



Digital Egypt Pioneers Initiative

مبادرة رواد مصر الرقمية

Num2Verify

Virtual Numbers Provider

Team Members

Mohamed Adel Mohamed
Mahmoud Gamal Shaker
Alaa Reda Farouq
Mohamed Ali

Acknowledgment

We would like to express our gratitude to Eng Hassan ELDash for his efforts over the course of the initiative.

Table of Contents

1	Introduction	6
2	System Requirements	7
2.1	User Registration (Sign Up)	7
2.2	User Login (Sign In)	7
2.3	Forgot Password	8
2.4	View Countries List	8
2.5	Claim a Number	9
2.6	View User's Claimed Numbers	9
2.7	View Number Messages	9
2.8	Delete a Number.....	10
3	Technologies	11
3.1	ReactJS.....	11
3.2	MUI.....	12
3.3	MongoDB.....	12
3.4	NodeJS.....	13
3.5	Express.....	14
3.6	AWS EC2	15
4	Front-End	16
4.1	Front-End Structure	16
4.2	API Fetcher with Axios	17
4.3	Data Storage with React Context	17
4.4	Pages	18
4.4.1	About.jsx	18
4.4.2	SignIn.jsx	19
4.4.3	SignUp.jsx.....	20
4.4.4	ActivateAccount.jsx.....	21

4.4.5	CountryList.jsx.....	22
4.4.6	CountryNumbers.jsx.....	23
4.4.7	ForgotPassword.jsx	24
4.4.8	Home.jsx	25
4.4.9	profile.jsx	26
4.4.10	userNumbers.jsx.....	27
4.4.11	NumberMessages.jsx.....	28
4.4.12	Privacy.jsx	29
5	Database	30
5.1	MongoDB Collections.....	30
5.1.1	User.....	30
5.1.2	Country	30
5.1.3	Numbers	31
5.1.4	UserNumbers	31
5.1.5	TokenBlockList	31
6	Back-end Server	32
6.1	Back-End Structure	32
6.2	Server and routers	33
6.2.1	numberRoutes.js.....	33
6.2.2	numberRoutes.swagger.js.....	33
6.2.3	userRoutes.js.....	33
6.2.4	userRoutes.swagger.js	33
6.3	Middlewares.....	34
6.3.1	authMiddleware.js	34
6.4	Controllers	34
6.4.1	numberController.js.....	34
6.4.2	userController.js.....	34

6.5	Services.....	34
6.5.1	findMessages.js.....	34
6.5.2	MongoDB_idTold.js.....	34
6.5.3	sendEmailCode.js	34
6.5.4	updateCountryNumberCounts.js.....	35
6.5.5	validateEmail.js	35
6.5.6	validatePassword.js.....	35
6.6	Swagger Docs.....	36
7	Hosting and Deployment.....	37
7.1	Building the Website.....	37
7.2	MongoDB Atlas Hosting	37
7.3	Amazon EC2 Hosting	37
8	Conclusion.....	38

1 Introduction

Num2Verify is an innovative platform that provides users with temporary virtual phone numbers to receive account verification messages securely and conveniently. Whether you're signing up for a new service, creating a social media account, or verifying your identity on various online platforms, Num2Verify offers a seamless solution to ensure your privacy and security.

With Num2Verify, you can bypass the need to share your personal phone number while still completing the necessary verification processes. Our service is designed for flexibility and ease of use, providing you with access to a wide range of virtual numbers that can receive SMS verification codes from different countries and services. Whether you're protecting your privacy or need a temporary number for multiple accounts, Num2Verify has you covered.

Num2Verify's intuitive platform is user-friendly, ensuring that anyone can quickly obtain and use a virtual number for verification purposes. Simply choose a number, receive your verification code, and complete your registration without revealing your personal phone number.

Key features of Num2Verify:

- Instant access to virtual numbers from multiple countries
- Quick and secure SMS reception for verification codes
- Enhanced privacy by keeping your personal number private
- Easy-to-use interface with seamless functionality

Num2Verify is your go-to solution for virtual phone numbers, providing both convenience and privacy for your online verification needs.

2 System Requirements

2.1 User Registration (Sign Up)

The system should allow new users to create an account by providing necessary details.

Inputs: Full Name, Email Address (Must be unique), Password (Must meet security requirements), Confirm Password

Validations:

- Email format should be valid.
- Password should meet complexity rules (e.g., minimum 8 characters, include uppercase, lowercase, numeric, and special characters).
- Confirm Password must match Password.

Process:

After successful submission, the system will send an account activation code via email to complete the registration process.

Output: User receives an activation code and a message indicating registration success and to check their email.

2.2 User Login (Sign In)

Registered users should be able to log in using their credentials.

Inputs: Email Address, Password

Validations:

- Email format must be valid.
- Password must match the one associated with the email.

Process:

The system will authenticate the user based on email and password combination.

After successful login, the user is redirected to their dashboard.

Output: User is logged in and taken to the dashboard.

Send Account Activation Code

After signing up, users will receive an account activation code via email to verify their email address.

Inputs: User's email (from registration).

Process: An activation email with a unique code/link will be sent to the registered email.

Validations: Ensure that the email is valid and was not previously used for an active account.

Output: Email is sent, and the system informs the user that the code was sent.

2.3 Forgot Password

Users who forgot their password should be able to reset it by providing their registered email.

Inputs: Email Address

Validations: Ensure the email is registered.

Process:

- The system sends a password reset link/code to the provided email.
- Users can follow the link or enter the reset code to reset their password.
- The new password should follow security rules.

Output: Email with reset instructions is sent, and a message is displayed informing the user.

2.4 View Countries List

The system should display a list of countries where virtual numbers are available.

Process: Fetch the list of available countries from the system's database.

Output: A list of countries with available virtual numbers is displayed. Each country should have a name and a flag icon.

View Numbers in a Country

Users should be able to view a list of available virtual numbers for any selected country.

Inputs: Country selection from the list.

Process: Based on the selected country, retrieve the available virtual numbers from the system's database.

Output: A list of virtual numbers for the chosen country is displayed, showing the number, status (available/claimed), and time since last use.

2.5 Claim a Number

Users should be able to claim a virtual number for receiving account verification messages.

Inputs: User login (must be authenticated), Selected virtual number

Validations:

- The number must be available for claiming.
- Ensure the user has not exceeded the maximum number of claimed numbers.

Process: Mark the number as claimed and associate it with the user.

Output: The number is marked as claimed and added to the user's account. A confirmation message is displayed.

2.6 View User's Claimed Numbers

Users should be able to view the virtual numbers they have claimed.

Inputs: User login (must be authenticated)

Process: Retrieve the list of numbers associated with the logged-in user from the system's database.

Output: A list of claimed numbers is displayed, showing the number, country, date of claiming, and status (active/expired).

2.7 View Number Messages

Users should be able to view messages (e.g., SMS) received by a claimed virtual number.

Inputs: User login (must be authenticated), Selected number from the user's claimed list

Process: Fetch the message history for the selected number from the system's database.

Output: A list of messages is displayed, showing the sender (if available), message content, and time received.

2.8 Delete a Number

Users should be able to delete (release) a virtual number they have claimed.

Inputs: User login (must be authenticated), Selected virtual number from the user's claimed list

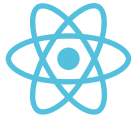
Validations: Ensure the number belongs to the user.

Process: Mark the number as released, making it available for others to claim. Remove the association of the number from the user's account.

Output: The number is successfully deleted, and a confirmation message is displayed.

3 Technologies

3.1 ReactJS



ReactJS is the frontend framework used in the Num2Verify platform. It is a popular open-source JavaScript library developed and maintained by Facebook for building user interfaces, particularly single-page applications.

Component-based architecture: React allows the creation of reusable UI components, making it easier to manage and maintain the codebase. This modularity ensures that different sections of the site, such as number lists, message views, and the user dashboard, are encapsulated in components that can be developed and debugged independently.

Virtual DOM: React's Virtual DOM feature optimizes rendering performance by minimizing direct DOM manipulation. It ensures that the site is responsive and provides a smooth user experience, even during frequent UI updates, such as refreshing messages or checking new numbers for verification.

State Management: With React's built-in state management (along with tools like React Context or external libraries like Redux if needed), Num2Verify manages dynamic data effectively, such as the availability of phone numbers, incoming messages, and user preferences.

React Hooks: Hooks such as `useState`, `useEffect`, and custom hooks help in managing component lifecycle events, API calls, and UI interactivity, ensuring a seamless experience when users interact with various components of the platform.

