Assignment8

Using mongoose defines the following collections (Sticky Notes) (10 Grades):

Users (0.5 Grade)	Notes (0.5 Grade)
• **name** (String, required) • **email** (String, Unique, required) • **Password** (String, required) • **Phone** (String, required) • **age** (Number) (Must be between 18 and 60)	• **title** (String, required) • **content** (String, required) • **userId** (ref to Users, required) • **createdAt** (Timestamp) • **updatedAt** (Timestamp)
Must Implemented: 1. Add a **custom validator** to the **"title"** field that ensure the title is **not entirely uppercase**. For example ("FIRST NOTE") ("First Note"). **(0.5 Grade)**	

APIs (Ensure you apply the folder structure we discussed)

A- User APIs (2.5 Grades)

1. **Signup (make sure that the email does not exist before) (Don't forget to hash the password and encrypt the phone). (0.5 Grade)**
   - **URL:** POST /users/signup

```
{
  "message": "Email already exists."
}
```

```
{
  "message": "User added successfully."
}
```

2. **Create an API for authenticating users (Login) and return a JSON Web Token (JWT) that contains the userId and will expire after "1 hour". (Get the email and the password from the body). (0.5 Grade)** • **URL:** POST /users/login

```
{
  "message": "Login successful",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0ODcx
}
```

```
{
  "message": "Invalid email or password"
}
```

3. **Update logged-in user information (Except Password). (If user want to update the email, check the new email doesn't exist before. (Get the id for the logged-in user (userId) from the token not the body) (send the token in the headers) (0.5 Grade)**
   - **URL:** PATCH /users

Input

```
{
  "name": "updated",
  "email": "updated@email.com",
  "age": 30
}
```

Output

```
{
  "message": "User updated",
}
```

```
{
  "message": "User not found"
}
```

```
{
  "message": "Email already exists.
}
```

4. **Delete logged-in user. (Get the id for the logged-in user (userId) from the token not the body) (send the token in the headers) (0.5 Grade)**
   - **URL:** DELETE /users

```
{
  "message": "User deleted"
}
```

```
{
  "message": "User not found"
}
```

5. **Get logged-in user data by his ID. (Get the id for the logged-in user (userId) from the token not the body) (send the token in the headers) (0.5 Grade)**
   - **URL:** GET /users

```
{
  "_id": "64d91c42d8979e1f30a12345",
  "name": "User1",
  "email": "User1@email.com",
  "password": "$2b$10$OqCjH.AxhjF2gYZGQIHybO8cm3fr0ZNRb/xj9fJPTJS5xEUHvfH9u",
  "phone": "U2FsdGVkX19kIeLZa1txT5N7XzQ21tNlFSKkJmInAfA=",
  "age": 25
}
```

Assignment8

B- Note APIs (6 Grades)

1. **Create a Single Note (Get the id for the logged-in user (userId) from the token not the body) (send the token in the headers) (0.5 Grade)**
   - **URL:** POST /notes

```
{
  "title": "Note1",
  "content": "this content for note1"
}
```

```
{
  "message": "Note created"
}
```

2. **Update a single Note by its id and return the updated note. (Only the owner of the note can make this operation) (Get the id for the logged-in user (userId) from the token not the body) (0.5 Grade)**
   - **URL:** PATCH /notes/:noteId => /notes/64d91c42d8979e1f30a12346

```
{
  "title": "updated",
  "content": "update content"
}
```

```
"message": "updated",
"note": {
  "id": "64d91c42d8979e1f30a12346",
  "title": "updated",
  "content": "update content",
  "userId": "64d91c42d8979e1f30a12345",
  "createdAt": "2024-12-14T12:00:00.000Z",
  "updatedAt": "2024-12-14T12:30:00.000Z"
}
```

```
{
  "message": "You are not the owner"
}
```

```
{
  "message": "Note not found"
}
```

3. **Replace the entire note document with the new data provided in the request body. (Only the owner of the note can make this operation) (Get the id for the logged-in user (userId) from the token not the body) (0.5 Grade)** • **URL**: PUT /notes/replace/:noteId=> /notes/replace/64d91c42d8979e1f30a12348

```
{
  "title": "updated",
  "content": "update content",
  "userId": "64d91c42d8979e1f30a12839"
}
```

```
{
  "id": "64d91c42d8979e1f30a12348",
  "title": "updated",
  "content": "update content",
  "userId": "64d91c42d8979e1f30a12839",
  "createdAt": "2024-12-13T16:00:00.000Z",
  "updatedAt": "2024-12-13T16:30:00.000Z"
}
```

```
{
  "message": "Note not found"
}
```

```
{
  "message": "You are not the owner"
}
```

4. **Updates the title of all notes created by a logged-in user.) (Get the new Title from the body) (Get the id for the logged-in user (userId) from the token not the body) (0.5 Grade)** • **URL**: PATCH /notes/all

```
{
  "message": "All notes updated"
}
```

```
{
  "message": "No note found"
}
```

