Micro-Service for Validation of Mobile Number

A REPORT

submitted to the CEO in partial fulfillment of the requirements for the TASK-Code Challenge MERN

JUNE 26, 2025

Table of Contents

Micro-Service for Validation of Mobile Number

Project Overview

- This project is a Mobile Number Validation System with:
  A microservice for phone number validation, built using the libphonenumber-js library.

```
“import { parsePhoneNumberFromString } from 'libphonenumber-js';”
```

- A RESTful API backend for CRUD operations
- A React frontend to interact with the API
- The backend uses Node.js and MongoDB, while the frontend is built with React + MUI (Material UI).

Functional Requirements Summary

1. Microservice for Mobile Number Validation (Task 1.1)

Endpoint: POST /validate

Validates mobile numbers using the libphonenumber-js library.

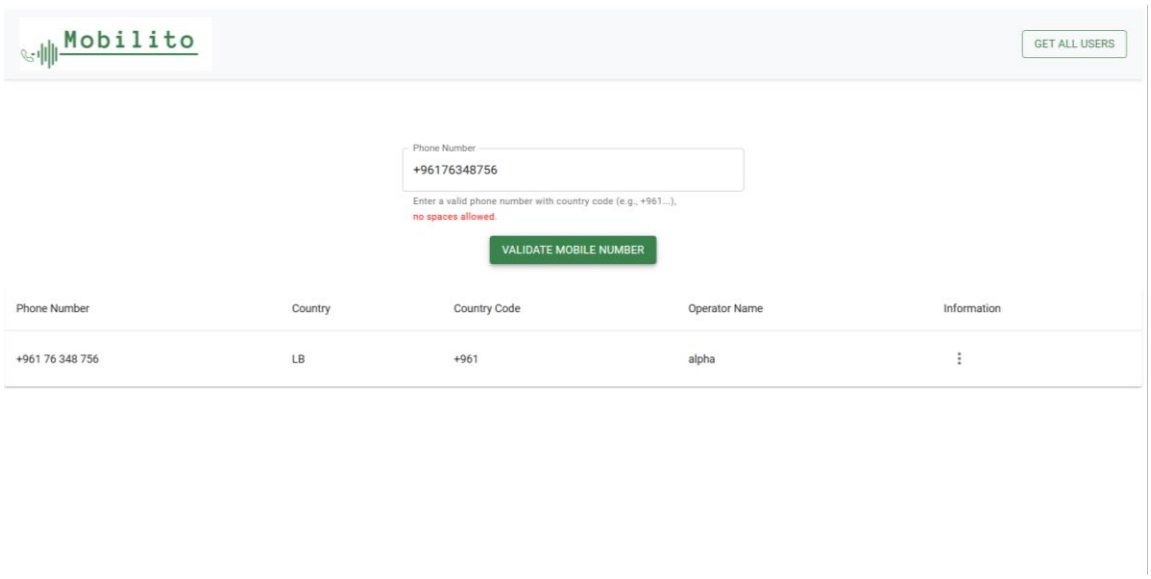
```
const handleValidate = async () => {
  const parsed = parsePhoneNumberFromString(number);

  if (parsed && parsed.isValid()) {
    setIsValid(true);
    const formattedNumber = parsed.formatInternational();

    const requestBody = {
      users_name: "Guest",
      phone_number: formattedNumber,
      description: "Validated by UI",
      operator_name: "Not available",
      country_name: parsed.country || "Unknown",
      country_code: parsed.countryCallingCode || "N/A",
    };
```

```
} else {
  setIsValid(false);
  setOpen(true);
  setShowTable(false);
}
```

If the number is valid, it sets the internal state isValid to true and formats the number to an international format. It then constructs a requestBody object with default values such as

```
const requestBody = {
        users_name: "Guest",
        phone_number: formattedNumber,
        description: "Validated by UI",
        operator_name: "Not available",
        country_name: parsed.country || "Unknown",
        country_code: parsed.countryCallingCode || "N/A",
      };
```

These details are later used to populate and display a table on the webpage. The table only contains: Phone Number, Country, Country Code, Operator Name, Information



However, if the number is not valid, the application sets isValid to false, hides the table using setShowTable(false), and opens a warning dialog using setOpen(true) to inform the user that the phone number entered is invalid.

