



CMPE 328
Homework 1

Tunç Gürsoy

64528127274

Instructor: Eren AYKIN

Date: 19/03/2021

CONTENTS

Technologies	3
Node.js	3
MongoDB	3
Heroku	3
Pug	3
Dependencies	3
Dotenv	3
Express	3
Mongoose	3
Mocha	3
Supertest	3
Structure	3
How to Run	4
Routes	4
"/"	4
GET Request	4
"/users"	4
GET Request	4
Post Request	5
"users/:id"	5
Get Request	5
Put or Patch Request	6
Delete Request	6
Html Interface ("/view" and "/viewsapi")	7
Home Page	7
Create User Page	7
Create User Check Page	8
All Users Page	8
One User Page	9
Selected user information page and delete and update options.	9
Update User Page	9
Update User Check Page	10
Delete user Check Page	10
Error Handling And CROS	11
Testing	11

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/>.

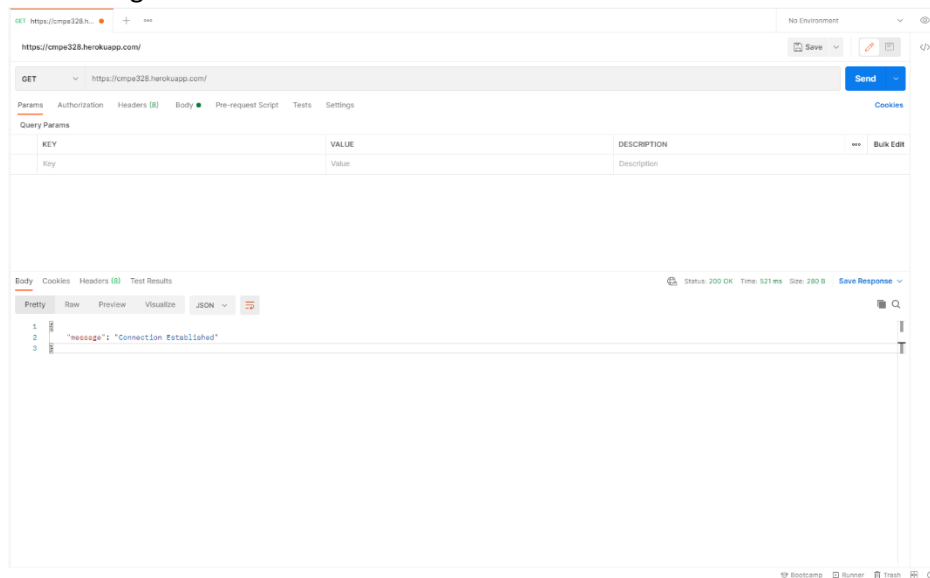
Routes

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

“/”

GET Request

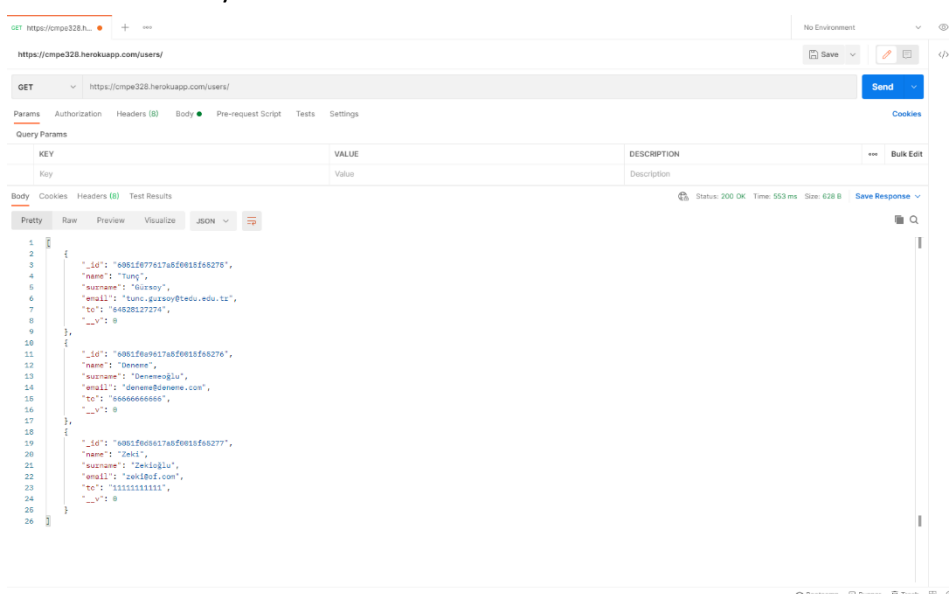
When we start the server default route it returns the connection established message



“/users”

GET Request

every user in database.