5. **Delete a single Note by its id and return the deleted note. (Only the owner of the note can make this operation) (Get the id for the logged-in user from the token not the body) (0.5 Grade)**
   - **URL:** DELET /notes/:noteId => /notes/64d91c42d8979e1f30a12346

```
{
  "message": "deleted",
  "note": {
    "id": "64d91c42d8979e1f30a12346",
    "title": "updated",
    "content": "update content",
    "userId": "64d91c42d8979e1f30a12345",
    "createdAt": "2024-12-14T12:00:00.000Z",
    "updatedAt": "2024-12-14T12:30:00.000Z"
  }
}
```

```
{
  "message": "Note not found"
}
```

```
{
  "message": "You are not the owner"
}
```

6. **Retrieve a paginated list of notes for the logged-in user, sorted by "createdAt" in descending order. (Get page and limit from query parameters) (Get the id for the logged-in user (userId) from the token not the body) (send the token in the headers) (0.5 Grade)**
   - **URL:** GET /notes/paginate-sort => for example /notes/paginate-sort?page=2&limit=3

```
{
  "id": "64d91c42d8979e1f30a12346",
  "title": "Meeting Notes",
  "content": "Discuss project updates",
  "userId": "64d91c42d8979e1f30a12345",
  "createdAt": "2024-12-14T12:00:00.000Z",
  "updatedAt": "2024-12-14T12:30:00.000Z"
},
{
  "id": "64d91c42d8979e1f30a12347",
  "title": "Shopping List",
  "content": "Buy groceries and snacks",
  "userId": "64d91c42d8979e1f30a12345",
  "createdAt": "2024-12-14T10:00:00.000Z",
  "updatedAt": "2024-12-14T10:30:00.000Z"
},
{
  "id": "64d91c42d8979e1f30a12348",
  "title": "Workout Plan",
  "content": "Cardio and strength training",
  "userId": "64d91c42d8979e1f30a12345",
  "createdAt": "2024-12-13T16:00:00.000Z",
  "updatedAt": "2024-12-13T16:30:00.000Z"
}
```

Assignment8

7. **Get a note by its id. (Only the owner of the note can make this operation) (Get the id for the logged-in user (userId) from the token not the body) (0.5 Grade)**
   - **URL:** GET /notes/:id => /posts/64a3baf1e567890124


```
{
  "id": "64d91c42d8979e1f30a12348",
  "title": "Workout Plan",
  "content": "Cardio and strength training",
  "userId": "64d91c42d8979e1f30a12345",
  "createdAt": "2024-12-13T16:00:00.000Z",
  "updatedAt": "2024-12-13T16:30:00.000Z"
}
```

```
{
  "message": "Note not found"
}
```

```
"message": "You are not the owner"
```


8. **Get a note for logged-in user by its content. (Get the id for the logged-in user (userId) from the token not the body) (0.5 Grade)**
   - **URL:** GET /notes/note-by-content => / notes/note-by-content?content=Workout Plan


```
{
  "id": "64d91c42d8979e1f30a12347",
  "title": "Shopping List",
  "content": "Buy groceries and snacks",
  "userId": "64d91c42d8979e1f30a12345",
  "createdAt": "2024-12-14T10:00:00.000Z",
  "updatedAt": "2024-12-14T10:30:00.000Z"
}
```

```
{
  "message": "No note found"
}
```


9. **Retrieves all notes for the logged-in user with user information, selecting only the "title, userId and createdAt" from the note and the "email" from the user. (Get the id for the logged-in user (userId) from the token not the body) (0.5 Grade)**
   - **URL:** GET /notes/note-with-user


```
{
  "_id": "64a3baf1e567890123",
  "title": "Note 1",
  "userId": {
    "email": "user1@email.com"
  },
  "createdAt": "2024-12-01T12:00:00Z"
},
{
  "_id": "64a3baf1e567890124",
  "title": "Note 2",
  "userId": {
    "email": "user1@email.com"
  },
  "createdAt": "2024-12-02T14:30:00Z"
},
{
  "_id": "64a3baf1e567890125",
  "title": "Note 3",
  "userId": {
    "email": "user1@email.com"
  },
  "createdAt": "2024-12-03T10:15:00Z"
}
```


10. **Using aggregation, retrieves all notes for the logged-in user with user information (name and email) and allow searching notes by the title. (1 Grade)**
    - **URL: GET /notes/aggregate => /notes/aggregate?title=Code Review Notes**


```
{
  "title": "Code Review Notes",
  "userId": "64d91c42d8979e1f30a12345",
  "createdAt": "2024-12-09T15:30:00.000Z",
  "user": {
    "name": "user1",
    "email": "user1@example.com"
  }
}
```


11. **Delete all notes for the logged-in user. (Get the id for the logged-in user (userId) from the token not the body) (0.5 Grade)**
    - **URL: DELETE /notes**


```
{
  "message":"Deleted"
}
```

Assignment8

Important Notes about postman

1. **Name the endpoint with a meaningful name like 'Add User', not dummy names.**
2. **Save your changes on each request (ctrl+s).**
3. **Include the Postman collection link (export your Postman collection) in the email with your assignment link**

Bonus (2 Grades)

How to deliver the bonus?

1- Solve the problem Longest Common Prefix on **LeetCode**

2- Inside your assignment folder, create a **SEPARATE FILE** and name it “bonus.js”

3- Copy the code that you have submitted on the website inside ”bonus.js” file