CS302 Python Project Indicative Marking Checklist 2019

Grade	Task/Feature Description	Done?
C	**Application runs following README instructions on Ubuntu Linux**	
	User can authenticate against the login server (using /api/ping)	
	User can see who is currently online (using /api/list_users)	
	User can generate a public/private keypair (and submit to /api/add_pubkey)	
	User can report connection info (to /api/report)	
	User can send and receive broadcasts to/from login server and other clients	
B-/B	Automatically refreshing page (or refreshing content) and/or notifications	
	Unicode support (including emojis)	
	(Good) auto content filtering via lists of blocked words or phrases	
	Good use of database(s)	
	Use of local encryption/hashing/data security (e.g. if passwords saved, they are encrypted/hashed)	
	User can send/receive private messages	
	User can search public broadcasts in some way (e.g. display only broadcasts from certain users, between certain times, that contain certain words ...)	
B/B+	Graceful error handling (No ugly 500 error pages)	
	Rate limiting on API	
	Private message interface (e.g. only show messages to and from a certain user, order by timestamp, mechanism to reply)	
	(Good) page templating, e.g. using Jinja2	
	Good inter-app security, including checking signatures and loginserver_records to ensure message authenticity	
A-/A	Retrieve and retransmit “offline” broadcasts and privatemessages (i.e. those sent while not online; implement and call /api/checkmessages)	
	Local favouriting/blocking of broadcasts/usernames/pubkeys	
	Markdown support in messages, including display of hotlinked external images (e.g. via `![A test image](https://……/image.png)`)	
	High standard of user experience (e.g. no lagging, awkward refreshing)	
	Attractive, cross-browser UI (e.g. looks the same in chrome/firefox)	
	2FA (Two factor authentication) e.g. for keeping private keys safe	
A/A+	Multiple sessions(users) supported simultaneously	
	Group conversations, including creating a group and inviting members (via a meta message), and sending and receiving messages	
	Receiving and transmitting meta messages for distributed meta information sharing (e.g. displaying other users favourite messages, blocking a message because your friend blocks it)	
	Saving/loading private data to the login server for seamless cross-client compatibility (encrypt/save/load/decrypt private data (e.g. keys/etc) to other student’s implementations; implement and call /api/add_privatedata, /api/get_privatedata)	
	Defence against injection attacks	

All inter-application features must be well-tested. In your README you must list at least two other students (their UPI) who have clients that your client works with.
You may test transmission of broadcasts and privatemessages with the login server, but this alone may not guarantee your mark (as you must also be able to receive broadcasts and privatemessages, and the login server will never transmit a message or a broadcast).

You must complete all the requirements for the C grade (minimum requirements, in bold) to be eligible for consideration for any higher grade.
Other grades will be awarded based on the number of completed features and their relative complexity, as indicated by the table. In general, completion of 3-4 features in each grade bracket will indicate eligibility for that (and below) grades. Items of a higher grade may be substituted for items of a lower grade. (I.e. if you completed all items in the A-/A and A/A+ grade section, it would not be necessary to complete items in the B-/B and B/B+ area).

Of course, how *well* a feature is implemented will also affect the final mark.

Please note that this marking checklist is indicative only, and may still change even after the final deadline if there are other features identified or if rebalancing the checklist is needed.
<u>You may implement things that are not on this checklist, consult with a TA to discuss what other features may be worth.</u>