

# **PATIENT'S UNIQUE IDENTIFICATION**

**MID TERM REPORT**

OF MINI PROJECT

**BACHELOR OF TECHNOLOGY**

Computer Science and Engineering Branch

**SUBMITTED BY :**

ABHINAV BHARDWAJ ( 181500009 )  
ADITYA SINGH CHAUHAN ( 181500046 )  
ANVIT GUPTA ( 181500127 )

**SUPERVISED BY :**

Mr. VAIBHAV DIWAN  
( Technical Trainer )



**GLA University, Mathura**

**( 2020 – 2021 )**

# Content

---

## **Abstract**

### **1. Introduction**

- a. General Introduction to the Project
- b. Area of Computer Science

### **2. Problem Definition**

### **3. Objective**

### **4. Technology being used**

### **5. Methodology**

### **6. Implementation**

### **7. Progress**

### **8. Screenshots**

## **Abstract**

We all can't disagree with the fact that there are so many people in our society who are patients of such kind of diseases which need a long duration of treatment. And changing multiple doctors in this long duration of treatment is very often. In some case, it may due to unsatisfactory treatment by his/her doctor, while in some cases patients are supposed to change their city because of other reasons, but in both cases, it becomes very hard to keep its medical history (like previous prescriptions, reports, test results, etc.) with him/her.

## **Introduction**

### **1.1 General Introduction to the Project**

Currently in our society, patient used to keep hard copy of reports and prescriptions that becomes very difficult to maintain. There is no central system in our country that provides a facility of accessing someone's medical history in case of any accident or emergency. We totally depend on patient or his/her family during treatment which results problematic delays.

We are going to design a system that will provide Signup and then Login facility to the patient. We'll try to help patient in maintaining record of previous reports and prescriptions. As well as our system will attach a Unique Patient Identification to every patient who creates account on our database. This Unique number will be used to view patient's report. We'll design an android app as well as a website to facilitate patient as well as guest. Guest view will be used in viewing report of any patient if you are having his/her Unique Identification Number. It'll be very important, if in case of any emergency, any paramedical staff, who is not having account on our database, wants to examine operate any patient then he/she can view patient's reports just by searching Patient's Unique Identification number.

## **Area of Computer Science**

The computer has brought revolution in every sphere of human life, whether it is business, education field, governance, medical science etc. The computer has reduced the human work load, businesses are going global and everything is available at the click of mouse. Most of the patients have to go to hospital which is a very time-consuming task. By this application they can keep in touch with them.

## **Problem Definition**

Most of the people nowadays prefer online shopping, this become more convenient for them as most of the things they want are available online and can be purchased with a single click. There are some areas of online shopping which needs to be covered like most of the time one has to visit shop to buy goods related to construction. So, we are trying to make an online platform which provide these types of construction material goods.

## **Objectives of the Project**

Unique Patient Identifier (UPI) will be an individual identification number of every patient issued by this system. The objective of this project is to collect Patient's basic details (like Name, DoB, Contact number, Blood Group, Address, etc.) as well as giving a facility to upload heap of reports and prescriptions, store them in a centralized database.

## **Technology being used**

### **Hardware Requirements :**

- Computer System with minimum 8GB of RAM

### **Software Requirements :**

- Windows/Linux OS
- Android Studio
- Visual Code Studio
- Robo 3T
- Postman
- Adobe XD

### **Programming language :**

- Java Programming
- JavaScript
- MERN Stack

## Methodology

We are using Android Java and React based Webpage Portal as the frontend with the backend made using MongoDB. To see what's inside the android application a user has to login, if the user is not register, he/she can also register himself/herself. App will keep records of patient's reports as well as past health issues which we have feed there.

The modules used in this app are listed as follows:

1. Main Dashboard
2. My Profile
3. BMI Calculator
4. Upload Record
5. Records
6. Logout

## Implementation Details

The implementation is divided in two parts:

### App Development

#### i. Backend Development:

The technology used for developing backend is MongoDB. Basically we have implemented the backend part using Mongo. We implemented an admin panel to monitor all the activities which will be happening in the android application. The owner of this application is having one admin login username and password, only the admin can add or remove the data, the changes will be reflected to the android application. The data will be sent to the android application using API's

An application program interface (API) is a set of routines, protocols, and tools for building software applications. Basically, an API specifies how software components should interact. The data will sent in JSON format.

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.

- #### ii. Frontend Development:
- We are using android studio, the tool which is used to develop android applications. The technology used for developing frontend is Android and Java. User will have to register themselves before login. We are parsing the data coming in JSON from the backend and displaying it on android application.

The modules used in this app are listed as follows:

1. Sign In – will use to sign in to the account.
2. Sign Up – will use to register his/her self.

3. Dashboard – It consist of 6 modules:
  - a. My Profile – It will show the profile of the user
  - b. Records – It will keep records of reports.
  - c. BMI Calculator – It will calculate BMI of user by some inputting some variables
  - d. Upload Record – It will use to upload/save his/her records.
  - e. ToDo – will ease you to mark to do work.

**Webpage based Portal:** We are working on this in two parts viz frontend and backend. Firstly, we are working on backend after that will move towards frontend. There might be some changes in the backend while working with frontend.

**i. Backend Development:** The project commences with designing of fundamental patient schema and document schema. We will use APIS for creating different routes. Unique Patient Identification Number (UPI) will be generated by the server every time the new user creates his/her account. We will differentiate user as patient and admin with the help of middleware. Admin will be able to use few more functionalities than patient like viewing total number of users, etc. The data will be sent to backend using APIs.

An application program interface (API) is a set of routines, protocols and tools for building software applications. Basically, an API specifies how software components should interact.

The data will be sent in JSON format.

JSON is a lightweight data-interchange format it is easy for humans to read and write. It is easy for machines to parse and generate.

We are using MongoDB Atlas for cloud storage during development phase because of its flexibility and scalability of document database, available as a fully managed service.

MongoDB Atlas is the global cloud database service for modern applications. Deploy fully managed MongoDB across AWS, Azure, or GCP. Best-in-class automation and proven practices guarantee availability, scalability, and compliance with the most demanding data security and privacy standards.

- ii. Frontend:** We will be using React, HTML, Bootstrap for frontend development part. On home page, there will be basic functionalities like BMI calculator or viewing public details of any patient using UPI. But to store his/her records, he/she has to create account using Signup Page and then signin using SignIn Page. After signing in into account, there will be various modules like My Profile, Records, Upload Record, BMI Calculator, Signout etc.

## PROGRESS

### On App based Portal:

- Created app using Android studio, so that it can be used for GUI based Mobile application.
- Created Home page / introductory page.
- Created Sing-in /login page through which user can access their account.
- Created Sing-up /register page through which user can register in the account by basic details.
- Created Dashboard which consist of 6 modules.
- Modules are –
  - My profile
  - Records
  - BMI Calculator
  - Upload Records
  - To-do

### On Web Page based Portal:

- Created online Database on MongoDB Atlas, so that it can be used for Web Page based Portal.
- Successfully connected Backend of the Web Page based Portal with database.
- Created a basic Schema for Patient.
- Created all the controllers required for creating a new account of the Patient (User).
- Created all the controllers required for signing in account of the Patient (User).
- Created all the controllers required for signing out Patient (User) from his/her account.

