BATALOG

Hadi Soufi - Thien Le - Kevin Keomalaythong - David Massey - Rahim Iqbal

TABLE OF CONTENTS

1. Project Definition

Scientists at UNCG record bat calls from all over the state to learn more about their behavior. There can be thousands of calls in a given night, so doing research on them can require years of manual work. Batalog seeks to do that automatically in a web interface by using deep learning techniques to classify bat calls. Specifically, we aim to identify whether bat calls are used for echolocation or for some other purpose. Once the data is cleaned and classified, we display the bat calls on a website and allow the user to interact with them through various functional modules.

2. Project Requirements

Functional

- Accurate Classification
    1. A convolutional neural network will be used to classify the type of bat call that is uploaded. The CNN should achieve a minimum of 90% accuracy during the training and testing phase prior to uploading any files.
    2. The model must be evaluated to remedy overfitting. This can be done by checking training accuracy versus testing accuracy.
- Interactive user interface
    1. Display the data in a meaningful way: Allow the user to be able to see the sorted bat data in a collection of album. The user will be able to favorite and mark the data.
    2. Allow the user to upload zero-crossing files for analysis: This function will give a front end design for the user to upload the file. While the zero-crossing files being processed in the background the user will get greet with a loading screen.
    3. Allow the user to login and create an account

Usability

- **User interface**: Web-based
- **Performance**: The speed the website can process user data and output in a meaningful way. However this will be a low priority because we want to focus on accuracy first.

System

1. Hardware
    a. Any Laptop and Desktop PC capable of running modern Operating System such as Mac OS X, Windows 10
    b. Stable internet connection
2. Software
    a. Any modern browser (e.g. Chrome, Firefox, Opera, Safari).
3. Database
    a. The system shall allow uploaded images to be sent into the database, and facilitate communication between database and web interface
    b. The database shall store PNG images in binary-large-object format and have a storage capacity of at least 50 GB

Security

1. Username/password
2. SSL Certification

3. Project Specification

Areas

- Machine Learning
- Web Development
- Database Management

Platform: Web

Genre: Research tool

Libraries

- **TensorFlow + Keras API**: Python machine learning library/interface set that supports deep learning, i.e. facilitates building (convolutional) neural networks.
- **Numpy**: Python library that introduces n-dimensional arrays including matrices, and facilitates linear algebra operations such as matrix multiplication.
- **Matplotlib**: Python 2D plotting library that allows visualization and statistical analysis of data.
- **Dash**: Dash is a framework based on Python. It is used to build an analytical web application. Primarily it builds a dashboard using Python.
- **Pillow 5**: A powerful image library for Python web development that will help with organizing the image libraries.
- **Django 2**: Django is a free and open source framework based on Python. Django has less freedom but it is more efficient and secure than other frameworks such as asp.net or visual studio.
- **SQLite3**: Python library to interface with SQL databases.

Frameworks

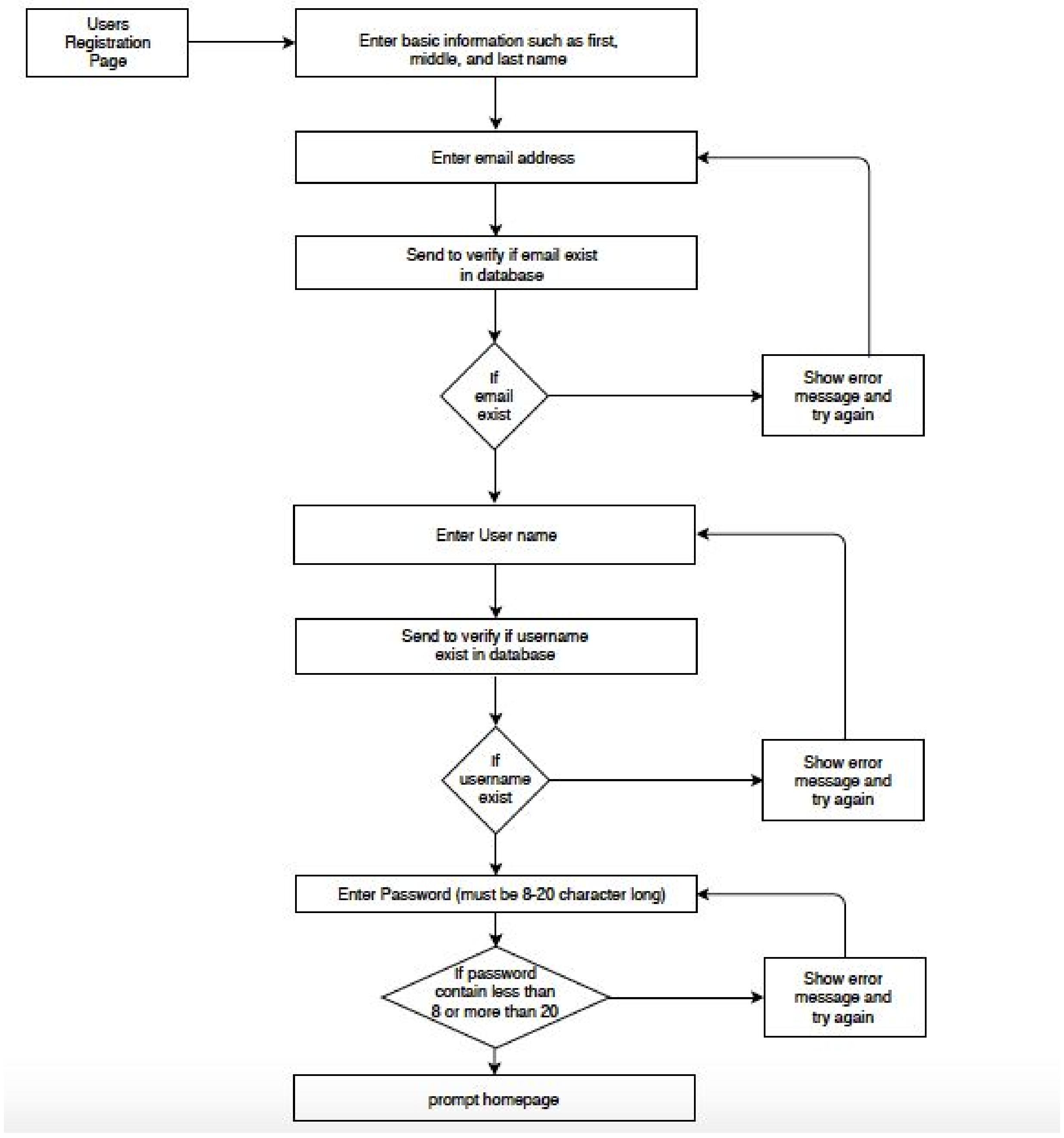
- **Django 2**: Python framework for building an HTML5 website
- **Nginx**: Web Server for Linux systems
- **Gunicorn**: Middleware for Linux systems
- **SQL**: Database and the interaction between the website and the database storage.

Development Environments

- **Jupyter Notebook**: Application used to write and execute Python code and visualize output.
- **Atom**: Text editor with built-in terminal for executing code

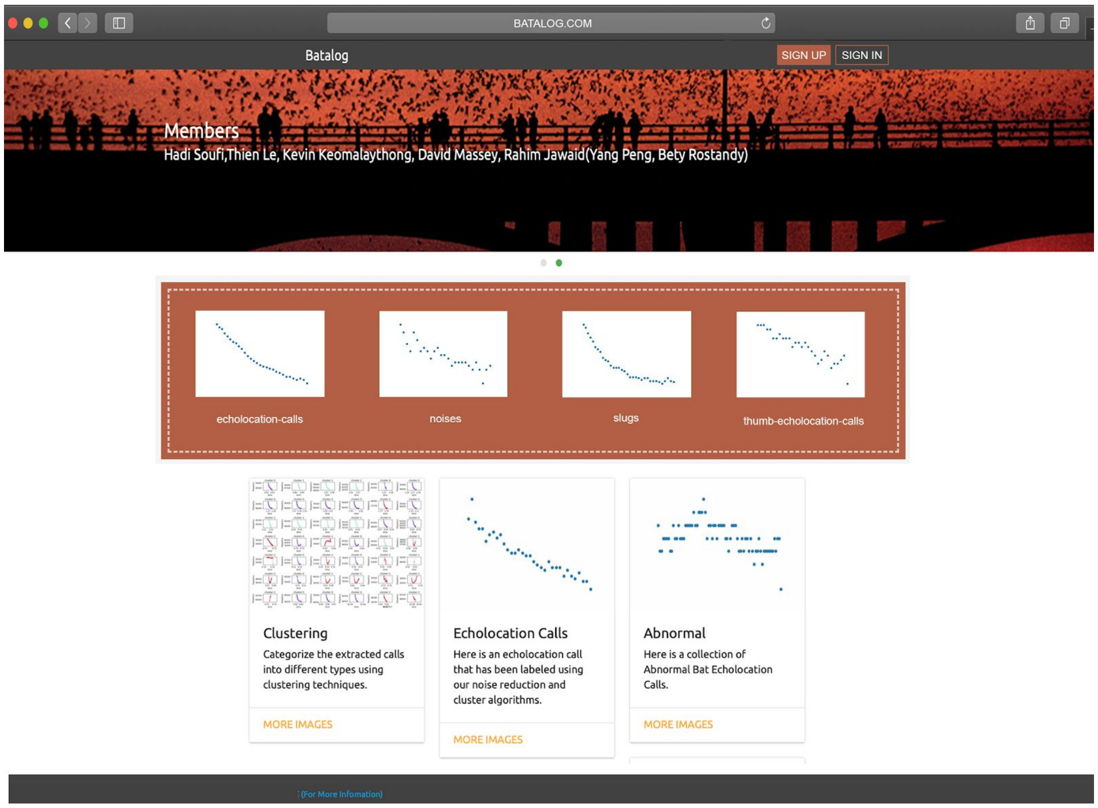
- **Sublime Text**: Text editor that helps to read/write Python and django code.
- **Pycharm**: Python IDE

4. System – Design Perspective



Users Registration Page
Enter basic information such as first, middle, and last name
Enter email address
Send to verify if email exist in database
If email exist
Show error message and try again
Enter User name
Send to verify if username exist in database
If username exist
Show error message and try again
Enter Password (must be 8-20 character long)
If password contain less than 8 or more than 20
Show error message and try again
prompt homepage

User registration page (Rahim)


BATALOG.COM

Batalog

SIGN UP SIGN IN

Members

Hadi Soufi,Thien Le, Kevin Keomalaythong, David Massey, Rahim Jawaid(Yang Peng, Bety Rostandy)

echolocation-calls

noises

slugs

thumb-echolocation-calls

Clustering

Categorize the extracted calls into different types using clustering techniques.

MORE IMAGES

Echolocation Calls

Here is an echolocation call that has been labeled using our noise reduction and cluster algorithms.

MORE IMAGES

Abnormal

Here is a collection of Abnormal Bat Echolocation Calls.

MORE IMAGES

: (For More Infomation)

Display Results (Thien)

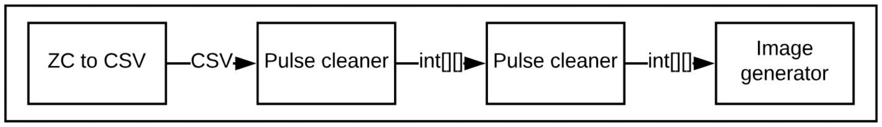
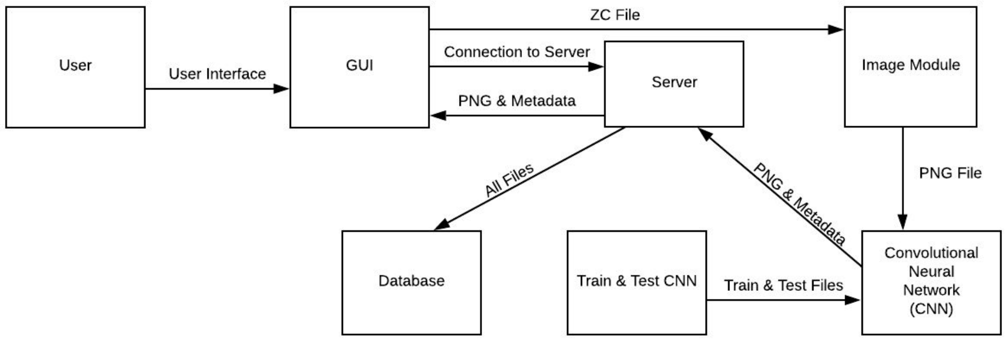
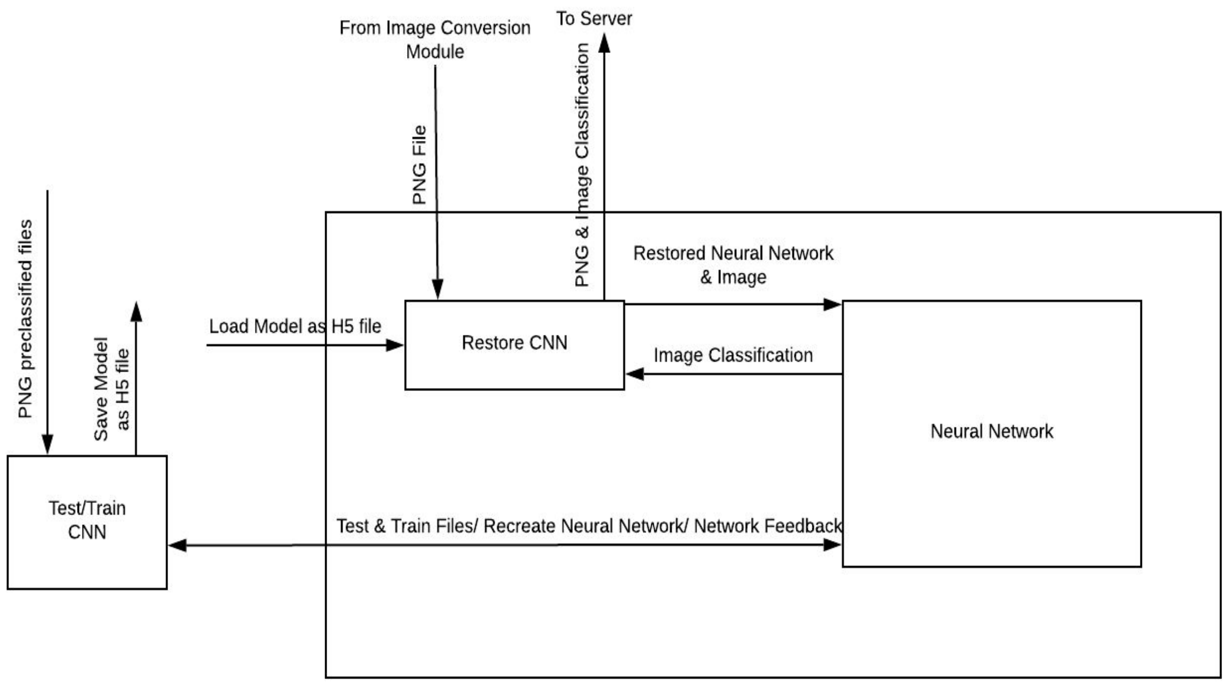


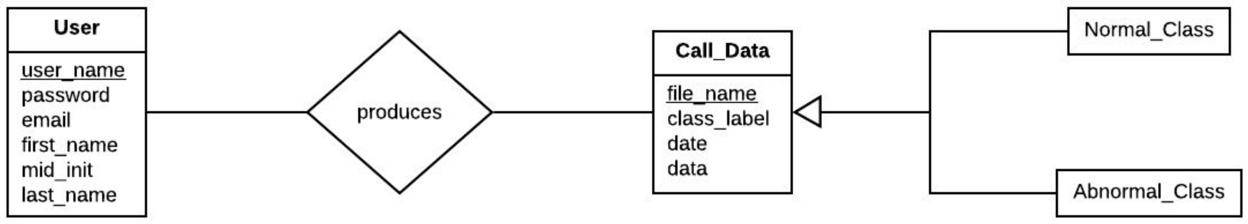
Image conversion module (Hadi)



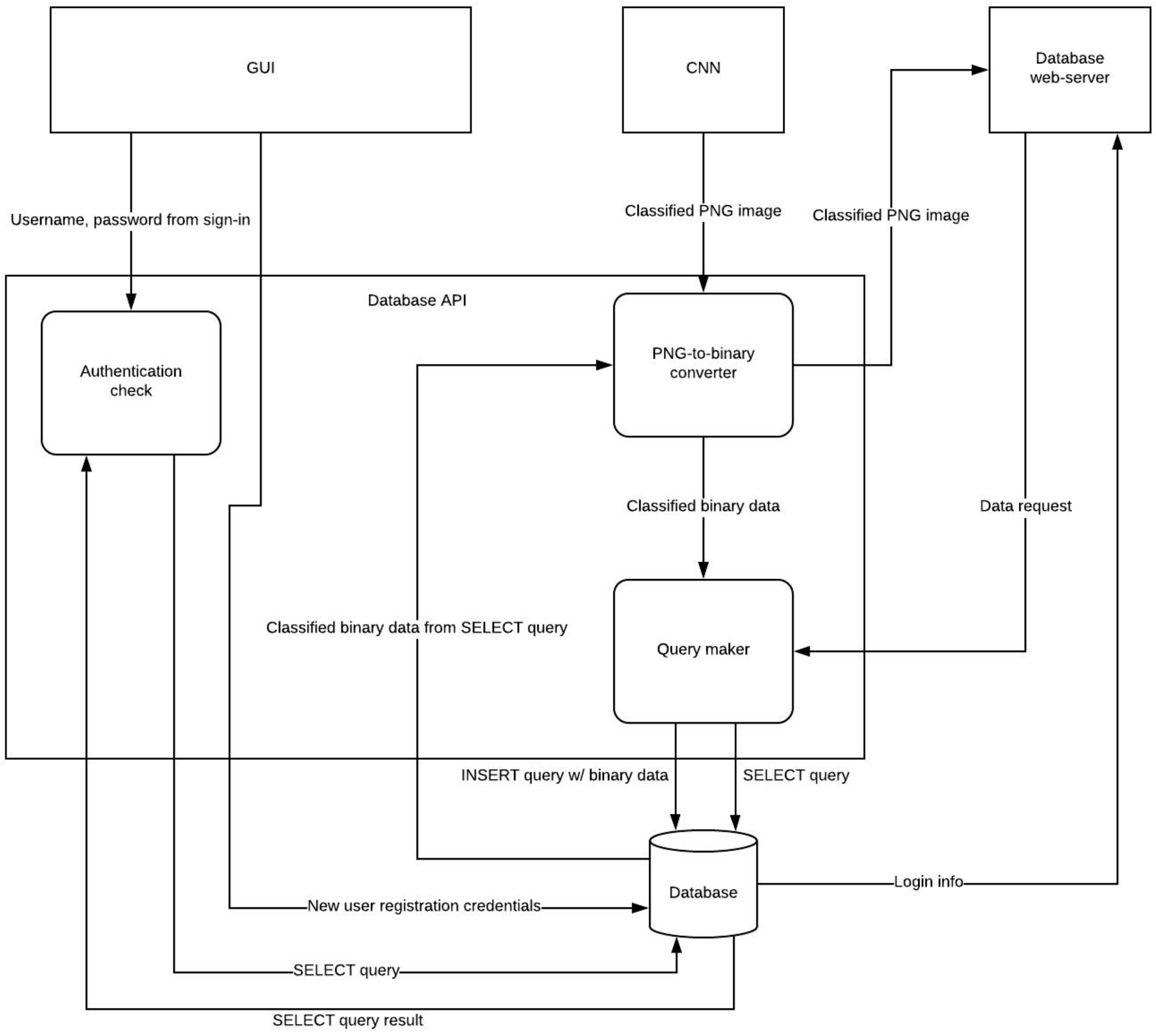
Overall operation - System Model (*David*)


