

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

This section should provide an brief overview of the entire document.

1.1 Purpose

State the purpose and intended audience for the document.

1.2 System Description

Give a brief description of the system. This could be a paragraph or two to give some context to this document.

1.3 Overview

Describe what the rest of the document contains and explain how the document is organised (e.g. "In Section 2 we discuss...in Section 3...").

2 Analysis Class Diagram

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

3 Architectural Design

This section should provide an overview of the overall architectural design of your application. Your overall architecture should show the division of the system into subsystems with high cohesion and low coupling.

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.

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

System relationships are defined in section 3.2 of this document.

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.

4 Class Responsibility Collaboration (CRC) Cards

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

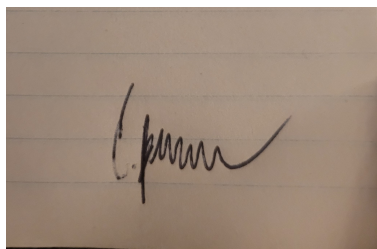
Class Name: Start Survey Search (Boundary)	
Responsibility:	Collaborators:
Knows Forum Handles survey submission events	Forum

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

- Add contributions



Nehad Shikh Trab

- Add contributions



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

