

Graphical user interface, text, application

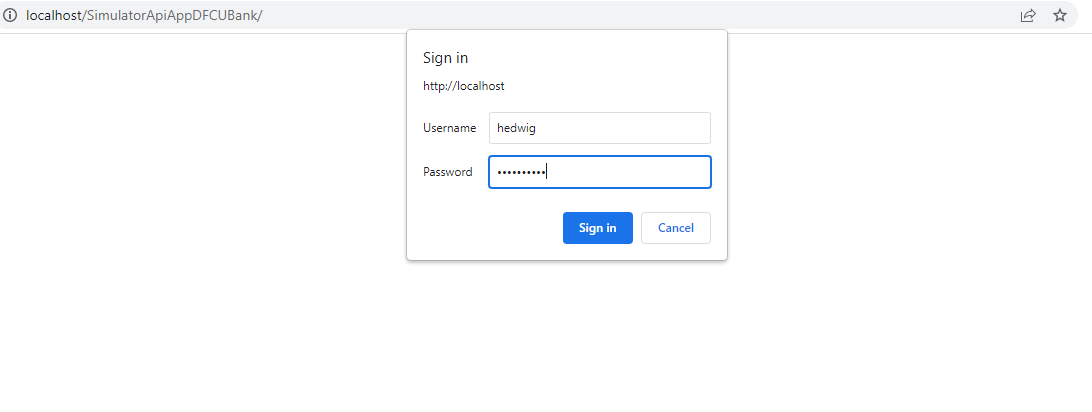
Description automatically generated

# **INTEGRATED APPLICATION ACCESS**

Here we access the application via ASP. Net hosted application as follows:



Provide your windows login credentials and sign in.



Load Random list of accounts.

Graphical user interface, text, application, email

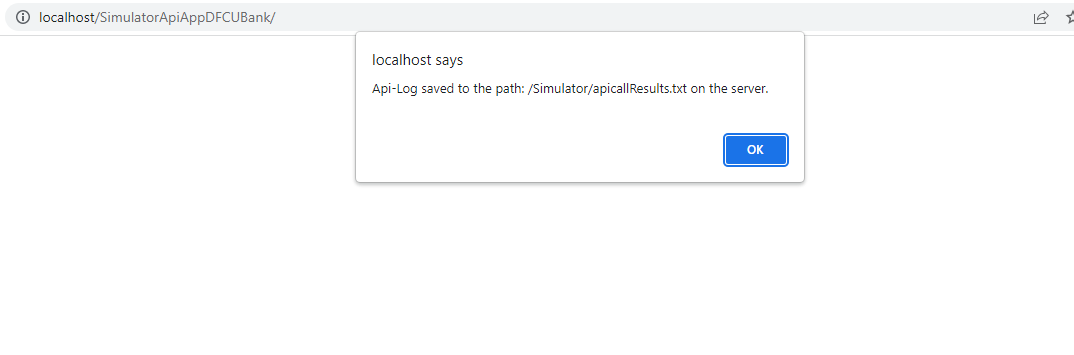
Description automatically generated

Click “Generate Random Accounts “

Graphical user interface, text, application, email

Description automatically generated

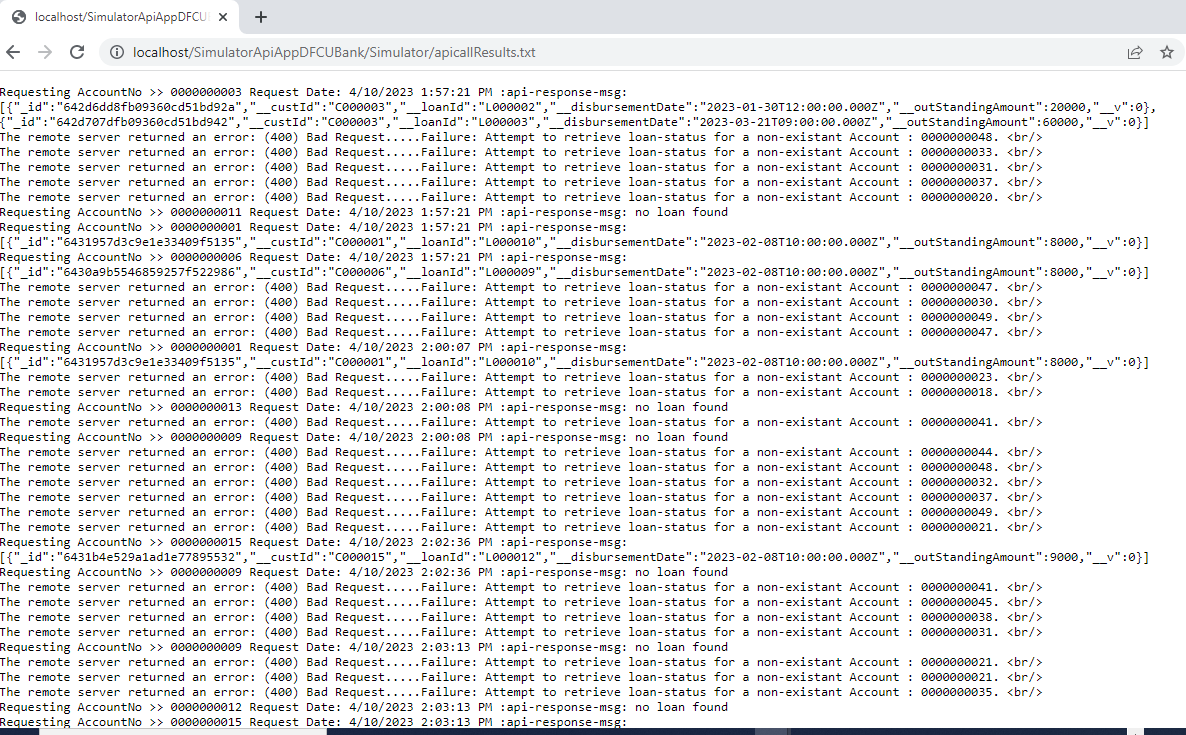
Request api for status of accounts—click ‘Invoke endpoint & commit “



Graphical user interface, text, application

Description automatically generated

See log on server file found on the Simulator server directory.



### END