



# **Administrator Manual**

## *Team 1*

Kyle Fritz

Laras Istiqomah

David Leiberg

Julian Sniffen

Nicholay Topin

## *Client*

MAPLE Lab (point of contact: John Winder)

5/15/17

Birdry  
Administrator Manual

**Table of Contents**

	<u>Page</u>
1. Introduction	2
1.1 Purpose of This Document	2
1.2 References	2
2. System Overview	2
2.1 Background	2
2.2 Hardware and Software Requirements	2
2.2.1 <i>Android Application Hardware and Software</i>	2
2.2.2 <i>Server Hardware and Software</i>	2
3. Administrative Procedures	3
3.1 Installation	3
3.1.1 <i>Android Application Installation</i>	3
3.1.2 <i>Server Installation</i>	4
3.2 Routine Tasks	5
3.3 Periodic Administration	5
3.4 User Support	5
4. Troubleshooting	6
4.1 Dealing with Error Messages and Failures	6
4.2 Known Bugs and Limitations	6
Appendix A – Team Review Sign-off	7
Appendix B – Document Contributions	8

## **1. Introduction**

### **1.1 Purpose of This Document**

This document is intended for anyone that would like to utilize the Birdry system for bird detection purposes. The application is comprised of a server-side component and an Android application. The Android portion can simply be installed on any applicable Android device; however, the server requires some simple setup. Both the System Requirements Document and the System Design Document will be helpful to have as references for this document.

### **1.2 References**

Throughout this document references will be made to:

1. The Birdry System Requirements Specification
2. The Birdry System Design Document
3. Caffe setup ([http://caffe.berkeleyvision.org/install\\_appt.html](http://caffe.berkeleyvision.org/install_appt.html))
4. PostgreSQL setup (<https://help.ubuntu.com/community/PostgreSQL>)
5. PGAdminIII connection instructions (<https://www.pgadmin.org/docs/pgadmin3/1.22/connect.html>)

## **2. System Overview**

### **2.1 Background**

The Birdry system administrator oversees the Birdry server and handles all of the images uploaded by users of the Birdry Android application, which is responsible for sending images to the Birdry server and informing the user of the corresponding bird points generated for sent image. The application allows the user to easily track the location, bird points, and time associated with each image. Additionally, users can inform the server administrator of incorrect results through a review process, which the administrator should review for image content.

### **2.2 Hardware and Software Requirements**

#### ***2.2.1 Android Application Hardware and Software***

The Birdry Android application supports any Android device running Android 5.0 (LOLLIPOP) or newer. The device must support camera and GPS functionality.

#### ***2.2.2 Server Hardware and Software***

The Birdry server components were developed and installed on a machine running Proxmox, a Debian-based distribution of Linux. Setup would be identical for any other Debian-based distribution. Other distributions of Linux may be suitable, depending

on whether the required dependencies are available. No specific hardware is required, though the recommended setup includes at least four gigabytes of memory, a modern CPU with at least two cores, and an Nvidia GPU with at least six gigabytes of memory which supports CUDA 7.5.

The Birdry server requires the use of Python 3 (version 3.5+), Django (version 1.11+), a local (on same machine) PostgreSQL (version 9.5+) database, and an installation of Caffe (November 2016 release or later). The following are also needed for server operation: the Python 3 libraries numpy, pyCaffe, psycopg2, Pillow, and optionally exifread (if performing server-side EXIF GPS extraction); Django and the Django REST Framework; git, if planning to install directly from a git repository instance; and PGAdminIII, which, while not required, is highly recommended for graphical database administration. Installation of these software requirements, along with the server code itself, is discussed below in section 3.1.2.

### **3. Administrative Procedures**

#### **3.1 Installation**

The Android application must be installed on each individual mobile device. A single server must be set up to service all application installations. The process for setting up both are given below, in sections 3.1.1 and 3.1.2.