3.2 MUI



Material-UI (MUI) is a React UI framework that follows Google's Material Design principles, used for building responsive, modern, and consistent user interfaces.

Pre-built UI Components: MUI provides a vast collection of reusable, customizable components such as buttons, dialogs, forms, icons, and navigation elements. This greatly accelerates development and ensures a consistent look and feel across the entire application.

Customizability: While MUI components follow Material Design standards out of the box, they are also highly customizable, allowing Num2Verify to maintain its own unique branding, including color schemes, typography, and layout.

Responsiveness: MUI ensures that the Num2Verify platform is fully responsive, making it accessible across devices of different sizes. Its grid system and layout utilities provide a mobile-first design approach, ensuring that the platform is usable on desktops, tablets, and smartphones.

Theming: MUI's theme provider allows developers to define global styles and themes that can be applied consistently across the app. Custom themes are particularly useful for setting site-wide design parameters, including colors, fonts, and spacing, ensuring a coherent user experience.

3.3 MongoDB



MongoDB is a NoSQL database used as the primary data storage solution for Num2Verify. As a document-oriented database, it stores data in flexible, JSON-like documents rather than the traditional table-based relational format.

Schema Flexibility: MongoDB's flexible schema makes it easy to store various types of data, such as user details, virtual phone numbers, verification message logs, and service provider information. This allows Num2Verify to adapt to changing requirements without the need for extensive database migrations.

Scalability: MongoDB's horizontal scaling capabilities ensure that the platform can handle large volumes of data efficiently as the number of users and virtual numbers increases. The platform can easily scale across multiple servers to meet growing demand.

Indexing and Performance: MongoDB supports a range of indexing options to optimize query performance, ensuring that users can quickly access virtual numbers and retrieve their verification messages with minimal latency.

Replica Sets and High Availability: MongoDB supports replica sets, ensuring that Num2Verify remains available even in the case of hardware or server failures. The database can be set up with multiple copies across servers to provide high availability and disaster recovery.

3.4 NodeJS



NodeJS is the runtime environment used for the backend of Num2Verify. It is an open-source, cross-platform, JavaScript runtime built on Chrome's V8 engine, which allows JavaScript to be executed on the server-side.

Asynchronous and Event-driven: NodeJS's non-blocking, event-driven architecture is particularly beneficial for Num2Verify, as it ensures efficient handling of concurrent requests. This is important when multiple users are retrieving or interacting with the virtual numbers simultaneously.


JavaScript Everywhere: By using NodeJS, Num2Verify employs JavaScript across the entire stack, from the frontend to the backend. This simplifies development, as developers can work with a single language across both client-side and server-side codebases.

NPM Ecosystem: NodeJS provides access to the extensive NPM (Node Package Manager) library, which offers thousands of packages that can be integrated into the project. This accelerates development by allowing the team to incorporate

existing libraries for tasks such as authentication, input validation, and API handling.

Scalability: NodeJS is highly scalable, making it ideal for handling real-time features, such as receiving and processing SMS messages from multiple phone numbers, and ensuring that Num2Verify can scale as the user base and service demand grow.

3.5 Express

 Express is the web application framework for NodeJS, providing robust functionality for building APIs and handling HTTP requests and responses.

Minimalist and Unopinionated: Express is lightweight and unopinionated, which gives the developers flexibility in designing the API structure for Num2Verify. It does not enforce a specific directory structure or require complex configurations, allowing for rapid development of the platform's backend.

Middleware Support: Express's middleware architecture enables modular functionality. For example, authentication middleware can handle user login sessions, while error-handling middleware can catch and manage API errors gracefully, providing smooth interaction between the frontend and backend.

Routing: Express provides an intuitive routing system, allowing Num2Verify to manage different API endpoints effectively. Whether handling user requests for phone numbers, processing SMS messages, or communicating with third-party verification services, Express's routing ensures that API endpoints are organized and scalable.

Security Features: Express supports several security best practices, including the use of Helmet for securing HTTP headers and express-session for managing user sessions securely. This ensures that Num2Verify adheres to web security standards, providing users with a safe and reliable platform.

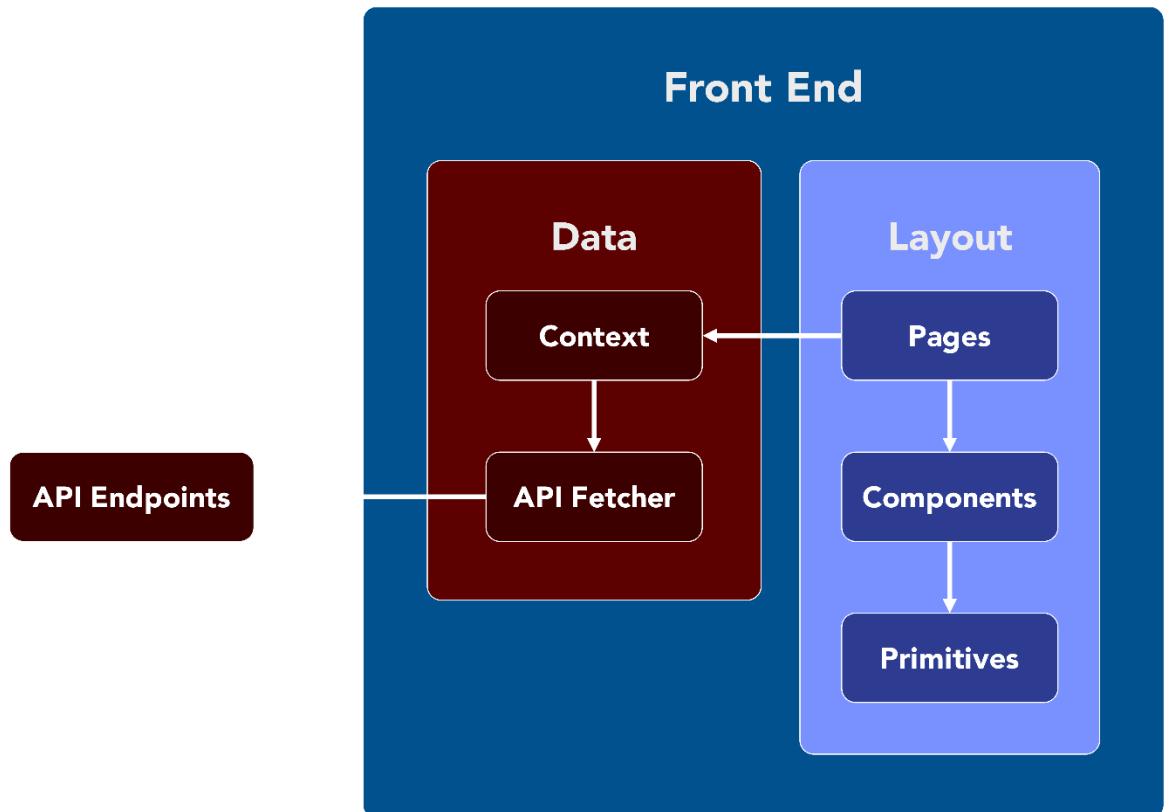
3.6 AWS EC2



Amazon EC2 (Elastic Compute Cloud) is a scalable, on-demand cloud computing service that allows you to run virtual servers (instances) to host applications. It provides resizable compute capacity in the cloud, enabling easy scaling to meet traffic demands. EC2 supports various instance types, operating systems, and configurations, offering flexibility and control over computing resources. It integrates with other AWS services, offers secure networking, and can be managed via the AWS Management Console, CLI, or API. Ideal for hosting websites, EC2 provides cost-effective, pay-as-you-go pricing.

