**Friends Management**

**Background:** For any application with a need to build its own social network, "Friends Management" is a common requirement which usually starts off simple but can grow in complexity depending on the application's use case.

Usually, applications would start with features like "Friend", "Unfriend", "Block", "Receive Updates" etc.

**Software:**

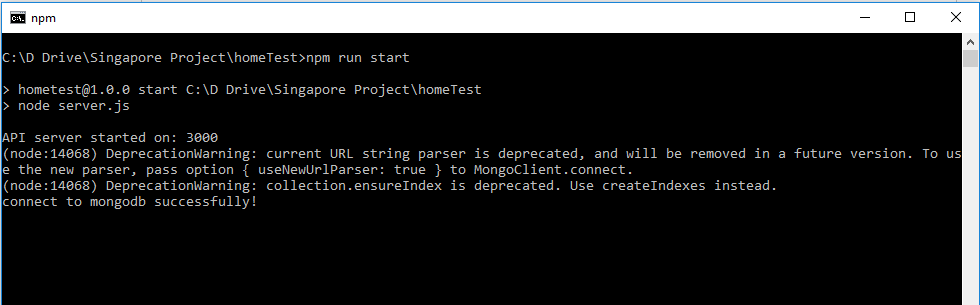
1. Nodejs
2. MangoDB

**Server setup:**

1. npm install --save-dev nodemon
2. npm install express --save
3. npm install mongoose –save
4. npm install intersect
5. npm install body-parser

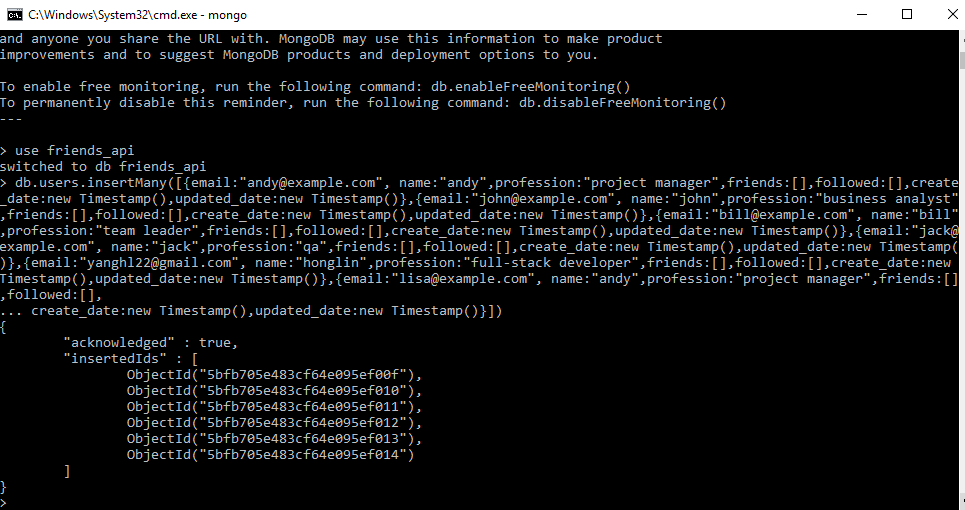
### Start the server

Run "npm run start" on your terminal to start the server.



