SQL

SQL is used to load to and from the users database to save the state of all current users between sessions. There seems to be sane parameterization of SQL implemented in the Users class that prevents direct interpolation, and thus provides adequate protection against the typical injection patterns.

User Input Sanitation:

Any user has the ability to store information into HTML code on their profile page, so this is one of the primary vectors for attacks. There is enforcement of password security routines and there is some sanitation for all input fields (including: password, color, about me, and picture).

However, it is still possible to perform a script injection using the URL for the profile picture, for example
*<img src="http://url.to.file.which/not.exist" onerror=**[insert potentially malicious code]**;">*

Because there is some URL sanitation, it would not "save" the malicious input; in the sense that users who would view the user's profile would not be affected (which pr. It nonetheless allows the attacker to execute the script from their profile.

Using an external library to further sanitize URLs (such as bleach) and using a tag blocking content security policy would have been a preventative measure.

Content Security Policy

There is a lack of well-defined CSP-headers that defines the content security policy. There is some reference to it in *app.py* as a nonce is generated, but it's not actively used to define a policy anywhere.

Due to the low functionality of the login page it would be good practice to pre-emptively add a very restrictive policy there. The profile page should always be considered a potential vector, because it contains user inputs, and some kind of policy would be especially welcome there also. A CSP would at least be somewhat mitigative against some of the exploits that are achievable using the technique above.

It's also worth noting that there is particularily no CSP with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks. This concern is practically muted by the fact that the html of the "About" section is completely "wrapped", and it is not possible to add any HTML elements to it at all. This would include anything from deceiving hyperlinks, to malicious <iframe> tags and other common clickjacking strategies. This is a tradeoff between functionality and security, as users cannot add "safe" elements like pictures or videos. This policy implementation should nonetheless be pre-emptively added in case such features would be implemented in the future.

Authentication & Access Control:

There is solid session authentication and there is seemingly no way to forcefully change the current user. Judging by the POST headers, there are some anti-CSRF measures in place. The user alice may not simply make an edit request to bob's profile, as she would not share his token.

Although the buddy system is seemingly unfinished, its access control elements are implemented correctly and safely shield users' information from non-buddies.

Displaying the hash for your own password when accessing /users seems like a very unnecessary, if not dangerous, feature, even if proper authentication is in place and would cause problems *if* there was some way to get around it.