From Image Conversion Module
To Server
PNG File
PNG & Image Classification
PNG preclassified files
Save Model as H5 file
Load Model as H5 file
Restore CNN
Restored Neural Network & Image
Image Classification
Neural Network
Test/Train CNN
Test & Train Files/ Recreate Neural Network/ Network Feedback

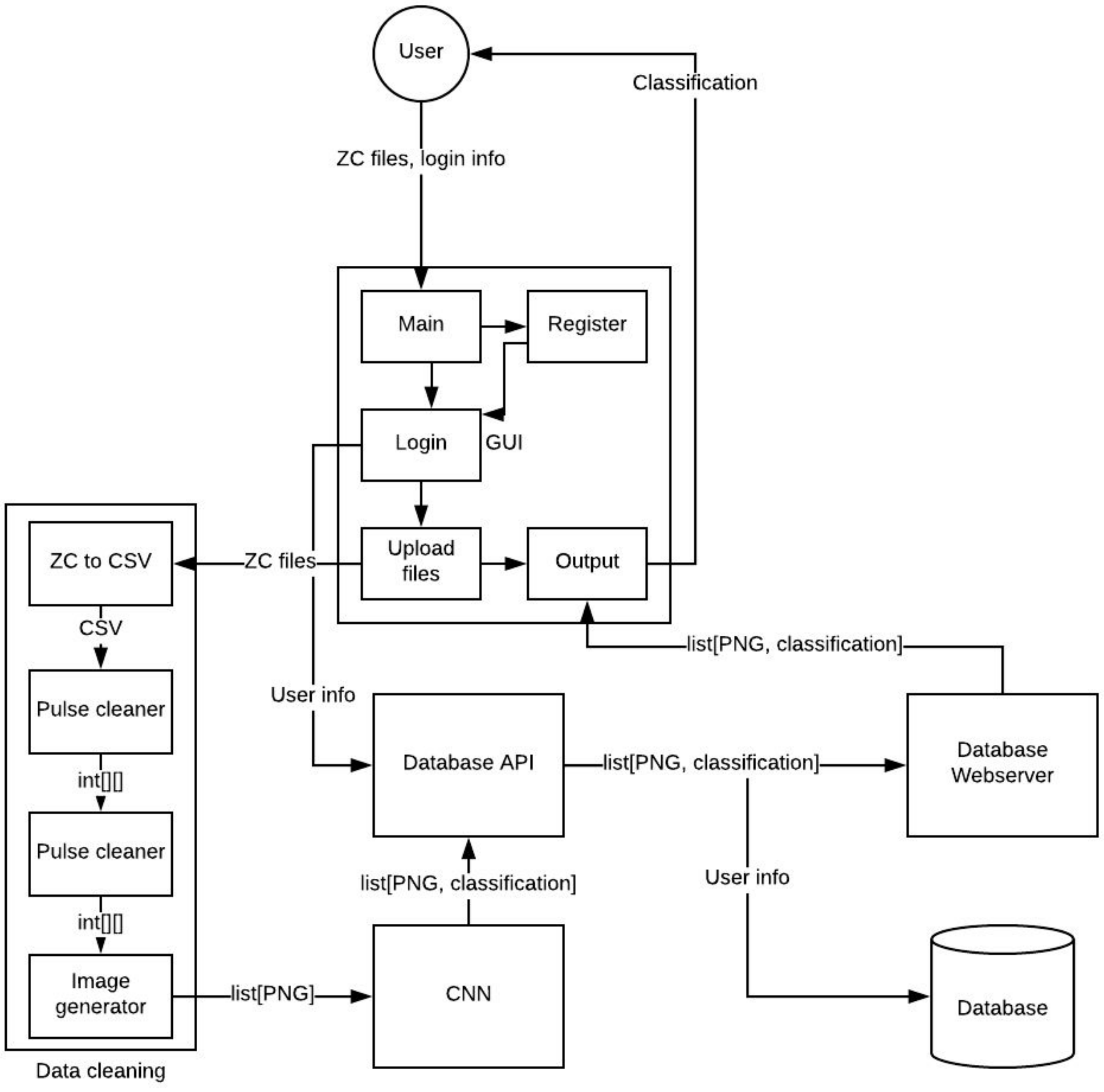
CNN (David)



Entity Relationship (E-R) Model (*Kevin*)



Database API (*Kevin*)



Controls (*Thien*), I/O (*Hadi*), DataFlow (*Kevin*)

5. System – Analysis Perspective

Subsystems

Front-end

- **User registration page**: Allows users to register on the website, so they can upload their ZC file. The users enter their basic information such as first, middle and last names. The register page also verifies the user email and username from the database. If username or email is present in the database, then it will display the error message and ask it to try again.
- **Display Results**: Allows user to upload and see the file and classification.
  - i. If the user is logged in, they can upload a zc file or set of zc files.
  - ii. While the file is being processed, the user will be greeted by a loading screen.
  - iii. Once the files have been processed, they will get displayed in separate albums.

Back-end

- **Connection from web to Database**:
  - i. CSS stylesheets
    1. Bootstrap- responsive grid system, extensive pre built components
  - ii. Digital Ocean Droplet Server
    1. A scalable compute platform with add-on storage, security, and monitoring capabilities to easily run production applications.
  - iii. Python Django Framework
    1. ORM- can be used to interact with application data from various relational databases such as SQLite, PostgreSQL and MySQL.
  - iv. Dependencies
    1. Pillow- Python Imaging Library.
    2. SQLite3- An embedded database, server-less and can run within our app.
- **Database API**: Handles GUI-database interactions and passes SQL queries to the database. The database stores PNG+metadata and user registration information.
- **Bat Call Extraction/Rendering**: Converts ZC files to PNG images. The images will then be fed to the CNN module for classification.
- **Convolutional Neural Network (CNN)**:Classifies bat pulses as normal or abnormal. Pulses are precleaned.

System

Name	Type
username	VARCHAR(255)
password	VARCHAR(255)
first_name	VARCHAR(255)
last_name	VARCHAR(255)
email	VARCHAR(255)
organization	VARCHAR(255)
password	VARCHAR(255)

auth_user

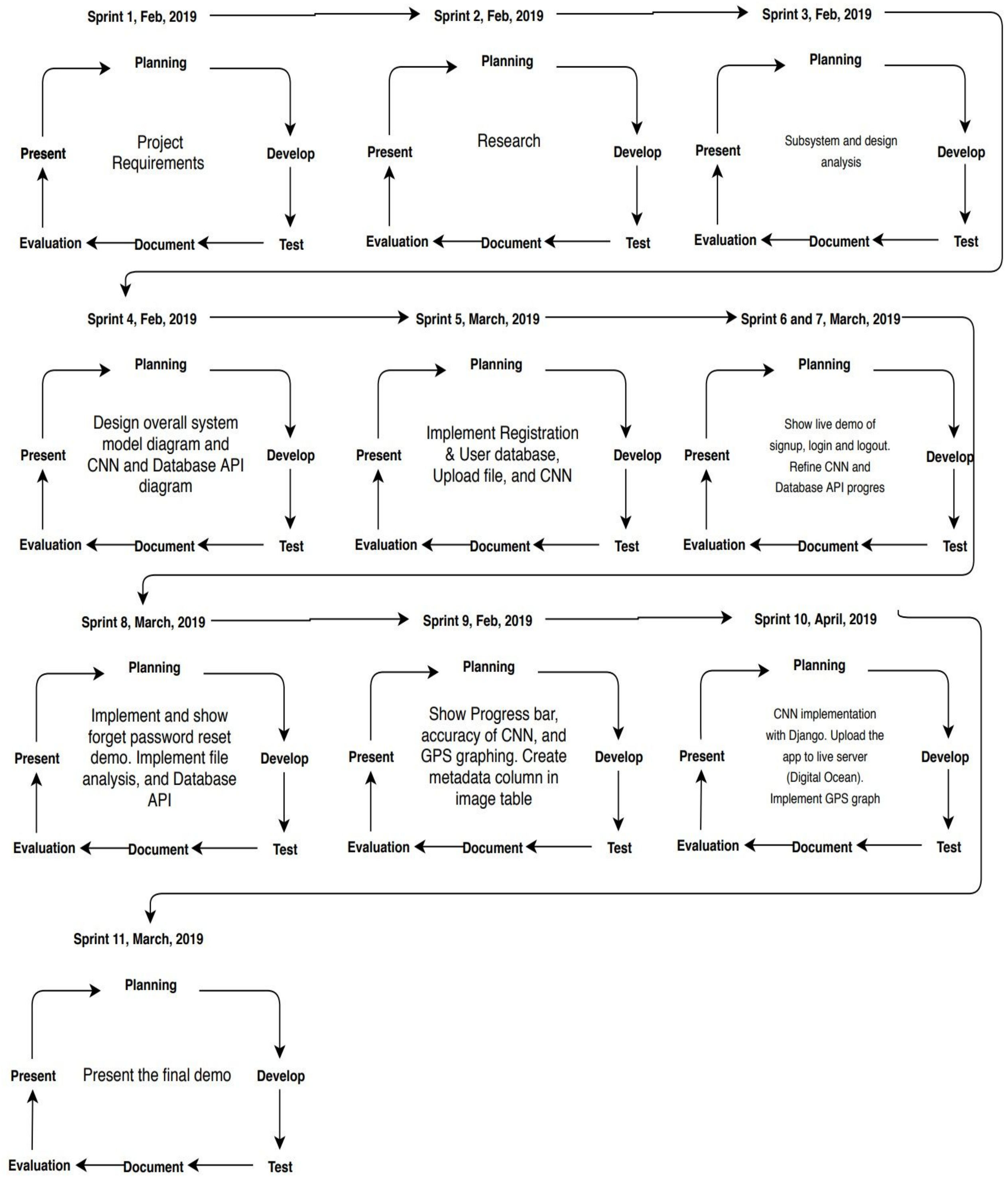
Name	Type	Description
name	VARCHAR(255)	Source filename
raw	BLOB	Processed bat call; (x, y) array
classification	VARCHAR(255)	CNN results; normal or abnormal
metadata	VARCHAR(255)	Source metadata
uid	INTEGER	Owner

images

Variable: Model

- Data type: Sequential object
- Description: This represents the CNN model used for training and predictions. The CNN is loaded and assigned to this variable prior to image classifications. It includes the network architecture, weights, and settings.

Process models


Sprint 1, Feb, 2019
Planning
Project Requirements
Present
Develop
Evaluation
Document
Test
Sprint 2, Feb, 2019
Research
Sprint 3, Feb, 2019
Subsystem and design analysis
Sprint 4, Feb, 2019
Design overall system model diagram and CNN and Database API diagram
Sprint 5, March, 2019
Implement Registration & User database, Upload file, and CNN
Sprint 6 and 7, March, 2019
Show live demo of signup, login and logout. Refine CNN and Database API progres
Sprint 8, March, 2019
Implement and show forget password reset demo. Implement file analysis, and Database API
Sprint 9, Feb, 2019
Show Progress bar, accuracy of CNN, and GPS graphing. Create metadata column in image table
Sprint 10, April, 2019
CNN implementation with Django. Upload the app to live server (Digital Ocean). Implement GPS graph
Sprint 11, March, 2019
Present the final demo

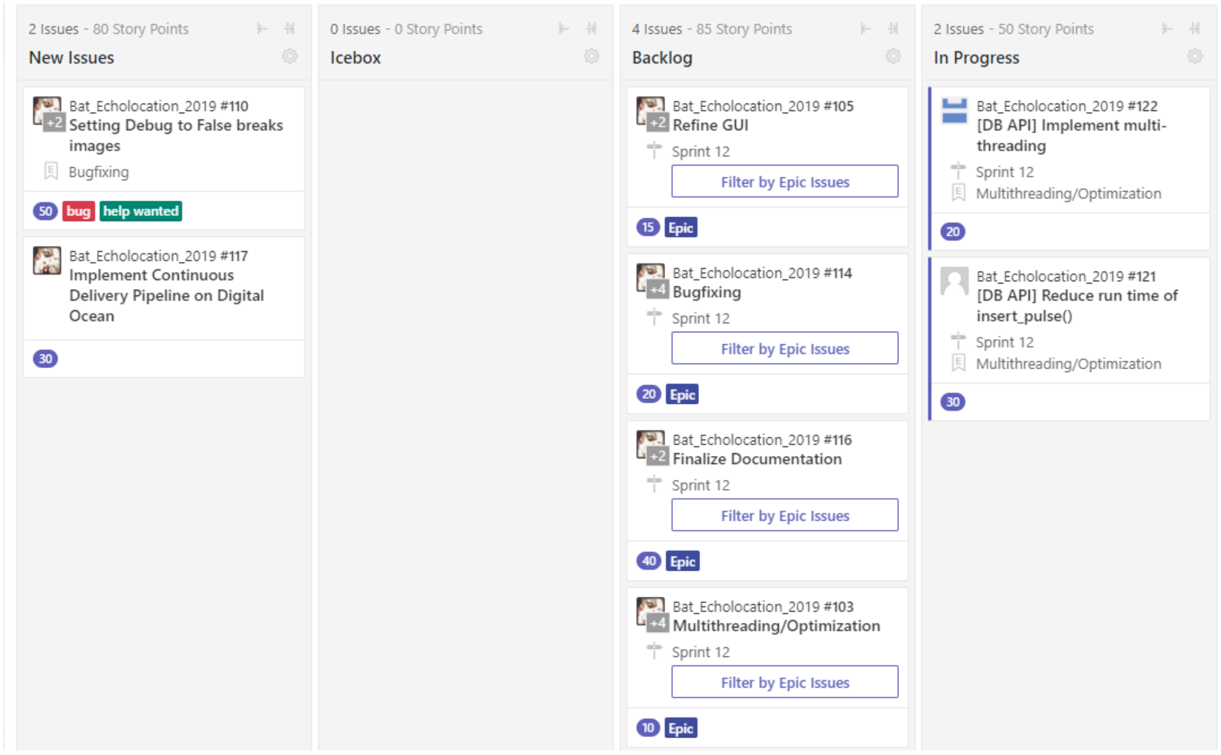
Agile Process

Algorithm Analysis

- **Entire system** - $O(n)$: Worst case of all subsystems.
- **CNN** - $O(1)$ All images conform to the same pixel size. They will be processed one at a time. The network layout remains unchanged between images.
- **Registration page** - $O(1)$: All the users follow the same steps for registering on the website
- **Zero-crossing file analysis** - $O(n)$: Almost entirely array operations. An increase in file size corresponds to a linear increase in runtime.
- **Database API** - $O(log\ n)$: The operations that the API pass to the DBMS are insert and search (SELECT query in SQL). Insertion is $O(1)$. Search is $O(log\ n)$ since each table will be indexed through B-trees.

6. Project Scrum Report

Product Backlog



New Issues (2 Issues - 80 Story Points)

- Bat_Echolocation_2019 #110 Setting Debug to False breaks images — Bugfixing — 50 bug help wanted
- Bat_Echolocation_2019 #117 Implement Continuous Delivery Pipeline on Digital Ocean — 30

Icebox (0 Issues - 0 Story Points)

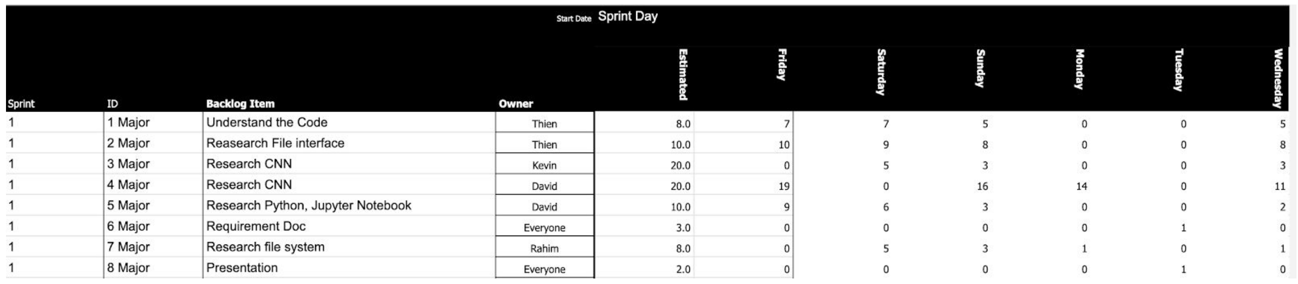
Backlog (4 Issues - 85 Story Points)

- Bat_Echolocation_2019 #105 Refine GUI — Sprint 12 — Filter by Epic Issues — 15 Epic
- Bat_Echolocation_2019 #114 Bugfixing — Sprint 12 — Filter by Epic Issues — 20 Epic
- Bat_Echolocation_2019 #116 Finalize Documentation — Sprint 12 — Filter by Epic Issues — 40 Epic
- Bat_Echolocation_2019 #103 Multithreading/Optimization — Sprint 12 — Filter by Epic Issues — 10 Epic

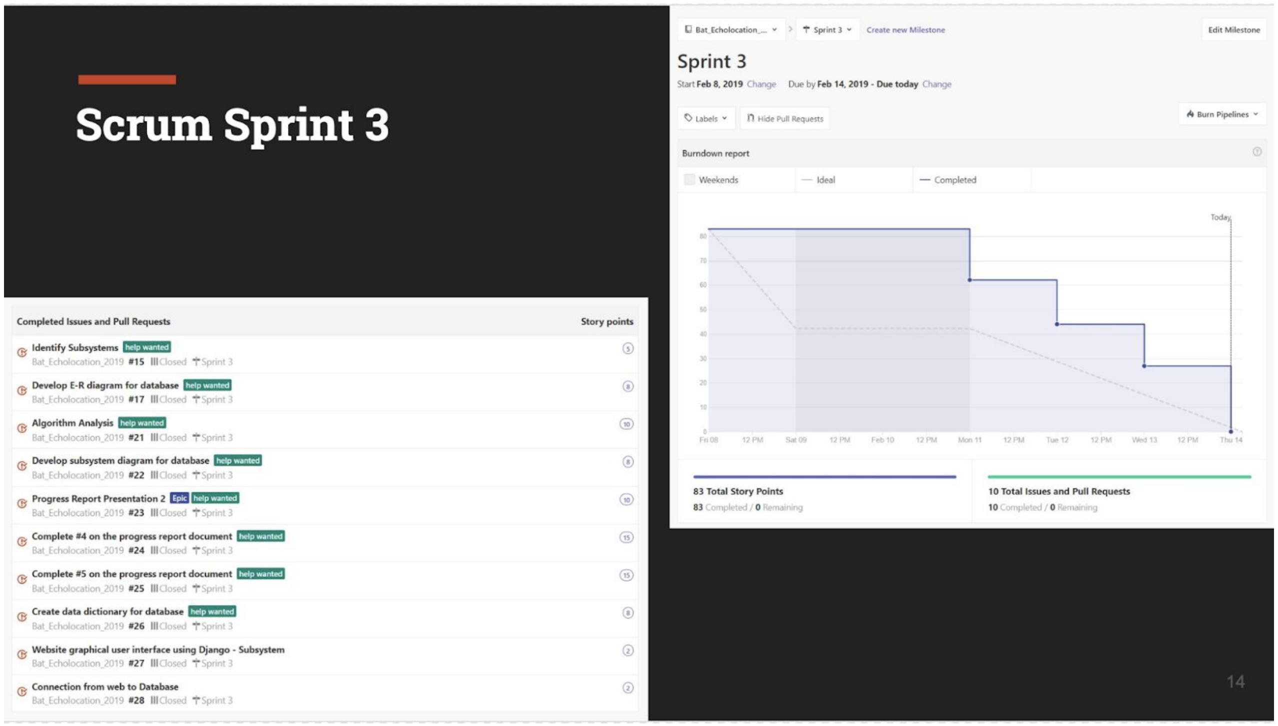
In Progress (2 Issues - 50 Story Points)

