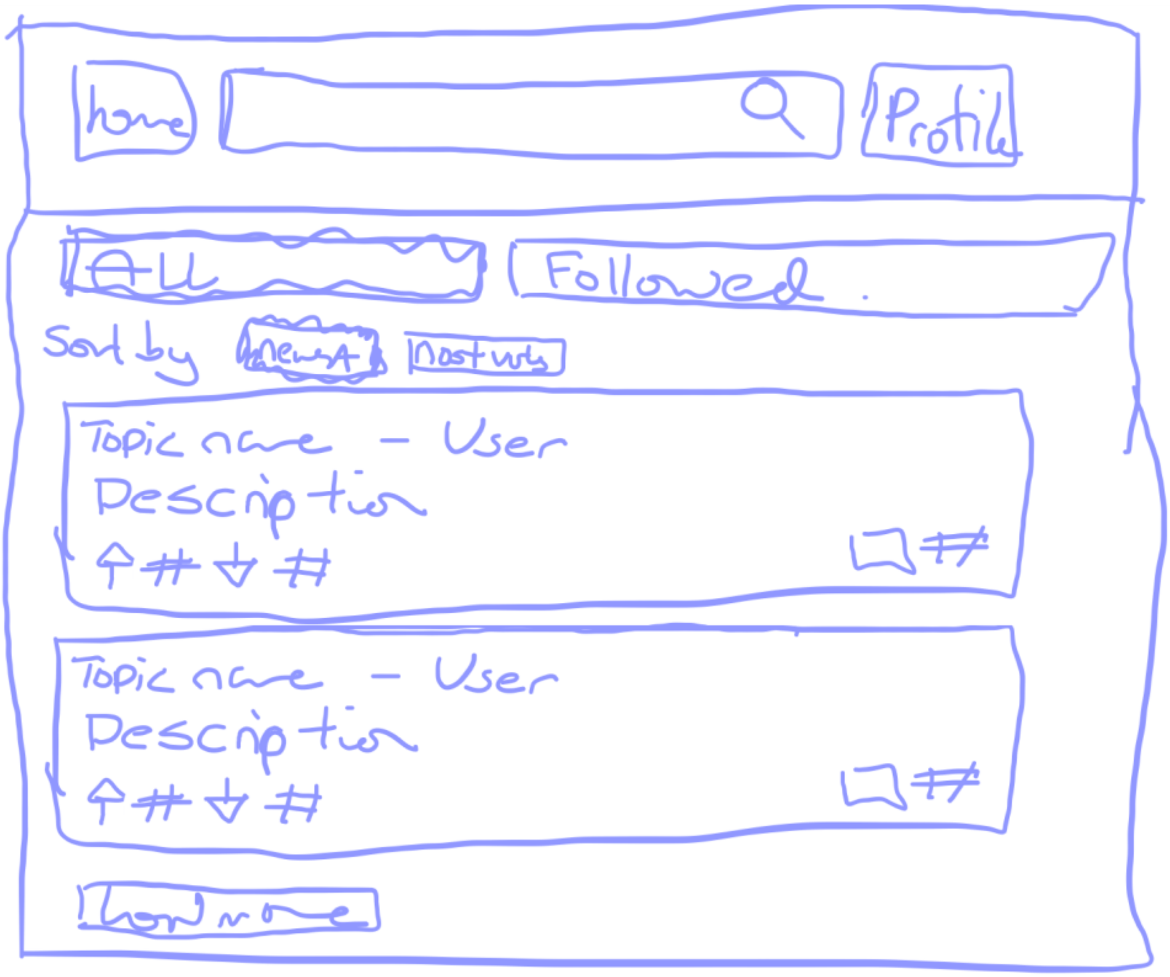
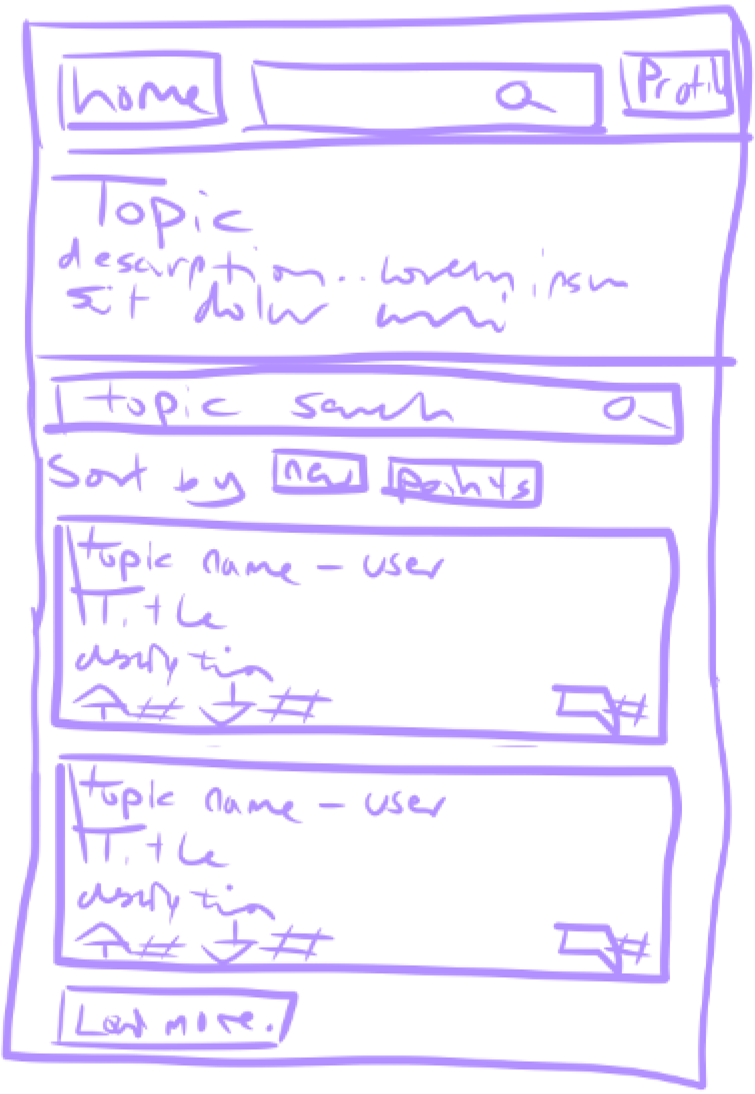
Section B - 0 words

