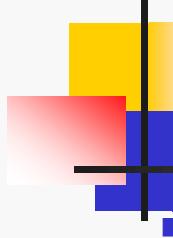


# **DMET 601**

# **Web Technologies and Usability**

**MERN Stack**

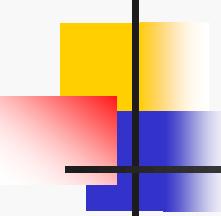
**Prof. Dr. Rimon Elias**



# Contents

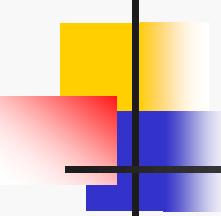
---

MERN stack



# Full Stack

- Full stack web development includes both client-side and server-side scripting.
- Full Stack comprises
  - Programming a **browser** using (e.g., JavaScript, jQuery, Angular, or Vue)
  - Programming a **server** using (e.g., PHP, ASP, Python, or Node)
  - Programming a **database** using (e.g., SQL, SQLite, or MongoDB)
  - This is in addition to HTML and CSS.

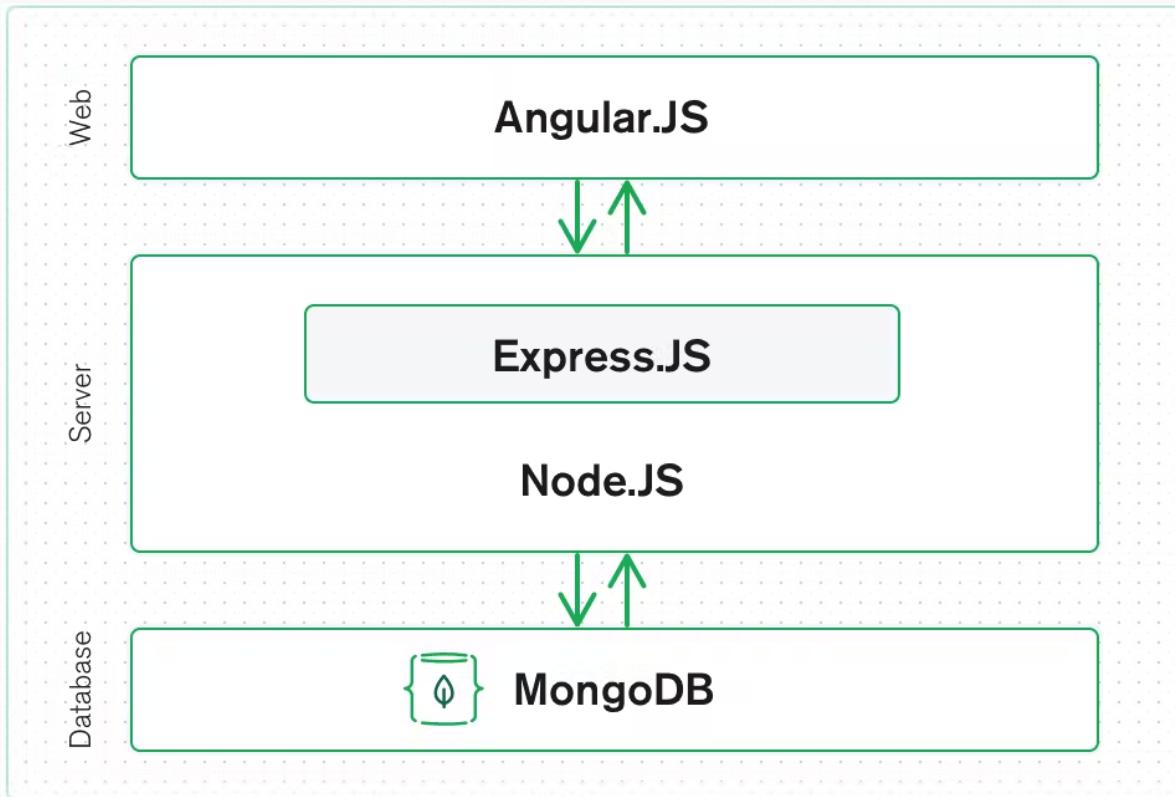


# MEAN Stack

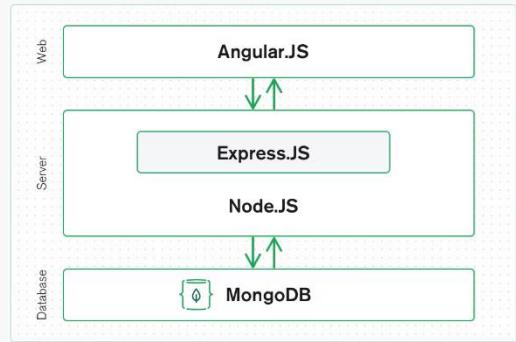
- The MEAN stack is a JavaScript-based framework for developing web applications.
  - MEAN is named after **M**ongoDB, **E**xpress, **A**ngular, and **N**ode
    - MongoDB — document database
    - Express(.js) — Node.js web framework
    - Angular(.js) — a client-side JavaScript framework
    - Node(.js) — the premier JavaScript web server
  - Variations:
    - MERN (replacing Angular.js with React.js)
    - MEVN (replacing Angular.js with Vue.js)
  - <https://www.mongodb.com/mean-stack>
  - <https://www.javatpoint.com/mean-stack-tutorial>
- (c) 2025, Dr. R. Elias MERN Stack

# MEAN Stack

Using MEAN, one can construct a 3-tier architecture (front end, back end, database) entirely using JavaScript and JSON.

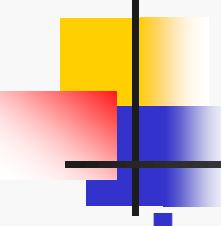


# MEAN Installation



## How to Install the MEAN Stack on Windows

- Install NodeJS on Windows.
- Install MongoDB on Windows. Create the MongoDB data directory. Start MongoDB Server on Windows.
- Install GIT on Windows.
- Install Angular-CLI on Windows.
  - <https://www.simplilearn.com/how-to-install-angular-and-nodejs-on-windows-article>
- Install the MEAN Boilerplate on Windows.
- Running Your Sample MEAN Application.
- <https://techievor.com/blog/post/how-to-install-the-mean-stack-on-windows>

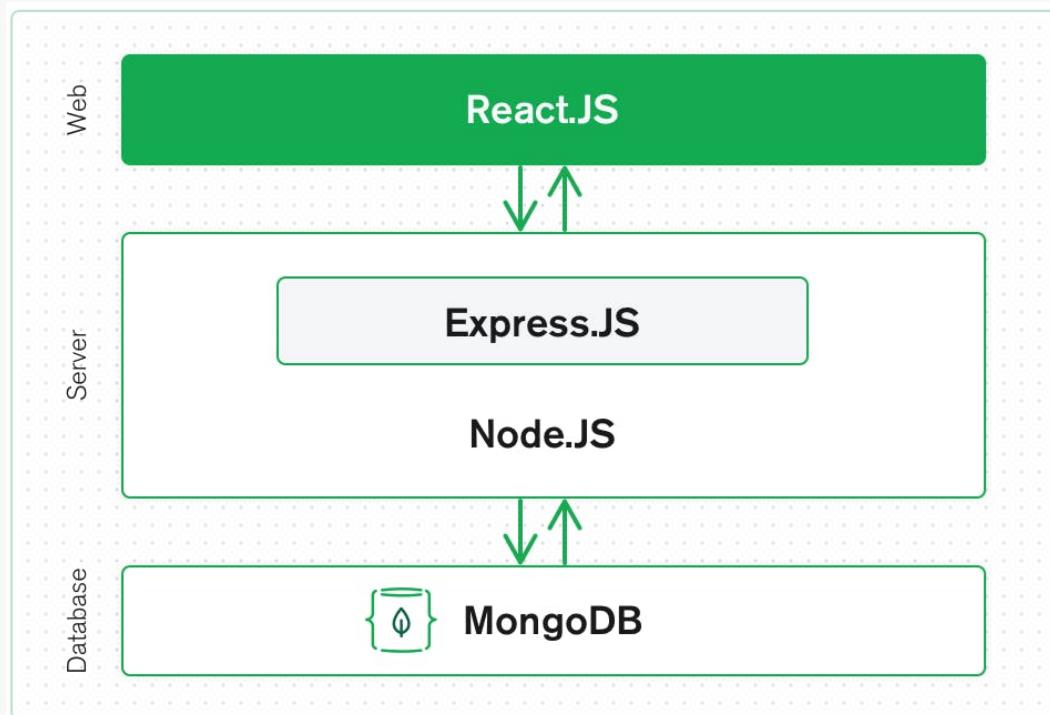


# MERN Stack

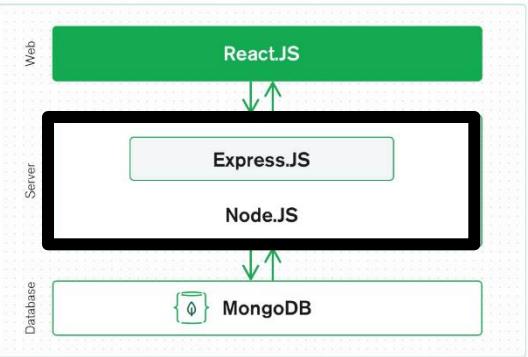
- MERN is one of several variations of the MEAN stack, where the traditional Angular.js front-end framework is replaced with React.js.
- MERN stands for MongoDB, Express, React, Node, after the four key technologies that make up the stack.
  - MongoDB — document database
  - Express(.js) — Node.js web framework
  - React(.js) — a client-side JavaScript framework
  - Node(.js) — the premier JavaScript web server
- <https://www.mongodb.com/mern-stack>

# MERN Stack

Using MERN, one can construct a 3-tier architecture (front end, back end, database) entirely using JavaScript and JSON.

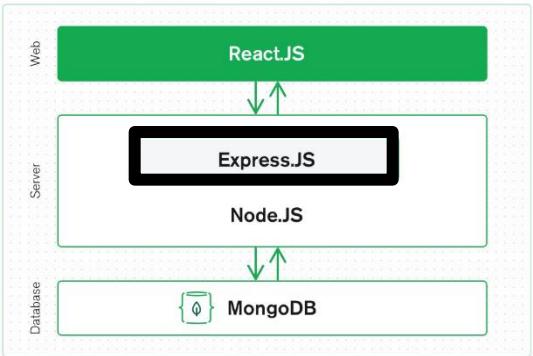


# Node.JS



- Node.js is an open source server environment.
- Using Node.js, you can run JavaScript on the server.

# Express.JS



The next level down in the MERN stack is Express.js.

- Express.js runs on a Node.js server.
- Express is a minimal and flexible Node.js web application framework that provides a robust set of features

```
npm install express --save
```

