What did i learned yesterday:

- We did recall for nodejs and mongoose and express logics
- We learned about node project structure, and to organize the folders and the files.
- Card.js - this file is creating a card structure schema, model(). We export it as a Card.
- Validators file - is like a template for using inside the Card schema types and also in other schemas when we need., NUMBER, URL, EMAIL, DEFAULT_VALIDATOR.
- In the app.js file we call mongoose, express, we connect to mongodb database, we get(), we post() ,we listen()
- We have file for createCard and updateCard, we call the createCard on app.js
- The goal of this project is to handle CRUS operation - get(), put(), post(), delete()

The code i did worm the memory:

```
const NUMBER = {
    type: Number,
    min: 0,
    require: true,
}

const URL = {
    type: string,
    RegExp:
/^(https?:\/\/)(localhost|((?:[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?\.)+[a-z
A-Z]{2,}))(?::\d{2,5})?(?:\/[^\s]*)?$/,
    require: true,
    // unique
}

const EMAIL = {
    type: string,
    RegExp: /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$/,
    require: true,
}

const DEFAULT_VALIDATOR = {
    type: string,
    minLength: 2,
    maxLength: 288,
}

module.exports = {NUMBER, URL, EMAIL, DEFAULT_VALIDATOR}
```

Issues in the code:

Wrong	Right	Why
`string`	`String`	JavaScript built-in constructor uses capital S
`RegExp:`	`match:`	Mongoose uses `match` for regex validation
`require:`	`required:`	Typo - the property name has a 'd'
missing `trim`	`trim: true`	Removes spaces from start/end of strings
missing `lowercase`	`lowercase: true`	Stores emails/urls in lowercase (consistency)

With commented Fixes:

```
const NUMBER = {
    type: Number,
    min: 0,
    require: true, // ❌ typo: should be "required (with 'd')"
}

const URL = {
    type: string, // ❌ should be "String" with capital "S"
    RegExp:
/^(https?:\/\/)(localhost|((?:[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?\.)+[a-zA-Z]{2,})
)(?::\d{2,5})?(?:\/[^\s]*)?$/, // ❌ wrong property name: should be 'match'
    require: true, // ❌ typo: "required"
    // unique
}

const EMAIL = {
    type: string, // ❌ should be "String" with capital "S"
    RegExp: /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$/, // ❌ wrong property name:
should be 'match'
    require: true, //❌ typo: 'required'
    // unique: true <-- add this!
}

const DEFAULT_VALIDATOR = {
    type: string, // ❌ should be with capital "S"
    minLength: 2,
    maxLength: 288,
    // ❌ missing: required, trim, lowercase
}

module.exports = {NUMBER, URL, EMAIL, DEFAULT_VALIDATOR}
```

Installing node and libraries:

Check if node is installed:
node --version
npm --version

Initialize node.js project:
npm init -y

Creates package.json
-y means ‘yes to all default options’
result : you now have a package.json.

Install libraries:
npm install express mongoose

Create .gitignore

Issues in the code:

validators.js:

```
// ❌ What you wrote:
lowerCase: true,

// ✅ Correct (all lowercase):
lowercase: true,
```

Card.js - createdAt - logic error:

Code	What Happens
`new Date().toISOString()`	Runs ONCE when server starts. All cards get same date.
`Date.now`	Runs EACH TIME a card is created. Each card gets its own date.

Notice: Sate.now without () → we pass the function itself, not its result.

Card.js - userId is empty string:

```
// ❌ What you wrote:
userId: '',

// ✅ Correct:
userId: {
    type: mongoose.Schema.Types.ObjectId,
    required: true,
},
```

Why ObjectId:
This field stores which user created the card. MongoDB uses ObjectId as unique identifiers. When you create a user later, each user gets an _id like 507f1f77bcf86cd799439011. That's what gets stored in userId.

cardsSvc.js:

First try:

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

const {Card} = require('../models/Card')

const CreateCard = async (card) => {
    try{
        const newCard = new Card(card);
    }
    catch(err){
        console.log(err.message);
    }
}
```

```
// ❌ don't need mongoose here
// const mongoose = require('mongoose');
// this file only uses the Card model, not mongoose directly.

