CMPE 328
Homework 1

Tunç Gürsoy

64528127274

Instructor: Eren AYKIN

Date: 19/03/2021

CONTENTS

Technologies

Node.js

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.

MongoDB

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server-Side Public License (SSPL).

Heroku

Heroku is a cloud platform as a service (PaaS) supporting several programming languages.

Applications that are run on Heroku typically have a unique domain used to route HTTP requests to the correct application container or *dyno*. Each of the dynos are spread across a "dyno grid" which consists of several servers. Heroku's Git server handles application repository pushes from permitted users.

All Heroku services are hosted on Amazon's EC2 cloud-computing platform.

Pug

Pug is a high-performance template engine heavily influenced by **Haml** and implemented with JavaScript for **Node.js** and browsers.

Dependencies

Dotenv

Dotenv is a zero-dependency module that loads environment variables from a .env file into **process.env**. Storing configuration in the environment separate from code is based on **The Twelve-Factor App** methodology.

Express

Fast, unopinionated, minimalist web framework for **node**. The Express philosophy is to provide small, robust tooling for HTTP servers, making it a great solution for single page applications, web sites, hybrids, or public HTTP APIs.

Express does not force you to use any specific ORM or template engine. With support for over 14 template engines via **Consolidate.js**, you can quickly craft your perfect framework.

Mongoose

Mongoose is a **MongoDB** object modeling tool designed to work in an asynchronous environment. Mongoose supports both promises and callbacks.

Mocha

Mocha is a testing library for Node.js, created to be a simple, extensible, and fast. It’s used for unit and integration testing, and it’s a great candidate for BDD (Behavior Driven Development).

Supertest

SuperTest is an HTTP assertions library that allows you to test your Node.js HTTP servers. It is built on top of SuperAgent library, which is an HTTP client for Node.js.

Structure

Database holds data of the user which id, name, surname, email and tc.. User cannot give ID to himself database gives them unique ID. API can perform CRUD (Create, Read, Update, Delete) operations. User can make these operations with and without HTML Interface. “/users” route is for the without html

interface. I used postman application to perform CRUD operations. “/view and /viewsapi” is for the html interface.

How to Run

To run this application locally start code is “npm start” on VS code. This application is also run on the Heroku Server. Application address link is https://cmpe328.herokuapp.com/ .

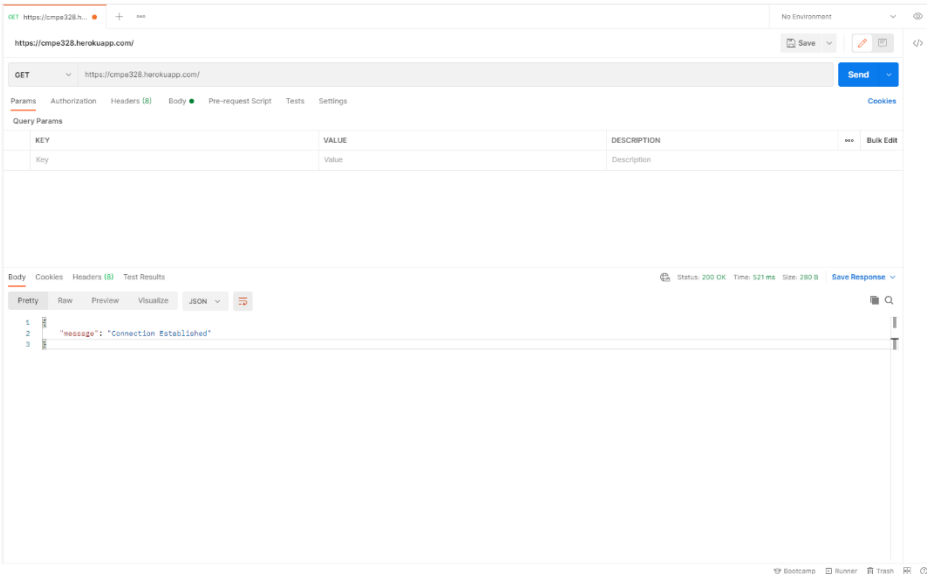
Routes

On local works on port 8080 so address of the http://localhost:8080/ . On Heroku server address is https://cmpe328.herokuapp.com/ .

