TalentTrade System Design

CSCC01 Summer 2024

Sprint 2

Table of Contents

Class Name: App	
Parent Class: None Subclass: None	
Responsibilities: ● Routing	Contributors: ● HomePage ● Signup ● Login ● ViewPost ● CreatePost ● ViewPostByCategory ● TopInCategory ● NotFound ● ViewProfile

Class Name: HomePage	
Parent Class: None Subclass None	
Responsibilities: ● Act as the landing page. ● Display“Most Needed Talents”, “Most Offered Talents”, “Most Popular Talents”, Make Popular Trades” ● User is directed to Make a Post page if user is logged in and clicks on Make a Post button ● User is directed to ViewPost page if user clicks on My Listings button	Contributors: ● Category

Class Name: Post	
Parent Class: None Subclass: None	
Responsibilities ● A reusable component for showing a post	Contributors

Class Name: LoginPage	
Parent Class: None Subclass: None	
Responsibilities ● Allows the user to log into their account ● Notifies the user if their input is incorrect ● Has a button to redirect the user to the sign up gape	Contributors ● Home page ● Sign up page

Class Name: CreatePost	
Parent Class: None Subclass: None	
Responsibilities ● Allow the user to create a post by inputting their need, offer, description and location ● Changes to be the edit post page when the edit button is clicked on a post, where the input fields are pre filled with the relevant information	Contributors ● Home page ● View listings

Class Name: ViewPostByCategory	
Parent Class: None Subclass: None	
Responsibilities ● Filters the posts by category to display the posts. The posts are filtered by needed talent, offered talent, or trade. ● Displays all the filtered posts where users can see the details of each post such as, what talent is needed, what is offered, and the description	Contributors: ● offerFilter ● FilterByLocation ● Post

Class Name: SignUpPage	
Parent Class: None Subclass: None	
Responsibilities ● Allows the user to sign up for an account ● Has input for their username, first name, last name, email, password and password confirmation ● Informs the user if they missed a field or if their password is not safe enough ● Redirects the user to the login page after creating an account	Contributors ● Login page

Class Name: TopBar	
Parent Class: None Subclass: None	
Responsibilities ● A reusable page header that has links to the user's account, create a post page, the home page and the search bar	Contributors ● HomePage ● ViewPost ● Create Post page

Class Name: Category	
Parent Class: None Subclass: None	
Responsibilities: ● A reusable category section to display the categories on the homepage. It displays the top 3 in either Most Needed Talent, Most Offered Talent, or Most Popular Trade. ● When the users click on the cards, it will bring users to the page that displays all the posts filtered by the need, offer or trade (ViewPostsByCategory page). ● When the more button is clicked, the user is redirected to the TopInCategory page	Contributors: ● ViewPostsByCategory ● TopInCategory

Class Name: ViewPost	
Parent Class: None Subclass: None	
Responsibilities: ● Display all posts by the current user ● Redirects to MakePost page if user clicks on the edit button ● Delete Post if user clicks on the delete post button	Contributors: ● Post

Class Name: TopInCategory	
Parent Class: none Subclass: none	
Responsibilities: ● The cards on this page are sorted from the most number of posts in the category to the least number of posts in the category (category can either be need, offer or trade). ● If a card is clicked, the user is redirected to the ViewPostsByCategory page	Contributors: ● ViewPostsByCategory

Class Name: NotFound	
Parent Class: none Subclass: none	
Responsibilities: ● A page for if users manage to navigate to somewhere that does not exist or to act as a temp page to link to	Contributors: ● HomePage

Class Name: ViewProfile	
Parent Class: none Subclass: none	
Responsibilities: ● Displays the profile for a given user ● Displays their name, bio, location, average rating and reviews ● If a user visits their own profile, the page will give them the ability to edit their profile and preview it	Contributors: ● ReviewCard

Class Name: reviewCard	
Parent Class: none Subclass: none	
Responsibilities: ● A reusable competent to display a review and the date it was created	Contributors:

Class Name: reviewDialog	
Parent Class: none Subclass: none	
Responsibilities: ● A component that opens a popup to allow users to give a rating and an optional review to another user	Contributors:

Class Name: filterByLocation	
Parent Class: None Subclass: None	
Responsibilities: ● A component that opens a dropdown that allows the user to filter posts based on their distance from a given location	Contributors:

Class Name: offerFilter	
Parent Class: None Subclass: None	
Responsibilities: ● A component that opens a dropdown that displays the different offered talents that the logged-in user has declared. It allows the user to filter the posts by the selected offered talents.	Contributors:

Class Name: ratings	
Parent Class: None Subclass: None	
Responsibilities: ● A component that opens a dropdown that allows the user to filter the posts by the rating of the user who posted it.	Contributors:

Class Name: user	
Parent Class: None Subclass: None	
Responsibilities: ● A reusable component to display a user when searched.	Contributors:

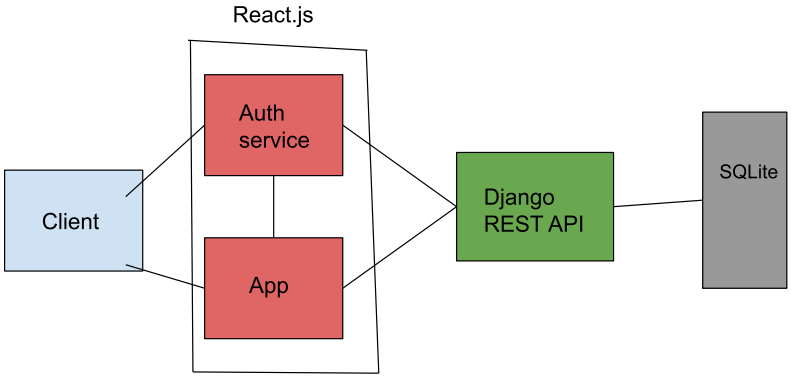
Class Name: searchUsers	
Parent Class: None Subclass: None	
Responsibilities: ● A page that displays the list of users that match the username searched in the search bar in TopBar	Contributors: ● User

The description of system interaction with the environment should indicate any dependencies or assumptions made about the operating environment of the system. E.g. OS, programming language compilers and virtual machine, DB's, network configuration, etc
We assume the server can run python and support the dependencies in requirements.txt.
We assume the client has a modern web browser and an internet connection.

Describe the architecture of the system that is the most abstract view of how your system is divided into components and how those components are interconnected. The architecture should be described with a diagram showing components and how they are related (or equivalent in words). Beware of designs based on a large number of components, they may signal a design that is overly complex.

System design:

We have a SQLite database that stores all of our data (users, posts, etc.), this is managed by a django REST API. Our client runs in a browser, using React.js for frontend. The frontend makes API requests to our Django backend. This API communicates using JSON. The front end can be split into the Authentication service, and the rest of the application.



The system decomposition should relate the system architecture to the detailed design, to identify the role of each component in the higher-level architectural view. Description of strategy for dealing with errors and exceptional cases (e.g. invalid user input, network or external system failure) that might arise in the use of the software. For anticipated errors and exceptions, a summary of how the software will respond in these situations

System decomposition:

Errors and exceptional cases: use try/catch to catch any potential errors, put reasonable effort into checks for invalid user input and give feedback to the user. On the user end, if an error occurs a log can be sent to the server for examination.