

Deliverable #2

SE 3A04: Software Design II – Large System Design

Tutorial Number: T03

Group Number: G6

Group Members:

- Cass Braun
- Nehad Shikh Trab
- Savvy Liu
- Tvesha Shah
- Victor Yu

IMPORTANT NOTES

- Please document any non-standard notations that you may have used
 - *Rule of Thumb*: if you feel there is any doubt surrounding the meaning of your notations, document them
- Some diagrams may be difficult to fit into one page
 - Ensure that the text is readable when printed, or when viewed at 100% on a regular laptop-sized screen.
 - If you need to break a diagram onto multiple pages, please adopt a system of doing so and thoroughly explain how it can be reconnected from one page to the next; if you are unsure about this, please ask about it
- Please submit the latest version of Deliverable 1 with Deliverable 2
 - Indicate any changes you made.
- If you do NOT have a Division of Labour sheet, your deliverable will NOT be marked

1 Introduction

1.1 Purpose

This document provides a high-level overview of the Gaim wildlife identification system architecture, including design considerations for the system and its subsystems. The document details architectural choices, subsystem interactions, and class-level responsibilities within the system.

The intended audience for this document includes internal stakeholders such as software developers, system architects, project managers, domain experts in wildlife identification, and potential investors interested in the technological aspects of Gaim. It is recommended that Deliverable 1 be reviewed prior to reading this document for a foundational understanding of the system requirements.

1.2 System Description

The system utilizes a Model-View-Controller (MVC) architecture combined with Repository and Blackboard architectural styles to efficiently process data from various knowledge sources and provide accurate species identification.

The system is designed to facilitate interactions between different subsystems with some subsystems incorporating blackboard architecture, while others incorporate repository architectural style.

1.3 Overview

The remainder of this document is structured as follows:

- **Section 2:** Provides the Analysis Class Diagram for Gaim, illustrating the relationships between key system components.
- **Section 3:** Discusses the overall architectural design of the system, including the rationale behind the chosen architectural styles, the division of the system into subsystems, and their respective functionalities.
- **Section 4:** Presents Class Responsibility Collaboration (CRC) cards, detailing the responsibilities of major classes and their interactions.

2 Analysis Class Diagram

This section should provide an analysis class diagram for your application.

3 Architectural Design

3.1 System Architecture

The Gaim app utilizes an Interaction-Oriented Software Architecture, specifically Model View Controller, to house all the subsystems. The view represents the interface for user interaction and the controller and model represent the three major subsystems and their respective database. The subsystems follow Data-Centric Architecture styles with classification subsystems using Blackboard. The agents are the independent experts (knowledge sources) with whom the user can interact. Synchronously, the blackboard, the active data store component, will take this information in, dividing the solution space, providing a non-deterministic answer and controlling the logic of the application.

System relationships are defined in section 3.2 of this document.

Within our system, the components are defined as the following:

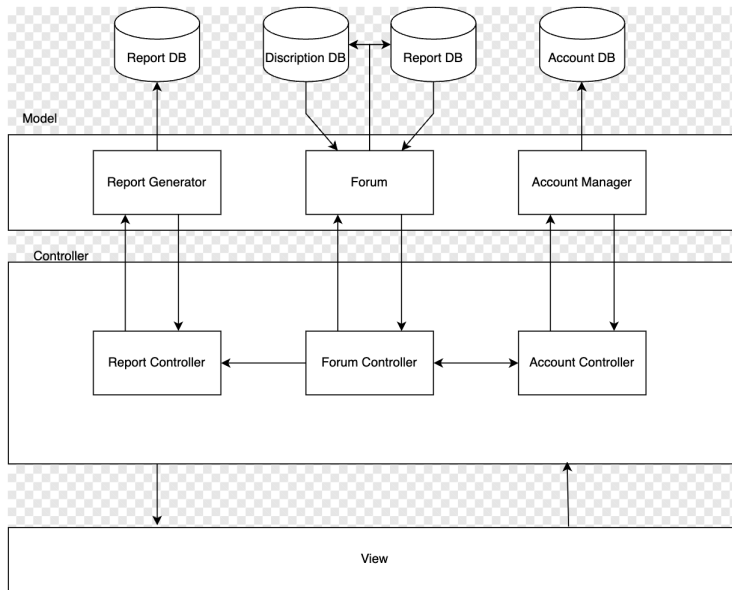
Subsystem	Purpose	Architectural Style
Classification Management	Start a search, submit an image, submit text, fill a survey	Blackboard
Generate a Report	View result, generate report, view score, save	Repository
Account Management	Create an account and log in to an account	Repository

Three databases are present on the model level of this architecture. An account database for account details, a classification database with background information regarding species identification (encompassing both description and image sources), and a report generation database to track user-saved searches.