- Bat_Echolocation_2019 #122 [DB API] Implement multi-threading — Sprint 12 — Multithreading/Optimization — 20
- Bat_Echolocation_2019 #121 [DB API] Reduce run time of insert_pulse() — Sprint 12 — Multithreading/Optimization — 30

Sprint Backlog



			Start Date	Sprint Day						
Sprint	ID	Backlog Item	Owner	Estimated	Friday	Saturday	Sunday	Monday	Tuesday	Wednesday
1	1 Major	Understand the Code	Thien	8.0	7	7	5	0	0	5
1	2 Major	Reasearch File interface	Thien	10.0	10	9	8	0	0	8
1	3 Major	Research CNN	Kevin	20.0	0	5	3	0	0	3
1	4 Major	Research CNN	David	20.0	19	0	16	14	0	11
1	5 Major	Research Python, Jupyter Notebook	David	10.0	9	6	3	0	0	2
1	6 Major	Requirement Doc	Everyone	3.0	0	0	0	0	1	0
1	7 Major	Research file system	Rahim	8.0	0	5	3	1	0	1
1	8 Major	Presentation	Everyone	2.0	0	0	0	0	1	0



Scrum Sprint 3

Bat_Echolocation_... > Sprint 3 · Create new Milestone · Edit Milestone

Sprint 3

Start **Feb 8, 2019** Change   Due by **Feb 14, 2019 - Due today** Change

Labels · Hide Pull Requests · Burn Pipelines

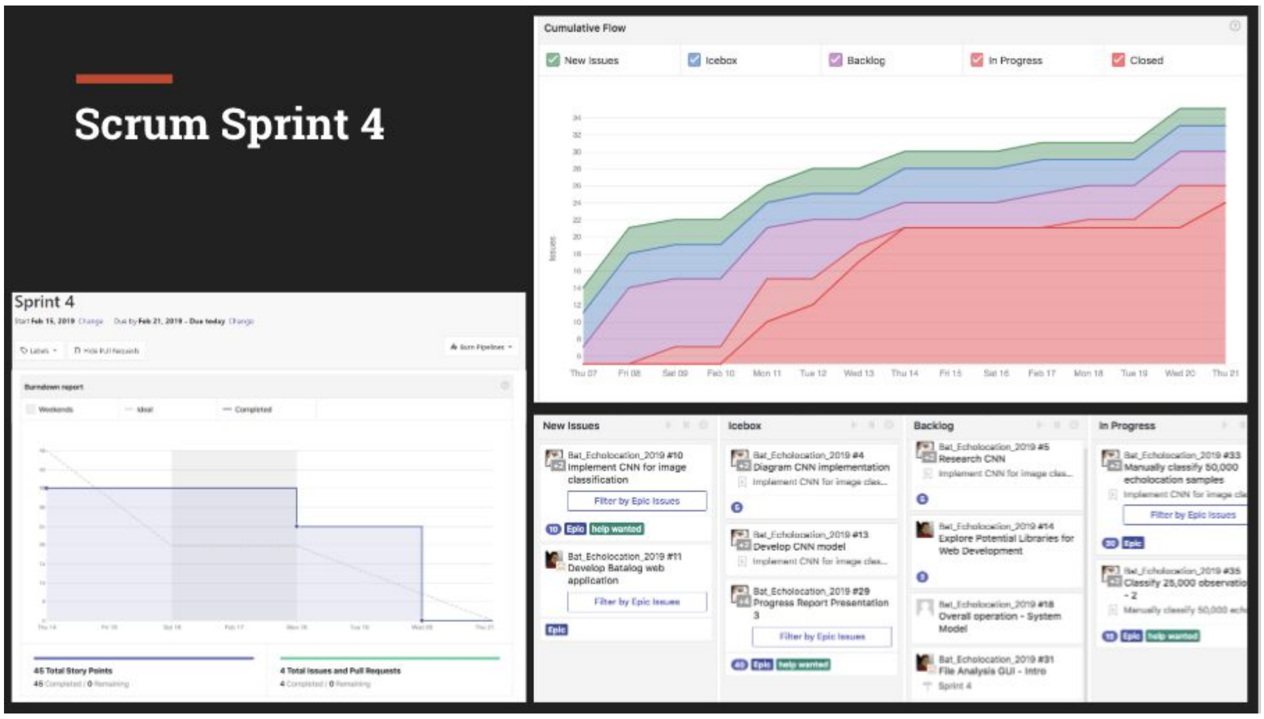
Burndown report

83 Total Story Points
83 Completed / **0** Remaining

10 Total Issues and Pull Requests
10 Completed / **0** Remaining

Completed Issues and Pull Requests	Story points
Identify Subsystems `help wanted` Bat_Echolocation_2019 **#15** Closed Sprint 3	5
Develop E-R diagram for database `help wanted` Bat_Echolocation_2019 **#17** Closed Sprint 3	8
Algorithm Analysis `help wanted` Bat_Echolocation_2019 **#21** Closed Sprint 3	10
Develop subsystem diagram for database `help wanted` Bat_Echolocation_2019 **#22** Closed Sprint 3	8
Progress Report Presentation 2 `Epic` `help wanted` Bat_Echolocation_2019 **#23** Closed Sprint 3	10
Complete #4 on the progress report document `help wanted` Bat_Echolocation_2019 **#24** Closed Sprint 3	15
Complete #5 on the progress report document `help wanted` Bat_Echolocation_2019 **#25** Closed Sprint 3	15
Create data dictionary for database `help wanted` Bat_Echolocation_2019 **#26** Closed Sprint 3	8
Website graphical user interface using Django - Subsystem Bat_Echolocation_2019 **#27** Closed Sprint 3	2
Connection from web to Database Bat_Echolocation_2019 **#28** Closed Sprint 3	2

14



Scrum Sprint 4

Sprint 4

Start **Feb 15, 2019** Change   Due by **Feb 21, 2019 - Due today** Change

Labels · Hide Pull Requests · Burn Pipelines

Burndown report

45 Total Story Points
45 Completed / **0** Remaining

4 Total Issues and Pull Requests
4 Completed / **0** Remaining

Cumulative Flow

New Issues

Bat_Echolocation_2019 #10
Implement CNN for image classification
Filter by Epic Issues
10 `Epic` `help wanted`

Bat_Echolocation_2019 #11
Develop Batalog web application
Filter by Epic Issues
`Epic`

Icebox

Bat_Echolocation_2019 #4
Diagram CNN implementation
Implement CNN for image clas...
6

Bat_Echolocation_2019 #13
Develop CNN model
Implement CNN for image clas...

Bat_Echolocation_2019 #29
Progress Report Presentation 3
Filter by Epic Issues
40 `Epic` `help wanted`

Backlog

Bat_Echolocation_2019 #5
Research CNN
Implement CNN for image clas...
6

Bat_Echolocation_2019 #14
Explore Potential Libraries for Web Development
3

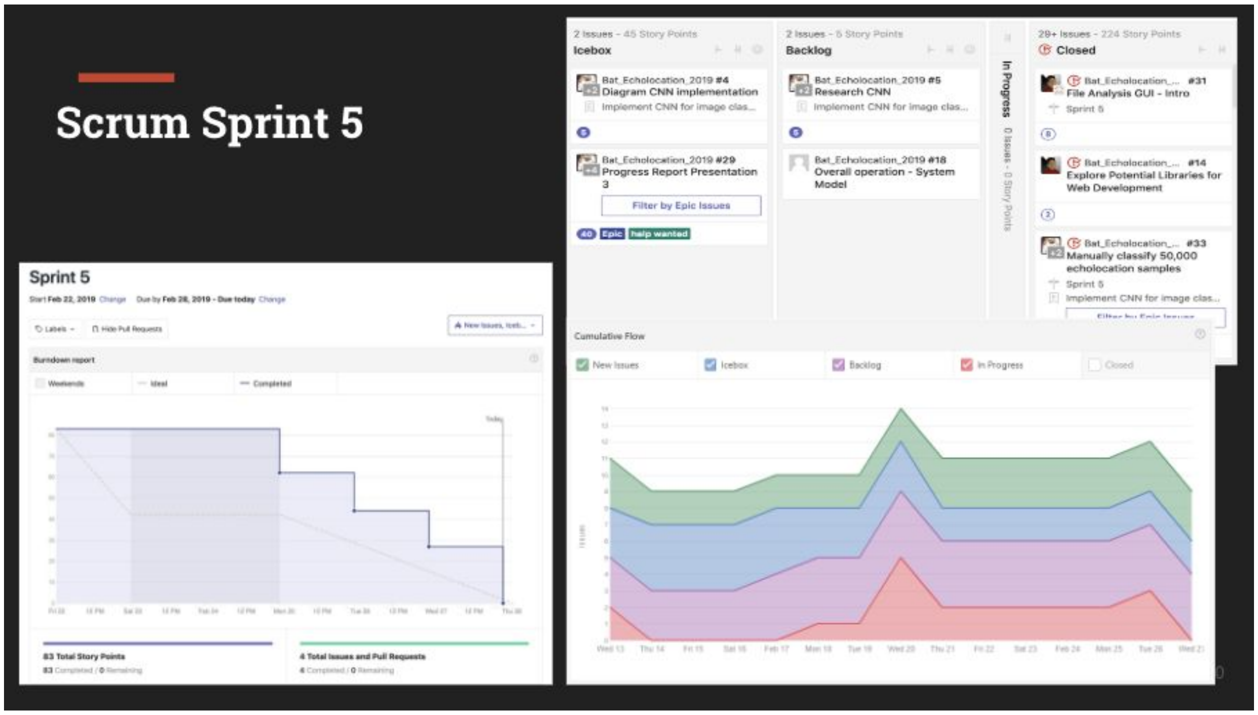
Bat_Echolocation_2019 #18
Overall operation - System Model

Bat_Echolocation_2019 #31
File Analysis GUI - Intro
Sprint 4

In Progress

Bat_Echolocation_2019 #33
Manually classify 50,000 echolocation samples
Implement CNN for image cla
Filter by Epic Issues
30 `Epic`

Bat_Echolocation_2019 #35
Classify 25,000 observatio - 2
Manually classify 50,000 ech
11 `Epic` `help wanted`



Scrum Sprint 5

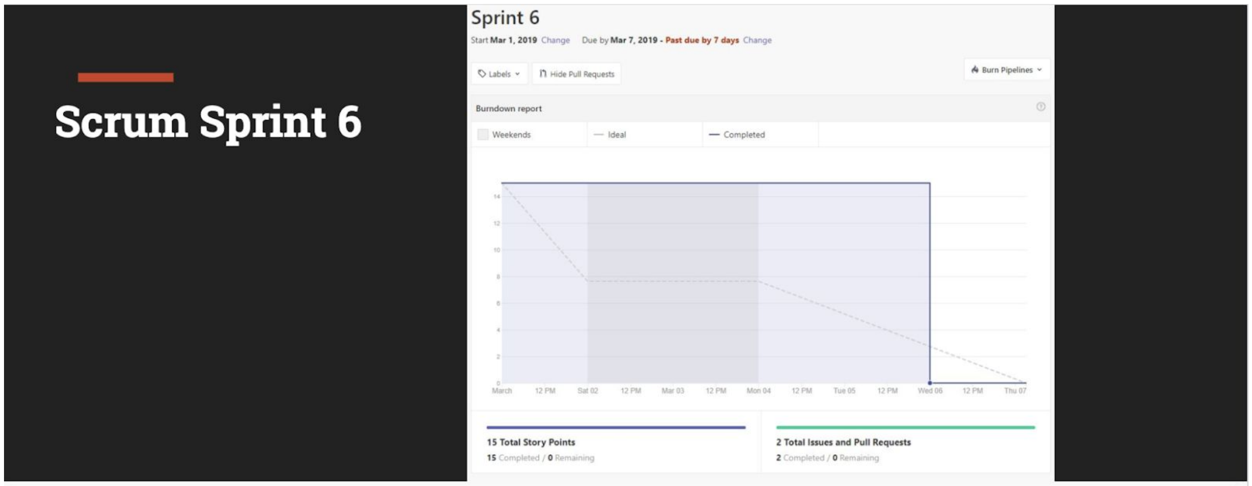
Sprint 5

Start **Feb 22, 2019** Change Due by **Feb 28, 2019 - Due today** Change

83 Total Story Points
83 Completed / **0** Remaining

4 Total Issues and Pull Requests
4 Completed / **0** Remaining

Icebox (2 Issues - 45 Story Points)	Backlog (2 Issues - 5 Story Points)	In Progress (0 Issues - 0 Story Points)	Closed (29+ Issues - 224 Story Points)
Bat_Echolocation_2019 #4 Diagram CNN implementation — Implement CNN for image clas... — 5	Bat_Echolocation_2019 #5 Research CNN — Implement CNN for image clas... — 5		Bat_Echolocation_... #31 File Analysis GUI - Intro — Sprint 5 — 8
Bat_Echolocation_2019 #29 Progress Report Presentation 3 — Filter by Epic Issues — 40 Epic help wanted	Bat_Echolocation_2019 #18 Overall operation - System Model		Bat_Echolocation_... #14 Explore Potential Libraries for Web Development — 2
			Bat_Echolocation_... #33 Manually classify 50,000 echolocation samples — Sprint 5 — Implement CNN for image clas... — Filter by Epic Issues



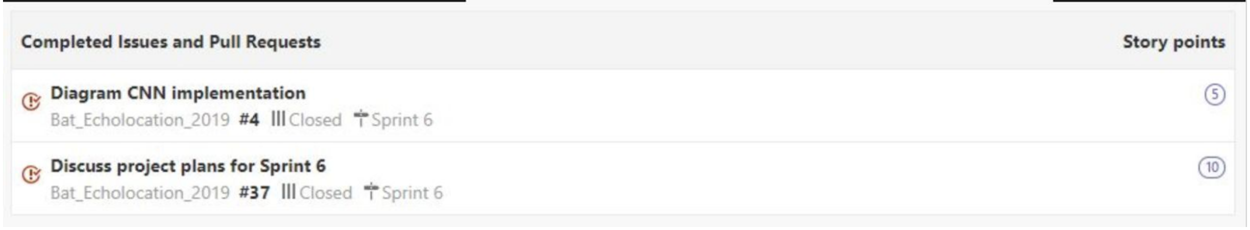
Scrum Sprint 6

Sprint 6

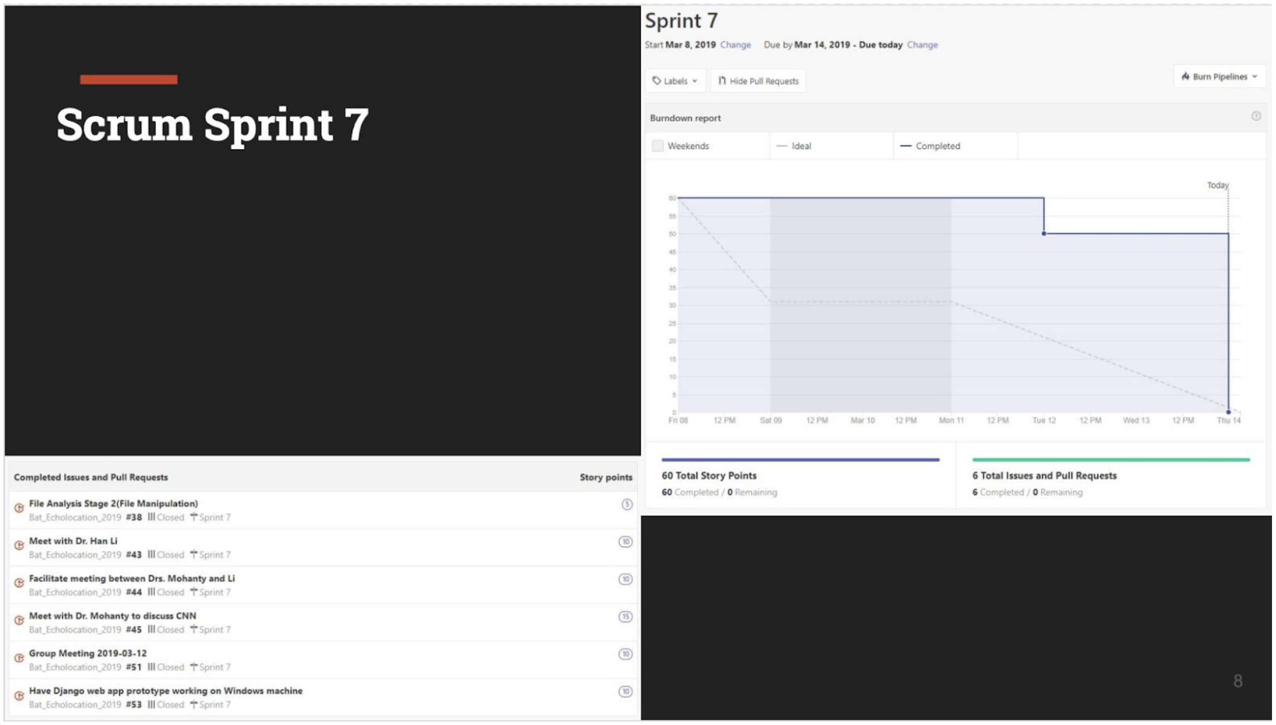
Start **Mar 1, 2019** Change Due by **Mar 7, 2019 - Past due by 7 days** Change

15 Total Story Points
15 Completed / **0** Remaining

2 Total Issues and Pull Requests
2 Completed / **0** Remaining



Completed Issues and Pull Requests	Story points
Diagram CNN implementation Bat_Echolocation_2019 **#4** Closed Sprint 6	5
Discuss project plans for Sprint 6 Bat_Echolocation_2019 **#37** Closed Sprint 6	10



