**Stolen Object Tracker: Continued Development**

**Server (whole):**

The server is written in the Go programming language. Many of our group members used liteide to work on the server. You can also use a text editor and compile via command line.

There are various improvements that could be made to the server at this time.

Not all requests that are handled respond to the sender of the request. Adding responses to all requests would improve the reliability of the system. To see an example of how the request/response system works, look in the webServer.go at the login handler. The webServer sends a request to the database and, depending on the response, will log the user in or tell the user the entered credentials are invalid.

The server could use a logging system. The easiest place to start would be logging all requests that are sent and the responses to them. All requests pass through the central server, so that would be a good place to capture them. Additionally, some events between the deviceHub and laptopHub, and between the deviceHub and gpsHub, could be worth logging.

**Database Controller:**

**Web Server:**

The web server needs to communicate with the website to get the Geogram’s PIN upon registration. At the time being, the service uses the Geogram’s default PIN, 1234. Security could be improved by allowing users to specify their own PIN.

The web server should encrypt traffic to and from web connections. At this time, information is sent in plaintext. This is a problem we hoped to tackle, but ran out of time beforehand.

**DeviceHub:**

**Website:**

**Windows Service:**

The Windows service was developed in visual studio. Opening the solution in Visual Studio to continue development.

The Windows service is left in a very good state. The only functionality that we were not able to implement was connecting to the unprotected Wi-Fi automatically to increase the likelihood of finding a stolen laptop.

The key-logger is also in a good state. One thing that could be done is to handle certain key combinations that are used as shortcuts, such as ctrl + s to save. Otherwise, most keys and key combos are captured.

To debug the service or key-logger, run them as you normally would. Use the “Attach to Process…” option on Visual Studio.