# Customer API Design

## 1. Create Customer

Route: POST /api/customer

Payload (body):

```json

{

"name": "John Doe",

"dateOfBirth": "1990-01-01",

"memberNumber": 1,

"interests": "movies, football"

}

```

Response:

```json

{

"\_id": "60d3b41ef682d44e6a1a3f5c",

"name": "John Doe",

"dateOfBirth": "1990-01-01T00:00:00.000Z",

"memberNumber": 1,

"interests": "movies, football",

"createdAt": "2023-10-11T12:00:00.000Z",

"updatedAt": "2023-10-11T12:00:00.000Z"

}

```

File: /app/api/customer/route.js

Test: curl -X POST -H "Content-Type: application/json" -d '{"name":"John Doe","dateOfBirth":"1990-01-01","memberNumber":1,"interests":"movies, football"}' http://localhost:3000/api/customer

## 2. Get All Customers

Route: GET /api/customer

Payload (body): None

Response:

```json

[

{

"\_id": "60d3b41ef682d44e6a1a3f5c",

"name": "John Doe",

"dateOfBirth": "1990-01-01T00:00:00.000Z",

"memberNumber": 1,

"interests": "movies, football",

"createdAt": "2023-10-11T12:00:00.000Z",

"updatedAt": "2023-10-11T12:00:00.000Z"

},

// ... more customers

]

```

File: /app/api/customer/route.js

Test: curl http://localhost:3000/api/customer

## 3. Get Customer by ID

Route: GET /api/customer/[id]

Payload (body): None

Response:

```json

{

"\_id": "60d3b41ef682d44e6a1a3f5c",

"name": "John Doe",

"dateOfBirth": "1990-01-01T00:00:00.000Z",

"memberNumber": 1,

"interests": "movies, football",

"createdAt": "2023-10-11T12:00:00.000Z",

"updatedAt": "2023-10-11T12:00:00.000Z"

}

```

File: /app/api/customer/[id]/route.js

Test: curl http://localhost:3000/api/customer/60d3b41ef682d44e6a1a3f5c

## 4. Update Customer

Route: PUT /api/customer/[id]

Payload (body):

```json

{

"name": "John Doe Updated",

"dateOfBirth": "1990-01-01",

"memberNumber": 1,

"interests": "movies, football, gym"

}

```

Response:

```json

{

"\_id": "60d3b41ef682d44e6a1a3f5c",

"name": "John Doe Updated",

"dateOfBirth": "1990-01-01T00:00:00.000Z",

"memberNumber": 1,

"interests": "movies, football, gym",

"createdAt": "2023-10-11T12:00:00.000Z",

"updatedAt": "2023-10-11T13:00:00.000Z"

}

```

File: /app/api/customer/[id]/route.js

Test: curl -X PUT -H "Content-Type: application/json" -d '{"name":"John Doe Updated","dateOfBirth":"1990-01-01","memberNumber":1,"interests":"movies, football, gym"}' http://localhost:3000/api/customer/60d3b41ef682d44e6a1a3f5c

## 5. Delete Customer

Route: DELETE /api/customer/[id]

Payload (body): None

Response:

```json

{

"message": "Customer deleted successfully"

}

```

File: /app/api/customer/[id]/route.js

Test: curl -X DELETE http://localhost:3000/api/customer/60d3b41ef682d44e6a1a3f5c

## Error Responses

For all endpoints, if an error occurs, the response will be in the following format:

```json

{

"error": "Error message describing what went wrong"

}

```

The HTTP status code will also be set appropriately (e.g., 404 for not found, 500 for server error).

## Notes

1. All endpoints require a connection to the MongoDB database, which is handled by the `dbConnect()` function from `/lib/db.js`.

2. The `dateOfBirth` field is stored as an ISO date string in the database but should be sent as "YYYY-MM-DD" in requests.

3. The `memberNumber` field should be unique for each customer.

4. The API uses the `Customer` model defined in `/models/Customer.js`.

5. All endpoints include error handling to catch and return appropriate error messages and status codes.

6. The Update Customer endpoint now requires all fields to be sent in the request body, not just the fields being updated.