Scrum Sprint 7

Sprint 7

Start **Mar 8, 2019** Change Due by **Mar 14, 2019 - Due today** Change

Labels Hide Pull Requests Burn Pipelines

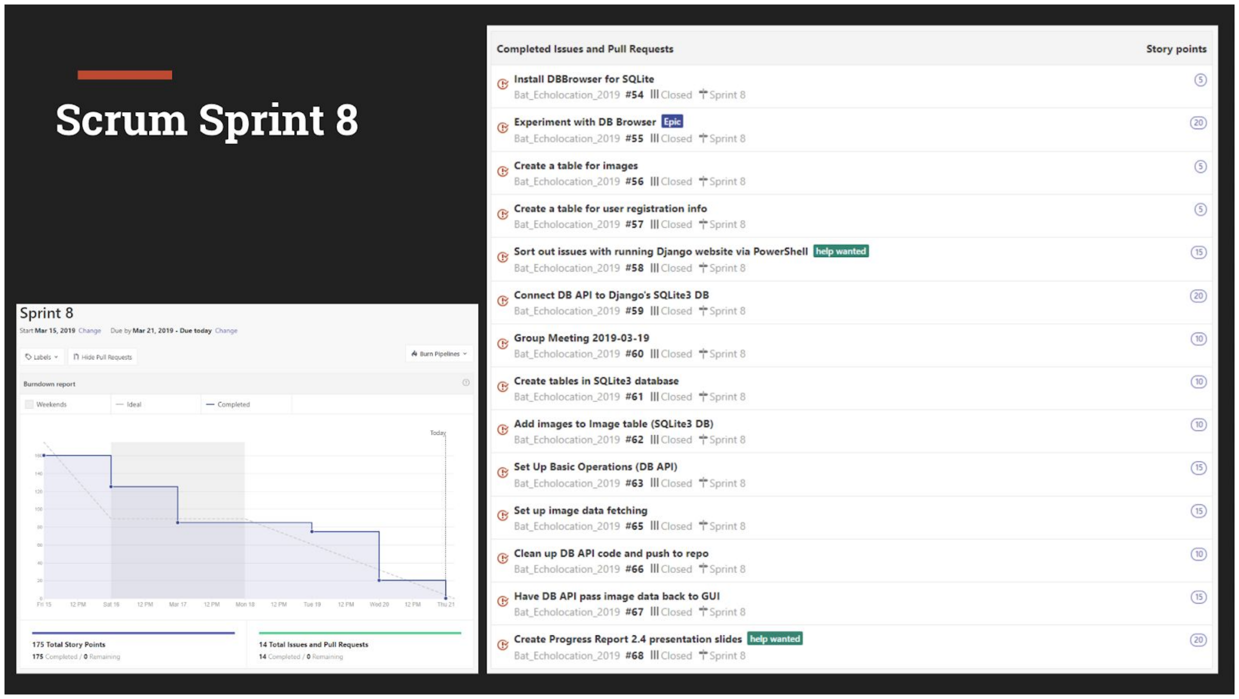
Burndown report

60 Total Story Points
60 Completed / **0** Remaining

6 Total Issues and Pull Requests
6 Completed / **0** Remaining

Completed Issues and Pull Requests	Story points
File Analysis Stage 2(File Manipulation) Bat_Echolocation_2019 **#38** Closed Sprint 7	5
Meet with Dr. Han Li Bat_Echolocation_2019 **#43** Closed Sprint 7	10
Facilitate meeting between Drs. Mohanty and Li Bat_Echolocation_2019 **#44** Closed Sprint 7	10
Meet with Dr. Mohanty to discuss CNN Bat_Echolocation_2019 **#45** Closed Sprint 7	15
Group Meeting 2019-03-12 Bat_Echolocation_2019 **#51** Closed Sprint 7	10
Have Django web app prototype working on Windows machine Bat_Echolocation_2019 **#53** Closed Sprint 7	10

8



Scrum Sprint 8

Sprint 8

Start **Mar 15, 2019** Change Due by **Mar 21, 2019 - Due today** Change

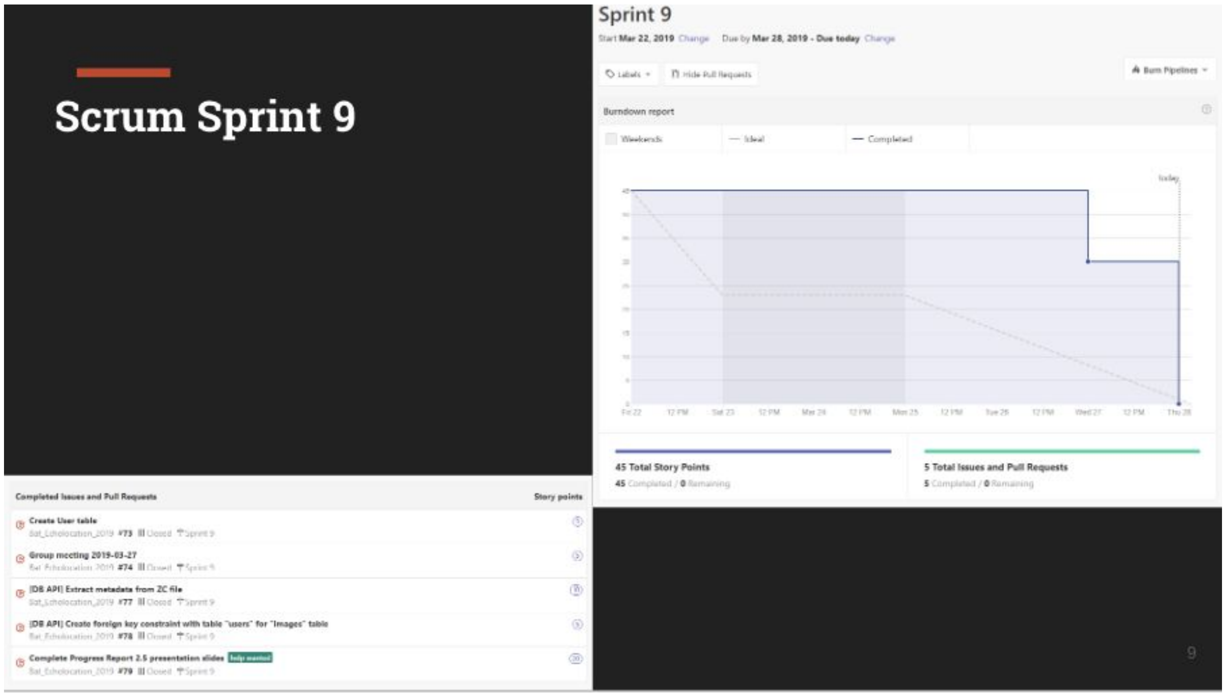
Labels Hide Pull Requests Burn Pipelines

Burndown report

175 Total Story Points
175 Completed / **0** Remaining

14 Total Issues and Pull Requests
14 Completed / **0** Remaining

Completed Issues and Pull Requests	Story points
Install DBBrowser for SQLite Bat_Echolocation_2019 **#54** Closed Sprint 8	5
Experiment with DB Browser Epic Bat_Echolocation_2019 **#55** Closed Sprint 8	20
Create a table for images Bat_Echolocation_2019 **#56** Closed Sprint 8	5
Create a table for user registration info Bat_Echolocation_2019 **#57** Closed Sprint 8	5
Sort out issues with running Django website via PowerShell help wanted Bat_Echolocation_2019 **#58** Closed Sprint 8	15
Connect DB API to Django's SQLite3 DB Bat_Echolocation_2019 **#59** Closed Sprint 8	20
Group Meeting 2019-03-19 Bat_Echolocation_2019 **#60** Closed Sprint 8	10
Create tables in SQLite3 database Bat_Echolocation_2019 **#61** Closed Sprint 8	10
Add images to Image table (SQLite3 DB) Bat_Echolocation_2019 **#62** Closed Sprint 8	10
Set Up Basic Operations (DB API) Bat_Echolocation_2019 **#63** Closed Sprint 8	15
Set up image data fetching Bat_Echolocation_2019 **#65** Closed Sprint 8	15
Clean up DB API code and push to repo Bat_Echolocation_2019 **#66** Closed Sprint 8	10
Have DB API pass image data back to GUI Bat_Echolocation_2019 **#67** Closed Sprint 8	15
Create Progress Report 2.4 presentation slides help wanted Bat_Echolocation_2019 **#68** Closed Sprint 8	20



Scrum Sprint 9

Sprint 9

Start **Mar 22, 2019** Change Due by **Mar 28, 2019 - Due today** Change

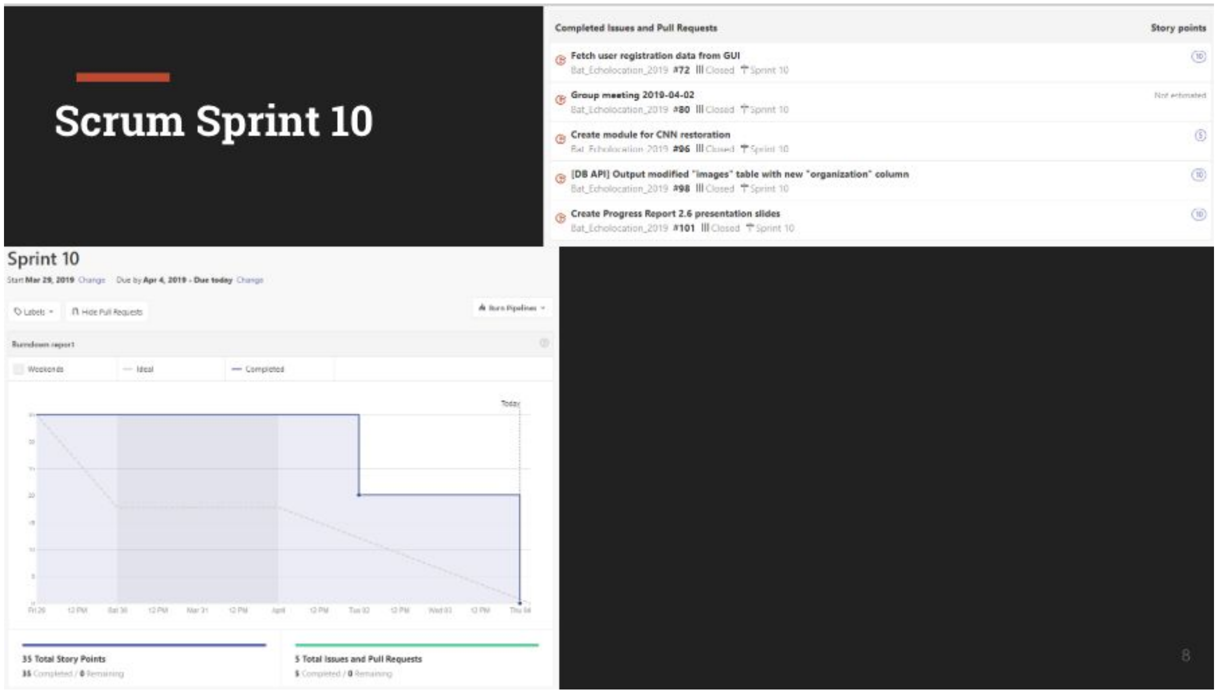
Burndown report

45 Total Story Points
45 Completed / **0** Remaining

5 Total Issues and Pull Requests
5 Completed / **0** Remaining

Completed Issues and Pull Requests	Story points
Create User table Bat_Echolocation_2019 **#73** Closed Sprint 9	5
Group meeting 2019-03-27 Bat_Echolocation_2019 **#74** Closed Sprint 9	5
[DB API] Extract metadata from ZC file Bat_Echolocation_2019 **#77** Closed Sprint 9	10
[DB API] Create foreign key constraint with table "users" for "images" table Bat_Echolocation_2019 **#78** Closed Sprint 9	5
Complete Progress Report 2.5 presentation slides help wanted Bat_Echolocation_2019 **#79** Closed Sprint 9	20

9



Scrum Sprint 10

Completed Issues and Pull Requests	Story points
Fetch user registration data from GUI Bat_Echolocation_2019 **#72** Closed Sprint 10	10
Group meeting 2019-04-02 Bat_Echolocation_2019 **#80** Closed Sprint 10	Not estimated
Create module for CNN restoration Bat_Echolocation_2019 **#96** Closed Sprint 10	5
[DB API] Output modified "images" table with new "organization" column Bat_Echolocation_2019 **#98** Closed Sprint 10	10
Create Progress Report 2.6 presentation slides Bat_Echolocation_2019 **#101** Closed Sprint 10	10

Sprint 10

Start **Mar 29, 2019** Change Due by **Apr 4, 2019 - Due today** Change

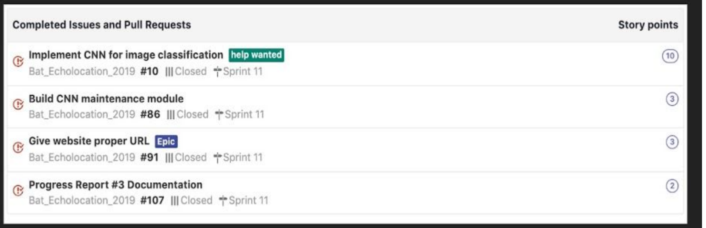
Burndown report

35 Total Story Points
35 Completed / **0** Remaining

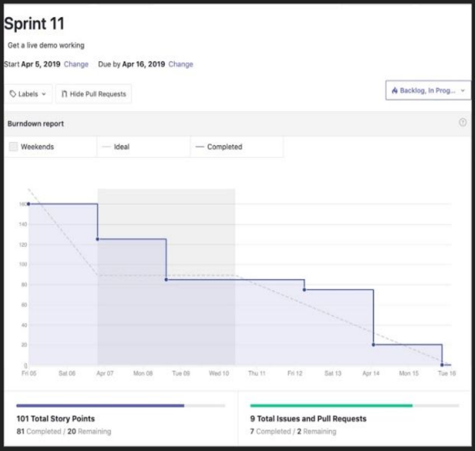
5 Total Issues and Pull Requests
5 Completed / **0** Remaining

8

Scrum Sprint 11



Completed Issues and Pull Requests	Story points
Implement CNN for image classification help wanted Bat_Echolocation_2019 #10 Closed Sprint 11	10
Build CNN maintenance module Bat_Echolocation_2019 #86 Closed Sprint 11	3
Give website proper URL Epic Bat_Echolocation_2019 #91 Closed Sprint 11	3
Progress Report #3 Documentation Bat_Echolocation_2019 #107 Closed Sprint 11	2



Sprint 11

Get a live demo working

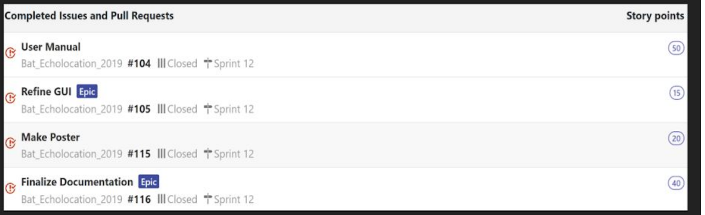
Start **Apr 5, 2019** Change   Due by **Apr 16, 2019** Change

101 Total Story Points
81 Completed / 20 Remaining

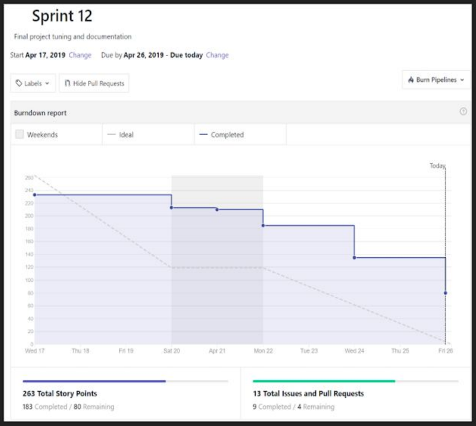
9 Total Issues and Pull Requests
7 Completed / 2 Remaining

15

Scrum Sprint 12



Completed Issues and Pull Requests	Story points
User Manual Bat_Echolocation_2019 #104 Closed Sprint 12	50
Refine GUI Epic Bat_Echolocation_2019 #105 Closed Sprint 12	15
Make Poster Bat_Echolocation_2019 #115 Closed Sprint 12	20
Finalize Documentation Epic Bat_Echolocation_2019 #116 Closed Sprint 12	40



Sprint 12

Final project tuning and documentation

Start **Apr 17, 2019** Change   Due by **Apr 26, 2019 - Due today** Change

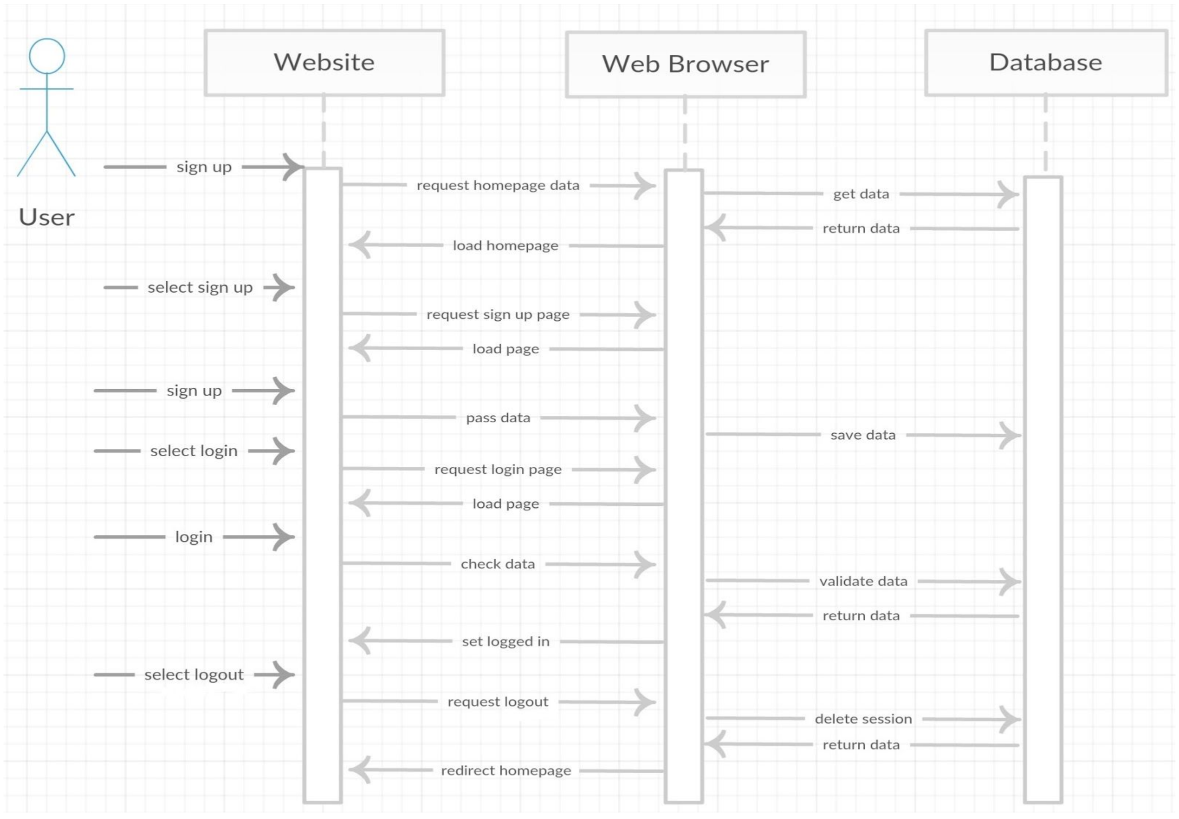
263 Total Story Points
183 Completed / 80 Remaining