# Express.JS: Hello World! Example

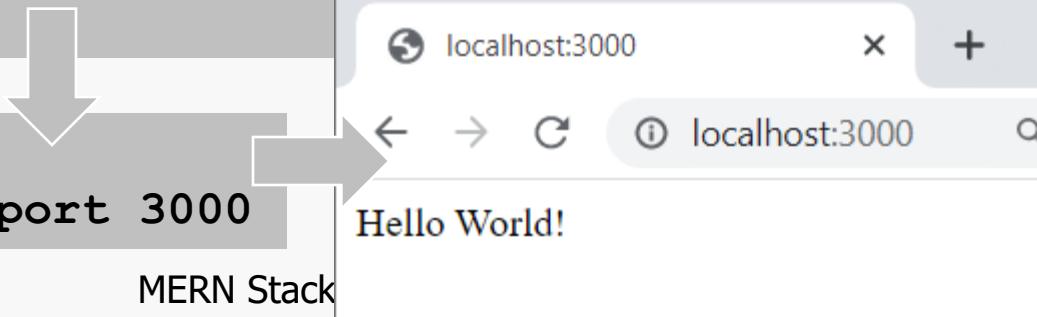
The next

```
const express = require('express')
const app = express()
const port = 3000

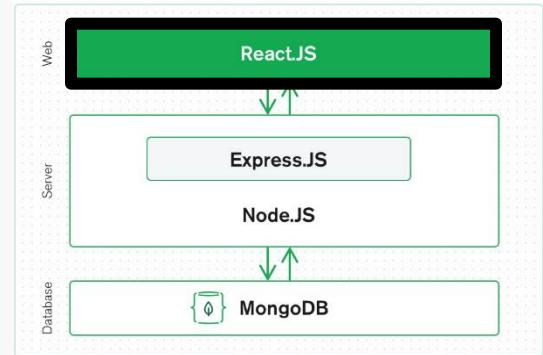
app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

F:\>node exp.js  
Example app listening on port 3000



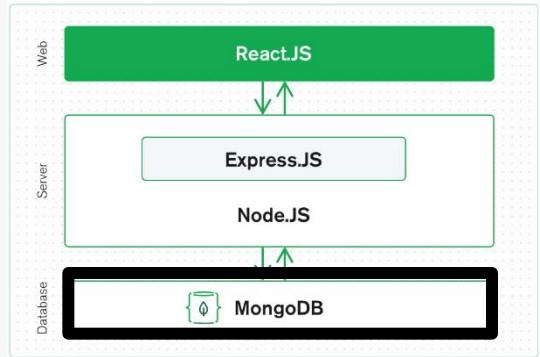
# React.js



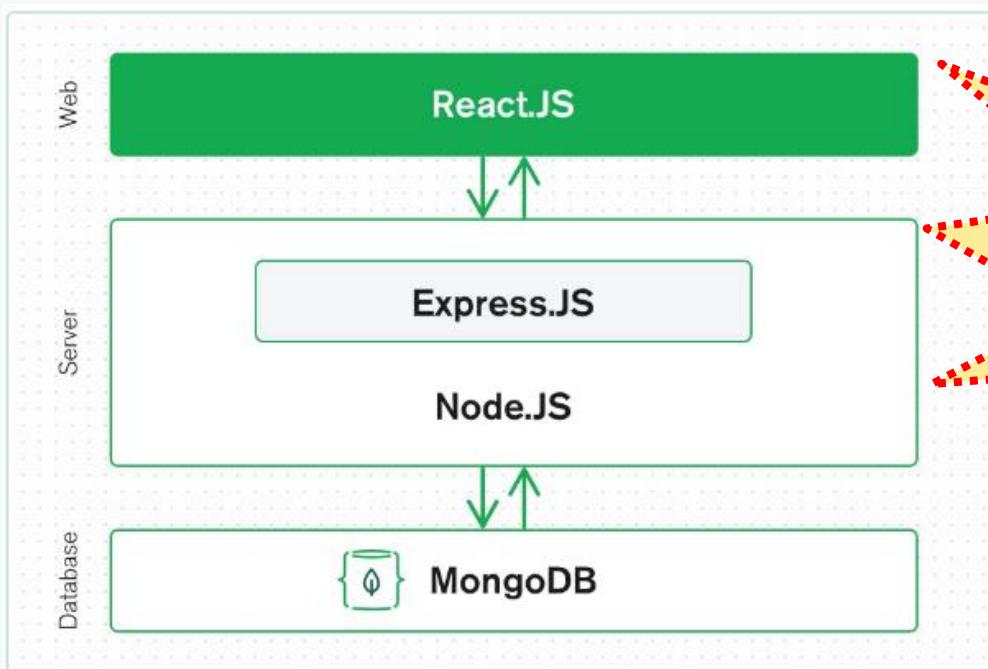
- React.js is the front end top tier of the MERN stack.
- It is a JavaScript framework for creating dynamic client-side applications
- Using React, one can build up interfaces and connect them to data on back-end server.

# MongoDB

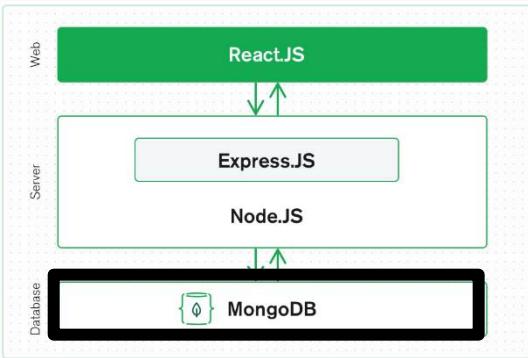
MongoDB is a **NoSQL** database with support for server-side **JavaScript** execution



# MERN



# Atlas: Start with MongoDB



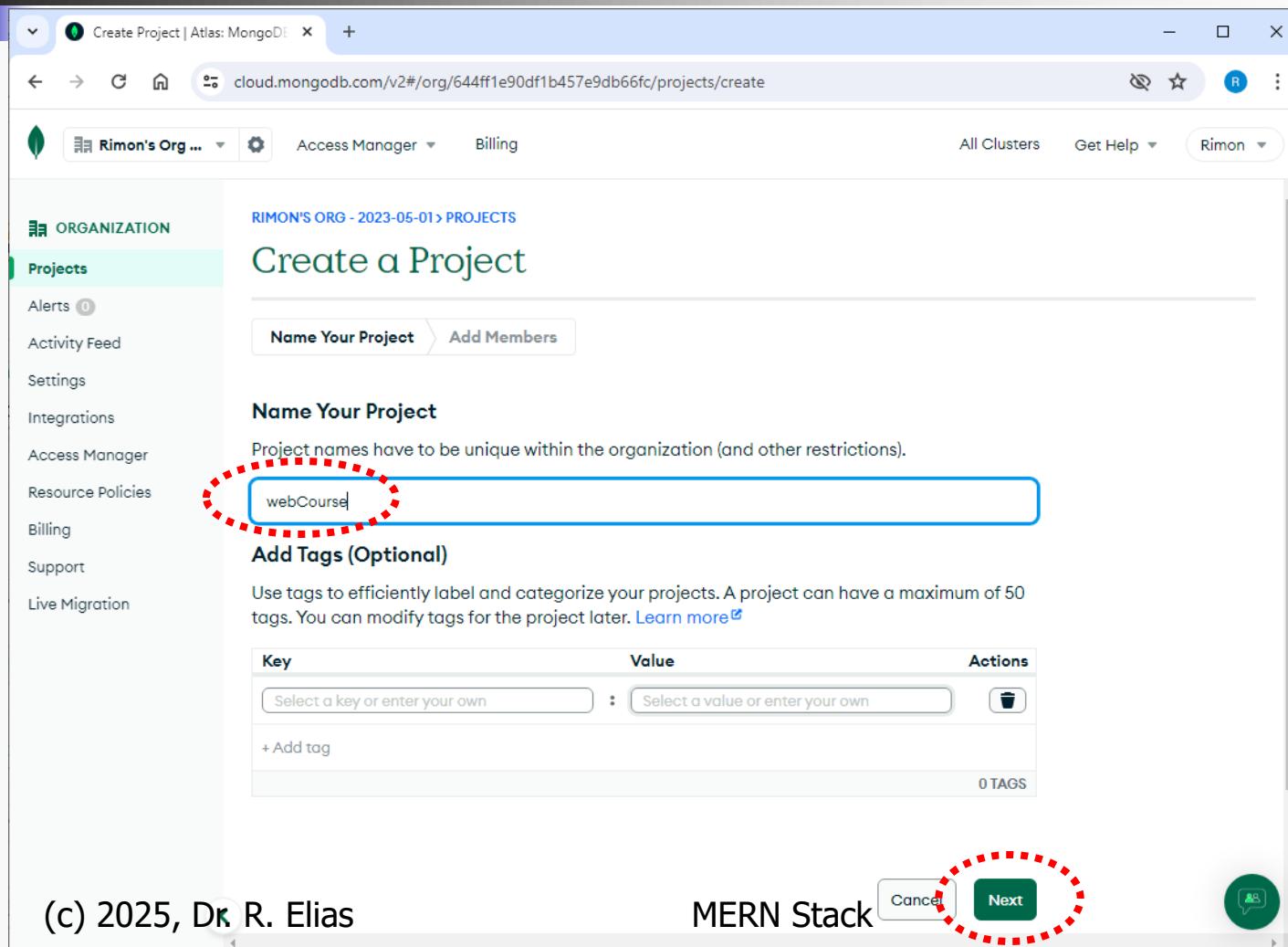
Go to <https://www.mongodb.com/> and sign in and go to “Projects”

The screenshot shows the MongoDB Atlas Projects page. On the left, a sidebar menu includes "Alerts 0", "Activity Feed", "Settings", "Integrations", "Access Manager", "Resource Policies", "Billing", "Support", and "Live Migration". The main area is titled "RIMON'S ORG - 2023-05-01" and displays a table of "Projects". The table has columns: Project Name, Clusters, Tags, Users, Teams, Alerts, and Actions. Two projects are listed: "rimonelias" and "rimonzelias", each with 1 Cluster. A search bar at the top says "Find a project...". In the top right corner, a green button labeled "New Project" is circled with a red dashed line.

Project Name	Clusters	Tags	Users	Teams	Alerts	Actions
rimonelias	1 Cluster	+ Add Tags	1 User	0 Teams	0 Alerts	...
rimonzelias	1 Cluster	+ Add Tags	1 User	0 Teams	0 Alerts	...

System Status: All Good | Last Login: 62.241.151.174  
©2025 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

# Atlas: Start with MongoDB



The screenshot shows the MongoDB Atlas 'Create Project' interface. On the left, a sidebar menu lists organization management options: Projects (selected), Alerts (0), Activity Feed, Settings, Integrations, Access Manager, Resource Policies, Billing, Support, and Live Migration. The main area is titled 'RIMON'S ORG - 2023-05-01 > PROJECTS' and 'Create a Project'. It features two tabs: 'Name Your Project' (highlighted with a red dashed circle) and 'Add Members'. Below these tabs, a section titled 'Name Your Project' contains a note: 'Project names have to be unique within the organization (and other restrictions)' and a text input field containing 'webCourse'. A section titled 'Add Tags (Optional)' follows, with a note: 'Use tags to efficiently label and categorize your projects. A project can have a maximum of 50 tags. You can modify tags for the project later.' It includes a table for adding tags, which currently shows '0 TAGS'. At the bottom right, there are 'Cancel' and 'Next' buttons, with the 'Next' button also highlighted with a red dashed circle.

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas 'Create Project' interface. The URL in the browser is [cloud.mongodb.com/v2#/org/644ff1e90df1b457e9db66fc/projects/create](https://cloud.mongodb.com/v2#/org/644ff1e90df1b457e9db66fc/projects/create). The left sidebar shows the organization structure: Rimon's Org ... > RIMON'S ORG - 2023-05-01 > PROJECTS. The main area is titled 'Create a Project' and is currently on the 'Name Your Project' step. Below it, there is a section for 'Add Members and Set Permissions'. A red dashed circle highlights the 'Create Project' button at the bottom right. A sidebar on the right lists member permissions:

- Project Owner**: Has full administration access
- Project Cluster Manager**: Can update clusters
- Project Data Access Admin**: Can access and modify a cluster's data and indexes, and kill operations
- Project Data Access Read/Write**: Can access a cluster's data and indexes, and modify data
- Project Data Access Read Only**: Can access a cluster's data and indexes

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas Project Overview interface. The URL in the browser is [cloud.mongodb.com/v2/6816261f43f60e5ab68fcebd#/overview](https://cloud.mongodb.com/v2/6816261f43f60e5ab68fcebd#/overview). The main content area displays the 'Overview' for the 'RIMON'S ORG - 2023-05-01 > WEBCOURSE' project. A large central graphic features three green cylindrical bars stacked vertically, with a small yellow character figure interacting with them. Below the graphic, the text 'Create a cluster' is displayed, followed by the instruction 'Choose your cloud provider, region, and specs.' A prominent green button with the text '+ Create' is centered at the bottom of this section, enclosed in a red dashed oval. To the left of the main content, a vertical sidebar lists various service categories: DATABASE, SERVICES, SECURITY, and more. To the right, there are sections for 'Toolbar', 'Featured Resources', and 'New On Atlas'.

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas 'Deploy your cluster' interface. At the top, there's a navigation bar with a back button, forward button, refresh button, and a search bar containing the URL [cloud.mongodb.com/v2/6816261f43f60e5ab68fcebd#/clusters/starterTemplates](https://cloud.mongodb.com/v2/6816261f43f60e5ab68fcebd#/clusters/starterTemplates). Below the navigation is a title 'Deploy your cluster' and a subtitle: 'Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.' Three cluster templates are listed:

- M10** **\$0.10/hour**  
Dedicated cluster for development environments and low-traffic applications.  
STORAGE RAM vCPU  
10 GB 2 GB 2 vCPUs
- Flex** **From \$0.011/hour**  
Up to \$30/month  
For application development and testing, with on-demand burst capacity for unpredictable traffic.  
STORAGE RAM vCPU  
5 GB Shared Shared
- Free**  
For learning and exploring MongoDB in a cloud environment.  
STORAGE RAM vCPU  
512 MB Shared Shared

A green callout at the bottom left states: **Free forever!** Your free cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

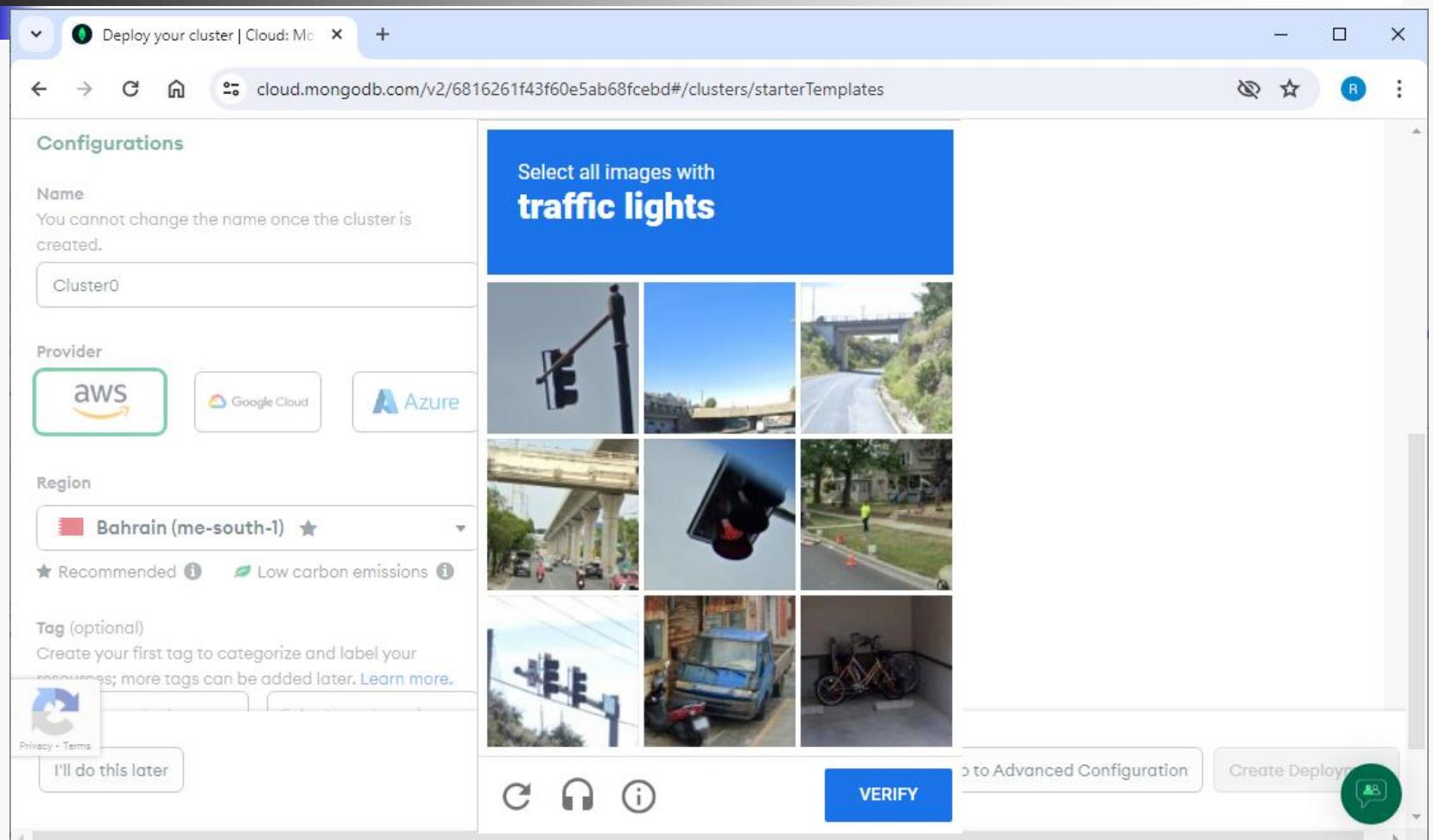
At the bottom are three buttons: 'I'll do this later', 'Go to Advanced Configuration', and 'Create Deployment' (highlighted with a green border).

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Cloud Atlas setup interface. It includes:

- Configurations**:
  - Name**: Cluster0
  - Provider**: AWS (selected)
  - Region**: Bahrain (me-south-1) (selected)
  - Tag (optional)**: Create your first tag to categorize and label your resources; more tags can be added later. [Learn more.](#)
- Quick setup**:
  - Automate security setup
  - Preload sample dataset
- Buttons**: I'll do this later, Go to Advanced Configuration, Create Deployment

# Atlas: Start with MongoDB



# Atlas: Start with MongoDB

The screenshot shows a web browser window for the MongoDB Atlas 'Project Overview' page. A modal dialog titled 'Connect to Cluster0' is open, showing the first step of a three-step wizard: 'Set up connection security'. The modal contains instructions to secure the cluster by setting access controls. It lists two tasks: 'Add a connection IP address' (with a note about local connectivity) and 'Create a database user' (with a note about autogenerated credentials). Below these tasks is a form for creating a database user, which includes fields for 'Username' (set to 'rimonzelias') and 'Password' (set to 'MFyp0DAHjtxHudv'). A red dashed circle highlights the 'Create Database User' button. To the right of the password field is a 'Copy' button, also highlighted with a red dashed circle. At the bottom of the modal are 'Close' and 'Choose a connection method' buttons.

# Atlas: Start with MongoDB

The screenshot shows a browser window for the MongoDB Atlas Project Overview. The URL is [cloud.mongodb.com/v2/6816261f43f60e5ab68fcebd#/overview?automateSecurity=true&connectCluster=Cluster0](https://cloud.mongodb.com/v2/6816261f43f60e5ab68fcebd#/overview?automateSecurity=true&connectCluster=Cluster0). A modal dialog titled "Connect to Cluster0" is displayed, showing a three-step process: 1. Set up connection security, 2. Choose a connection method, and 3. Connect. Step 1 is completed with a green checkmark. Step 2 is highlighted with a red dashed oval around the "Choose a connection method" button. Step 3 is partially visible. The modal also contains instructions about securing the cluster and creating a database user. The left sidebar shows the project structure with "webCourse" selected under "DATABASE". The right sidebar shows "Toolbar" and "Featured Resources" sections.

Project Overview | Cloud: Mong x +

cloud.mongodb.com/v2/6816261f43f60e5ab68fcebd#/overview?automateSecurity=true&connectCluster=Cluster0

Atlas

webCourse

Overview

DATABASE

Clusters

SERVICES

Atlas Search

Stream Processing

Triggers

Migration

Data Federation

SECURITY

Quickstart

Backup

Close

Choose a connection method

Connect to Cluster0

Set up connection security

Choose a connection method

Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

1. Add a connection IP address

Your current IP address (62.241.151.174) has been added to enable local connectivity. Only an IP address you add to your Access List will be able to connect to your project's clusters. Add more later in [Network Access](#).

2. Create a database user

A database user has been added to this project. Create another user later in [Database Access](#). You'll need your database user's credentials in the next step.

Toolbar

Resources (3)

Featured Resources

JAVASCRIPT / NODE..

Connect to your data ..

Query your data: CRUD ..

Mongoose Query ..

Integrate with ..

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas interface with a central modal window titled "Connect to Cluster0". The modal is divided into three steps: 1. Set up connection security (completed), 2. Choose a connection method (selected), and 3. Connect (not yet reached). The second step, "Choose a connection method", is highlighted with a red dashed border around the "Drivers" section. This section contains an icon of a computer monitor with code and the text: "Drivers" and "Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)". Below this, there are four more sections: "Compass", "Shell", "MongoDB for VS Code", and "Atlas SQL", each with a brief description and a right-pointing arrow. At the bottom of the modal are "Go Back" and "Close" buttons. The background shows the left sidebar with "Overview", "DATABASE", "SERVICES", and "SECURITY" sections, and the right sidebar with "Toolbar", "Featured Resources", and "Sample Apps".

(c) 2

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas interface with a modal window titled "Connect to Cluster0". The modal is divided into three steps: "Set up connection security" (completed), "Choose a connection method" (completed), and "Connect" (in progress). The "Connecting with MongoDB Driver" section contains the following steps:

- 1. Select your driver and version**  
We recommend installing and using the latest driver version.  
Driver: Node.js  
Version: 6.7 or later
- 2. Install your driver**  
Run the following on the command line:  
`npm install mongodb`  
[View MongoDB Node.js Driver installation instructions.](#)
- 3. Add your connection string into your application code**  
Use this connection string in your application  
 View full code sample  

