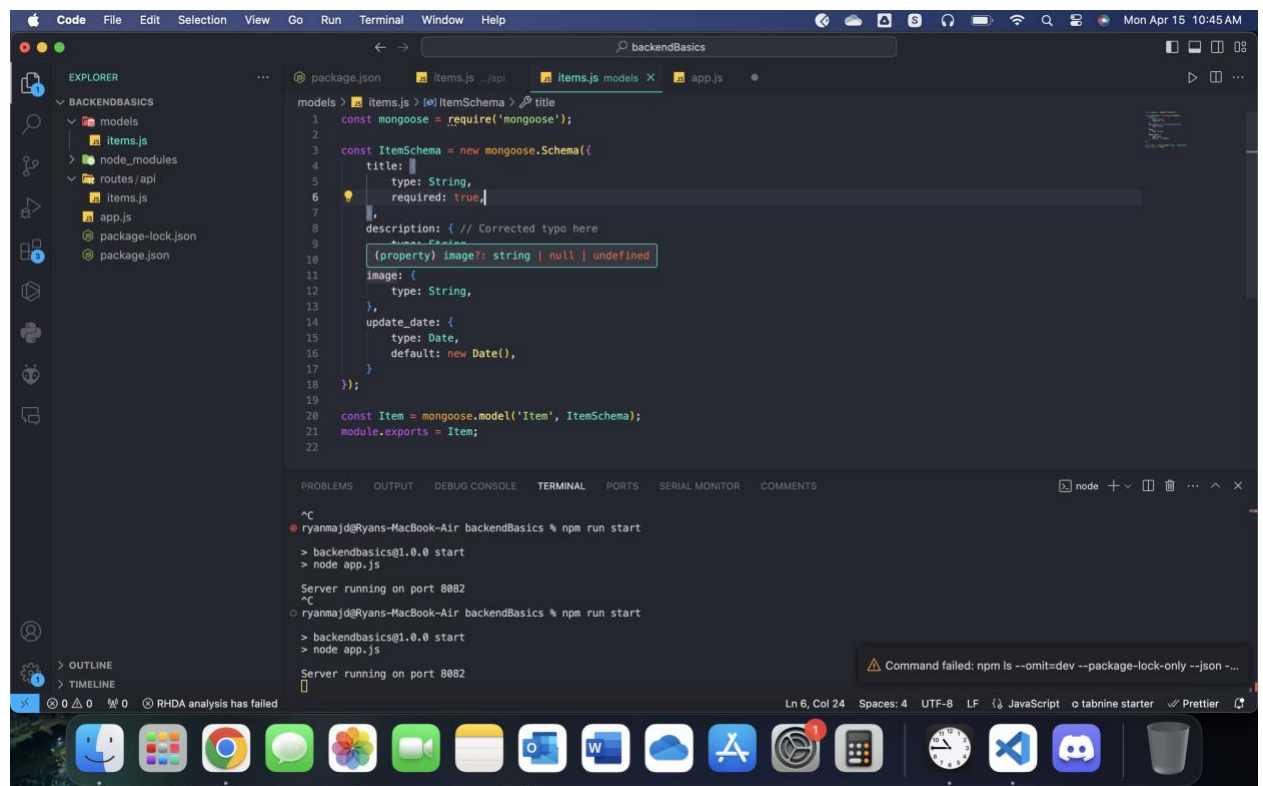


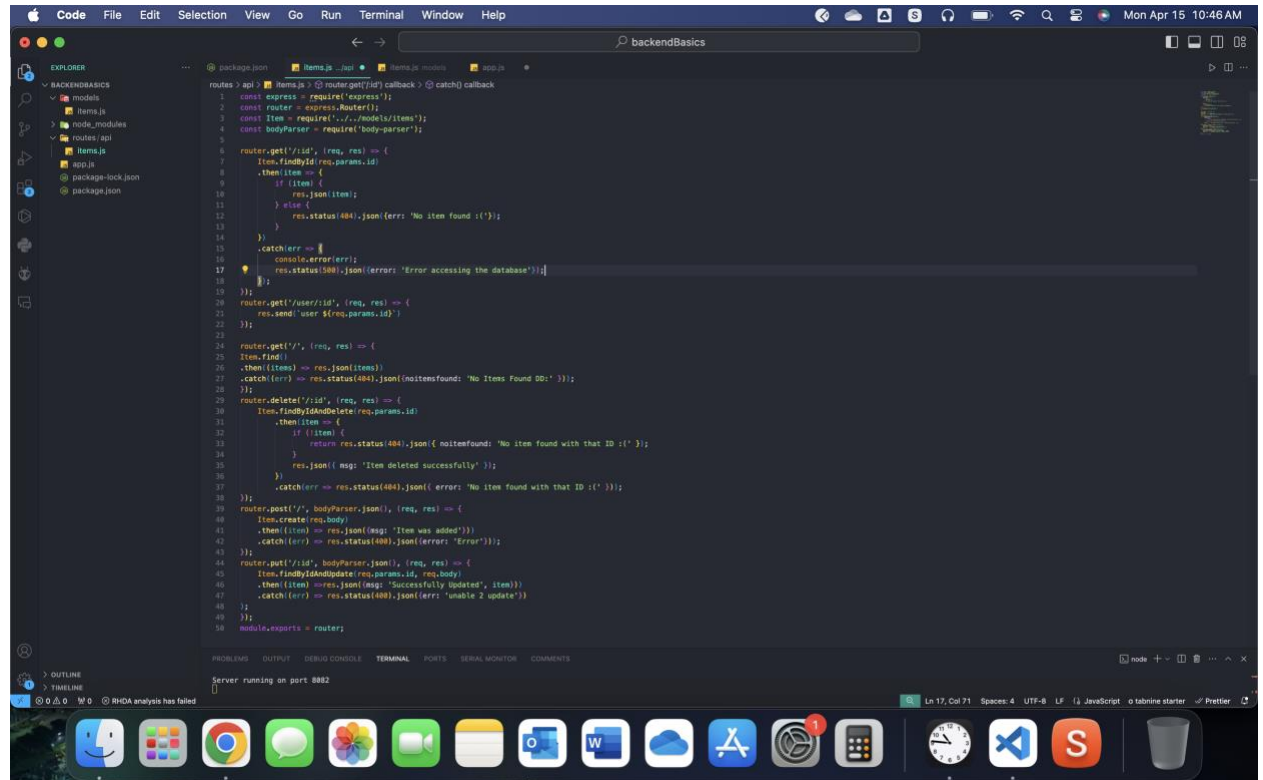
Model source code



The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure: `models`, `items.js`, `node_modules`, `routes/api`, `items.js`, `app.js`, `package-lock.json`, and `package.json`. The main editor displays the `items.js` file in the `models` directory. The code defines a Mongoose schema for an `Item` model. A red box highlights the `image` field definition: `image: { type: String, required: true }`. The terminal at the bottom shows the command `npm run start` being executed, and the server is running on port 8082. A status bar at the bottom indicates the file is at line 6, column 24, using UTF-8 encoding and the JavaScript language.

```
const mongoose = require('mongoose');
const ItemSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true
  },
  description: { // Corrected typo here
    type: String,
    required: true
  },
  image: {
    type: String,
    required: true
  },
  update_date: {
    type: Date,
    default: new Date(),
  }
});
const Item = mongoose.model('Item', ItemSchema);
module.exports = Item;
```