2. REST API Endpoints

2.1 Add User (Task 1.2)

POST /adduser

```
app.post('/adduser', async (req, res) => {
  const { users_name, phone_number, description, operator_name,
country_name, country_code } = req.body;
  try {
    const existingUser = await MobileSystem.findOne({ phone_nbr:
phone_number });
    if (existingUser) {
      return res.status(200).json({
        message: 'Phone number already exists',
        user: existingUser,
      });
    }
    const newUser = new MobileSystem({
      users_name,
      phone_nbr: phone_number,
      description,
      operator_name,
      country_name,
      country_code,
    });
    await newUser.save();
    return res.status(201).json({
      message: 'New user added',
      user: newUser,
    });
  } catch (err) {
    console.error('DB error:', err);
    return res.status(500).json({ error: 'Internal server error' });
  }
});
```

Request Body:

{
 "users_name": "Guest",
 "phone_number": "+96170123456",
 "description": "Validated by UI",
 "operator_name": "Not available",
 "country_name": "Lebanon",
 "country_code": "961"

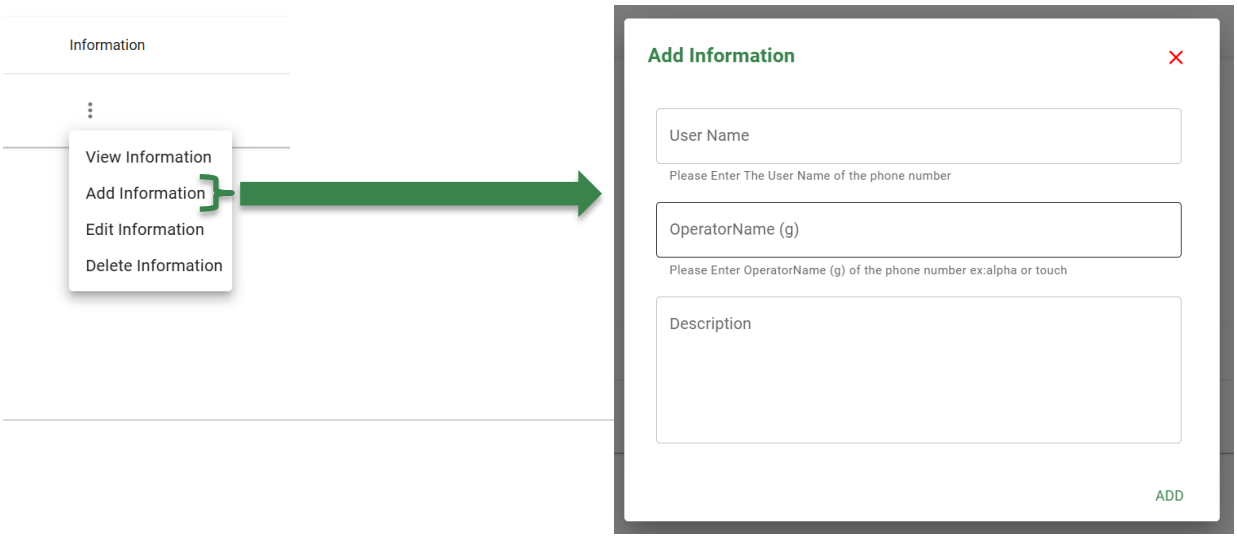
}
Response if user exists:
{
 "message": "Phone number already exists",
 "user": { ... }
}

All of this occurs after the phone number is validated. The code then calls the adduser API to add the user to the database. If the user already exists, their details are retrieved and displayed in the table. If not, default data is shown in the table instead.

```
const handleValidate = async () => {
  const parsed = parsePhoneNumberFromString(number);

  if (parsed && parsed.isValid()) {
    setIsValid(true);
    const formattedNumber = parsed.formatInternational();

    const requestBody = {
      users_name: "Guest",
      phone_number: formattedNumber,
      description: "Validated by UI",
      operator_name: "Not available",
      country_name: parsed.country || "Unknown",
      country_code: parsed.countryCallingCode || "N/A",
    };

    try {
      const response = await fetch("http://localhost:3000/adduser", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(requestBody),
      });

      const data = await response.json();
      if (!response.ok) throw new Error(data.error || "Unknown error");

      const user = data.user;

      setRows([
        {
          number: user.phone_nbr,
          country: user.country_name,
          countryCode: user.country_code,
          operatorName: user.operator_name,
        },
      ]);
      setShowTable(true);
      setOpen(false);
    } catch (error) {
      console.error("Error:", error);
      alert("There was an error contacting the server.");
    }
```

```
_id: ObjectId('685c348f04d2c9fb53a42220')
users_name : "Mohamad Amin Kabbani"
phone_nbr : "+961 76 348 756"
description : "Validated by UI"
operator_name : "alpha"
country_name : "LB"
country_code : "+961"
__v : 0
```

In the Last part of the table, it contains Information which is all found in a separate tsx file (in other words, ADD, Edit, Delete, View Information are all in separate file called Menu.tsx):



here the user can add additional information for the specific phone number.

