The following high-level pseudocode was created as a reader’s guide for non-programmers to understand the functionalities of each component of the Citizen Preservationist application.

Readers with a stronger technical background will find more nuanced information on the functionalities of this hybrid mobile/desktop software directly in the heavily commented source code on our GitHub repository. [doi:10.5281/zenodo.3895281](https://doi.org/10.5281/zenodo.3895281)

---

**CitPres DataCollector mobile app:**

1) The app opens to a welcome screen that asks the user for a given, predetermined:

 a) user unique ID

 b) session/location unique ID

2) After these metadata are collected, the user is taken to a simple prompt where she can decide whether to take in-app video training or skip:

 a) where the user may take the video training.

 b) where the user may skip and continue with photo taking.

3) The app initiates the target device for cameras to use as the main camera:

 a) for phones, the rear camera is chosen.

 b) for computers, the first webcam is chosen.

3) A simple text and image are slowly flashed so that the user can see the camera video feed and is reminded to tap to take a picture.

5) After the screen is tapped, the user is taken to a page where she can review the image taken:

 a) where the user may return and retake the picture.

b) where the user may accept the given picture as the artifact and continue to input metadata.

 c) the location and time are taken and recorded.

6) Upon confirmation, the user is prompted with a series of entries that provide user-entered metadata for the chosen artifact:

 a) where the user may quit out of this artifact and retake a picture.

 b) where the user may accept the given metadata.

7) Once the user enters metadata and moves to review, she is presented with a page that displays all given metadata for review:

 a) where the user may go back and edit the metadata previously entered.

 b) submit and save the given artifact.

8) When submitting the artifact, the image and all metadata are compressed into a json file where:

 a) the image is compressed and encrypted into a text format for later transfer.

 b) saved to a location that can be locally retrieved from the target device.

 c) a copy of the artifact’s picture is also copied into the phone’s default image gallery.

9) After saving the json, the user is given confirmation that the file was saved, and:

 a) the user may restart from the beginning or,

 b) quit out of the application entirely.

---

CitPres Map desktop app:

To allow for broader application to any site of cultural significance users can add any map to overlay data captured on top of.

1) Once CitPres Map is launched, a C# script allows for adding any raster map:

 a) where the script takes in a jpeg, png, or tiff image.

 b) where the script takes in the dimensions of the map and coordinates (in WGS84 format) for all four corners of the real location that corresponded with locations on the map.

2) After the raster map and coordinates are entered, using the known GPS coordinates of corners of the image, along with a relatively accurate scaled ration of pixel to GPS coordinate, the script approximately calculates all data point locations on the image map to allow for artifacts’ real-world coordinates to be plotted on the map.

3) Once the map setup is complete, the user is prompted to navigate to a folder containing all data points, sorted in subfolders by date. These folders should automatically be done once the data is downloaded from the DataCollector mobile app.

4) CitPres Map then traverses every subfolder, loading all the data and importing them into the map engine on different pages, so that they can be hid and shown individually.

5) The main page of the CitPres Map includes the ability to scroll the entirety of the map to see all data points, scroll through the given timeline of data points, and change the given folder using a button in the top right corner.

---

CitPres Convert and Save auxiliary script:

1) When the script is loaded, it is designed for ease to work in the same directory it is placed, such that the directory is formatted as follows:

 a) the directory has subdirectories.

 b) the subdirectories are categorized by the date the data was taken.

 c) the subdirectories contain json files with the following metadata fields:

 "Access Restrictions"

 "Campus/Unit"

 "Copyright Status"

 "Creator"

 "Date"

 "Description 1"

 "Description 1 Type"

 "Description 2"

 "Description 2 Type"

 "Extent"

 "Identifier"

 "Language"

 "Language Code"

 "Place 1 Authority ID"

 "Place 1 Coordinates"

 "Place 1 Name"

 "Subject"

 "Subject (Name) 1 Name Type"

 "Coverage"

 "Title"

 "Type"

 "Artifact Group"

 "Artifact Type"

 "Condition"

 "Artifact Category"

 "Material"

2) The script will then traverse every folder, and do the following for each folder:

a) create two folders to store the recovered image files, and singular csvs for specialized use.

b) create and keep track of manifest jsons and csvs for each dated folder, to be stored in the parent directory.

 c) loads and records every single data point.

 d) writes image to /Pictures.

 e) writes csv to /CSVs.

 f) then, once all data is recorded,

 1) writes all jsons to ./{date}.json.

 2) writes all csvs to ./{date}.csv.