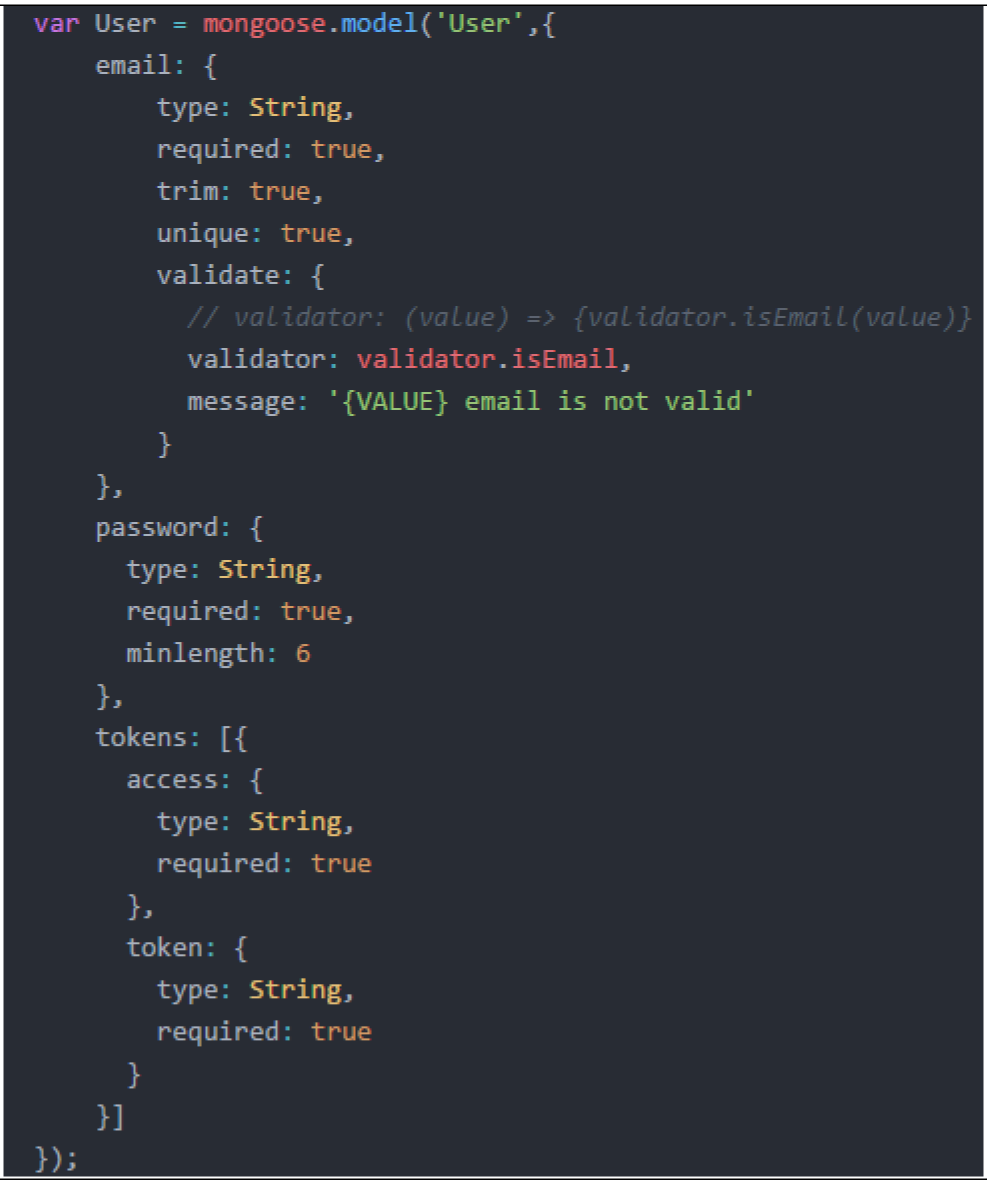
Authentication

Security and Authentication in Node.js

Project
MongoDB To-do App
Install
npm install mongoose --save npm install validator --save
References
https://www.npmjs.com/package/validator http://mongoosejs.com/docs/validation.html