13 Total Issues and Pull Requests
9 Completed / 4 Remaining

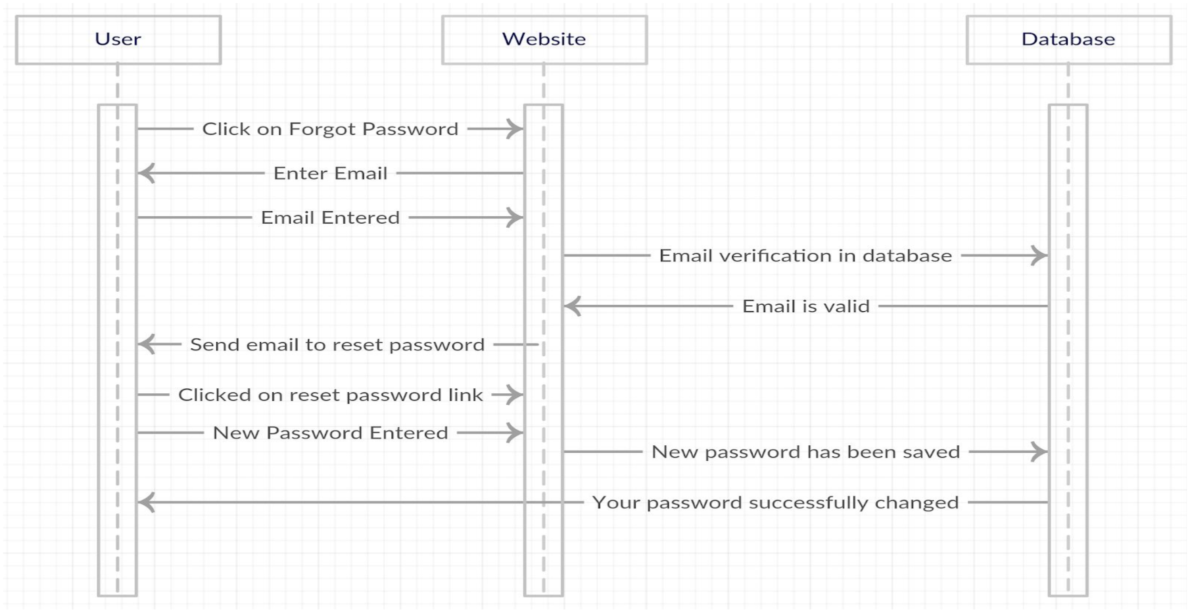
15

7. Subsystems

7.1 User Account Management - Rahim Iqbal

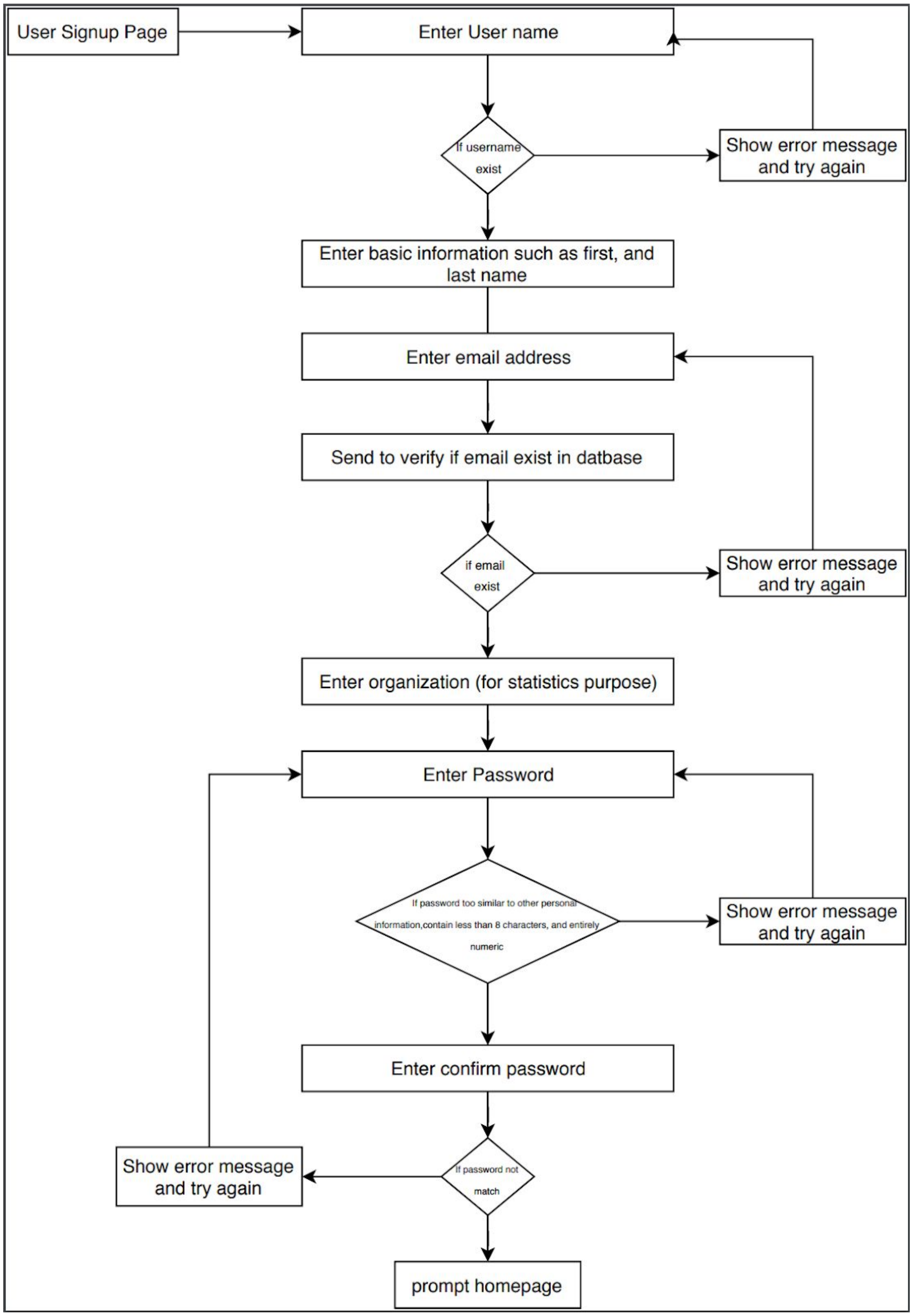


Sequence diagram of User account system



User
Website
Database
Click on Forgot Password
Enter Email
Email Entered
Email verification in database
Email is valid
Send email to reset password
Clicked on reset password link
New Password Entered
New password has been saved
Your password successfully changed

Forget Password Sequence


User Signup Page
Enter User name
If username exist
Show error message and try again
Enter basic information such as first, and last name
Enter email address
Send to verify if email exist in datbase
if email exist
Show error message and try again
Enter organization (for statistics purpose)
Enter Password
If password too similar to other personal information,contain less than 8 characters, and entirely numeric
Show error message and try again
Enter confirm password
If password not match
Show error message and try again
prompt homepage

Signup page diagram

I wrote this subsystem in Python, Javascript, and HTML5 using a functional approach. While the user can be loosely considered an object, it’s more of an active state than a true object.

The first design plan only included the signup page and required a custom SQL table. It turned out that Django comes with its own authorization system, so revisions were required. The second design plan was to create signup, login, and logout feature as shown above in *Sequence diagram of User account system*. The plan was to use Django’s auth_user table to store the user signup information such as username, email, password, etc. For creating the login and logout function we planned to use Djangos built in view functions.

The final design plan included a password reset feature, emailing the user both their username and password. We also added an organization field to user registration for statistical gathering. Finally, we updated the password requirements to make user passwords more secure.

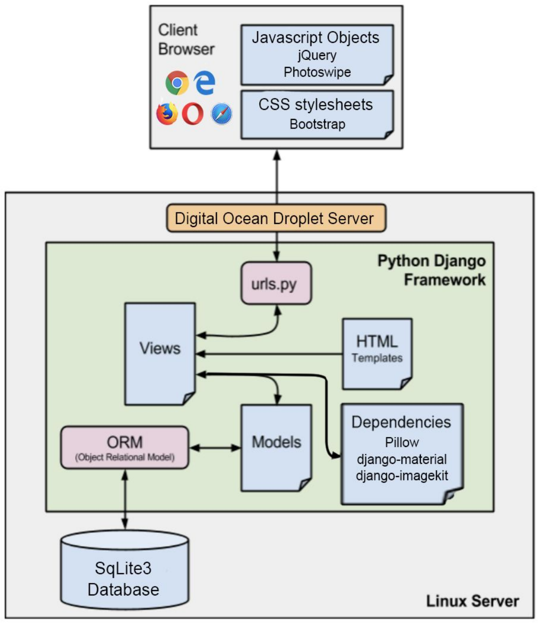
Data dictionary

Table	Column Name	Data Type	Description
auth_user	username	VARCHAR (255)	Store the username of users
auth_user	First name	VARCHAR (255)	First name of the user
auth_user	Last name	VARCHAR (255)	Last name of the user
auth_user	email	VARCHAR (255)	Store the email address. Each user should have different email address
auth_user	password	VARCHAR (255)	User password. Must contain at least 8 characters and cannot be entirely numeric.
organizations	username	VARCHAR (255)	Foreign key from auth_user table.
organizations	organization	VARCHAR (255)	User’s organization. Used for statistics

When testing, I always made sure that my code worked perfectly on my local machine before pushing to GitHub. I double checked that, when a new account is registered, the information was saved to the DB and that the user is redirected to the homepage.

User training: If you don’t have an account, click signup to create one. If you do, click login to log in.

7.2 Front End - Thien



Initial design and model

The framework we picked is Django since it is a Python web based programming language. The design is fairly simple as our first priority is to let the user use our tool to analyze their ZC data

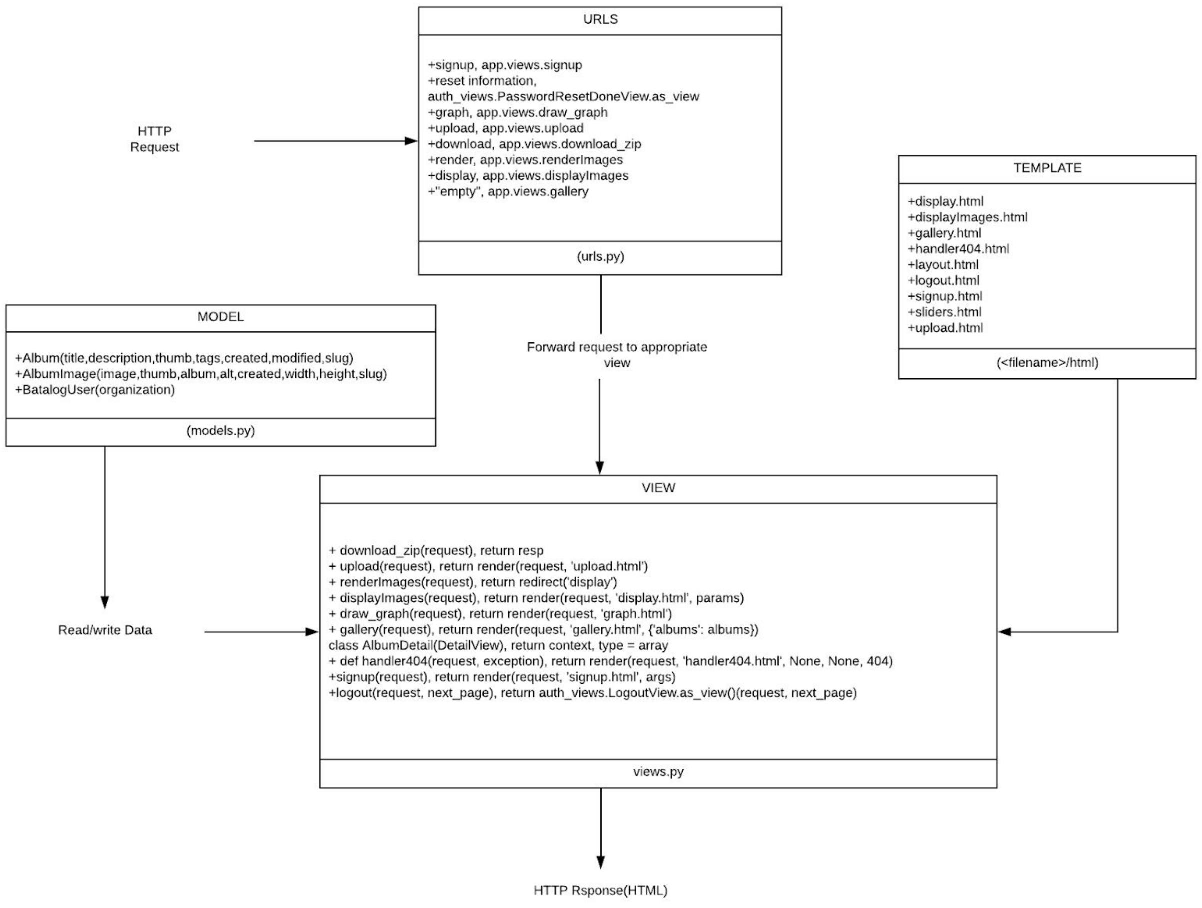
The front end is composed of templates (HTML5/CSS/Javascript), plus the views functions (Python 3)

Once the front end was completed the next objective was to connect to the back ends. These back end or functional modules are Display Images, Download as Zip, and Graph. Please refer to 7.3 to find out more information about these functions.

The design choice for this front end the Django framework. Django takes a functional approach to its design and so do I. Most of the implementations are similar to the Django documentation. There are many reasons that Django is the to do web platform we use for this project.

- Django is a Python web framework which most of our group members are familiar with. This allows all of us to develop the website anywhere rather be restricted to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X.
- Versatile - Django is very well known and have been used on almost any type of website so it was a good opportunity to pick it up.
- Secure - Django user and admin management have to be one of the best functions that it provides to the developer. It also has built in clickjacking, cross-site scripting, SQL injection.

Before we make any final commit and push to the repo we would always to make sure it works perfectly on my local machine first. Another way we carry out the testing process to have a copy of the repo and test it before committing.

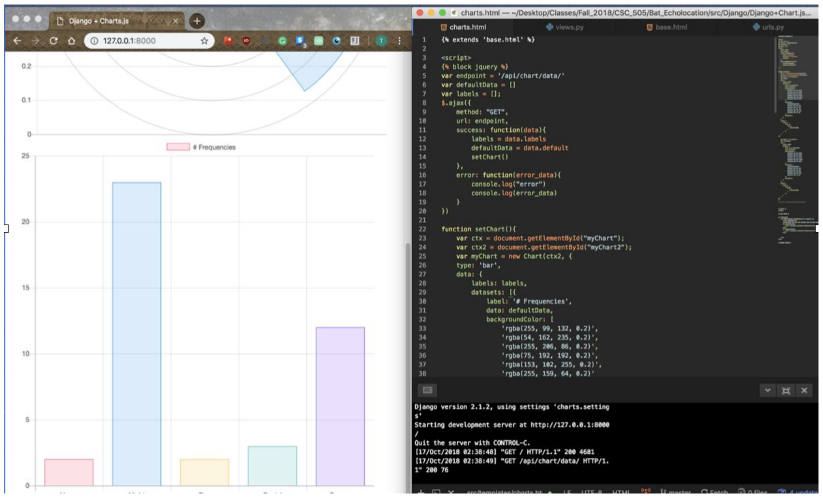

URLS
+signup, app.views.signup
+reset information,
auth_views.PasswordResetDoneView.as_view
+graph, app.views.draw_graph
+upload, app.views.upload
+download, app.views.download_zip
+render, app.views.renderImages
+display, app.views.displayImages
+"empty", app.views.gallery
(urls.py)
HTTP Request
TEMPLATE
+display.html
+displayImages.html
+gallery.html
+handler404.html
+layout.html
+logout.html
+signup.html
+sliders.html
+upload.html
(<filename>/html)
MODEL
+Album(title,description,thumb,tags,created,modified,slug)
+AlbumImage(image,thumb,album,alt,created,width,height,slug)
+BatalogUser(organization)
(models.py)
Forward request to appropriate view
VIEW
+ download_zip(request), return resp
+ upload(request), return render(request, 'upload.html')
+ renderImages(request), return redirect('display')
+ displayImages(request), return render(request, 'display.html', params)
+ draw_graph(request), return render(request, 'graph.html')
+ gallery(request), return render(request, 'gallery.html', {'albums': albums})
class AlbumDetail(DetailView), return context, type = array
+ def handler404(request, exception), return render(request, 'handler404.html', None, None, 404)
+signup(request), return render(request, 'signup.html', args)
+logout(request, next_page), return auth_views.LogoutView.as_view()(request, next_page)
views.py
Read/write Data
HTTP Rsponse(HTML)

Data dictionary

- This Front End illustrates the following concepts on top of the Django Framework:
  - Using django-material and materializecss for building Django UI.
  - Using django-imagekit for building resizing images.
  - Using photoswipe javascript library for more rich image gallery user experience.
- Javascript Objects
  - jQuery - a library that helps navigate through documents, create animations and handle events easier.
  - Photoswipe - an image gallery for web development.
- CSS stylesheets
  - Bootstrap - responsive grid system, extensive pre built components
- Digital Ocean Droplet Server
  - A scalable compute platform with add-on storage, security, and monitoring capabilities to easily run production applications.
- Python Django Framework

        - ORM - can be used to interact with application data from various relational databases such as SQLite, PostgreSQL and MySQL.
        - Dependencies
            - Pillows - Python Imaging Library.
- SqLite3 Database
    - An embedded database, server-less and can run within our app.

Refinements



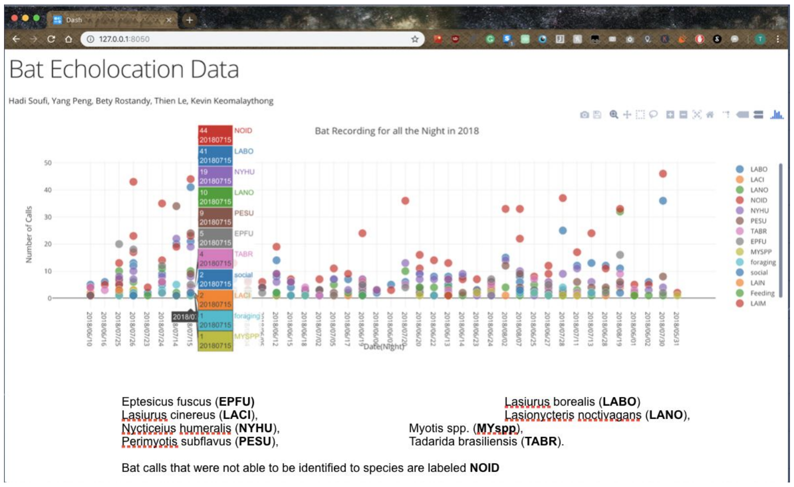
Django + Charts.js
127.0.0.1:8000
# Frequencies
charts.html — ~/Desktop/Classes/Fall_2018/CSC_505/Bat_Echolocation/src/Django/Django+Chart.js...
charts.html
views.py
base.html
urls.py

```
{% extends 'base.html' %}

<script>
{% block jquery %}
var endpoint = '/api/chart/data/'
var defaultData = []
var labels = [];
$.ajax({
    method: "GET",
    url: endpoint,
    success: function(data){
        labels = data.labels
        defaultData = data.default
        setChart()
    },
    error: function(error_data){
        console.log("error")
        console.log(error_data)
    }
})

function setChart(){
    var ctx = document.getElementById("myChart");
    var ctx2 = document.getElementById("myChart2");
    var myChart = new Chart(ctx2, {
    type: 'bar',
    data: {
        labels: labels,
        datasets: [{
            label: '# Frequencies',
            data: defaultData,
            backgroundColor: [
                'rgba(255, 99, 132, 0.2)',
                'rgba(54, 162, 235, 0.2)',
                'rgba(255, 206, 86, 0.2)',
                'rgba(75, 192, 192, 0.2)',
                'rgba(153, 102, 255, 0.2)',
                'rgba(255, 159, 64, 0.2)'
```

```
Django version 2.1.2, using settings 'charts.setting
s'
Starting development server at http://127.0.0.1:8000
/
Quit the server with CONTROL-C.
[17/Oct/2018 02:38:48] "GET / HTTP/1.1" 200 4681
[17/Oct/2018 02:38:49] "GET /api/chart/data/ HTTP/1.
1" 200 76
```

First Take on the Project: Django + Chart.js

This is the first take on displaying bat echolocation data. The benefits of using Chart.js is that is very straightforward and easy to use. However, it is very hard to incorporate with the zero crossing file analysis since it will only read data from csv format. There were no changes to the initial since this is the first implementation from the barebone django beginner project.