“/”

GET Request

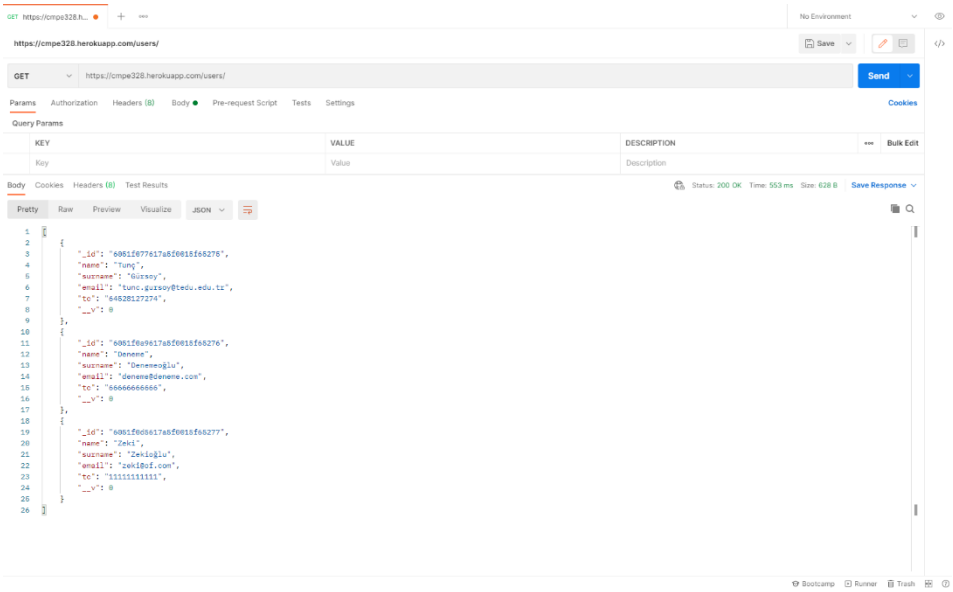
When we start the server default route it returns the connection established message



“/users”

GET Request

every user in database.



Post Request

Creates new user in database.

POST https://cmpe328....

https://cmpe328.herokuapp.com/users/

POST https://cmpe328.herokuapp.com/users/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "name":"Deneme2",
    "surname":"Deneme2oğlu",
    "email":"Deneme2@deneme.com",
    "tc":22222222222
}
```

Body Cookies Headers (8) Test Results Status: 201 Created Time: 472 ms Size: 416 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "message": "This user saved",
    "savedUser": {
        "_id": "6051f182617a5f0015f65278",
        "name": "Deneme2",
        "surname": "Deneme2oğlu",
        "email": "Deneme2@deneme.com",
        "tc": "22222222222"
    }
}
```

“users/:id”

Get Request

Gets the requested User.

GET https://cmpe328.h...

https://cmpe328.herokuapp.com/users/6051f182617a5f0015f65278

GET https://cmpe328.herokuapp.com/users/6051f182617a5f0015f65278 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