// ❌ what you wrote
// const {Card} = require('../models/Card');

// correct:
const Card = require('../models/Card');

const CreateCard = async (card) => {
    try{
        const newCard = new Card(card);
```

```
        // ❌ created but not saved to database;
        newCard = await newCard.save();
        return newCard;
        // ❌ nothing returned!
    }
    catch(err){
        console.log(err.message);
    }
}
```

```
const mongoose = require('mongoose');  // ← Mongoose is used HERE

const CardSchema = new mongoose.Schema({...});

const Card = mongoose.model('Card', CardSchema);  // ← Creates model with ALL
methods

module.exports = Card;  // ← Exports the model (with methods included!)
```

When you call `mongoose.model('Card', CardSchema)`, Mongoose returns an object that **already includes** all these methods:
- `.find()`
- `.findById()`
- `.findByIdAndUpdate()`
- `.save()`
- etc.

Visual Explanation

```
Card.js                                  cardsService.js
─────────────────────────────            ──────────────────────────────

const mongoose = require('mongoose')

const Card = mongoose.model(...)         const Card = require('./Card')
     │                                        │
     │                                        │
     ▼                                        ▼
┌──────────────────────────┐                 Card.find()     ✅ Works!
│  Card Model              │            Card.findById() ✅ Works!
│  ├── .find()             │             Card.save()     ✅ Works!
│  ├── .findById()         │
```

```
│   ├── .findByIdAndUpdate()│          No need to require mongoose
│   ├── .save()             │          because Card already has
│   └── ... more methods    │          all the methods!
└──────────────────────────────┘
```

Why do we use return here?
Without return?
Without return, the data is trapped inside the function

```
// ❌ WITHOUT return:
const getCards = async () => {
    let cards = await Card.find();
    // cards exists here... but dies when function ends
}

let result = await getCards();
console.log(result);  // undefined 😢
```

app.js:

First try:

```
const mongoose = require('');
const express = require('express');

const app = express()
app.use(express.json());

app.get('/cards', (req,res) => {
    res.send('connectend')
});

const connectToMongoDB = async () => {
    try{
        await mongoose.connect('mongodb://127.0.0.1:27017/newCardsData')
    }
    catch(err){
        console.log();

    }
}

app.listen('/cards', (req, res) => {

})
```

Issues:

Issue 1 → Missing 'mongoose' string:

```
// ❌ What you wrote:
const mongoose = require('');

// ✅ Correct:
const mongoose = require('mongoose');
```

Issue 2 → Missing service imports:

```
// ❌ Missing:
// You need to import your service functions!

// ✅ Add this:
const { createNewCard, getCards } = require('./service/cardsService');
```

Issue 3 → app.listen() syntax wrong:

```
// ❌ What you wrote:
app.listen('/cards', (req, res) => {

})

// ✅ Correct:
app.listen(PORT, () => {
    console.log('Server running on port ' + PORT);
    connectToMongoDB();
})
```

Why?

Parameter	What it should be
First	PORT number (like 8181)
Second	Callback function (no req, res - just () => {})

Issue 4 → Routes need to use service functions:

```
// ❌ What you wrote:
app.get('/cards', (req, res) => {
    res.send('connectend')  // Just sends text, doesn't get real cards!
});

// ✅ Correct:
```

```
app.get('/cards', async (req, res) => {
    try {
        let cards = await getCards();  // Get cards from database
        res.send(cards);
    } catch (err) {
        res.status(400).send(err.message);
    }
});
```

Issue 5: missing post route