**Mongo Server**

1. Open terminal from Program files where MongoDB is install (C:\Program Files\MongoDB\Server\4.0\bin)
2. Run "mongo" command on your terminal to start Mongo shell.
3. Run below 2 commands to insert fields to Mongodb.
4. use friends\_api
5. db.users.insertMany([{email:"andy@example.com", name:"andy",profession:"project manager",friends:[],followed:[],create\_date:new Timestamp(),updated\_date:new Timestamp()},{email:"john@example.com", name:"john",profession:"business analyst",friends:[],followed:[],create\_date:new Timestamp(),updated\_date:new Timestamp()},{email:"bill@example.com", name:"bill",profession:"team leader",friends:[],followed:[],create\_date:new Timestamp(),updated\_date:new Timestamp()},{email:"jack@example.com", name:"jack",profession:"qa",friends:[],followed:[],create\_date:new Timestamp(),updated\_date:new Timestamp()},{email:"yanghl22@gmail.com", name:"honglin",profession:"full-stack developer",friends:[],followed:[],create\_date:new Timestamp(),updated\_date:new Timestamp()},{email:"lisa@example.com", name:" lisa ",profession:"project manager",friends:[],followed:[], create\_date:new Timestamp(),updated\_date:new Timestamp()},{email:"common@example.com", name:"common",profession:"project manager",friends:[],followed:[], create\_date:new Timestamp(),updated\_date:new Timestamp()},{email:"kate@example.com", name:"kate",profession:"project manager",friends:[],followed:[], create\_date:new Timestamp(),updated\_date:new Timestamp()}])



**User Stories**

1. **As a user, I need an API to create a friend connection between two email addresses.**

The API should receive the following JSON request:

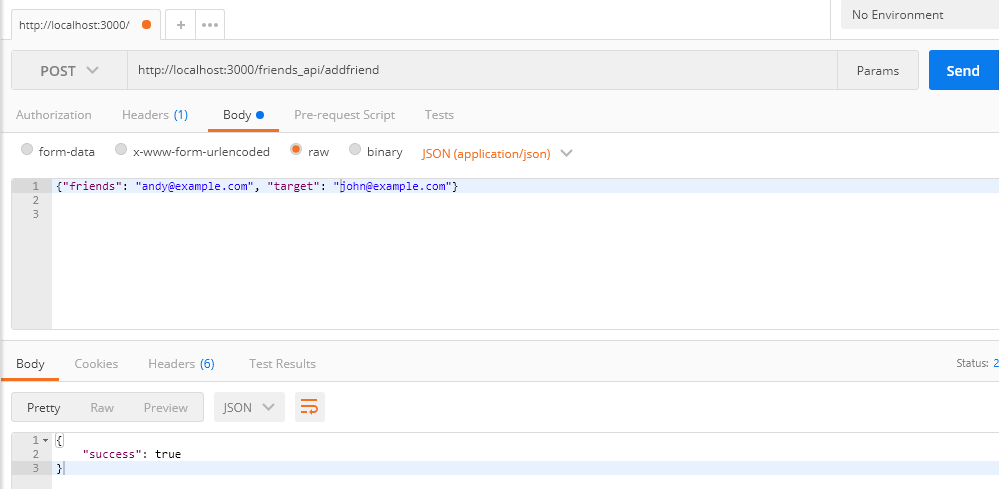
{“friends": "andy@example.com", "target": "john@example.com”}

The API should return the following JSON response on success:

{

"success": true

}



1. **As a user, I need an API to retrieve the friends list for an email address.**

The API should receive the following JSON request:

{

email: 'andy@example.com'

}

The API should return the following JSON response on success:

{

"success": true,

"friends" :