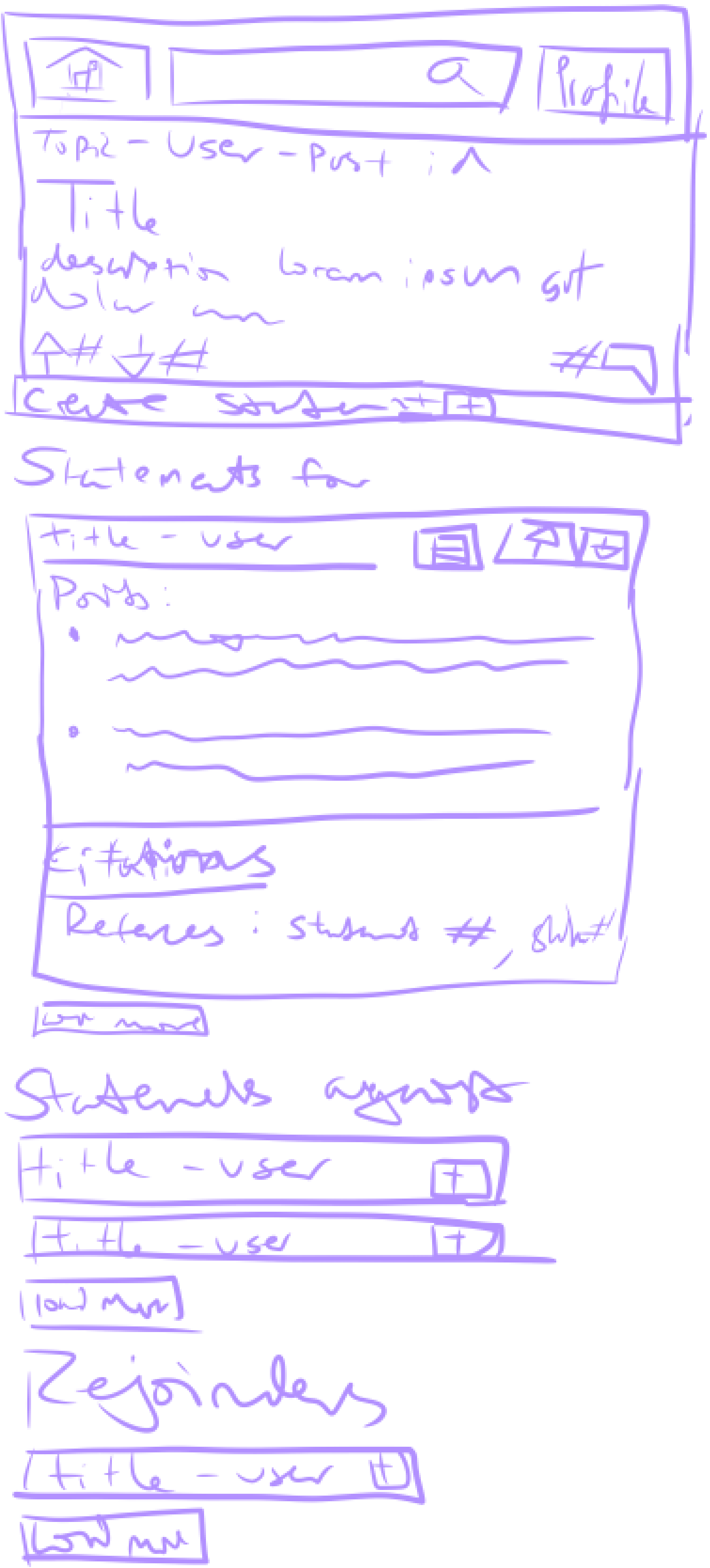
Interface



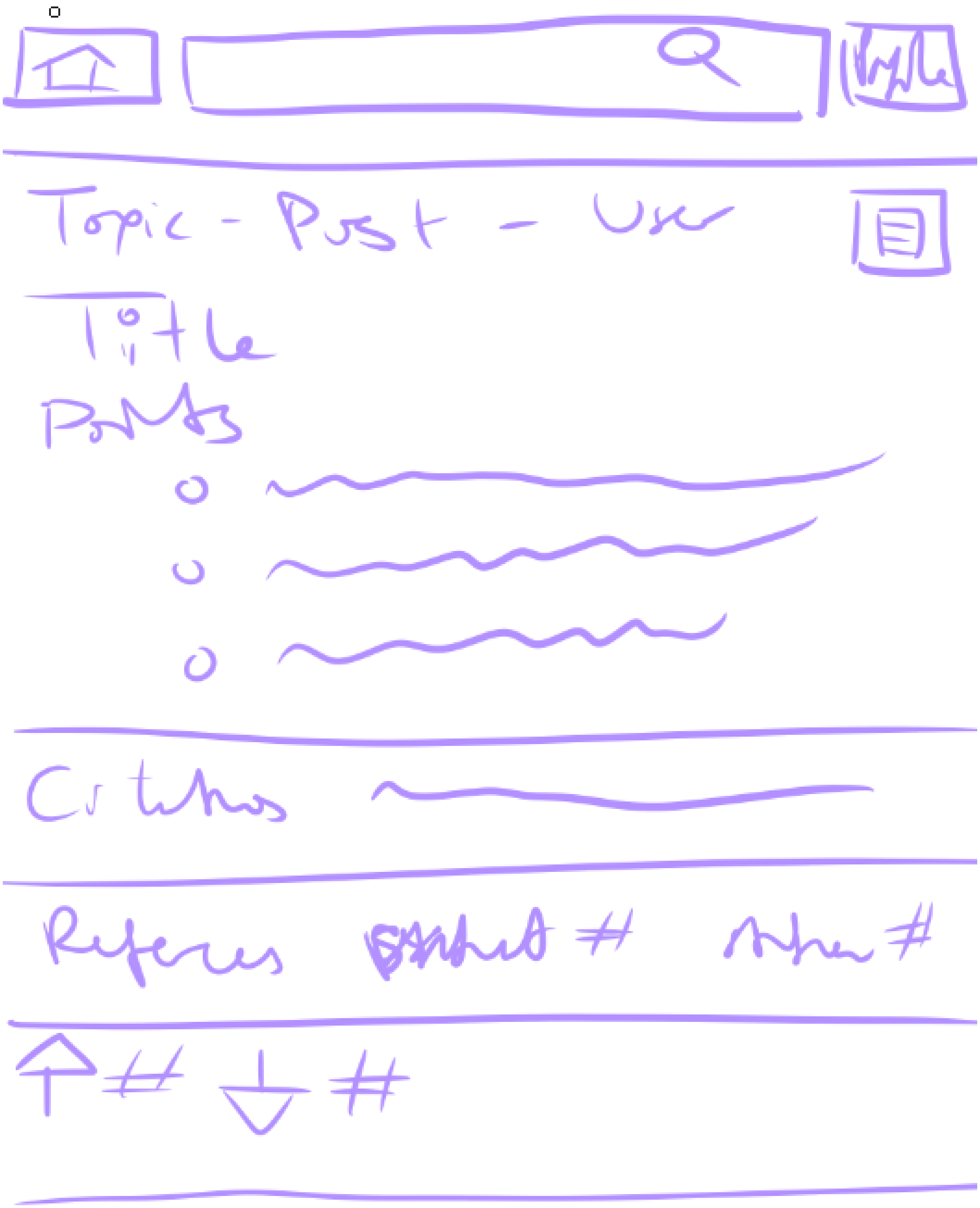
Home page when user is logged in.



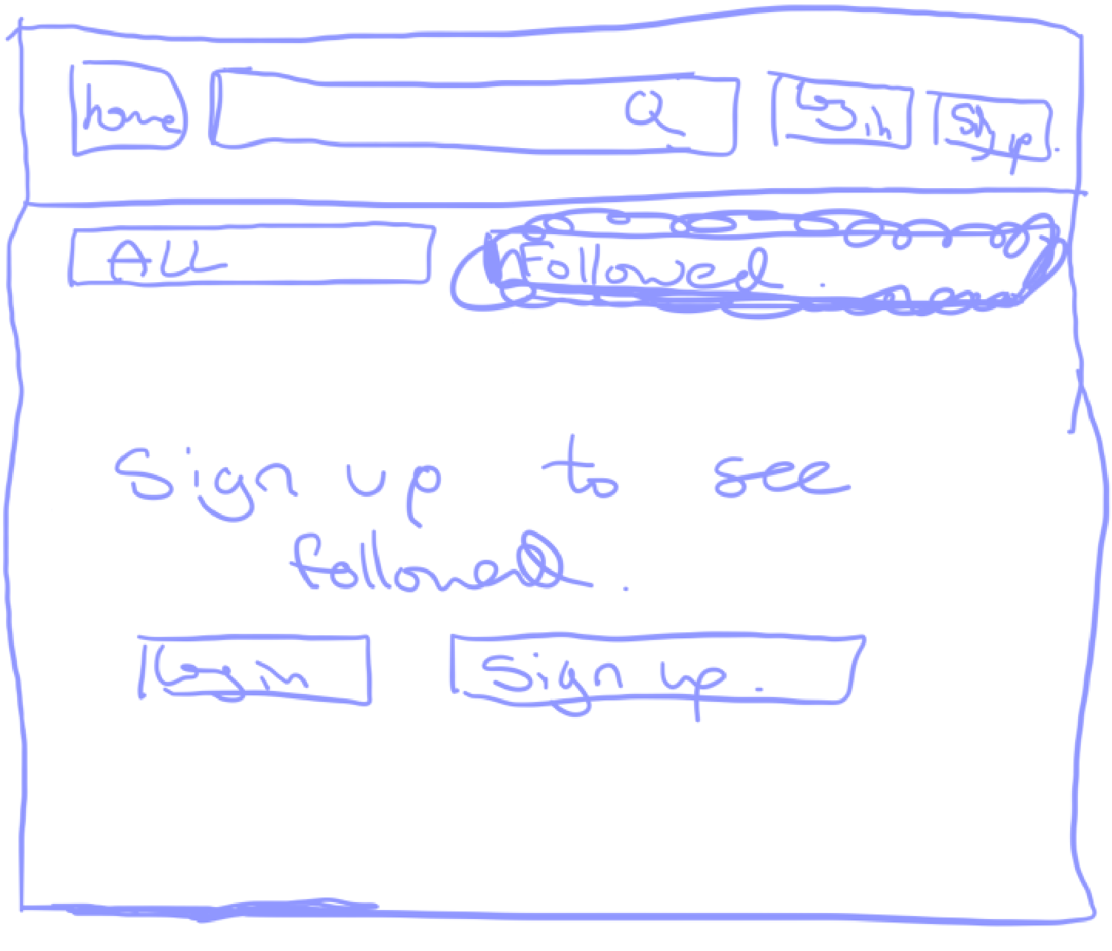
Topic page when user is logged in.



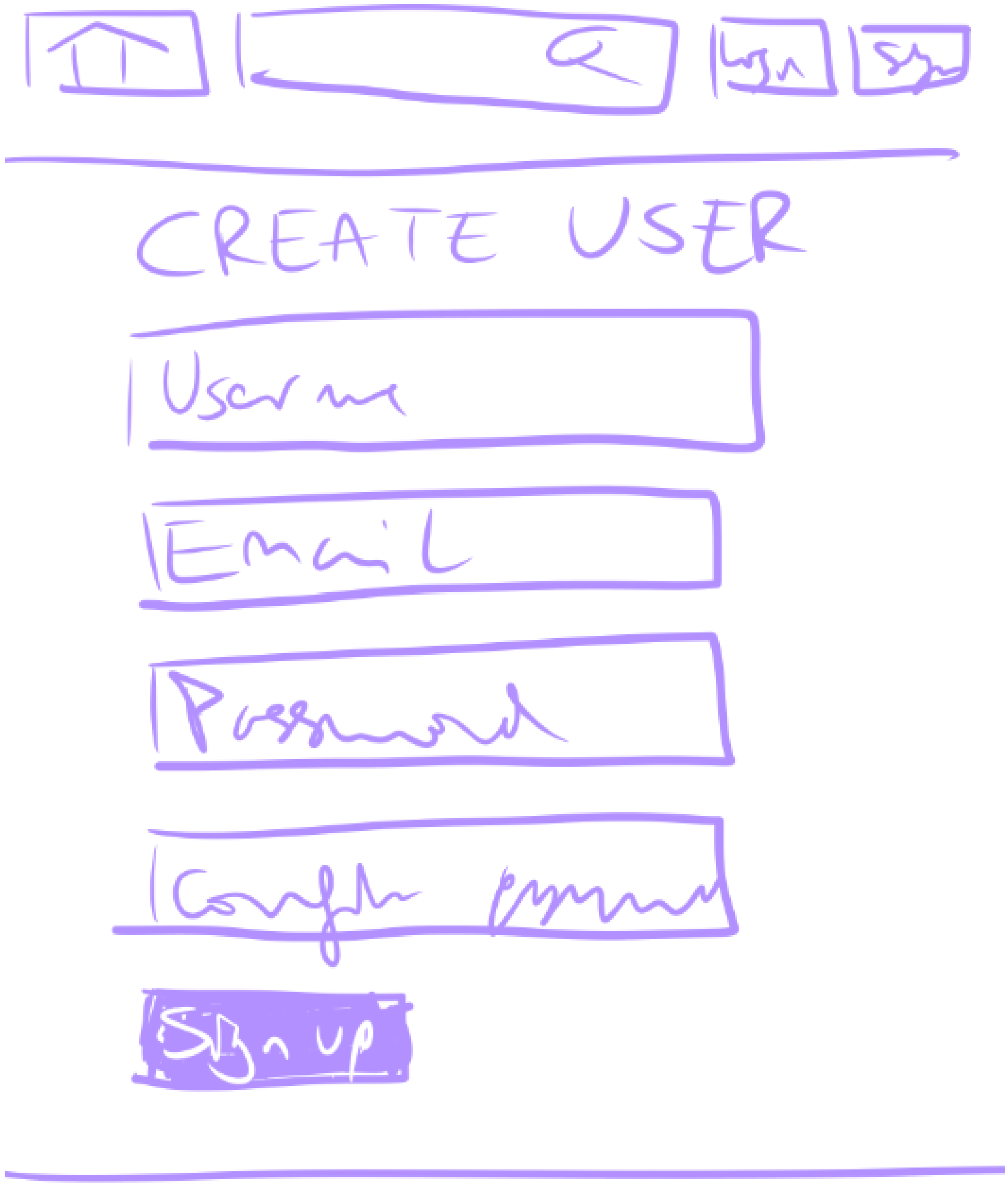
Debate page when user is logged in.