Post Request

Creates new user in database.

The screenshot shows a REST client interface with a POST request to `https://cmpe328.herokuapp.com/users/`. The request body is a JSON object: `{ "name": "Deneme2", "surname": "Deneme2oglu", "email": "Deneme2@deneme.com", "tc": "2222222222" }`. The response status is 201 Created, with a time of 472 ms and a size of 416 B. The response body is a JSON object: `{ "message": "This user saved", "savedUser": { "_id": "6051f182617a5f0015f65278", "name": "Deneme2", "surname": "Deneme2oglu", "email": "Deneme2@deneme.com", "tc": "2222222222" } } }`.

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

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

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

```
1 {
2   "name": "Deneme2",
3   "surname": "Deneme2oglu",
4   "email": "Deneme2@deneme.com",
5   "tc": "2222222222"
6 }
```

Body Cookies Headers (8) Test Results

Status: 201 Created Time: 472 ms Size: 416 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "This user saved",
3   "savedUser": {
4     "_id": "6051f182617a5f0015f65278",
5     "name": "Deneme2",
6     "surname": "Deneme2oglu",
7     "email": "Deneme2@deneme.com",
8     "tc": "2222222222"
9   }
10 }
```

“users/:id”

Get Request

Gets the requested User.

The screenshot shows a REST client interface with a GET request to `https://cmpe328.herokuapp.com/users/6051f182617a5f0015f65278`. The response status is 200 OK, with a time of 539 ms and a size of 377 B. The response body is a JSON object: `{ "_id": "6051f182617a5f0015f65278", "name": "Deneme2", "surname": "Deneme2oglu", "email": "Deneme2@deneme.com", "tc": "2222222222", "__v": 0 }`.

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

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

```
1 {
2   "_id": "6051f182617a5f0015f65278",
3   "name": "Deneme2",
4   "surname": "Deneme2oglu",
5   "email": "Deneme2@deneme.com",
6   "tc": "2222222222",
7   "__v": 0
8 }
```

Put or Patch Request

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

The screenshot shows a REST client interface with a PUT request to `https://cmpe328.herokuapp.com/users/6051f182617a5f0015f65278`. The request body is a JSON object: `{ "propName": "name", "value": "Deneme3"}, {"propName": "surname", "value": "Deneme3suz"}`. The response status is 200 OK, and the response body is `"result": "user Updated"`.

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

```
{
  "propName": "name", "value": "Deneme3"},
  "propName": "surname", "value": "Deneme3suz"
}
```

```
"result": "user Updated"
```

Example Body:

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

Delete Request

Deletes the requested user from database.

The screenshot shows a REST client interface with a DELETE request to `https://cmpe328.herokuapp.com/users/6051f182617a5f0015f65278`. The response status is 200 OK, and the response body is `"result": "user Deleted"`.

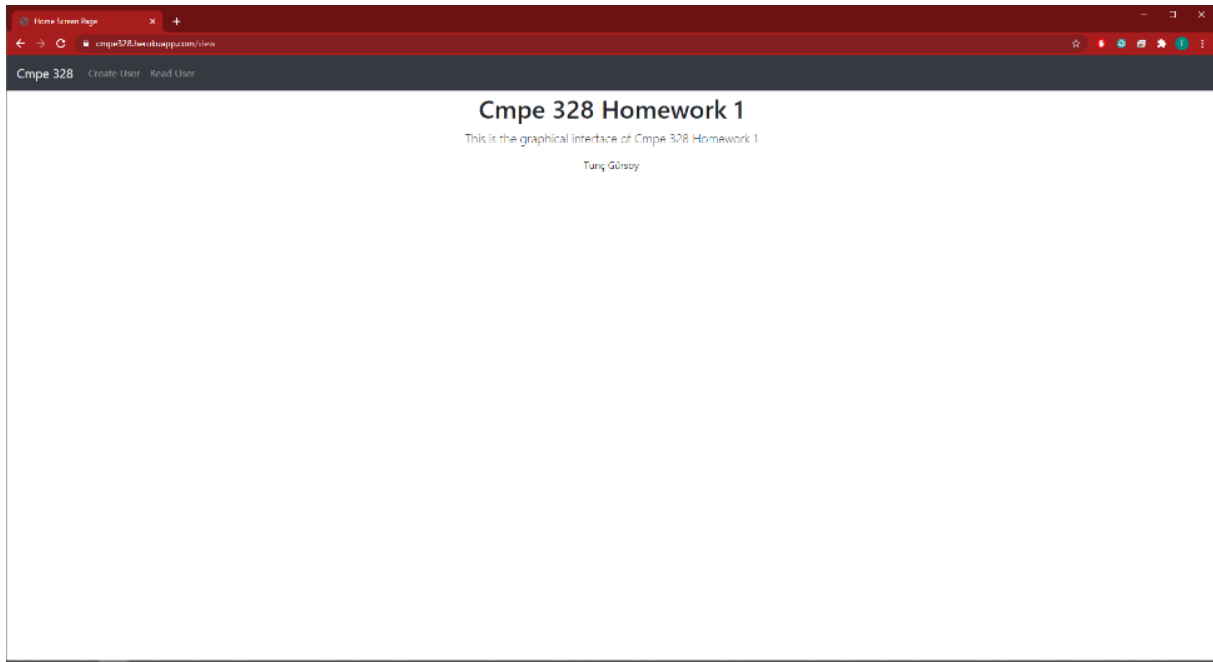
```
DELETE https://cmpe328.herokuapp.com/users/6051f182617a5f0015f65278
```