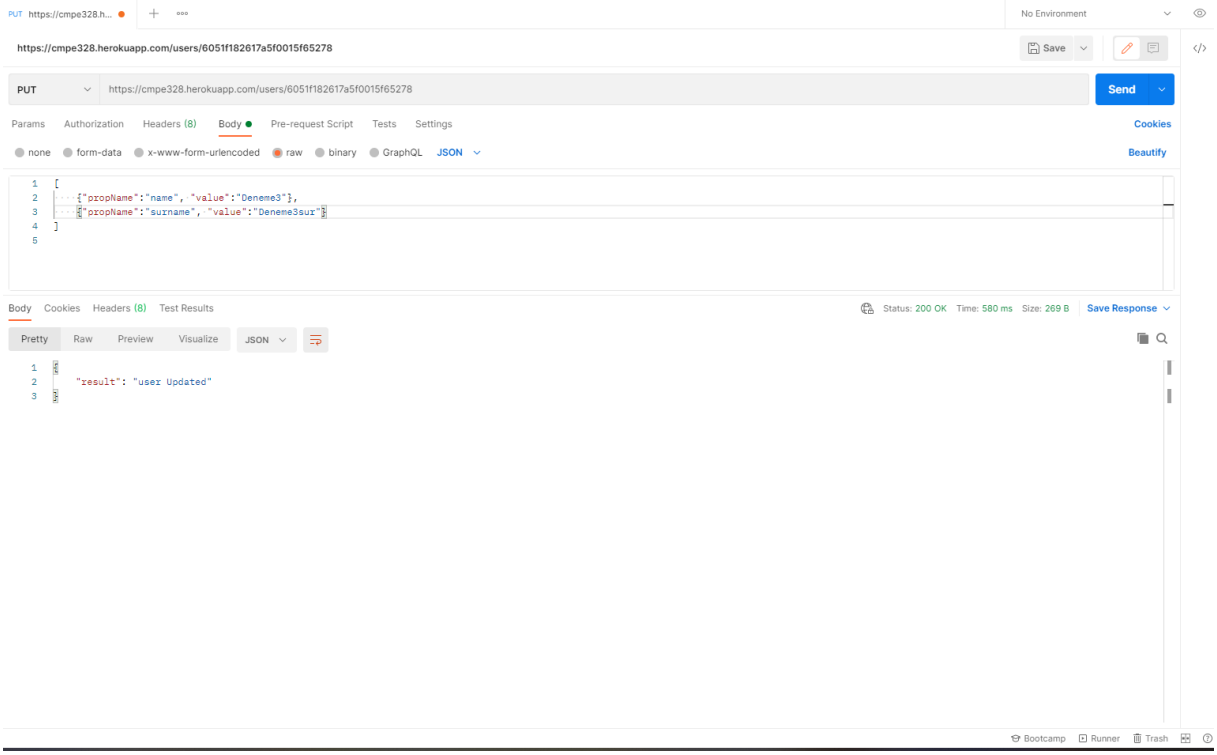
Body Cookies Headers (8) Test Results Status: 200 OK Time: 539 ms Size: 377 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "_id": "6051f182617a5f0015f65278",
    "name": "Deneme2",
    "surname": "Deneme2oğlu",
    "email": "Deneme2@deneme.com",
    "tc": "22222222222",
    "__v": 0
}
```

Put or Patch Request

Updates the requested user with given body (body should always be as in the example)



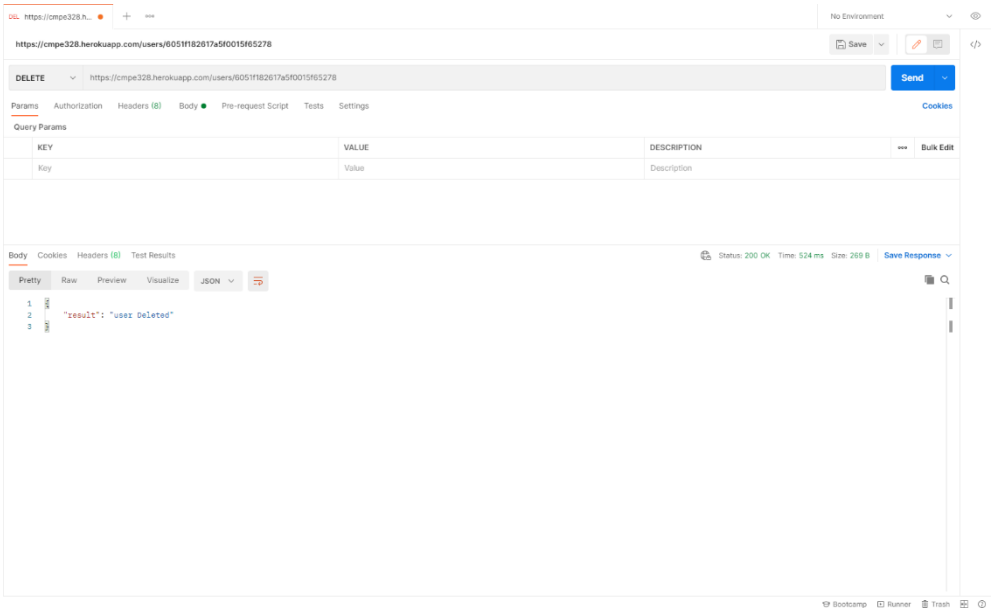
Example Body:

```
[
    {"propName":"name", "value":"ankara"},
    {"propName":"surname", "value":"Anıtkabir"}