Route source code



The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure: `models`, `items.js`, `node_modules`, `routes/api`, `items.js`, `app.js`, `package-lock.json`, and `package.json`. The main editor displays the `items.js` file in the `routes/api` directory. The code defines a Mongoose schema for an `Item` model. A red box highlights the `image` field definition: `image: { type: String, required: true }`. The terminal at the bottom shows the command `npm run start` being executed, and the server is running on port 8082. A status bar at the bottom indicates the file is at line 17, column 71, using UTF-8 encoding and the JavaScript language.

```
const express = require('express');
const router = express.Router();
const Item = require('../models/item');
const bodyParser = require('body-parser');

router.get('/', (req, res) => {
  Item.find((err, items) => {
    if (err) {
      console.error(err);
      res.status(500).json({ error: 'Error accessing the database' });
    }
    res.json(items);
  });
});

router.get('/:id', (req, res) => {
  Item.findById(req.params.id, (err, item) => {
    if (err) {
      console.error(err);
      res.status(500).json({ error: 'Error accessing the database' });
    }
    if (!item) {
      res.status(404).json({ error: 'No item found with that ID' });
    }
    res.json(item);
  });
});

router.post('/', (req, res) => {
  Item.create(req.body, (err, item) => {
    if (err) {
      console.error(err);
      res.status(500).json({ error: 'Error creating item' });
    }
    res.json(item);
  });
});

router.put('/:id', (req, res) => {
  Item.findByIdAndUpdate(req.params.id, req.body, {
    new: true,
    runValidators: true,
  }, (err, item) => {
    if (err) {
      console.error(err);
      res.status(500).json({ error: 'Error updating item' });
    }
    if (!item) {
      res.status(404).json({ error: 'No item found with that ID' });
    }
    res.json(item);
  });
});

router.delete('/:id', (req, res) => {
  Item.findByIdAndDelete(req.params.id, (err, item) => {
    if (err) {
      console.error(err);
      res.status(500).json({ error: 'Error deleting item' });
    }
    if (!item) {
      res.status(404).json({ error: 'No item found with that ID' });
    }
    res.json({ msg: 'Item deleted successfully' });
  });
});

module.exports = router;
```

Add an item (2)

The screenshot displays two web browser windows. The top window is Postman, showing a POST request to `localhost:8082/api/items/` with a JSON body: `{ "title": "Ryan Majd", "image": "https://ugahacks.com/img/team-photos/uh9/personal-pruned/ryan-majd.png" }`. The response is `{ "msg": "Item was added" }`. The bottom window is the MongoDB Atlas interface, showing the 'test' database with a collection named 'items'. It displays two documents with fields like `_id`, `title`, `image`, and `update_date`.

Postman Window:

- URL: `web.postman.co/workspace/My-Workspace-222d7856-9a60-4091-8f58-a4bdb005020a/request/create?requestId=462a8123-1e58-4b76-9b...`
- Method: POST
- URL: `localhost:8082/api/items/`
- Body (raw):

```
1 {
2   "title": "Ryan Majd",
3   "image": "https://ugahacks.com/img/team-photos/uh9/personal-pruned/ryan-majd.png"
4 }
5
```
- Status: 200 OK, Time: 76 ms, Size: 259 B
- Response (JSON):

```
1 {
2   "msg": "Item was added"
3 }
```

MongoDB Atlas Window:

- Database: test
- Collection: items
- Documents:
- Document 1:

```
{
  "_id": "ObjectID('661d3e4716c2eae44a038191')",
  "title": "Pixel Gun 3D!!!",
  "image": "https://m.media-amazon.com/images/I/81mD3-ALcCL.png",
  "update_date": "2024-04-15T13:47:29.366+00:00",
  "__v": 0
}
```
- Document 2:

```
{
  "_id": "ObjectID('661d3e4716c2eae44a038191')",
  "title": "Ryan Majd",
  "image": "https://ugahacks.com/img/team-photos/uh9/personal-pruned/ryan-majd.png",
  "update_date": "2024-04-15T14:43:13.732+00:00",
  "__v": 0
}
```

Get item (2)

The image shows two screenshots of a web browser and a REST client (Postman) interface.

Top Screenshot (Postman):

- URL:** `web.postman.co/workspace/My-Workspace-222d7856-9a60-4091-8f58-a4bdb005020a/request/create?requestId=462a8123-1e58-4b76-9b...`
- Environment:** `localhost:8082/api/item`
- Method:** `GET`
- URL:** `localhost:8082/api/items/661d3e4716c2eae44a038191`
- Body:**

```
{
  "title": "Ryan Majd",
  "image": "https://ugahacks.com/img/team-photos/uh9/personal-pruned/ryan-majd.png"
}
```
- Status:** `200 OK`, `Time: 66 ms`, `Size: 420 B`
- Response Body (JSON):**