```
mongodb+srv://rimonzelias:<db_password>@cluster0.rkvu3dn.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0
```

Replace `<db_password>` with the password for the `rimonzelias` database user. Ensure any option params are [URL encoded](#).

**RESOURCES**

- [Get started with the Node.js Driver](#)
- [Access your Database Users](#)
- [Node.js Starter Sample App](#)
- [Troubleshoot Connections](#)

**MERN Stack**

**Done**

(c) 2025, Dr. R. Elias

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas Project Overview interface. The left sidebar has a red dashed circle around the 'Clusters' link under the 'DATABASE' section. The main area shows an 'Overview' for 'RIMON'S ORG - 2023-05-01 > WEBOURSE'. It features a 'Clusters' section with 'Cluster0' (Connect, Edit configuration), 'Add data', 'Load sample data', 'Data modeling templates', and '+ Add Tag'. Below this is an 'Application Development' section with 'Get connection string' and '...' buttons. A 'Toolbar' on the right lists 'Resources (3)', 'Performance Tools', 'Upgrade cluster', 'BACKUPS', 'PERFORMANCE ADVISOR', and 'REAL-TIME PERFORMANCE'. The bottom left of the sidebar shows 'New On Atlas (3)' and 'Goto'.

# Atlas: Start with MongoDB

Collection = DB table

**Clusters**

Find a database deployment...

Cluster0 Connect View Monitoring Browse Collections ...

Enhance Your Experience

For production throughput and richer metrics, upgrade to a dedicated cluster now!

Upgrade

R 0 W 0 05/03/25 - 05:46 PM

Connections 0 05/03/25 - 05:46 PM

In 0.00 B/s Out 0.00 B/s 05/03/25 - 05:46 PM

100.0/s 13.0 1.12 KB/s

VERSION	REGION	TYPE	BACKUPS	LINKED APP SERVICES	ATLAS SQL	ATLAS SEARCH
8.0.8	AWS / Bahrain (me-south-1)	Replica Set - 3 nodes	Inactive	None Linked	Connect	Create Index

+ Add Tag

(c) 2020, Dr. R. Elias

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas Data Services interface. On the left, a sidebar menu includes sections for Clusters (selected), Services (Atlas Search, Stream Processing, Triggers, Migration, Data Federation), and Security (Quickstart, Backup, Database Access, Network Access, Advanced). The main area displays the 'Cluster0' database under 'RIMON'S ORG - 2023-05-01 > WEBOURSE > DATABASES'. The 'Collections' tab is active, showing 'DATABASES: 0' and 'COLLECTIONS: 0'. A prominent call-to-action at the bottom says 'Explore Your Data' with sub-points: 'Find: run queries and interact with documents', 'Indexes: build and manage indexes', 'Aggregation: test aggregation pipelines', and 'Search: build search indexes'. Two buttons are visible: 'Load a Sample Dataset' and 'Add My Own Data' (circled in red).

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas interface. A modal window titled "Create Database" is open in the foreground. Inside the modal, there are fields for "Database name" (containing "DMET"), "Collection name" (containing "studnets"), and "Additional Preferences" (with a dropdown set to "Select"). At the bottom right of the modal are "Cancel" and "Create" buttons, with the "Create" button circled in red. The background shows the main Atlas dashboard with various navigation tabs like Overview, Clusters, and Services. A large yellow callout box with the text "Collection = DB table" is overlaid on the top right of the screen.

Collection = DB table

Create Database

Database name

DMET

Collection name

studnets

Additional Preferences

Select

Create

Cancel

Cluster0 Data | Cloud: MongoDB

cloud.mongodb.com/v2/6816261f43f60e5ab68fc...#metrics/replica...

R

Atlas

Rimon...

Access Manager

Overview

DATABASES

Clusters

SERVICES

Atlas Search

Stream Processor

Triggers

Migration

Data Federation

SECURITY

-south-1)

Atlas Search

REFRESH

29

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas web interface. The top navigation bar includes tabs for 'Data Services' (selected), 'Charts', 'Access Manager', 'Billing', 'All Clusters', 'Get Help', and 'Rimon'. The left sidebar lists 'Clusters' (selected), 'webCourse', 'Overview', 'DATABASE', 'SERVICES', 'Atlas Search', 'Stream Processing', 'Triggers', 'Migration', 'Data Federation', 'SECURITY', 'Quickstart', 'Backup', 'Database Access', and 'Network Access'. The main content area displays 'RIMON'S ORG - 2023-05-01' and 'Cluster0'. The 'Collections' tab is selected, showing 'DMET.studnets' with storage size 4KB, logical data size 0B, total documents 0, and index size 4KB. It features 'Find', 'Indexes', 'Schema Anti-Patterns' (with 0 issues), 'Aggregation', 'Search Indexes', and an 'INSERT DOCUMENT' button (circled in red). A large yellow box highlights the text 'Document = DB table row'. The bottom right corner shows a green speech bubble icon.

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas web interface. The left sidebar includes sections for Overview, DATABASE, Clusters (selected), SERVICES, SECURITY, and Goto. The main area displays an 'Insert Document' dialog for the 'students' collection. The dialog contains the following JSON document:

```
1   { "_id": {"$oid": "681cc621171521bb6ba42b16"},  
2     "name": "Rimon",  
3     "address": "GUCh",  
4     "ID": 123456  
5   }
```

Below the document, there are 'Cancel' and 'Insert' buttons. The background shows the 'Performance Advisor' and 'Online Archiving' sections of the Atlas dashboard.

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas web interface. The left sidebar has a red dashed box around the 'Clusters' item under the 'DATABASES' section. The main area shows the 'Cluster0' database details: Version 8.0.8, Region AWS Bahrain (me-south-1). The 'Collections' tab is selected, displaying the 'DMET.students' collection. It shows storage size (36KB), logical data size (201B), total documents (3), and index sizes (36KB). Below this, there's a search bar for namespaces, a 'Find' button, and a 'Generate queries from natural language in Compass' input field. The 'QUERY RESULTS: 1-2 OF 2' section displays two document snippets:

```
_id: ObjectId('681cc621171521bb6ba42b16')
name : "Rimon"
address : "GUC"
ID : 123456
```