]
```

Delete Request

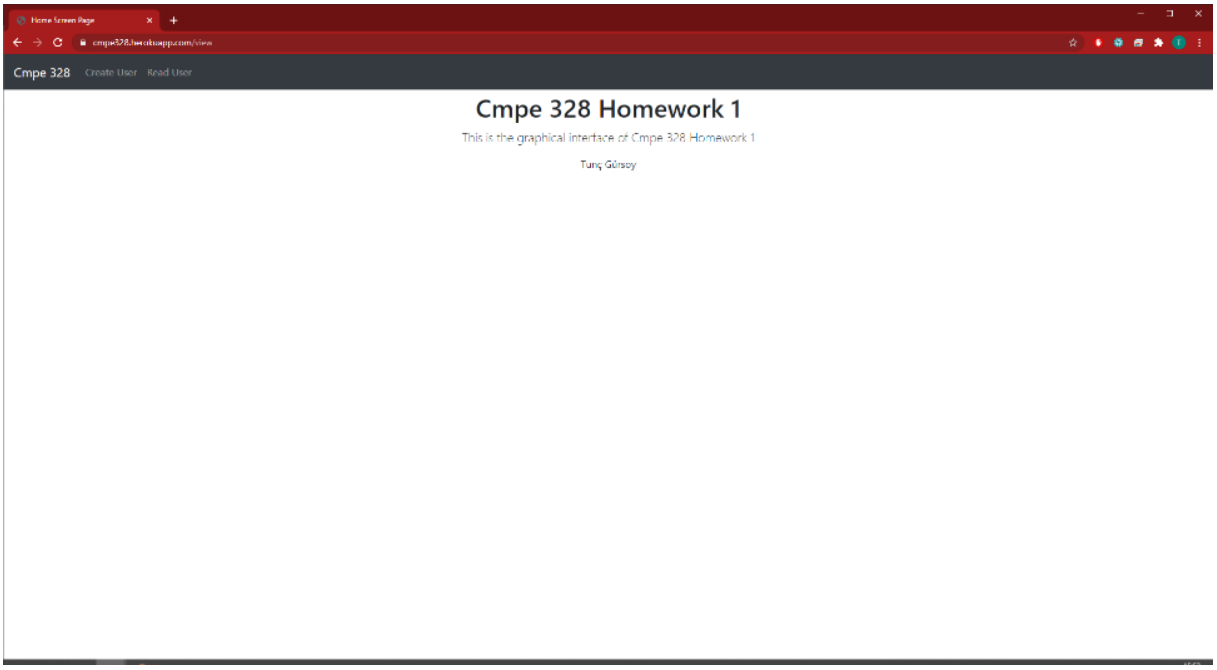
Deletes the requested user from database.



Html Interface (“/view” and “/viewsapi”)

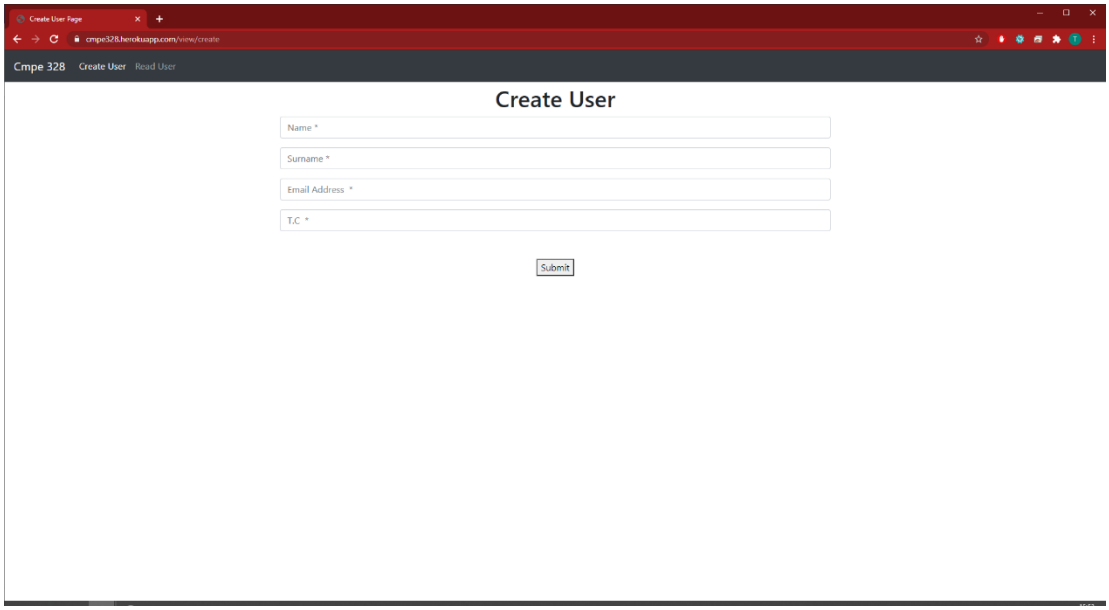
Home Page

Html Interface’s home page



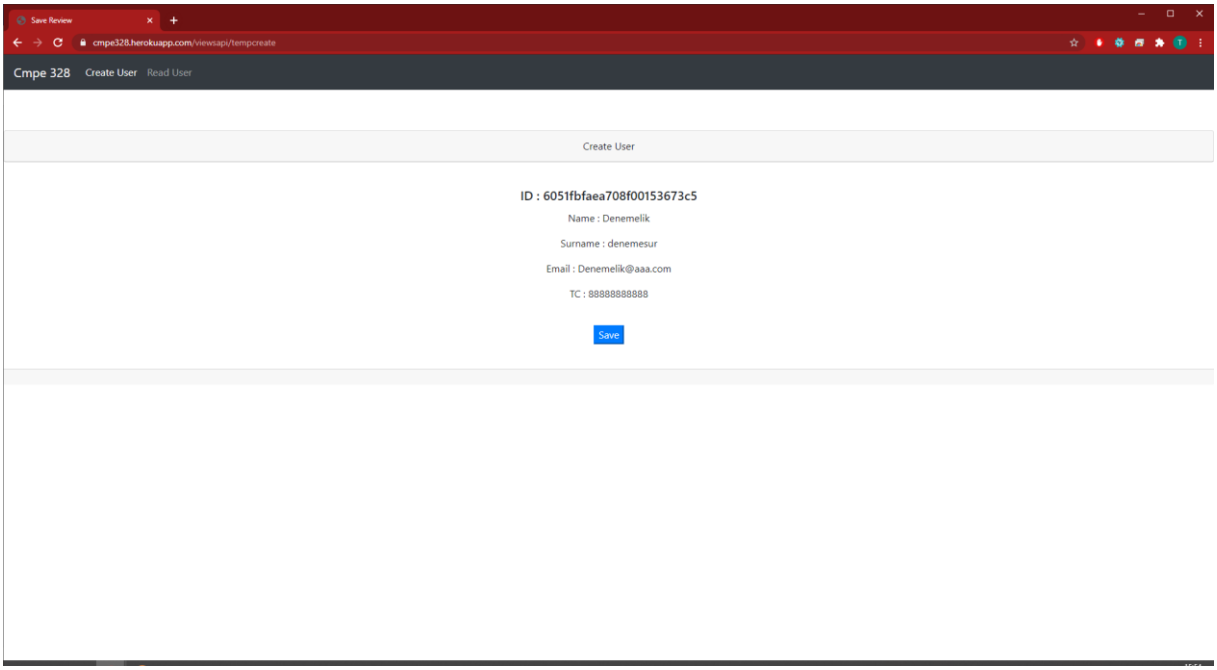
Create User Page

