Junior phase

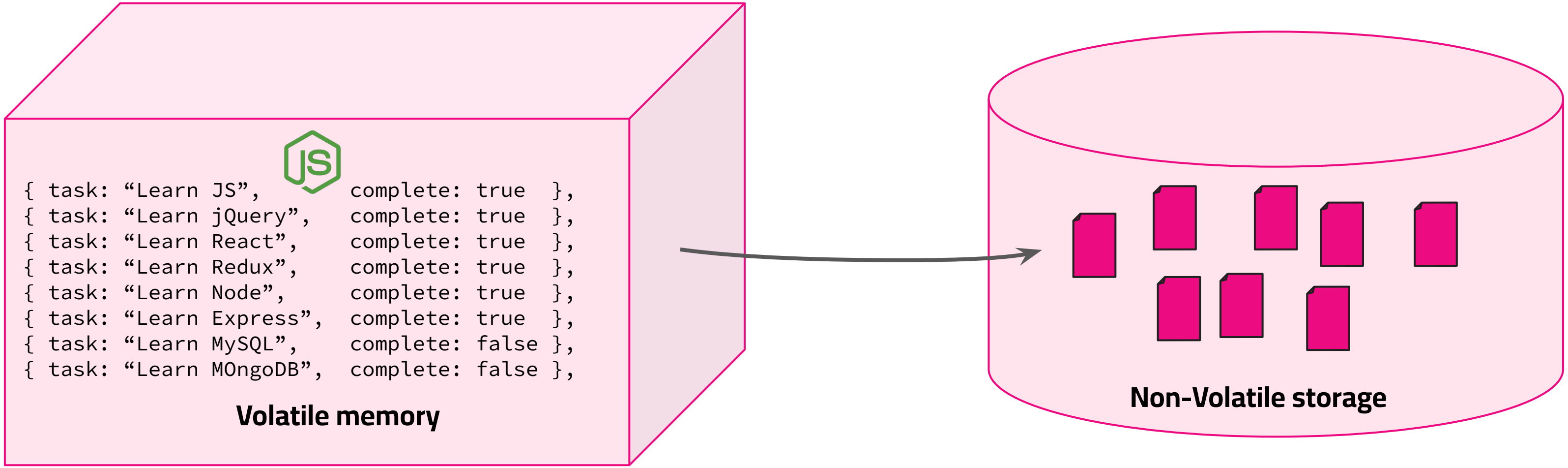
Database Management Systems

An introduction to DMS

Duration: 30 minutes
Q&A: 5 minutes by the end of the lecture

In our previous app:

We used the Node.js **fs** module to store data in files.



Our persistance solution should

- Only rewrite portions of state on change
- Be capable of storing more than can be held in memory
- Efficiently access and address the stored information in multiple, flexible, useful ways
- Allow more than one process simultaneous access
- Avoid inconsistency of state due to changes
- Retain information through a power failure