##### ***3.1.1 Android Application Installation***

The Android application must be configured in order to work with the server which you have set up. This guide recommends and assumes that you use Android Studio to implement your changes. In order for the network requests to be routed correctly, the following methods in the Android code base must be changed:

- File: PictureDetailActivity.java
  - reviewImage()
- File: MainActivity.java
  - sendImage()
  - testAuth()
  - getAuth()

Each method specifies an IP address to connect to. Simply replace the existing IP address with the IP address of your server.

Example assuming your server's IP address is 12:34:56:78

"http://74.98.84.98:8000/test" becomes "http://12:34:56:78:8000/test"

Once the code base has been changed, select the Build APK option from the Build tab in Android Studio. Transfer the newly created APK to your Android device through any method you are comfortable with. Selecting the APK from the device's filesystem will install the application and make it available to launch.

### 3.1.2 Server Installation

It is necessary to install and configure the requisite software and associated libraries for the Birdry server to operate properly. Given the installer has an internet connection, there are a number of different methods to acquire and install the software described in section 2.2.2. However, if the installation is to take place on a Debian-based Linux distribution (as it was during development), the Debian apt tool and the Python pip tool may be used for installing the necessary packages. The required packages are as follows:

Install using Debian apt:

```
nvidia-cuda-toolkit
caffe-tools-cuda
libcaffe-cuda*
python3-caffe-cuda
libcaffe-cuda-dev
caffe-doc
caffe-cuda
python3-pip
postgresql
postgresql-contrib
pgadmin3
git
```

Install using Python 3 pip:

```
django
djangoRESTframework
Pillow
psycopg2
exifread
```

If not using Debian, Caffe must be manually installed. Instructions can be found on Caffe's website ([http://caffe.berkeleyvision.org/install\\_appt.html](http://caffe.berkeleyvision.org/install_appt.html)). If the server does not have a GPU, a CPU-only version of Caffe must be installed. For Debian-based systems, this can be achieved by installed "caffe-cpu" instead of "caffe-cuda". For non-Debian systems, follow the instructions given on the main Caffe website for a CPU-only installation.

Upon installation of the PostgreSQL database, the database must be set up with an available database instance to fill, and configured to allow remote user access via password. If using a Debian-based operating system, use the instructions from the following link (<https://help.ubuntu.com/community/PostgreSQL>). Upon setting up the database, configure the Django settings file in the Birdry server, `birdryServer/settings.py`, to connect to the database instance at its name and with its password. Instructions for connecting the recommended database administration software PGAdminIII (<https://www.pgadmin.org/docs/pgadmin3/1.22/connect.html>) should be followed to connect to the database. Lastly, the Birdry backend directory contains some

pre-constructed SQL commands that may be run to create the views “allBirds,” “notBirds,” and “userPictures” which organize data generated by the server and stored in the database by all images containing birds, all not containing birds, and relating usernames directly to pictures, respectively. These SQL commands may be run in PGAdminIII’s query execution tool or manually with the psql command line tool (with the PGAdminIII option being recommended).

### **3.2 Routine Tasks**

The system administrator is able to start and stop the Birdry server easily by navigating to the server directory and running the command `python3 manage.py runserver 0:8000`, which will bind the server to any given IP interface on port 8000, eliminating the need to run the server as root on a privileged port. Starting the server brings up the Django server feed, and the server can be stopped again by sending the TERM signal (with CTRL + C in a terminal window, e.g.).

The system administrator should regularly review images marked by users as requiring review, and perform manual review of the marked images to determine their contents appropriately and update the database as needed. Images which have not been marked for review should be reviewed at random to help ensure the system is operating properly. The administrator should keep track of images that have been denoted as birds and those denoted as not birds for later off-site backup (see section 3.3), for which the pre-constructed database views “allBirds” and “notBirds” may prove helpful. The administrator should also monitor the Django server feed indicating which endpoints are being accessed and what responses are being delivered, noting any discrepancies or patterns of errors.