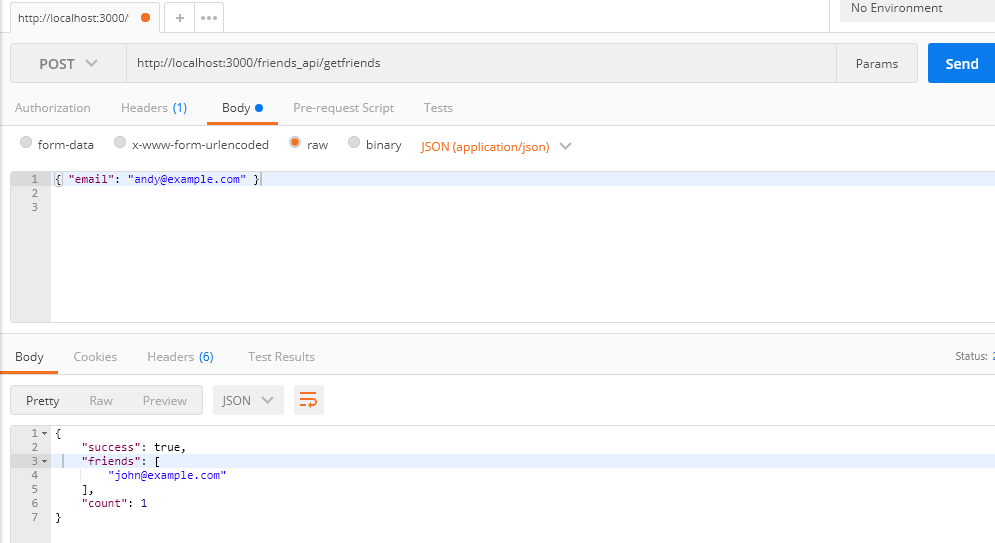
[

'john@example.com'

],

"count" : 1

}



**3. As a user, I need an API to retrieve the common friends list between two email addresses.**

The API should receive the following JSON request:

{

friends:

[

'andy@example.com',

'john@example.com'

]

}

The API should return the following JSON response on success:

{

"success": true,

"friends" :

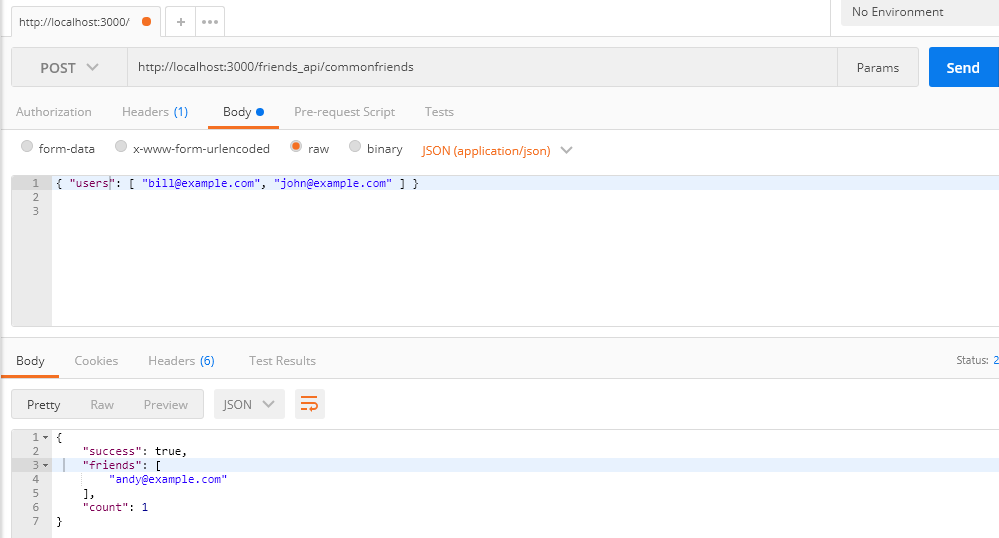
[

'common@example.com'

],

"count" : 1

}



**4. As a user, I need an API to subscribe to updates from an email address.**

Please note that "subscribing to updates" is NOT equivalent to "adding a friend connection".

The API should receive the following JSON request:

{

"requestor": "lisa@example.com",

"target": "john@example.com"