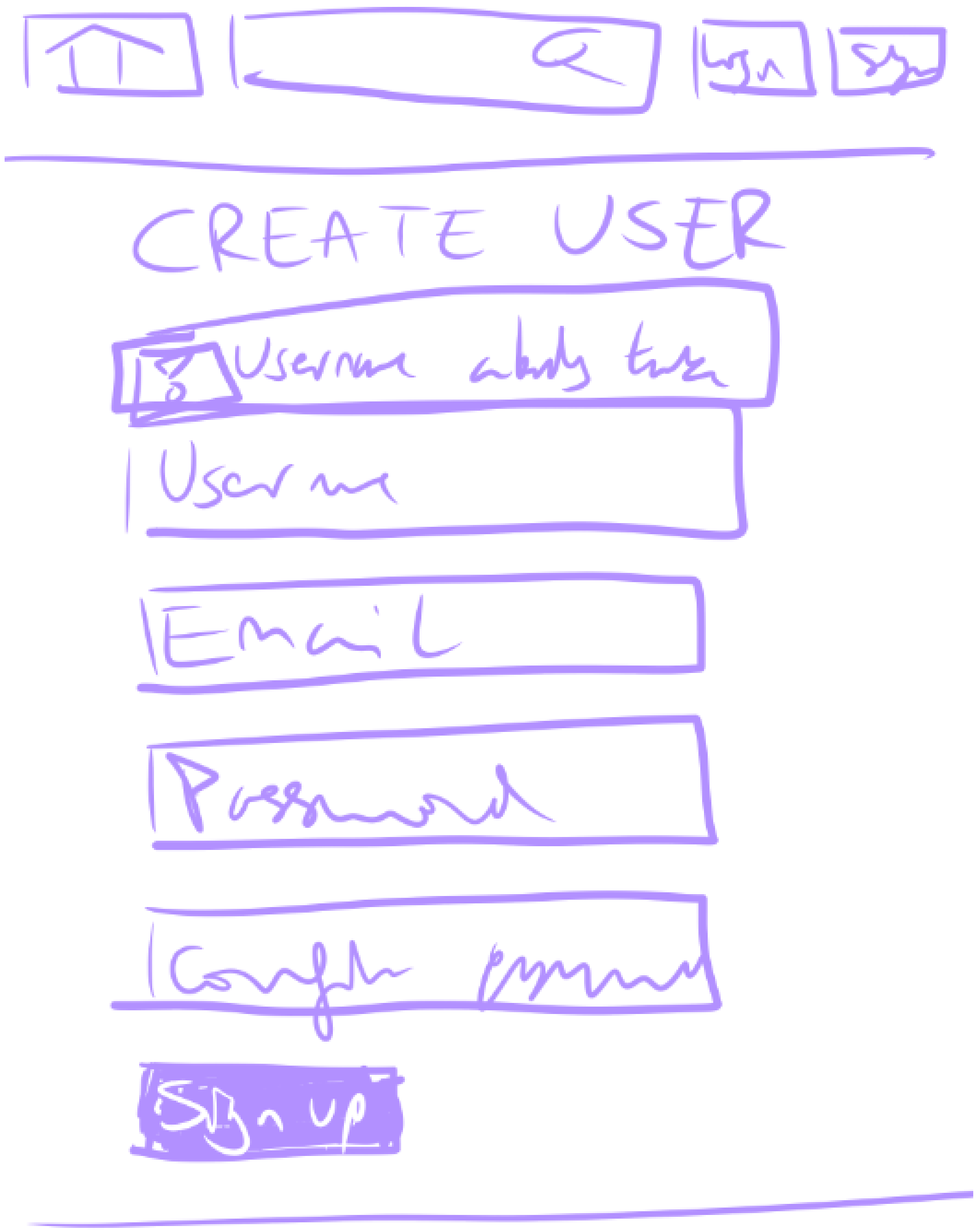
Point page when user is logged in.



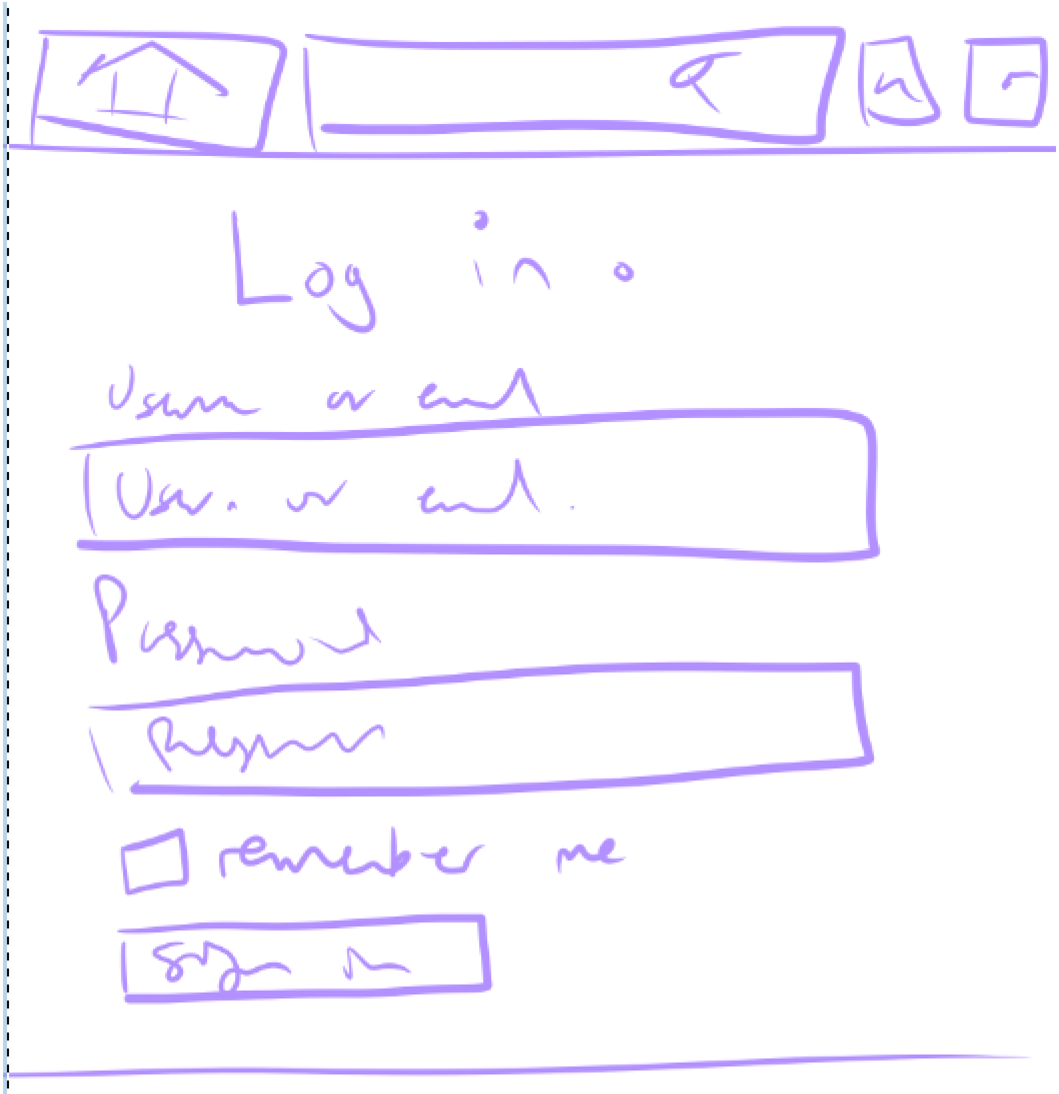
Home page when on the following tab and user is not logged in.



