Birdry

System Design Document

Team 1
Kyle Fritz
Laras Istiqomah
David Leiberg
Julian Sniffen
Nicholay Topin

Client
MAPLE Lab (point of contact: John Winder)

4/5/17

Birdry
System Design Document

Table of Contents

1 Introduction

1.1 Purpose of This Document

The purpose of this document is to describe the design of the Birdry application. Key topics covered in this document include the high level system architecture, lower level class designs, and persistent data design of Birdry, as well as a matrix connecting the system components to its corresponding functional requirements.

1.2 References

Throughout this document references will be made to:

1. The Birdry System Requirements Document
2. The Birdry User Interface Design Document

2 System Architecture

2.1 Architectural Design

Figure 1 outlines the architecture of the Birdry system, which follows a conventional client-server pattern comprising two major components with their own significant subcomponents. The first, the Birdry Android application, provides the interface for users to interact with the system. It makes use of the native Android API to control camera, navigation, and storage functionality for the application. It also utilizes Android's geolocation functions in conjunction with the Google Maps API to communicate with Google's servers to generate the map for the map view in the application (see the User Interface Design Document for further details).

Secondly, the Birdry server controls the logic for receiving and classifying the images taken and sent by users through the application, along with storing the images. The server uses the Django REST Framework to communicate with the application over HTTP and URL endpoints. The classification functionality is performed by an ImageNet neural network made using the Caffe framework which determines whether a received image contains a bird or not. Relevant user and image information (including the bird-or-not determination made by the neural network) and association is stored in a PostgreSQL database, which is used in tandem with the the native operating system filesystem to store the images themselves.