```
"result": "user Deleted"
```

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

Home Page

Html Interface’s home page



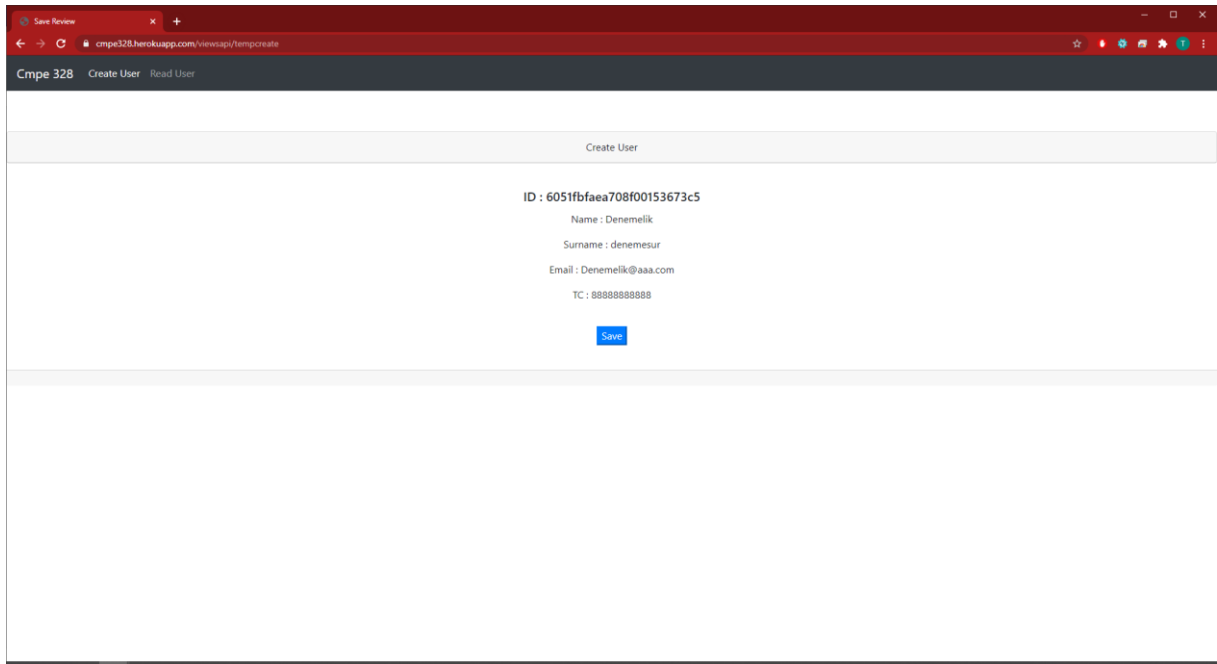
Create User Page

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

A screenshot of a web browser displaying the "Create User" page. The browser's address bar shows the URL `cmpe328.herokuapp.com/view/create`. The page has a dark blue header with the text "Cmpe 328" and navigation links "Create User" and "Read User". The main content area is white and features the title "Create User" in bold. Below the title, there are four input fields labeled "Name *", "Surname *", "Email Address *", and "T.C *". A "Submit" button is located below the input fields.

Create User Check Page

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



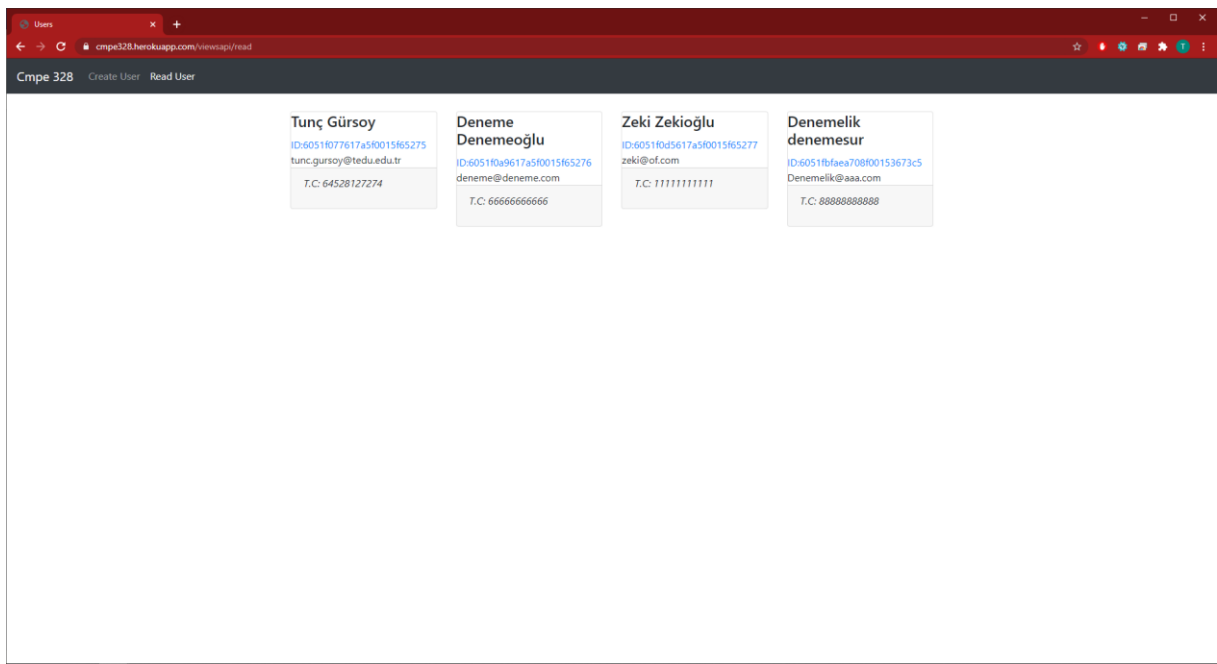
The screenshot shows a web browser window with the URL `cmpe328.herokuapp.com/views/api/tempcreate`. The page title is "Create User". The user's input details are displayed as follows:

Field	Value
ID	6051fbfaea708f00153673c5
Name	Denemelik
Surname	denemesur
Email	Denemelik@aaa.com
TC	88888888888

A blue "Save" button is located below the input details.

All Users Page

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