```
_id: ObjectId('681cc6ab171521bb6ba42b17')
name : "Ahmed"
address : "New Cairo"
ID : 654321
```

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas web interface. The top navigation bar includes a search bar, a user dropdown for "Rimon", and various management links like "Access Manager" and "Billing". The left sidebar lists organizational units: "webCourse" (selected), "RIMON'S ORG - 2023-05-01", and "WEBCOURSE". Below these are sections for "DATABASE", "Clusters" (selected), "SERVICES", "Atlas Search", "Stream Processing", "Triggers", "Migration", "Data Federation", and "SECURITY" (Quickstart, Backup, Database Access, Network Access, Advanced). The main content area is titled "Clusters" and displays "RIMON'S ORG - 2023-05-01 > WEBCOURSE". It features a search bar, a "Find a database deployment..." button, and a "Create" button. A cluster named "Cluster0" is highlighted with a red dashed circle. Buttons for "Connect", "View Monitoring", "Browse Collections", and "..." are shown next to it. Below this, a section titled "Visualize Your Data" encourages building dashboards and charts. It includes a "Dismiss" button and a "Explore Charts" button. Three line charts show metrics: "R 0.007" (blue line, peaking at 0.02/s), "Connections 14.0" (blue line, peaking at 14.0), and "In 71.45 B/s" (orange line, peaking at 1.20 KB/s). On the right, "Data Size" is listed as 36.08 KB / 512.00 MB. The bottom navigation bar includes links for "VERSION", "REGION", "TYPE", "BACKUPS", "LINKED APP SERVICES", "ATLAS SQL", and "ATLAS SEARCH". A small green circular icon with a speech bubble is in the bottom right corner.

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas web interface. On the left, a sidebar menu includes sections for Overview, DATABASE (Clusters), SERVICES (Atlas Search, Stream Processing, Triggers, Migration, Data Federation), SECURITY (Quickstart, Backup, Database Access, Network Access, Advanced), and a search bar. The main content area is titled "Connect to your application" and lists several tools:

- Drivers**: Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.).
- Compass**: Explore, modify, and visualize your data with MongoDB's GUI. This item is highlighted with a red dashed circle.
- Shell**: Quickly add & update data using MongoDB's Javascript command-line interface.
- MongoDB for VS Code**: Work with your data in MongoDB directly from your VS Code environment.
- Atlas SQL**: Easily connect SQL tools to Atlas for data analysis and visualization.

At the bottom of the main content area are "Go Back" and "Close" buttons. To the right, a sidebar shows "Rimon's Org" with a "Create" button, and a status panel displays "Data Size 36.11 KB / Last 26 minutes 512.00 MB".

# Atlas: Start with MongoDB

The screenshot shows the MongoDB Atlas 'Clusters' page with a modal dialog titled 'Connect to Cluster0'. The dialog has three steps: 'Set up connection security' (completed), 'Choose a connection method' (completed), and 'Connect' (step 3). The 'Choose a connection method' step is active, showing options for 'MongoDB Compass' and 'MongoDB Shell'. A red dashed oval highlights the 'I have MongoDB Compass installed' button, which is selected. Another red dashed oval highlights the 'Done' button at the bottom right of the dialog.

Clusters | Cloud: MongoDB Clusters

cloud.mongodb.com/v2/6817c0c6a443e05b9aa0f657#/clusters/connect?clusterId=Cluster0

Atlas

Clusters

Connect to Cluster0

Set up connection security

Choose a connection method

Connect

Connecting with MongoDB Compass

I don't have MongoDB Compass installed

I have MongoDB Compass installed

1. Choose your version of Compass

1.38 or later

See your Compass version in "About Compass"

2. Copy the connection string, then open MongoDB Compass

Use this connection string in your application

`mongodb+srv://rimonzelias:<db_password>@cluster0.rkvu3dn.mongodb.net/`

Replace `<db_password>` with the password for the `rimonzelias` user. Ensure any options are URL encoded. You can edit your database user password in Database Access.

RESOURCES

Connect with Compass Import and Export Data

Access your Database Users Troubleshoot Connections

Go Back

Done

Clusters Get Help Rimon

Data Size 20.00 KB / 512.00 MB (0%) Last 7 minutes 512.00 MB

ATLAS SEARCH Create Index

# MongoDB Compass

The screenshot shows the MongoDB Compass application window. The title bar reads "MongoDB Compass". The menu bar includes "Connections", "Edit", "View", and "Help". The main interface has a sidebar on the left labeled "Compass" with sections for "My Queries" and "CONNECTIONS". The "CONNECTIONS" section contains a search bar and a message stating "You have not connected to any deployments." A green button labeled "+ Add new connection" is highlighted with a red dashed oval. The central area features a magnifying glass icon over a cloud, with the text "Welcome to MongoDB Compass" and "To get started, connect to an existing server or". Below this is another green button "+ Add new connection". A light blue box on the right contains the text "New to Compass and don't have a cluster?" and "If you don't already have a cluster, you can create one for free using [MongoDB Atlas](#)". A green button labeled "CREATE FREE CLUSTER" is located at the bottom of this box.

MongoDB Compass

Connections Edit View Help

Compass

Welcome +

My Queries

CONNECTIONS + ...

Search connections

You have not connected to any deployments.

+ Add new connection

Welcome to MongoDB Compass

To get started, connect to an existing server or

+ Add new connection

New to Compass and don't have a cluster?

If you don't already have a cluster, you can create one for free using [MongoDB Atlas](#)

CREATE FREE CLUSTER

# MongoDB Compass

The screenshot shows the MongoDB Compass application interface. The main window title is "MongoDB Compass". The menu bar includes "Connections", "Edit", "View", and "Help". The left sidebar has sections for "Compass" (My Queries) and "CONNECTIONS" (Search connections, Add new connection). The central area is titled "New Connection" with the sub-instruction "Manage your connection settings". It contains fields for "URI" (with a value of "mongodb+srv://rimonzelias:\*\*\*\*\*@cluster0.rkvu3dn.mongodb.net/"), "Edit Connection String" (a toggle switch), "Name" (set to "cluster0.rkvu3dn.mongodb.net"), "Color" (set to "No Color"), and a checkbox for "Favorite this connection" (unchecked). Below these are "Advanced Connection Options" and three buttons at the bottom: "Cancel", "Save", "Connect", and "Save & Connect" (which is highlighted with a red dashed oval). To the right of the connection form are two informational callouts: "How do I find my connection string in Atlas?" and "How do I format my connection string?", each with a "See example" link.

MongoDB Compass

Connections Edit View Help

Compass

{ My Queries

CONNECTIONS

Search connections

You have not connected to any deployments.

+ Add new connection

New Connection

Manage your connection settings

URI ⓘ

mongodb+srv://rimonzelias:\*\*\*\*\*@cluster0.rkvu3dn.mongodb.net/

Edit Connection String ⚙

Name

cluster0.rkvu3dn.mongodb.net

Color

No Color

Favorite this connection

Favoriting a connection will pin it to the top of your list of connections

Advanced Connection Options

Cancel

Save

Connect

Save & Connect

How do I find my connection string in Atlas?

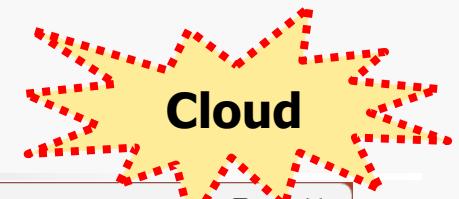
If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.  
See example ↗

How do I format my connection string?

See example ↗

37

# MongoDB Compass



The screenshot shows the MongoDB Compass interface connected to a cluster. The left sidebar lists connections: cluster0.rkvu3dn.mongodb.net (selected), admin, config, and local. The main area shows the 'students' collection with two documents. The first document is:

```
_id: ObjectId('681cc621171521bb6ba42b16')
name : "Rimon"
address : "GUC"
ID : 123456
```

The second document is:

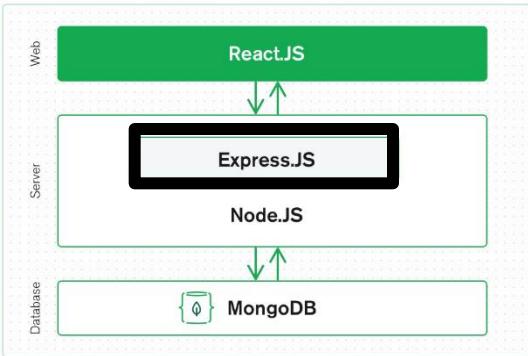
```
_id: ObjectId('681cc6ab171521bb6ba42b17')
name : "Ahmed"
address : "New Cairo"
ID : 654321
```

# MongoDB Compass

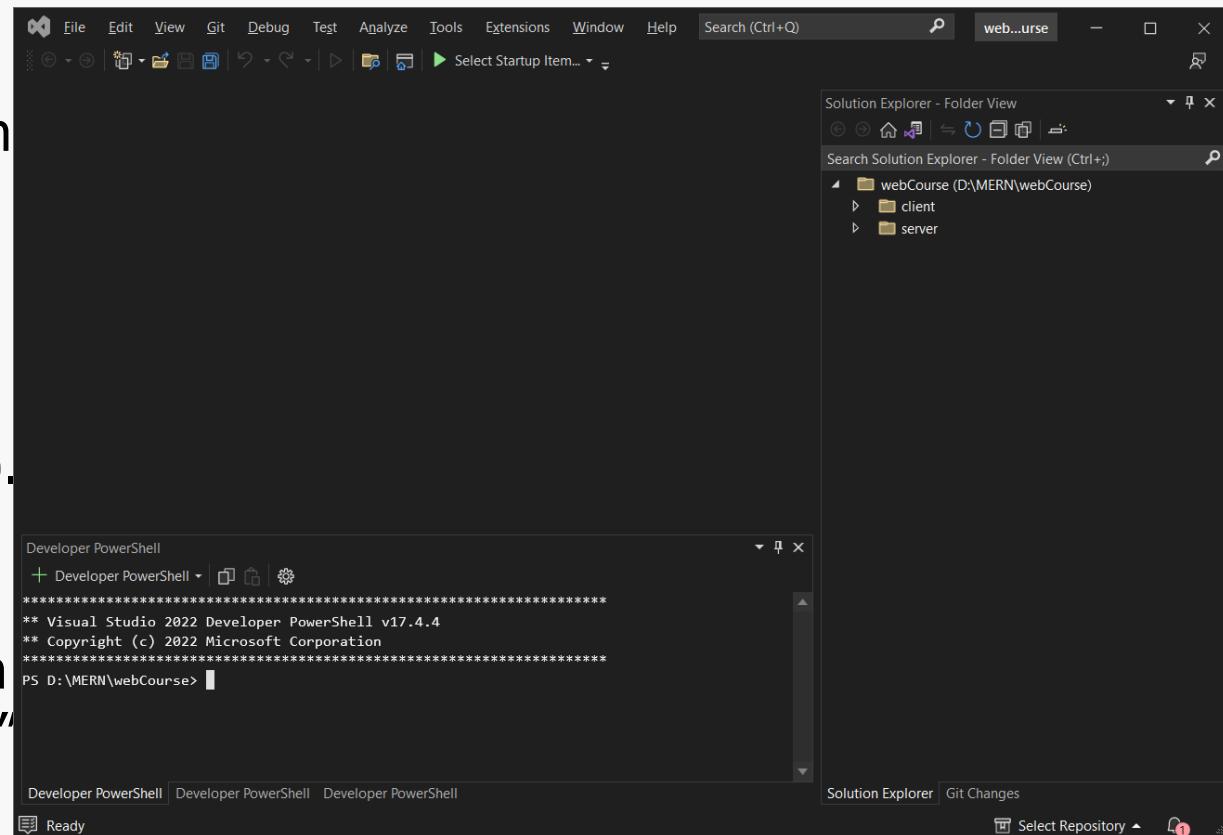
The screenshot shows the MongoDB Compass interface connected to a cluster. The left sidebar displays connections, with 'cluster0.rkvv3dn.mongodb.net' selected. Under this connection, the 'DMET' database is chosen, and its 'students' collection is highlighted. The main workspace shows the 'Documents' tab selected, displaying two documents. A red dashed oval highlights the 'Insert document' button, which is used to add new data. One document is visible with the following fields:

```
_id: ObjectId('681cc6ab171521bb6ba42b17')
name : "Ahmed"
address : "New Cairo"
ID : 654321
```

# Express



- Under your project folder (i.e., webCourse), create two folders; server and client.
- We will use the folder **server** with **Express** (back end) and
- We will use the folder **client** with **React** (front end).
- Right-click on the webCourse folder and choose “Open with Visual Studio”



# Express

The screenshot shows the Microsoft Visual Studio Code interface with a dark theme. In the top left, there's a decorative graphic of overlapping colored squares (yellow, red, blue). The top bar includes the VS Code logo, a search bar with the placeholder "Search (Ctrl+Q)", and a tab labeled "web...urse". The menu bar has options like File, Edit, View, Git, Project, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a "Select Startup Item..." dropdown.

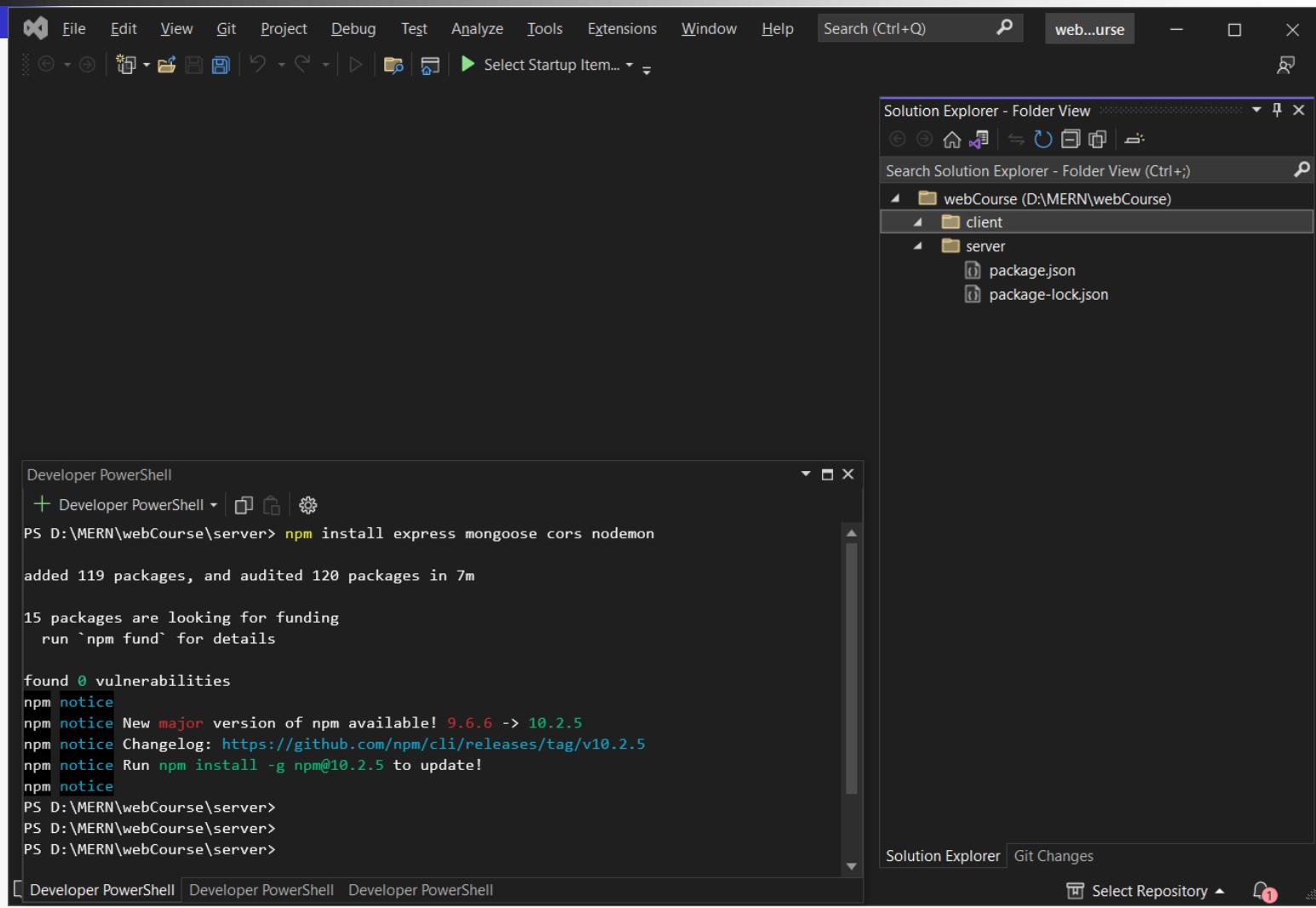
In the bottom left, a "Developer PowerShell" terminal window is open, showing the command `npm init -y` being run in the directory `D:\MERN\webCourse\server`. The output indicates that `package.json` was created. The terminal displays the contents of the `package.json` file:

```
{  
  "name": "server",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\"$Error: no test specified\\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC"  
}
```

To the right of the terminal is the "Solution Explorer" panel, titled "Solution Explorer - Folder View". It shows a folder structure for a project named "webCourse" located at "D:\MERN\webCourse". The "server" folder contains a file named "package.json". A large, light-yellow starburst graphic with a red dashed border is overlaid on the right side of the screen, containing the text "Now we can get other packages". A large white arrow points from the "package.json" file in the Solution Explorer towards this starburst.

At the bottom, there are tabs for "Developer PowerShell" (which is active), "Git Changes", and "Select Repository". The status bar at the very bottom shows the message "Ready".

# Express and Other Packages

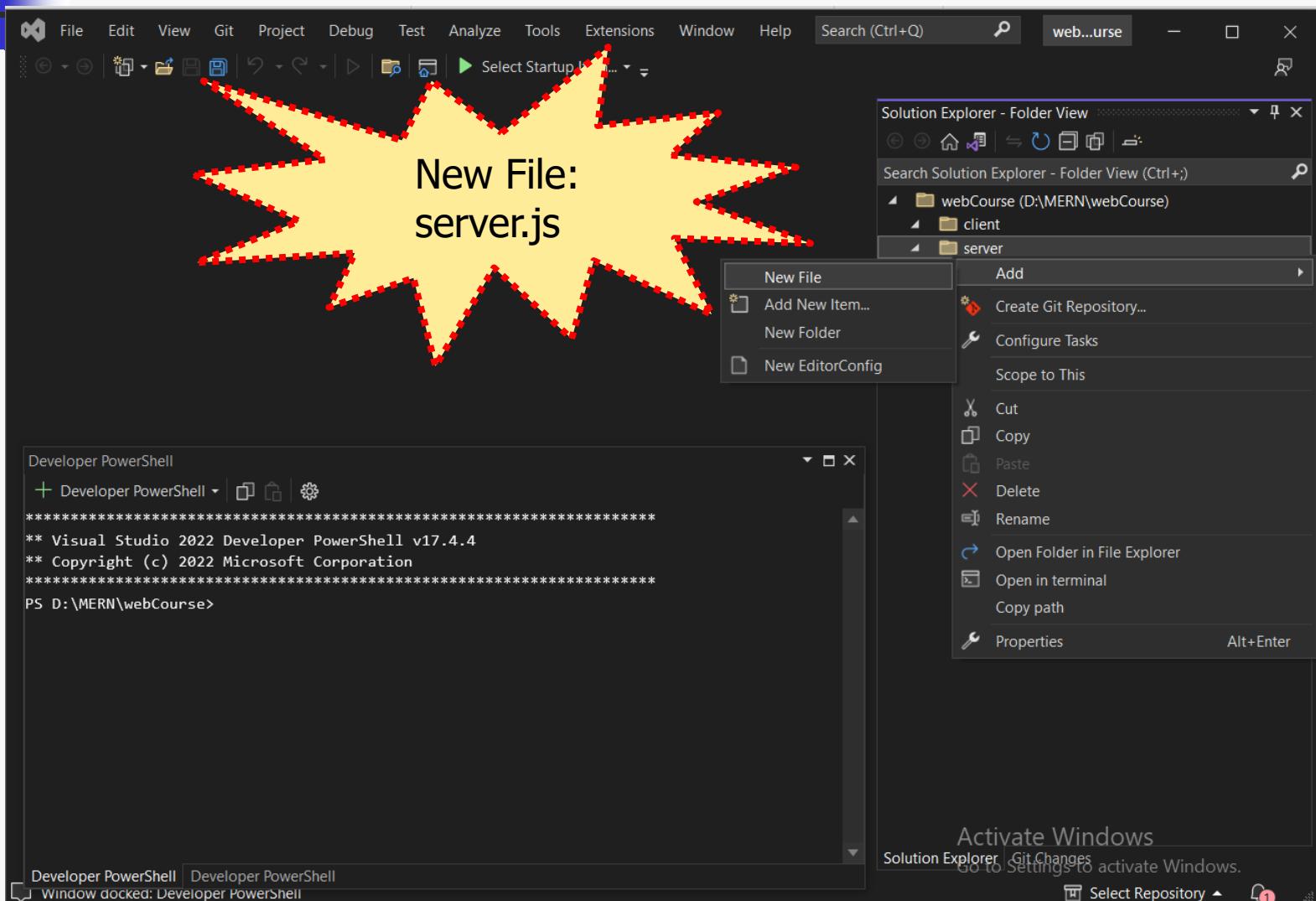


The screenshot shows a dark-themed instance of Visual Studio Code. At the top is the menu bar with options like File, Edit, View, Git, Project, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar. Below the menu is a toolbar with various icons. On the right side, there's a "Solution Explorer - Folder View" panel showing a project structure: webCourse (D:\MERN\webCourse) containing client and server folders, with package.json and package-lock.json files in the server folder. At the bottom left is a "Developer PowerShell" terminal window. The terminal output shows the command `npm install express mongoose cors nodemon` being run, followed by a message indicating 119 packages were added and audited, and 15 packages were looking for funding. It also mentions 0 vulnerabilities and an npm update notice. The terminal prompt shows the current directory as D:\MERN\webCourse\server.

```
Developer PowerShell
PS D:\MERN\webCourse\server> npm install express mongoose cors nodemon
added 119 packages, and audited 120 packages in 7m
15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New major version of npm available! 9.6.6 -> 10.2.5
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.2.5
npm notice Run npm install -g npm@10.2.5 to update!
npm notice
PS D:\MERN\webCourse\server>
PS D:\MERN\webCourse\server>
PS D:\MERN\webCourse\server>
```

# server.js



# server.js

A screenshot of the Visual Studio Code (VS Code) interface. The main editor window shows the file `server.js` with the following code:

```
const express = require("express")
const app = express()

app.listen("3001", ()=>{
  console.log("Server working...")
})
```

The line `app.listen("3001", ()=>{` is highlighted with a red dotted rectangle. A callout box labeled "Port 3001" points to this line. The line `console.log("Server working...")` is also highlighted with a red dotted rectangle. A callout box labeled "What is done when listening to port 3001" points to this line.

The terminal at the bottom shows the command `PS D:\MERN\webCourse\server> node server` and the output `Server working...`. A callout box labeled "Notice that no prompt appears!" points to the terminal area.

VS Code status bar: 195 %, No issues found, Ln: 7 Ch: 3 TABS CRLF

Solution Explorer - Folder View: webCourse (D:\MERN\webCourse) / client / server / package.json / package-lock.json / server.js

Search Solution Explorer - Folder View (Ct)

MERN Stack

Activate Windows  
Go to Settings to activate Windows.

(c) 2023, Dr. R. Elias

Ready

Select Repository

44

# server.js

The screenshot shows a Visual Studio Code interface with the following components:

- Code Editor:** Displays the `server.js` file content:

```
const express = require("express")
const app = express()

app.listen("3001", ()=>{
  console.log("Server working...")
})
```
- Solution Explorer:** Shows the project structure: `webCourse (D:\MERN\webCourse)` containing `client`, `server`, `package.json`, `package-lock.json`, and `server.js`.
- Terminal:** Shows the command `Developer PowerShell` running in the background.
- Browser:** A browser window with an **Error** message at `localhost:3001`:

Cannot GET /

localhost:3001
- Bottom Status Bar:** Includes "Activate Windows", "Solution Explorer: Git Changes", "Go to Settings to activate Windows.", "Select Repository", and a notification icon.

# server.js

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The main editor window displays the file `server.js` with the following code:

```
1 const express = require("express")
2 const app = express()
3
4 app.get("/", (req, res)=>{
5     res.send("Test")
6 }
7 )
8
9 app.listen("3001", ()=>{
10    console.log("Server working...")
11 })
```

Annotations in red text are overlaid on the code:

- A box labeled "Where to execute" covers the first two lines of code.
- A box labeled "What to execute in the browser" covers the `res.send("Test")` line.
- A box labeled "What to show in the terminal" covers the `console.log("Server working...")` line.
- A box at the bottom labeled "Ctrl+c to exit and re-type node server" covers the terminal area.

The Solution Explorer on the right shows a project structure:

```
webCourse (D:\MERN\webCourse)
  + client
  + server
    package.json
    package-lock.json
    server.js
```

The Developer PowerShell terminal at the bottom shows the command `node server` being run, with the output "Server working...".

Bottom status bar: Item(s) Saved, Select Repository, 1, Activate Windows, Go to Settings to activate Windows.

# server.js

A screenshot of the Visual Studio Code (VS Code) interface. The main editor window shows the file `server.js` with the following code:

```
const express = require("express")
const app = express()

app.get("/students", (req, res) => {
  res.send("Test")
})

app.listen("3001", () => {
  console.log("Server working...")
})
```

The code uses green dashed boxes to highlight the `express` module import, the `app` variable declaration, the `get` method call, the `res.send` call, the `listen` method call, and the `console.log` statement. A red dashed circle highlights the word `Test` in the browser's address bar. The browser window at the bottom shows the URL `localhost:3001/students` and the response `Test`. The status bar at the bottom left shows the path `PS D:\MERN\webCourse\server>` and the command `node server`, with the output `Server working...`.

# Mongo DB + Express: server.js

VS Code interface showing the file `server.js*`. The code defines an Express app with a route for '/students' and starts it on port 3001.

```
const express = require("express")
const app = express()

app.get("/students", (req, res) => {
  res.send("Server working...")
})

app.listen("3001", () => {
  console.log("Server working...")
})
```

The line `res.send("Server working...")` is highlighted with a red dashed box and a red arrow points from the text "We want to get the info from MongoDB and display it here!" to it.

We want to get the info from MongoDB and display it here!

Output window shows:

```
PS D:\MERN\webCourse\server> node server
Server working...
```

# server.js

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with a Node.js application running in a browser. The code editor displays `server.js` with syntax highlighting and a red dashed box around the mongoose connection code. The terminal at the bottom shows the command `node server` being run and the output "Server working...". A browser window shows the URL `localhost:3001/students` with the response "Test".