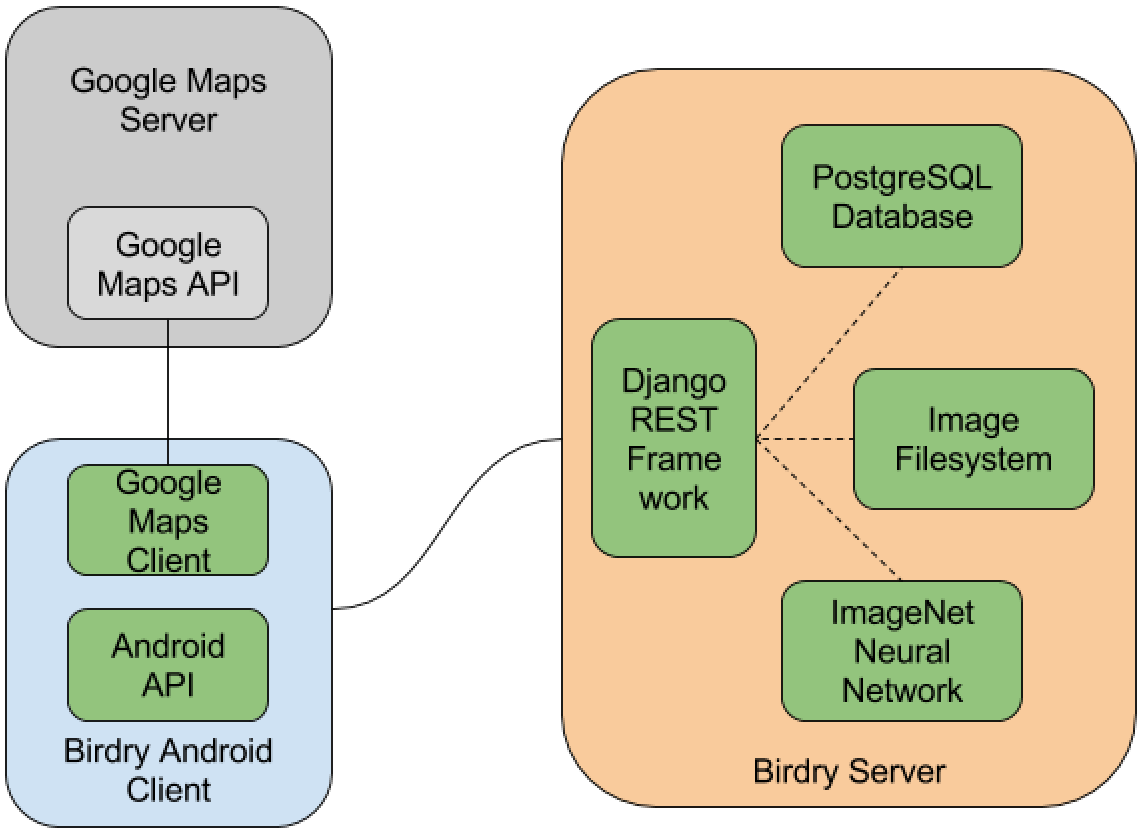


Figure 1: Birdry Architectural Diagram

2.2 Decomposition Description

The Android application, as shown in Figure 2, is centered around the MainActivity class. This class handles the user's interaction with the device. The class requests permissions necessary for the application to function from the user with the *requestPermissions* method, including permissions to use the device's camera, local storage, and geolocation capabilities. Additionally, the MainActivity class acts as the liaison between the Birdry application and the Birdry server, handling HTTP communications on the application's end.

Each of the three views (map, gallery, camera, and detailed) are supported by a supplemental class. Each of these is an Android Fragment class. This allows the views to be swipeable and maintain their state in the background. Each fragment class only needs a way to create a new instance of itself (*newInstance*) and a way to set up the view associated with the fragment (*onCreateView*). The specifics of the view are then described within the layout configuration file(s) associated with each fragment. For diagrams of what the views look like, see the User Interface Design Document. The MainActivity class runs the fragment corresponding to the current view.

The camera preview allows for displaying element two of the camera view (see User Interface Design Document). This is done using the *surfaceCreated*, *surfaceDestroyed*, and *surfaceChanged* methods, which generate and remove the camera view. The camera view itself is obtained by overlaying the camera preview over the BlankFragment, which is a dummy fragment used solely for this purpose.

The MapFragment uses the Google Maps API to show the map to the user, so long as all of the proper permissions are given. Using functions from the API, we are able to set markers for picture locations, as well as move the view of the map to the user's current location. This fragment can detect when a market is selected in order to transition into the detailed view. A drop-down radial option menu is used to present the user with different criteria by which to limit the displayed markers.

The GalleryFragment shows thumbnail icons for photographs taken. Like the MapFragment, it also uses a drop-down radial option menu to present filter options. When a thumbnail icon is clicked, the MainActivity transitions to the detailed view.

To render the detailed view, the DetailFragment is used. This fragment displays a photograph alongside its information (see User Interface Design Document for more details). Buttons are used to register user input on this view: one for returning to the previous fragment (GalleryFragment or MapFragment) while keeping the photograph, one for deleting the photo and then returning to the previous fragment, and one for toggling whether the photograph is of a bird.