The page where the user can register himself by entering his personal information.



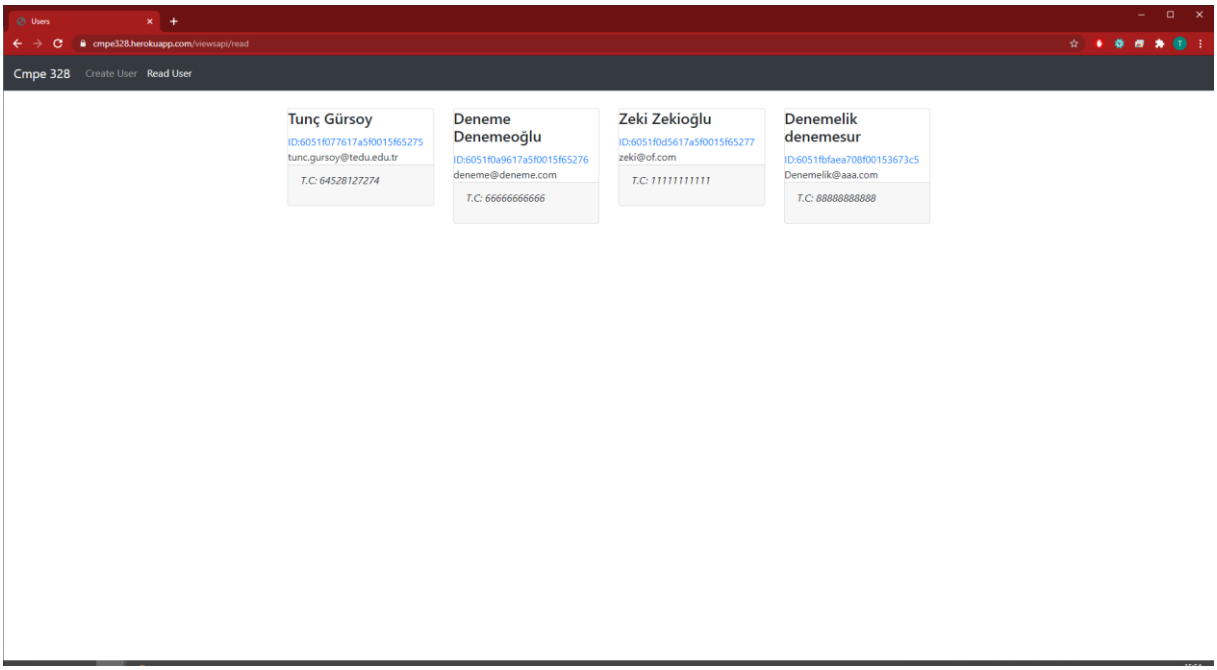
Create User Check Page

User see information which he entered to the system before entering the database.



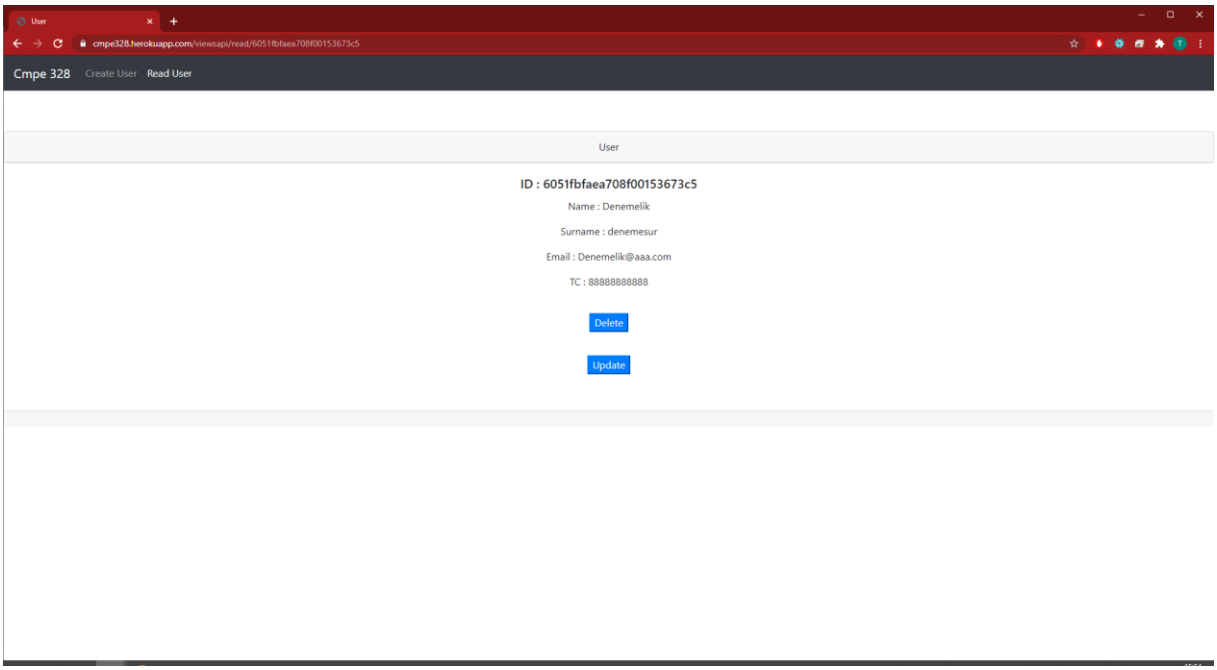
All Users Page

