

How to hash password?

Install module “bcryptjs” for password security

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE  powershell

PS D:\reactshrenik\Ex8-Express-Js\#7-Express-MongoDb-Create-Schema\mongo-schema-demo> npm i --save bcryptjs
npm WARN mongo-schema-demo@1.0.0 No description
npm WARN mongo-schema-demo@1.0.0 No repository field.

+ bcryptjs@2.4.3
added 1 package from 6 contributors and audited 84 packages in 1.983s
found 0 vulnerabilities

PS D:\reactshrenik\Ex8-Express-Js\#7-Express-MongoDb-Create-Schema\mongo-schema-demo>
```

Import “bcryptjs” (Line No.5) in controller

```
JS user.js  X
controller > JS user.js > ...
1  const router = require("express").Router();
2  const user = require("../model/user");
3  const { check, validationResult } = require("express-validator");
4  const bodyParser = require("body-parser");
5  const bcrypt = require("bcryptjs");
6
```

Call hashSync() function to encrypt password

```
37 //response.end(JSON.stringify({status:true, message:"User data is correct"},null,3));
38 const hashedPassword = bcrypt.hashSync(request.body.password, 10);
39
40 user.create(
41   {
42     username:request.body.username,
43     password:hashedPassword,
44     email:request.body.email
45   },
```

Run Project

Insert one more record and check output in
mongodb

Sending form data from postman

The screenshot shows the Postman interface for a POST request. The request is titled "Untitled Request" and is sent to "localhost:3002/api/user/create". The request body is configured as "x-www-form-urlencoded" and contains three form fields: "username" with value "rahul", "password" with value "admin111", and "email" with value "rahul77@gmail.com". The response is a JSON object: {"status": true, "message": "Insert Successfull"}.

POST localhost:3002/api/user/create

Untitled Request

POST localhost:3002/api/user/create

Send Save

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies Code

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

<input checked="" type="checkbox"/>	username	rahul	
<input checked="" type="checkbox"/>	password	admin111	
<input checked="" type="checkbox"/>	email	rahul77@gmail.com	
	Key	Value	Description

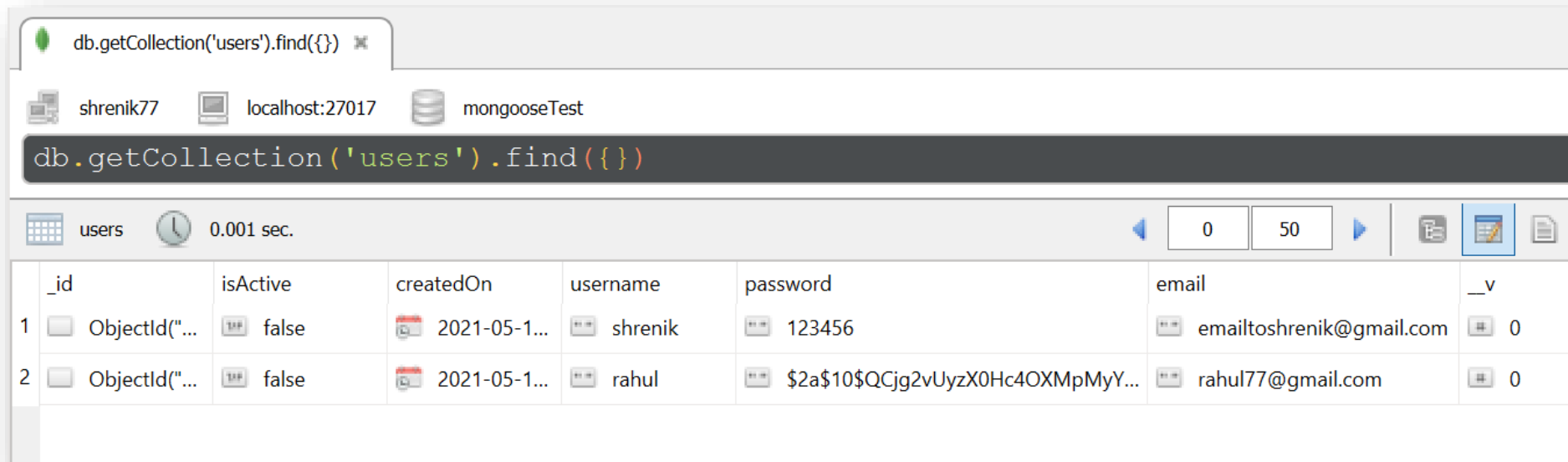
Body Cookies Headers (5) Test Results

Status: 200 OK Time: 321 ms Size: 212 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": true,
3   "message": "Insert Successfull"
4 }
```

Output in robo 3T



The screenshot shows the Robo 3T application interface. At the top, a tab displays the query `db.getCollection('users').find({})`. Below the tab, the connection details are shown: `shrenik77` (username), `localhost:27017` (host), and `mongooseTest` (database). The query `db.getCollection('users').find({})` is entered in the command field. Below the command field, the results are displayed in a table. The table has columns: `_id`, `isActive`, `createdOn`, `username`, `password`, `email`, and `__v`. Two rows of data are shown. The first row has `_id` as `ObjectId("...")`, `isActive` as `false`, `createdOn` as `2021-05-1...`, `username` as `shrenik`, `password` as `123456`, `email` as `emailtoshrenik@gmail.com`, and `__v` as `0`. The second row has `_id` as `ObjectId("...")`, `isActive` as `false`, `createdOn` as `2021-05-1...`, `username` as `rahul`, `password` as `$2a$10$QCjg2vUyzX0Hc4OXMpMyY...`, `email` as `rahul77@gmail.com`, and `__v` as `0`.

db.getCollection('users').find({})

shrenik77 localhost:27017 mongooseTest

db.getCollection('users').find({})

users 0.001 sec. 0 50

	_id	isActive	createdOn	username	password	email	__v
1	ObjectId("...")	false	2021-05-1...	shrenik	123456	emailtoshrenik@gmail.com	0
2	ObjectId("...")	false	2021-05-1...	rahul	\$2a\$10\$QCjg2vUyzX0Hc4OXMpMyY...	rahul77@gmail.com	0