User Module Documentation

Overview

The `User` module is responsible for managing user information in the messaging application. It includes basic user data, profile, settings, and a list of contacts. It also provides JSON serialization and deserialization for storage or client-server communication.

This module is used for:

- Storing and retrieving user information
- Managing the user's contacts
- Providing user data in JSON format
- Validating user information

Classes and Structures

Class `User`

Attributes:

Name	Type	Description
`id`	`std::string`	Unique user identifier
`email`	`std::string`	User email
`username`	`std::string`	Username
`passwordHash`	`std::string`	Password hash
`customUrl`	`std::string`	Custom URL for the user
`profile`	`Profile`	User profile information
`settings`	`Settings`	User settings
`contacts`	`std::vector<Contact>`	List of user contacts

Methods:

Getters and Setters

- `getId() / setId(id)` – Get or set the user ID
- `getEmail() / setEmail(email)` – Get or set the email
- `getUsername() / setUsername(username)` – Get or set the username
- `getPasswordHash() / setPasswordHash(passwordHash)` – Get or set the password hash
- `getCustomUrl() / setCustomUrl(customUrl)` – Get or set the custom URL
- `getProfile() / setProfile(profile)` – Get or set the profile
- `setProfileFromJson(json)` – Set profile from a JSON string
- `getSettings() / setSettings(settings)` – Get or set settings
- `setSettingsFromJson(json)` – Set settings from a JSON string
- `getContacts() / setContacts(contacts)` – Get or set the list of contacts
- `addContact(contact)` – Add a new contact

JSON

- `toJson()` – Convert the entire user information to JSON
- `fromJson(jsonStr)` – Load user information from a JSON string
- `fromJson(json)` – Load user information from a `nlohmann::json` object

Validation

- `isValid()` – Check if the essential user information (id, email, username, passwordHash) is complete

---

Structure `Contact`

Attributes:

Name	Type	Description
`id`	`std::string`	Contact ID
`name`	`std::string`	Contact name

Methods:

- `toJson()` – Convert the contact to JSON
- `fromJson(j)` – Load the contact from JSON

Usage Example

```
User user;
user.setId("123");
user.setUsername("Ali");
user.setEmail("ali@example.com");
user.setPasswordHash("hashed_password");

// Adding a contact
User::Contact contact;
contact.id = "456";
contact.name = "Sara";
user.addContact(contact);

// Convert to JSON
std::string jsonStr = user.toJson();

// Load from JSON
User newUser;
newUser.fromJson(jsonStr);
```

Dependencies

- `nlohmann/json.hpp` for JSON handling
- `Profile.h` for user profile
- `Settings.h` for user settings

Development Notes

- The `fromJson` method silently ignores errors for invalid JSON input.
- The `isValid` method only performs basic checks and does not validate email format or password strength.
- The `Contact` class is nested within `User` and contains only basic contact information.