2.2 Update User (Task 1.3)

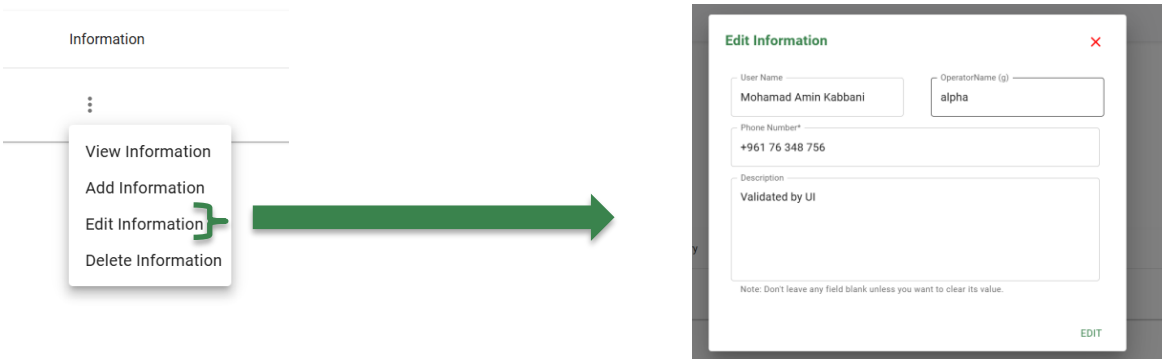
PUT /edituser/:phone_nbr

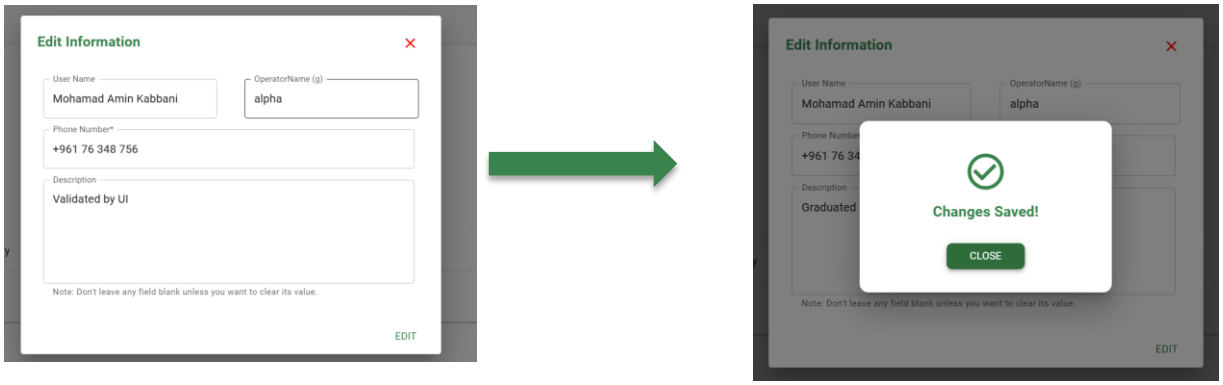
```
const handleEditUser = async () => {
  try {
    const parsed = parsePhoneNumberFromString(phoneNumber);

    if (!parsed || !parsed.isValid()) {
      alert('Invalid phone number');
      return;
    }

    const formattedNumber = parsed.formatInternational(); // e.g., "+961 3 348 756"

    const requestBody = {
      users_name: userName,
      phone_number: formattedNumber,
      description: description,
      operator_name: operatorName,
      country_name: parsed.country || 'Unknown',
      country_code: `+${parsed.countryCallingCode}`,
    };

    const response = await fetch(`http://localhost:3000/edituser/${phone_nbr}`, {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(requestBody),
    });

    const data = await response.json();
    console.log(data);
    handleCloseAddInfo();

    setOpenDoneDialog(true);
  } catch (error) {
    console.error('Error editing user:', error);
    alert('There was an error editing the user.');
  }
};
```

```
// ✅ Edit user
app.put('/edituser/:phone_nbr', async (req, res) => {
  const { phone_nbr } = req.params;
  const { users_name, phone_number, description, operator_name, country_name, country_code } = req.body;
  try {
    const updatedUser = await MobileSystem.findOneAndUpdate(
      { phone_nbr },
      {
        users_name,
        phone_nbr: phone_number,
        description,
        operator_name,
        country_name,
        country_code,
      },
      { new: true }
    );
    if (!updatedUser) {
      return res.status(404).json({ message: 'User not found' });
    }
    res.json({
      message: 'User updated successfully',
      user: updatedUser,
    });
  } catch (error) {
    console.error('Error updating user:', error);
    res.status(500).json({ error: 'Error updating user' });
  }
});
```