Our system architecture incorporates the Repository and Blackboard architecture styles. The blackboard style is chosen because it supports multiple independent knowledge sources that can be independently called and allows for a logical data store to present a final output. This is beneficial as our app involves multiple experts/sources of input that the user can utilize, either exclusively or in tandem, to classify a species. Due to this functionality, Blackboard architecture is ideal for classification management as we can narrow down the solution space based on agents and manage them based on the status of the data store. Additionally, in terms of further expansion, it will become very easy to incorporate additional experts/knowledge sources, giving the user more potential use cases for the application.

Another one of the architectures we chose is the Repository Architecture Style as it supports the direct fetching of deterministic outputs from agent requests. This architectural style supports large complex information systems where different components need to access various areas of information. This is the type of access the account management subsystem and report generation system will require, as many users should be able to use the application at the same time and access their specific reports and data. Additionally, this style allows the account management system to easily access stored user information, making it easy for agents to create new accounts and pull credentials for verification. This style also supports data integrity, for backups and restores, which is ideal for account security. Furthermore, for the generated report subsystem, the repository style allows direct access to information from our report generation database and enables the subsystem to display the requested report based on user (agent) input. This architectural style is ideal for expanding new applications, making it easy for us to manage user data, ensure data integrity, and potentially expand the user base.

The overall architecture style chosen for the interaction between these subsystems is the Model View Controller. This architecture style is used because it supports the connection between the user view (presentation) and the data model (backend component) through a controller component that manages input situations and communicates with the appropriate subsystem/database functionality. We can also support frequent data changes, which is essential in the hunting sector which has a changing landscape based on seasonality and new regulations. Further, it is easy to update current functionalities and logic in one component without changing the entire system. This can be beneficial for a newer product because of frequent updates in the user interface or changes to classification logic.



3.2 Subsystems

Provide a list of your subsystems, with a brief description of each. Be sure to document its purpose and relationship to other subsystems.

3.2.1 Classification Management

Brief description. Be sure to document its purpose and relationship to other subsystems.

3.2.2 Generate Report

Brief description. Be sure to document its purpose and relationship to other subsystems.

3.2.3 Account Management

Brief description. Be sure to document its purpose and relationship to other subsystems.

4 Class Responsibility Collaboration (CRC) Cards

Class Name: Success Message (Forum - Boundary)	
Responsibility:	Collaborators:
Knows Forum Displays success message to user upon successful operation Provides feedback on completed processes	Forum

Class Name: Failure Message (Forum - Boundary)	
Responsibility:	Collaborators:
Knows Forum Displays error message when a process fails Provides guidance on next steps for users	Forum

Class Name: Login Error (Account - Boundary)	
Responsibility:	Collaborators:
Knows Account Manager Displays login failure message Provides options to reset password or re-enter credentials	Account Manager

Class Name: Login Success (Account - Boundary)	
Responsibility:	Collaborators:
Knows Account Manager Confirms successful login and redirects user to dashboard Provides session authentication for continued use	Account Manager

Class Name: Forum (Controller)	
Responsibility:	Collaborators:
Knows Free Form Search Knows Survey Search Knows Image Search Knows Success Message Knows Failure Message Knows Free Form Agent Knows Survey Agent Knows Image Agent	Start Free Form Search Start Image Search Start Survey Search Success Message Failure Message Free Form Agent Survey Agent Image Agent

Class Name: Start Image Search (Boundary)	
Responsibility:	Collaborators:
Knows Forum Handles image submission events	Forum

Class Name: Start Freeform Search (Boundary)	
Responsibility:	Collaborators:
Knows Forum Handles freeform test submission events	Forum

Class Name: Start Survey Search (Boundary)	
Responsibility:	Collaborators:
Knows Forum Handles survey submission events	Forum

Class Name: Generate Report (Boundary)	
Responsibility:	Collaborators:
Knows Report Generator Knows Report Database Report Generator Report Database	Forum

Class Name: Report Generator (Controller)	
Responsibility:	Collaborators:
Knows Forum Knows Report Database Report Database View Reports	Forum

Class Name: Report Database (Entity)	
Responsibility:	Collaborators:
Knows Report Generator Knows View Reports View Reports	Report Generator

Class Name: View Reports (Boundary)	
Responsibility:	Collaborators:
Knows Report Database Knows Report Generator Report Generator	Report Database