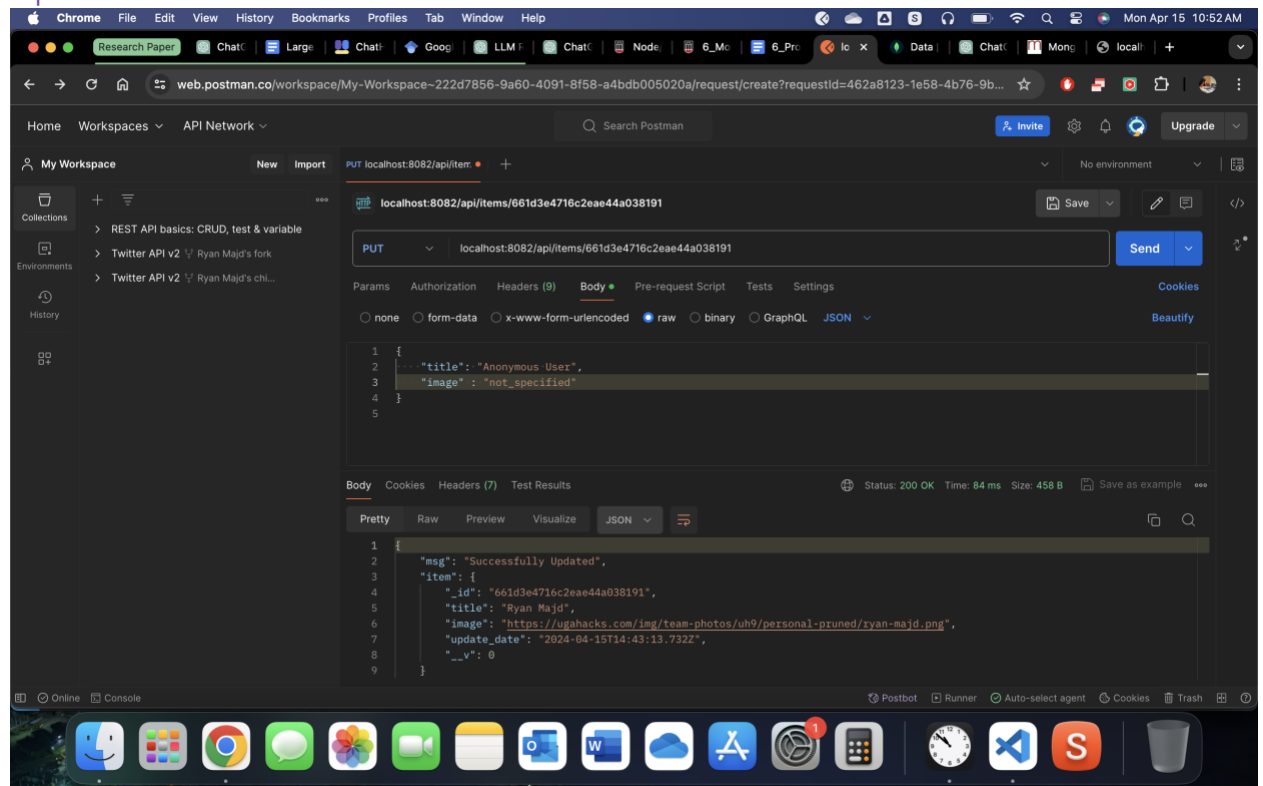
```
{
  "_id": "661d3e4716c2eae44a038191",
  "title": "Ryan Majd",
  "image": "https://ugahacks.com/img/team-photos/uh9/personal-pruned/ryan-majd.png",
  "update_date": "2024-04-15T14:43:13.732Z",
  "__v": 0
}
```

Bottom Screenshot (Web Browser):

- URL:** `localhost:8082/api/items/661d3e4716c2eae44a038191`
- Response:**

```
// 20240415105123
// http://localhost:8082/api/items/661d3e4716c2eae44a038191
{
  "_id": "661d3e4716c2eae44a038191",
  "title": "Ryan Majd",
  "image": "https://ugahacks.com/img/team-photos/uh9/personal-pruned/ryan-majd.png",
  "update_date": "2024-04-15T14:43:13.732Z",
  "__v": 0
}
```

Update item



Delete item

