```

// main.js

import express from 'express';

import bodyParser from 'body-parser';

import mongoose from 'mongoose';

import cors from 'cors';

import apiRoutes from './routes/api';

const app = express();

app.use(cors());

app.use(bodyParser.json());

mongoose.connect('mongodb://localhost/banana-cloud', { useNewUrlParser: true });

mongoose.connection.on('error', (err) => {

console.error(`MongoDB connection error: ${err}`);

process.exit(-1);

});

app.use('/api', apiRoutes);

const server = app.listen(3000, () => {

console.log(`Banana Cloud API listening on port ${server.address().port}`);

});

```

```

// models/user.js

import mongoose from 'mongoose';

const userSchema = new mongoose.Schema({

username: String,

email: String,

password: String,

created\_at: { type: Date, default: Date.now },

updated\_at: { type: Date, default: Date.now }

});

export default mongoose.model('User', userSchema);

```

```

// routes/api.js

import express from 'express';

import bcrypt from 'bcryptjs';

import jwt from 'jsonwebtoken';

import User from '../models/user';

const router = express.Router();

router.post('/signup', (req, res) => {

const { username, email, password } = req.body;

if (!username || !email || !password) {

return res.json({ success: false, message: 'Missing required fields' });

}

const salt = bcrypt.genSaltSync();

const hash = bcrypt.hashSync(password, salt);

const newUser = new User({ username, email, password: hash });

newUser.save((err) => {

if (err) {

console.error(err);

return res.json({ success: false, message: 'Error creating user' });

}

const token = jwt.sign({ id: newUser.\_id }, 'banana-secret', { expiresIn: '1h' });

return res.json({ success: true, token });

});

});

export default router;

```

This is just a basic outline of what the code for the Banana Cloud platform looks like. It includes a main server file with some dependencies and settings, a user model file that defines the structure of user objects in the database, and an API routes file that handles requests to the /api endpoint. The signup route in the API file uses the bcrypt library to hash passwords and the jsonwebtoken library to create JSON Web Tokens for authentication.