In the Last part of the table it contains Information:

If the User wants to edit the information





```
_id: ObjectId('685c348f04d2c9fb53a42220')
users_name : "Mohamad Amin Kabbani"
phone_nbr : "+961 76 348 756"
description : "Validated by UI"
operator_name : "alpha"
country_name : "LB"
country_code : "+961"
__v : 0
```

```
_id: ObjectId('685c348f04d2c9fb53a42220')
users_name : "Mohamad Amin Kabbani"
phone_nbr : "+961 76 348 756"
description : "Graduated from LAU"
operator_name : "alpha"
country_name : "LB"
country_code : "+961"
__v : 0
```

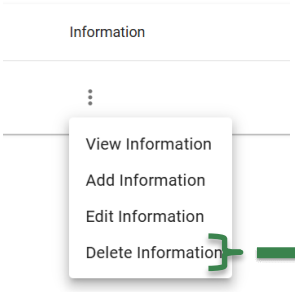
2.3 Delete User (Task 1.4)

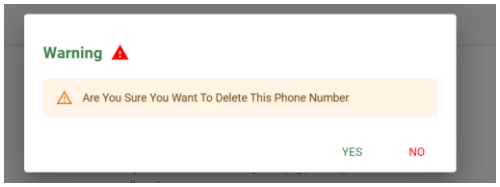
DELETE /deleteuser/:phone_nbr

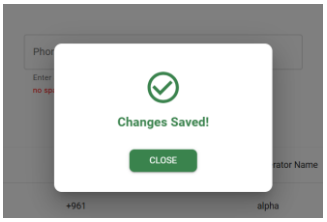
Same Process as the edit user:

```
const handleDeleteUser = async () => {
  try {
    const response = await fetch(`http://localhost:3000/deleteuser/${phone_nbr}`, {
      method: 'DELETE',
    });

    const data = await response.json();
    console.log(data);
  } catch (error) {
    console.error('Error deleting user:', error);
  }

  handleCloseDeleteInfo(); // Close the menu after deletion
  handleClose();

  setOpenDoneDialog(true);
};
```

```
// ✅ Delete user
app.delete('/deleteuser/:phone_nbr', async (req, res) => {
  const { phone_nbr } = req.params;
  try {
    const deletedUser = await MobileSystem.findOneAndDelete({ phone_nbr });
    if (!deletedUser) {
      return res.status(404).send('User not found.');
    }
    res.send(`User with phone_nbr ${phone_nbr} deleted successfully.`);
  } catch (error) {
    console.error('Error deleting user:', error);
    res.status(500).send('Error deleting user.');
  }
});
```








```
_id: ObjectId('685c348f04d2c9fb53a42220')
users_name : "Mohamad Amin Kabbani"
phone_nbr : "+961 76 348 756"
description : "Graduated from LAU"
operator_name : "alpha"
country_name : "LB"
country_code : "+961"
__v : 0
```

```
_id: ObjectId('685c367604d2c9fb53a42227')
users_name : "Maher Kabbani"
phone_nbr : "+961 3 348 756"
description : "Works in Lamakina a Family business opened by his father Amin Kabbani …"
operator_name : "alpha"
country_name : "LB"
country_code : "+961"
__v : 0
```

```
_id: ObjectId('685d1d03726e4e95eba1642c')
users_name : "Maram Anadoli"
phone_nbr : "+961 3 797 177"
description : "Maher Kabbani's Wife"
operator_name : "touch"
country_name : "LB"
country_code : "+961"
__v : 0
```


```
_id: ObjectId('685c367604d2c9fb53a42227')
users_name : "Maher Kabbani"
phone_nbr : "+961 3 348 756"
description : "Works in Lamakina a Family business opened by his father Amin Kabbani …"
operator_name : "alpha"
country_name : "LB"
country_code : "+961"
__v : 0
```

```
_id: ObjectId('685d1d03726e4e95eba1642c')
users_name : "Maram Anadoli"
phone_nbr : "+961 3 797 177"
description : "Maher Kabbani's Wife"
operator_name : "touch"
country_name : "LB"
country_code : "+961"
__v : 0
```