```
const express = require("express")
const app = express()

const mongoose = require("mongoose")
mongoose.connect("mongodb://localhost:27017/DMET")

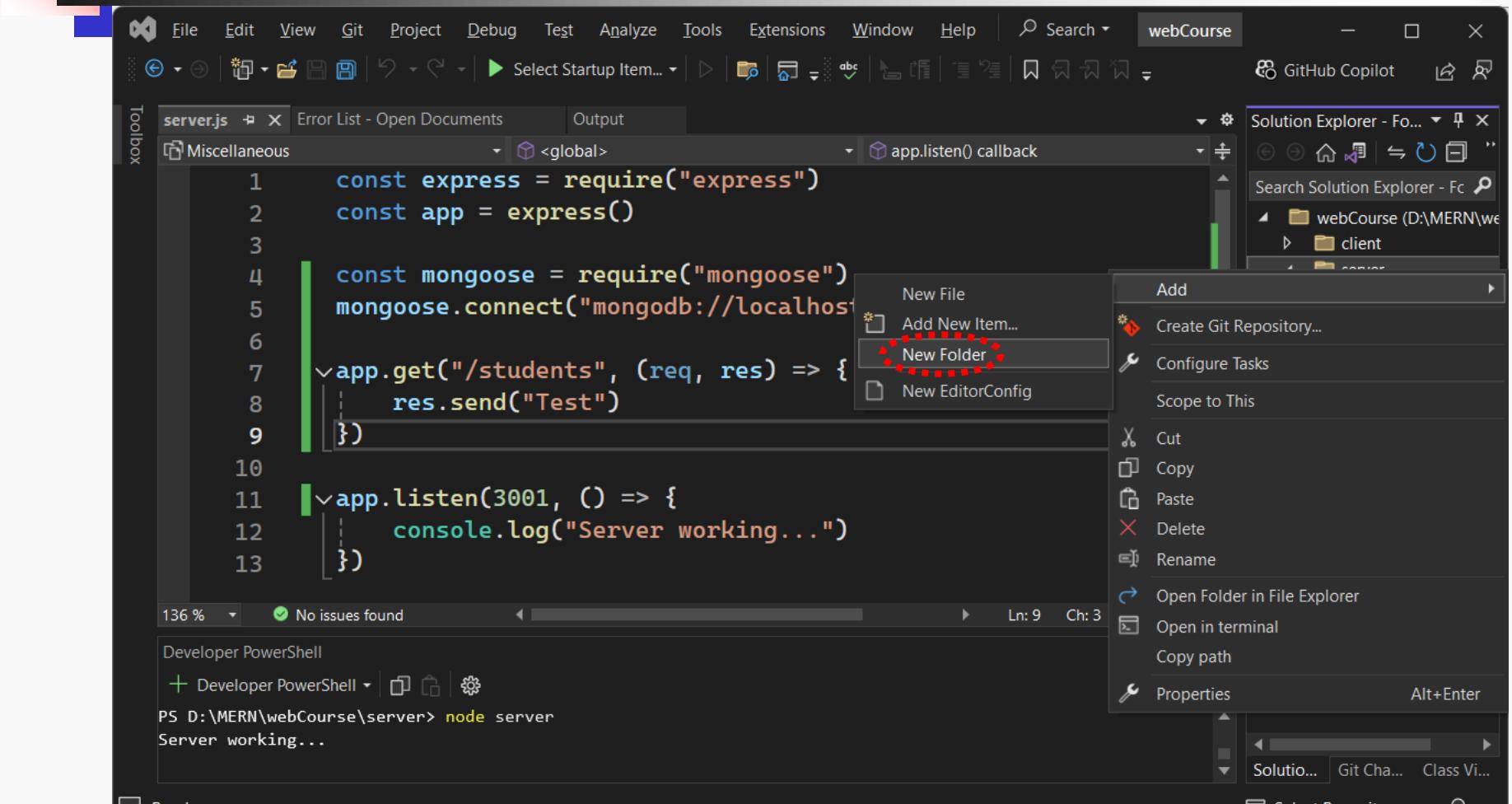
app.get("/students", (req, res) => {
    res.send("Test")
})

app.listen(3001, () => {
    console.log("Server working...")
})
```

VS Code interface elements include:

- File, Edit, View, Git, Project, Debug, Test, Analyze, Tools, Extensions, Window, Help menus.
- Search bar with "webCourse".
- Toolbar with icons for file operations.
- Solution Explorer sidebar showing project structure: `webCourse` (D:\MERN\we), `client`, `server`, `JS server.js`, `package-lock.json`, `package.json`.
- Output panel showing "No issues found".
- Terminal showing "Developer PowerShell" and the command `PS D:\MERN\webCourse\server> node server` followed by "Server working...".
- Browser window showing the result of the request to `localhost:3001/students`.

# server.js: Interaction using Model



A screenshot of the Visual Studio Code (VS Code) interface. The main editor window displays the file `server.js` with the following code:

```
1  const express = require("express")
2  const app = express()
3
4  const mongoose = require("mongoose")
5  mongoose.connect("mongodb://localhost:27017/test")
6
7  app.get("/students", (req, res) => {
8      res.send("Test")
9  })
10
11 app.listen(3001, () => {
12     console.log("Server working...")
13 })
```

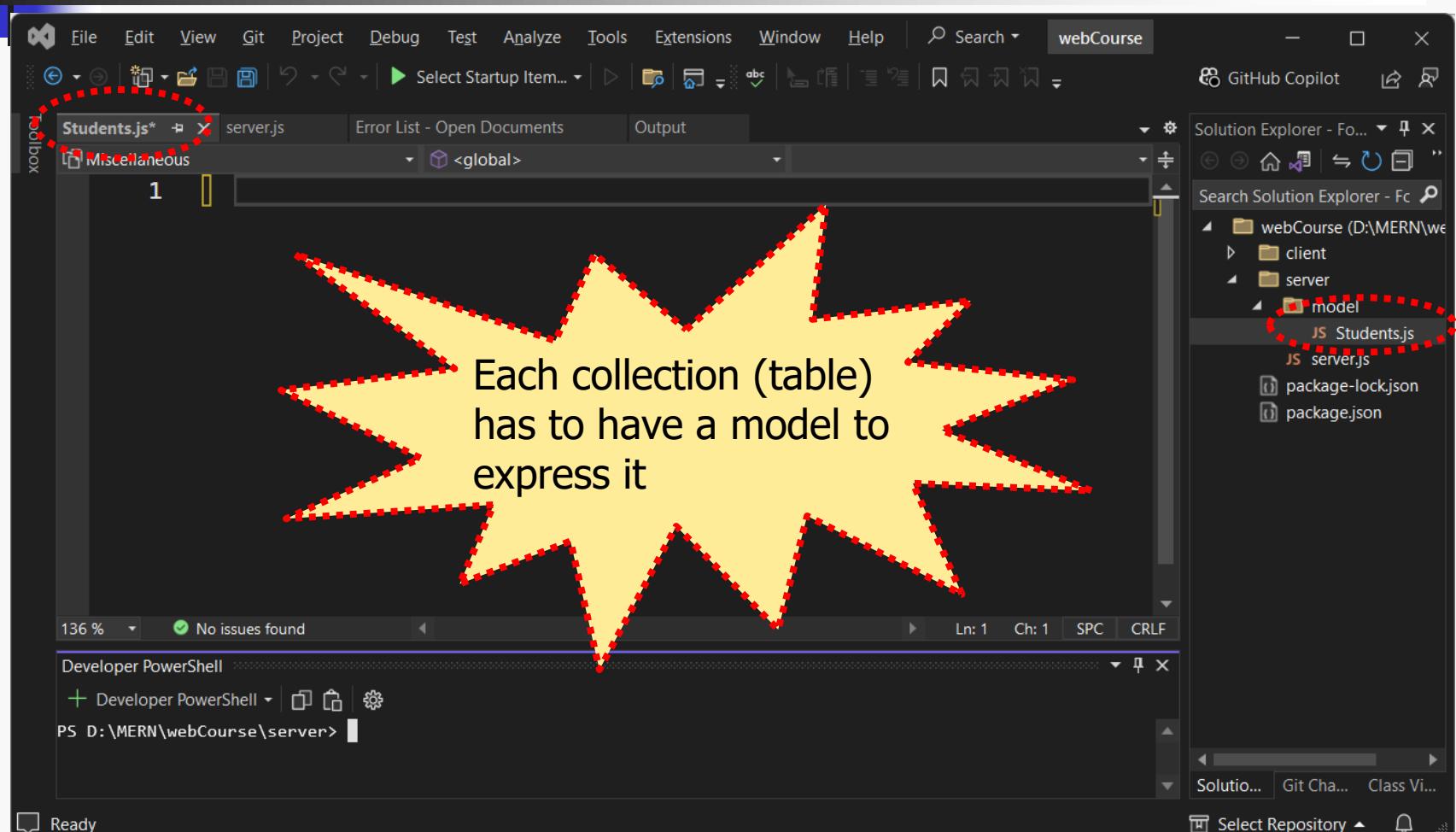
The code uses Express.js and Mongoose to handle a GET request to `/students` and log "Server working..." when the server starts.

A context menu is open over the line `const mongoose = require("mongoose")`, with the option `New Folder` highlighted with a red dotted selection box. The menu also includes other options like `New File`, `Add New Item...`, and `New EditorConfig`.

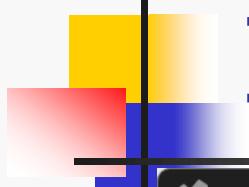
The bottom terminal shows the command `node server` being run, with the output "Server working..." displayed.



# Persons.js: Interaction using Model

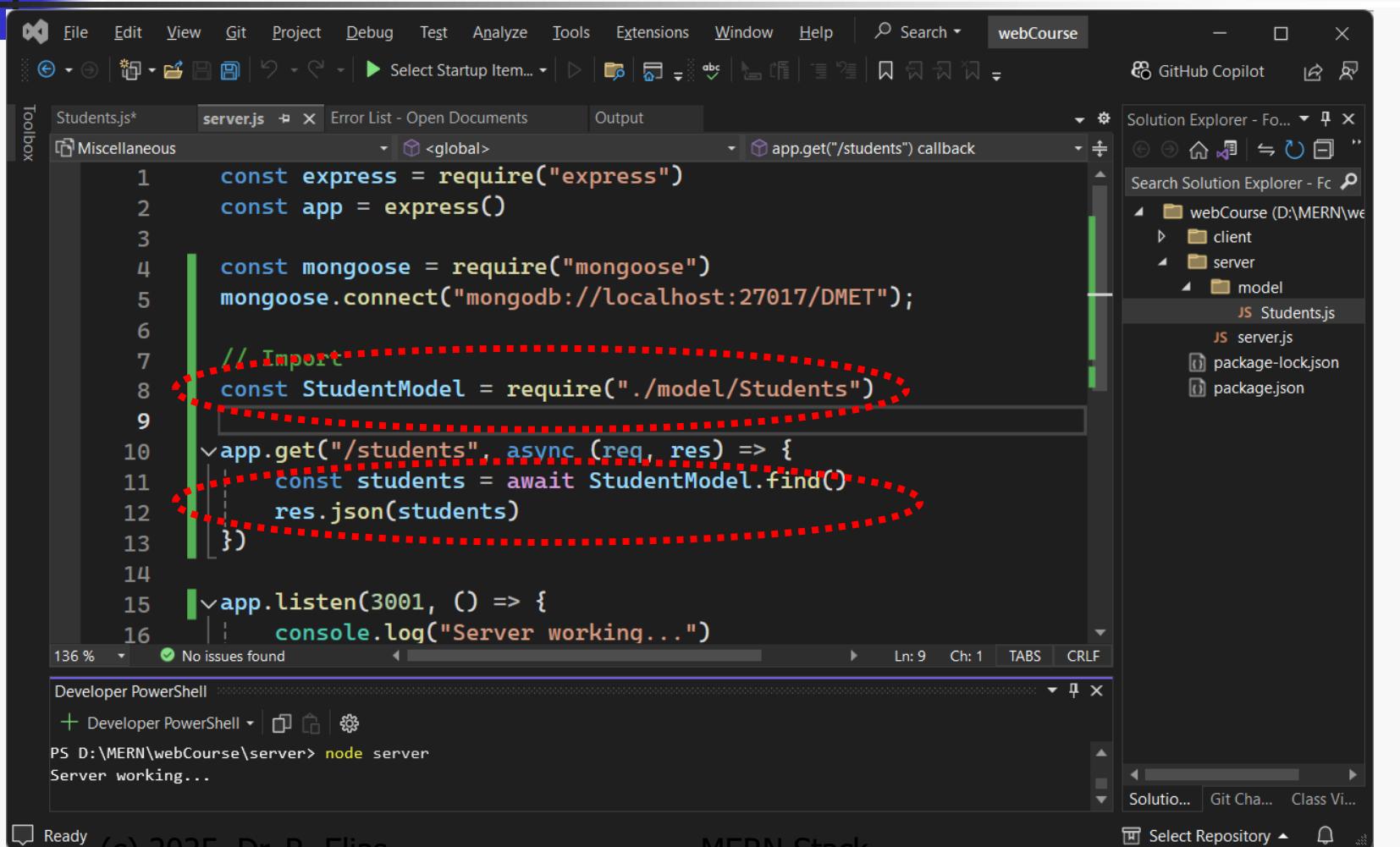


# Persons.js: Interaction using Model



```
File Edit View Git Project Debug Test Analyze Tools Extensions Window Help Search webCourse
GitHub Copilot
Students.js* server.js Error List - Open Documents Output
Miscellaneous <global> StudentModel
1 const mongoose = require("mongoose")
2
3 const StudentSchema = new mongoose.Schema({
4   name: String, address: String, ID: Number
5 },
6   { collection: 'students' }
7 )
8
9 const StudentModel = mongoose.model("Students", StudentSchema)
10 module.exports = StudentModel
136 % No issues found
Developer PowerShell
+ Developer PowerShell | 
PS D:\MERN\webCourse\server>
Ready Select Repository
(c) 2025, Dr. R. Liias MERN Stack
Solution Explorer - Fo... Search Solution Explorer - Fc
webCourse (D:\MERN\webCourse)
client
server
model
Students.js
server.js
package-lock.json
package.json
Solution... Git Cha... Class Vi...
52
```

# Mongo DB + Express: Interaction using Model



A screenshot of the Visual Studio Code (VS Code) interface. The main area shows a code editor with a dark theme containing JavaScript code for a Node.js application. The code imports 'express' and 'mongoose', connects to a MongoDB database, and defines an endpoint to retrieve student data from a 'StudentModel'. A red dashed box highlights the import statement and the find() call. The 'Solution Explorer' panel on the right shows the project structure with files like 'Students.js', 'server.js', 'package-lock.json', and 'package.json'. The 'Developer PowerShell' panel at the bottom shows the command 'node server' being run, with the output 'Server working...'. The status bar at the bottom indicates the file is 'Ready'.