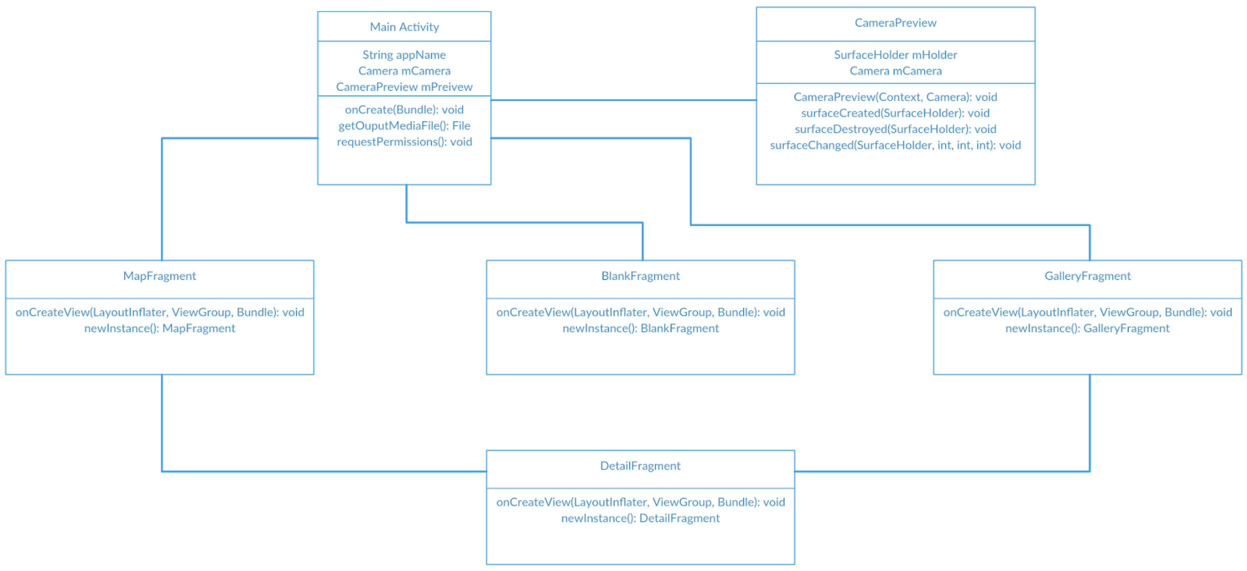

Main Activity
String appName
Camera mCamera
CameraPreview mPreivew
onCreate(Bundle): void
getOuputMediaFile(): File
requestPermissions(): void
CameraPreview
SurfaceHolder mHolder
Camera mCamera
CameraPreview(Context, Camera): void
surfaceCreated(SurfaceHolder): void
surfaceDestroyed(SurfaceHolder): void
surfaceChanged(SurfaceHolder, int, int, int): void
MapFragment
onCreateView(LayoutInflater, ViewGroup, Bundle): void
newInstance(): MapFragment
BlankFragment
onCreateView(LayoutInflater, ViewGroup, Bundle): void
newInstance(): BlankFragment
GalleryFragment
onCreateView(LayoutInflater, ViewGroup, Bundle): void
newInstance(): GalleryFragment
DetailFragment
onCreateView(LayoutInflater, ViewGroup, Bundle): void
newInstance(): DetailFragment

Figure 2: Class diagram of the Android application

3 Persistent Data Design

3.1 Database Descriptions

Figure 3 demonstrates the Birdry data model used in the PostgreSQL database. Users are identified by unique device ID, stored as strings and used as the User entity's primary key. Though simple in the current design, user data could be easily extended in the future to relate other information to users, such as emails. Images carry with them significantly more information in contrast. Images are primarily keyed on image IDs represented as strings, but additionally are characterized by the date they were taken (a date and time), the latitudinal and longitudinal coordinates of where the picture was taken, and of course, a Boolean indicator of whether or not the picture is of a bird as determined by the ImageNet neural network.