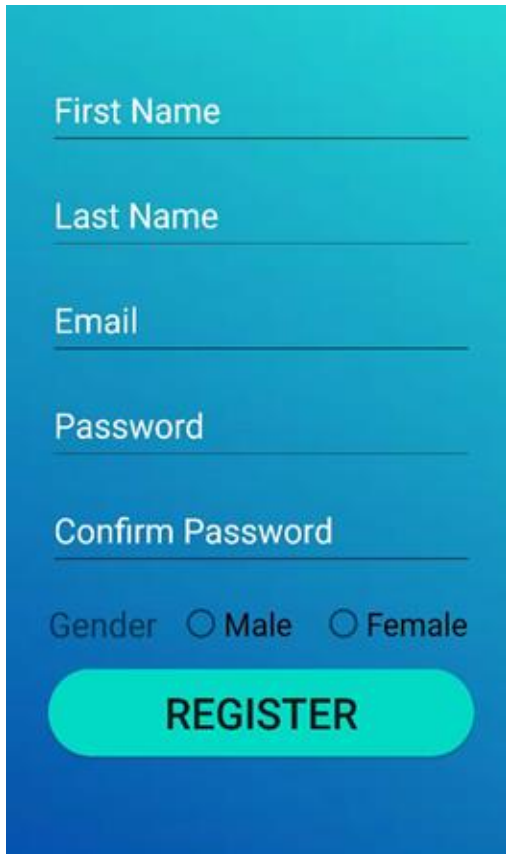
- Created the route required for creating new account of the Patient (User) as `http://localhost:XXXXXX /api/signup`.
- Created the route required for signing in account of the Patient (User) as `http://localhost:XXXXXX /api/signin`.
- Created the route required for signing out Patient (User) from his/her account as `http://localhost:XXXXXX /api/signout`.



# SCREENSHOTS

## Android Application:

**Signup Page**



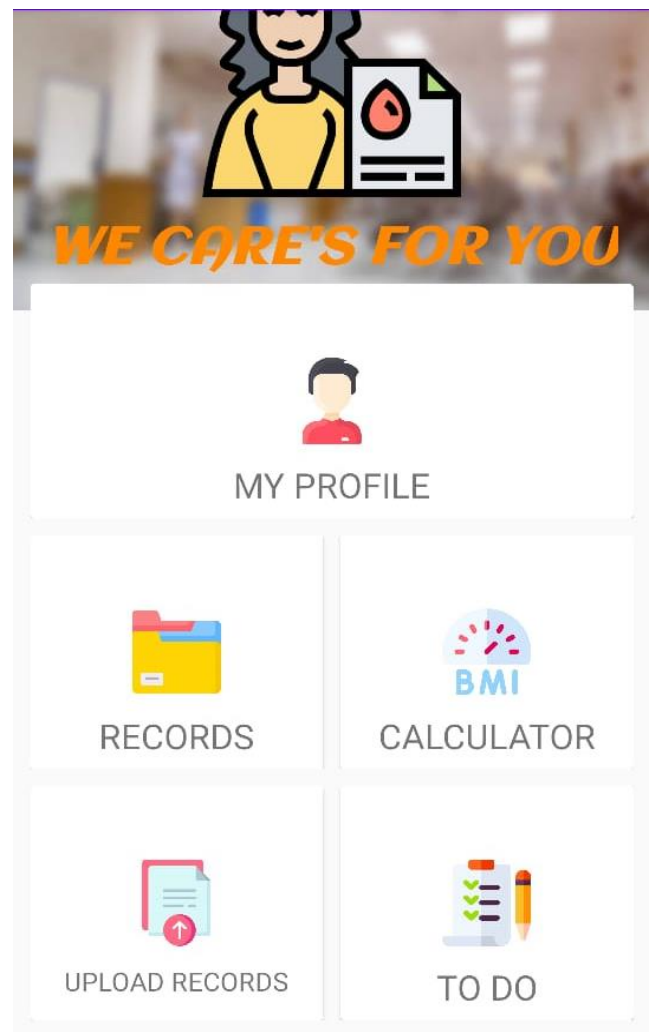
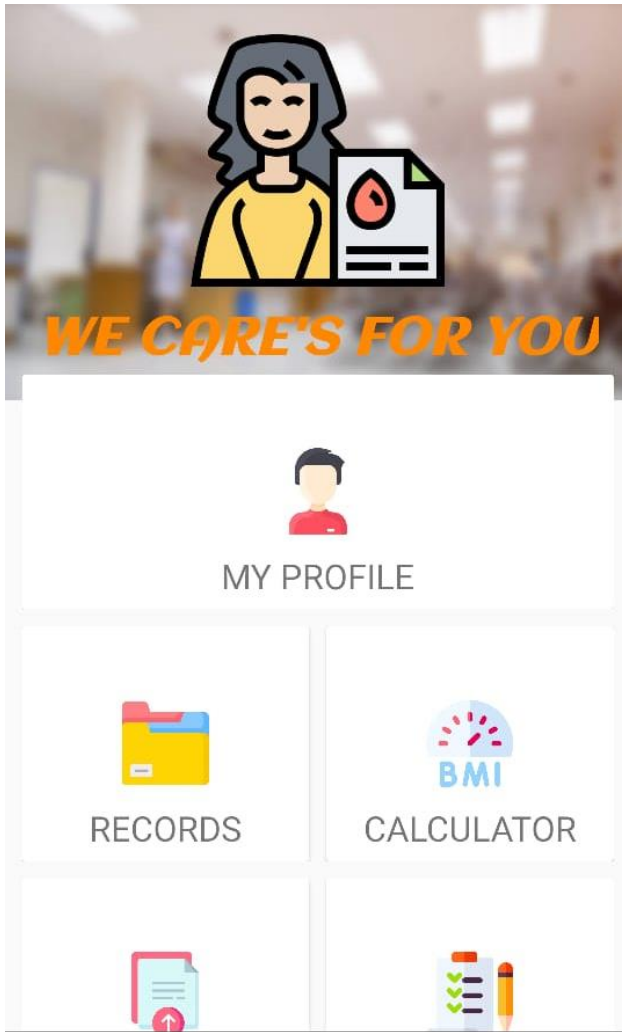
The Signup Page features a teal-to-blue gradient background. It contains five text input fields: 'First Name', 'Last Name', 'Email', 'Password', and 'Confirm Password'. Below these fields is a 'Gender' section with two radio buttons labeled 'Male' and 'Female'. At the bottom is a large, rounded, orange button with the text 'REGISTER' in bold black capital letters.

**Signin Page**



The Signin Page has a teal-to-blue gradient background. At the top is a circular logo featuring a caduceus and a hand holding a document, with the text 'PATIENT'S UNIQUE IDENTIFICATION' below it. Below the logo are two text input fields: 'Email' and 'Password'. The 'Password' field has an eye icon to its right. Below these fields is a large, rounded, orange button with the text 'SIGN IN' in bold black capital letters. At the bottom, there is a link that says 'Don't have an account? Create one' in a smaller, lighter blue font.