User can see all the users created and can click to any user's id and enter that user page.



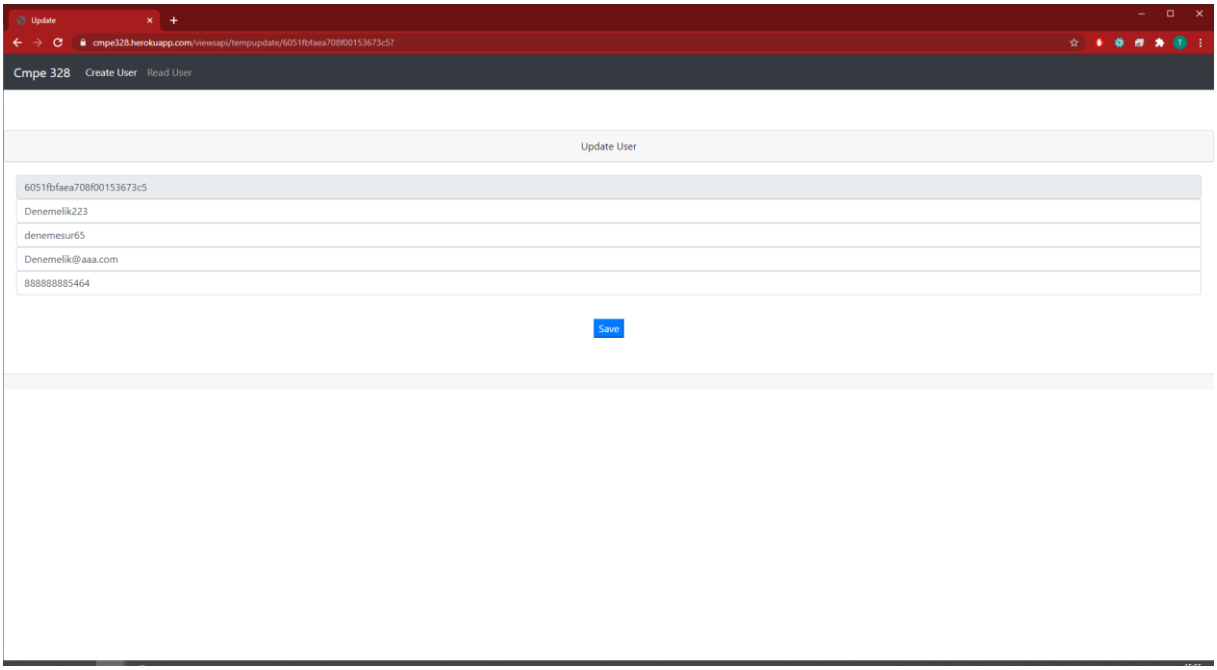
One User Page

Selected user information page and delete and update options.



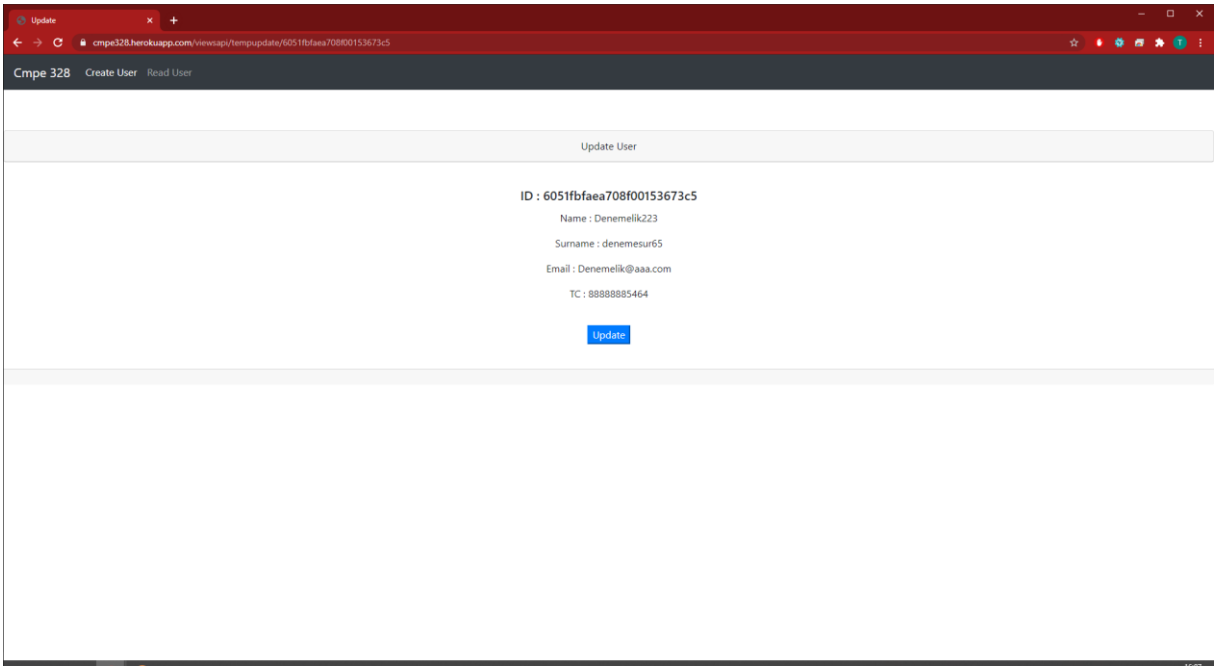
Update User Page

Selected user update page selected with any section except id