Created by: Wael Zoaiter - Github

isEmail



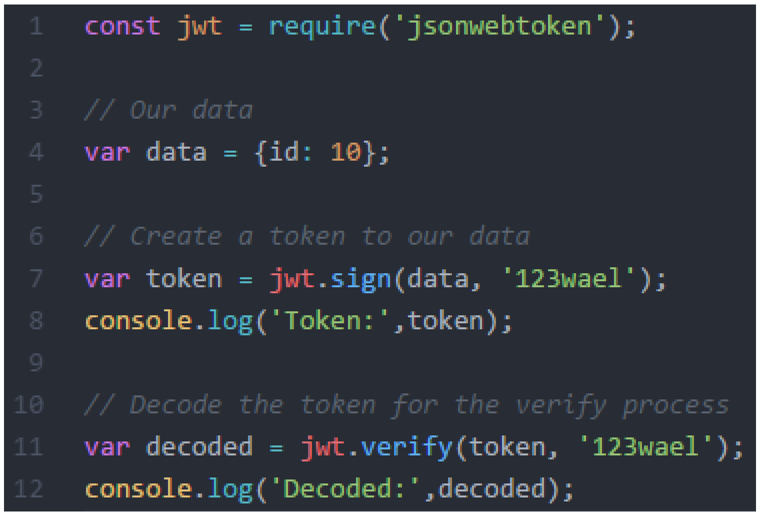
```
var User = mongoose.model('User',{
    email: {
        type: String,
        required: true,
        trim: true,
        unique: true,
        validate: {
          // validator: (value) => {validator.isEmail(value)}
          validator: validator.isEmail,
          message: '{VALUE} email is not valid'
        }
    },
    password: {
      type: String,
      required: true,
      minlength: 6
    },
    tokens: [{
      access: {
        type: String,
        required: true
      },
      token: {
        type: String,
        required: true
      }
    }]
});
```

Created by: Wael Zoaiter - Github

J.W.T

JSON Web Token

Sign and Verify



```
const jwt = require('jsonwebtoken');

// Our data
var data = {id: 10};

// Create a token to our data
var token = jwt.sign(data, '123wael');
console.log('Token:',token);

// Decode the token for the verify process
var decoded = jwt.verify(token, '123wael');
console.log('Decoded:',decoded);
```

jwt**.sign(**data**,** secret key**) : Hashing our data with our secret key.**

jwt**.verify(**token**,** secret key**) : Encode our token with our secret key.**

General

Instance Method:

Instances of Models are documents. Documents have many of their own built-in instance methods. We may also define our own custom document instance methods too.

`e.g. userSchema.methods.toJSON = function() {…}`

JSON:

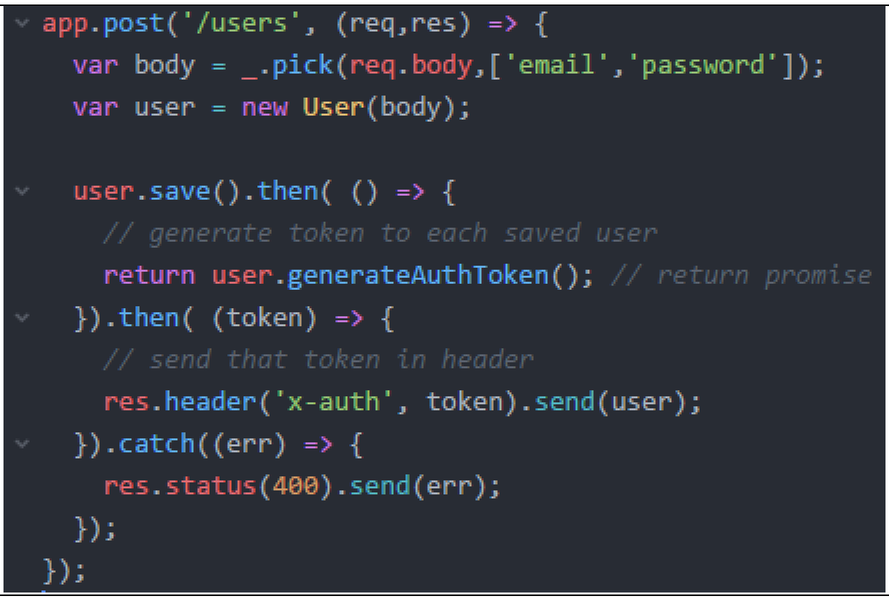
`JSON.stringify(value, undefined, space)` method converts a JavaScript value to a JSON string.

`.toJSON()` convert an object into JSON string.