```
// ✅ Add this:
app.post('/cards', async (req, res) => {
    try {
        let card = await createNewCard(req.body);
        res.send(card);
    } catch (err) {
        res.status(400).send(err.message);
    }
});
```

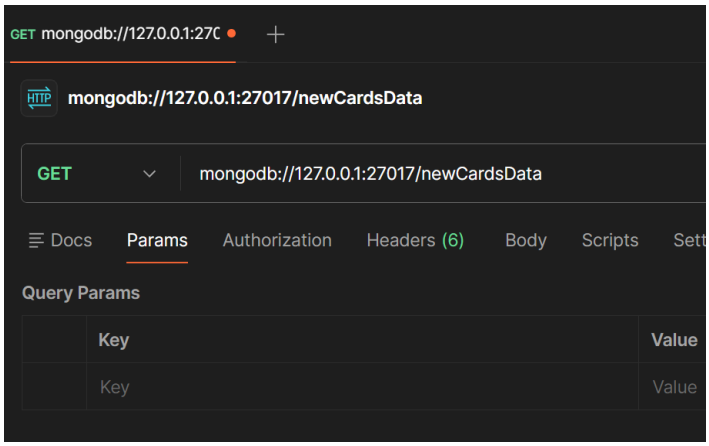
Issue 6 → empty console.log():

```
// ❌ What you wrote:
catch(err) {
    console.log();  // Logs nothing!
}

// ✅ Correct:
catch(err) {
    console.log('MongoDB connection error:', err.message);
}
```

Maintenance - תחזוקה
approximately - בערך

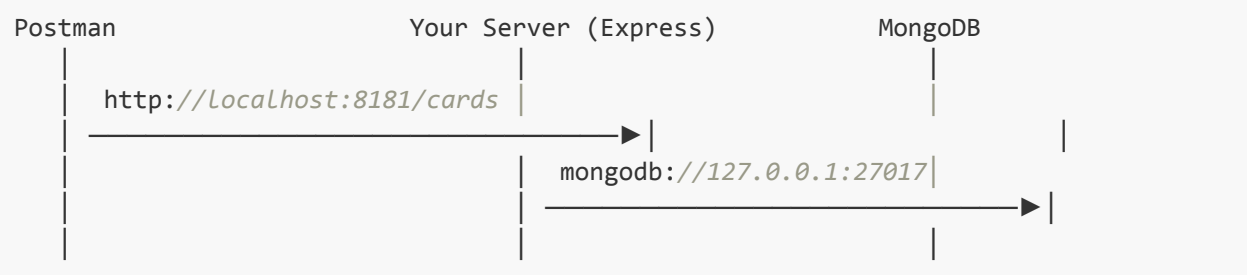
Wrong:



You're trying to connect a postman to mongoDB. That's not how it works!

Postman vs mongoDB

Tool	URL	Purpose
Postman	http://localhost:8181/cards	Talk to YOUR Express server
mongoose.connect()	mongodb://127.0.0.1:27017/...	Server talks to database



post man talks to express. Express talk to mongoDB. Postman never talks directly to mongoDB!

Need to learn this code:

```
const express = require('express');
const mongoose = require('mongoose');
const { createCard, getCards, updateCard, deleteCard } =
require('./cards/models/cardsAccessDataService');
const Card = require('./cards/models/mongodb/Card');

const app = express();
const PORT = 8181;

app.use(express.json());

const connectToDB = async () => {
  try {
    await mongoose.connect('mongodb://127.0.0.1:27017/cardsServer');
    console.log('Connected to MongoDB');
  }
  catch(err) {
    console.log('MongoDB connection error:', err.message);
  }
};

// Test route
app.get('/', (req, res) => {
  res.send('Server is running!');
});

// GET all cards
app.get('/cards', async (req, res) => {
  try {
    let cards = await getCards();
    res.send(cards);
  }
  catch(err) {
    res.status(400).send(err.message);
  }
});

// GET single card by ID
app.get('/cards/:id', async (req, res) => {
  try {
    let card = await Card.findById(req.params.id);
    if (!card) {
      return res.status(404).send('Card not found');
    }
    res.send(card);
  }
  catch(err) {
    res.status(400).send(err.message);
  }
});