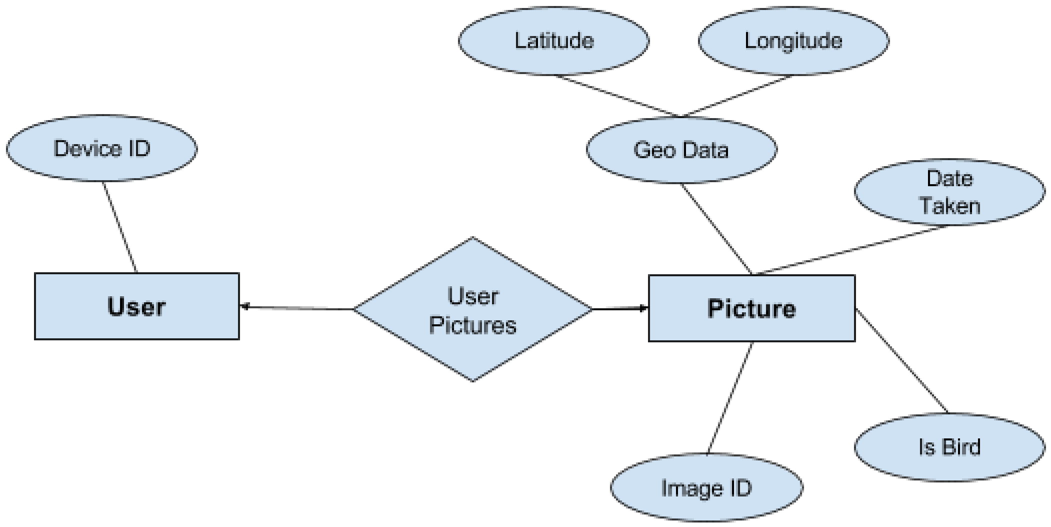


Figure 3: Entity-Relationship Model of Database

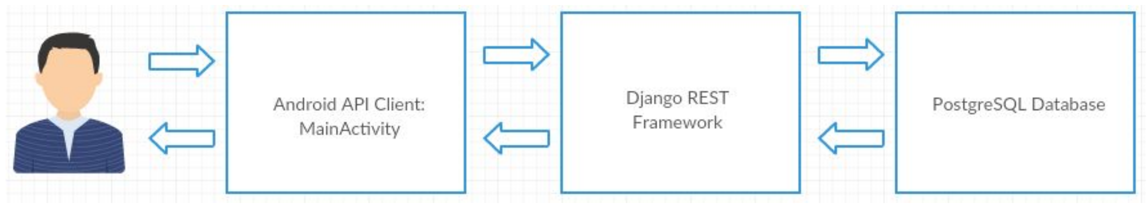
3.2 File Descriptions

Birdry does not implement any unique file types, instead making use of existing and already standardized file types. Specifically, images are communicated and processed in the JPEG format. Both the Android application and server store images using the default file system. References to an image (within the application or within the server database) use the image's name as the ID by which to retrieve the image. The name is guaranteed to be unique as it is generated from device-specific information and the time the image was taken.

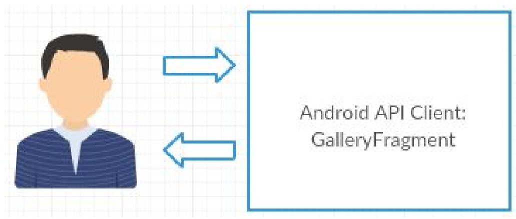
4 Requirements Matrix

Please refer to the Birdry System Requirements Specification for details regarding the corresponding use cases.