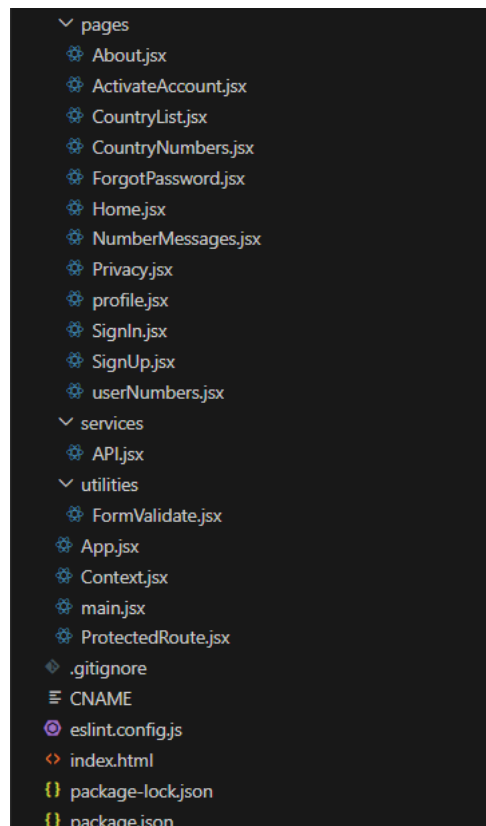
4 Front-End

4.1 Front-End Structure



```
web-app
├── dist
├── node_modules
├── public
├── src
├── jsonData
├── modules
├── components
│   ├── AppAppBar.jsx
│   ├── AppFooter.jsx
│   ├── CountryCard.jsx
│   ├── CountryNumberCard.jsx
│   ├── DataCardContainer.jsx
│   ├── Markdown.jsx
│   ├── NumberCard.jsx
│   ├── ProductHero.jsx
│   ├── SearchBox.jsx
│   ├── SlidingCard.jsx
│   ├── SlidingCardSection.jsx
│   └── Snackbar.jsx
```

```
primitives
├── AppForm.jsx
├── Button.jsx
├── CardPaper.jsx
├── FormFields.jsx
├── GlowShadow.jsx
├── ListSelect.jsx
├── PageTitle.jsx
├── SubmitButton.jsx
├── SubmitCancelButtonCombo.jsx
├── SubmitError.jsx
├── TextField.jsx
├── Typography.jsx
└── theme.jsx
```

4.2 API Fetcher with Axios

The API Fetcher module utilizes Axios, a promise-based HTTP client for JavaScript, to communicate with the Num2Verify backend services. Axios simplifies the process of making requests to our API, handling responses, and managing errors efficiently.

4.3 Data Storage with React Context

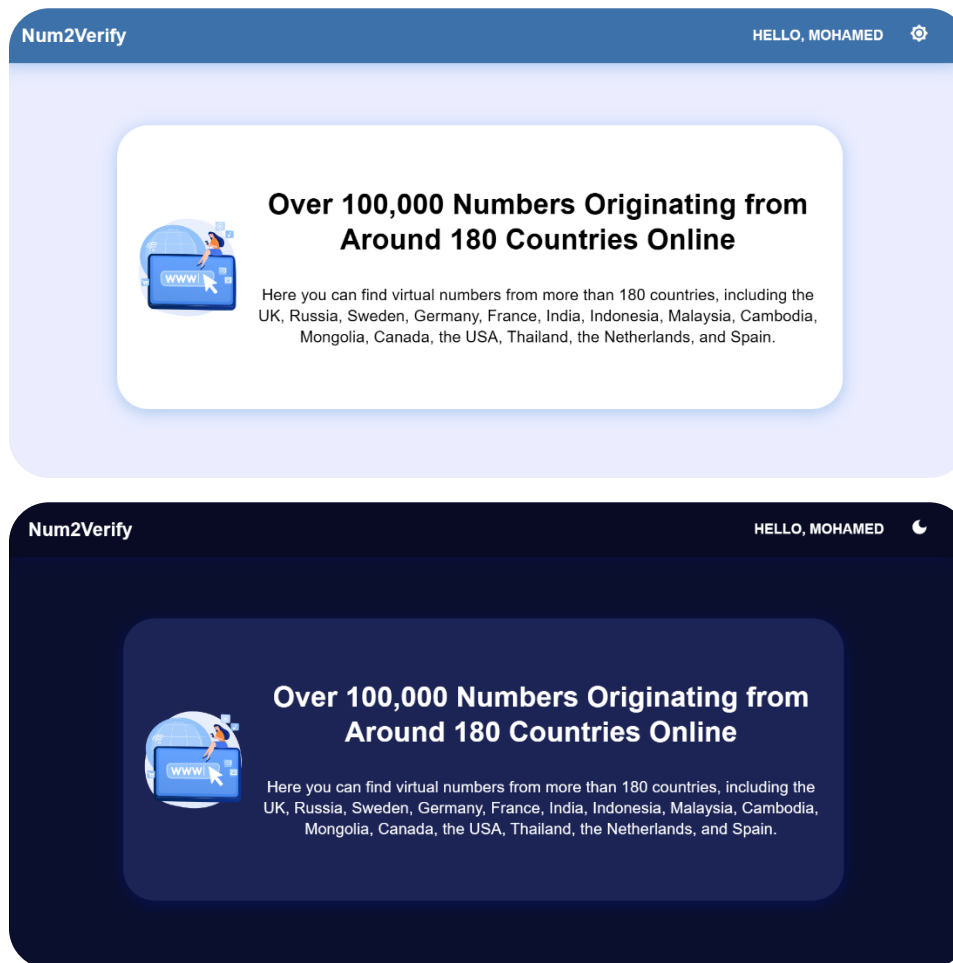
React Context provides a way to manage global state for the Num2Verify application. It allows for easier data sharing across components without the need to pass props down manually through each level of the component tree.

4.4 Pages

4.4.1 About.jsx

Purpose: Provides information about Num2Verify, including its services and benefits.

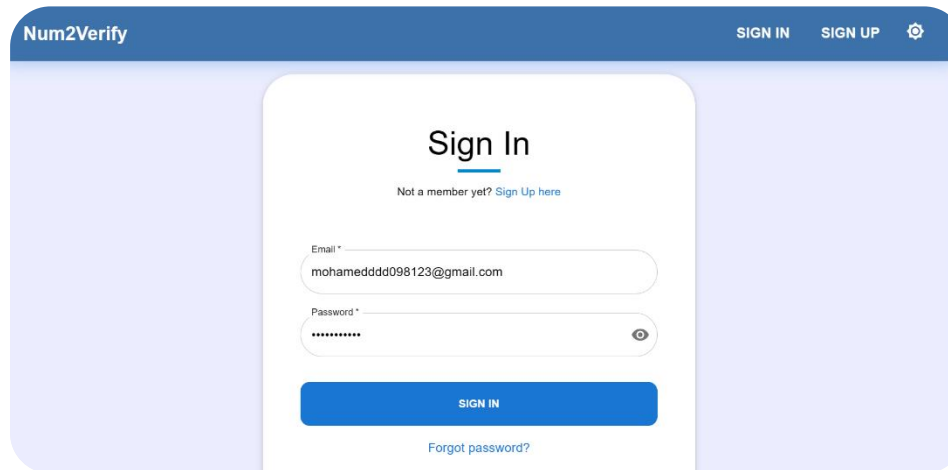
Features: Displays a brief overview of the platform and links to additional resources or support.



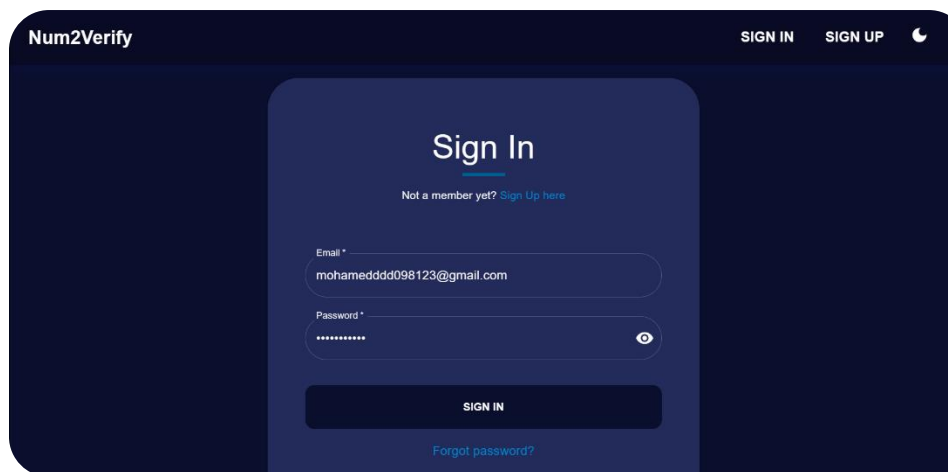
4.4.2 SignIn.jsx

Purpose: Allows users to log in to their Num2Verify account.

Features: Input fields for email and password, "Remember me" checkbox, and login button.



The image shows a light-themed web interface for the Num2Verify application. At the top, a dark blue header bar contains the "Num2Verify" logo on the left and "SIGN IN" and "SIGN UP" links on the right, along with a settings icon. The main content area has a light purple background. In the center, a white rounded rectangle contains the "Sign In" title, a link for "Not a member yet? Sign Up here", and two input fields: "Email *" with the value "mohamedddd098123@gmail.com" and "Password *" with masked characters. A blue "SIGN IN" button is positioned below the inputs, and a "Forgot password?" link is at the bottom.