## Main Dashboard



# Webpage based Portal:

Successfully linked Project with MongoDB Cloud Storage (MongoDB Atlas): -

JS patient.jsJS auth.jsJS app.js

```
24
25
26
27
28
29 // middlewares
30 app.use(bodyParser.json()); //use url encoded instead of bodyParser.json()....go and check dis on bodyParser site,How to use it.
31 app.use(cookieParser());
32 app.use(cors());
33
34
35 //Routes
36 app.use("/api", authRoutes);
37
38
39
40
41 // Port
42 const port = process.env.PORT || 10000;
43
44 // starting server
45 app.listen(port, () => {
46 |   console.log('App is running at ${port}');
47 | });
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): \*.\*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
App is running at 10000
DB CONNECTED

Ln 27, Col 1 Spaces: 4 UTF-8 CRLF JavaScript Go Live Prettier

Mini Project - Patient... Access Manager Support Billing

All Clusters ABHINAV

Patient's Unique-Identifi... Atlas Realm Charts

DATA STORAGE

Clusters

Triggers

Data Lake

SECURITY

Database Access

Network Access

Advanced

MINI PROJECT - PATIENT'S PORTAL > PATIENT'S-UNIQUE-IDENTIFICATION > CLUSTERS

PUI

VERSION 4.2.10 REGION

Overview Real Time Metrics Collections Profiler Performance Advisor Online Archive BETA Command Line Tools

DATABASES: 1 COLLECTIONS: 1

+ Create Database

NAMESPACES

PUI

patients

PUI.patients

COLLECTION SIZE 0B TOTAL DOCUMENTS: 0 INDEXES TOTAL SIZE 20KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

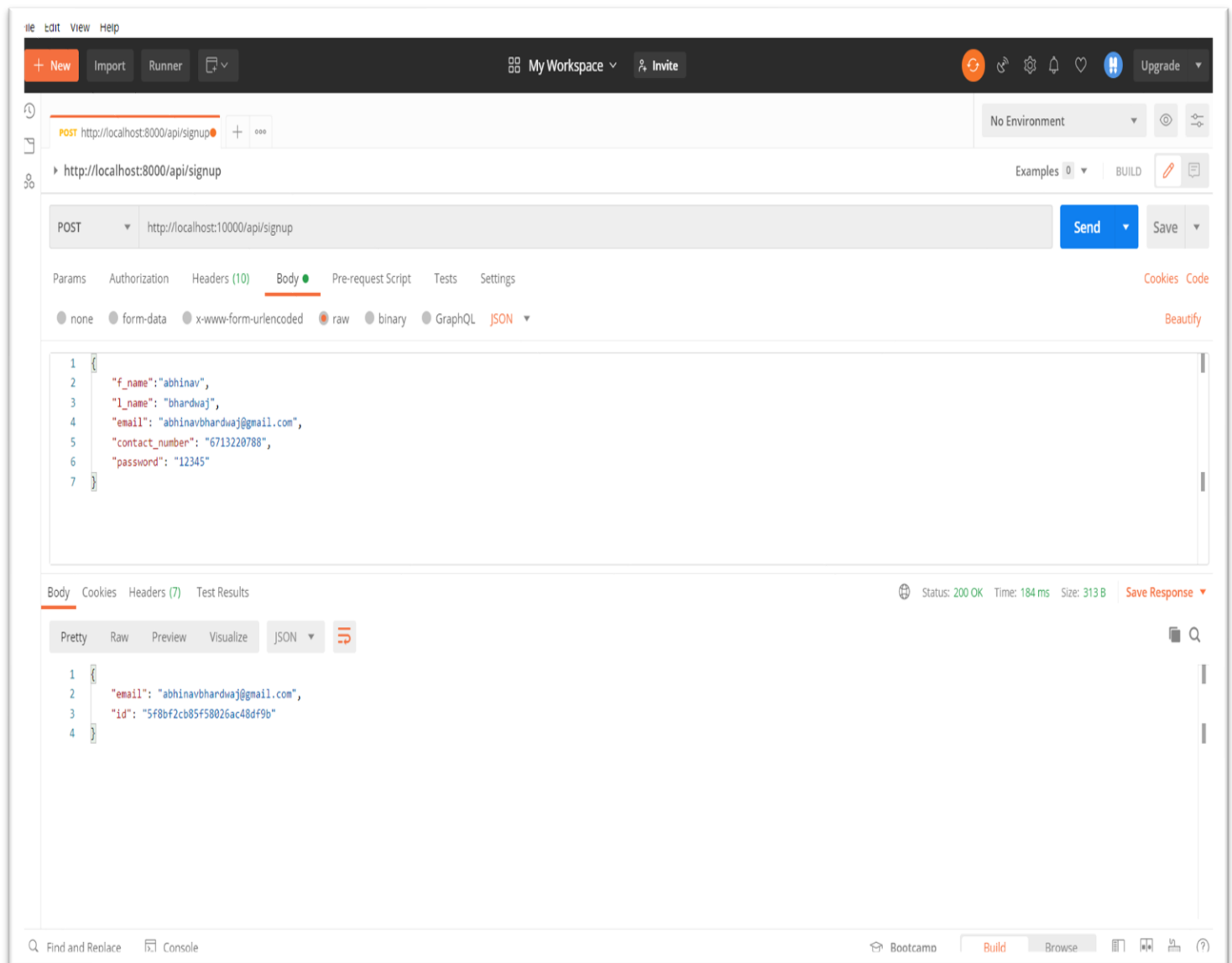
FILTER {"filter":"example"} Find Reset

QUERY RESULTS 0

Feature Requests

# Successful Signup using Postman:-

## 1<sup>st</sup> Patient



## 2<sup>nd</sup> Patient

The screenshot shows the Postman interface with a POST request to `http://localhost:8000/api/signup`. The request body is a JSON object with the following fields:

```
1 {
2   "f_name": "abhi",
3   "l_name": "rajput",
4   "email": "abhirajput@gmail.com",
5   "contact_number": "6713220799",
6   "password": "12345"
7 }
```

The response is a 200 OK status with a time of 170 ms and a size of 308 B. The response body is a JSON object with the following fields:

```
1 {
2   "email": "abhirajput@gmail.com",
3   "id": "5f8bf31485f58026ac48df9c"
4 }
```

## 3<sup>rd</sup> Patient

The screenshot shows the Postman interface with a POST request to `http://localhost:8000/api/signup`. The request body is a JSON object with the following fields:

```
1 {
2   "f_name": "abhi",
3   "l_name": "singh",
4   "email": "abhisingh@gmail.com",
5   "contact_number": "6713220777",
6   "password": "12345"
7 }
```

The response is a 200 OK status with a time of 164 ms and a size of 307 B. The response body is a JSON object with the following fields:

```
1 {
2   "email": "abhisingh@gmail.com",
3   "id": "5f8bf34985f58026ac48df9d"
4 }
```

QUERY RESULTS 1-3 OF 3

> `_id: ObjectId("5f8bf2cb85f58026ac48df9b")`  
`role: 0`  
`> documents: Array`  
`f_name: "abhinav"`  
`l_name: "bhardwaj"`  
`email: "abhinavbhardwaj@gmail.com"`  
`contact_number: "6713220788"`  
`salt: "02d68ce0-1116-11eb-846a-4ddb1f5eeb91"`  
`encry_password: "716340caaf391e1e1234f2d8bd89a9a1ce496139a4469532733b15c83b298571"`  
`createdAt: 2020-10-18T07:46:19.955+00:00`  
`updatedAt: 2020-10-18T07:46:19.955+00:00`  
`__v: 0`



`_id: ObjectId("5f8bf31485f58026ac48df9c")`  
`role: 0`  
`> documents: Array`  
`f_name: "abhi"`  
`l_name: "rajput"`  
`email: "abhirajput@gmail.com"`  
`contact_number: "6713220799"`  
`salt: "2e184fb0-1116-11eb-846a-4ddb1f5eeb91"`  
`encry_password: "81b383ec15d39d6f43075908cac399e38dacab9f2daf33aa0291316d4d5f5840"`  
`createdAt: 2020-10-18T07:47:32.527+00:00`  
`updatedAt: 2020-10-18T07:47:32.527+00:00`  
`__v: 0`

`_id: ObjectId("5f8bf34985f58026ac48df9d")`  
`role: 0`  
`> documents: Array`  
`f_name: "abhi"`  
`l_name: "singh"`  
`email: "abhisingh@gmail.com"`  
`contact_number: "6713220777"`  
`salt: "4d61cd60-1116-11eb-846a-4ddb1f5eeb91"`  
`encry_password: "446a3c3637e98042aa42b97e51b8ad9cf55de805278e0fb8feec176d24a9ad0"`  
`createdAt: 2020-10-18T07:48:25.015+00:00`



If tried to signup with duplicate data that is already in database, no entry in database

Postman interface showing a POST request to `http://localhost:8000/api/signup`. The request body is a JSON object:

```
{  "f_name": "abhi",  "l_name": "singh",  "email": "abhisingh@gmail.com",  "contact_number": "6713228777",  "password": "12345"}
```

The response status is `400 Bad Request` with a message: `"err": "NOT able to save patient in DB"`.

Atlas database interface showing query results for a patient database. The results show three documents, each with fields: `_id`, `role`, `documents`, `f_name`, `l_name`, `email`, `contact_number`, `salt`, `encry_password`, `createdAt`, `updatedAt`, and `__v`.

Document 1:

```
{  "_id": ObjectId("5f8bf2cb85f58026ac48df9b"),  "role": 0,  "documents": Array,  "f_name": "abhinav",  "l_name": "bhardwaj",  "email": "abhinavbhardwaj@gmail.com",  "contact_number": "6713228788",  "salt": "02d68ce0-1116-11eb-846a-4ddb1f5eeb91",  "encry_password": "716348caaf391e1e1234f2d8bd89a9a1ce496139a4469532733b15c83b298571",  "createdAt": 2020-10-18T07:46:19.955+00:00,  "updatedAt": 2020-10-18T07:46:19.955+00:00,  "__v": 0}
```

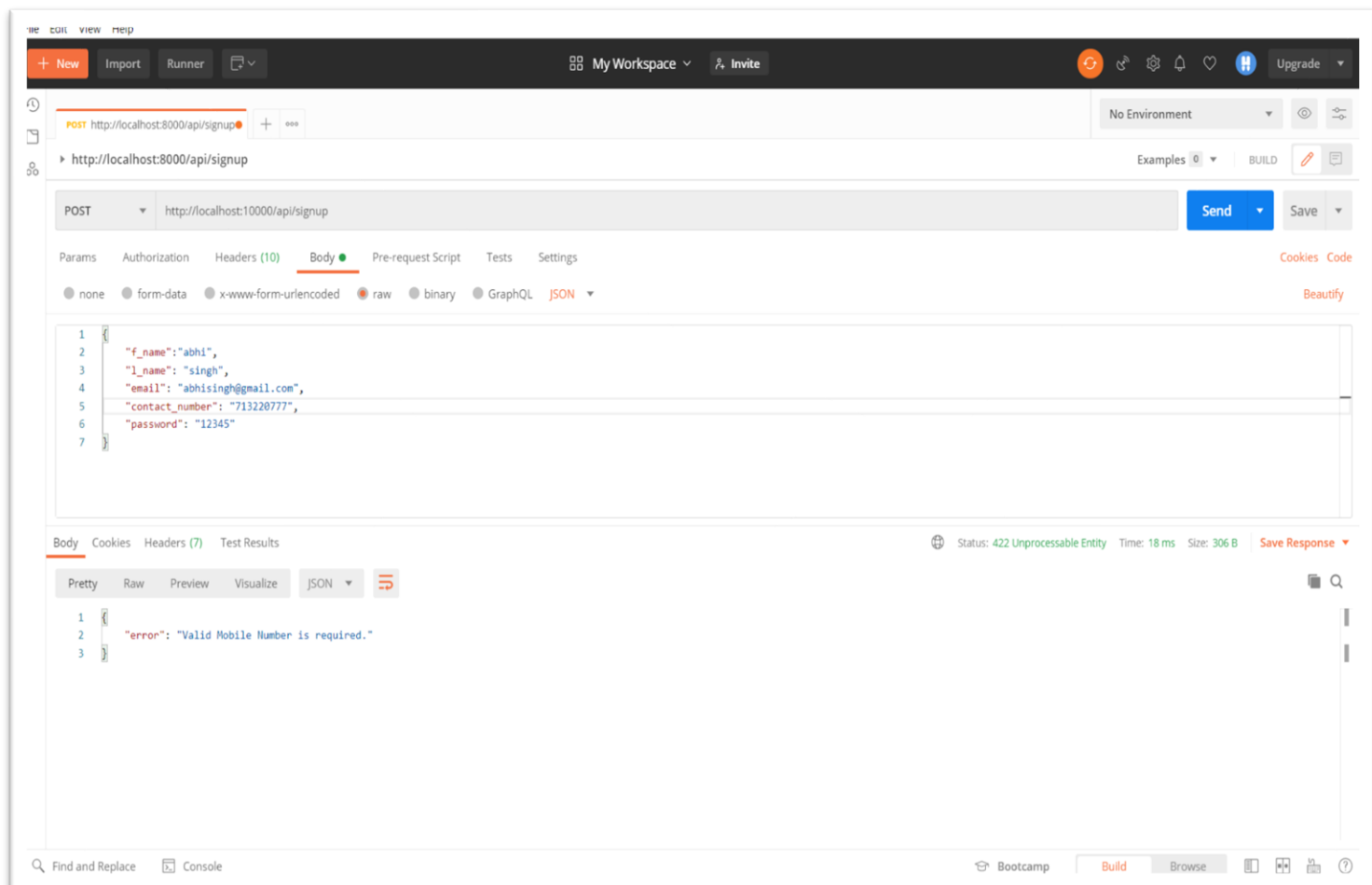
Document 2:

```
{  "_id": ObjectId("5f8bf31485f58026ac48df9c"),  "role": 0,  "documents": Array,  "f_name": "abhi",  "l_name": "rajput",  "email": "abhirajput@gmail.com",  "contact_number": "6713228799",  "salt": "2e184fb0-1116-11eb-846a-4ddb1f5eeb91",  "encry_password": "81b383ec15d39d6f43075980cac399e38dacab9f2daf33aa0291316d4d5f5840",  "createdAt": 2020-10-18T07:47:32.527+00:00,  "updatedAt": 2020-10-18T07:47:32.527+00:00,  "__v": 0}
```

Document 3:

```
{  "_id": ObjectId("5f8bf34985f58026ac48df9d"),  "role": 0,  "documents": Array,  "f_name": "abhi",  "l_name": "singh",  "email": "abhisingh@gmail.com",  "contact_number": "6713228777",  "salt": "4d61cd60-1116-11eb-846a-4ddb1f5eeb91",  "encry_password": "446a3c3637e98d42aa42b97e51b8ad9cf55de885278e0fb8feec176d24a9ad0",  "createdAt": 2020-10-18T07:48:25.015+00:00,  "updatedAt": 2020-10-18T07:48:25.015+00:00,  "__v": 0}
```

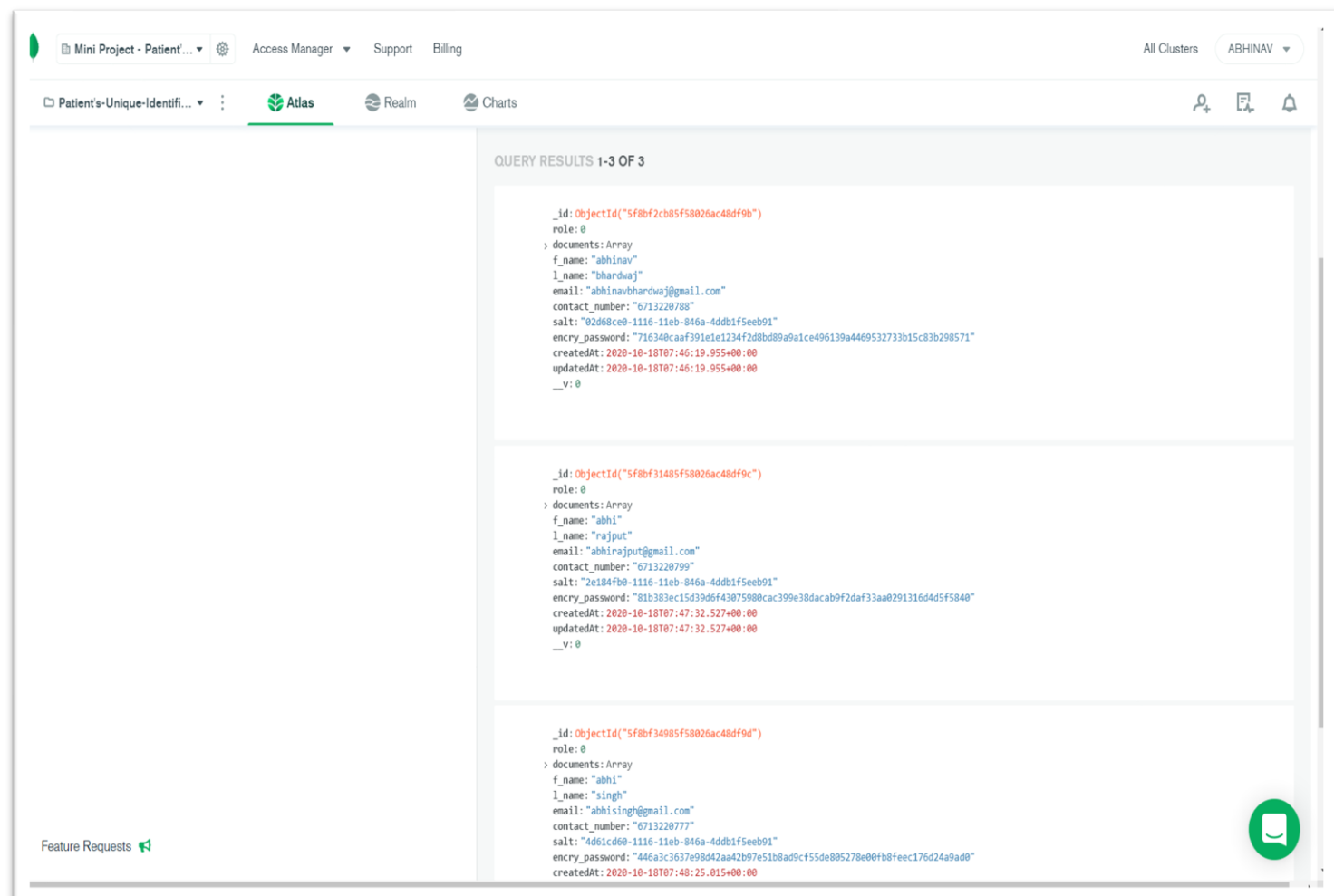
## If tried to signup with invalid mobile number, no entry in database



The screenshot shows a REST client interface with a POST request to `http://localhost:8000/api/signup`. The request body is a JSON object:

```
{  "f_name": "abhi",  "l_name": "singh",  "email": "abhisingh@gmail.com",  "contact_number": "713220777",  "password": "12345"}
```

The response status is **422 Unprocessable Entity** with a message: `"error": "Valid Mobile Number is required."`



The screenshot shows the MongoDB Atlas interface with the following query results:

QUERY RESULTS 1-3 OF 3

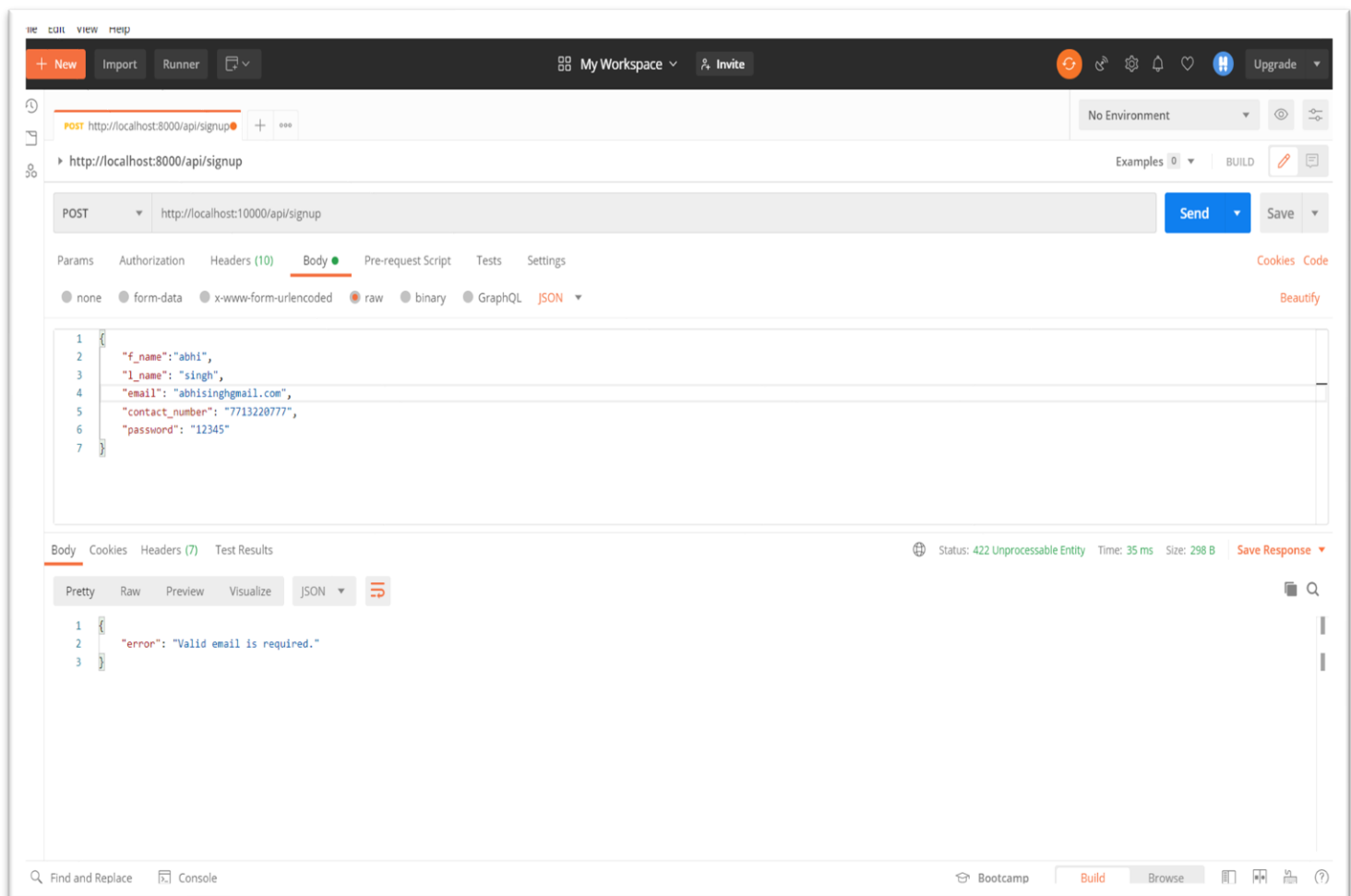
```
{  "_id": ObjectId("5f8bf2cb85f58026ac48df9b"),  "role": 0,  "documents": Array,  "f_name": "abhinav",  "l_name": "bhardwaj",  "email": "abhinavbhardwaj@gmail.com",  "contact_number": "6713220788",  "salt": "02d68ceb-1116-11eb-846a-4ddb1f5eeb91",  "encry_password": "716340caaf391e1e1234f2d8b089a9a1ce496139a4469532733b15c83b298571",  "createdAt": 2020-10-18T07:46:19.955+00:00,  "updatedAt": 2020-10-18T07:46:19.955+00:00,  "__v": 0}
```

```
{  "_id": ObjectId("5f8bf31485f58026ac48df9c"),  "role": 0,  "documents": Array,  "f_name": "abhi",  "l_name": "rajput",  "email": "abhirajput@gmail.com",  "contact_number": "6713220799",  "salt": "2e184f0b-1116-11eb-846a-4ddb1f5eeb91",  "encry_password": "81b383ec15d39d6f43075980cac399e38dacab9f2daf33aa0291316d4d5f5840",  "createdAt": 2020-10-18T07:47:32.527+00:00,  "updatedAt": 2020-10-18T07:47:32.527+00:00,  "__v": 0}
```

