2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW) | 978-1-6654-4057-8/20/$31.00 ©2021 IEEE | DOI: 10.1109/VRW52623.2021.00183

# RED: A Real-Time Datalogging Toolkit for Remote Experiments

Sam Adeniyi*
University of Minnesota

Evan Suma Rosenberg
University of Minnesota

Jerald Thomas
University of Minnesota

**ABSTRACT**

The ability to conduct experiments on virtual reality systems has become increasingly compelling as the world continues to migrate towards remote research, affecting the feasibility of conducting in-person studies with human participants. The Remote Experiment Datalogger (RED) Toolkit is an open-source library designed to simplify the administration of remote experiments requiring continuous real-time data collection. Our design consists of a REST server, implemented using the Flask framework, and a client API for transparent integration with multiple game engines. We foresee the RED Toolkit serving as a building block for the handling of future remote experiments across a multitude of circumstances.

**Index Terms:** Human-centered computing—Human computer interaction (HCI)—HCI design and evaluation methods—User studies; Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Virtual reality;

## 1 INTRODUCTION

Conducting remote experiments has been an increasingly popular approach over the past decade, with some researchers applying crowd sourcing systems such as online survey websites [4]. Although this allows for large numbers of responses, the accuracy and dependability of this information is often in question. Unsurprisingly, researchers in the virtual reality (VR) community have rarely been able to take advantage of these crowdsourcing solutions and have largely relied on in-lab testing to conduct experiments. This has been historically due to the lack of access to head-mounted display (HMD) devices for remote participants. However, low-cost and standalone HMDs like the Oculus Quest are becoming much more prevalent in household settings, making performing easily deployable or remote VR experiments more feasible. Despite this, one common limitation that these devices share is the lack of ability to easily access files on the system. This in turn makes collecting experiment data from the devices unnecessarily difficult without a custom ad hoc solution.

Our research lab, as with most others around the world, quickly realized that due to the unusual circumstances caused by the 2020 Coronavirus pandemic, conducting remote VR experiments will be increasingly necessary in order to follow government guidelines and regulations. There are many factors to running remote VR experiments, one of the most important being the collection and storage of the experiment data. To this end, we have designed and released the Remote Experiment Datalogging (RED) Toolkit. RED comprises a series of components for researchers to administer remote experiments and collect data from remote or standalone VR devices in a seamless manner. The toolkit can be split into two main parts: a REST server and multiple client implementations. The REST server is composed of a collection of endpoints that facilitate administrative actions such as experiment creation and management, as well as participant actions such as registration and datalogging. The client implementations provide useful interfaces for interacting with the RED server via these exposed endpoints. We have currently written several client implementations, which we will discuss in Section 2, but our hope is that by open-sourcing this project, more client implementations will be contributed by the community.

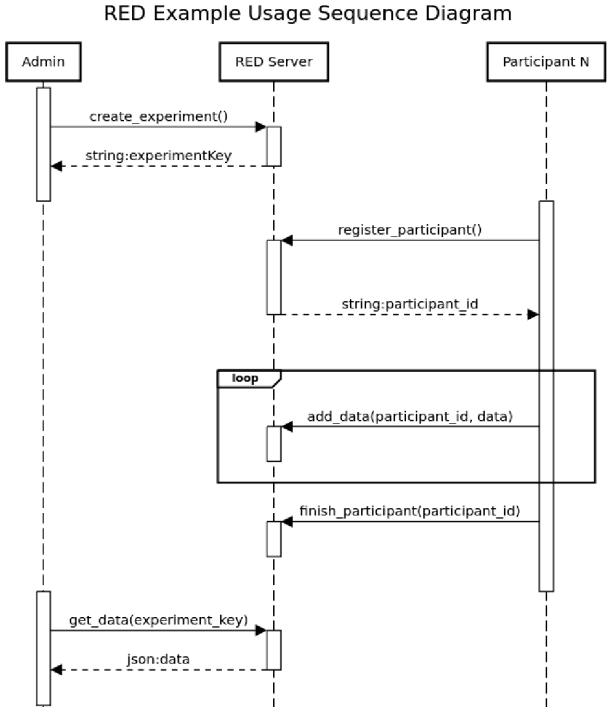


Figure 1: A UML Sequence Diagram for basic usage. Note that this diagram shows the bare minimum use case and does not reflect all of the functionality that RED provides.

## 2 ARCHITECTURE AND OVERVIEW

RED uses a REST Application Programming Interfaces (API). REST, which stands for REpresentational State Transfer, was first presented by Roy Fielding in his 2000 dissertation and is described as an "architectural style for distributed hypermedia systems" [3]. This architecture is based on HTTP(S) calls (GET, POST, PUT, etc.) and is the backbone for most of the modern World Wide Web. RESTful systems are stateless, meaning that the client does not need to know the state of the server and vice versa. Changes made to the client side will not affect the server operation, thus improving the flexibility of the interface and simplifying the server components. As previously stated, the RED toolkit is composed of the server as well as a collection of client implementations.

The basic workflow for conducting an experiment using the RED toolkit is shown in Figure 1. Using one of the clients described below, the experiment administrator tells the RED server to create a new experiment. This experiment object is stored in a database on the server and contains key information regarding the experiment. Then using a provided client (though not necessarily the same one

*e-mail: {adeni026, suma, thoma891}@umn.edu

978-1-6654-4057-8/21/$31.00 ©2021 IEEE
DOI 10.1109/VRW52623.2021.00183

as the administrator), participants can subsequently register for the experiment to get a unique participant ID, repeatedly add data until the experiment is finished, and then let the server know that they are finished. After the participants have finished, the experiment administrator then collects the data from the server.

RED Server. The RED server employs the Flask framework, a commonly used micro-web framework for creating APIs in Python that does not require any additional dependencies [2]. This handles all the back-end functionality, mainly allowing for querying through multiple endpoints. Flask's small footprint and ease of use makes deployment of the RED server relatively simple and painless.

The RED server endpoints can be divided into two main categories: admin endpoints and participant endpoints. Admin endpoints are designed to be called only by machines controlled directly by the experimenter or their colleagues. The main distinction between admin and participant endpoints is that calls to the admin endpoints must provide an experiment key. Currently, the admin endpoints allow an experimenter to create and delete experiments, view the registered participants, and download experiment data.

Participant endpoints are designed to be called by machines controlled by the participants. Our current use case assumes the participant machines are self-contained VR headsets such as the Oculus Quest, but any machine with a connection to the internet could serve as a participant machine. Participant endpoints allow experiment programs running on the participant machine to register a participant to an experiment and add experiment data from the participant machine to the server's database.

Researchers have an ethical responsibility to keep participant data, even anonymous data, private and secure. The traditional model for a system like this would be for experimenters to create a user account and for the system to provision permissions for them to access data from the experiments that they create. It would also need to support the sharing of permissions so that the accounts belonging to research team members could also access the data. This model, while common, is not without potential security flaws such as users employing weak passwords (or passwords that they use elsewhere) and privilege escalation. User provisioning also requires an amount of system administration overhead that we were seeking to avoid. To get around these issues we decided to implement a simpler resource token based model. In our system, this translates to the server requiring a token in order for users to access experiment data. As RED does not require a complex user model, the system does not lose any functionality by not implementing user accounts.

Once a user creates an experiment the server generates a unique token, referred to as the experiment key, for accessing that experiment's data. The experiment key can be shared by the experimenter to others who should also have access to the experiment data. The experiment key is a random 128 bit version 4 Universal Unique Identifier (UUID4), which guarantees a unique identifier and is virtually impossible to guess, unlike a human generated password. Using a version 4 UUID allows us to avoid conflicting identifiers in our system, and unlike UUID version 1, it does not use the computer's MAC address in generation which would allow for more potential security vulnerabilities. It is important to note that while we have taken steps to secure the RED server and the data that it holds, we are not security experts. Again, our intent is that by open-sourcing this project the community can further strengthen the security measures that have already been outlined.

RED Clients. Currently, we have created three client implementations for the RED toolkit: a Unity client, a JavaScript client, and a Python client. The Unity client is designed to allow interactions with the RED server to either occur in the editor (for admin actions), or during playtime (for participant actions). This allows the management of experiment data to be completely contained within the Unity environment. The JavaScript client allows for interacting with the RED server from a browser and can be embedded in web-based game engines such as Babylon.js or can simply be used within a HTML page (for example, to create a survey). Using the JavaScript client, we have also implemented a web portal in the Flask application for researchers to create and manage experiments, as well as download experiment data, directly from their browser. The Python client includes an API module as well as a command line interface tool for managing and testing RED experiments. These clients are simple to integrate into experiments with minimal effort.

Public Release. The RED toolkit has been released using the the MIT open-source license in the hopes that the community will find it useful and potentially contribute to the project. The source code and documentation can be found on GitHub [1].

## 3 Conclusion and Future Work

Although the RED toolkit has been deployed on a university-controlled server, extensive testing must still be conducted under a variety of research contexts. Currently, our lab is developing two remote experiments using Unity, and two WebXR experiments using Babylon.js. We also would like to develop and release an additional client implementation for the Unreal engine. We have focused on implementing VR experiments with RED due to the nature of our research, but we would like to point out that its data collection capabilities are not necessarily restricted to VR. The toolkit, therefore, could be deployed in a much broader variety of research applications.

Aside from testing, there are a few additional features that we would like to see implemented. First we would like to add a feature to restrict who can create experiments on a particular RED server instance. We plan to implement another token based system in which a unique and pre-shared token would be required to create an experiment. These tokens could be distributed by the system administrator to those that should have experiment creation capabilities. By creating this set of pre-shared tokens, we can further monitor anyone that is trying to access the RED server and reduce the affordances of any malicious actors. Second, we would like to transition the server from HTTP to HTTPS in order to establish an encrypted communication channel that protects from network eavesdropping and adds another layer of confidentiality.

We have presented the Remote Experiment Datalogging Toolkit, a platform designed to simplify the handling and management of data for experiments conducted on standalone or remote VR devices. RED is simple to use and is ready to integrate with the most common VR experience design platforms. It is also open-source, so it is possible to create a client for a particular VR experience design platform if it does not already exist. We believe that this system can greatly simplify experiment design and management as our community continues to conduct experiments remotely, but we also expect it to be useful even for in-person experiments that rely on standalone VR systems such as the Oculus Quest.


## Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1901423.


## References


[1] Remote Experiment Datalogging Toolkit. `https://github.umn.edu/illusioneering/RED`.

[2] F. A. Aslam, H. N. Mohammed, J. M. Mohd, M. A. Gulamgaus, and P. Lok. Efficient way of web development using python and flask. *International Journal of Advanced Research in Computer Science*, 6(2), 2015.

[3] R. T. Fielding and R. N. Taylor. *Architectural styles and the design of network-based software architectures*, vol. 7. University of California, Irvine Irvine, 2000.

[4] B. Huber and K. Z. Gajos. Conducting online virtual environment experiments with uncompensated, unsupervised samples. *Plos one*, 15(1):e0227629, 2020.