Agile Estimation of Cards

Project Name: Matchpoint

Feature ID: 4

Feature name: Friends page

Description: The feature allows the user to access and view their current friends, add friends, and send friend requests. The user is also able to connect with players he or she has previously played with. The social page also allows the user to view other friends' user profiles. Users can also set up chat groups with their friends in order to communicate and plan any future game.

Feature type: Customer

Estimated work effort (idea days): 7

Story points: 5

Planned iteration:1

Customer value (C, H, M, L): Critical

User: Player, Browser

Usage frequency (daily,weekly,monthly,other): Daily

Requirements uncertainly (H, M, L): Low

Technical uncertainly (H, M, L): Low

Dependencies w/other features: Homepage (after login, navigation bar)

Acceptance: The user can accept friend requests, add players in recent games, and the search box can find friends based on their names.

Back of card

Task	Ideal days	Groups	Assigned
Create user interfaces in friends pages for good layout	3	Development	Yuchen Zhu
Complete the basic functions of the user interface that have been implemented	4	Development	Yuchen Zhu

1-page tech research report

Security

The security of developing a website or open Web application is a key consideration during development. A simple error in the code can cause a lot of trouble. Attackers can obtain access rights to system data or user information based on the vulnerability, the privacy of users will be violated, and enterprises may suffer serious data leakage and economic losses.

For our Matchpoint web app, we can implement security in many ways. Many of the methods in the Django stack help us implement security as well. For content security, we can use Content Security policies (CSPS). CSPS is an additional layer of security that prevents malicious content from executing in a trusted web environment, helping to detect and mitigate attacks. Django can implement cross-site scripting protection (XSS). XSS attacks are usually implemented by storing malicious scripts in a database for retrieval and display to the user. Django templates escape certain characters that are particularly dangerous for HTML, but they're not foolproof. Django also has clickjacking protection. In supporting browsers, it is possible to prevent the rendering of a website in a frame. It can effectively prevent our users from being attacked while using the application. For connection security, we can use HTTPS for encryption. HTTPS enables two applications or devices in a network to exchange information confidentially and stably. The system using HTTPS can choose its own security parameters, which greatly improves the reliability and security of the system, prevents man-in-the-middle attacks, and ensures the safe transmission of data information. HTTPS is implemented in Django to encrypt communication between the client and the server. By enabling TSL/SSL/HTTPS, a web server can encrypt all communication traffic between a website and the browser, including authentication and other data sent via plain text. This secure connection ensures that our users are connected to the target server, thus protecting the data transfer. For data security, we can use cookies, which are small pieces of data sent by the server to the user's web browser. User data can be saved on the client side for simple caching and user identification. Secure cookies can only be transmitted over an HTTPS encrypted connection and cannot be easily stolen. Cross-site Request Forgery Protection (CSRF) is implemented in Django. CSRF attacks allow malicious users to perform actions with the credentials of other users without their knowledge or consent. CSRF protection can check the secret in each post request (using secure cookies). This ensures that a malicious user cannot simply place a form on the user's website and then have another logged in user inadvertently submit the form, thus protecting the user's information.

In conclusion, in terms of implementing network security, our web applications need to be implementing content security, connection security, and data security. While Django is well-defended against some common threats, including XSS and CSRF attacks, it's not without vulnerabilities. Therefore, developers must pay attention to every security detail on the web to avoid any security risks.

Development prep: tech stack/CI

These are screenshots of the frontend UI of the friends page and a git commit of the frontend code of the friends page uploaded to the git server.