Our persistance solution should

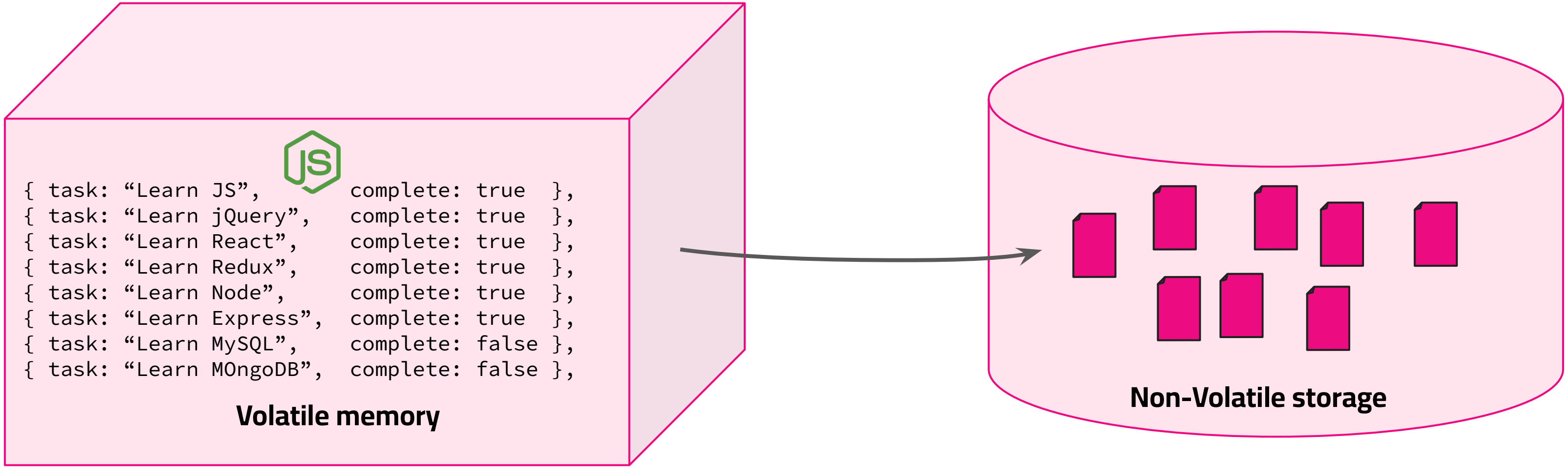
- Only rewrite portions of state on change
- Be capable of storing more than can be held in memory
- ✗ Efficiently access and address the stored information in multiple, flexible, useful ways
- ✗ Allow more than one process simultaneous access
- ✗ Avoid inconsistency of state due to changes
- Retain information through a power failure

Our persistance solution should

- ✖ Efficiently access and address the stored information in multiple, flexible, useful ways
- ✖ Allow more than one process simultaneous access
- ✖ Avoid inconsistency of state due to changes

What is holding our solution back ?

The data is disorganized!



Looking forward...

Can we find an alternative non-volatile storage mechanism?

- Better organizes the data
- Increases the efficiency of operations on the data

Databases

Database

- An **organized** collection of related data

What are some good ways to **organize** data ?

Data Structures!

Databases

Just like data structures, databases use similar primitives:

- Key-Value pairs
- Tables
- Plain text documents
- Graphs

DMS

Databases

Though different, all databases organize data such that it is predictable and highly functional

Data Table

id	user	text	room
412	fred	aw!	lobby
413	sunny	snap	park
414	allen	omg	park
415	pamela	yo!	lobby
416	fred	dude	park
417	pamela	huh?	park
418	sunny	umm	lobby
419	sunny	yo!	roof
420	fred	duh!	park
421	allen	snap	lobby

Text Document


```
{'id':412,'user':'fred','text':'aw!','room':'
lobby'}|{'id':413,'user':'sunny','text':'snap
','room':'park'}|{'id':414,'user':'allen','te
xt':'omg','room':'park'}|{'id':415,'user':'pa
mela','text':'yo!','room':'lobby'}|{'id':416,
'user':'fred','text':'dude','room':'park'}|{'
id':417,'user':'pamela','text':'huh?','room':
'park'}|{'id':418,'user':'sunny','text':'umm'
,'room':'lobby'}|{'id':419,'user':'sunny','te
xt':'yo!','room':'roof'}|{'id':420,'user':'fr
ed','text':'duh!','room':'park'}|{'id':421,'u
ser':'allen','text':'snap','room':'lobby'}
```

Databases

Though different, all databases organize data such that it is predictable and highly functional

Data Table

id	user	text	room
412	fred	aw!	lobby
413	sunny	snap	park
414	allen	omg	park
415	pamela	yo!	lobby
416	fred	dude	park
417	pamela	huh?	park
418	sunny	umm	lobby
419	sunny	yo!	roof
420	fred	duh!	park
421	allen	snap	lobby

Text Document

{'id':412,'user':'fred','text':'aw!','room':'
lobby'}|{'id':413,'user':'sunny','text':'snap
','room':'park'}|{'id':414,'user':'allen','te
xt':'omg','room':'park'}|{'id':415,'user':'pa
mela','text':'yo!','room':'lobby'}|{'id':416,
'user':'fred','text':'dude','room':'park'}|{'
id':417,'user':'pamela','text':'huh?','room':
'park'}|{'id':418,'user':'sunny','text':'umm'
,'room':'lobby'}|{
xt':'yo!','room':'
ed','text':'duh!',
ser':'allen','text

Note that data is usually stored in different and more efficient encodings like BSON

Databases

For now, let’s focus on **Relational Databases.**

Databases

In Relational Databases, data is stored in tables

- **Each table represents a data entity**
- The table columns represent attributes of the entity
- Each row in the table represents a single record or instance

Messages			
ID	User	Text	Room
412	fred	aw!	lobby
413	sunny	snap	park
414	allen	omg	park
415	pamela	yo!	lobby
416	fred	dude	park
417	pamela	huh	park
418	sunny	umm	lobby

Databases

In Relational Databases, data is stored in tables

- Each table represents a data entity
- **The table columns represent attributes of the entity**
- Each row in the table represents a single record or instance

Messages			
ID	**User**	**Text**	**Room**
412	fred	aw!	lobby
413	sunny	snap	park
414	allen	omg	park
415	pamela	yo!	lobby
416	fred	dude	park
417	pamela	huh	park
418	sunny	umm	lobby

Databases

In Relational Databases, data is stored in tables

- Each table represents a data entity
- The table columns represent attributes of the entity
- **Each row in the table represents a single record or instance**

Messages			
ID	User	Text	Room
412	fred	aw!	lobby
413	**sunny**	**snap**	**park**
414	allen	omg	park
415	pamela	yo!	lobby
416	fred	dude	park
417	pamela	huh	park
418	sunny	umm	lobby

Relational Databases

Relational Database tables have hard constraints

- The size and type of each column is fixed
- Data that does not fit these constraints will be truncated/ rejected

```
|id |user    |text    |room
|
|---|--------|--------|------
|
|412|fred---|aw!----|lobby-
|
|413|sunny--|snap---|park--
|
|414|allen--|omg----|park--
|
```

DMS

Databases

Relational Database tables have hard constraints

- The size and type of each column is fixed
- Data that does not fit these constraints will be truncated/ rejected

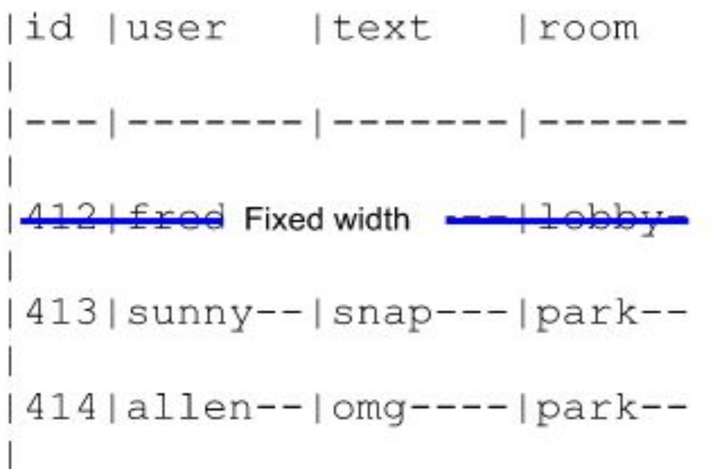
```
|id |user    |text    |room
|
|---|--------|--------|------
|
|412|fred---|aw!----|lobby-
|
|413|sunny--|snap---|park--
414|allen--|omg----|park--
|
```

Could the message ‘Greetings’ from user ‘Samantha’ in room ‘Kitchen’ be added?

Relational Databases

Tabular constraints make the data easier to work with.

- The size of each record is predefined and predictable
- This predictability improves the performance accessing information.



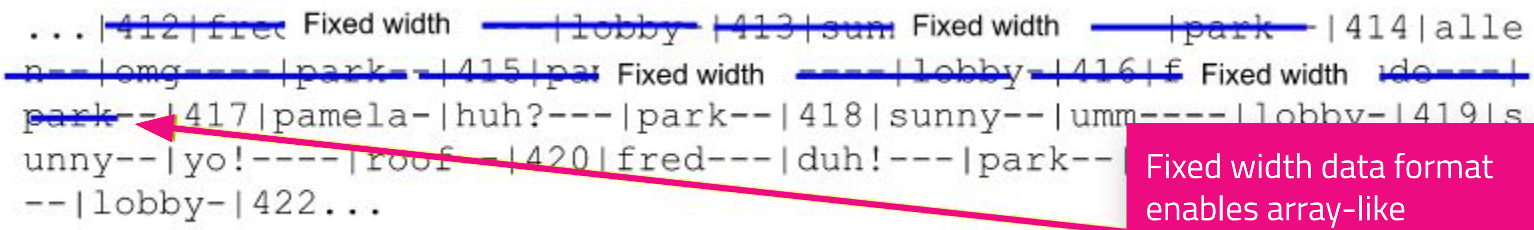
CodingLab Powered by RBK

DMS

Relational Databases

Tabular constraints make the data easier to work with.

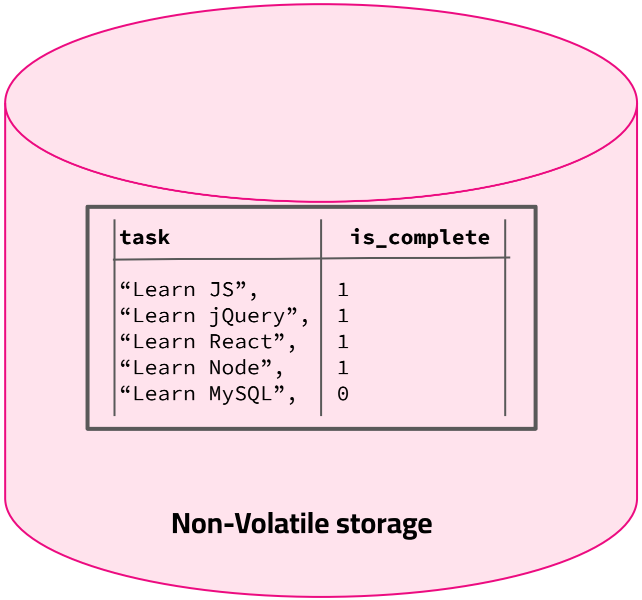
- The size of each record is predefined and predictable
- This predictability improves the performance of accessing information.
- SQL DBs uses array-like techniques to achieve **O(1)**



Fixed width data format enables array-like constant-time access to any index.

Relational Databases

We can store reliably organized, efficient data structures.



CodingLab Powered by RBK

DMS

Relational Databases

How do we create/ manage/ interact with this data ?

```
[
{ task: “Learn JS”,       complete: true  },
{ task: “Learn jQuery”,   complete: true  },
{ task: “Learn React”,    complete: true  },
{ task: “Learn Node”,     complete: true  },
{ task: “Learn MySQL”,    complete: false },
]
```

Volatile memory & scripts

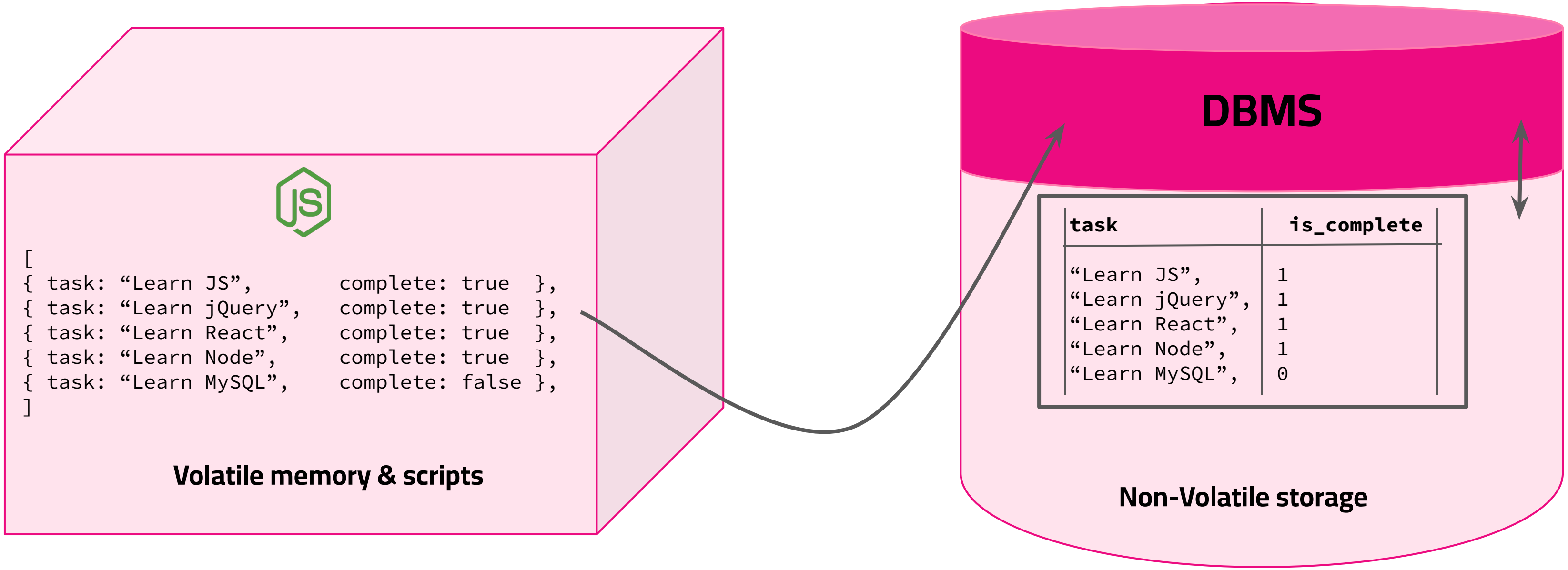
?

task	is_complete
“Learn JS”,	1
“Learn jQuery”,	1
“Learn React”,	1
“Learn Node”,	1
“Learn MySQL”,	0

Non-Volatile storage

Relational Databases

How do we create/ manage/ interact with this data ?



Database Management Systems

Database Management Systems

The software that optimizes and manages the storage and retrieval of data from databases.

- A database is just a collection of data
- A database management system enables you to interact with that database

*Often the term “ database” is used to describe both the data and the DBMS.

Database Management Systems

MySQL

SQLite

neo4j

PostgreSQL

mongoDB

Database Management Systems

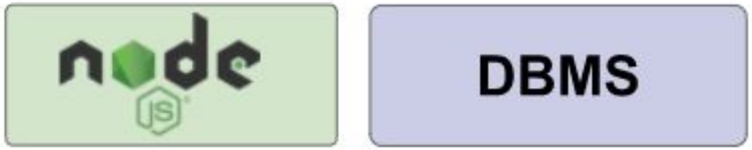
DBMS’s achieve similar goals through different means

- MySQL and SQLite both maintain relational databases
    - SQLite is server-less and self-contained, creating databases which exist entirely within the project files
    - MySQL runs on an isolated server allowing for many advanced features and greater performance

Database Management Systems

How does this fit into my application?

- A DBMS is completely independent process
- To interact with your database, you must connect and communicate with the DBMS process



Database Management Systems

How does this fit into my application?

- A DBMS is completely independent process
- To interact with your database, you must connect and communicate with the DBMS process

The DBMS becomes the third separate layer in our stack

Connecting to your DBMS

What is needed to communicate between two processes?

- A connection
  - We must be able to connect from some client
- A communication protocol
  - Probably not HTTP in this case
- Messages
  - In a language the recipient understands

Connecting to your DBMS

What is needed to communicate between two processes?

- **A connection**
  - **We must be able to connect from some client**
- A communication protocol
  - Probably not HTTP in this case
- Messages
  - In a language the recipient understands

Connecting to your DBMS

Creating a DBMS connection

- To connect to the DBMS from our Node.js server, we can install a driver module.
- Drivers expose a JS interface and transmit commands.
- Could we connect clients other than our server?

Connecting to your DBMS

What is needed to communicate between two processes?

- A connection
  - We must be able to connect from some client
- **A communication protocol**
  - **Probably not HTTP in this case**
- Messages
  - In a language the recipient understands

Thankfully, we don't typically worry about this since it's hidden behind a JS interface

Connecting to your DBMS

What is needed to communicate between two processes?