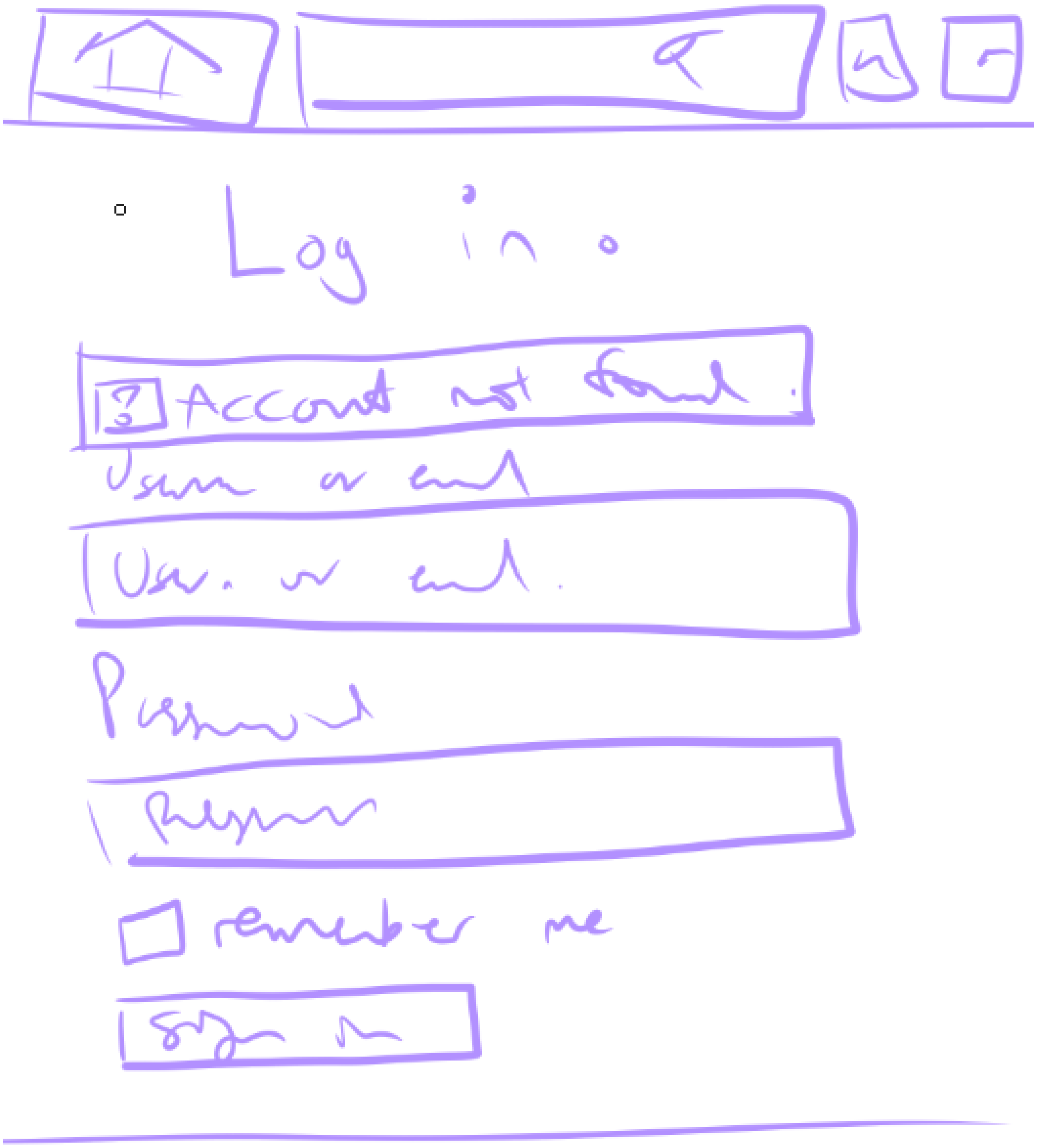
Create account page



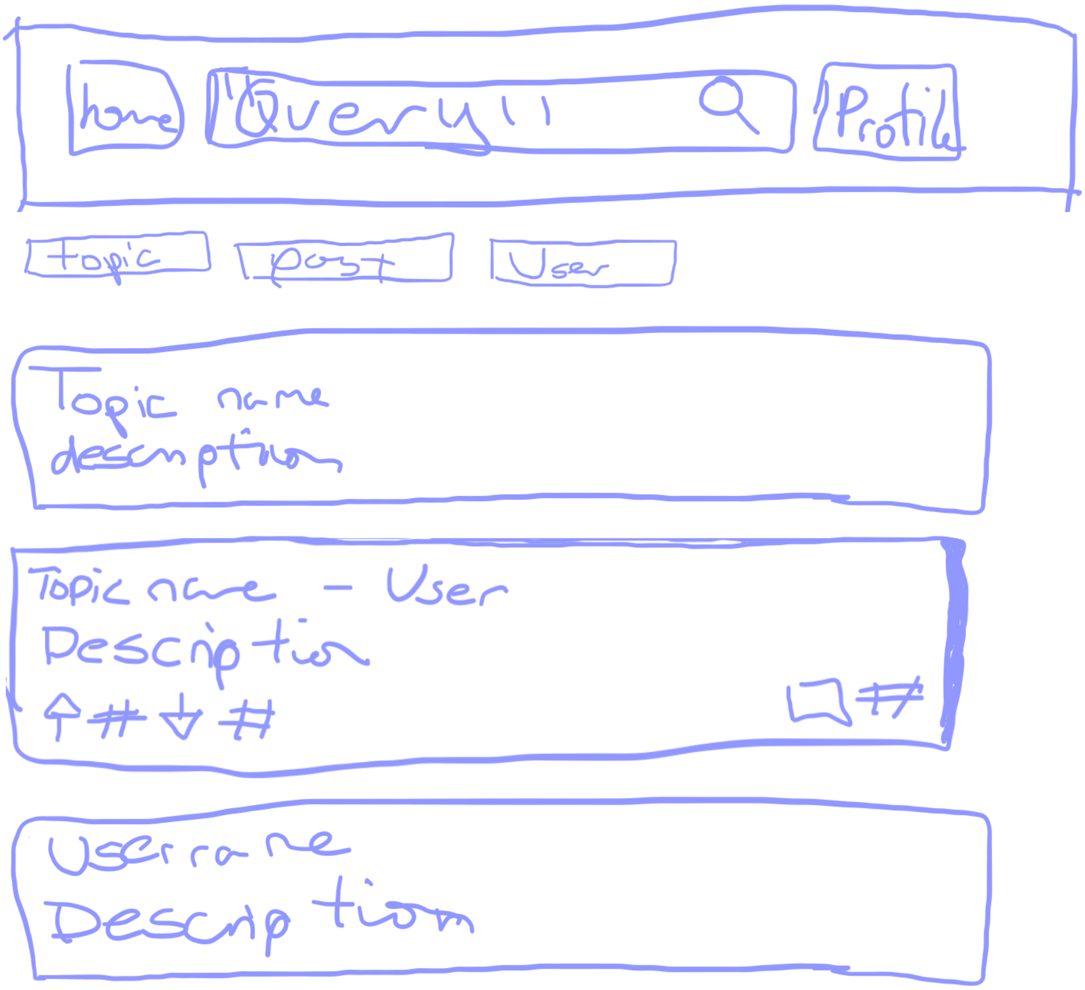
Create account page when there is an error in one of the fields


Log in.
Username or email
Username or email.
Password
Password
remember me
Sign in

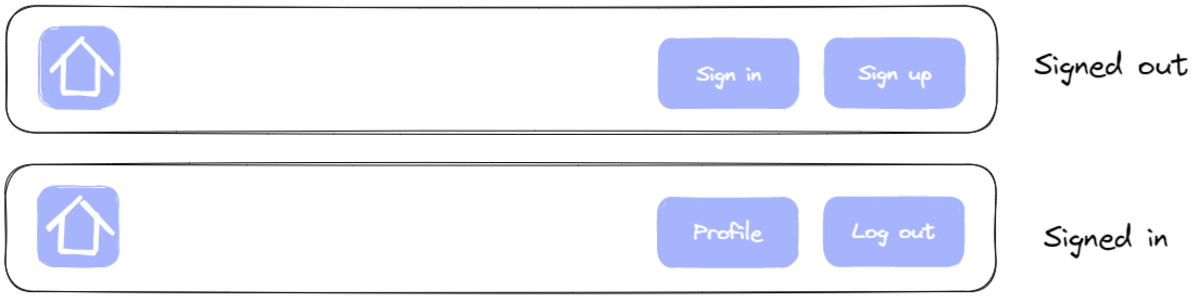
Log in page



Log in page when one of the fields has an error in them.