The image shows a dark-themed version of the Num2Verify Sign In form. The header bar is dark blue with the "Num2Verify" logo and "SIGN IN", "SIGN UP", and a moon icon. The background is a dark navy blue. The central white rounded rectangle contains the "Sign In" title, a link for "Not a member yet? Sign Up here", and two input fields: "Email *" with the value "mohamedddd098123@gmail.com" and "Password *" with masked characters. A dark blue "SIGN IN" button is positioned below the inputs, and a "Forgot password?" link is at the bottom.

4.4.3 SignUp.jsx

Purpose: Facilitates new user registration.

Features: Form for email, password, and confirmation, along with terms agreement checkbox.

The image displays two versions of a web application's 'Sign Up' form, stacked vertically. Both versions feature a dark blue header with the 'Num2Verify' logo on the left and 'SIGN IN' and 'SIGN UP' links on the right. The top version has a light blue background, while the bottom version has a dark blue background. The form itself is a white rounded rectangle in the top version and a dark blue rounded rectangle in the bottom version. It contains the title 'Sign Up' with a blue underline, a link 'Already have an account?' in blue, and three input fields: 'Email *', 'Password *' (with an eye icon), and 'Confirm Password *' (with an eye icon). The bottom version also includes a small white moon icon in the top right corner of the header.

4.4.4 ActivateAccount.jsx

Purpose: Facilitates account activation after sign-up via an activation link or code.

Features: Input field for activation code, submission button, and success/error messaging.

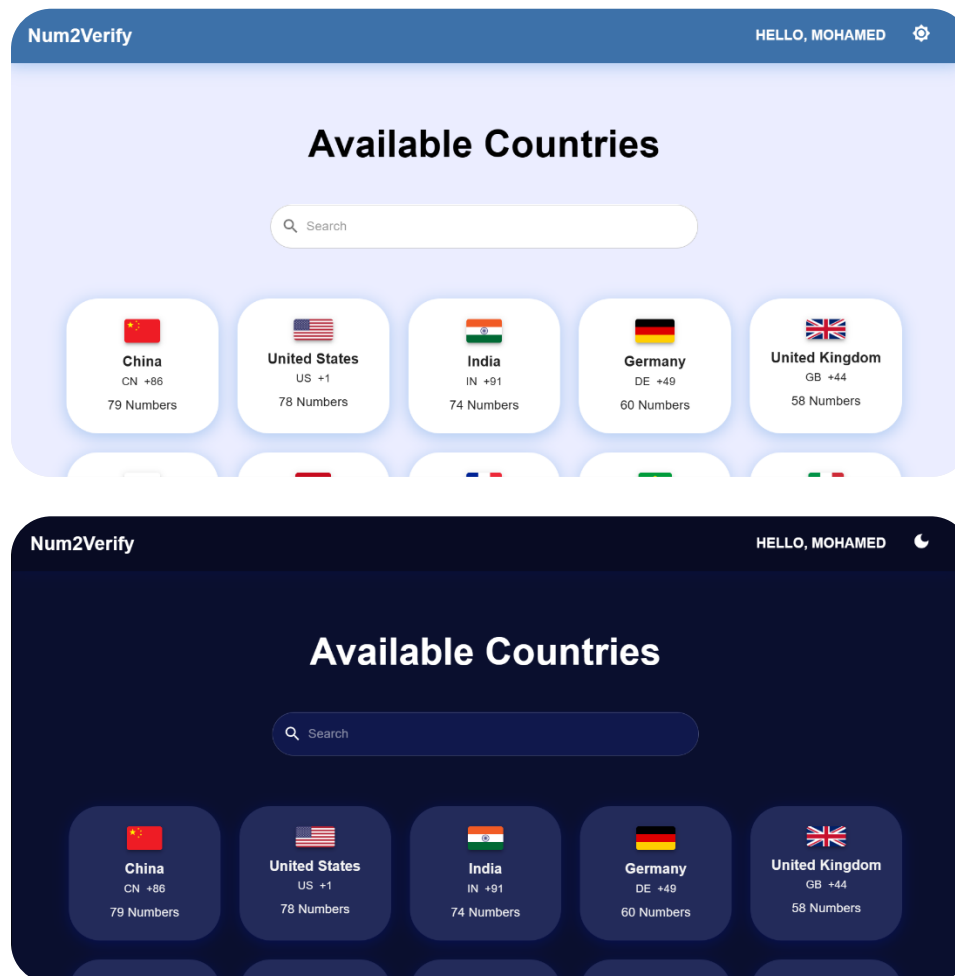
The image displays two versions of the 'Enter Activation Email' form for the Num2Verify application. The top version is in a light blue theme, and the bottom version is in a dark blue theme. Both forms are centered on the page and contain the following elements:

- Header:** 'Num2Verify' logo on the left, and 'SIGN IN', 'SIGN UP', and a settings gear icon on the right.
- Title:** 'Enter Activation Email' with a small blue underline under the word 'Activation'.
- Input Field:** A rounded rectangular field with the placeholder text 'Email *'.
- Button:** A blue button with the text 'GET CODE'.

4.4.5 CountryList.jsx

Purpose: Displays a list of countries that offer virtual numbers for verification.

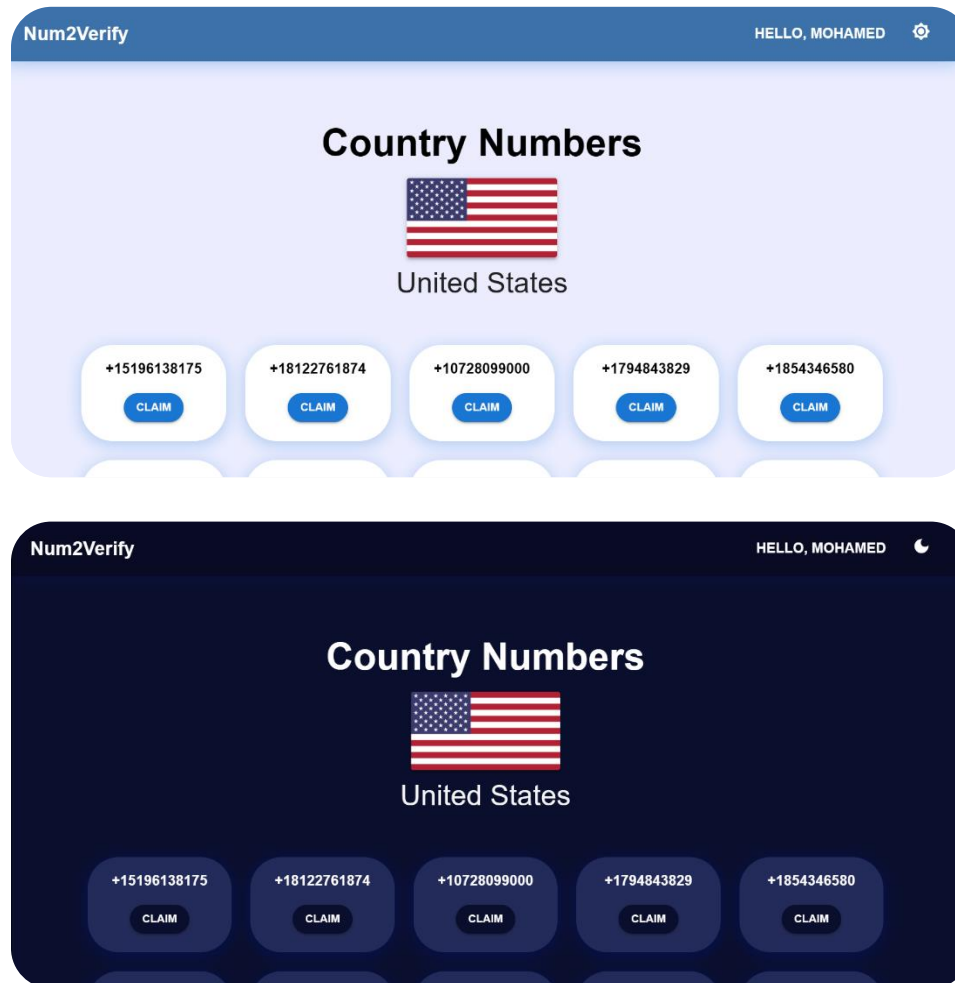
Features: Search functionality and dynamic filtering by country.



4.4.6 CountryNumbers.jsx

Purpose: Shows available virtual numbers for a selected country.

Features: List of phone numbers, selection options, and availability status.



4.4.7 ForgotPassword.jsx

Purpose: Allows users to reset their password.

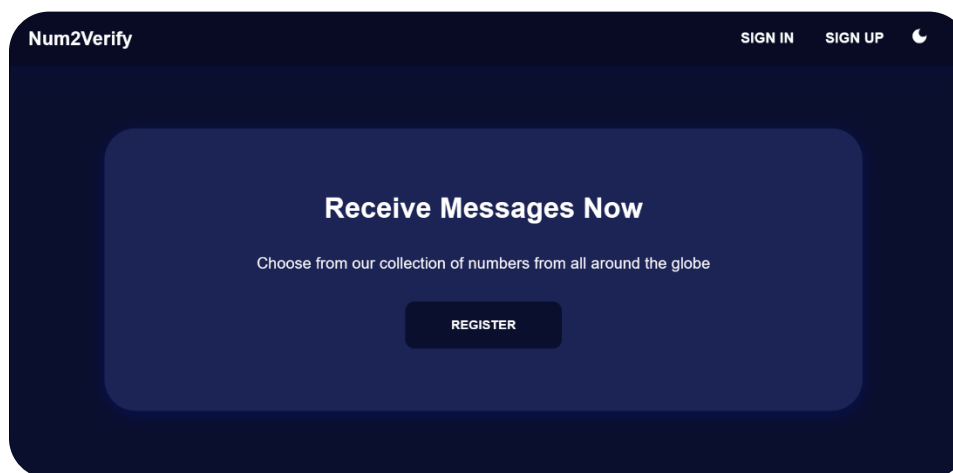
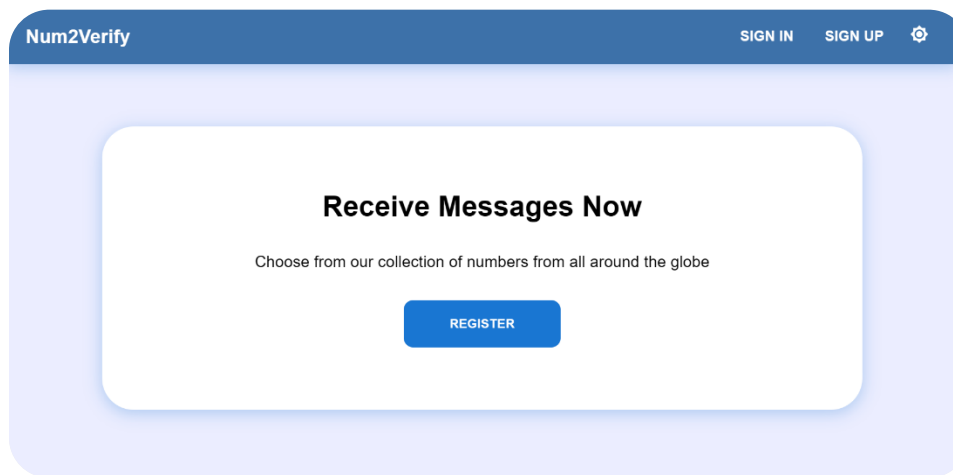
Features: Email input for password recovery, validation, and link to reset password.

The image displays two versions of a web form titled "Enter Your Email" for the "Num2Verify" application. The top version features a light blue background with a white form container. It includes a header bar with "Num2Verify", "SIGN IN", "SIGN UP", and a settings icon. The form has a title "Enter Your Email" with a blue underline on "Your", an "Email *" input field, and a blue "GET CODE" button. The bottom version is a dark-themed variant with a dark blue background and a dark blue form container. It maintains the same layout and text but uses white and light blue for the form elements. The header bar in the dark theme includes a moon icon for theme toggling.

4.4.8 Home.jsx

Purpose: Acts as the landing page of Num2Verify, providing an overview of services.

Features: Service highlights, call-to-action buttons, and quick links to sign up or sign in.



4.4.9 profile.jsx

Purpose: Allows users to view and edit their account details.

Features: Form for updating user information such as name, email, and password.

The image displays two versions of a web application's profile page, one in a light blue theme and one in a dark blue theme. Both versions show a user profile with the name 'Mohamed' and email 'mohamedddd098123@gmail.com'. The top navigation bar includes the 'Num2Verify' logo, the user's name 'HELLO, MOHAMED', and a settings icon. The profile section is titled 'Profile' and features a placeholder for a profile picture. Below the picture are two input fields: 'Email *' and 'First Name *'. The 'Email *' field contains the text 'mohamedddd098123@gmail.com' and the 'First Name *' field contains the text 'Mohamed'.

Num2Verify HELLO, MOHAMED

Profile

Email *

mohamedddd098123@gmail.com

First Name *

Mohamed

Num2Verify HELLO, MOHAMED

Profile

Email *

mohamedddd098123@gmail.com

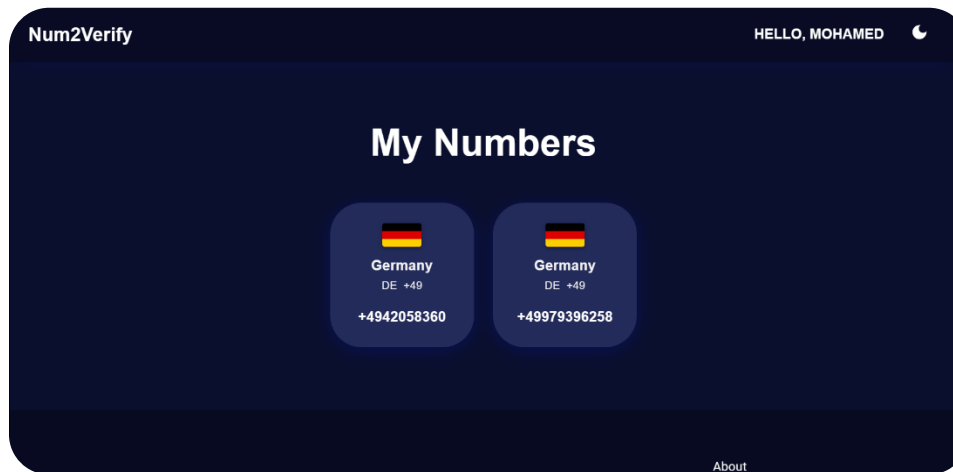
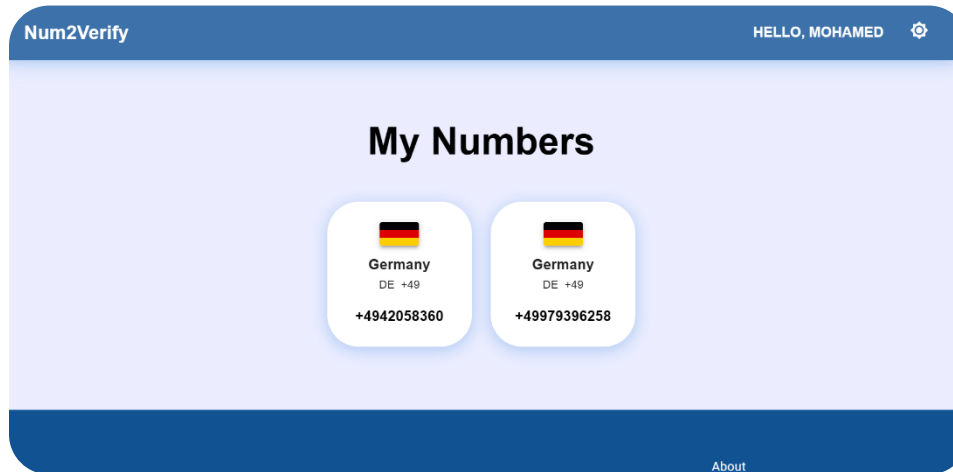
First Name *

Mohamed

4.4.10 userNumbers.jsx

Purpose: Displays a list of virtual numbers the user has used or reserved.