```
{  "_id": ObjectId("5f8bf34985f58026ac48df9d"),  "role": 0,  "documents": Array,  "f_name": "abhi",  "l_name": "singh",  "email": "abhisingh@gmail.com",  "contact_number": "6713220777",  "salt": "4d61cd60-1116-11eb-846a-4ddb1f5eeb91",  "encry_password": "446a3c3637e98d42aa42b7e51b8ad9cf55de085278e0fb8feec176d24a9ad0",  "createdAt": 2020-10-18T07:48:25.015+00:00}
```



## If tried to signup with invalid email address, no entry in database



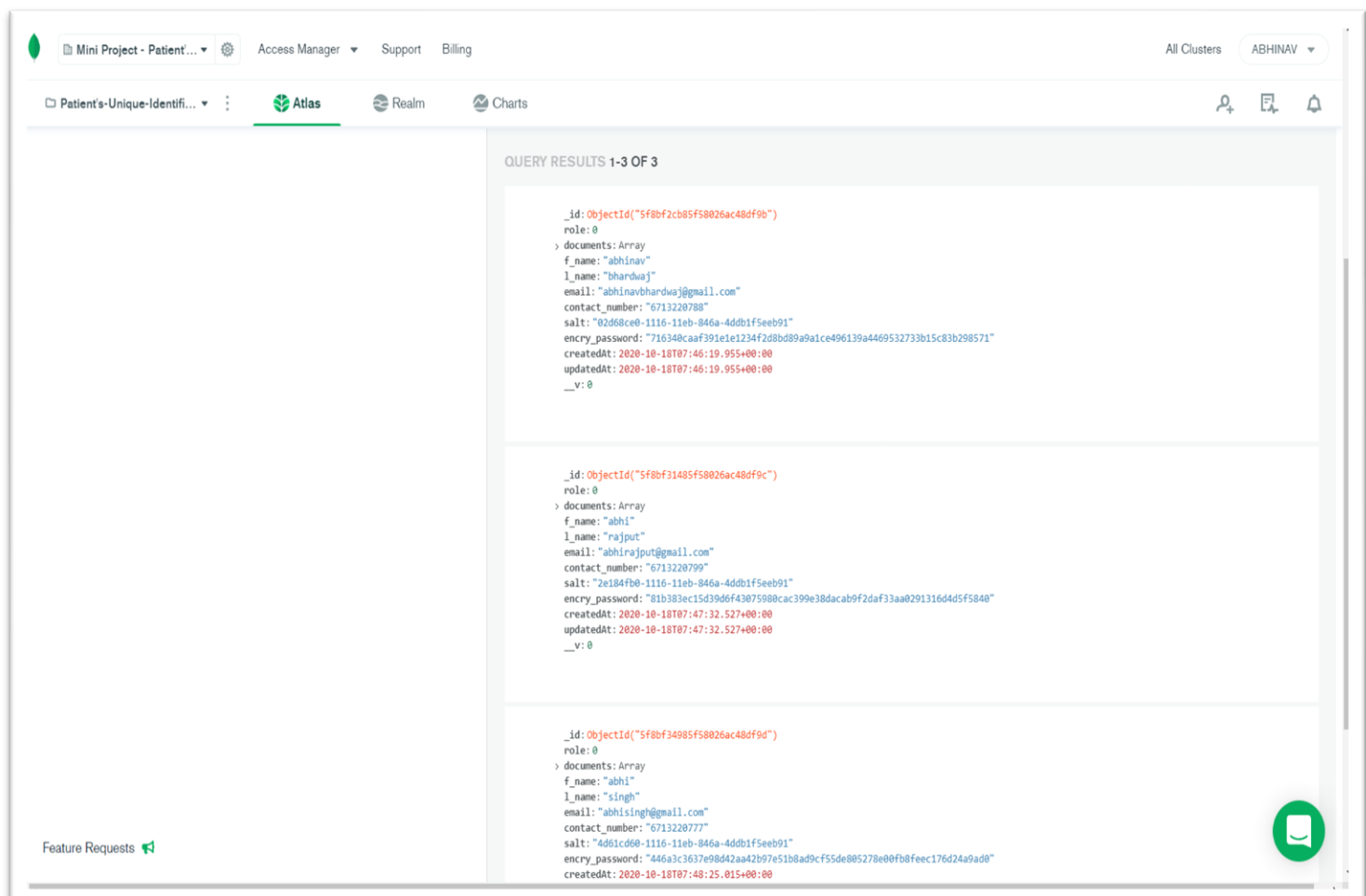
The screenshot shows the Postman interface for a POST request to `http://localhost:8000/api/signup`. The request body is a JSON object:

```
{  "f_name": "abhi",  "l_name": "singh",  "email": "abhisinh@gmail.com",  "contact_number": "7713220777",  "password": "12345"}
```

The response is a JSON object with an error message:

```
{  "error": "Valid email is required."}
```

The status is 422 Unprocessable Entity, Time: 35 ms, Size: 298 B.



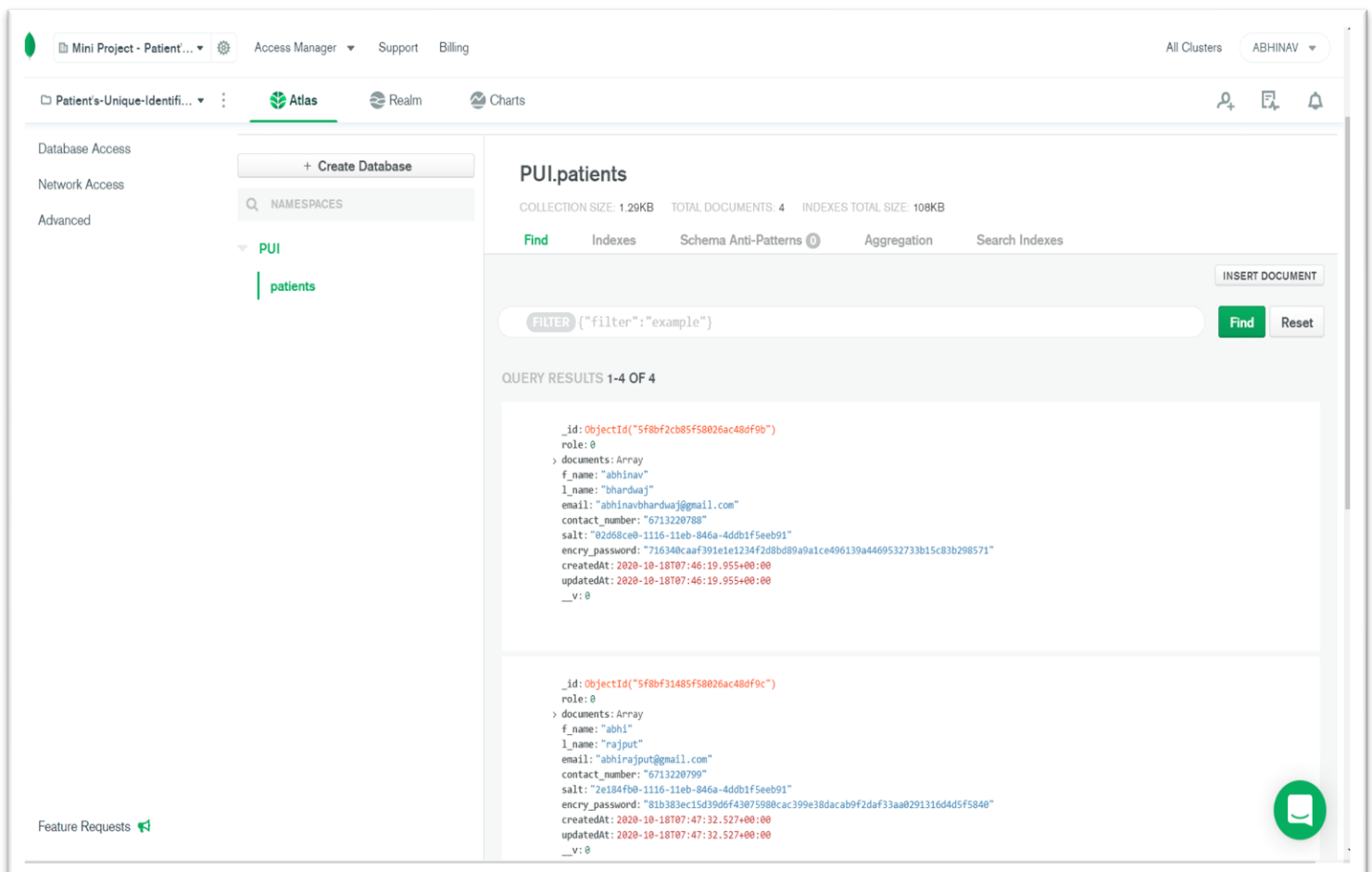
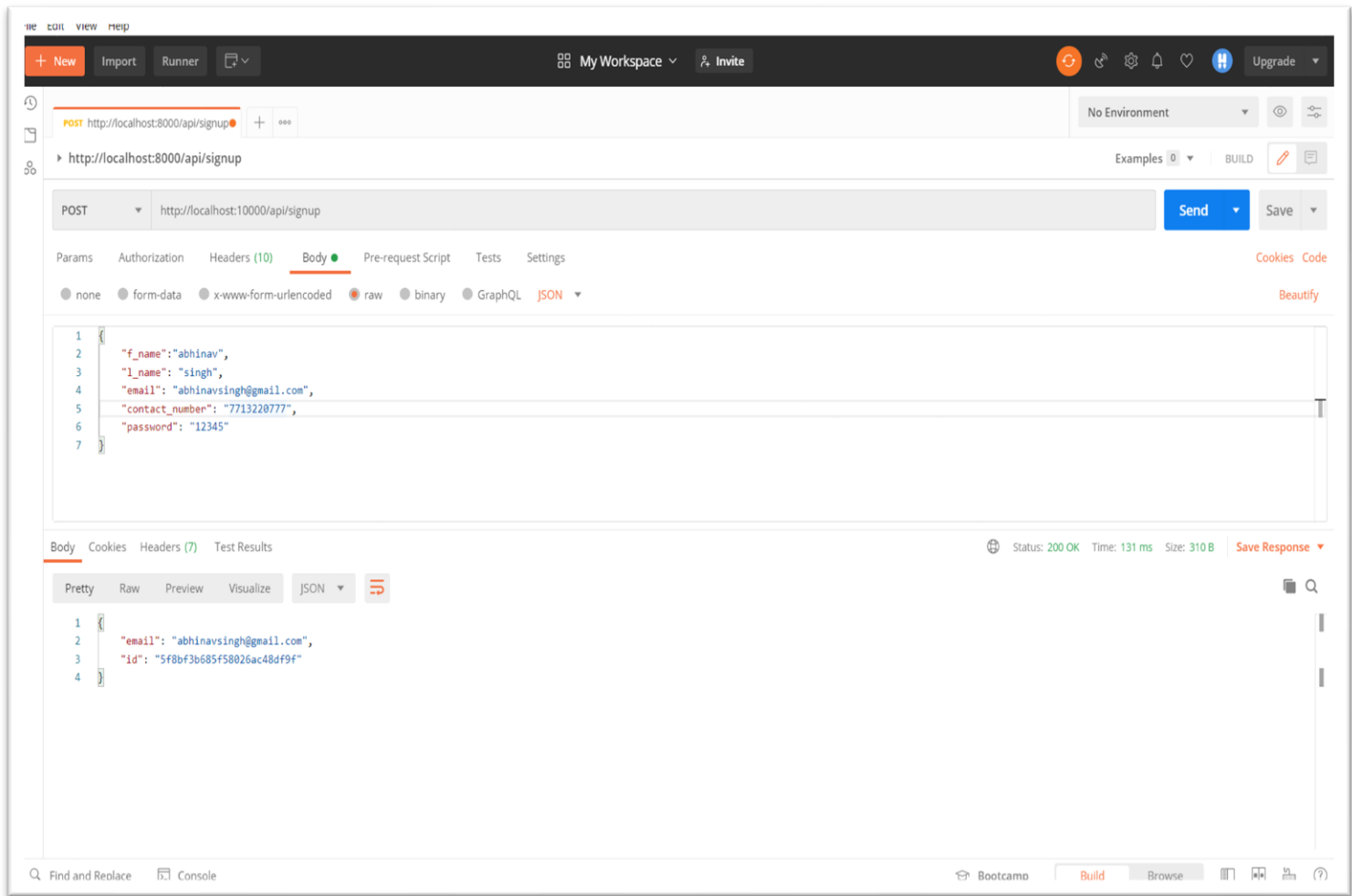
The screenshot shows the Atlas database interface. The query results show three documents:

```
{  "_id": ObjectId("5f8bf2cb85f58026ac48df9b"),  "role": 0,  "documents": Array,  "f_name": "abhinav",  "l_name": "bhardwaj",  "email": "abhinavbhardwaj@gmail.com",  "contact_number": "6713220788",  "salt": "02d68ce0-1116-11eb-846a-4ddb1f5eeb91",  "encry_password": "716340caf391e1c1234f2d0bd09a9a1ce496139a4469532733b15c83b298571",  "createdAt": 2020-10-18T07:46:19.955+00:00,  "updatedAt": 2020-10-18T07:46:19.955+00:00,  "__v": 0}
```

```
{  "_id": ObjectId("5f8bf31485f58026ac48df9c"),  "role": 0,  "documents": Array,  "f_name": "abhi",  "l_name": "rajput",  "email": "abhirajput@gmail.com",  "contact_number": "6713220799",  "salt": "2e184fb0-1116-11eb-846a-4ddb1f5eeb91",  "encry_password": "81b383ec15d39d6f43075908cac399e38dacab9f2daf33aa0291316d4d5f5840",  "createdAt": 2020-10-18T07:47:32.527+00:00,  "updatedAt": 2020-10-18T07:47:32.527+00:00,  "__v": 0}
```

```
{  "_id": ObjectId("5f8bf34985f58026ac48df9d"),  "role": 0,  "documents": Array,  "f_name": "abhi",  "l_name": "singh",  "email": "abhisinh@gmail.com",  "contact_number": "6713220777",  "salt": "4d61cd60-1116-11eb-846a-4ddb1f5eeb91",  "encry_password": "446a3c3637e98d42aa42b97e51b8ad9cf55de805278e00fb0feec176d24a9ad0",  "createdAt": 2020-10-18T07:48:25.015+00:00,  "updatedAt": 2020-10-18T07:48:25.015+00:00,  "__v": 0}
```

And, again signup with proper details



```
f_name: "abhi"
l_name: "rajput"
email: "abhirajput@gmail.com"
contact_number: "6713220799"
salt: "2e184fb0-1116-11eb-846a-4ddb1f5eeb91"
encry_password: "81b383ec15d39d6f43075980cac399e38dacab9f2daf33aa0291316d4d5f5840"
createdAt: 2020-10-18T07:47:32.527+00:00
updatedAt: 2020-10-18T07:47:32.527+00:00
_v: 0
```

> `_id: ObjectId("5f8bf34985f58026ac48df9d")`    