Created by: Wael Zoaiter - Github

User Authentication

server-side

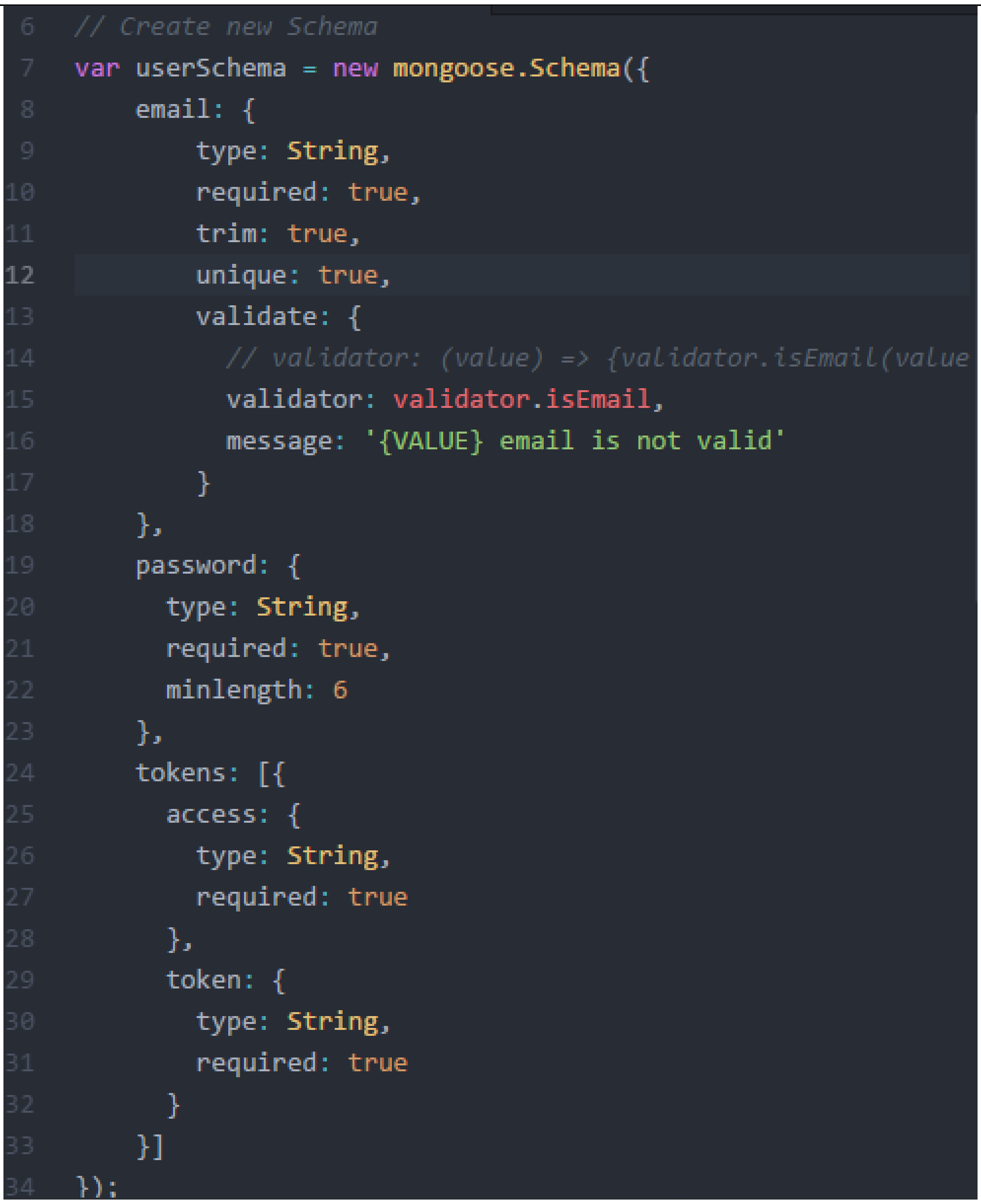


```
app.post('/users', (req,res) => {
  var body = _.pick(req.body,['email','password']);
  var user = new User(body);

  user.save().then( () => {
    // generate token to each saved user
    return user.generateAuthToken(); // return promise
  }).then( (token) => {
    // send that token in header
    res.header('x-auth', token).send(user);
  }).catch((err) => {
    res.status(400).send(err);
  });
});
```

