Database design 4 tables

Database name: Alyssa_v1

user table:
id, nickname, email, password, created_time

font table:
id, user_id, fontname, copyright, version, created_time, last_modified_time, active

glyph table:
id, font_id, charname, created_time, active

validation_code table:
user_email, text, created_time

Note: all created_time is TIMESTAMP type, active is BOOL type. The creation of the database and table should be written as a SQL script so that we can easily repeat it in the server.

API design:

All API follows the same format by receiving a JSON data from client and returning a JSON result to client

1. Signup API:
   API name: alyssa_user_signup.php
   input JSON:
   {
   “email” : “example@example.com”,

"password" : "**********",
"nickname": "some nickname"
}

output JSON:
{
"success": true/false,
"message": "some message"

}

Note: for the input, we should check them to avoid sql injection, for the password, it should be encrypted using "salt", please check http://php.net/manual/en/faq.passwords.php for more detail.
For the output, if the operation is successfully executed, return true, and give message like "User information has been successfully added". If there's something wrong, return false, and use "message" to return the error information.

After the user information successfully added into the database, create a root directory using the user id as the directory name, all incoming user data will be stored under this root directory.

2. Login API:
API name: alyssa_user_login.php
Input JSON:
{
"email" : "example@example.com",
"password" : "**********"

}
output JSON:
{
“success”: true/false,
“message”: “some message”,
(only if “success” equals true)

“nickname” : “user_nickname”,

(only if they are available)
“active_font” : “active font”,
“all_fonts_info” : [
{
“fontname” : “font1”,
“num_of_finished_chars” : 300
},
{
“fontname” : “font2”,
“num_of_finished_chars” : 500
}]
}
Note: “num_of_finished_chars” is the number of active glyph for this font.

3. Request validation code API:
API name: alyssa_request_validation_code.php
Input JSON:
{
“email” : “example@example.com”
}
Output JSON:

{
“success” : true/false,
“message”: “some message”
}
Note: check the email address in user table, if it exists, generate a validation code consisting of four digits such as “3893” and use it as a string in case we need more complicated one in the future and email this code to user. Put the code and its created time to validation_code table. The tricky part is to figure out how to send emails programmatically and avoid being regarded as spam. The validation_code also needs to be “salted” just as the password.

4. Confirm Validation Code API:
API name: alyssa_confirm_validation_code.php
Input JSON:
{
“email” : “example@example.com”,
“validation_code” : “1234”
}
Output JSON:
{
“success” : true/false,
“message”: “some message”
}
Note: check if the validation_code matches, and also compare the current time to its created_time, if the time difference more than 20 min, it’s regarded as expired.

5. Reset Password API:

API name: alyssa_user_reset_password.php
Input JSON:
{
“email” : “example@example.com”,
“validation_code”: “1456”
“new_password” : “some new password”
}
output JSON:
{
“success” : true/false,
“message”: “some message”
}

6. Create New Font API:
API name: alyssa_create_font.php
Input JSON:
{
“email” : “example@example.com”,
“password” : “some new password”,
“fontname” : “font1”,
“copyright” : “Copyright @ user”,
“version” : “version 1.0”
}
output JSON:
{
“success” : true/false,
“message”: “some message”
}
Note: we should create a new directory with font_id as the name under the root directory named by the user_id. Set this font to active.

7. Create New Glyph API:
API name: alyssa_create_glyph.php
Input JSON:
{
“email” : “example@example.com”,
“password” : “some new password”,
“fontname” : “font1”,
“charname” : “汉”,
“image” : base64encoded data
}
output JSON:
{
“success” : true/false,
“message”: “some message”
}
Note: put the image under directory user_id/font_id/glyph_id.xxx, update the font’s last_modified_time to current time. Set this glyph to active state. (Convert it to svg named after its Unicode, update the font file, I will handle this part later)

8. Fetch latest font data API:
API name: alyssa_fetch_latest_font.php
Input JSON:
{
“email” : “example@example.com”,
“password” : “some new password”,
“fontname” : “font1”

(optional, if missing, always return font data if available)

“last_modified_time” : unix timestamp
}
output JSON:
{
“success” : true/false,
“message”: “some message”,
(only if success equals true)
“font” : base64encoded font data,
“last_modified_time” : unix timestamp
}
Note: if the font data is still fresh, return success as false, and indicate the font data is still fresh in message. Return success as true only if the data is available and newer than the client’s data.

9. Email user the latest font
API name: alyssa_email_font.php
Input JSON:
{
“email” : “example@example.com”,
“password” : “some new password”,
“fontname” : “font1”
}
output JSON:
{
“success” : true/false,
“message”: “some message”
}

10. Switch active font:
API name: alyssa_switch_active_font.php

Input JSON:
{
“email” : “example@example.com”,
“password” : “some new password”,
“fontname” : “font1”
}
output JSON:
{
“success” : true/false,
“message”: “some message”
}
Note: activate this font and deactivate the previous one

11. Update user email:
API name: alyssa_update_email.php
Input JSON:
{
“email” : “example@example.com”,
“password” : “some new password”,
“new_email” : “newemail@example.com”
}
output JSON:
{
“success” : true/false,
“message”: “some message”
}
12. Fetch new book:
API name: alyssa_fetch_book.php
Input JSON:
{
“book_title” : “西游记”

}
output JSON:
{
	“success” : true/false,
	“message”: “some message”,
	“book”: base64encoded data
}