```
role: 0
documents: Array
  f_name: "abhi"
  l_name: "singh"
  email: "abhisingh@gmail.com"
  contact_number: "6713220777"
  salt: "4d61cd60-1116-11eb-846a-4ddb1f5eeb91"
  encry_password: "446a3c3637e98d42aa42b97e51b8ad9cf55de085278e0fb0feec176d24a9ad0"
  createdAt: 2020-10-18T07:48:25.015+00:00
  updatedAt: 2020-10-18T07:48:25.015+00:00
  _v: 0
```

```
_id: ObjectId("5f8bf3b685f58026ac48df9f")
role: 0
documents: Array
  f_name: "abhinav"
  l_name: "singh"
  email: "abhinavsingh@gmail.com"
  contact_number: "7713220777"
  salt: "8e663df0-1116-11eb-846a-4ddb1f5eeb91"
  encry_password: "39ca6cf6a33a600dd84678cc571303e151f0cfa81edd33ecf88942f900bea26f"
  createdAt: 2020-10-18T07:50:14.096+00:00
  updatedAt: 2020-10-18T07:50:14.096+00:00
  _v: 0
```



# Successful signed in with correct credentials

The screenshot shows the Postman interface with a POST request to `http://localhost:9000/api/signin`. The request body is a JSON object:

```
{  "email": "abhinavbhardwaj@gmail.com",  "password": "12345"}
```

The response is a JSON object with a token and patient information:

```
{  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfcm9udGkiOiJhbnR5cCI6IkpXVCJ9.eyJfcm9udGkiOiJhbnR5cCI6IkpXVCJ9",  "patient": {    "_id": "5f8bf2cb85f58026ac48df9b",    "email": "abhinavbhardwaj@gmail.com",    "role": 0  }  }
```

The status is 200 OK, Time: 457 ms, Size: 673 B.

The screenshot shows the MongoDB Atlas interface for a cluster named `pui-shard-00-02.0fxmk.mongodb.net:27017`. The cluster is in the `Electable` state, version `4.2.10`, and located in the `Mumbai (ap-south-1)` region.

The `Status` tab is selected, showing a line graph of `Opcounters` over time. The graph shows a significant spike in the `command` counter around 07:10. The legend on the right defines the counters:

- command**: The average rate of commands performed per second over the selected sample period
- query**: The average rate of queries performed per second over the selected sample period
- update**: The average rate of updates performed per second over the selected sample period
- delete**: The average rate of deletes performed per second over the selected sample period
- getmore**: The average rate of getMores performed per second on any cursor over the selected sample period. On a primary, this number can be high even

System Status: All Good Last Login: 47.247.181.159

©2020 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

# Signing out

The screenshot shows the Postman interface with a GET request to `http://localhost:9000/api/signout` successfully executed. The response is a JSON object: `{ "message": "User signout successfully" }`. The status is 200 OK, time is 50 ms, and size is 350 B.

**Request Details:**

- Method: GET
- URL: `http://localhost:9000/api/signout`

**Response Details:**

- Status: 200 OK
- Time: 50 ms
- Size: 350 B
- Message: "User signout successfully"

The screenshot shows the MongoDB Atlas dashboard for a cluster named "Mini Project - Patient...". The selected shard is "pui-shard-00-02.0fxmk.mongodb.net:27017". The dashboard displays a line graph of Opcounters over time, with a peak around 07:10. The graph shows the average rate of commands, queries, updates, deletes, and getMores performed per second over the selected sample period.

**Cluster Information:**

- Cluster Name: Mini Project - Patient...
- Shard Name: pui-shard-00-02.0fxmk.mongodb.net:27017
- Type: Electable
- Version: 4.2.10
- Region: Mumbai (ap-south-1)

**Opcounters Graph:**

- Y-axis: Opcounters (0/S to 1.2/S)
- X-axis: Time (07:00 to 07:30)
- Legend: command, query, update, delete, getmore

**System Status:** All Good. Last Login: 47.247.181.150

**Footer:** ©2020 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

# Trying to sign in with incorrect email address that does not exist in database

The screenshot shows the Postman application interface. At the top, there's a menu bar with 'File', 'Edit', 'View', and 'Help'. Below it, a toolbar contains buttons for '+ New', 'Import', 'Runner', and a dropdown menu. The main workspace area shows a collection of requests. The selected request is a POST to 'http://localhost:9000/api/signin'. The request body is a JSON object: 

```
{  "email": "abhinavbhardaj@gmail.com",  "password": "12345"}
```

. The response is displayed in the bottom pane, showing a status of '400 Bad Request' with the message: 

```
{  "error": "Patient email does not exists"}
```

. The status bar at the bottom indicates 'Status: 400 Bad Request', 'Time: 693 ms', and 'Size: 294 B'.

The screenshot shows the MongoDB Atlas console. The top navigation bar includes 'Mini Project - Patient...', 'Access Manager', 'Support', and 'Billing'. The main content area displays the 'pui-shard-00-02.0fxmk.mongodb.net:27017' cluster. The 'Status' section shows a line graph of 'Opcounters' over time, with a peak around 07:10. The graph is titled 'Opcounters' and shows the average rate of commands performed per second over the selected sample period. The legend on the right lists the following counters: 

- command**: The average rate of commands performed per second over the selected sample period
- query**: The average rate of queries performed per second over the selected sample period
- update**: The average rate of updates performed per second over the selected sample period
- delete**: The average rate of deletes performed per second over the selected sample period
- getmore**: The average rate of getMores performed per second on any cursor over the selected sample period. On a primary, this number can be high even

 The bottom of the console shows the 'System Status' as 'All Good' and the 'Last Login' as '47.247.181.159'.

# Trying to sign in with incorrect password

The screenshot shows a REST client interface with a workspace containing three requests. The selected request is a POST to `http://localhost:10000/api/signin`. The body is a JSON object with `"email": "abhinavbhardwaj@gmail.com"` and `"password": "12356"`. The response is displayed in the bottom panel, showing a JSON object with `"error": "Email and password do not match"`. The status is 401 Unauthorized, with a time of 694 ms and a size of 297 B.

```
POST http://localhost:10000/api/signin
```

```
{  "email": "abhinavbhardwaj@gmail.com",  "password": "12356"}
```

Body | Cookies | Headers (7) | Test Results

Status: 401 Unauthorized Time: 694 ms Size: 297 B Save Response

```
{  "error": "Email and password do not match"}
```

The screenshot shows the MongoDB Atlas monitoring dashboard for a specific shard: `pui-shard-00-02.0fxmk.mongodb.net:27017`. The dashboard displays a line chart for 'Opcounters' over a 1-hour period. The chart shows a significant spike in the 'command' counter around 07:40. The right sidebar provides definitions for the counters: **command** (average rate of commands), **query** (average rate of queries), **update** (average rate of updates), **delete** (average rate of deletes), and **getmore** (average rate of getMores). The system status is 'All Good' and the last login was at 47.247.181.159.

Mini Project - Patient's Portal

Atlas | Realm | Charts

DATA STORAGE

- Clusters
- Triggers
- Data Lake

SECURITY

- Database Access
- Network Access
- Advanced

Mini Project - Patient's Portal > Patient's Unique Identification > PUI-SHARD-00-02.0FXMK.MONGODB.NET:27017

**pui-shard-00-02.0fxmk.mongodb.net:27017**

TYPE: Electable VERSION: 4.2.10 REGION: Mumbai (ap-south-1)

Status

GRANULARITY: Auto ZOOM: 1 hour CURRENT DISPLAY: 10/18/2020 07:07am TO 10/18/2020 08:07am AT 1 MINUTE GRANULARITY

Opcounters

command: The average rate of commands performed per second over the selected sample period

query: The average rate of queries performed per second over the selected sample period

update: The average rate of updates performed per second over the selected sample period

delete: The average rate of deletes performed per second over the selected sample period

getmore: The average rate of getMores performed per second on any cursor over the selected sample period. On a primary, this number can be high even

System Status: All Good Last Login: 47.247.181.159

© 2020 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales