

# Final Project

# Nộp bài

## ◆ Slides

- Slide 01: Tên đề tài, danh sách nhóm
- Slide 02: Link github source code (KHÔNG NỘP SOURCE CODE THÙ'A)
- Slide 03: Phân công công việc
- Các slide còn lại:
  - giao diện chức năng
  - mô tả giải pháp kỹ thuật (sử dụng express) để thực hiện các chức năng

## ◆ Chuẩn bị demo trên máy local

# Cấu trúc project

```
> JSON
> node_modules
✓ public
  > images
  ✓ javascripts
    JS mongodb.js
    JS users.js
  ✓ pages
    <> create.html
    <> delete.html
    <> home.html
    <> update.html
  > stylesheets
  JS index.js
  {} package-lock.json
  {} package.json
```

# Database

- ◆ **CSDL MongoDBCompass**
- ◆ id và image bắt buộc + 5 thuộc tính khác
- ◆ id được tạo tự động (xem đăng ký nhóm)
- ◆ collection bán cấu trúc
- ◆ Số lượng mẫu tin
  - Cá nhân: 20
  - Nhóm 2 sv: 30
  - Nhóm 3 sv: 40
  - Nhóm 4 sv: 50

# Trang chủ

- ◆ Truy cập địa **127.0.0.1:3000** để mở trang chủ
  - Trang chủ có **thông tin về website** và **menu**



# Chức năng (1)

## ◆ Chức năng **Read**

- Hiện thị danh sách **có sắp xếp**
  - Phân trang mỗi trang tối đa 10 mẫu tin
  - Xem **chi tiết** mẫu tin
- Tìm kiếm theo danh mục có sẵn
- Tìm kiếm gần đúng theo thuộc tính kiểu string
- Tìm kiếm trong khoảng theo thuộc tính kiểu number
- Tìm kiếm theo nhiều thuộc tính: OR, AND

# Chức năng (2)

- ◆ Chức năng **Create**: menu gồm 3 item
  - Create One (bắt buộc)
  - Create Many (nâng cao): Import file json
- ◆ Chức năng **Update**
  - Update One
  - Update Many
- ◆ Chức năng **Delete**
  - Delete One
  - Delete Many

# Giao tiếp giữa client và server

- ◆ Sử dụng **fetch()** để gửi yêu cầu đến server bằng phương thức get, post, put, delete



# Ôn bài - Parameters

## ◆ **req.body.name**

- truy cập dữ liệu gửi trong yêu cầu GET, POST, PUT và DELETE (json)

## ◆ **req.query.name** (query parameters)

- truy cập dữ liệu gửi trong yêu cầu GET
- <http://127.0.0.1:3000/read?name=Thanh&email=thanh%40abc>

## ◆ **req.params.name**

- Tham số động (dynamic parameters)
- `app.get('/users/:name', (req, res) => {...`

# Ôn bài - \$regex - Regular Expression

- ◆ { name: { \$regex: '^Nguyen' } } //start
- ◆ { email: { \$regex: '@gmail.com.vn\$' } } //end
- ◆ { name: { \$regex: 'Nguyen' , \$options: 'i' } }
- ◆ { city: { \$regex: 'Hue|Saigon|Hanoi' } }
- ◆ { code: { \$regex: '^[a-z]+\$' } }
- ◆ { date: { \$regex: '^[0-9]{4}-[0-9]{2}-[0-9]{2}\$' } } //yyyy-mm-dd

# \$regex - Examples

```
const query = {  
  age: { $gt: 18 },  
  gender: 'male'  
};
```

```
const query = {  
  age: { $gt: 18 },  
  gender: 'male',  
  name: { $regex: "thanh", $options: 'i'}  
};
```

```
const query = {  
  $or: [  
    { age: { $gt: 18 } },  
    { gender: 'male' }  
  ]  
};
```

# Xuất danh sách có sắp xếp

- ◆ Sắp xếp theo id tăng dần

```
db.users.find().sort({id: 1})
```

- ◆ Sắp xếp theo id tăng dần và name giảm dần

```
db.users.find().sort({id: 1, name: -1})
```

- ◆ Tìm id lớn nhất

```
db.users.find().sort({id: -1}).limit(1)
```



# Tạo ID tự động (1)





```
6  // Append item to array
7  let newUser = { id: 'USER003', name: 'Alice Johnson', email: 'alice@example.com' };
8  users.push(newUser);
9
10 // Generate newId
11 let maxId = 0;
12 for (let i = 0; i < users.length; i++) {
13     // Extracts characters from index 4 to the end
14     const userId = parseInt(users[i].id.substring(4));
15     if (userId > maxId) {
16         maxId = userId;
17     }
18 }
19 const newId = 'USER' + String(maxId + 1).padStart(3, '0');
20 // Add new user to array
21 users.push({ id: newId, name: 'New User', email: 'NewUser@example.com' });
```



# Tạo ID tự động (2)

```
151 async function getNextIdNumber(client, dbName, collectionName) {
152     const db = client.db(dbName);
153     const collection = db.collection(collectionName);
154
155     const lastBook = await collection.find().sort({id: -1}).limit(1).toArray();
156
157     if (lastBook.length === 0) {
158         return 1;
159     } else {
160         const lastId = lastBook[0].id;
161         const nextIdNumber = parseInt(lastId.replace("BOOK", "")) + 1;
162         return nextIdNumber;
163     }
164 }
```

# Tìm kiếm theo danh mục

Giá	Loại điện thoại	Nhu cầu
<div>Dưới 2 triệu</div> <div>Từ 2 - 4 triệu</div> <div><b>Từ 4 - 7 triệu</b></div> <div>Từ 7 - 13 triệu</div> <div>Từ 13 - 20 triệu</div> <div>Trên 20 triệu</div> <div> Hoặc chọn mức giá phù hợp với bạn ▼</div>	<div> Android</div> <div> iPhone (iOS)</div> <div> Điện thoại phổ thông</div>	<div>Chơi game / Cấu hình cao</div> <div>Pin khủng trên 5000 mAh</div> <div><b>Chụp ảnh, quay phim</b></div> <div>Livestream</div> <div>Mỏng nhẹ</div>

# Tìm kiếm theo điều kiện (1)

Name:

Email:

Year of birth:

Display 10 latest documents

# Tìm kiếm theo điều kiện (2)

Name:

Email:

Year of birth:

```
<> read.html > ...
1      <form action="/read" method="GET">
2          <label for="name">Name:</label>
3          <input type="text" id="name" name="name"><br><br>
4
5          <label for="email">Email:</label>
6          <input type="email" id="email" name="email"><br><br>
7
8          <label for="birthYear">Year of birth:</label>
9          <input type="number" id="birthYearFr" name="birthYearFr" placeholder="from/gte">
10         <input type="number" id="birthYearTo" name="birthYearTo" placeholder="to/lte">
11
12         <button type="submit">Search</button>
13     </form>
```

# Tìm kiếm theo điều kiện (3)

127.0.0.1:3000/read?name=Thanh&email=.com.vn&birthYearFr=1990&birthYearTo=2000

Name:

Email:

Year of birth:

req.query.name



# Tìm kiếm theo điều kiện (4)

```
16 app.get('/read', (req, res) => {
17   const name = req.query.name;
18   const email = req.query.email;
19   const birthYearFr = req.query.birthYearFr;
20   const birthYearTo = req.query.birthYearTo;
21
22   const query = {};
23
24   if (name) {
25     query.name = { $regex: name, $options: 'i' };
26   }
27
28   if (email) {
29     query.email = { $regex: email, $options: 'i' };
30   }
31
32   if (birthYearFr && birthYearTo) {
33     query.birthYear = { $gte: parseInt(birthYearFr), $lte: parseInt(birthYearTo) };
34   } else if (birthYearFr && !birthYearTo) {
35     query.birthYear = { $gte: parseInt(birthYearFr) };
36   } else if (!birthYearFr && birthYearTo) {
37     query.birthYear = { $lte: parseInt(birthYearTo) };
38   }
39
40   console.log(query);
41   res.send(query)
42 });
```

# localStorage

*// Lưu dữ liệu*

```
localStorage.setItem('key', 'value');
```

*// Lấy dữ liệu*

```
let value = localStorage.getItem('key');
```

*// Xóa dữ liệu theo key*

```
localStorage.removeItem('key');
```

*// Xóa toàn bộ dữ liệu*

```
localStorage.clear();
```

Tính năng	localStorage	sessionStorage	Cookies
Thời gian lưu trữ	Vĩnh viễn (trừ khi xóa)	Chỉ khi tab mở	Do bạn đặt (expires)
Dung lượng	~5–10 MB	~5 MB	~4 KB
Gửi kèm HTTP Request	✗ Không	✗ Không	✓ Có

# Create One (1)

```

1  <body onload="setFocusName()">
2    <form id="createForm">
3      <label for="name">Name:</label>
4      <input type="text" id="name" name="name" required><br><br>
5
6      <label for="email">Email:</label>
7      <input type="email" id="email" name="email" required><br><br>
8
9      <button type="button" onclick="addUser()">Add</button>
10     <button type="button" onclick="submitUsers()">Submit</button>
11   </form>
12
13   <div id="message"></div>
14
15 </body>
16 <script>
17   var users = [];
18   > function addUser() { ...
26   > function submitUsers() { ...
42   > function outputMessage(message, color) { ...
47   > function resetForm() { ...
53   > function setFocusName(){ ...
56 </script>

```

Name:

Email:

The user [Nguyen Thi Thanh, ntthanh@gmail.com] added

Name:

Email:

The user [Thanh Nguyen, thanh.nguyen@gmail.com] added

# Create One (2)

```
18 function addUser() {  
19     var name = document.getElementById("name").value;  
20     var email = document.getElementById("email").value;  
21     var user = { name: name, email: email };  
22  
23     users.push(user);  
24     outputMessage(`The user [${name}, ${email}] added`, 'red');  
25 }  
26 > function submitUsers() { ...  
42 function outputMessage(message, color) {  
43     var messageElement = document.getElementById('message');  
44     messageElement.textContent = message;  
45     messageElement.style.color = color;  
46 }
```

Name:

Email:

The user [Nguyen Thi Thanh, ntthanh@gmail.com] added



# Create One (3)

```

17  var users = [];
18  > function addUser() { ...
26  function submitUsers() {
27      fetch('/create', {
28          method: 'POST',
29          headers: {'Content-Type': 'application/json'},
30          body: JSON.stringify(users)
31      })
32      .then(response => response.json())
33      .then(data => {
34          console.log('Response from server:', data);
35          alert(data.message);
36          resetForm();
37      })
38      .catch(err => {
39          console.error('Error:', err);
40      });
41  }

```

127.0.0.1:3000 says

2 users created successfully!

OK

```

47  function resetForm() {
48      document.getElementById('createForm').reset();
49      document.getElementById('message').textContent = '';
50      setFocusName();
51      users = [];
52  }
53  function setFocusName(){
54      document.getElementById('name').focus();
55  }

```



# Create One (4)

JS index.js > ...

```
1  const express = require('express')
2  const app = express()
3  const port = 3000
4
5  const bodyParser = require('body-parser');
6  app.use(bodyParser.urlencoded({ extended: false }));
7  app.use(bodyParser.json());
8
9  > app.get('/', (req, res) => { ...
11 });
12 app.post('/create', (req, res) => {
13   const users = req.body;
14   console.log(users);
15   // insertMany() to MongoDB here
16   const message = `${users.length} users created successfully!`
17   res.json({ message: message });
18 });
19
20 app.listen(port, () => console.log(`The app listening on port ${port}!`))
```

# Create One (5)

```
19 app.post('/create', async(req, res) => {
20   try {
21     await client.connect();
22     console.log('Connect to MongoDB successfully!!!');
23
24     const users = req.body;
25     console.log(users);
26
27     console.log('start inserting to db...');
28     usersCollection = await client.db(dbName).collection(collectionName);
29     await usersCollection.insertMany(users)
30
31     const content = `${users.length} users created successfully!`
32     res.json({ message: content }); // response to fetch()
33   } catch (err) {
34     console.error('Error:', err);
35   } finally {
36     if (client) {
37       // Close the connection to MongoDB
38       await client.close();
39       console.log('MongoDB connection closed');
40     }
41   }
42 });
```

# Update

User ID:

User Name:

*findOne()*

ID:

Name:

Email:

*find()*

ID	Name	Email	Actions
1004689	Nguyen Thi Thanh	ntthanh@example.com	<a href="#">Edit</a>
2310088	Thanh Thi Nguyen	thanhnguyen@example.com	<a href="#">Edit</a>

# Update One (1)

User ID:

User Name:

```
<form action="/read/for/update" method="GET">
```

```
const querystring = require('querystring');
```

```
25 // READ for update
26 app.get('/read/for/update', async(req, res) => {
27   const id = req.query.userId;
28   const query = {id: { $regex: id, $options: 'i' }}
29   try {
30     await client.connect();
31     console.log('Connect to MongoDB successfully!!!');
32     usersCollection = await client.db(dbName).collection(collectionName);
33     editUser = await usersCollection.findOne(query)
34
35     const queryParams = querystring.stringify(editUser);
36     res.redirect(`/update.html?${queryParams}`);
```



# Update One (2)

ID:  `<input type="text" id="id" name="id" readonly><br><br>`

Name:

Email:

`<button type="button" onclick="updateUser()">Update</button>`



# Update One (3)

```
19  <script>
20    const queryParams = new URLSearchParams(window.location.search);
21    const oldUser = Object.fromEntries(queryParams.entries());
22    let newUser = oldUser
23
24    document.getElementById('id').value = oldUser.id;
25    document.getElementById('name').value = oldUser.name;
26    document.getElementById('email').value = oldUser.email;
27
28    function updateUser() {
29      newUser.name = document.getElementById('name').value
30      newUser.email = document.getElementById('email').value
31
32      // check if no values updated, display message and return
33
34      fetch('/update', {
35        method: 'PUT',
36        headers: {'Content-Type': 'application/json'},
37        body: JSON.stringify(newUser)
38      })
```

# Update One (4)

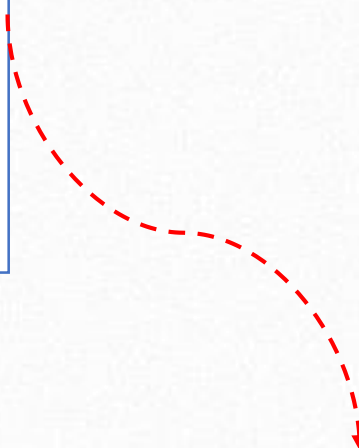
```
74 app.put('/update', async(req, res) => {
75   try {
76     await client.connect();
77     console.log('Connect to MongoDB successfully!!!');
78
79     const newUser = req.body;
80     const id = newUser.id
81
82     usersCollection = await client.db(dbName).collection(collectionName);
83
84     await usersCollection.updateOne({id: id}, {$set: {name: newUser.name, email: newUser.email}})
85
86     const content = `User id = ${newUser.id} updated successfully!`
87     res.json({ message: content }); // response to fetch()
```

```
fetch('/update', {
  method: 'PUT',
  headers: {'Content-Type': 'application/json'},
  body: JSON.stringify(newUser)
})
.then(response => response.json())
.then(data => {
  alert(data.message);
  // resetForm();
})
.catch(err => {
  alert('Can not update: ' + err);
});
```

# Delete (1)

User ID:

User Name:



ID	Name	Email	Actions
1004689	Nguyen Thi Thanh	ntthanh@example.com	<a href="#">Delete</a>
2310088	Thanh Thi Nguyen	thanhnnguyen@example.com	<a href="#">Delete</a>
			<a href="#">Delete All</a>

# Delete (2)

```
20 app.get('/delete', (req, res) => {
21     var users = [
22         { id: 1, name: "John Doe", email: "john@example.com" },
23         { id: 2, name: "Jane Smith", email: "jane@example.com" },
24         { id: 3, name: "Bob Johnson", email: "bob@example.com" }
25     ];
26     // userData: get from client side
27     const queryString = new URLSearchParams({ userData: JSON.stringify(users) }).toString();
28     res.redirect('/delete.html?' + queryString);
29 });
```

```
<script>
    // get users data from server
    const queryString = window.location.search;
    const urlParams = new URLSearchParams(queryString);
    const users = JSON.parse(urlParams.get('userData'));
```



# Delete (3)

```
<table>
  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Email</th>
      <th>Action</th>
    </tr>
  </thead>
  <tbody id="userTable">
    <!-- Users data displayed here -->
  </tbody>
</table>
```

```
<script>
  // get users data from server
  const queryString = window.location.search;
  const urlParams = new URLSearchParams(queryString);
  const users = JSON.parse(urlParams.get('usersData'));

  var userTable = document.getElementById("userTable");

  users.forEach(function(user) {
    var row = document.createElement("tr");
    var idCell = document.createElement("td");
    idCell.textContent = user.id;
    row.appendChild(idCell);
    // name and email

    var actionCell = document.createElement("td");
    var deleteLink = document.createElement("a");
    deleteLink.href = "#";
    deleteLink.textContent = "Delete";
    deleteLink.addEventListener("click", function() {
      if (confirm("Delete " + user.name + "?")) {
        fetch("delete/user/" + user.id, {
          method: "DELETE"
        })
      }
    })
  })
}
```

# Bài tập tại lớp (BT04)

## ◆ Chức năng Create

- Hiện thực chức năng Create, id tạo tự động
- MongoDB Compass: Collection đã đăng ký file excel

## ◆ Start server: trang create.html hiển thị

## ◆ Nộp bài:

- create.html: giao diện form nhập
- server.js: gồm các api xử lý create

ID (tự động):	<input type="text"/>
Phân loại:	--Chọn phân loại-- ▾
Tên sách:	<input type="text"/>
	<input type="button" value="Lưu"/>

ID (tự động):	BOOKTLLS0002
Phân loại:	Lịch sử ▾
Tên sách:	<input type="text"/>
	<input type="button" value="Lưu"/>