COMPSYS302 Project B Indicative Marking Checklist – mwon724

Grade	Task/Feature Description	Done?
C	Application runs following README instructions on Linux	Yes
	User can log in	Yes
	User can see who is currently online	Yes
	User can see and edit their profile page, and see profiles for other users	Yes
	User can send, receive, and view messages+files with someone online	Yes
B/B-	Automatically refreshing page (or refreshing content) and/or notifications	Yes
	Good use of database(s)	Yes
	Use of encryption/hashing/data security within the application	
	Embedded media player for audio/video	Yes
	Displays confirmation of message receipt	
	Show user status (at a minimum, including online, away, offline)	Yes
	(Local) blacklisting of disruptive clients / unwanted users	Yes
	Logs out from the server upon application exit	Yes
B+/B	Communication of text formatting of messages	Yes[1]
	Rate Limiting	Yes
	Use of effective inter-app encryption/hashing/data security including handshake	
	Use of threading for communicating with login server regularly	Yes
	Search for profile pages and messages	Yes
	(Good) Page Templating	
	Modular and Pythonic code, including commenting and documentation	Yes[2]
	Group conversations	
A	Fallback P2P Networking if login server goes down	
	Multiple sessions (users) supported simultaneously	Yes
	Unit Testing	
	Nice user interface (preferably compatible cross-browser)	Yes[3]
	Offline Messaging (without central server)	
	2FA (Two Factor Authentication)	Yes
	Events co-ordination system	
	Fails graciously when interacting with substandard clients	Yes[4]

Footnotes:

[1] Markdown messages can currently be received and formatted correctly, and sent and formatted correctly. However, the frontend expects the user to format their message in markdown format manually and select to send the message as Markdown.
[2] Evidence of commenting and modular/pythonic code can be found throughout all the .py files in the project. Documentation for my APIs is found in the root folder, called 'fort-secure-chat-documentation-v2'.
[3] Tested on latest versions of Google Chrome, Firefox and Safari on Ubuntu.
[4] Evidence can be found in login.py with try-except clauses throughout the code.

Note: There is a known issue throughout Ubuntu lab computers where database access is restricted with "OperationalError: Database is locked". To stop this, open up SQLite Manager, delete an entry, and close SQLite Manager to stop this bug. I recommend deleting values out of user_credentials as there is minimal impact on the running of this program (credentials will be re-created once two-factor authentication is successful).