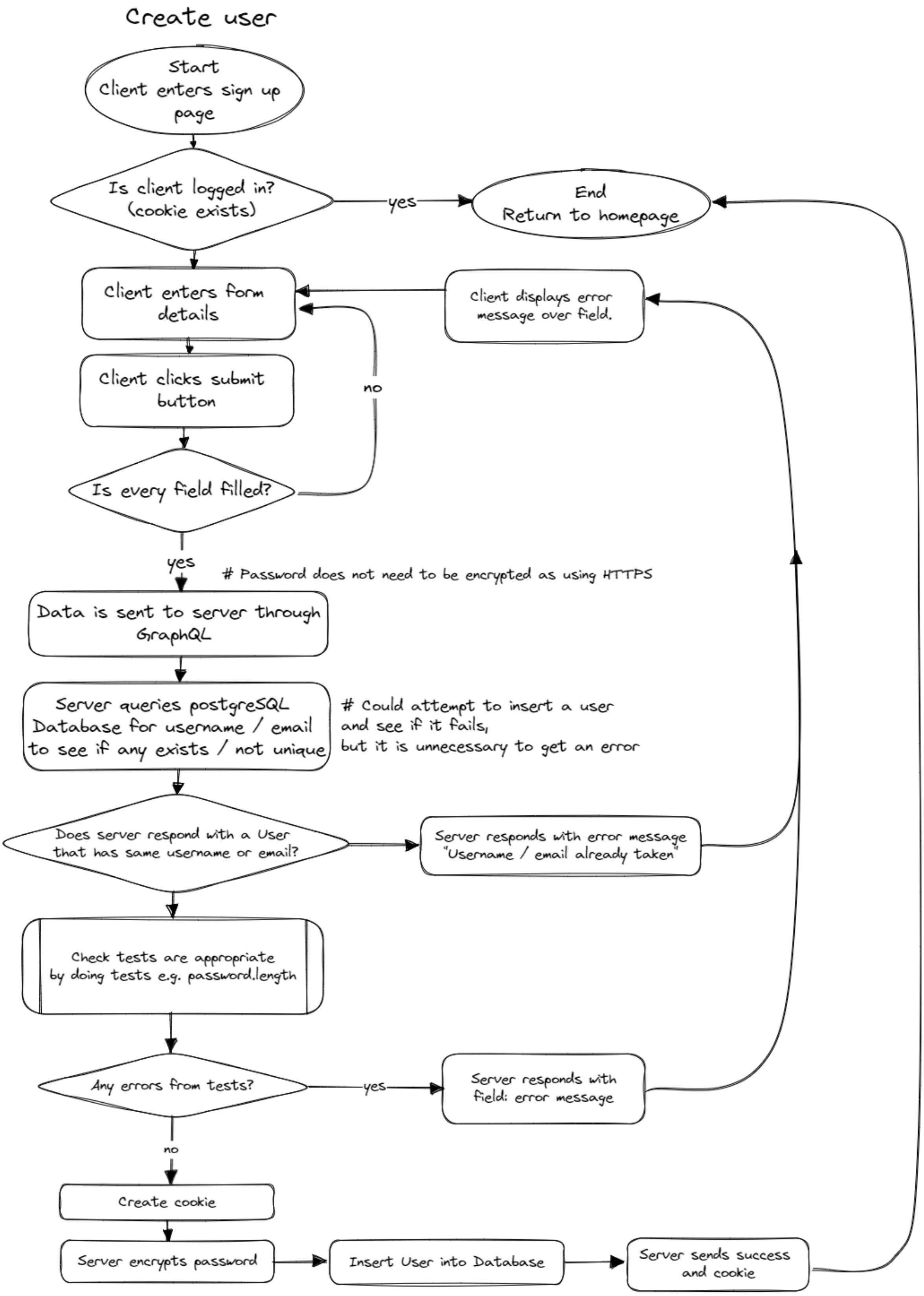


Search page (returns all unless specific is chosen)

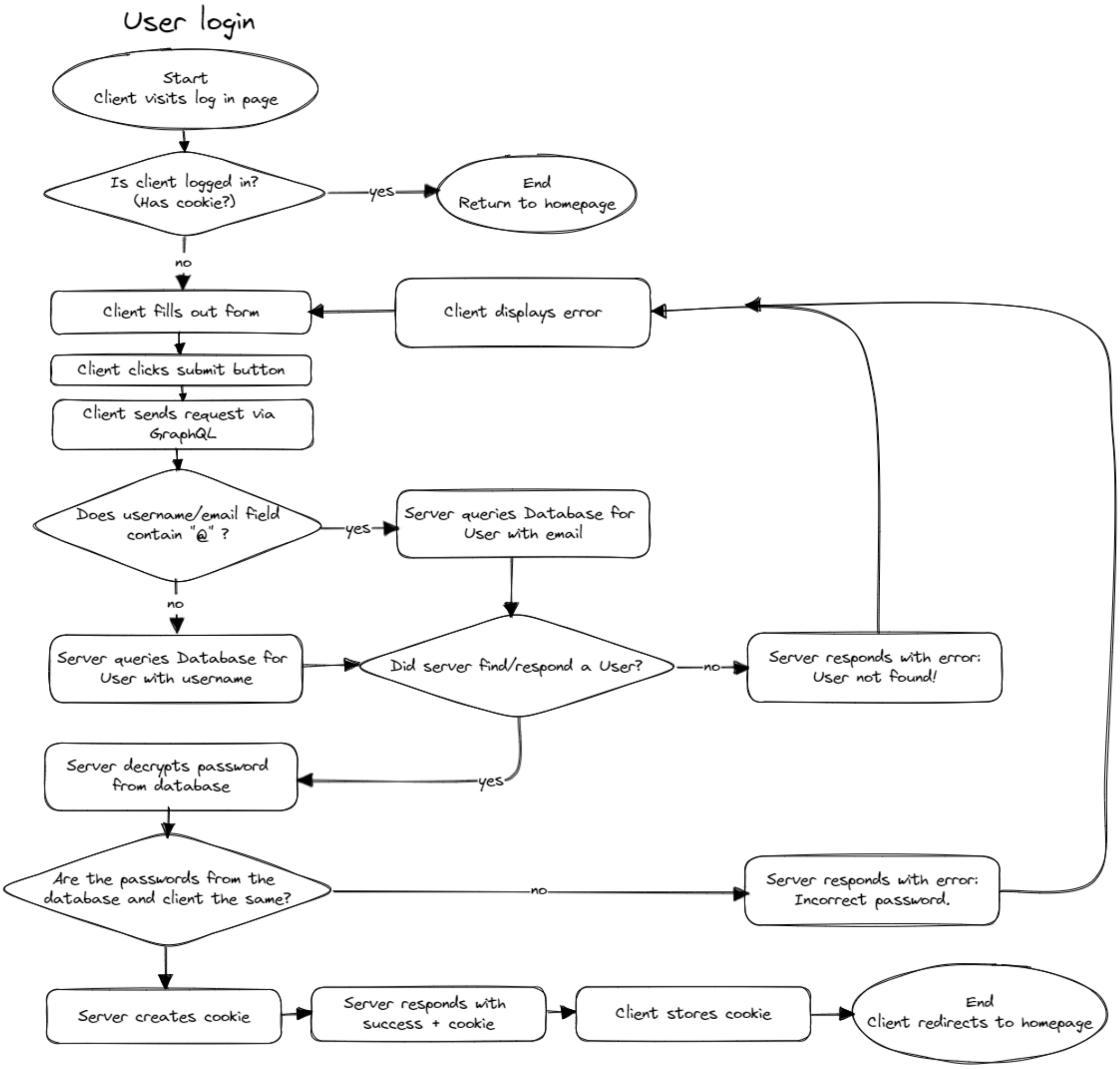


Updated navbar (removed search)

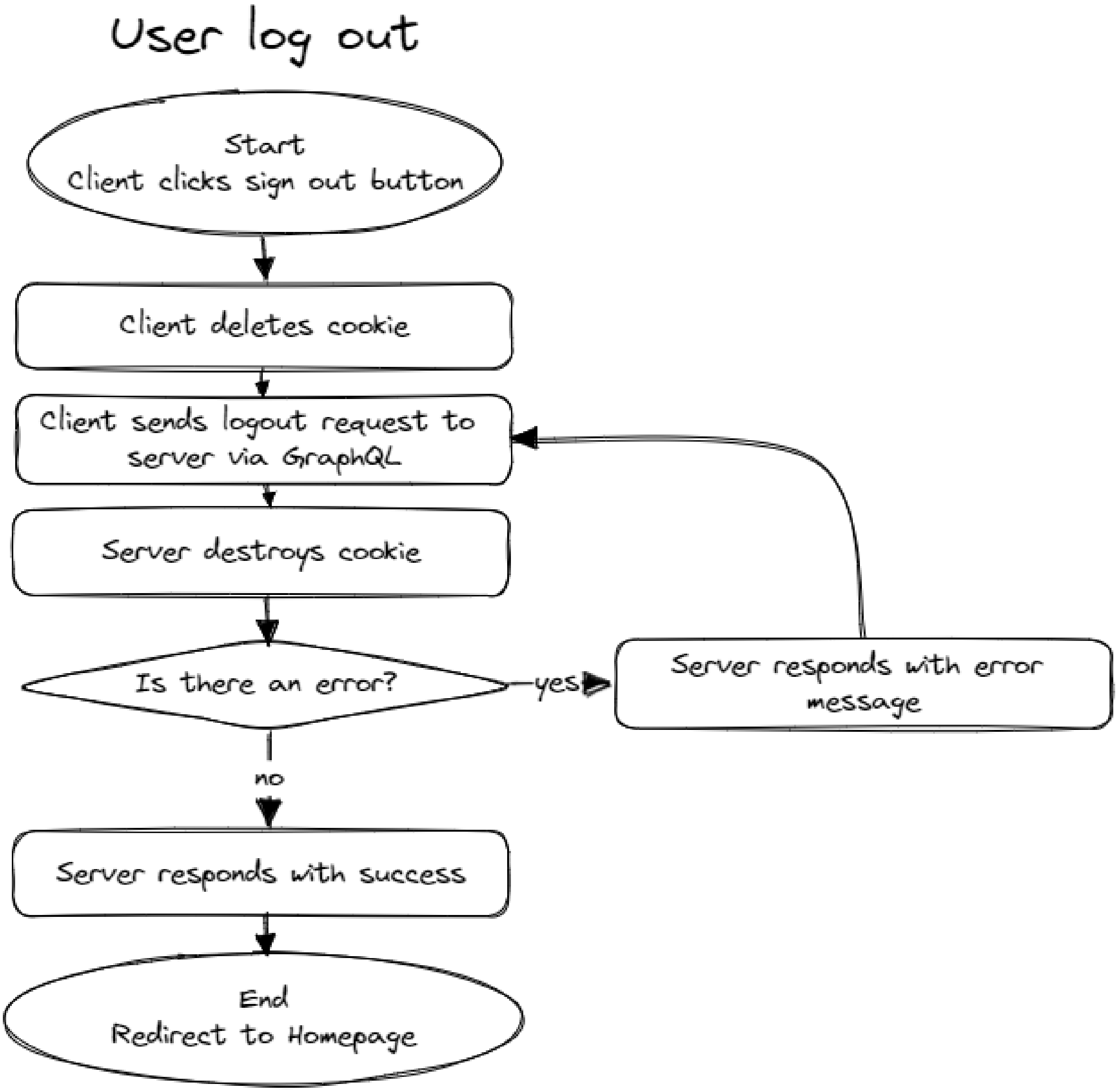
Flow charts:


Create user
Start
Client enters sign up page
Is client logged in? (cookie exists)
yes
End
Return to homepage
Client enters form details
Client displays error message over field.
Client clicks submit button
no
Is every field filled?
yes
# Password does not need to be encrypted as using HTTPS
Data is sent to server through GraphQL
Server queries postgreSQL Database for username / email to see if any exists / not unique
# Could attempt to insert a user and see if it fails, but it is unnecessary to get an error
Does server respond with a User that has same username or email?
Server responds with error message "Username / email already taken"
Check tests are appropriate by doing tests e.g. password.length
Any errors from tests?
yes
Server responds with field: error message
no
Create cookie
Server encrypts password
Insert User into Database
Server sends success and cookie

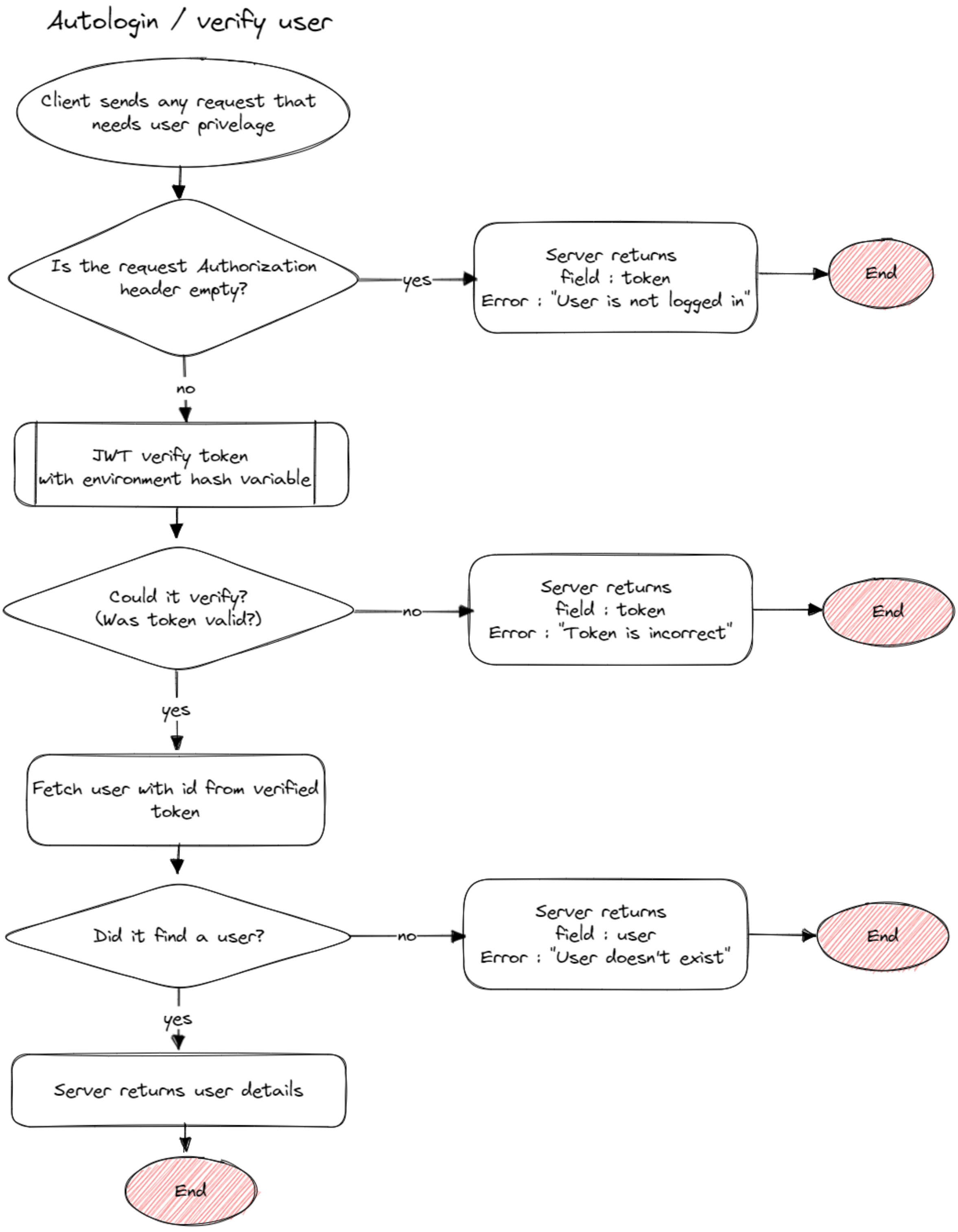
Creating a user (Signing up)


User login
Start
Client visits log in page
Is client logged in?
(Has cookie?)
yes
End
Return to homepage
no
Client fills out form
Client displays error
Client clicks submit button
Client sends request via GraphQL
Does username/email field contain "@" ?
yes
Server queries Database for User with email
no
Server queries Database for User with username
Did server find/respond a User?
no
Server responds with error: User not found!
yes
Server decrypts password from database
Are the passwords from the database and client the same?
no
Server responds with error: Incorrect password.
Server creates cookie
Server responds with success + cookie
Client stores cookie
End
Client redirects to homepage

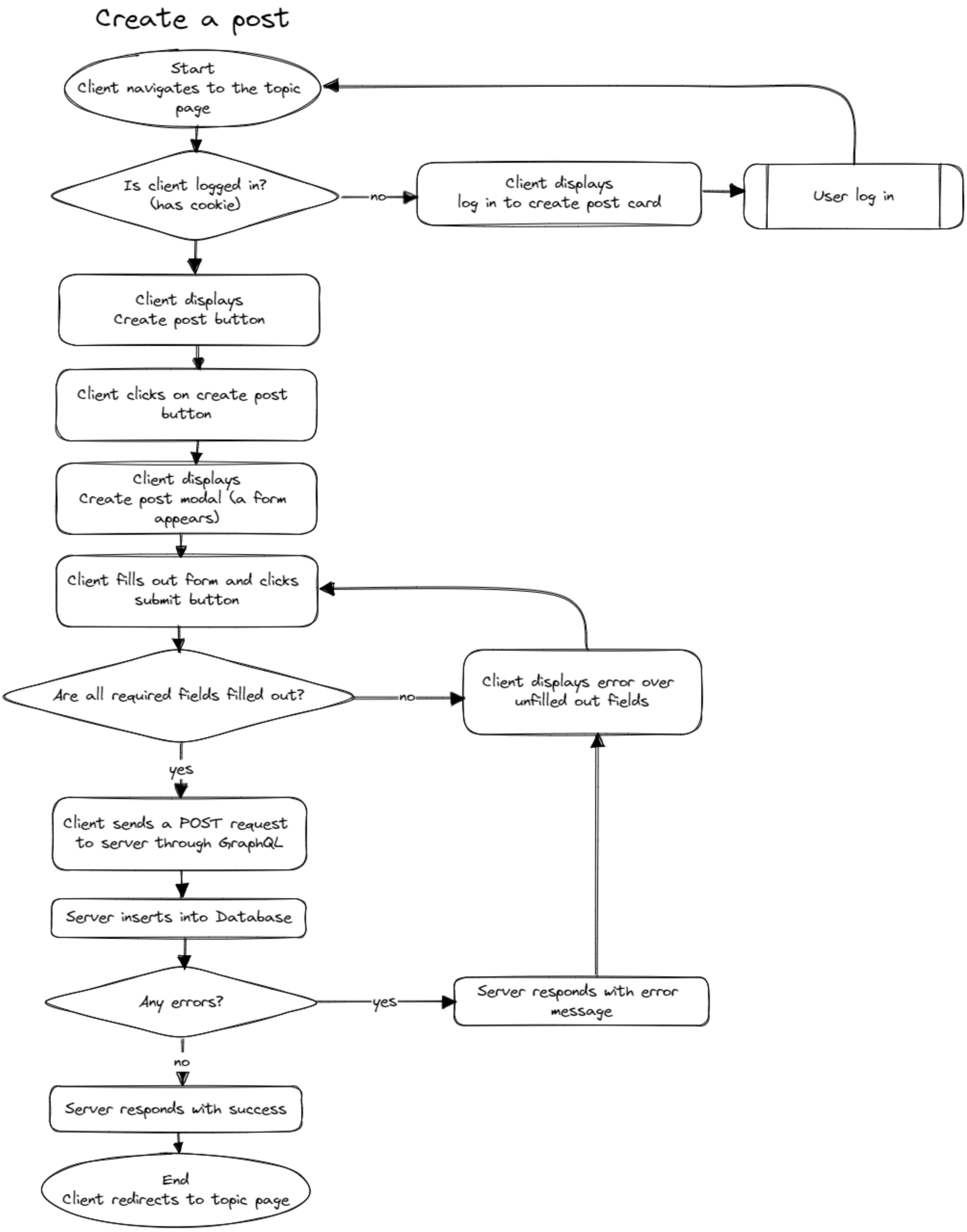
User login


User log out
Start
Client clicks sign out button
Client deletes cookie
Client sends logout request to server via GraphQL
Server destroys cookie
Is there an error?
yes
Server responds with error message
no
Server responds with success
End
Redirect to Homepage