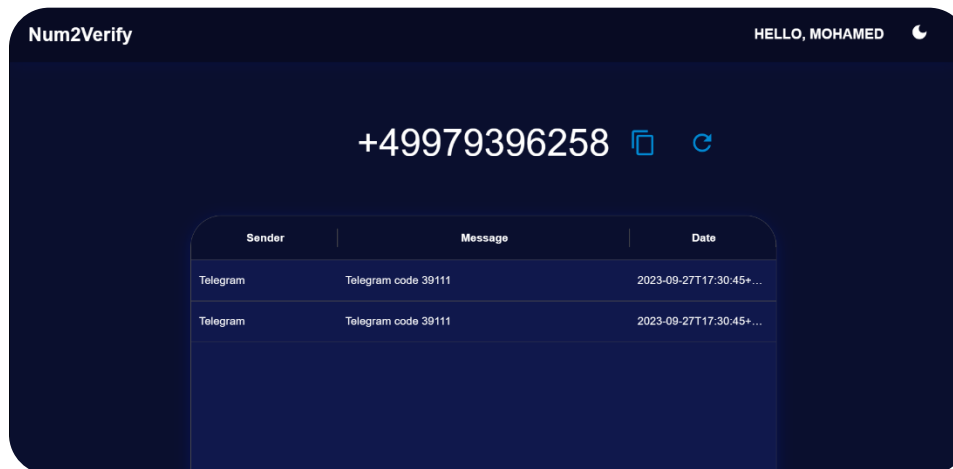
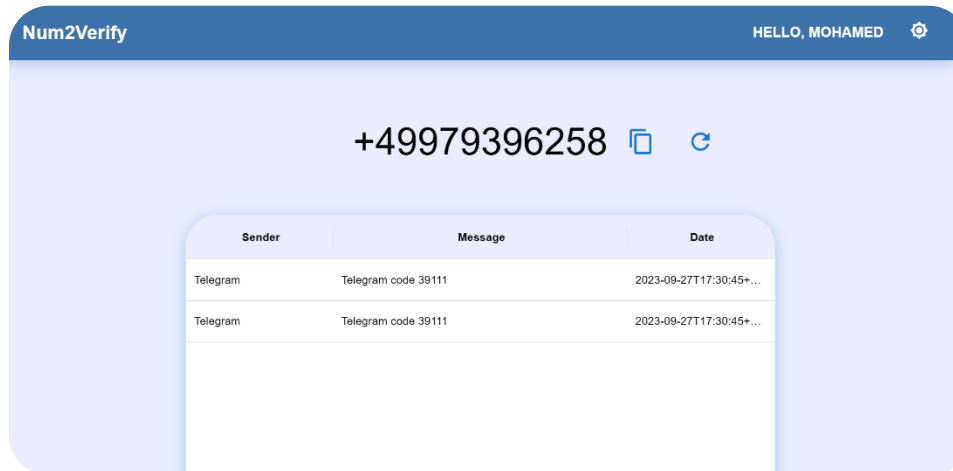
Features: Overview of active numbers, expiration dates, and message history links.



4.4.11 NumberMessages.jsx

Purpose: Displays messages received by a selected virtual number.

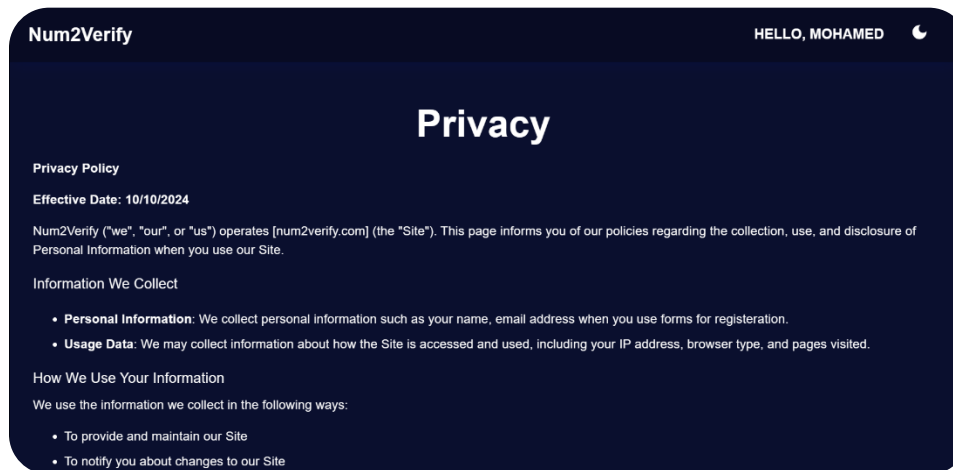
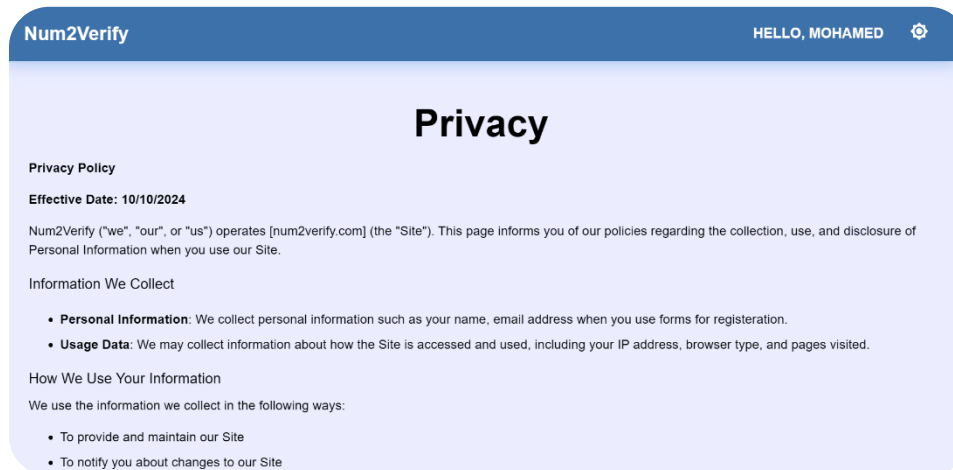
Features: Message inbox with timestamps, sender info, and message content.



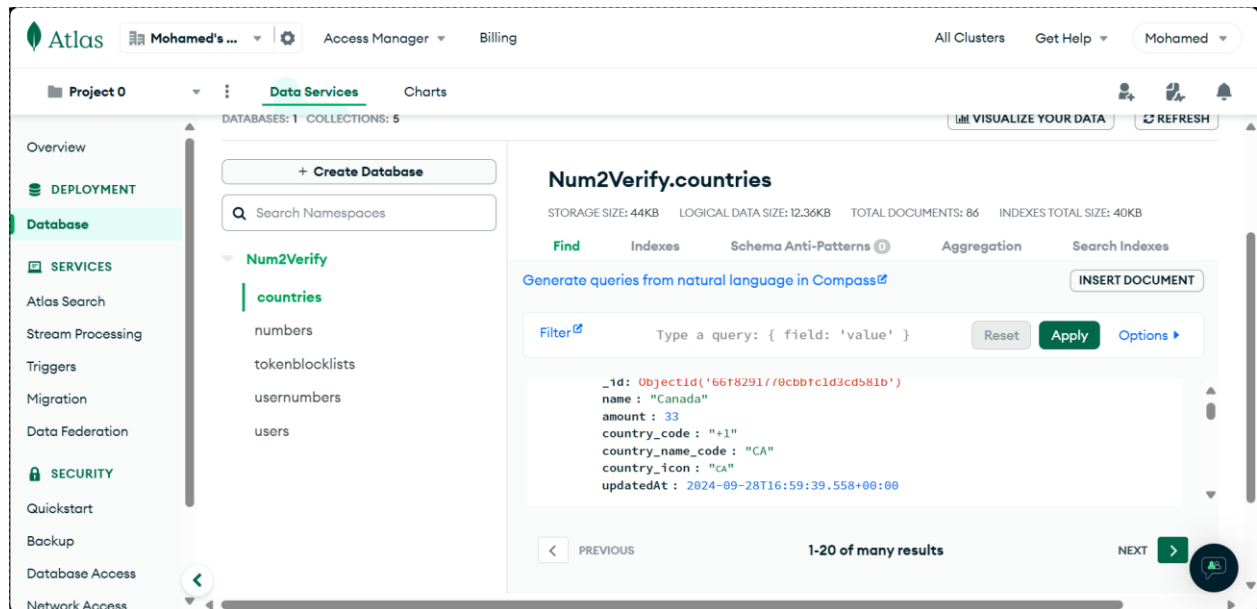
4.4.12 Privacy.jsx

Purpose: Outlines the privacy policy of Num2Verify.

Features: Text content explaining data handling, user privacy, and terms.



5 Database



5.1 MongoDB Collections

5.1.1 User

Represents a registered user on the platform.

Fields:

email (String, required, unique): User's email (validated format).

password (String, required): User's password (hashed).

first_name (String): User's first name.

last_name (String): User's last name.

picture (String): Profile picture URL.

numbers_limit (Number, default: 10): Limit of virtual numbers a user can hold.

is_active (Boolean, default: false): Whether the user is activated.

activation_code (Number, required, default: 0): Activation code for verifying account.

password_reset_code (String, default: ""): Code for password reset, if applicable.

5.1.2 Country

Represents supported countries for virtual numbers.

