You are creating packages that you must be able to hand to someone else, and they set up and install the whole thing.

Some general comments on Project 1:

• It needs to be much easier for a user or system administrator to install and test each step of the installation, such as:
  (a) all system components need to display and continually whether information is being sent, and what information is being sent,
  (b) an initial data packet should get sent immediately upon starting the App, and
  (c) it needs to be easier to set the sampling interval down to 5 s or so. The user or system administrator should not have to wait 5 minutes to see data pass through.
• It is simple to enter a testing mode, or to send a new data packet, to test components.

Need some way to thoroughly test that all three components are working, and let each component work independently as well as possible.

• Using pythonanywhere for the server, but ix for the mysql database, increases the complexity, and fragility, of the system. It makes more sense to put both of these subsystems—the mysql server and the flask server—on just one server.

• The installation document says "The pythonanywhere site is now successfully hosted and you can navigate to that site on any device." But navigating to that site does not work. It causes an "Internal Server Error". It turns out that this is actually an indicator that something is working. But a better indicator would be something like "geolocation server up. mysql database not working yet." The user needs to determine their progress after every step, and needs as much as information that the system can possibly provide to understand progress, and troubleshoot roadblocks.

• A number of errors appeared in the server log that could have been checked-for in the code, such as: "ReferenceError: Can't find variable: google"
  Rather than effectively requiring the user to do the troubleshooting.
  One rule of thumb is, effectively, to not rely on anyone for anything; in other words, to not trust that external services will always work exactly the same; in other words, any interaction with an external service should be permitted to fail, and your system responds appropriately (even if just by letting the user know what happened).

Better testing is needed.
• Does not appear to have been tested on a Mac. Windows was not specified as target installation machine. In general a developer should test on a range of platforms that are likely to be used.
• It does not appear as if a new-install or "clean" install (effectively on a machine with no project-specific dependencies set up in advance) was tested, or at least not on a range of different machines.
• The system should provide the user more immediate and transparent feedback with regards to the current status of the system, such as whether a connection was made, if data are being sent, if the location information is correct, and so on.
The information was all there but embedded in different server logs, and in the programmers interpreting system behaviors (such as the appearance or lack of appearance of information) to mean certain things.

- The browser read "Could not retrieve your location - 12:47:15" six minutes after starting it up. It would be helpful to know sooner that something is not working.

- Many of the documents and artifacts need better identifying information. I appreciate that an effort was clearly made to do this, but please see "How to Submit" Item 2. "What is this thing?" needs to be better-answered. For example, with the User Documentation and the Install instructions, more information is needed to explain what this document is, such as what system it is describing. And "Group 1" means little without some context.

- "When submitting a system for someone to use or evaluate, include (and document) sample data files that help to illustrate how the program works, and all of its features."

Permitted in Project 2:
  Flask
  venv
  pythonanywhere but I don't recommend it

But understand that all of the custom configuration you create for yourself during development must be recreated by a naive system administrator to install your system.

Every single thing you do to customize or set up your fragile environment must be done at install time. Each also shortens the number of months or years that the install would actually work.

You must manage your dependencies.

Your submissions are treated as software packages to be fully installed, not evidence that you built something.

Your software must install.

Suggestions to Improve

Suggestion #1: Keep it simple. Use a few, well-established technologies, and as few black boxes as possible. For example: ix rather than pythonanywhere.

Suggestion #2: Provide lots of checks during the installation.
"Fail fast. Fail Clean."
For example: for a pathname of user directory - Don't hard-code them into shell scripts. Better is to use a configuration file. If you do hard code pathnames, provide the user with a means of confirming they are correct, or have your script confirm they are correct and, if they are not, fail quickly and with clear error message.

Suggestion #3: Have a novice user (someone from another group, ideally someone not using all of the same technologies) try to do your install with no help whatsoever.

New Ideas for Project 2?
New Requirements? New system?
- Limit dependencies.
- More user interaction.
• Test the components better.
• What requirements would support testing the components?
• What system design details would make it easier to test the components?

• Every component must work independently with injected sample data that are automatically provided in the code for each module itself.

Perhaps Redo Project 1 but in a manner that addresses these issues. Generate requirements that emphasize:
  • minimizing black-box dependencies.
  • components reporting on status and failures in real time.
  • the installation process.
  • testing of the installation process as a first-class problem.