

# 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 a Data-Centric Architecture, specifically the blackboard architecture style for majority of the application. The agents are the independent experts (knowledge sources) that our user can interact. Synchronously, the blackboard, the active data store component, will take this information in, dividing the solution space and 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	Blackboard
Account Management	Create an account and log in to an account	Repository

System relationships are defined in section 3.2 of this document.

In addition, 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 classification, and a search/report history 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 data store logic. 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. Further, for the generated report subsystem, we can extend that functionality by creating agents for “report types” (rather than just one general agent) and have our data store update and logically control the information those report agents display. Moreover, the data and solutions we are presenting to our clients are non-deterministic, making blackboard architecture the ideal choice to ensure we are not relying solely on agent logic and have our data store making active decisions based on a set of facts.

Another one of the architectures we chose is the Repository Architecture Style, because it supports direct fetching of deterministic outputs from agents. This architecture style allows for large complex information systems where different components need to access different aspects of information. This is the type of situation the account management subsystem will require, as many users should be able to use the application at the same time. Furthermore, this style allows the account management system to easily access stored user information, making it easy for agents to create new accounts and full credentials for verification. This style supports data integrity, for backups and restores, which is ideal for account security. Overall, this architect style is ideal for an expanding new application, making it easy for us to manage user data, ensure data integrity and potentially expand our new user base.

### 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

This section should contain all of your CRC cards.

- Provide a CRC Card for each identified class
- Please use the format outlined in tutorial, i.e.,

<b>Class Name:</b>	
<b>Responsibility:</b>	<b>Collaborators:</b>

<b>Class Name: (Specify Class Name)</b>	
<b>Responsibility:</b>	<b>Collaborator:</b>
Knows Start an image search	Start an image search
Knows Start a textual search	Start a textual search
Knows Start a survey search	Start a survey search
Knows Photo Scanner	Photo Scanner
Knows Error/timeout Message	Error/timeout Message
Knows Current Status	Current Status
Knows Search History	Search History
Knows Success message	Success message
Knows Search details (report)	Search details (report)
Knows Account Management	Account Management

Class Name: Error/Timeout Message (Boundary)	
Responsibility:	Collaborators:
Handles time-out events Handles Image Submission error events Handles Text submission error events Handles Survey submission error events Handles report generation error events	Classification Management Data Store

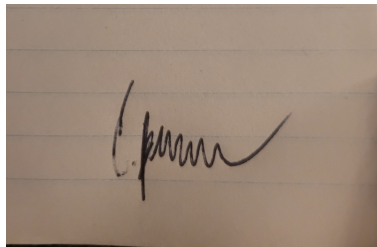
Class Name: Current Status (Boundary)	
Responsibility:	Collaborators:
Knows Classification Management Data Store Knows User Search Status	Classification Management Data Store

## 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

A stylized handwritten signature in black ink, consisting of a large loop followed by a horizontal line.

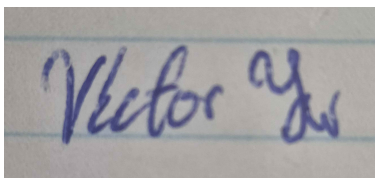
**Tvesha Shah**

- 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 Blackboard DataStore
- Section 4 - CRC card Error/Timeout Message (Boundary)
- Section 4 - CRC card Current Status (Boundary)
- Managed github set up and assisted with formatting

A complex, stylized handwritten signature in black ink, featuring multiple overlapping loops and a long horizontal stroke at the bottom.

**Victor Yu**

- Add contributions

A handwritten signature in blue ink on lined paper, reading "Victor Yu".