The screenshot shows a web browser window with the URL `cmpe328.herokuapp.com/views/api/read`. The page title is "Users". The list of users is displayed as follows:

Name	ID	Email	TC
Tunç Gürsoy	60518077617a5f0015f65275	tunc.gursoy@tedu.edu.tr	64528127274
Deneme Denemeoğlu	6051f0a9617a5f0015f65276	deneme@deneme.com	66666666666
Zeki Zekioglu	6051f0d5617a5f0015f65277	zeki@of.com	11111111111
Denemelik denemesur	6051fbfaea708f00153673c5	Denemelik@aaa.com	88888888888

One User Page

Selected user information page and delete and update options.

User

ID : 6051fbfaea708f00153673c5

Name : Denemelik

Surname : denemesur

Email : Denemelik@aaa.com

TC : 88888888888

Delete

Update

Update User Page

Selected user update page selected with any section except id

Update User

6051fbfaea708f00153673c5

Denemelik223

denemesur65

Denemelik@aaa.com

888888885464

Save

Update User Check Page

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

Update User

ID : 6051fbfaea708f00153673c5

Name : Denemelik223

Surname : denemesur65

Email : Denemelik@aaa.com

TC : 88888885464

Update

Delete user Check Page

User information check before delete from database

Are You Sure To Delete This User ?

ID : 6051fbfaea708f00153673c5

Name : Denemelik223

Surname : denemesur65

Email : Denemelik@aaa.com

TC : 88888885464

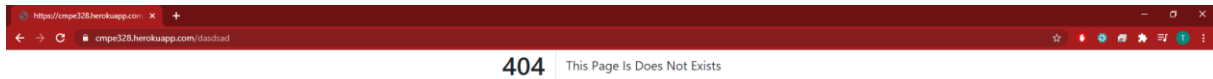
Delete

Cancel

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.