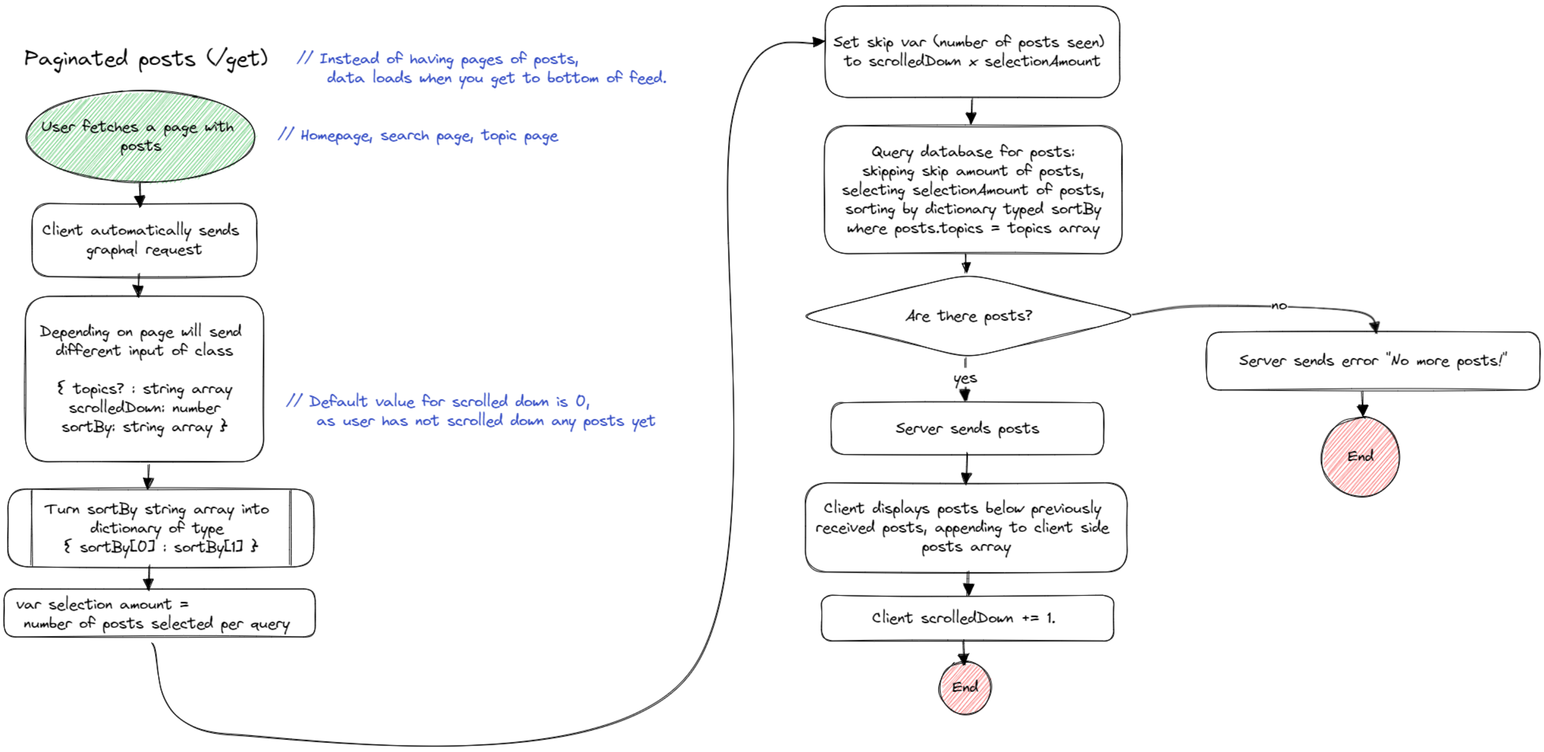
User log out


Autologin / verify user
Client sends any request that needs user privelage
Is the request Authorization header empty?
yes
Server returns field : token Error : "User is not logged in"
End
no
JWT verify token with environment hash variable
Could it verify? (Was token valid?)
no
Server returns field : token Error : "Token is incorrect"
End
yes
Fetch user with id from verified token
Did it find a user?
no
Server returns field : user Error : "User doesn't exist"
End
yes
Server returns user details
End

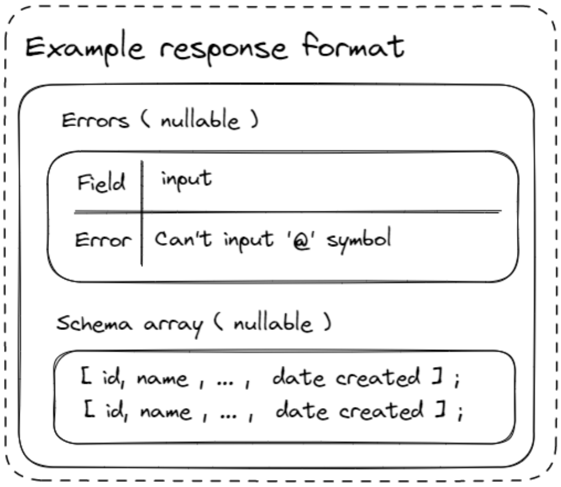
Automatic log in / verify user function


Create a post
Start
Client navigates to the topic page
Is client logged in?
(has cookie)
no
Client displays
log in to create post card
User log in
Client displays
Create post button
Client clicks on create post button
Client displays
Create post modal (a form appears)
Client fills out form and clicks submit button
Are all required fields filled out?
no
Client displays error over unfilled out fields
yes
Client sends a POST request to server through GraphQL
Server inserts into Database
Any errors?
yes
Server responds with error message
no
Server responds with success
End
Client redirects to topic page

Create a post

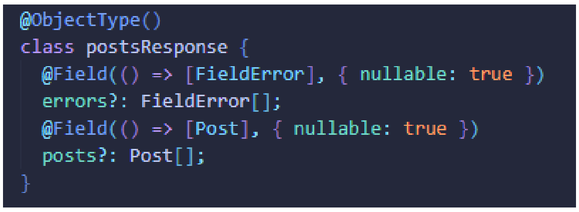

Paginated posts (/get)
// Instead of having pages of posts, data loads when you get to bottom of feed.
User fetches a page with posts
// Homepage, search page, topic page
Client automatically sends graphql request
Depending on page will send different input of class
{ topics? : string array
scrolledDown: number
sortBy: string array }
// Default value for scrolled down is 0, as user has not scrolled down any posts yet
Turn sortBy string array into dictionary of type { sortBy[0] : sortBy[1] }
var selection amount = number of posts selected per query
Set skip var (number of posts seen) to scrolledDown x selectionAmount
Query database for posts: skipping skip amount of posts, selecting selectionAmount of posts, sorting by dictionary typed sortBy where posts.topics = topics array
Are there posts?
no
Server sends error "No more posts!"
End
yes
Server sends posts
Client displays posts below previously received posts, appending to client side posts array
Client scrolledDown += 1.
End

Paginated fetching posts



Field	input
Error	Can't input '@' symbol

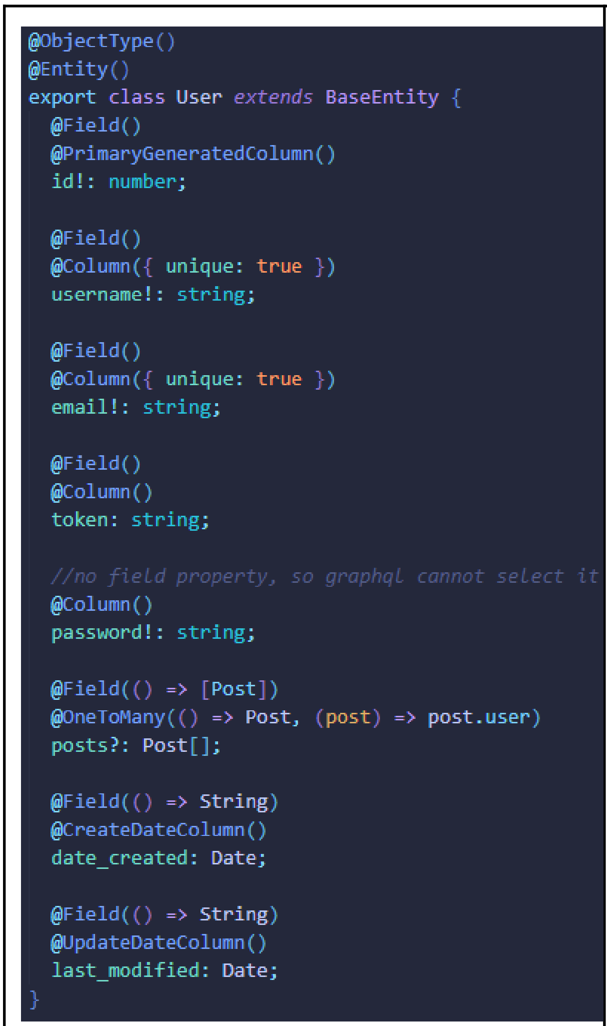
[id, name , ... , date created] ;
[id, name , ... , date created] ;



```
@ObjectType()
class postsResponse {
  @Field(() => [FieldError], { nullable: true })
  errors?: FieldError[];
  @Field(() => [Post], { nullable: true })
  posts?: Post[];
}
```

