



# 23.6 First API Todo App

1.

```
❏ .env.development
❏ .env.example
❏ .env.production
JS app.js
JS package.json
```

2.

```
backend > ❏ .env.development
```

```
1 MONGO_DB_USERNAME=root
2 MONGO_DB_PASSWORD=root
3 MONGO_DB_DATABASE=todo-app-test|
```

```
backend > ❏ .env.production
```

```
1 MONGO_DB_USERNAME=root
2 MONGO_DB_PASSWORD=root
3 MONGO_DB_DATABASE=todo-app
```

3.

```
> 📁 controllers
> 📁 models
> 📁 node_modules
> 📁 routers
```



# 23.6 First API Todo App

4.

```
const ENV = process.env.NODE_ENV || 'production'
require('dotenv').config({
  path: `.env.${ENV}`
});

// External Module
const express = require("express");
const mongoose = require("mongoose");
const bodyParser = require("body-parser");

// Local Module
const errorController = require("./controllers/errorController");
const itemsRouter = require("./routers/itemsRouter");

const MONGO_DB_URL =
  `mongodb+srv://${process.env.MONGO_DB_USERNAME}:${process.env.MONGO_DB_PASSWORD}@kgcluster.ie6mb.mongodb.net/${process.env.MONGO_DB_DATABASE}`;

const app = express();

app.use(bodyParser.urlencoded({ extended: true }));
app.use(itemsRouter);
app.use(errorController.get404);

const PORT = process.env.PORT || 3000;
mongoose.connect(MONGO_DB_URL).then(() => {
  app.listen(PORT, () => {
    console.log(`Server running at: http://localhost:\${PORT}`);
  });
});
```



# 23.6 First API Todo App

backend > models > `TodoItem.js` > ...

6.

```
1  const mongoose = require("mongoose");
2
3  const todoItemSchema = new mongoose.Schema({
4    task: {
5      type: String,
6      required: true
7    },
8    date: {
9      type: Date,
10     required: true
11   },
12   completed: {
13     type: Boolean,
14     default: false
15   },
16   createdAt: {
17     type: Date,
18     default: Date.now
19   }
20 });
```

5.

backend > controllers > `errorController.js` > ...

```
1  exports.get404 = (req, res, next) => {
2    res.status(404).json({ message: 'Page Not Found' });
3  };
```



## 23.6 First API Todo App

7.

```
const express = require("express");  
const itemsController = require("../controllers/itemsController");  
const itemsRouter = express.Router();
```

```
itemsRouter.post("/todos", itemsController.postItem);
```

```
module.exports = itemsRouter;
```

```
exports.postItem = async (req, res, next) => {  
  try {  
    console.log(req.body);  
    const todoItem = new TodoItem(req.body);  
    const item = await todoItem.save();  
    res.json(item);  
  } catch (err) {  
    next(err);  
  }  
};
```



## 23.6 First API Todo App

8,9.

```
const bodyParser = require("body-parser");
const cors = require("cors");

app.use(bodyParser.urlencoded({ extended: true }));
app.use(cors());
app.use(express.json());
app.use(itemsRouter);
app.use(errorController.get404);
```

---



## 23.7 API for Fetch Items

```
itemsRouter.get("/todos", itemsController.getItems);  
itemsRouter.post("/todos", itemsController.postItem);
```

```
exports.getItems = async (req, res, next) => {  
  try {  
    const items = await TodoItem.find();  
    res.json(items);  
  } catch (err) {  
    next(err);  
  }  
};
```

```
export const todoItemToClientModel = (serverItem) => {  
  return {  
    id: serverItem._id,  
    todoText: serverItem.task,  
    todoDate: serverItem.date  
  }  
}
```

```
const formattedDate = new Date(todoDate).toLocaleDateString('en-US', {  
  year: 'numeric',  
  month: 'long',  
  day: 'numeric'  
});
```



## 23.8 API for Deleting Items

```
itemsRouter.get("/todos", itemsController.getItems);  
itemsRouter.post("/todos", itemsController.postItem);  
itemsRouter.delete("/todos/:id", itemsController.deleteItem);
```

```
exports.deleteItem = async (req, res, next) => {  
  try {  
    const { id } = req.params;  
    await TodoItem.findByIdAndDelete(id);  
    res.json({ id, message: "Item deleted" });  
  } catch (err) {  
    next(err);  
  }  
};
```