Fields:

name (String, required, unique): Country name.
amount (Number, default: 0): Cost associated with the country.
country_code (String, required): Country calling code (e.g., +1).
country_name_code (String, required): ISO country code (e.g., US).
country_icon (String): Optional URL for the country icon.

5.1.3 Numbers

Represents virtual phone numbers available for verification.

Fields:

country (ObjectId, required, ref: Country): Reference to the associated country.
number (String, required, unique): The virtual phone number.

5.1.4 UserNumbers

Represents the association between a user and their assigned virtual numbers.

Fields:

user (ObjectId, required, ref: User): Reference to the user.
number (ObjectId, required, ref: Number, unique): Reference to the assigned virtual number.

5.1.5 TokenBlockList

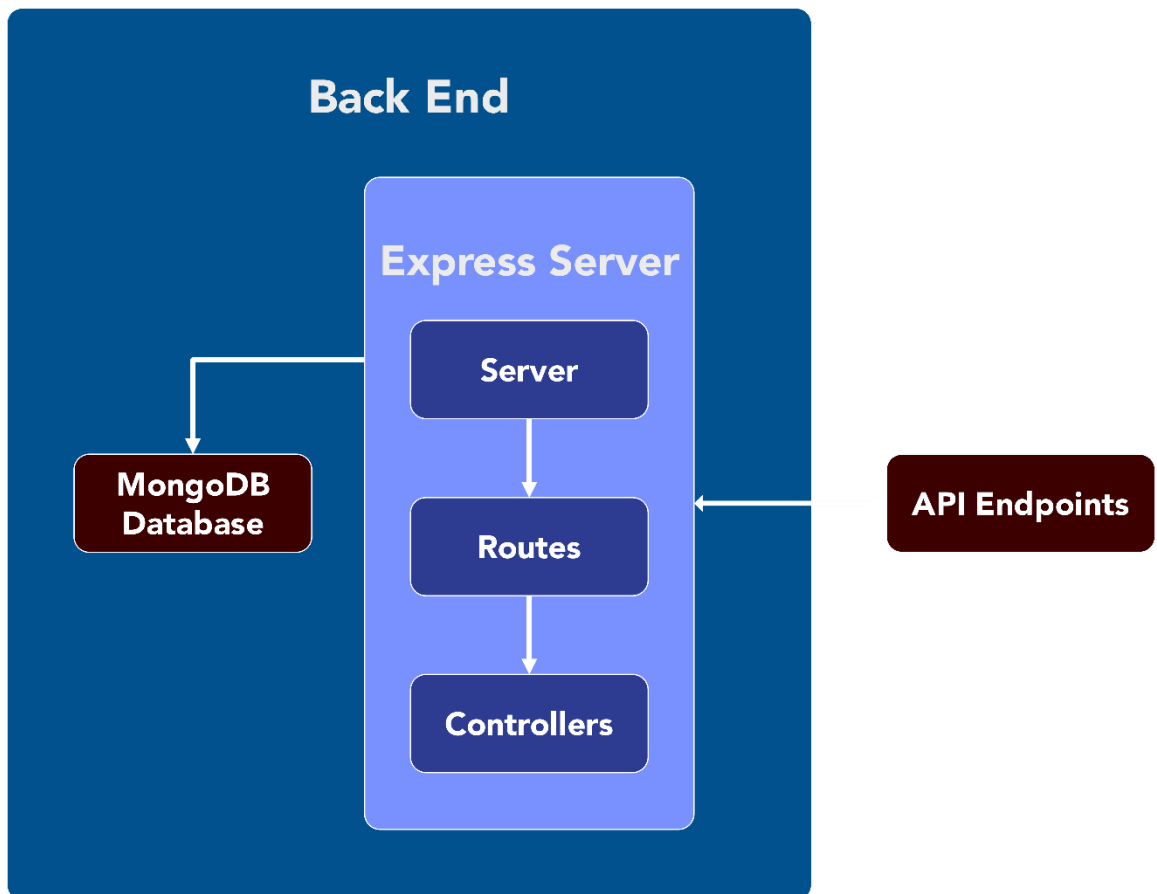
Stores invalidated tokens for security purposes.

Fields:

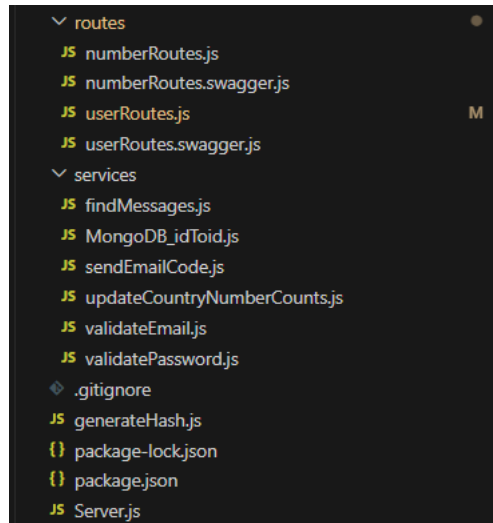
user_token (String, required, unique): The token string to be blocked.

6 Back-end Server

6.1 Back-End Structure



```
NUM2VERIFY
├── node-server
│   ├── config
│   │   └── dbConfig.js
│   ├── controllers
│   │   ├── numberController.js
│   │   └── userController.js
│   ├── middleware
│   │   ├── authMiddleware.js
│   │   └── errorMiddleware.js
│   └── models
│       ├── JSONSource
│       ├── CountryModel.js
│       ├── NumberModel.js
│       ├── TokenBlockList.js
│       ├── UserModel.js
│       └── UserNumberModel.js
```

6.2 Server and routers

6.2.1 numberRoutes.js

Handles routes related to fetching virtual numbers or countries.

Main route: /list for retrieving available numbers and countries.

Middleware: verifyToken for secure access to routes.

6.2.2 numberRoutes.swagger.js

Swagger documentation setup for numberRoutes.js, used for API documentation and testing.

6.2.3 userRoutes.js

Manages user-related routes, including registration, login, profile, password recovery, and virtual number management.

Routes: /register, /login, /profile, /numbers, /activation-code, etc.

Middleware: verifyToken for secure access, verifyAdmin for admin-only routes.

6.2.4 userRoutes.swagger.js

Swagger documentation setup for userRoutes.js, aiding in API testing and generating API docs.

6.3 Middlewares

6.3.1 authMiddleware.js

Contains authentication middlewares such as `verifyToken` for verifying JWTs and `verifyAdmin` for admin role validation.

6.4 Controllers

6.4.1 numberController.js

Handles logic for fetching virtual numbers and countries available for use.

Key function: `getCountriesOrNumbers`.

6.4.2 userController.js

Manages user-related actions:

Registration: `registerUser` validates email, password, and sends an activation code.

Login: `loginUser` authenticates users, generates JWT tokens.

Profile: `getProfile`, `updateProfile`, `deleteProfile`.

Virtual Numbers: `getUserNumbers`, `claimUserNumber`, `deleteUserNumber`, `getUserNumberMessages`.

Password Management: `getForgetPasswordCode`, `sendForgetPasswordReset`, `passwordReset`.

6.5 Services

6.5.1 findMessages.js

Fetches messages received by virtual numbers for account verification.

6.5.2 MongoDB_idToId.js

Utility for converting MongoDB `_id` to a custom ID format for easier referencing.

6.5.3 sendEmailCode.js

Service for sending email-based verification codes, such as account activation codes.

6.5.4 updateCountryNumberCounts.js

Updates the available number counts for each country after a number is claimed.