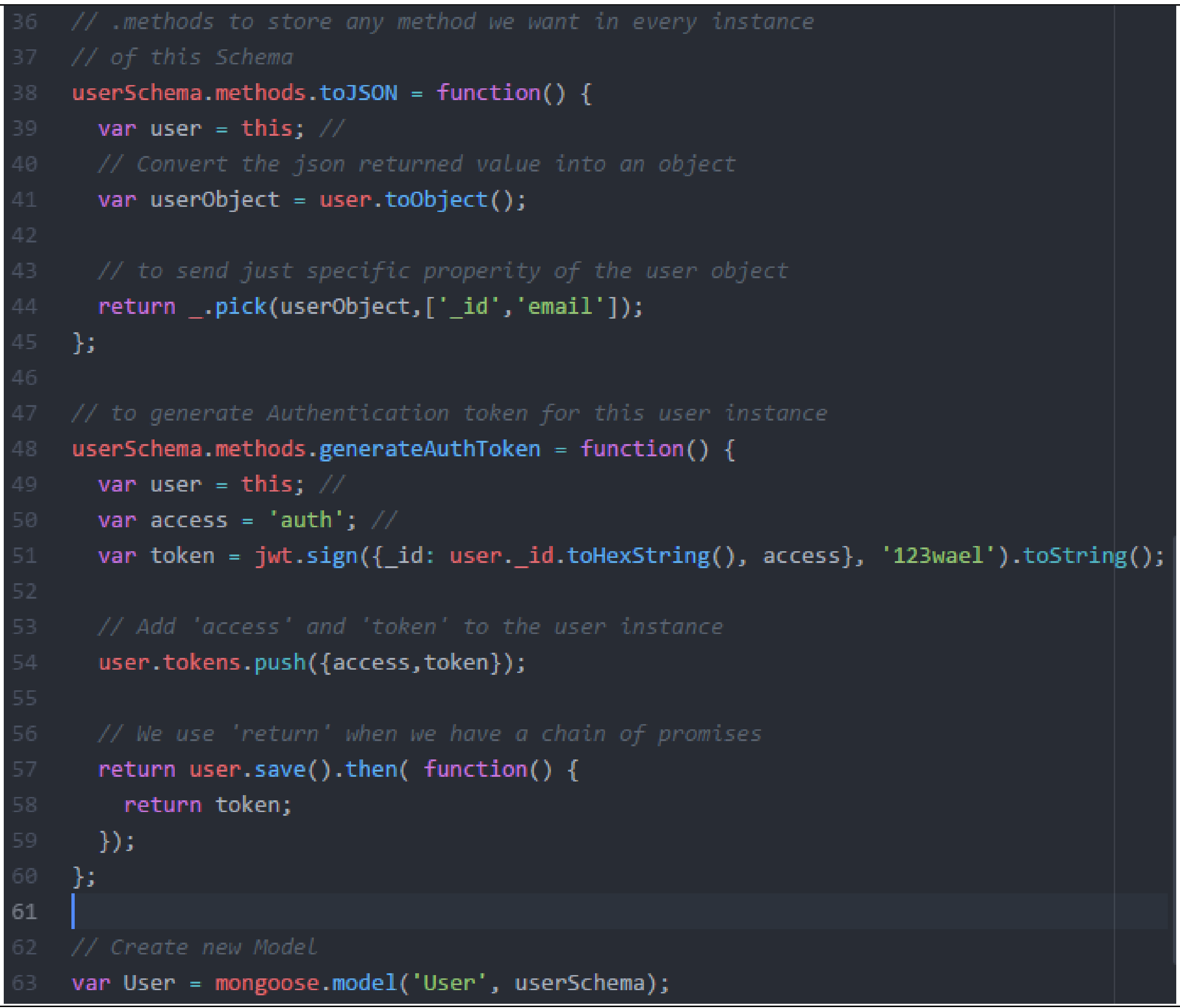
user-model

Created by: Wael Zoaiter - Github



```
// Create new Schema
var userSchema = new mongoose.Schema({
    email: {
        type: String,
        required: true,
        trim: true,
        unique: true,
        validate: {
          // validator: (value) => {validator.isEmail(value
          validator: validator.isEmail,
          message: '{VALUE} email is not valid'
        }
    },
    password: {
      type: String,
      required: true,
      minlength: 6
    },
    tokens: [{
      access: {
        type: String,
        required: true
      },
      token: {
        type: String,
        required: true
      }
    }]
});
```

Created by: Wael Zoaiter - Github



```
// .methods to store any method we want in every instance
// of this Schema
userSchema.methods.toJSON = function() {
  var user = this; //
  // Convert the json returned value into an object
  var userObject = user.toObject();

  // to send just specific properity of the user object
  return _.pick(userObject,['_id','email']);
};

// to generate Authentication token for this user instance
userSchema.methods.generateAuthToken = function() {
  var user = this; //
  var access = 'auth'; //
  var token = jwt.sign({_id: user._id.toHexString(), access}, '123wael').toString();

  // Add 'access' and 'token' to the user instance
  user.tokens.push({access,token});

  // We use 'return' when we have a chain of promises
  return user.save().then( function() {
    return token;
  });
};

// Create new Model
var User = mongoose.model('User', userSchema);
```

Created by: Wael Zoaiter - Github