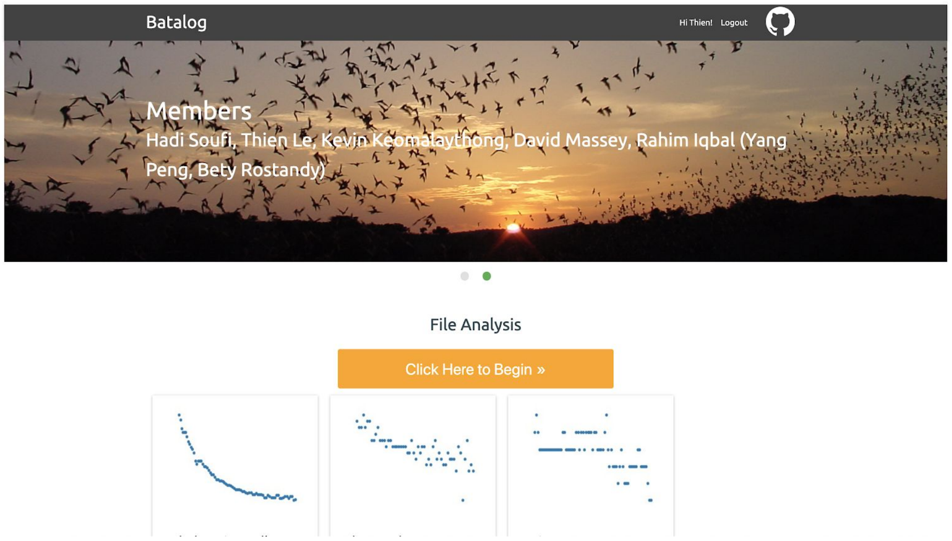


Eptesicus fuscus (**EPFU)**
Lasiurus cinereus (**LACI**),
Nycticeius humeralis (**NYHU**),
Perimyotis subflavus (**PESU**),
Lasiurus borealis (**LABO**)
Lasionycteris noctivagans (**LANO**),
Myotis spp. (**MYspp**),
Tadarida brasiliensis (**TABR**).

Bat calls that were not able to be identified to species are labeled **NOID**

Second take on the Project: Django + Dash

The second refined on this project was with Dash, a productive Python framework for building web applications. Written on top of Flask, Plotly.js, and React.js, Dash is ideal for building data visualization apps with highly custom user interfaces in pure Python. Dash is amazing at forming visualization on the data that have been fed into the views.py. However, this was refined since our focus on the project has changed. We wanted to focus more of the mass file analysis rather than just individual points on each graph.



Members
Hadi Soufi, Thien Le, Kevin Keomalaythong, David Massey, Rahim Iqbal (Yang Peng, Bety Rostandy)

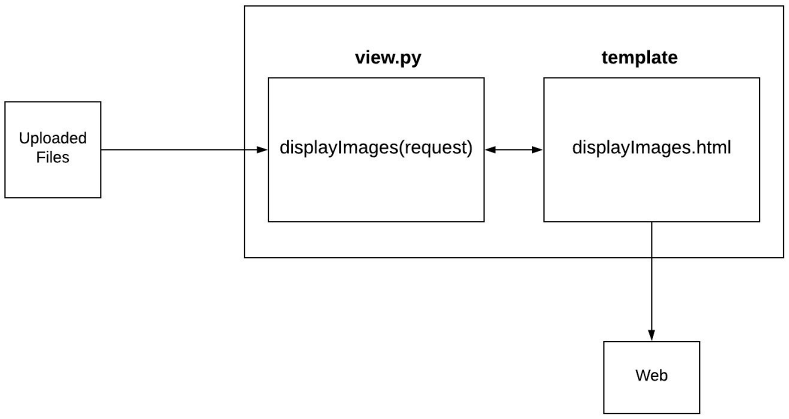
With the focus been altered that was mentioned above, this is now our latest refinement to the website. We're now focused on providing tools for the end user. The main tool is composed of classification and CNN analysis, and allows the user to submit any zero crossing file or zip file. The user can then download and view the results.

7.3 Functional Modules

Display Images - Thien

This sub-module's goal is to display the uploaded images into a user-friendly output where they can see the results on the web. Most of the implementation is from *views.py* and *displayImages.html.* The method *display_images* takes in the files from the uploaded section and return that to the template. The *display_images.html* then reformats the returned images into two columns, one for abnormal and the other echolocation.

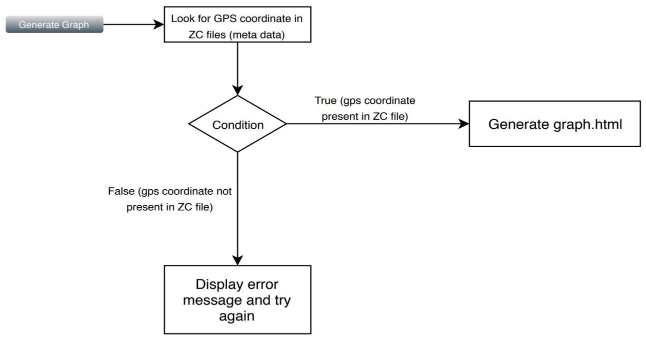
DisplayImages Sub-module



Mapping - Rahim

The first design plan was to generate a plotly graph using GPS coordinates taken from ZC metadata. The second design plan was to add an exception in case the zero crossing file contains no GPS coordinates. That way it will lead to error message and ask to try again.

Generate graph Sub-module

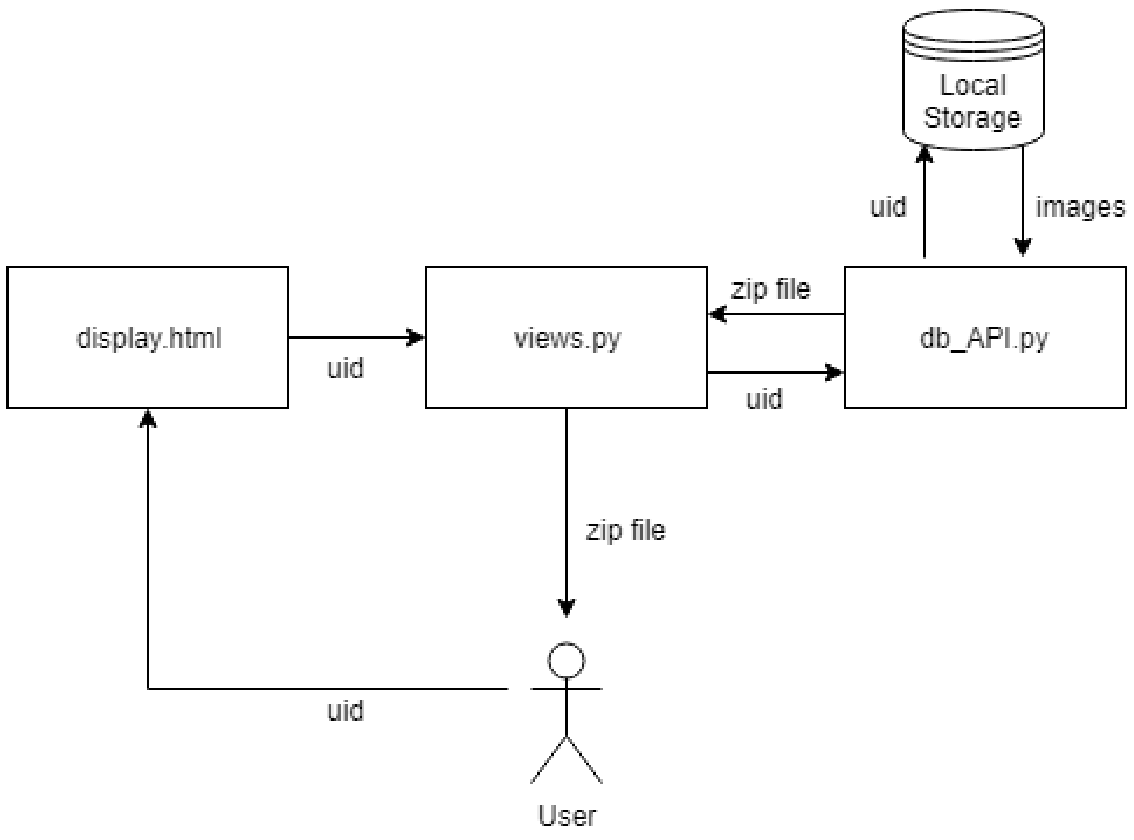

Generate Graph
Look for GPS coordinate in ZC files (meta data)
Condition
True (gps coordinate present in ZC file)
Generate graph.html
False (gps coordinate not present in ZC file)
Display error message and try again

The GPS graph was tested on a local machine using a random GPS coordinates. After making sure that the GPS graph working perfectly in local machine, I pushed the code to repo.

- Data dictionary
    - Table: Images
    - When the user upload ZC files it will generate data inside metadata column. The data include several things from the files such as datetime, species, gps position/coordinates, etc. The graph look for gps position/coordinate in metadata column. If there isn't any, the page will error out

User training: After your calls have been processed, simply press 'Generate Map' to see a map of all your calls. If no GPS information is associated with your data, then you may return to Display.

Download as Zip - Hadi



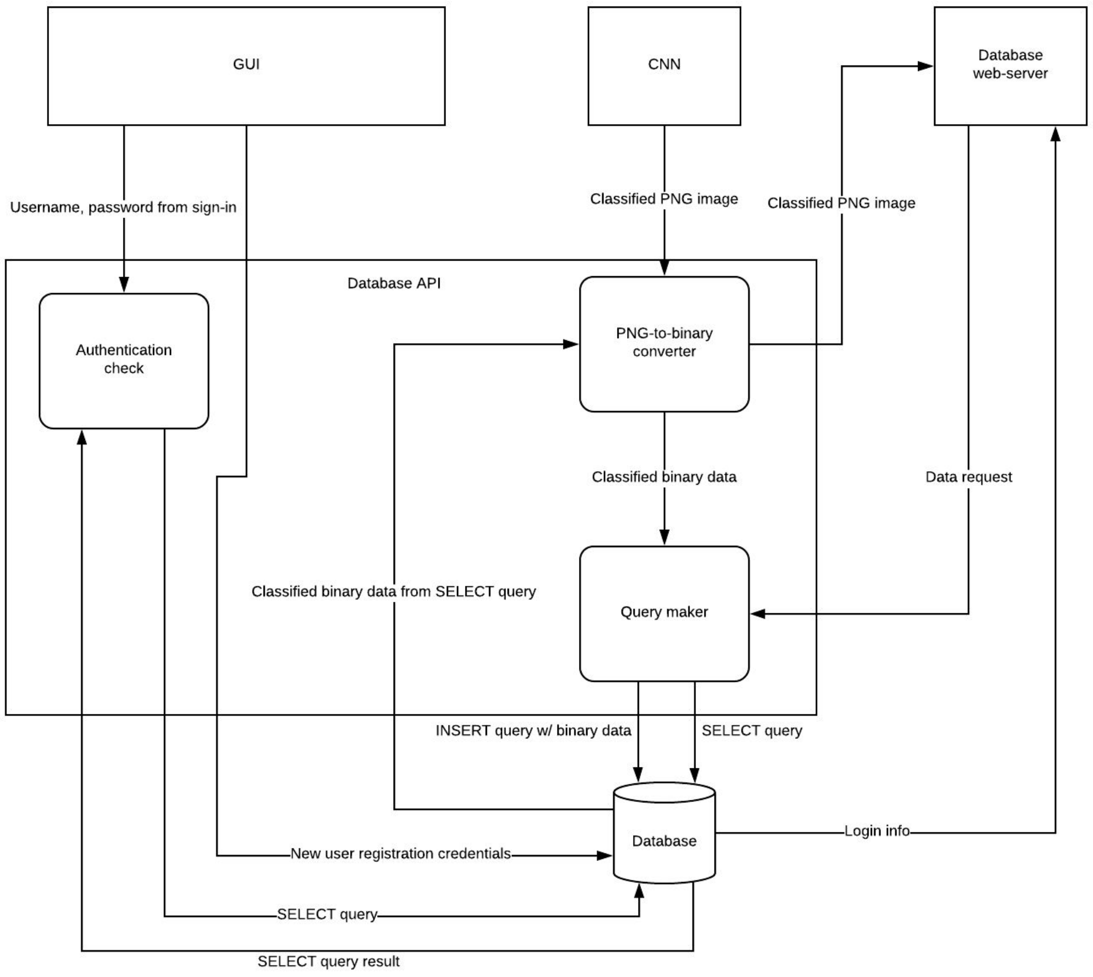
I chose a functional approach for this system. The user presses a button in display.html which sets off a chain of function calls ending in db_API.py. The zipping function in db_API.py was written in Python 3. The frontend was written in HTML with Django.

Testing was very straightforward. If I could successfully download a zip file, no errors were thrown, and no files were leftover, I considered it successful.

User training: After you’ve uploaded your data, you may want to download it for further analysis. Simply press the ‘Download as Zip’ button on the display page to receive a zip file with all of your pulses sorted by type.

7.4 Database API - Kevin

The initial design of the database API is shown below.

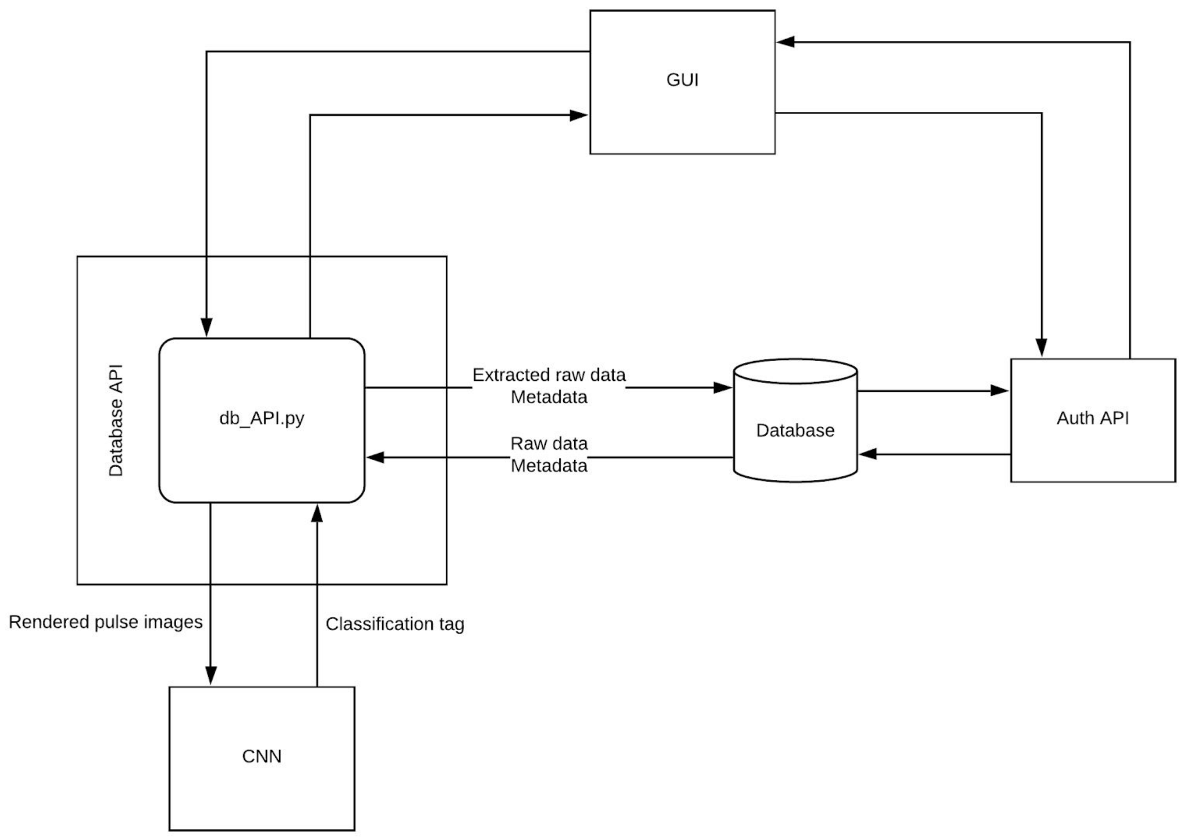


This design consists of three sub-modules: a PNG/binary converter, a query handler, and a user authentication handler. The converter would convert PNG to binary and vice-versa. The query handler would create SQL queries based on GUI data. The user authentication handler would perform two tasks:

- If a user creates an account, their registration information is sent to the query handler and inserted into a special table in the database.
- If a user tries to login, their request is checked against the “users” table and either authorized or denied depending on whether a match is found.

This design was expanded from a previous design of the database API used for the Fall 2018 Data Science version of this project. However, later in development the user authenticator was cut from the design in favor of Django’s built in system, and so the

API will be modifying its own user authentication table. The updated design is shown below.



This subsystem was developed using a functional approach. The entire API consists of functions that prepare the data to be inserted into the database and convert ZC data into human-readable format. It uses Python's native SQLite3 library to support these operations. Said library supports a limited version of the SQL language, but is used here because the Django framework uses an SQLite3 database for local storage.

The database management itself is only concerned with two tables, “pulses”, where the raw pulse data is stored, and “auth_user”, Django's own user authentication table. The data dictionary for the "pulses" table is as follows.

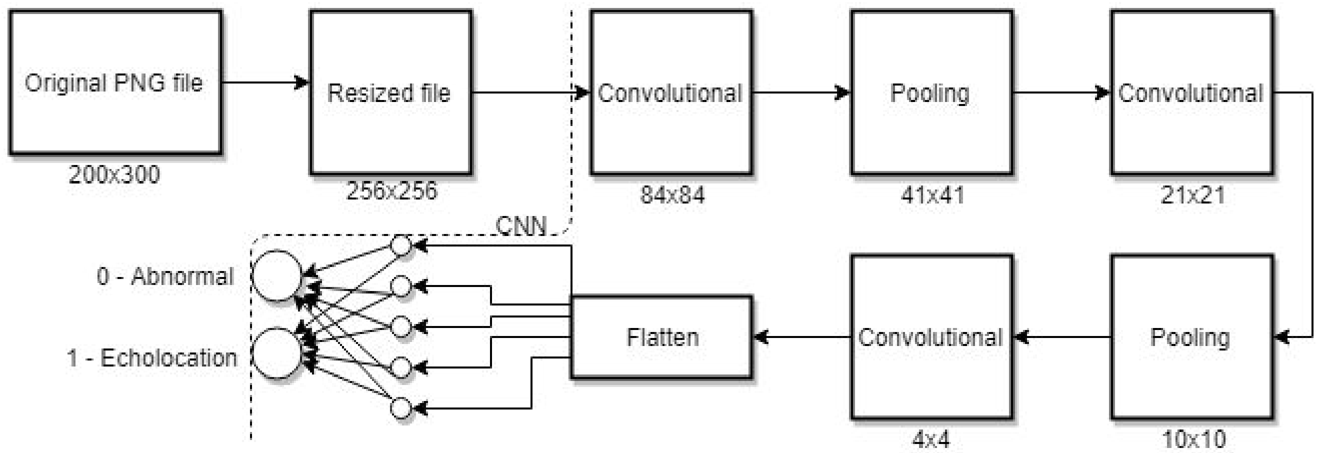
pulses			
Name	**Data Type**	**References**	**Description**
name	VARCHAR		Name of the ZC file with pulse number

raw	BLOB		Raw pulse data in x/y coordinates
classification	VARCHAR		Classification tag for a pulse; 0 for “echolocation”, 1 for “abnormal”
metadata	VARCHAR		Data about the series of pulses, including date, timestamp, GPS coordinates, etc.
uid	INT		UID of the user that uploaded the file
name	VARCHAR	auth_user	Name of the ZC file with pulse number

The data dictionary for the "auth_user" table borrows from that of the user authentication subsystem.