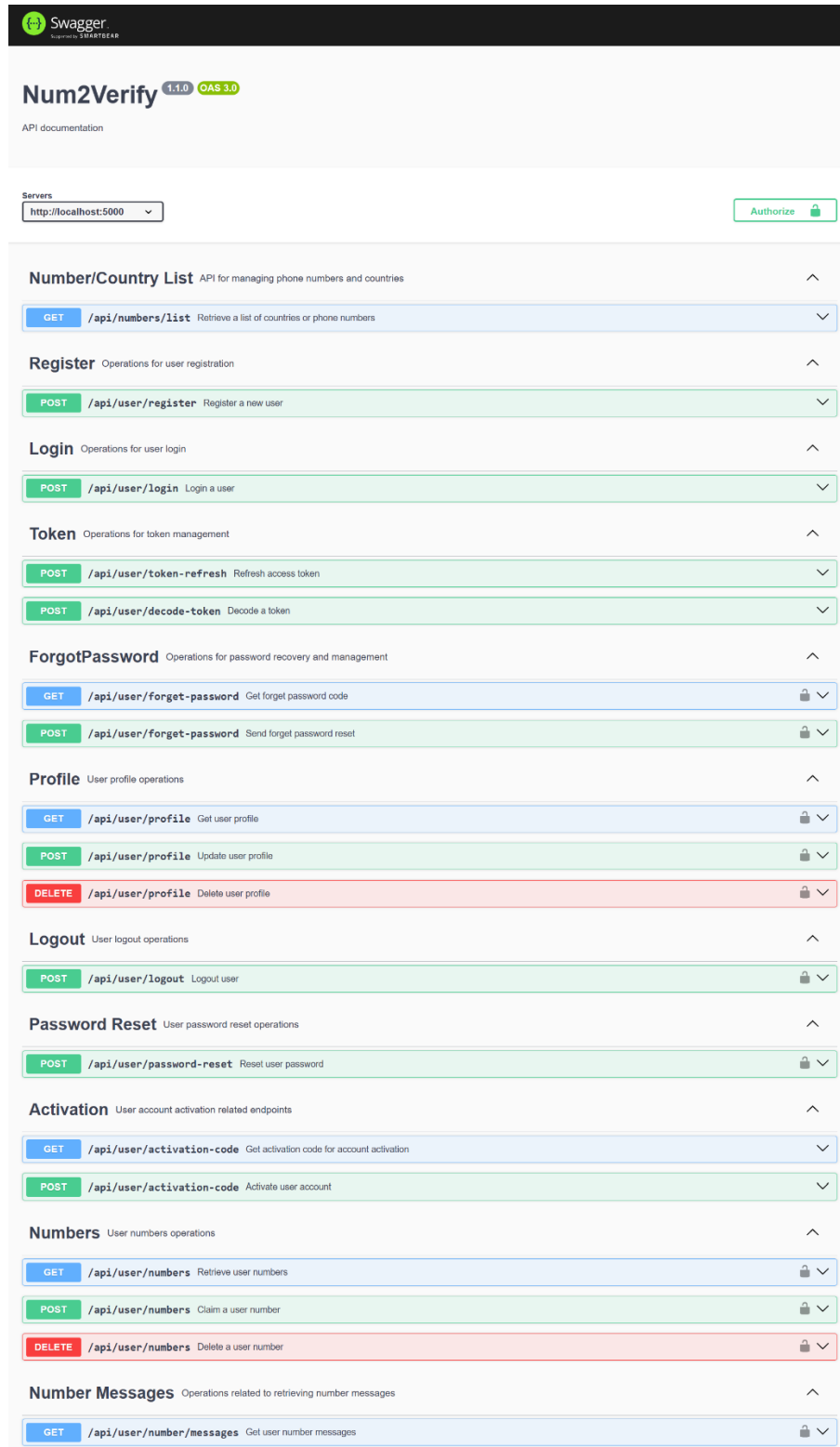
6.5.5 validateEmail.js

Validates email format and structure for user registration and login.

6.5.6 validatePassword.js

Ensures password strength by validating it against pre-set rules for user security.

6.6 Swagger Docs



The image shows the Swagger UI for the Num2Verify API. The header includes the Swagger logo and the text "Num2Verify 1.1.0 OAS 3.0". Below the header, there is a "Servers" section with a dropdown menu showing "http://localhost:5000" and an "Authorize" button. The main content area lists various API endpoints grouped by category. Each endpoint is shown with its HTTP method, path, and a brief description. The endpoints are color-coded: blue for GET, green for POST, and red for DELETE. Some endpoints have a lock icon, indicating they are protected.

Number/Country List API for managing phone numbers and countries

- GET `/api/numbers/list` Retrieve a list of countries or phone numbers

Register Operations for user registration

- POST `/api/user/register` Register a new user

Login Operations for user login

- POST `/api/user/login` Login a user

Token Operations for token management

- POST `/api/user/token-refresh` Refresh access token
- POST `/api/user/decode-token` Decode a token

ForgotPassword Operations for password recovery and management

- GET `/api/user/forget-password` Get forget password code
- POST `/api/user/forget-password` Send forget password reset

Profile User profile operations

- GET `/api/user/profile` Get user profile
- POST `/api/user/profile` Update user profile
- DELETE `/api/user/profile` Delete user profile

Logout User logout operations

- POST `/api/user/logout` Logout user

Password Reset User password reset operations

- POST `/api/user/password-reset` Reset user password

Activation User account activation related endpoints

- GET `/api/user/activation-code` Get activation code for account activation
- POST `/api/user/activation-code` Activate user account

Numbers User numbers operations

- GET `/api/user/numbers` Retrieve user numbers
- POST `/api/user/numbers` Claim a user number
- DELETE `/api/user/numbers` Delete a user number

Number Messages Operations related to retrieving number messages

- GET `/api/user/number/messages` Get user number messages

7 Hosting and Deployment

7.1 Building the Website

To prepare the application for deployment, we execute `npm run build` to create a production-ready build. This command compiles the source code, optimizes assets, and generates static files in a `dist` directory.

7.2 MongoDB Atlas Hosting

Setup:

Create a MongoDB Atlas account and set up a new cluster.

Choose the appropriate configuration (e.g., free tier for development or higher tiers for production).

Database Connection:

Obtain the connection string from MongoDB Atlas.

Update the application configuration to include the connection string, ensuring it includes the necessary credentials (username and password) and specifies the database name.

Data Security:

Set up IP whitelisting in the Atlas dashboard to allow access only from specific IP addresses (e.g., your application server).

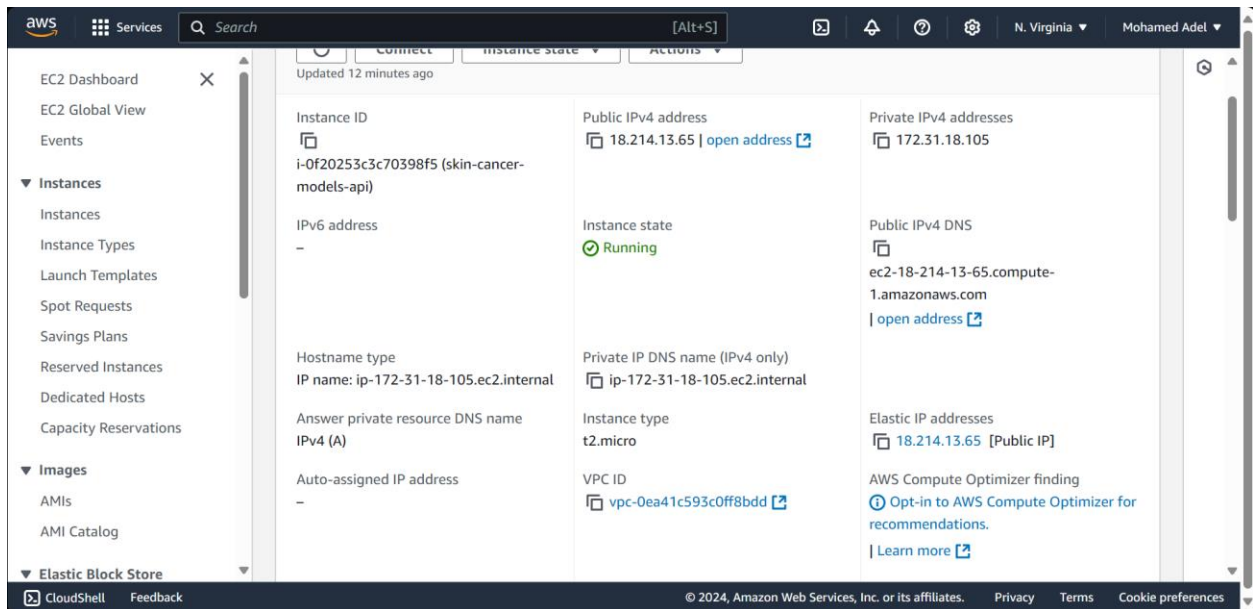
Use environment variables to store sensitive credentials securely.

7.3 Amazon EC2 Hosting

We launched a new EC2 instance using the AWS Management Console. Choose an appropriate instance type based on the expected load (we used `t2.micro`).

We then transferred the files by uploading to S3, and downloading the files on there.

We used PM2 to run the Node server, and Nginx to serve the react app.



8 Conclusion

Num2Verify provides a reliable solution for individuals and businesses seeking virtual numbers for account verification messages. By leveraging our service, users can enhance their online security and maintain privacy without compromising accessibility. Our platform is designed to be user-friendly, ensuring that anyone can easily acquire and manage virtual numbers.

With features such as real-time message delivery, global coverage, and robust privacy protections, Num2Verify stands out as a trusted partner in the digital landscape.