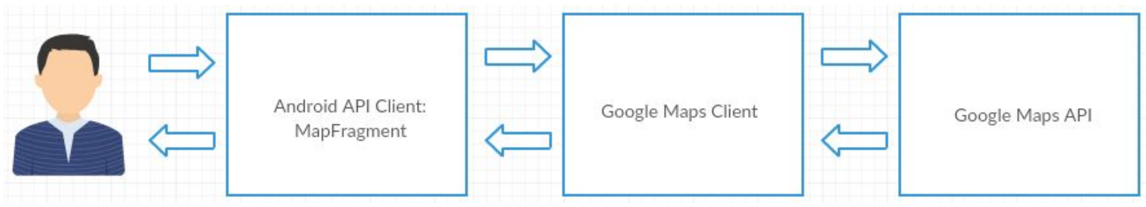
Use Case 1: Take Picture



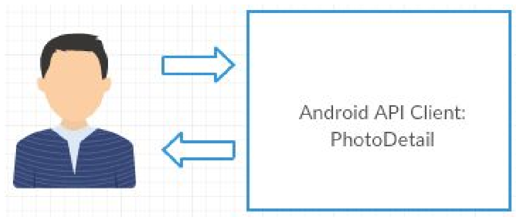
Use Case 2: View Gallery



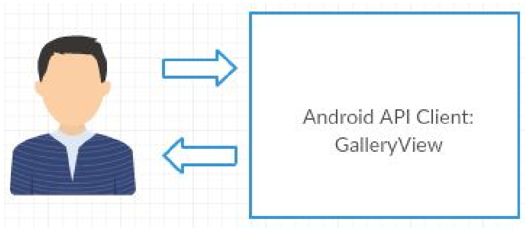
Use Case 3: View Map



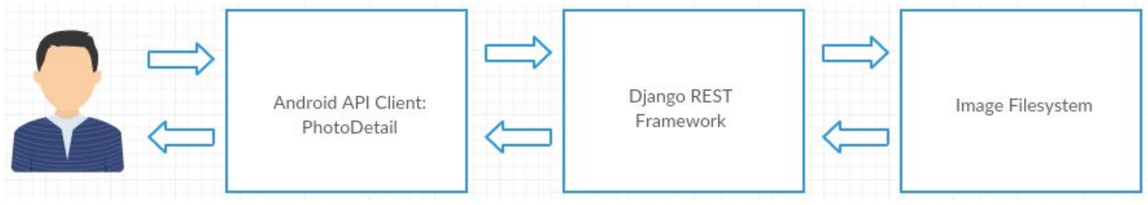
Use Case 4: Viewing Picture Detail



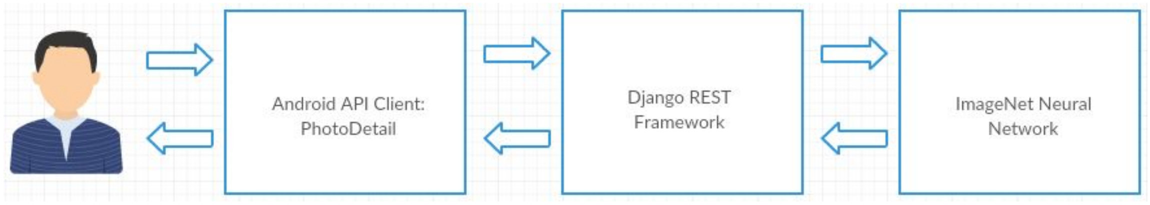
Use Case 5: Filter Photos



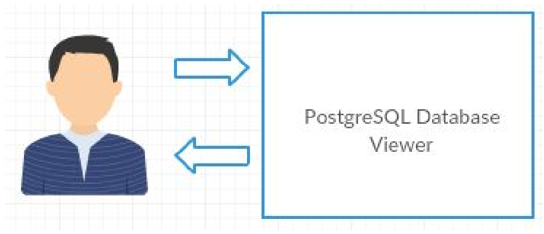
Use Case 6: Delete Picture



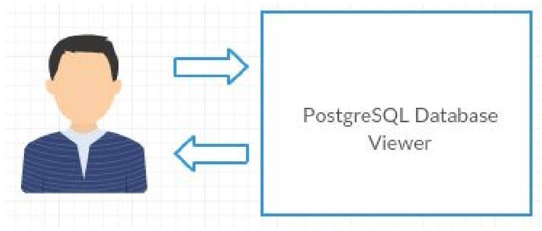
Use Case 7: Ask for Review



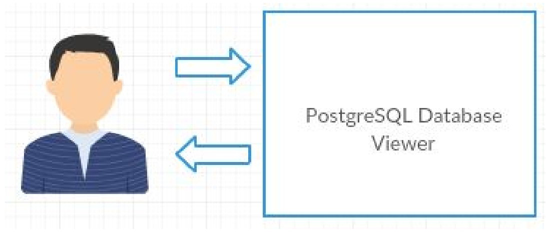
Use Case 8: View Database



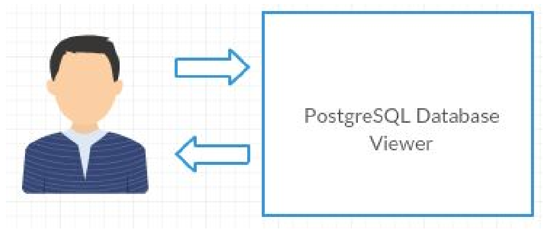
Use Case 9: Admin Search Picture



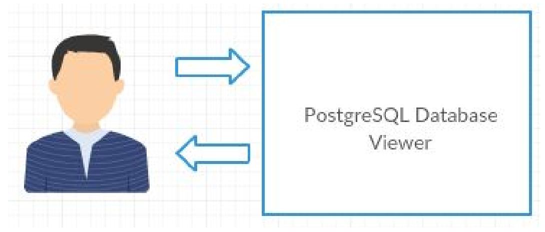
Use Case 10: Admin Picture Detail



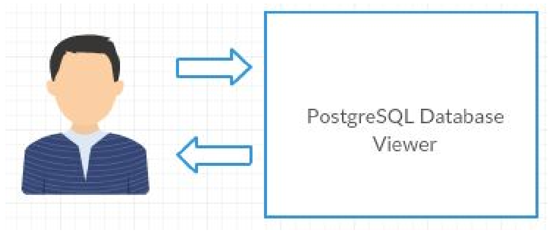
Use Case 11: Admin Add Picture Detail



Use Case 12: Admin Delete Picture



Use Case 13: Admin Filter



Appendix A – Agreement Between Customer and Contractor

We, the clients and members of the development team, undersign here to indicate our agreement that the completed Birdry application (frontend and backend) will be designed as stated/illustrated in this document.

If future changes are necessary, a revised hard copy will be presented to the client. Once approved by the client and signed off by all team members, the revisions will come into effect.

Clients

Name: ____________________ Signature: _____________________ Date: ________

Comments:

____

____

Name: ____________________ Signature: _____________________ Date: ________

Comments:

____

____

Team

Name: ____________________ Signature: _____________________ Date: ________

Name: ____________________ Signature: _____________________ Date: ________