### **3.3 Periodic Administration**

To prevent server storage space from overflowing, user uploaded images should be periodically removed from their upload locations and moved off-machine to a separate backup machine, preferably one where another neural network instance can use them for further training and refinement of its classification capabilities. The database of image metadata should similarly be backed up to a separate machine. Less regularly, perhaps every three or four backups or so, the server neural network should be substituted for a copy of the off-site neural network being trained on user images.

### **3.4 User Support**

For additional support, you can contact us via e-mail to our customer service representative, who will respond within 24 hours.

## **4. Troubleshooting**

### **4.1 Dealing with Error Messages and Failures**

If you are experiencing problems with the camera portion of the application, try closing and then reopening Birdry. If the Android application crashes repeatedly, it is recommended that the Birdry cache be deleted and the application reinstalled. Other errors can occur because you have not given Birdry the proper permissions. You can check this by going into your settings app and selecting the Birdry application, where you will be able to grant these permissions manually if needed.

### **4.2 Known Bugs and Limitations**

There is only one known bug that exists within our Android application. If the Birdry Android application is closed while sending an image to the server, the image will never receive the response from the server. As a result, the image will be stuck in the processing state permanently. Due to architectural limitations, there is no current remedy for this situation; however, the image will not impact any other functionalities within the application. Additionally, the image can be removed from the application normally if necessary.

One additional limitation of the application is icon scaling when run on devices with differing screen sizes. The functionality of the application will not be compromised, however certain icons may appear inconveniently small on larger devices. This limitation arose because there was not sufficient access to different Android devices during the testing process.

On the server side, error handling and logging is currently limited to what exists in the code base and the Django server console output, which only detail what status codes were returned from errors and at which endpoints they occurred. For example, if a Python 3-level error were to occur while running the server that is not handled by the existing error handlers, then a server error code of 500 is returned to the user, with no additional information logged for the administrator to see what went wrong.

## Appendix A – Team Review Sign-off

We, the members of the development team, undersign here to indicate that every member of the team has reviewed and approved the contents and format of this Administrator Manual, and that their comments and/or concerns (if any) are listed here.

Name: Kyle Fritz

Signature: 


Date: 5/15/17

Comments:

N/A

---

Name: Laras Istiqomah

Signature: 

Date: 5/15/17

Comments:

N/A

---

Name: David Leiberg

Signature: 

Date: 5/15/17

Comments:

N/A

---

Name: Julian Sniffen

Signature: 

Date: 5/15/17

Comments:

N/A

---

Name: Nicholay Topin

Signature: 

Date: 5/15/17

Comments:

N/A

---



## **Appendix B – Document Contributions**

Below is an estimate of the contribution from each team member, including specific work done and percentage of total document completed.

*Kyle Fritz*

Contributions:

- 2.2.1 Android Application Hardware and Software
- 4.1 Dealing with Error Messages and Failures
- Revisions throughout

Est. Contribution: 15%

*Laras Istiqomah*

Contributions:

- Table of contents revision and document review
- 3.4 User Support

Est. Contribution: 15%

*David Leiberg*

Contributions:

- 2.1 Background
- 2.2.2 Server Hardware and Software Requirements
- 3.1.2 Server Installation
- 3.2 Routine Tasks
- 3.3 Periodic Administration
- 4.2 Known Bugs and Limitations

Est. Contribution: 20%

*Julian Sniffen*

Contributions:

- 1.1 Purpose of This Document
- 2.1 Background
- 2.2.1 Android Hardware and Software Requirements
- 3.1.1 Android Installation
- 4.1 Dealing with Errors and Messages on Android
- 4.2 Known Bugs and Limitations

Est. Contribution: 30%

*Nicholay Topin*

Contributions:

- 2.2.2 Server Hardware and Software
- 3.1.2 Server Installation
- Revisions throughout

Est. Contribution: 20%