Trail Camera Image Analyzer

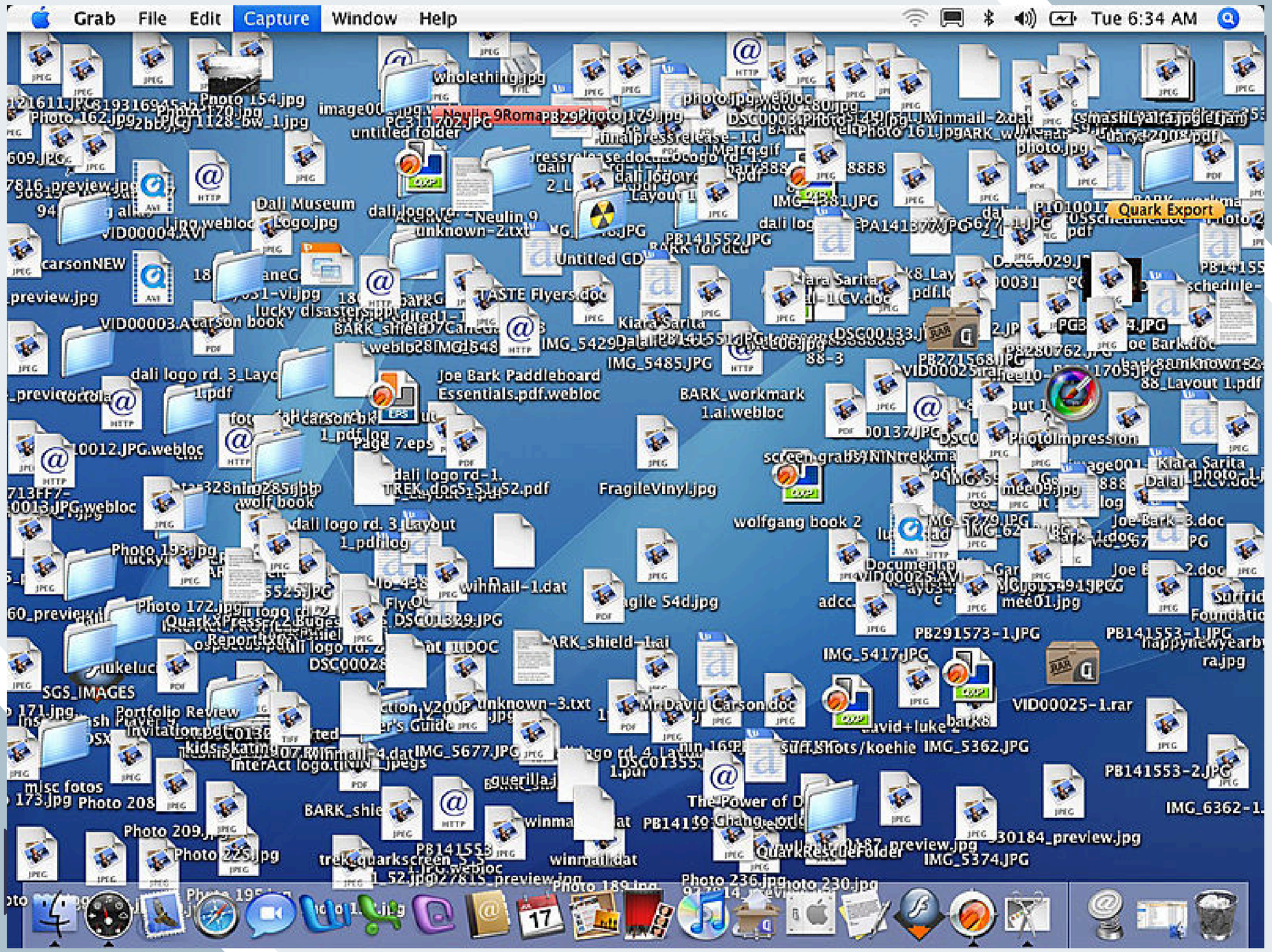
Presented by Gabe Galindo

CYBR 408 Final Project

THE PROBLEM

What’s the Struggle?