Class Name: Save Report (Boundary)	
Responsibility:	Collaborators:
Knows Report Generator Report Database	Report Generator

Class Name: Account Manager (Controller)	
Responsibility:	Collaborators:
Knows Login Knows Reset Password Knows Create Account Knows Login Error Knows Login Success Knows Forum Coordinates account interactions from user	Login Reset Password Create Account Login Error Login Success Forum

Class Name: Account Database (Entity)	
Responsibility:	Collaborators:
Knows Account Manager Keeps record of all accounts	Account Manager

Class Name: Login (Boundary)	
Responsibility:	Collaborators:
Knows Account Manager Displays Login Form	Account Manager

Class Name: Reset Password (Boundary)	
Responsibility:	Collaborators:
Knows Account Manager Displays reset password form Relays new password information to Account Manager	Account Manager

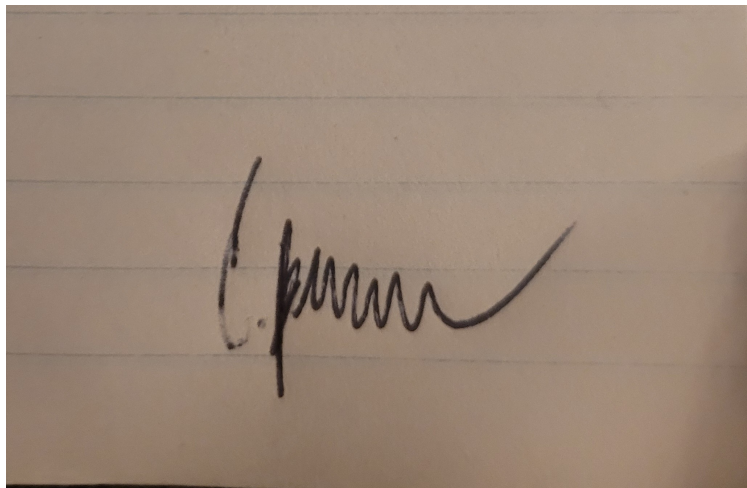
Class Name: Create Account (Boundary)	
Responsibility:	Collaborators:
Knows Account Manager Displays create account screen Runs process to create new account Relays new account information to Account Manager	Account Manager

A Division of Labour

Include a Division of Labour sheet which indicates the contributions of each team member. This sheet must be signed by all team members.

Cass Braun

- Section 3 - Write description for Generate Report Subsystem
- Section 4 - CRC card Account Manager (Controller)
- Section 4 - CRC card Account Database (Entity)
- Section 4 - CRC card Login (Boundary)
- Section 4 - CRC card Reset Password (Boundary)
- Section 4 - CRC card Create Account (Boundary)



Nehad Shikh Trab

- Added purpose
- Added system description
- Added overview

- Added Success Message CRC card
- Added Failure Message CRC card
- Added Login Error CRC Card
- Added Login Success CRC Card



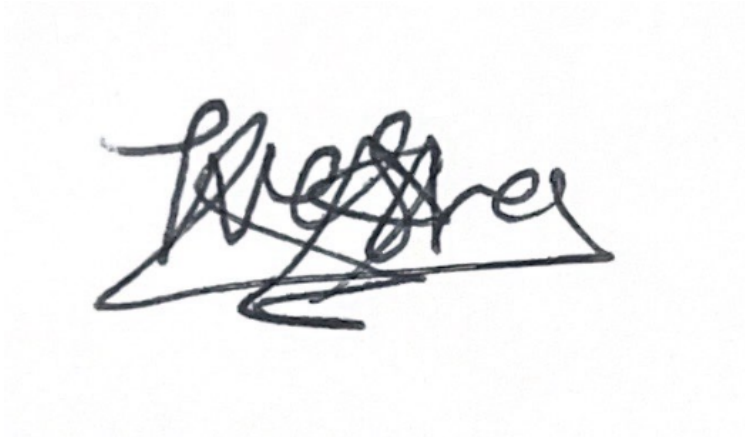
Savvy Liu

- Add contributions



Tvesha Shah

- Section 3 - Identify and explain the overall architecture of your system
- Section 3 - Provide the reasoning and justification of the choice of architecture
- Section 4 - CRC card Forum
- Section 4 - CRC card Start Image Search (Boundary)
- Section 4 - CRC card Start Freeform Search (Boundary)
- Section 4 - CRC card Start Survey Search (Boundary)
- Managed github set up and assisted with formatting



Victor Yu

- Add contributions