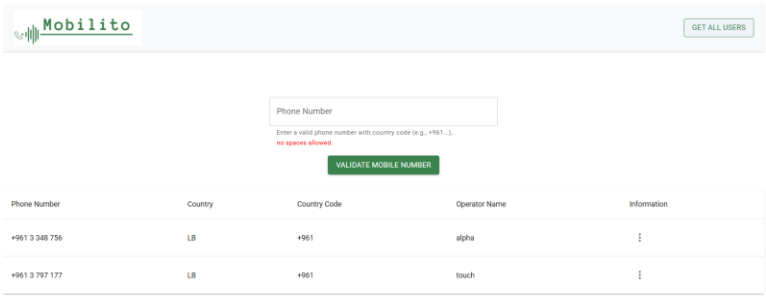
2.4 Get All Users (Task 1.5)

GET /getallusers

```
const handleGetAllUsers = async () => {
  try {
    const response = await fetch("http://localhost:3000/getallusers");
    const data = await response.json();

    if (!response.ok || !Array.isArray(data.users)) {
      throw new Error(data.error || "Invalid response");
    }

    const userRows: RowData[] = data.users.map((user: any) => ({
      number: user.phone_nbr,
      country: user.country_name,
      countryCode: user.country_code,
      operatorName: user.operator_name,
    }));

    setRows(userRows);
    setShowTable(true);
    setOpen(false);
  } catch (error) {
    console.error("Error fetching users:", error);
    alert("Failed to fetch users.");
  }
};
```

```
app.get('/getallusers', async (req, res) => {
  try {
    const users = await MobileSystem.find();
    res.json({ users });
  } catch (error) {
    console.error('Error fetching all users:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```

In this application, the process of retrieving all users is handled through a function passed as a prop from the parent component Mobile_Page.tsx to the child component Header.tsx. Inside Mobile_Page.tsx, a function named handleGetAllUsers is defined, which sends a GET request to the backend API endpoint /getallusers using the fetch method. Once the response is received, the user data is mapped into a specific structure and stored in the component's state using setRows. This data is then used to render a table showing all users. The function handleGetAllUsers is passed down to the Header component via props under the name onGetAllUsers. In the Header.tsx file, a button labeled "Get All Users" is rendered. When this button is clicked, it calls the onGetAllUsers function, which triggers the API call and displays the user data. This setup follows a common React pattern where a child component communicates with its parent by invoking functions provided through props.



Why Phone Number as ID?

The phone number is used as the unique ID for each user because:
- Phone numbers are naturally unique.
- Avoids generating IDs.
- Easier to query/update/delete users directly.

Frontend (React)

- Location: Mobile_Page.tsx, Menu.tsx
- Handles add, validate, error handling, table display
- Button in header calls getallusers

Backend (Node.js + PostgreSQL)

Database Table: MOBILE_SYSTEM
Key Fields:
- users_name
- phone_nbr (Primary key)
- description
- operator_name
- country_name
- country_code

Running the Project

Backend Setup

1. npm install
2. node server.js
3. Runs on http://localhost:3000

Frontend Setup

1. cd client
2. npm install
3. npm install libphonenumber-js
4. npm run dev
5. Runs on http://localhost:5173

🗀 Directory Structure

```
backend/
├── server.js
├── database.js
└── frontend/
    ├── src/
    │   ├── component/
    │   │   ├── Header.tsx
    │   │   ├── Header.css
    │   │   ├── Menu.tsx
    │   │   └── Done.tsx
    │   ├── Mobile_Page/
    │   │   ├── Mobile_Page.tsx
    │   │   └── Mobile_Page.css
```

Note used ChatGpt for the design of the Directory Structure