## 23.9 Improving UI Elements

1. Remove bootstrap
2. Add tailwind to the project.
3. Create the whole app using tailwind classes.





## 23.10 Adding Complete Item functionality

1. Add a UI element to mark the item complete.
2. Add a component state based on which items will be striked through, default value should come from server.
3. Add a toggleComplete handler in TodoItem that sends a PATCH request to update the completed flag and expects the updated item in return. Also the value of state must be updated based on the final value.
4. Add a route and API in the backend to update the todoItem.

# 23.10 Adding Complete Item functionality

1,2. 

```
<input
  type="checkbox"
  checked={isComplete}
  onChange={toggleComplete}
  className="w-5 h-5 rounded ■border-gray-300 ■text-blue-600 ■focus:ring-blue-500"
/>
```

```
const TodoItem = ({ id, todoText, todoDate, completed }) => {
  const { deleteTodoItem } = useContext(TodoItemsContext);
  const [isComplete, setIsComplete] = useState(completed);
```

```
<div className={`flex flex-col ${isComplete ? '■text-gray-400 line-through' : '■text-gray-700'}`} >
  <span className="font-medium">{todoText}</span>
  <span className="text-sm ■text-gray-500">{formattedDate}</span>
</div>
```

## 23.10 Adding Complete Item functionality

```
3.  const toggleComplete = () => {  
    fetch(`http://localhost:3000/todos/${id}`, {  
      method: 'PATCH',  
      headers: { 'Content-Type': 'application/json' },  
      body: JSON.stringify({ completed: !isComplete })  
    })  
    .then(res => res.json())  
    .then(data => {  
      setIsComplete(data.completed);  
    })  
    .catch(err => console.log(err));  
  }
```

# 23.10 Adding Complete Item functionality

4.

```
itemsRouter.get("/todos", itemsController.getItems);
itemsRouter.post("/todos", itemsController.postItem);
itemsRouter.delete("/todos/:id", itemsController.deleteItem);
itemsRouter.patch("/todos/:id", itemsController.updateItem);
```

```
exports.updateItem = async (req, res, next) => {
  try {
    const { id } = req.params;
    const { completed } = req.body;
    const updatedItem = await TodoItem.findByIdAndUpdate(
      id,
      { completed: completed },
      { new: true }
    );

    res.json(updatedItem);
  } catch (err) {
    next(err);
  }
};
```



## 23.11 Deploying React App

```
prashantjain@Prashants-MacBook-Pro frontend % az extension add --name staticwebapp
No stable version of 'staticwebapp' to install. Preview versions allowed.
The installed extension 'staticwebapp' is in preview.
```

```
prashantjain@Prashants-MacBook-Pro frontend % npm run build
```

```
> 6-todo-ui-bootstrap@0.0.0 build
> vite build
```

```
vite v5.4.3 building for production...
```

```
✓ 38 modules transformed.
```




dist/index.html	0.46 kB	gzip: 0.30 kB
dist/assets/index-kY9ww-QL.css	9.31 kB	gzip: 2.58 kB
dist/assets/index-DGJMYW8J.js	147.18 kB	gzip: 47.49 kB

```
✓ built in 384ms
```





