## **Backend API Documentation for Animimic Web project**

**Backend Tech stack:**

* MongoDB, Node JS/Express JS

**API Details:**

1. **For new user registration:**

POST <https://animimic.onrender.com/api/register>

Sample input for user registration:

{

"name":"Ramesh Jindal",

"mobile": "9840690988",

"username": "rameshj",

"email":"ramesh.jindal@gmail.com",

"password":"ramesh90"

}

Please note: Other parameters as per the User Schema can also be sent in the user registration itself. Alternatively, other parameters of User Schema can sent as a separate request as update request by invoking the update API.

1. **For user login:**

POST <https://animimic.onrender.com/api/login>

Login will be based on email and password. On successful login, a token will be returned to the front-end in API response. The token has to be used in all future API calls except login and register APIs. The mongodb autogenerated userid will be returned along with token in the response.

Sample input for user login:

{

"email": "ramesh.jindal@gmail.com",

"password":"ramesh90"

}

1. **For User details update**

PUT <https://animimic.onrender.com/api/update>

Email is mandatory to identify the user in database table. Name, Mobile, Security Question(secquestion) and Security Answer(secanswer) fields can be updated through the update api call.

Sample data for user details update:

{

"email": "ramesh.jindal@gmail.com",

"secquestion1":"Name of your first pet?",

"secanswer1": "jilla"

}

1. **For Password reset**

POST <https://animimic.onrender.com/api/reset>

The secanswer is mandatory field to validate before updating the new password for the user. Secanswer can be any one of the 3 security answers set by the user earlier. Email and Secuanswer are the required parameters for password reset.

Sample input for password reset:

{

"email":"test20@example.com",

"password":"testpassword20",

"secquestion": "secquestion",

"secanswer": "secanswer"

}

1. **For updating a blog:**

<https://animimic.onrender.com/api/updateblog>

Please use the Blogs schema given by Niharika to update the blog. The commented ‘usercomments’ section of the Blogs schema can also be used instead of ‘comments’ field.

Sample input for updating a blog

{

    "blog\_id": "672e401902e529870a894b9d",

    "title": "test2.2",

    "banner": "test2.3",

    "des": "test2.1",

    "content": "test2.1",

    "tags": "test2.1",

    "author": "6703f19ddab5de393f4252f7",

    "activity": {

      "total\_likes": "test",

      "total\_comments": "2",

      "total\_reads": "2",

      "total\_parent\_comments": "2"

    },

    "comments": "672e32f31f33e599710e1d01",

    "usercomments":[

        {"comment": "mycomment", "email": "MMS@gmail.com", "user": "672dc8c63d50739365b6bde6" },

         {"comment": "mycomment2", "email": "MMS2@gmail.com", "user": "672dc8c63d50739365b6bde6" }

    ],

    "draft": "true"

  }

1. **For Deleting a blog:**

<https://animimic.onrender.com/api/deleteblog>

Sample input for delete blog api call is:

{

    "id":"672e32f31f33e599710e1d01"

}

1. **For getting a blog by Author and blog\_id**

<https://animimic.onrender.com/api/getblog>

sample input to get blogs by Author

{

"author": "6703f19ddab5de393f4252f7"

}

Sample input to get blogs by Author and banner

{

" banner ": "test53",

"author": "6703f19ddab5de393f4252f7"

}

Sample input to get blogs by Banner

{

"banner": "test2.3"

}

1. **For getting all blogs**

https://animimic.onrender.com/api/allblogs

Sample input to get all blogs

{}

1. **For blog creation**

https://animimic.onrender.com/api/blog

Sample input to create blog

{

    "blog\_id": "test55",

    "title": "test444",

    "banner": "test54",

    "des": "test",

    "content": "test",

    "tags": "test",

    "author": "6703f19ddab5de393f4252f7",

    "activity": {

      "total\_likes": "0",

      "total\_comments": "0",

      "total\_reads": "0",

      "total\_parent\_comments": "0"

    },

    "comments": "6703f19ddab5de393f4252f7",

    "draft": "true"

  }

1. **New API added on 13-11-24 to delete user**

DELETE <https://animimic.onrender.com/api/delete>

Sample input to delete a blog

{

"id":"672f6599909a74daf931b7e4"

}

Note: The id is the id that was received in response to the successful user login.

### User Schema:

const userSchema = mongoose.Schema({

name: String,

mobile: String,

username:

{

type: String,

//required: true,

unique: true

 },

email:

{

type: String,

//required: true,

unique: true

},

password: String,

secquestion1: String,

secanswer1: String,

secquestion2: String,

secanswer2: String,

secquestion3: String,

secanswer3: String,

photolink: String,

account\_info:

{

    total\_posts:

{

type: Number,

default: 0

    },

    total\_reads:

{

type: Number,

default: 0

    },

},

blogs:

{

    type: [ Schema.Types.ObjectId ],

    ref: 'blogs',

    default: [],

},

});

module.exports = mongoose.model('User', userSchema);

### Blogs schema:

const mongoose = require("mongoose");

const { Schema } = mongoose;

const blogSchema = mongoose.Schema (

{

    blog\_id:

{

        type: String,

        required: true,

        unique: true,

   },

title: {

        type: String,

        required: true,

    },

banner:

{

        type: String,

        // required: true,

},

des:

{

        type: String,

        maxlength: 200,

        // required: true

},

  content:

{

      type: [],

        // required: true

},

    tags:

{

    type: [String],

        // required: true

},

author:

{

      type: Schema.Types.ObjectId,

        required: true,

        ref: 'users'

  },

activity:

{

total\_likes:

{

  type: Number,

  default: 0

},

total\_comments:

{

  type: Number,

  default: 0

},

total\_reads:

{

    type: Number,

    default: 0

},

total\_parent\_comments:

{

      type: Number,

      default: 0

},

    },

comments:

{

type: [Schema.Types.ObjectId],

ref: 'comments'

},

usercomments:

{

type: [],

comment: String,

email: String,

user:Schema.Types.ObjectId,

  },

  draft:

{

type: Boolean,

default: false

  }

},

{

    timestamps:

{

  createdAt: 'publishedAt'

    }

})

module.exports = mongoose.model("Blogs", blogSchema);