- Trail cameras collect tons of images
- Reviewing photos manually is time-consuming
- No easy way to extract useful insights without advanced tools or a bunch of time
- Most users just want fast, clear answers


Grab File Edit Capture Window Help
Tue 6:34 AM
Quark Export

MY SOLUTION

Introducing TCIA

- Web-based tool to analyze wildlife images
- Built using Java Spring Boot and Thymeleaf
- Sorts images and gives useful analytics based on filename data

HOW IT WORKS

- Filenames will be labeled:
  - Original: IMG_7342.JPG
  - New: deer_1_11-07-2022_08:42:54_PM.JPG
- Application parses these and extracts:
  - Species
  - Quantity
  - Date
  - Time (hour and AM/PM)
- Images are grouped and analyzed after labeling is complete

CODE

BEHIND THE SCENES

UPLOAD CONTROLLER

```java
// UploadController class - controller class handles uploading trail camera images, labeling, organizing, and analytics
@Controller  no usages  Gabe Galindo
public class UploadController {
    // store uploaded images in List
    private final List<MultipartFile> uploadedImages = new ArrayList<>();  8 usages

    // handles folder upload, saves images to directory path "code/backend - Trail Camera Image Analyzer/uploads"
    @PostMapping("/upload")  no usages  Gabe Galindo
    public String handleFolderUpload(@RequestParam("images") List<MultipartFile> images, Model model) {
        String redirect = "redirect:/label?currIndex=0";
        if (images == null || images.isEmpty()) {
            System.out.println("No images uploaded!!!");
            redirect = "error";
        } else {
            System.out.println("\nUploading... " + (images.size() - 1) + " images");
            uploadedImages.clear();
            for (MultipartFile img : images) {
                String fileName = img.getOriginalFilename();
                if (fileName != null && fileName.contains(".JPG")) {
                    try {
                        Path uploadPath = Paths.get( first: "uploads/" + fileName);
                        Files.createDirectories(uploadPath.getParent());
                        img.transferTo(uploadPath);
                        uploadedImages.add(img);
                        System.out.println("Saved Image: " + fileName);
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                }
            }
            System.out.println();
        }
        return redirect;
    }
```

UPLOAD CONTROLLER

```java
// processes submitted label data, renames image file name, creates a species folder and moves it into species folder
@PostMapping("/label/next")  no usages  Gabe Galindo
public String handleFormData(@RequestParam("currIndex") int currIndex,
                             @RequestParam("species") String species,
                             @RequestParam("quantity") int quantity) {
    String originalFileName = "uploads/" + uploadedImages.get(currIndex).getOriginalFilename();
    String[] parts = originalFileName.split( regex: "/");
    String folderName = parts[1];
    // extract creation date from image metadata
    MetadataService metadataService = new MetadataService();
    ImageMetadata imageMetadata = metadataService.extractMetadata(originalFileName);
    String date = imageMetadata.formatDate();
    // rename file based on label info and metadata
    String newFileName = species + "_" + quantity + "_" + date + ".JPG";
    // create species folder if it does not exist
    Path specieFolderPath = Paths.get( first: "uploads/"+ folderName + "/" + species);
    if (!Files.exists(specieFolderPath)) {
        try {
            Files.createDirectories(specieFolderPath); // Create the folder
            System.out.println("\nCreated folder for species: " + species + "\n");
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
    // move file with new name to species folder
    String originalPath = Paths.get(originalFileName).toString();
    String newPath = specieFolderPath.resolve(newFileName).toString();
    try {
        Files.move(Path.of(originalPath), Path.of(newPath));
        System.out.println("File path changed from '" + originalPath + "' => '" + newPath);
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
    return "redirect:/label?currIndex=" + (currIndex + 1);
}
```

UPLOAD CONTROLLER

```java
// shows the done page with images and analytics
@GetMapping("/done")  no usages  Gabe Galindo
public String showDonePage(@RequestParam(value = "species", required = false, defaultValue = "all") String species,
                           Model model) {
    List<String> imagePaths = new ArrayList<>();
    List<ImageData> analyticsList = new ArrayList<>();
    if (!uploadedImages.isEmpty()) {
        String[] temp = uploadedImages.getFirst().getOriginalFilename().split( regex: "/");
        String uploadPath = "uploads/" + temp[0] + "/";
        File baseDir = new File(uploadPath);
        // load images based on selected species
        if (species.equals("all")) {
            for (File speciesFolder : Objects.requireNonNull(baseDir.listFiles())) {
                if (speciesFolder.isDirectory()) {
                    for (File img : Objects.requireNonNull(speciesFolder.listFiles())) {
                        String fileName = img.getName();
                        imagePaths.add("/" + temp[0] + "/" + speciesFolder.getName() + "/" + fileName);
                        parseFileNameToData(fileName, analyticsList);
                    }
                }
            }
        } else {
            File speciesFolder = new File(baseDir, species);
            if (speciesFolder.exists() && speciesFolder.isDirectory()) {
                for (File img : Objects.requireNonNull(speciesFolder.listFiles())) {
                    String fileName = img.getName();
                    imagePaths.add("/" + temp[0] + "/" + species + "/" + fileName);
                    parseFileNameToData(fileName, analyticsList);
                }
            }
        }
    }
```

```java
// create analytics maps
Map<String, Integer> speciesCount = new HashMap<>();
Map<String, Integer> speciesQuantity = new HashMap<>();
Map<LocalDate, Integer> dailyCount = new TreeMap<>(); // Keeps dates sorted
Map<String, Integer> hourlyCount = new HashMap<>();
for (ImageData data : analyticsList) {
    speciesCount.merge(data.getSpecies(),  value: 1, Integer::sum);
    speciesQuantity.merge(data.getSpecies(), data.getQuantity(), Integer::sum);
    dailyCount.merge(data.getDate(), data.getQuantity(), Integer::sum);
}
// parse filenames for hourly activity
for (String path : imagePaths) {
    try {
        String filename = new File(path).getName();
        filename = filename.replace( target: ".JPG",  replacement: "").replace( target: ".jpg",  replacement: "");
        String[] parts = filename.split( regex: "_");

        if (parts.length >= 5) {
            int quantity = Integer.parseInt(parts[1]);
            String time = parts[3];
            String amPm = parts[4];
            String hour = time.split( regex: ":")[0];
            String hourLabel = hour + " " + amPm;

            hourlyCount.put(hourLabel, hourlyCount.getOrDefault(hourLabel,  defaultValue: 0) + quantity);
        }
    } catch (Exception e) {
        System.out.println("Error parsing filename: " + path);
        throw new RuntimeException(e);
    }
}
// get top 3 most active hours
List<Map.Entry<String, Integer>> topHours = hourlyCount.entrySet() Set<Entry<...>>
        .stream() Stream<Entry<...>>
        .sorted((e1, e2) -> e2.getValue().compareTo(e1.getValue()))
        .limit( maxSize: 3)
        .collect(Collectors.toList());
// pass data to view
model.addAttribute( attributeName: "selectedSpecies", species);
model.addAttribute( attributeName: "images", imagePaths);
model.addAttribute( attributeName: "speciesCount", speciesCount);
model.addAttribute( attributeName: "speciesQuantity", speciesQuantity);
model.addAttribute( attributeName: "dailyCount", dailyCount);
model.addAttribute( attributeName: "topHours", topHours);
return "done";
```

UPLOAD CONTROLLER

```java
// helper method to parse filename into ImageData object
private void parseFileNameToData(String fileName, List<ImageData> list) {  2 usages  Gabe Galindo
    try {
        String name = fileName.replace( target: ".JPG",  replacement: "").replace( target: ".jpg",  replacement: "");
        String[] parts = name.split( regex: "_");
        String species = parts[0];
        int quantity = Integer.parseInt(parts[1]);
        String dateStr = parts[2];
        String timeStr = parts[3] + "_" + parts[4];
        DateTimeFormatter dateFormat = DateTimeFormatter.ofPattern("MM-dd-yyyy");
        DateTimeFormatter timeFormat = DateTimeFormatter.ofPattern("hh:mm:ss_a");
        LocalDate date = LocalDate.parse(dateStr, dateFormat);
        LocalTime time = LocalTime.parse(timeStr, timeFormat);
        list.add(new ImageData(species, quantity, date, time));
    } catch (Exception e) {
        System.out.println("Error parsing file: " + fileName + " - " + e.getMessage());
    }
}
```

METADATA SERVICE

```java
// MetadataService class - service class to extract metadata from image files
public class MetadataService {  3 usages  Gabe Galindo

    // extracts creation date metadata from the given image file path
    public ImageMetadata extractMetadata(String filePath) {  1 usage  Gabe Galindo
        ImageMetadata imageMetadata;
        try {
            File imageFile = new File(filePath);
            Metadata metadata = ImageMetadataReader.readMetadata(imageFile);
            // get EXIF dir that has the date the image was created
            ExifSubIFDDirectory exifDirectory = metadata.getFirstDirectoryOfType(ExifSubIFDDirectory.class);
            Date creationDate = null;
            if (exifDirectory != null) {
                creationDate = exifDirectory.getDate(ExifSubIFDDirectory.TAG_DATETIME_ORIGINAL);
            }
            // wrap metadata into ImageMetaData object
            imageMetadata = new ImageMetadata(creationDate);
        } catch (ImageProcessingException | IOException e) {
            throw new RuntimeException("Failed to extract metadata from image: " + filePath, e);
        }
        return imageMetadata; // return ImageMetadata object
    }
}
```

POJO

```java
// Image Metadata class
public class ImageMetadata {  6 usages  Gabe Galindo
    private Date creationDate;  2 usages

    // constructors
    public ImageMetadata() {  no usages  Gabe Galindo
        this.setCreationDate(null);
    }

    public ImageMetadata(Date creationDate) {  1 usage  Gabe Galindo
        this.setCreationDate(creationDate);
    }

    // helpers
    public String formatDate() {  1 usage  Gabe Galindo
        String formatDate = null;
        SimpleDateFormat inputFormat = new SimpleDateFormat( pattern: "EEE MMM dd HH:mm:ss zzz yyyy", Locale.ENGLIS
        SimpleDateFormat outputFormat = new SimpleDateFormat( pattern: "MM-dd-yyyy_hh:mm:ss_a");
        outputFormat.setTimeZone(TimeZone.getTimeZone( ID: "UTC"));
        try {
            Date date = inputFormat.parse(this.getCreationDate().toString());
            formatDate = outputFormat.format(date);
        } catch (ParseException e) {
            System.err.println("Failed to parse date: " + this.getCreationDate().toString());
            throw new RuntimeException(e);
        }
        return formatDate;
    }

    // getters
    public Date getCreationDate() { return this.creationDate; }

    // setters
    public void setCreationDate(Date creationDate) { this.creationDate = creationDate; }

    // to string
    public String toString() {  Gabe Galindo
        String temp;
        temp = "Image Metadata Object:\n\tcreation date = {" + this.getCreationDate() + "}\n";
        return temp;
    }
}
```

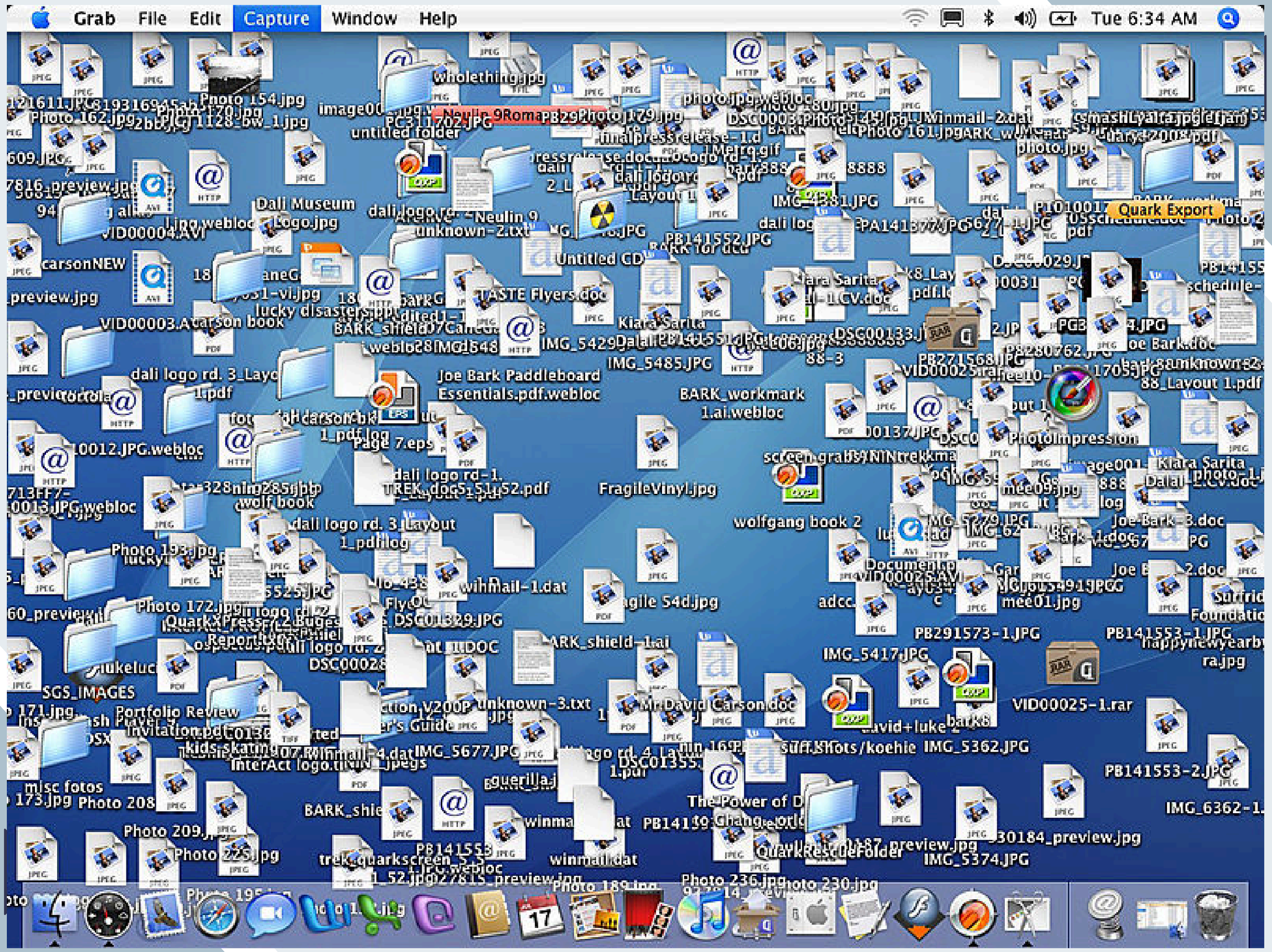
```java
// ImageData class
public class ImageData {  5 usages  Gabe Galindo
    private String species;  2 usages
    private int quantity;  2 usages
    private LocalDate date;  2 usages
    private LocalTime time;  2 usages

    // constructors
    public ImageData() {  no usages  Gabe Galindo
        this.setSpecies("\0");
        this.setQuantity(-1);
        this.setDate(null);
        this.setTime(null);
    }

    public ImageData(String specie, int quantity, LocalDate date, LocalTime time) {  1 usage  Gabe Galindo
        this.setSpecies(specie);
        this.setQuantity(quantity);
        this.setTime(time);
        this.setDate(date);
    }

    // getters
    public String getSpecies() { return this.species; }

    public int getQuantity() { return this.quantity; }

    public LocalDate getDate() { return this.date; }

    public LocalTime getTime() { return this.time; }

    // setters
    public void setSpecies(String species) { this.species = species; }

    public void setQuantity(int quantity) { this.quantity = quantity; }

    public void setDate(LocalDate date) { this.date = date; }

    public void setTime(LocalTime time) { this.time = time; }

    // to string
    public String toString() {  Gabe Galindo
        String temp;
        temp = "Image Data Object:\n\tSpecie: " + this.getSpecies() + "\n" +
                "\n\tQuantity: " + this.getQuantity() + "\n" +
```

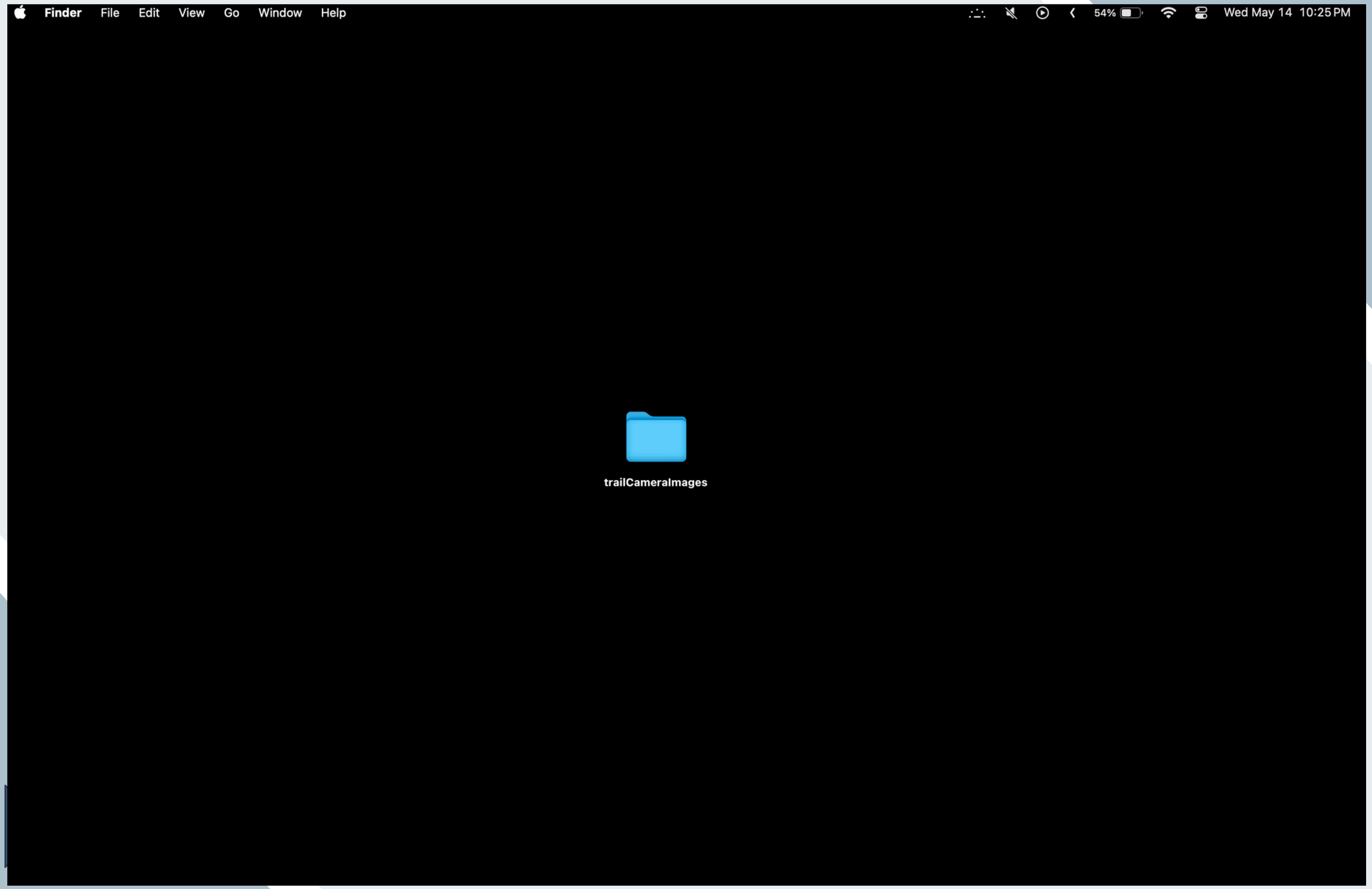
CS CONCEPTS USED

- Iterative Programming
  - Used loops to filter, count, and analyze image data.
- Object-Oriented Programming (OOP)
  - Created ImageData and ImageMetadata classes to structure image info.
- Data Structures
  - Used ArrayList, HashMap, and TreeMap for organizing images and analytics.
- Algorithms
  - Parsed filenames and sorted data to find top activity times.
- Software Engineering
  - Used Spring Boot with MVC and Thymeleaf for clean, maintainable structure.
- OS Concepts
  - Worked with file systems: reading image folders, organizing directories.

CONCLUSION

- Auto-Sorts Images by Species
  - Automatically organizes uploaded images into folders based on species name from the filename.
- Species-Based Filtering
  - Easily filter and browse images by selected species in the web interface.
- Total Image & Animal Counts
  - View quick stats showing how many images and how many animals were captured per species.
- Activity by Date
  - Displays animal activity trends over time, showing how many animals were seen each day.
- Top Hours of Activity
  - Highlights the top 3 most active times of day based on photo timestamps.
- Organized File System
  - Keeps image folders structured and clean by species to reduce manual sorting work.


Grab File Edit Capture Window Help
Tue 6:34 AM
Quark Export


Finder File Edit View Go Window Help
54%
Wed May 14 10:25 PM
trailCameraImages

THANK YOU

FOR YOUR ATTENTION