Because the database API is more focused on the back-end, user training for this specific subsystem would be unnecessary. As for testing, it pertains more to the development of the database API and ensuring that its connections with other subsystems go through without any errors.

7.5 CNN - David



The first CNN design plan was to take other CNN designs and implement them. The result was low accuracy on the files tested and high training losses. The belief was that all CNN's were swappable with one another. Further research proved this incorrect.

Future CNN design plans were to make the layers based on the convolutional and pooling layer formulas below (Refined Model Analysis). The first design using those formulas was a CNN that resized images to 256x256 from the original 200x300. The squares represent the layers and the output image dimensions are listed below them.

- Other CNN designs were constructed by resizing images to 216x324, 216x288, 250x250, and 512x512 in separate models.

Refinements

Models using resized images

- Pro: Using the formulas for convolutional and pooling layers, the layers can be determined. The layers for CNN's that use image sizes of 216x324, 216x288, 256x256, and 512x512 can be made from the factors which divide into these dimensions. More layers mean more features that may be detected by the CNN which may increase accuracy.
- Cons:
    1. Resizing removes or adds pixels to the images which may affect accuracy of the CNN.
    2. Convolutional layers require filters to have odd dimensions. For a model that requires images of 256x256 or 512x512, with both sizes being a power of 2, different filter sizes have to be calculated. Since output image sizes have to be whole numbers, sometimes the only stride available is one.

Models using original images

- Pro: Images are unaltered which eliminates Con 1 above.
- Con: It's difficult to calculate the layers of the dimensions 200x300 without using their factors. Both are divisible by 100 which makes their last dimensions to be 2x3 if only factors of 100 are used. If more layers are to be added, other values, other than the factors, will force some layers to have a stride of one. Two extended 200x300 models were planned and tested which made the last image dimension to be 2x7 and 1x26. The accuracies of both were lower than the model that used only the factors of 100.

Changes from initial model

From the diagram in section 4, the Test/Train CNN section was added to reflect the use of external code to make a CNN model and incorporate a way to manually or automatically retrain it.

CNN's that required the original images to be resized were designed on paper and a few were tested. After testing, a CNN that didn't require any resizing had higher accuracy and was used.

Refined model analysis

The following formulas were used:

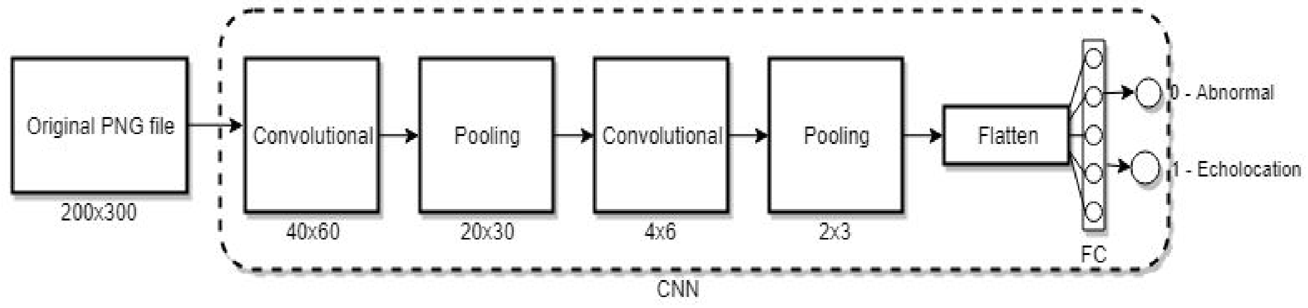
Convolutional Layer:

Output size=Dimension size - Filter size+2(Padding)/Stride

Pooling Layer:

Output size=Dimension size - Filter size/Stride

The diagram below shows the layers along with their output image sizes. The layers of the CNN that are used in training are also present in the final model. Each image consists of three dimensions: height, width, number of channels. The convolutional layer filter size is required to be an odd number. The convolutional layers use a filter size of five, and the pooling layers use a filter size of two. These represent factors of 100. A dropout of 50% was added after all convolutional layers and to the first fully connected layer. The use of dropout is a regularization tool that minimizes and delays the presence of overfitting but is not used in the final model to process images. Dropout in training extends the region of low training losses and provides a region to use for models before the training starts to overfit. Following the dropout is an activation layer. Batch normalization was tested but provided high losses and low accuracy. The number of filters starts with 96 in the first convolutional layer and increases to 768 in the first fully connected layer. This process adds more types of filter grids to use for feature detection. The number of filters may be used as an added dimension to describe the images in the layers.



Refined Diagram

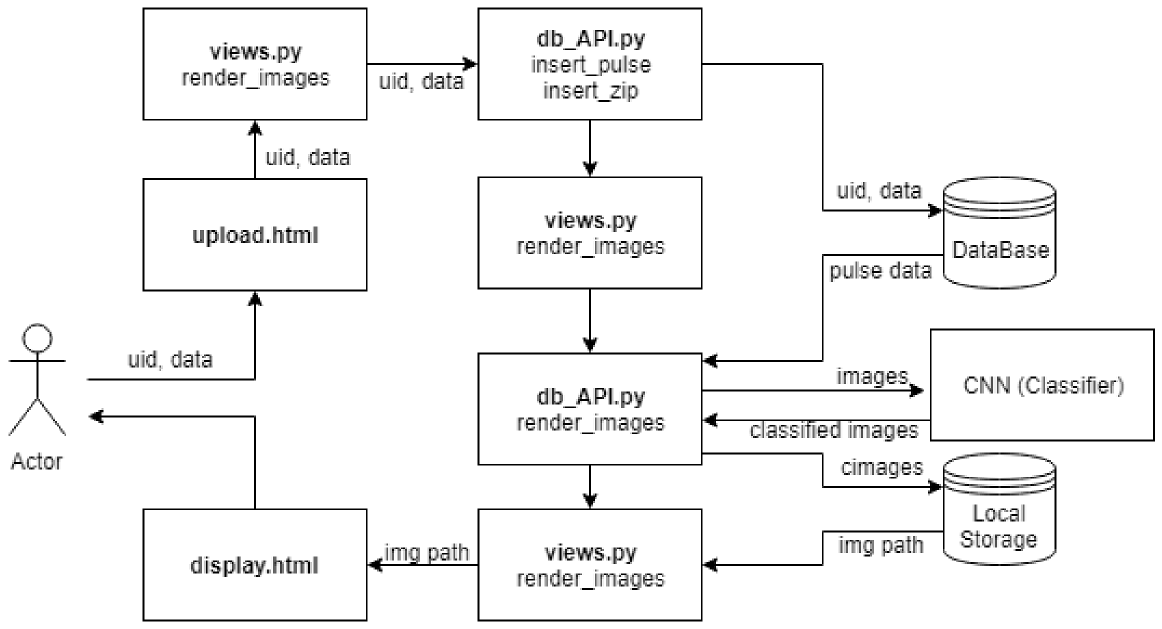
Original images are of dimensions 200x300 with the number of channels being 3 or RGB. The following describes the layers using the formulas above:

1. Convolutional: Filter size: 5 Stride: 5 Number of filters: 96
2. Pooling: Filter size: 2 Stride: 2
3. Convolutional: Filter size: 5 Stride: 5 Number of filters 128
4. Pooling: Filter size: 2 Stride: 2
5. Flatten
6. Dense fully connected: 768 neurons
7. Output dense fully connected: 2 neurons representing 2 classes. The output is a float that is 0 for abnormal call images and 1 for echolocation call images.

Name	**Data Type**	**Description**
CNN model	H5	The CNN is created and saved while training using the h5py library. This represents the CNN layers, weights, and parameters.
Images	PNG	Each image is of dimension 200x300 with 3 channels representing RGB. The pixels are rescaled to 0-255 for training and predictions. Each image is the result of extraction from a ZC file.

I wrote this in Python and took a functional approach. Testing consisted of training models and testing them with all validation abnormal and echolocation images. The Modelcheckpoint function provided models to test at various points in training. Matplotlib provided graphs to visualize training results. The goal was to test models that had low training losses and high accuracy as reported by Keras. The steps-per-epoch and validation steps parameters are known to be calculated from the formula: Number of images/batch size. Batch size is adjustable by the tester. Lower batch sizes were used to minimize memory errors and maximize speed of the epochs. The tradeoff is the number of epochs required to process all images increases with a smaller batch size. The valley of low training losses is extended in addition to using a higher dropout and smaller learning rate. The number of epochs required to process all training images was between 1.5 to 3 while the validation images required one. A copy of the abnormal training files were flipped, and echolocation training files were rotated two degrees and added to the files to train. The highest abnormal file accuracy was 91%, and echolocation accuracy was 98% using the same model.

7.6 Bat Call Extraction- Hadi



Initially, we rendered the images and stored them in the database. However we realized that this was inefficient and would cause our database usage to rapidly bloom. Instead, we moved rendering the images to a user session and now delete them after it ends.

Another refinement is that we now only store the x/y coordinates of a ZC file, which is far more space efficient. The first model had a time complexity of O(n) + u, where n is the number of inserted images and u is the number of times a user logs in multiplied by the time complexity of each login. Our new model has a time complexity of O(n) * u.

We also incorporated classification into db_API.render_images. Immediately after it's rendered, it's sent to the classifier. Calling this a refinement or update to the original design is a bit of a stretch, since we always knew it was going to be incorporated into the system somehow. We didn't know exactly where it was going to go, so it wasn't explicitly designed into the system.

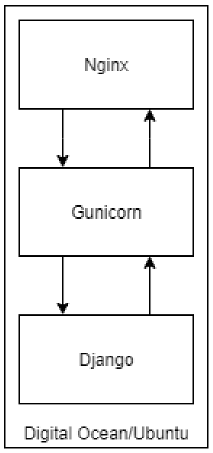
The other major update to the model was adding a handler for zip files. All it does is iterate through the zip and call insert_pulses on each file.

I chose a functional approach to this module. All of the functions were written in Python 3, the frontend was written in HTML with Django, and the database calls were written in SQL.

Testing was time consuming, as there were several features that all needed to be debugged. I started from the layer farthest away from the user and worked my way out. The functions farthest from the user were both the most complicated and the most isolated, so I wrote them in Jupyter Notebook and tested them thoroughly. Then I tested each function in views.py and finally tested the HTML buttons. I considered my module complete if it received ZC data, put it into the DB, turned it into images, separated the data by user, and displayed the data after completion.

User Training: Once you've created an account and logged in, you will see a button labeled 'Click Here to Begin' on the home page. Clicking it will take you to the Upload page, where you can select your files. Currently we accept zero crossing and .zip files. After pressing 'Upload', our system will process your files. Depending on how much data you give us, it may take several minutes to process everything. When it's all done, you'll be redirected to the display page, where all of your data will be classified.

7.7 Bring the website online - Hadi



We didn’t make a diagram of this initially since none of us had any experience with making a live website before. We knew it was possible & that our project didn’t have any unique requirements, so we decided to wing it and see what happened.

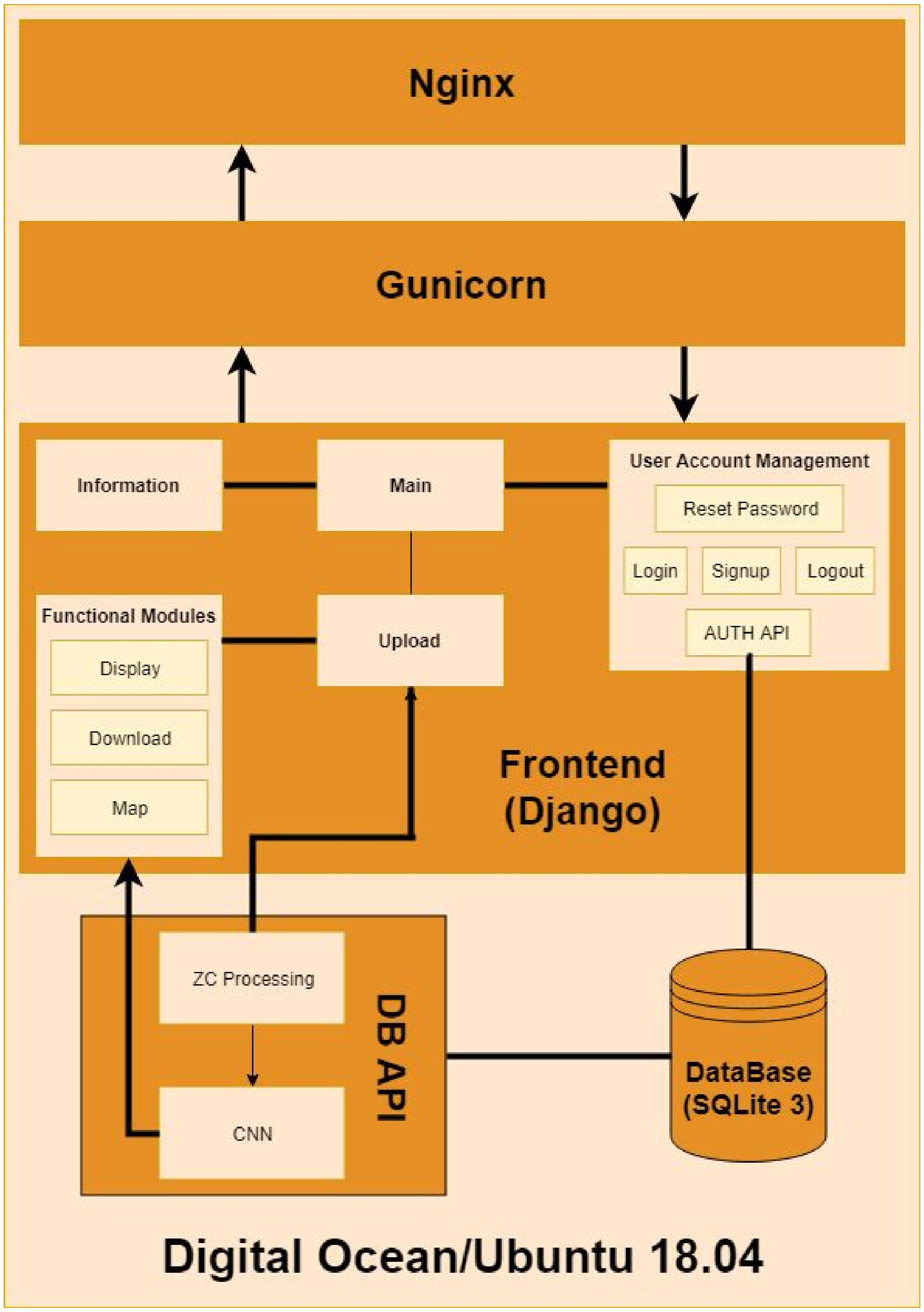
We chose Digital Ocean to host our website because it’s affordable, simple, and scalable. As a bonus, it provides a DNS hosting service. After loading the project onto Digital Ocean, I experimented with WSGI services and settled on Gunicorn. I chose Nginx as our web server because Digital Ocean has a nice tutorial for setting it up.

This is an object oriented approach, but I didn’t do any coding for this part. Most of my work involved fiddling around with the various modules and trying to get all the settings right.

I tested my setup by building it from the inside out. I started by bringing Django online in a virtual environment to make sure there was nothing wrong with the project & that my Digital Ocean configurations were correct. Then I brought Gunicorn into the mix and tested to see if it worked, and finally Nginx. I considered this module complete if we had a domain name and all other modules worked normally while using it.

8. Complete System

Batalog
GitHub Repository
User Manual



Team Member Descriptions

- Hadi Soufi
    - I am the project lead, I was responsible for directing the project and maintaining contact both within the group & with outside entities.
    - I built the ZC pulse processing module, zip generator, prototyped the mapping tool, brought the system online, and implemented various quality of life features.
- Thien Le
    - I am the Scrum Master of the group who responsible with organization meetings and make sure we carried out our sprint recap and retrospective meetings.
    - I am responsible for the overall look and feel of the website. The website should be easy to use and user-friendly. Additionally, I picked up another subsystem that displays rendered and classified bat calls.
    - I am responsible for most of the graphic related works such as making the poster and presentation slides. While everyone contributed equally to the content of these documents, I made sure that everything had a high quality look and feel for our audience.
- Kevin Keomalaythong
    - I am the person responsible for developing the database API. My work has been focused on the back-end side of the project, which involves communication between the API and the database as well as other back-end modules. I develop the API while ensuring that it is able to receive data from the GUI and output data to the CNN.
- David Massey
    - I am responsible for the training and testing of the CNN. I explored many options for parameter settings and structure. The goal was to make it as accurate and efficient as possible. I used the CNN to make classifications of images and output the results.
- Rahim Iqbal
    - I am responsible for creating a user account management system. I made sure that the user can register for the website and all the information is stored into the database without any complication. I also ported the call mapping application to HTML and made sure that it failed gracefully.

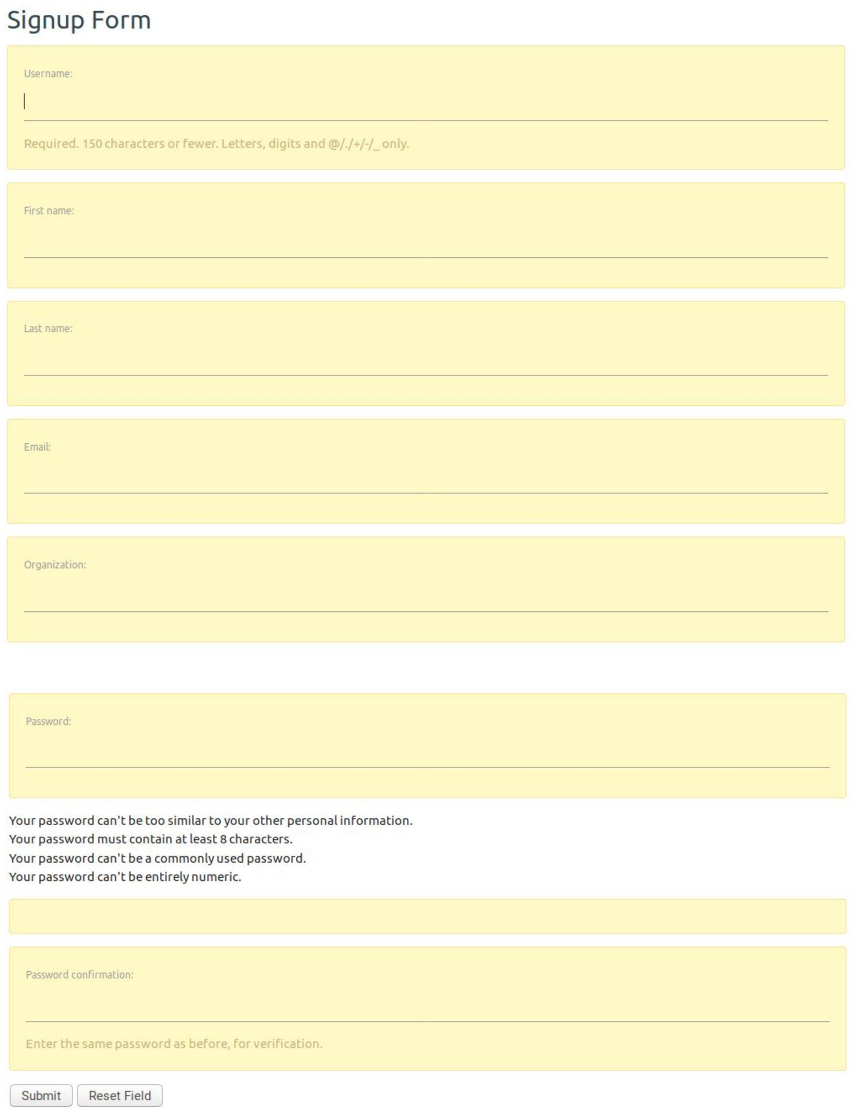
User Manual

1 – Using the Website

Welcome to the Batalog! If this is your first time using the Batalog website, then you must create an account in order to upload images and perform data analysis. Otherwise, skip to the next section.