```
const express = require("express")
const app = express()

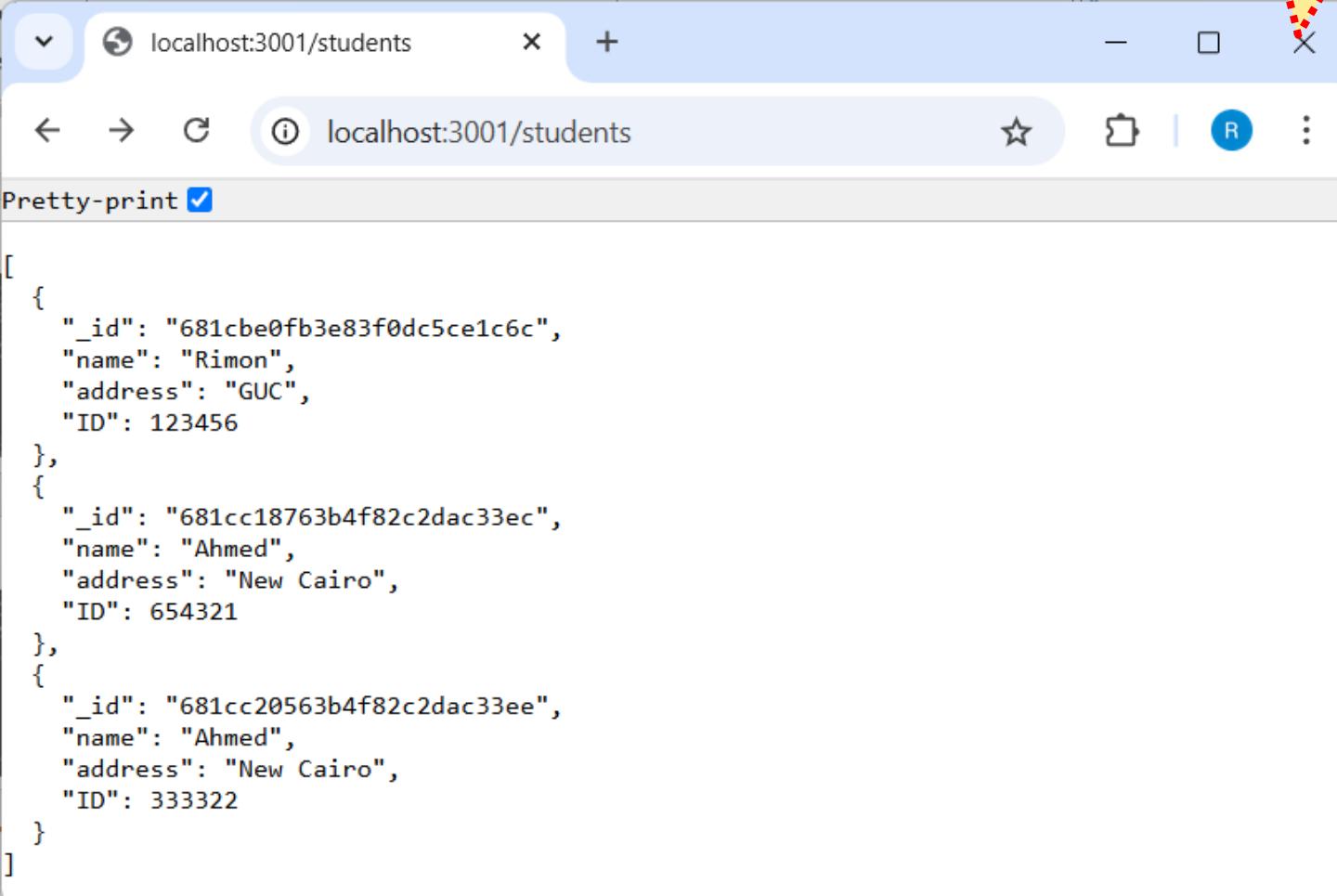
const mongoose = require("mongoose")
mongoose.connect("mongodb://localhost:27017/DMET");

// Import
const StudentModel = require("./model/Students")

app.get("/students", async (req, res) => {
  const students = await StudentModel.find()
  res.json(students)
})

app.listen(3001, () => {
  console.log("Server working...")
})
```

# Back-end



```
[  
  {  
    "_id": "681cbe0fb3e83f0dc5ce1c6c",  
    "name": "Rimon",  
    "address": "GUC",  
    "ID": 123456  
  },  
  {  
    "_id": "681cc18763b4f82c2dac33ec",  
    "name": "Ahmed",  
    "address": "New Cairo",  
    "ID": 654321  
  },  
  {  
    "_id": "681cc20563b4f82c2dac33ee",  
    "name": "Ahmed",  
    "address": "New Cairo",  
    "ID": 333322  
  }  
]
```

# React

The screenshot shows a Visual Studio Code interface with a dark theme. The top bar includes File, Edit, View, Git, Project, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar set to "webCourse". The left sidebar has a "Toolbox" tab selected. The main editor area shows "server.js" with the following code:

```
const app = express()

const mongoose = require("mongoose")
mongoose.connect("mongodb://localhost:27017/DMET");

// Import
const StudentModel = require("./model/Students")
```

The status bar at the bottom left indicates "136 %", "No issues found", and "Ln: 9 Ch: 1 TABS CRLF". Below the editor is a "Developer PowerShell" terminal window with the following output:

```
PS D:\MERN\webCourse\client> npx create-react-app .
Need to install the following packages:
  create-react-app@5.1.0
Ok to proceed? (y) y

Creating a new React app in D:\MERN\webCourse\client.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1322 packages in 3m
268 packages are looking for funding
```

To the right of the editor is the "Solution Explorer" panel, which displays the project structure:

```
WebCourse (D:\MERN\webCourse)
  - client
    - public
    - src
    - .gitignore
    - package-lock.json
    - package.json
    - README.md
  - server
    - model
      - JS Students.js
      - JS server.js
    - package-lock.json
    - package.json
```

The bottom right corner shows the page number "55".

The screenshot shows a Microsoft Visual Studio Code interface with the following components:

- Top Bar:** File, Edit, View, Git, Project, Debug, Test, Analyze, Tools, Extensions, Window, Help, Search (with "webCourse" selected).
- Left Sidebar:** Toolbox icon.
- Document Area:** Shows two files: "Students.js\*" and "server.js". "server.js" is highlighted with a red dashed circle. The code for "server.js" is as follows:

```
const app = express();

const mongoose = require("mongoose");
mongoose.connect("mongodb://localhost:27017/test");

// Import
const StudentModel = require("./models/StudentModel");

app.get("/students", (req, res) => {
  StudentModel.find((err, students) => {
    if (err) {
      res.send(err);
    } else {
      res.json(students);
    }
  });
});

app.listen(3001, () => {
  console.log(`Server is running on port 3001`);
});
```

- Output Area:** Shows "No issues found".
- Terminal:** Developer PowerShell window showing the command "npm start" being run in the directory "D:\MERN\webCourse\client". The output shows the server starting on port 3001 and the client starting on port 3000. A red dashed circle highlights the "Starting the development server..." and "Compiled successfully!" messages.
- Browser Preview:** A browser window titled "React App" showing the URL "localhost:3000". A red dashed circle highlights the URL bar.
- Right Side:** Solution Explorer and Search Solution Explorer panes.



Edit `src/App.js` and save to reload.

[Learn React](#)

# React

Students.js\*

server.js

Miscellaneous

```
const app = express()
const mongoose = require("mongoose")
mongoose.connect("mongodb://localhost:27017/DMET")
// Import
const StudentModel = require("./model/Students")
```

Output

136 % No issues found

Developer PowerShell

PS D:\MERN\webCourse\client> npm start

> client@0.1.0 start  
> react-scripts start

(node:27792) [DEP\_WEBPACK\_DEV\_SERVER\_ON\_AFTER\_SETUP\_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.  
(Use `node --trace-deprecation ...` to show where the warning was created)  
(node:27792) [DEP\_WEBPACK\_DEV\_SERVER\_ON\_BEFORE\_SETUP\_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.  
Starting the development server...  
Compiled successfully!

You can now view client in the browser.

Solution Explorer - Folder View

Search Solution Explorer - Folder View (C)

webCourse (D:\MERN\webCourse)

- client
  - public
    - favicon (2).ico
    - favicon.ico
    - index.html
    - logo192 (2).png
    - logo192.png
    - logo512.png
    - manifest.json
    - robots.txt
  - src
    - App.css
    - App.js
    - App.test.js
    - index.css
    - index.js
    - logo.svg
    - reportWebVitals.js
    - setupTests.js

gitignore

package-lock.json

package.json

README.md

Solution Explorer Git Changes Class View

Delete highlighted files

The screenshot shows a dual-monitor setup of Microsoft Visual Studio Code. The left monitor displays the client-side code, and the right monitor displays the server-side code.

**Left Monitor (Client Side):**

- File Explorer:** Shows files like `index.js*`, `index.html`, `App.js`, and `Students.js*`.
- Code Editor:** The `index.js` file contains the following code:

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';

6 const root = ReactDOM.createRoot(document.getElementById('root'));
7 root.render(
8   <React.StrictMode>
9     <App />
</React.StrictMode>
);
13
14 // If you want to start measuring performance in your app
15 // to log results (for example: reportWebVitals(console.log))
16 // then you can import the reporting library like so:
17 // import * as reportWebVitals from './reportWebVitals';
```

- Output:** Shows "Successfully installed TypeScript typings".
- Terminal:** Developer PowerShell output: PS D:\MERN\webCourse\client> npm start

**Right Monitor (Server Side):**

- Solution Explorer:** Shows the project structure for `webCourse` (D:\MERN\webCourse). It includes `client`, `server`, and `model` folders, along with various files like `index.html`, `App.js`, `Students.js`, `server.js`, `package.json`, and `README.md`.
- Code Editor:** The `index.js` file in the `server` folder contains:

```
1 import express from 'express';
2 import mongoose from 'mongoose';
3 import cors from 'cors';
4 import dotenv from 'dotenv';
5 import { ApolloServer } from 'apollo-server-express';
6 import { typeDefs, resolvers } from './schema';
7 import { Students } from './model';

8 const app = express();
9 app.use(cors());
10 app.use(express.json());
11
12 const apolloServer = new ApolloServer({
13   typeDefs,
14   resolvers,
15   context: ({ req }) => ({ req })
16 });
17
18 apolloServer.applyMiddleware({ app });
19
20 mongoose.connect('mongodb://localhost:27017/test', {
21   useNewUrlParser: true,
22   useUnifiedTopology: true
23 });
24
25 app.listen(4000, () => {
26   console.log('Server is running on port 4000');
27 });
```

- Output:** Shows "Successfully installed TypeScript typings".
- Terminal:** Developer PowerShell output: PS D:\MERN\webCourse\server> npm start

**Bottom Status Bar:** Shows "Successfully installed TypeScript typings".

# App.js

The screenshot illustrates a development environment for a MERN Stack application. The top half shows the Visual Studio Code interface with two tabs open: `index.js` and `App.js`. The `App.js` tab is active, displaying the following code:

```
function App() {
  return (
    <h1>Front end working..</h1>
  );
}

export default App;
```

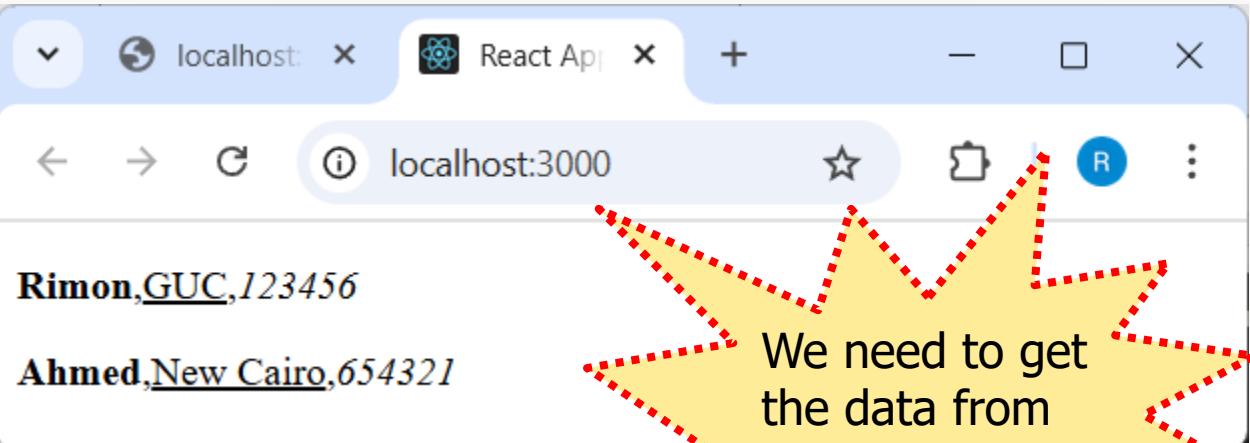
The bottom half shows a browser window displaying the output of the application at `localhost:3000/students`, which shows the text "Front end working..".