Name: ____________________ Signature: _____________________ Date: ________

Name: ____________________ Signature: _____________________ Date: ________

Name: ____________________ Signature: _____________________ Date: ________

Appendix B – 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 System Design Document, and that their comments and/or concerns (if any) are listed here.

Name: ______________________ Signature: _______________________ Date: ________

Comments:

____

____

Name: ______________________ Signature: _______________________ Date: ________

Comments:

____

____

Name: ______________________ Signature: _______________________ Date: ________

Comments:

____

____

Name: ______________________ Signature: _______________________ Date: ________

Comments:

____

____

Name: ______________________ Signature: _______________________ Date: ________

Comments:

____

____

Appendix C – 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
<u>Contributions:</u>
- Appendix A, B, C
- Section 1.1-1.2
- Section 2.2 text co-author and figure update
- Formatting
- Document Review
- Figure 2 fix

<u>Est. Contribution:</u> 20%

Laras Istiqomah
<u>Contributions:</u>
- Formatting
- Document review

<u>Est. Contribution:</u> 5%

David Leiberg
<u>Contributions:</u>
- Section 2.1, Section 3.1 - 3.2
- Section 2.2 text co-author
- Figure 1, Figure 2
- Document Review

<u>Est. Contribution:</u> 17%

Julian Sniffen
<u>Contributions:</u>
- Section 2.2 outline and first figure
- Section 4: Requirements Matrix

<u>Est. Contribution:</u> 20%

Nicholay Topin
<u>Contributions:</u>
- Section 2.2 text co-author
- Editing of Section 3.1 - 3.2
- Review with Client

<u>Est. Contribution:</u> 18%