# 23.11 Deploying React App


```
prashantjain@Prashants-MacBook-Pro MERNLiveFrontend % az staticwebapp create \
--name MERNLiveFrontend \
--resource-group omprashantjain_rg_7471 \
--location "Central US" \
--source https://github.com/Complete-Coding/MERNLiveFrontend \
--branch main \
--app-location "/" \
--output-location "dist" \
--login-with-github
Please navigate to https://github.com/login/device and enter the user code 8D2F-CE49 to activate and retrieve your github per
sonal access token
Waiting up to '14' minutes for activation
{
  "allowConfigFileUpdates": true,
  "branch": "main",
  "buildProperties": null,
  "contentDistributionEndpoint": "https://content-dm1.infrastructure.4.azurestaticapps.net",
  "customDomains": [],
  "databaseConnections": [],
  "defaultHostname": "white-grass-0bb31e810.4.azurestaticapps.net",
  "enterpriseGradeCdnStatus": "Disabled",
  "id": "/subscriptions/166737a1-e433-403b-aacb-e39a9d24b588/resourceGroups/omprashantjain_rg_7471/providers/Microsoft.Web/st
aticSites/MERNLiveFrontend",
  "identity": null,
  "keyVaultReferenceIdentity": "SystemAssigned",
  "kind": null,
  "linkedBackends": [],
  "location": "Central US",
  "name": "MERNLiveFrontend",
  "privateEndpointConnections": [],
  "provider": "GitHub",
  "publicNetworkAccess": null,
  "repositoryToken": null,
  "repositoryUrl": "https://github.com/Complete-Coding/MERNLiveFrontend",
  "resourceGroup": "omprashantjain_rg_7471",
  "sku": {
    "capabilities": null,
    "capacity": null,
    "family": null,
    "locations": null,
    "name": "Free",
    "size": null,
    "skuCapacity": null,
    "tier": "Free"
  },
  "stagingEnvironmentPolicy": "Enabled",
  "tags": null,
  "templateProperties": null,
  "type": "Microsoft.Web/staticSites",
  "userProvidedFunctionApps": null
}
```





### Authorize Azure CLI

 This authorization was requested from Ghaziabad 160.202.37.212 on December 3rd, 2024 at 17:40 (IST)  
Make sure you trust this device as it will get access to your account.


**Azure CLI by AzureAppServiceCLI**  
wants to access your omprashantjain account

**Repository webhooks and services**  
Admin access


**Repositories**  
Public and private

**Workflow**  
Update GitHub Action Workflow files.

**Organization access**

 KnowledgeGateCoding X

Grant

 Complete-Coding X

Grant

Cancel

Authorize  
AzureAppServiceCLI

Requested from Ghaziabad 160.202.37.212 on December 3rd, 2024 at 17:40 (IST)



## 23.11 Deploying React App

SEARCH

http://mern-live-backend.azurewebsites.net

Aa \_ab \*

//mern-live-backend.azurewebsites.net

AB

files to include

files to exclude

4 results in 3 files - [Open in editor](#)

▼

🌀

AddTodo.jsx

src/components

M

1

fetch("http://mern-live-backend.azurewebsites.net//mern-live-backend.azurewebsites.net/todos", {

▼

🌀

LoadItems.jsx

src/components

M

1

fetch("http://mern-live-backend.azurewebsites.net//mern-live-backend.azurewebsites.net/todos")

▼

🌀

TodolItem.jsx

src/components

M

2

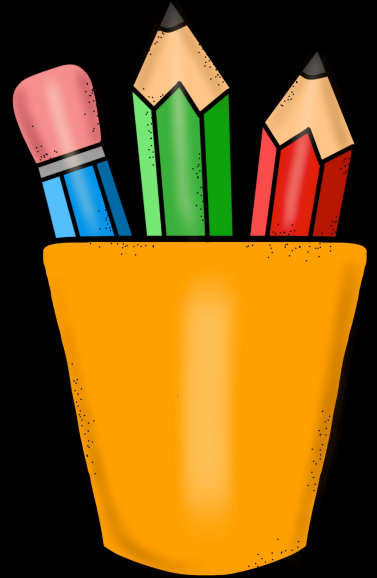
fetch(`http://mern-live-backend.azurewebsites.net//mern-live-backend.azurewebsites.net/todos/\${i...`

fetch(`http://mern-live-backend.azurewebsites.net//mern-live-backend.azurewebsites.net/todos/\${i...



# Revision

1. What are Async Requests
2. What are REST APIs
3. Decoupling Frontend & Backend
4. Routes & HTTP Methods
5. REST Core Concepts
6. First API Todo App
7. API for Fetch Items
8. API for Deleting Items
9. Improving UI Elements
10. Adding Complete Item functionality
11. Deploying React App







# Practise Milestone

Take your **todo-app** forward:

1. Add other features like editing an item text.
2. Changing the date of completion.