// POST new card
app.post('/cards', async (req, res) => {
  try {
    let card = await createCard(req.body);
    res.send(card);
  }
```

```
  catch(err) {
    res.status(400).send(err.message);
  }
});

// PUT update card
app.put('/cards/:id', async (req, res) => {
  try {
    let card = await updateCard(req.params.id, req.body);
    if (!card) {
      return res.status(404).send('Card not found');
    }
    res.send(card);
  }
  catch(err) {
    res.status(400).send(err.message);
  }
});

// DELETE card
app.delete('/cards/:id', async (req, res) => {
  try {
    let card = await deleteCard(req.params.id);
    if (!card) {
      return res.status(404).send('Card not found');
    }
    res.send(card);
  }
  catch(err) {
    res.status(400).send(err.message);
  }
});

// PATCH like/unlike card
app.patch('/cards/:id', async (req, res) => {
  try {
    let card = await likeCard(req.params.id, req.body.userId);
    res.send(card);
  }
  catch(err) {
    res.status(400).send(err.message);
  }
});

// Start server
app.listen(PORT, () => {
  console.log('Server running on port ' + PORT);
  connectToDB();
});
```

New topics to learn and recall:

```
// GET single card by ID
app.get('/cards/:id', async (req, res) => {
  try {
    let card = await Card.findById(req.params.id);
    if (!card) {
      return res.status(404).send('Card not found');
    }
    res.send(card);
  }
  catch(err) {
    res.status(400).send(err.message);
  }
});

// PUT update card
app.put('/cards/:id', async (req, res) => {
  try {
    let card = await updateCard(req.params.id, req.body);
    if (!card) {
      return res.status(404).send('Card not found');
    }
    res.send(card);
  }
  catch(err) {
    res.status(400).send(err.message);
  }
});

// DELETE card
app.delete('/cards/:id', async (req, res) => {
  try {
    let card = await deleteCard(req.params.id);
    if (!card) {
      return res.status(404).send('Card not found');
    }
    res.send(card);
```

```
  }
  catch(err) {
    res.status(400).send(err.message);
  }
});
```

card Access file:

```
const Card = require("./mongodb/Card");

const createCard = async (newCard) => {
  try {
    let card = new Card(newCard);
    card = await card.save();
    return card;
  } catch (error) {
    throw new Error("Mongoose " + error.message);
  }
};

const getCards = async () => {
  try {
    let cards = await Card.find();
    return cards;
  } catch (error) {
    throw new Error("Mongoose " + error.message);
  }
};

const updateCard = async (cardId, newCard) => {
  try {
    const card = await Card.findByIdAndUpdate(cardId, newCard, { new: true });
    return card;
  } catch (error) {
    throw new Error("Mongoose " + error.message);
  }
};

const deleteCard = async (cardId) => {
  try {
    const card = await Card.findByIdAndDelete(cardId);
    return card;
  } catch (error) {
    throw new Error("Mongoose " + error.message);
  }
};
```

```
// NEW: Like/Unlike toggle
const likeCard = async (cardId, userId) => {
  try {
    let card = await Card.findById(cardId);
    if (!card) {
      throw new Error("A card with this ID cannot be found in the database");
    }

    if (card.likes.includes(userId)) {
      // User already liked → Remove the like
      let newLikesArray = card.likes.filter((id) => id != userId);
      card.likes = newLikesArray;
    } else {
      // User hasn't liked → Add the like
      card.likes.push(userId);
    }

    await card.save();
    return card;
  } catch (error) {
    throw new Error("Mongoose " + error.message);
  }
};


module.exports = { createCard, getCards, updateCard, deleteCard , likeCard};
```

Learn - new topics (syntax and logic):

- patch()
- put()
- post()
- delete()
- Function: updateCard()
- Function: deleteCard()
- Function: likeCard()

[gitHub Project→]