- A connection
  - We must be able to connect from some client
- A communication protocol
  - Probably not HTTP in this case
- **Messages**
  - **In a language the recipient understands**

Connecting to your DBMS

Database Management Systems don’t speak JavaScript

- JavaScript is great scripting language, but it isn’t great for data access and manipulation
- DBMS’s will use specialized, query languages
- These languages will focus on individual statements that can be used to query the database
- Languages often resemble the shape of the data within the databases they access

Query Languages

Database Management Systems don't speak JavaScript

- DBMS's will use specialized, query languages
- Languages often resemble the shape of the data within the databases they access
  - e.g Cypher is used for the graph database Neo4J

```
MATCH (user: {name:'John'} )-[:FRIEND]->(follower)
```

Query Languages

Database Management Systems don't speak JavaScript

- DBMS's will use specialized, query languages
- Languages often resemble the shape of the data within the databases they access
  - e.g Cypher is used for the graph database Neo4J
  - **SQL is used for relational databases**

```
SELECT * FROM friends f JOIN users u ON u.id = f.id;
```

Structured Query Language

```
mysql> show tables;
```

First, let's look at all the tables in our database with **show tables.**

DMS

```
mysql> show tables;

+---------------------+
|   Tables_in_school |
+---------------------+
| classes             |
| classes_students    |
| students            |
| teachers            |
+---------------------+
4 rows in set (0.00 sec)
```

First, let’s look at all the tables in our database with **show tables.**

```
mysql> describe classes;
```

We can use the **describe** command to look at the columns for a particular table.

DMS

```
mysql> describe classes;

+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | int(11)      | NO   | PRI | NULL    | auto_increment|
| name       | varchar(225) | NO   |     | NULL    |                |
| room       | varchar(255) | NO   |     | NULL    |                |
| teacher_id | int(11)      | YES  | MUL | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
4 rows in set (0.04 sec)
```

We can use the **describe** command to look at the columns for a particular table.

```
mysql> select * from classes;
```

Now let's look at the specific data in our classes table.

```
mysql> select * from classes;

+--------+---------+------+-------------+
|   id   | name    | romm |  techer_id  |
+--------+---------+------+-------------+
|   1    | CS 101  | 114  |      2      |
|   2    | CS 102  | 114  |      2      |
|   3    | CS 151  | 222  |      3      |
|   4    | CS 245  | 118  |      3      |
|   5    | CS 330  | 220  |      3      |
|   6    | PSY101  | 318  |      1      |
|   7    | PSY201  | 318  |      1     `|
+--------+---------+------+-------------+
7 rows in set (0.00 sec)
```

The asterisk character tells MySQL " I want to see data for all of the columns in this table".

```
mysql> select name, room from classes;
```

What if we only wanted to see the class name and its location? We can specify columns by referring to them by name in our query.

```
mysql> select name, room from classes;

+---------+-------+
| name    | romm  |
+---------+-------+
| CS 101 | 114   |
| CS 102 | 114   |
| CS 151 | 222   |
| CS 245 | 118   |
| CS 330 | 220   |
| PSY101 | 318   |
| PSY201 | 318   |
+---------+-------+
7 rows in set (0.01 sec)
```

What if we only wanted to see the class name and its location? We can specify columns by referring to them by name in our query.

```
mysql> select name, room from classes where teacher_id=2;
```

Let's start imagining some real world scenarios. Let's say I know a particular teacher's ID number and would like to see their class schedule.

CodingLab Powered by RBK

DMS

```
mysql> select name, room from classes where teacher_id=2;

+-----------+--------+
| name      | room   |
+-----------+--------+
| CS 101    | 114    |
| CS 102    | 114    |
+-----------+--------+
2 rows in set (0.01 sec)
```

This is easy because we specified a teacher_id column in our classes table. The where keyword in our query introduces a constraint: we only want to see records that satisfy a certain condition.

```
mysql> insert into classes (name,room, teacher_id)
       values (“CS101”, 25, 6);

Query OK, 1 row affected (0.02 sec)
```

Here’s an example of an INSERT statement which will add values to the specified fields (columns) in a new record (row)

Data Storage Goals, revisited

- Only rewrite portions of state on change
- Be capable of storing more than can be held in memory
- Efficiently access and address the stored information in multiple, flexible, useful ways
- Allow more than one process simultaneous access
- Avoid inconsistency of state due to changes
- Retain information through a power failure

Databases offer an ideal-er solution for data persistence and management

For any Full Stack application, utilizing a database for data storage is almost mandatory.

That’s it