1.1 – Creating an Account

1. In the home page, hover the mouse to the top right corner of the window and click "Signup".
2. Input your user credentials in the entry fields.



Signup Form

Username:

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name:

Last name:

Email:

Organization:

Password:

Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.

Password confirmation:

Enter the same password as before, for verification.

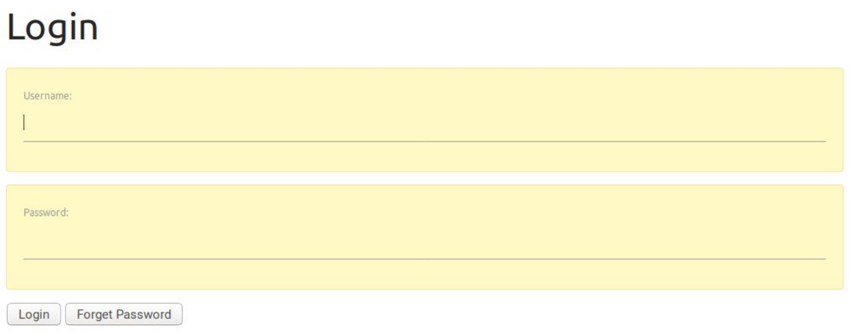
3. Click "Submit" to create your new account. You will be redirected to the home page.

- If you are not satisfied with your input, then click "Reset Field", which clears the fields, and start over

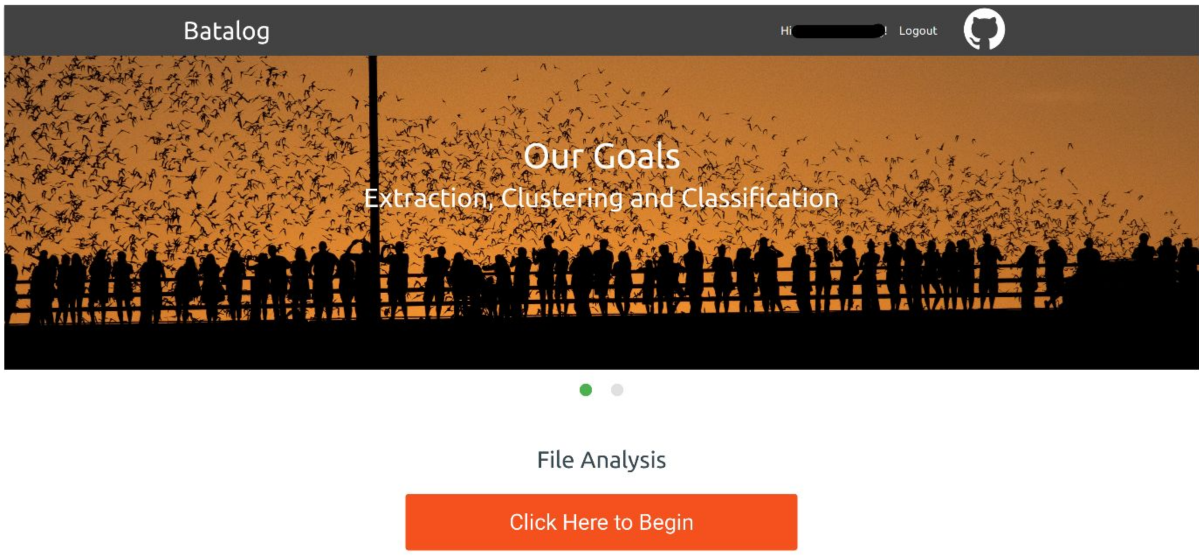
1.2 Log in to Your Account

Before you can perform your data analysis, you must log in to your account; this applies to newly-registered users.

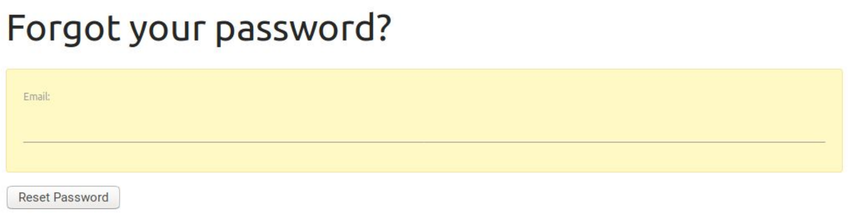
1. In the home page, click "Login". This will take you to the login page.
2. In the login page, enter the username and password that you used to create your account, and then click "Login".



3. If your credentials match, then you are granted access to the website. This is made clear when you see the "Click here To Begin!" button on the homepage, as well as "Hi (username)!" in the top right corner, next to "Logout".



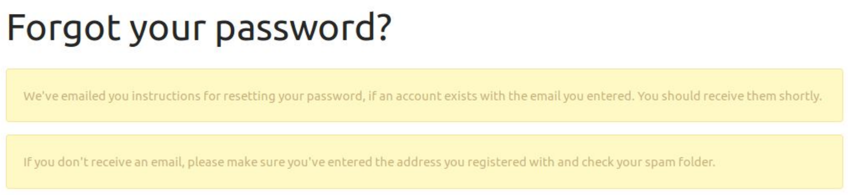
- If you either have forgotten your password and are unable to log in, need to make changes to your current password, or want to use a new password, then click “Forget Password”, which takes you to the following webpage.



Forgot your password?

Email:

- From there, enter the email address you used to make your account and click “Reset Password”. You will be directed to another webpage containing the following information:

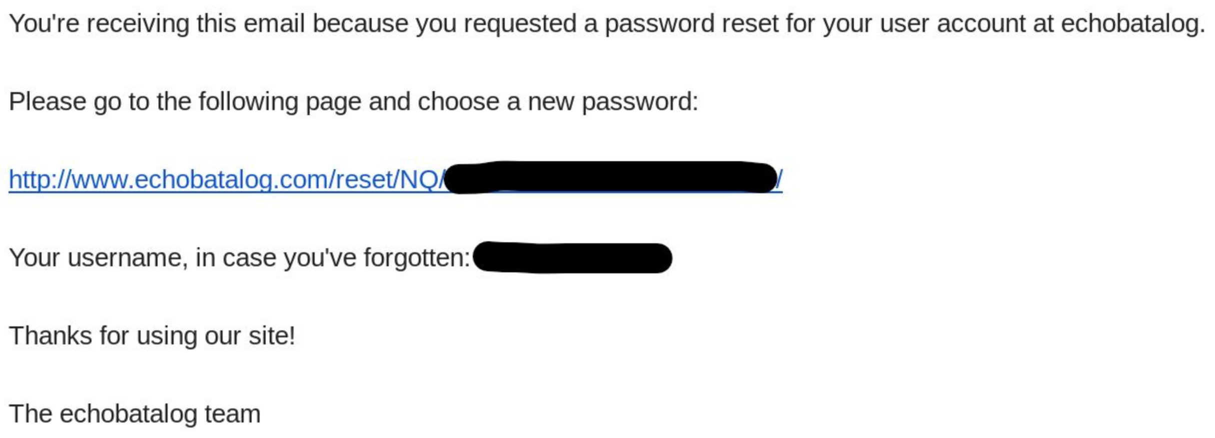


Forgot your password?

We've emailed you instructions for resetting your password, if an account exists with the email you entered. You should receive them shortly.

If you don't receive an email, please make sure you've entered the address you registered with and check your spam folder.

- The website will send an email with the subject line “Password reset on echobatalog” to the email address that you entered in the previous page. The email contains a link to the “Reset Password” page.



You're receiving this email because you requested a password reset for your user account at echobatalog.

Please go to the following page and choose a new password:

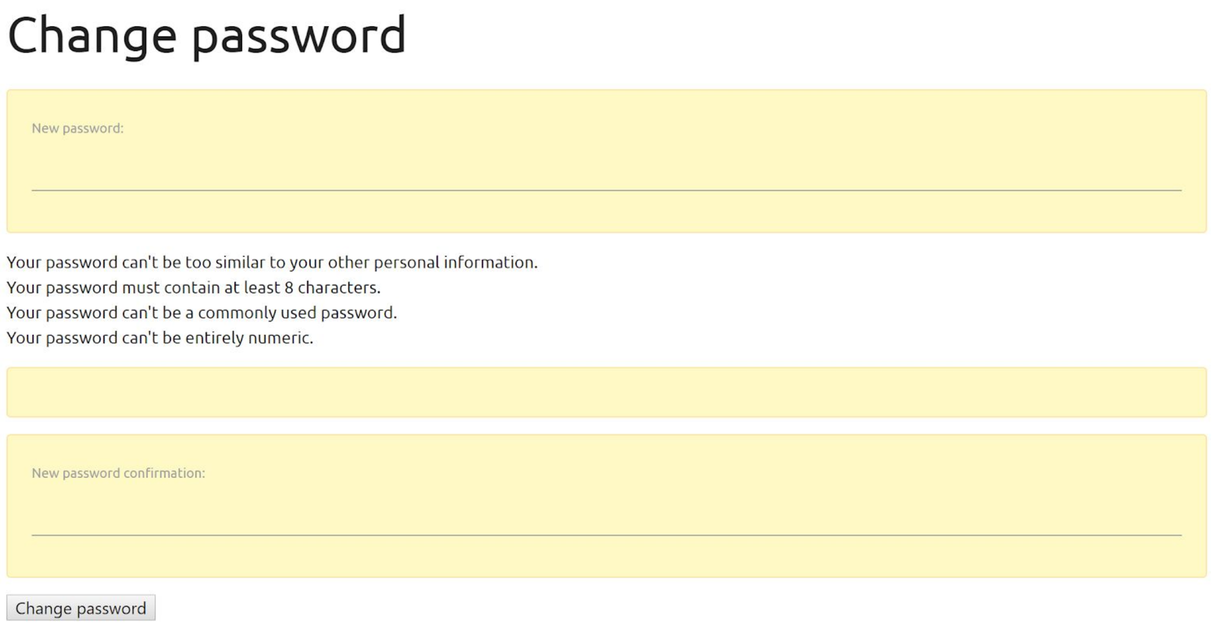
http://www.echobatalog.com/reset/NQ/[redacted]/

Your username, in case you've forgotten: [redacted]

Thanks for using our site!

The echobatalog team

- Clicking the link will take you to the “Change password” page, where you’ll be required to enter and confirm your new password. Once you have satisfied the requirements, click “Change password”.


Change password

New password:

Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.

New password confirmation:

Change password

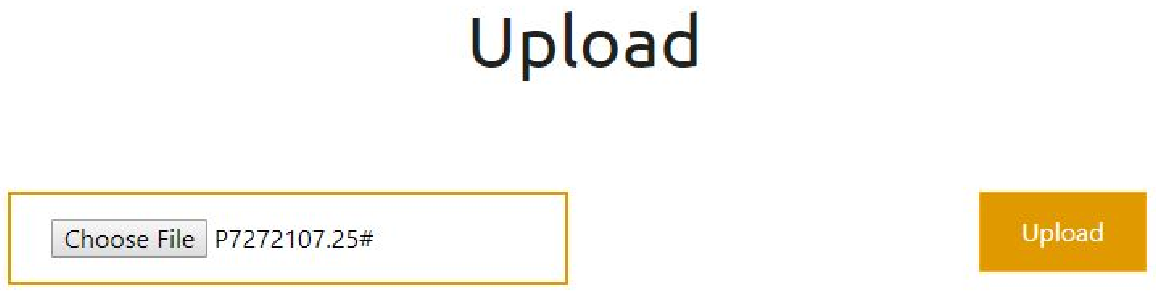
- If you use the link again after having clicked on it already, then the webpage will notify you that you will need to perform another password reset procedure.

2 – Using the Data Analysis Modules

Now that you've successfully logged in to the website, you can start using the main functional modules. To do this, you need some data to work with first before you can analyze.

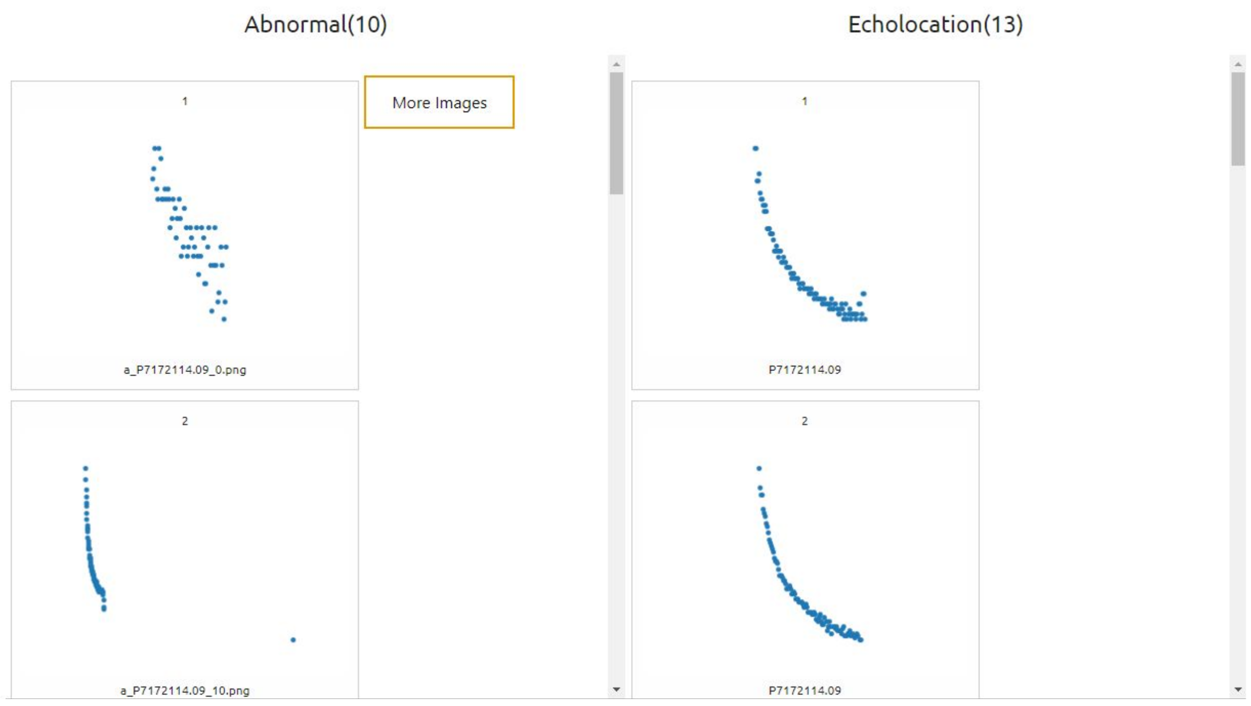
2.1 – Upload Files

The system is designed to take zero-crossing files, which have an extension of two numbers followed by the pound sign (for example, ".45#") as parsed by most computers. To begin, click the button to proceed to the upload page.



1. Upload your file or files. If you want to upload multiple files, then first place them in a compressed folder with either a .zip or .zca extension, before uploading. Click "Choose file" to select a zero-crossing file or a compressed folder file and then click "Upload".

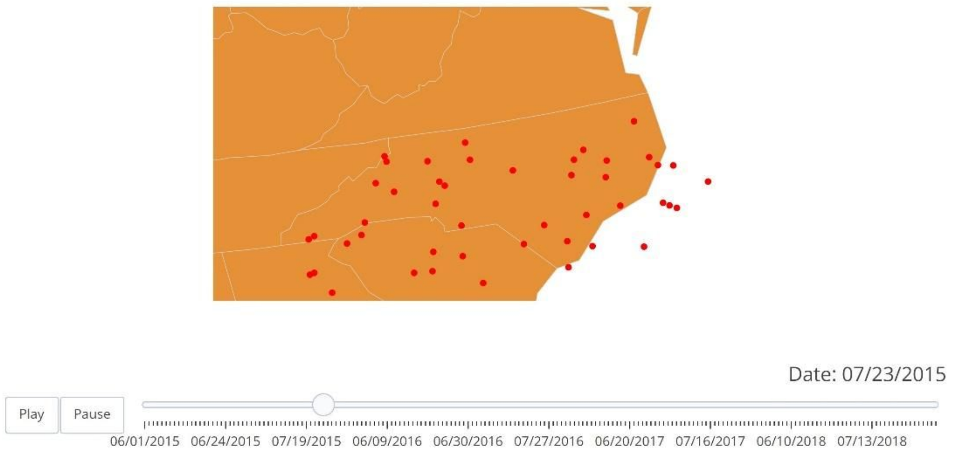
2. You will be treated to a loading GIF as the system processes your uploaded ZC file/s. Once the system is finished processing, you will be directed to a gallery with the images of extracted pulses, in two separate columns "Abnormal" and "Echolocation".



You can scroll through each column to see all the pulses that the system generated. In addition, one column may have more images than the website is able to show, so you can click "More Images" to view the rest of the pulses in that column.

2.2 – Data Analysis

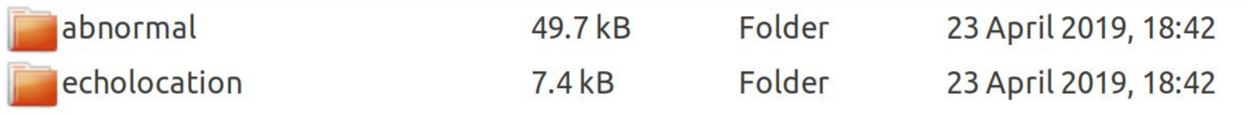
Once you have your pulses, you can view the GPS locations of these bat calls in a map plot. This can be done by clicking "Generate Map".



You also have the option to upload more files and obtain more results. Click "Go back" to return to the previous page and then click "Upload more files", which takes you to the upload page -

refer to Section 2.1 for upload instructions. Once you upload a new file or a compressed folder of multiple files, then you will see the resulting pulses from those files, excluding the previous pulses.

If you want to keep your results and save them to a local storage, then click “Download as zip”. The compressed folder, results.zip, will be downloaded to wherever your “downloads” directory is located on your machine. It contains two folders, “abnormal” and “echolocation”, and each folder contains the same images you see on the “Here Are Your Images!” page.



abnormal	49.7 kB	Folder	23 April 2019, 18:42
echolocation	7.4 kB	Folder	23 April 2019, 18:42

3 – Logging Out

Once you are finished with your data analysis, you can go ahead and log out by clicking “Logout” on the top right corner of the webpage. This will return you to the home page. The results that you received during your login session will be cleared out the moment you log out of the website. When you log back in, you will have to reupload the same files that you used in your previous login session if you want to have those same pulses again. Otherwise, you can perform a new analysis on your next login session.