Project Review

This review is intended to provide feedback and insight from the developers of this flask-based application regarding the design and development issues that were encountered over the course of the project. The aim is to address not only the issues that arose in the creation of this prototype, but also to identify obstacles that would likely be faced if the product were to be further developed into a more advanced version.

Generally speaking, using flask as the basis of the project was not too difficult or problematic. It's a simple and easy web development framework, particularly in terms of setup and getting an application running. It fits the small-scale nature of the project because it is not inherently complicated. However, for more complex applications flask would not be ideal, as it lacks some broad features used in other web application development stacks.

The project implementation could be considered poor and ad-hoc. The interface was designed to work specifically with flaskr.py and uses some of flaskr's quirks. When entering data in certain web forms it is passed as a list object rather than a string; the interface automatically assumes it's getting a list even though it only wants a string. The interface turns the data into a string in a crude way (i.e. without checking if it is actually a list object). More type checking in the interface would be beneficial as any object type can be passed in Python which may cause errors.

Implementing the code would be much better if objects were used. Currently, the code works with methods only and it deals with lists and strings primarily. If objects were implemented, it would help with type issues and make it easier to pass information. Persistence is in the form of strings and integers stored in tables with sqlite3, and retrieved data is often in

the form of tuples, which are difficult to use directly. Using a more secure or a more object-friendly system would be beneficial, especially if objects were used in the implementation – a more robust database such as sqlAlchemy would be recommended.

Security was not a requirement during the initial prototype phase, but it should be considered if the product were to be progressed any further. Specifically, the database should be secured against direct viewing and information gathering by repeated querying of the database, and encryption of user information would be a necessity.

In some cases, the design of the modules and interfaces is noticeably fragile, and would likely be infeasible if the project were to be expanded. For example, the database is hard coded into userGroups.py; if the file were to change name, the functionality would break. It would be better if the code received either the database path or a connection to the database. Issues such as these were overlooked largely because they were easy to work around within the boundaries of this project, but are certainly poor design choices in general.

Overall, there are many significant weaknesses in the framework used for this project; if the developers were to attempt to progress the product any further, much of the current design would have to be overhauled to avoid these issues growing in severity. However, for the scope imposed by the nature of the project (i.e. as a beginner-level software engineering exercise), the flask framework was adequate, as it was simple enough to understand, and also provided a very solid basis for the application in the form of flaskr.