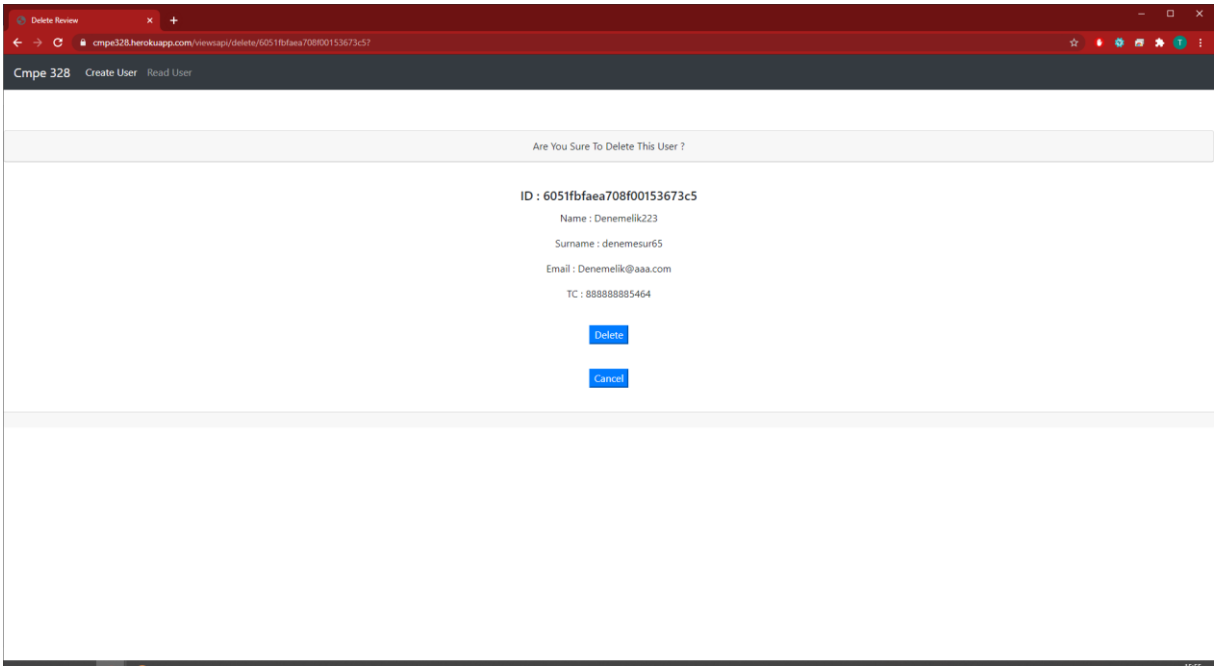
Update User Check Page

User information which he updated to the system before entering the database.



Delete user Check Page

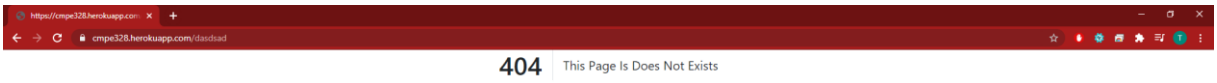
User information check before delete from database



Error Handling And CROS

When the application encounters an error, it redirects to the error page with the error code and description, as in the example.

In this application have CROS so user will not get an error about this reason.



Testing

I used unit testing method to test this API with Mocha testing library. To start testing “npm test” on Terminal of the VS code.