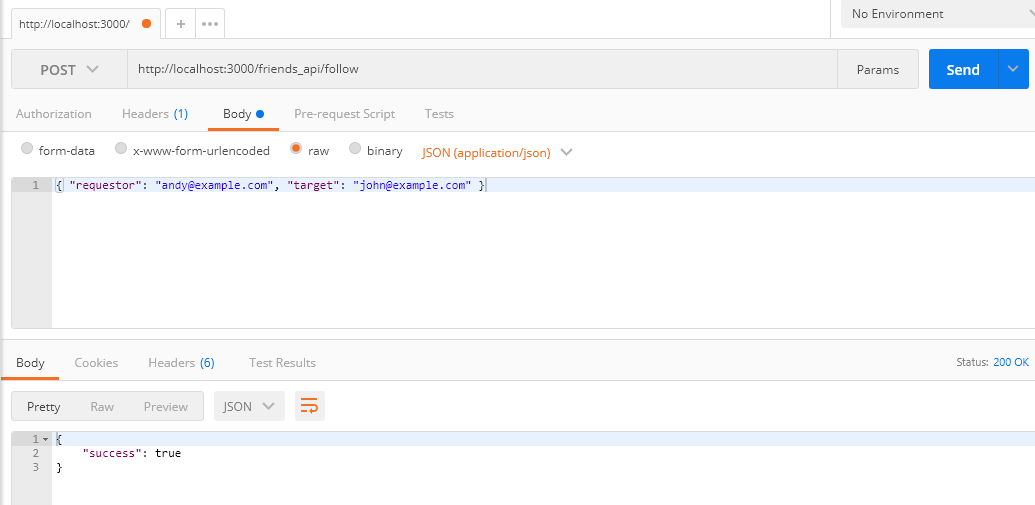
}

The API should return the following JSON response on success:

{

"success": true

}



**5. As a user, I need an API to block updates from an email address.**

Suppose "andy@example.com" blocks "john@example.com":

* • if they are connected as friends, then "andy" will no longer receive notifications from "john"
* • if they are not connected as friends, then no new friends connection can be added

The API should receive the following JSON request:

{

"requestor": "andy@example.com",

"target": "john@example.com"

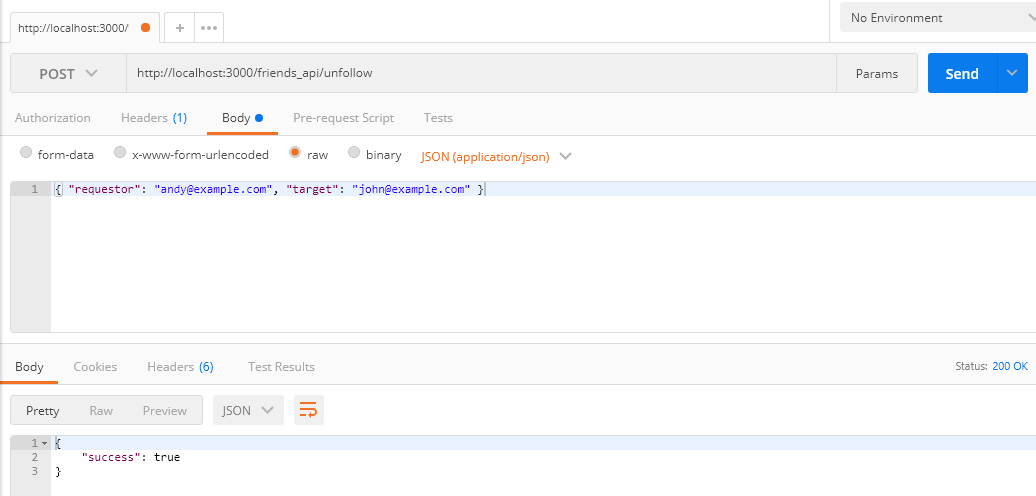
}

The API should return the following JSON response on success:

{

"success": true

}



**6. As a user, I need an API to retrieve all email addresses that can receive updates from an email address.**

Eligibility for receiving updates from i.e. "john@example.com":

* • has not blocked updates from "john@example.com", and
  1. • at least one of the following: o has a friend connection with "john@example.com"
  2. o has subscribed to updates from "john@example.com"

o has been @mentioned in the update

The API should receive the following JSON request:

{

"sender": "john@example.com",

"text": "Hello World! kate@example.com"

}

The API should return the following JSON response on success:

{

"success": true

"recipients":

[

"lisa@example.com",

"kate@example.com"

]

}