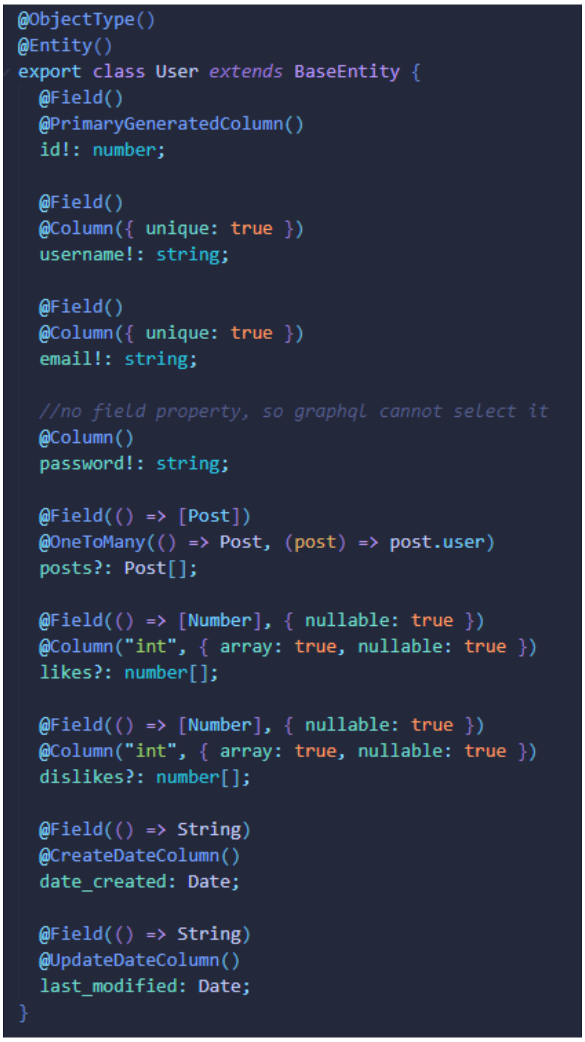
Generic response format diagram and example code snippet, for data response by server

Database schemas:



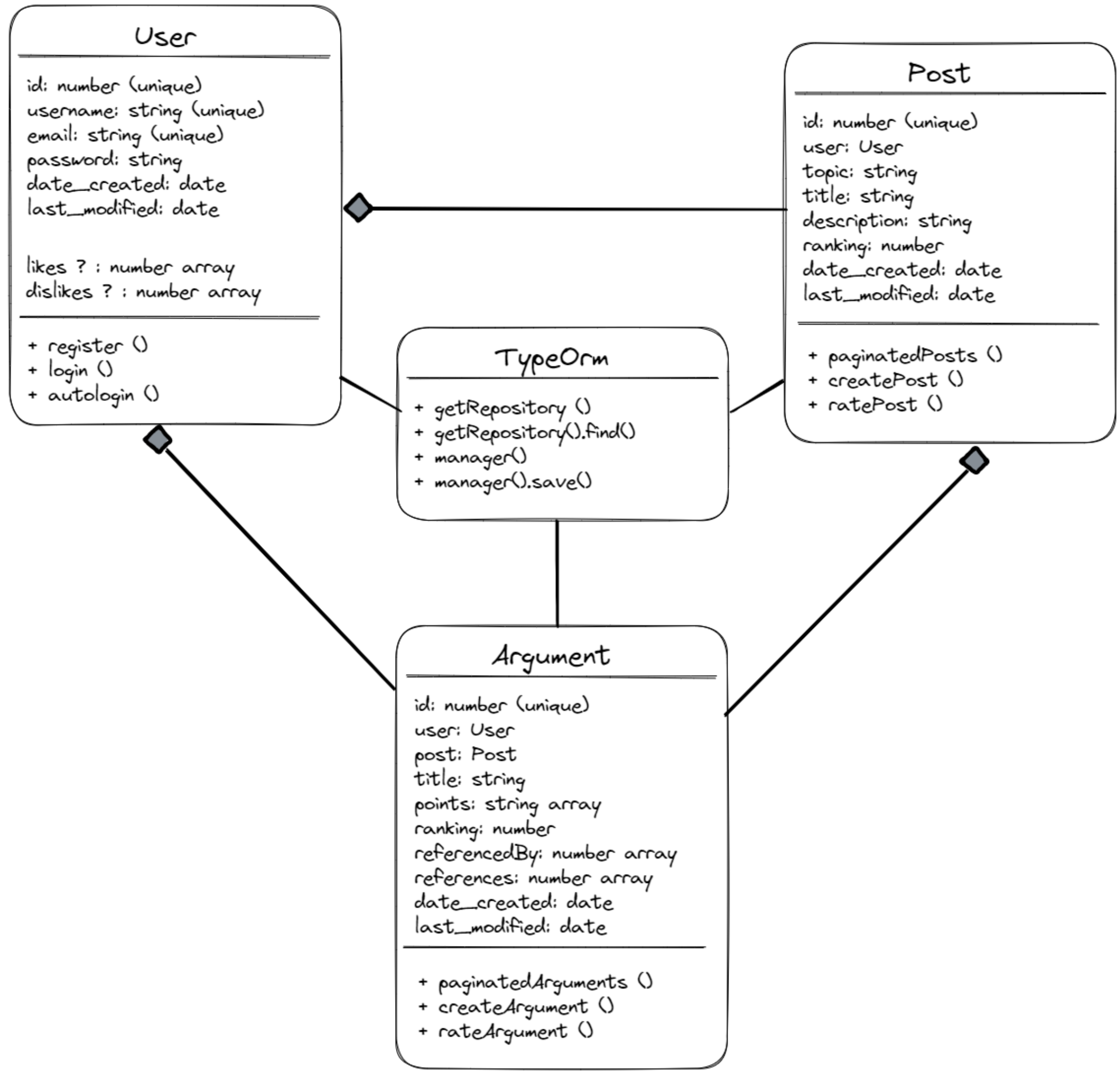
```
@ObjectType()
@Entity()
export class User extends BaseEntity {
  @Field()
  @PrimaryGeneratedColumn()
  id!: number;

  @Field()
  @Column({ unique: true })
  username!: string;

  @Field()
  @Column({ unique: true })
  email!: string;

  @Field()
  @Column()
  token: string;

  //no field property, so graphql cannot select it
  @Column()
  password!: string;

  @Field(() => [Post])
  @OneToMany(() => Post, (post) => post.user)
  posts?: Post[];

  @Field(() => String)
  @CreateDateColumn()
  date_created: Date;

  @Field(() => String)
  @UpdateDateColumn()
  last_modified: Date;
}
```

Old User Schema



```
@ObjectType()
@Entity()
export class User extends BaseEntity {
  @Field()
  @PrimaryGeneratedColumn()
  id!: number;

  @Field()
  @Column({ unique: true })
  username!: string;

  @Field()
  @Column({ unique: true })
  email!: string;

  //no field property, so graphql cannot select it
  @Column()
  password!: string;

  @Field(() => [Post])
  @OneToMany(() => Post, (post) => post.user)
  posts?: Post[];

  @Field(() => [Number], { nullable: true })
  @Column("int", { array: true, nullable: true })
  likes?: number[];

  @Field(() => [Number], { nullable: true })
  @Column("int", { array: true, nullable: true })
  dislikes?: number[];

  @Field(() => String)
  @CreateDateColumn()
  date_created: Date;

  @Field(() => String)
  @UpdateDateColumn()
  last_modified: Date;
}
```

New User Schema



UML Diagram of schemas

Test plan

Action to be tested	**Test method**	**Expected result**	**Success criteria**
User can create an account	Create an account using the form	That the account will be created and can query it from the database	1
Clients should be able to create and delete posts	Input correct data into form, submit and then delete it.	Should appear in application, Can then be deleted by only the owner user.	2

User should be able to reference other users arguments	Create an argument and include multiple posts id	The other arguments should be updated with new argument id and the post should include the references.	3
User should be able to reference other users arguments	Create an argument and include an id of a post that doesn't exist	The argument should be created but omit the ids of posts that don't exist	3
Post should be structured correctly	Create a post and arguments for a post of correct and incorrect types	Any erroneous data should not be imputed into database, but correct data should show up	4

Users should be able to give weight to posts	Give weight as like, unlike and go from like to dislike	Post rating should go:
		Like = +1
		Unlike = -1
		Like -> dislike = -2
			6

View a user's posts	Go to user's page	Should load all of users posts	7
User can save a topic	Save a topic	Should save the topic to the user in db, as long as topic is in whitelist	8
Accessible through a web browser	Run all of these tests on a computer that wasn't used for development.	All tests should work accordingly on another computer	9
You can view a users posts, relatively quickly	Include correct inputs and run the query	Load a certain amount of data each time.	7 and 11
Users can follow specific topics	Save some topics and go to the following tab while logged in.	Topics should automatically be included in the inputs for the search query.	8
Account data is secure	Create an account	Check that the database contains a hashed password	10

Should be able to browse a topic	Navigate to a topic page	Query should automatically include the topic	12
The website should look minimalistic	Show the client the website and ask him what he thinks	Client should be happy with the design	13