The Solution Explorer on the right side shows the project structure:

- `webCourse` (Selected)
- `client`
  - `public`
    - `index.html`
  - `src`
    - `App.css`
    - `App.js`
    - `index.js`
- `server`
  - `server.js`
  - `model`
    - `Students.js`
- `gitignore`
- `package-lock.json`
- `package.json`
- `README.md`

Large arrows indicate the flow from the code in `App.js` to the browser output and the project structure.

# App.js



The screenshot shows the Visual Studio Code interface. The left pane displays the code for 'App.js':1 export default function App() {  
2 const student = [  
3 {name: "Rimon", address: "GUC", ID: 123456},  
4 {name: "Ahmed", address: "New Cairo", ID: 654321}  
5 ]  
6  
7 return (  
8 student.map(student => {  
9 return( <p><b>{student.name}</b>,<br/>  
10 <u>{student.address}</u>,<br/>  
11 <i>{student.ID}</i></p>)  
12 })  
13 //<h1>Front end working..</h1>  
14 );  
15}The right pane shows the project file structure under 'File Explorer':

```
App  
  student  
  index.js  
  index.html  
  App.js  
  JS index.js  
  .gitignore  
  package-lock.json  
  package.json  
  README.md  
  server  
    model  
      JS Students.js  
      JS server.js  
      package-lock.json  
      package.json
```

The bottom status bar indicates 'webpack compiled successfully'.

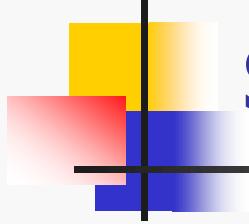
# Axios

The screenshot shows a Microsoft Visual Studio Code interface with the following elements:

- Top Bar:** File, Edit, View, Git, Project, Debug, Test, Analyze, Tools, Extensions, Window, webCourse.
- Toolbar:** Includes icons for file operations like Open, Save, Find, and Replace, along with GitHub Copilot and other developer tools.
- Code Editor:** Shows a Developer PowerShell terminal window. The command `npm install axios` is being run, followed by output indicating 11 packages were added and audited, and 17 packages are looking for funding. It also mentions 0 vulnerabilities.
- Toolbox:** A vertical sidebar on the left.
- Solution Explorer:** A sidebar on the right showing the project structure: webCourse (D:\MERN\webCourse) with client, public (index.html), src (App.css, App.js, index.js), and server folders containing .gitignore, package-lock.json, package.json, and README.md.
- Bottom Status Bar:** Ready, Select Repository, Git Changelog, Class View.

A red dashed arrow points from the text "Package to call the api" in a tooltip to the word "axios" in the terminal output.

Package to call the api  
<http://localhost:3001/students>  
You may use fetch as well



# server.js

**npm install cors**

```
const express = require("express")
const app = express()

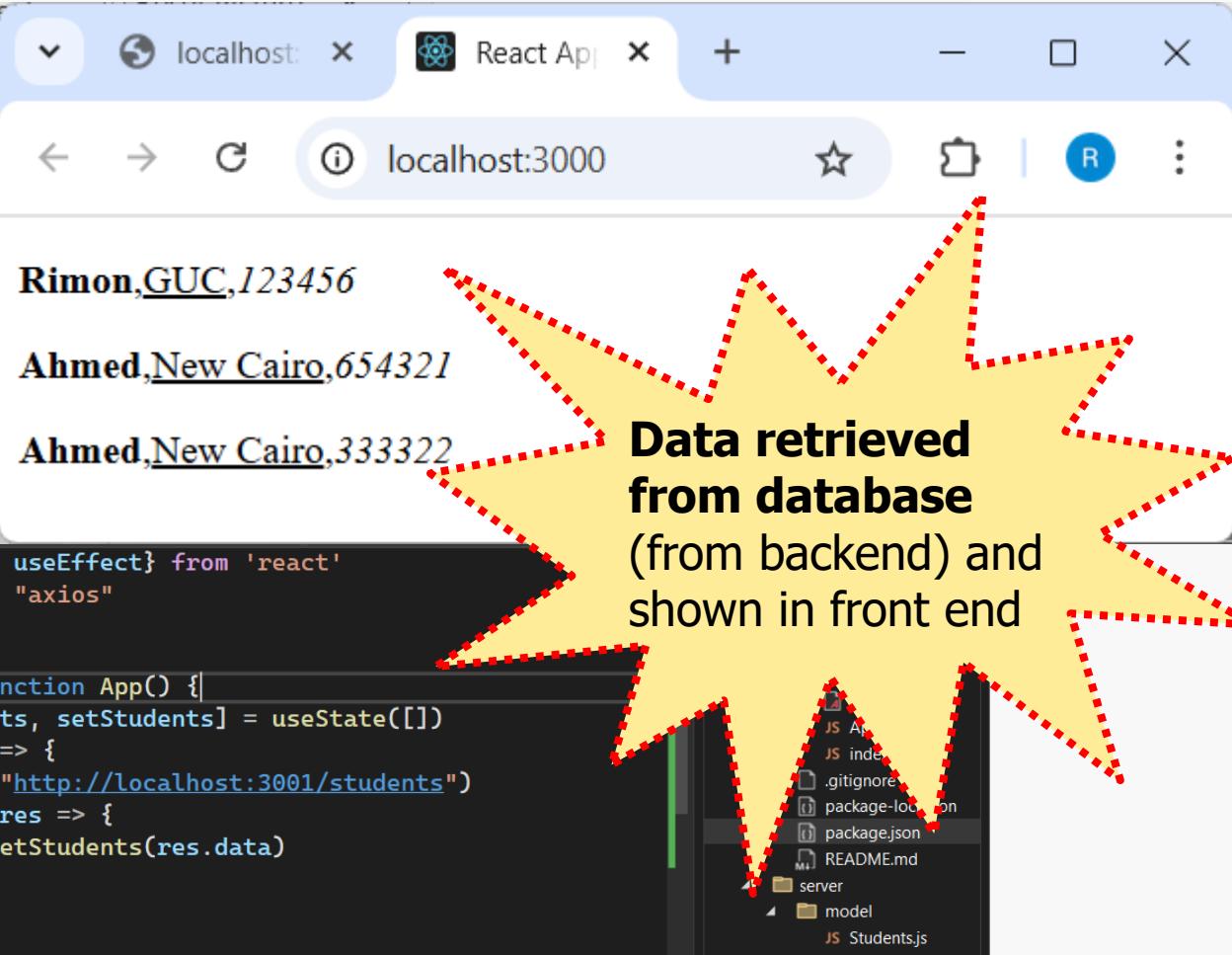
var cors = require('cors');
app.use(cors());

const mongoose = require("mongoose")
```

Add these lines

# App.js

```
1 import {useState, useEffect} from 'react'
2 import axios from "axios"
3
4 export default function App() {
5   const [students, setStudents] = useState([])
6   useEffect(() => {
7     axios.get("http://localhost:3001/students")
8       .then(res => {
9         setStudents(res.data)
10      })
11    }, [])
12
13   return (
14     students.map(student => {
15       return( <p><b>{student.name}</b>,
16             <u>{student.address}</u>,
17             <i>{student.ID}</i></p>)
18     })
19   )
20 }
```



Make sure you are running both  
**node server** and  
**npm start**

# User Interface in App.js

The screenshot shows the Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows files like index.js, index.html, App.js (the active file), Students.js, and server.js.
- Search Bar:** Displays "Search webCourse".
- Code Editor:** The `App.js` file contains the following code:

```
46 <form onSubmit={insert}>
47   Name: <input type="text" placeholder="name" /> <br />
48   Address: <input type="text" placeholder="address" /> <br />
49   ID: <input type="text" placeholder="ID" /> <br />
50   <input type="submit" value="Insert Student" />
51 </form>
```

A red dashed circle highlights the `onSubmit={insert}` event handler, and a callout box labeled "To be created" points to it.
- Terminal:** Shows the URL `localhost:3000/?`.
- Browser Preview:** A light blue window displays the rendered user interface with three input fields for Name, Address, and ID, and a submit button labeled "Insert Student".
- Status Bar:** Shows "Ln: 51 Ch: 6 SPC MIXED".
- Bottom Right:** Includes a "Select Repository" button and a page number "64".

# server.js

The screenshot shows a development environment with a code editor and a browser window.

**Code Editor (VS Code):**

- File:** index.js, index.html, App.js\*, Students.js, server.js (active tab), Miscellaneous.
- Content (server.js):**

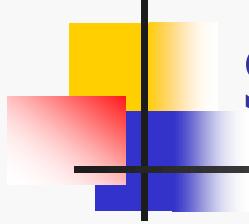
```
16     res.json(students)
17   })
18
19 // insert
20 app.post("/insert", async (req, res) => {
21   const student = req.body // Info sent
22   const newStudent = new StudentModel(student)
23   await newStudent.save()
24
25   res.json(student)
26 }
27
28 app.listen(3001, () => {
29   console.log("Server working...")
}
```

**Browser:** localhost:3000/?

Name: Marwan  
Address: Giza  
ID: 678678  
Insert Student

# App.js

```
5      const [students, setStudents] = useState([])
6      const [name, setName] = useState("")
7      const [address, setAddress] = useState("")
8      const [ID, setID] = useState("")  
  
17  
18      const insert = async (e) => {
19          await Axios.post("http://localhost:3001/insert", {
20              name: name, // "Emad",
21              address: address, // "GUC",
22              ID: ID // 222244
23          })
24          alert('Inserted!')  
  
34      <form onSubmit={insert}>
35          Name: <input type="text" value={name} placeholder="name" onChange={e=>setName(e.target.value)} /> <br />
36          Address: <input type="text" value={address} placeholder="address" onChange={e => setAddress(e.target.value)} />
37          ID: <input type="text" value={ID} placeholder="ID" onChange={e => setID(e.target.value)} /> <br />
38          <input type="submit" value="Insert Student" />
39      </form>
```



# server.js

```
1 const express = require("express")
2 const app = express()
3 app.use(express.json())
```

# Interface & Compass

The image displays three distinct MongoDB interfaces:

- Left Panel (Form):** A simple web-based form for inserting student data. It includes fields for Name (Marwan), Address (Giza), and ID (678678), along with a "Insert Student" button.
- Middle Panel (Compass):** The MongoDB Compass interface. The title bar shows "MongoDB Compass - localhost:27017/DMET.students". The top navigation bar includes Connections, Edit, View, Collection, and Help. Below the title bar, there's a "Compass" logo and a "My Queries" section. The main area is titled "Connections (1)" and shows a list with "localhost:27017" expanded to reveal the "DMET" database, which contains the "students" collection. The "Documents" tab is selected, displaying four documents with their \_id, name, address, and ID values:
  - `_id: ObjectId('681cbe0fb3e83f0dc5ce1c6c')`  
name : "Rimon"  
address : "GUC"  
ID : 123456
  - `_id: ObjectId('681cc18763b4f82c2dac33ec')`  
name : "Ahmed"  
address : "New Cairo"  
ID : 654321
  - `_id: ObjectId('681cc20563b4f82c2dac33ee')`  
name : "Ahmed"  
address : "New Cairo"  
ID : 333322
  - `_id: ObjectId('681dad10845506074864d4f7')`  
name : "Marwan"  
address : "Giza"  
ID : 678678  
\_\_v : 0
- Right Panel (Local):** A colorful interface titled "Local" featuring a yellow background with red dotted lines forming a map-like pattern. It includes a "Open MongoDB shell" button and other interface elements.

# Back-end



```
Pretty-print  [ {    "_id": "681cbe0fb3e83f0dc5ce1c6c",    "name": "Rimon",    "address": "GUC",    "ID": 123456 }, {    "_id": "681cc18763b4f82c2dac33ec",    "name": "Ahmed",    "address": "New Cairo",    "ID": 654321 }, {    "_id": "681cc20563b4f82c2dac33ee",    "name": "Ahmed",    "address": "New Cairo",    "ID": 333322 }, {    "_id": "681dad10845506074864d4f7",    "name": "Marwan",    "address": "Giza",    "ID": 678678,    "__v": 0 } ]
```