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	✓
	User participates in network health checks by regularly calling /api/client_ping on other clients and by serving /api/client_ping requests	✓
B-/B	Automatically refreshing page (or refreshing content) and/or notifications	✓
	Unicode support (including emojis)	partial - can receive
	(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 ...)	partial support present
B/B+	Graceful error handling (No ugly 500 error pages)	✓
	Rate limiting on API (based on timeout)	✓
	Private message interface (e.g. only show messages to and from a certain user, order by timestamp, mechanism to reply)	partial - on database file
	(Good) page templating, e.g. using Jinja2	✓
	Good inter-app security, including checking signatures and loginserver_records to ensure message authenticity	
	Use of API keys with Login server instead of HTTP BASIC on all requests (i.e. use /api/load_new_apikey)	✓
	Manage user status i.e. online/busy/away, including the sending of 'offline' to /api/report on sign out/application close	✓
A-/A	Retrieve and retransmit "offline" broadcasts and privatemessages (i.e. those sent while not online; implement and call /api/checkmessages)	✓ [illegible]
	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, and sending and receiving messages	partial - invitation can be send but messages can't be decryted by recipient.
	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	✓

* I have extensively tested with fsan110, lobe655, utri092.

* Depending on what the others have implemented, I have tested accordingly.