

Deliverable #2

SE 3A04: Software Design II – Large System Design

March 9, 2025

Tutorial Number: T03

Group Number: G4

Group Members:

- Gregory, Noel Chungath
- Maurer, Daniel Hans
- Elbasiouni, Karim
- Fawaz, Nour
- Kashif, Haniya

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 GameOracle system architecture, including key design considerations and the various subsystem architectures. It is intended for internal stakeholders, which include project managers, domain experts, developers, and investors. Prior documentation from Deliverable 1 should be reviewed, and technical knowledge will aid in understanding this document as a whole.

1.2 System Description

This system follows multiple architectures for the different sections of the system. This approach will provide flexibility in the data handling as its dynamical interacts with other agents. The project uses two subsystems: the repository and the blackboard architecture, both allowing for data to be driven as the main form. The blackboard architecture was primarily used for identification, while other components were driven by the repository architecture.

1.3 Overview

Overview Section 2 presents an Analysis Class Diagram for GameOracle, illustrating the essential components of the system. In Section 3, we further explain the reasoning for our choice of architectural design and also discuss the alternatives and reasons behind our design decisions. In the final Section 4, we create various Class Responsibility Collaboration cards that represent the individual classes, their responsibilities, and interactions between them.

2 Analysis Class Diagram

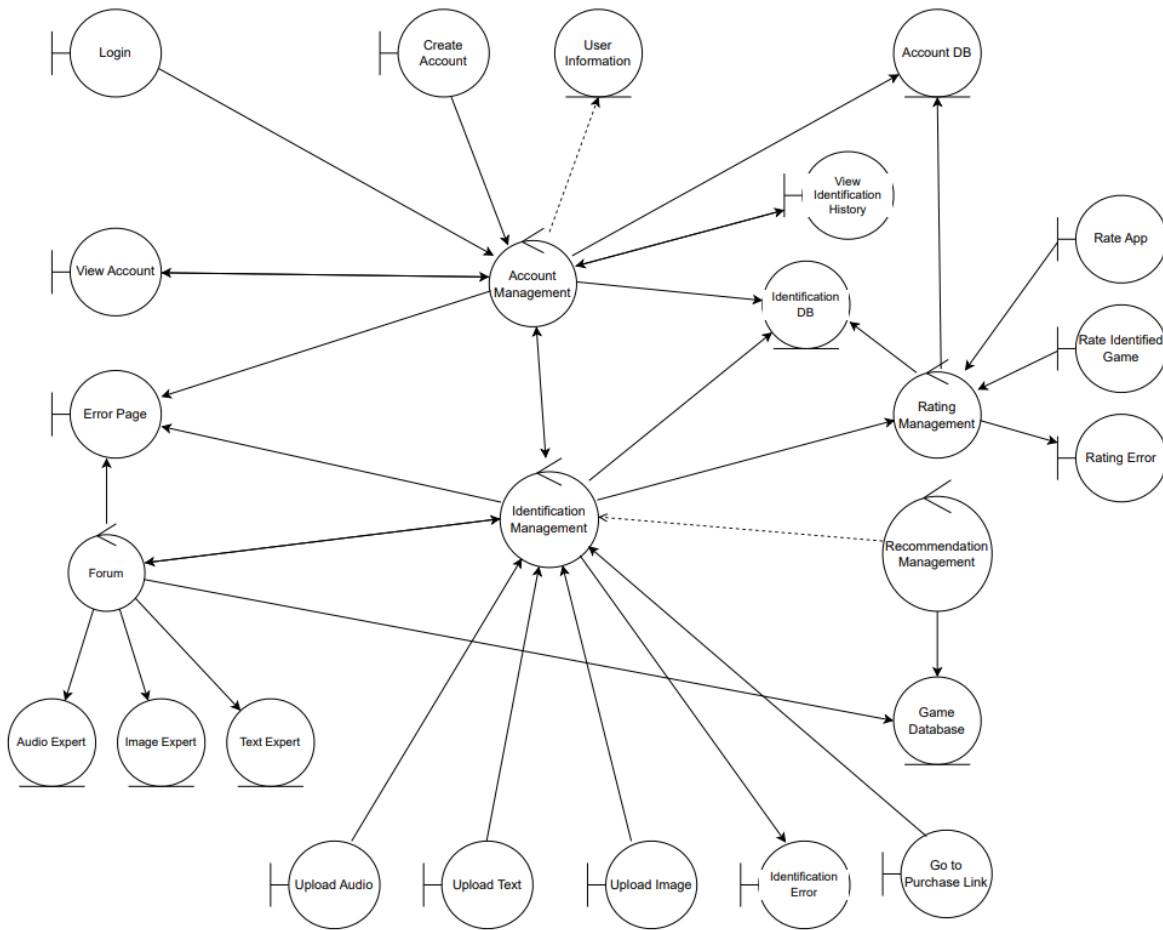


Figure 1: Analysis Class Diagram

3 Architectural Design

3.1 System Architecture

GameOracle utilizes a mix of the two primary Data-Centered Software Architecture - namely, the Blackboard Architectural Style and the Repository Architectural Style. Splitting the functionalities into cohesive sections, their styles are defined below:

Subsystem	Purpose	Architectural Style
Account Management	User registration, login, profile management	Repository
Identification Management	Processes user uploads to identify games	Blackboard
Rating Management	Rate identified games and app abilities	Repository
Recommendation Management	Generate game recommendations based on user history	Repository

The Blackboard Architectural Style is used due to the type of problem being solved by this system. Focusing purely on the main functionality of game identification; due to the unreliability of the input data, a deterministic answer is not possible. This is within the applicable domain of the Blackboard style. Additionally, the Blackboard style's key characteristic is to consist of multiple active, independent knowledge sources interacting via a central data store, directly aligning with GameOracle's three experts and central forum. Enforcing that these experts remain independent and linked only to the forum ensures the principle of high cohesion and low coupling is followed, and allows for alterations to be made to each expert in development with ease. Furthermore, the need to add more experts was mentioned within the system's initial requirements, something the Blackboard style addresses. This scalability is then an aspect of this part of GameOracle.

The other architecture utilized is the Repository Architectural Style. Each of the other functionalities (account management, rating management) make use of a relational database, something that is within the applicable domain of repositories. Considering that some of these databases may be shared between multiple controller classes and multiple users, this also fits the description of many software clients accessing a central data store - a large and complex information system typical of repositories. Supporting scalability for further clients is also critical as GameOracle's userbase expands. One issue that may need to be dealt with is the repository's main vulnerability - the central data store. However, if standard backups are made frequently, accessibility risks can be greatly reduced.

After analyzing our project, we understood that the Pipe and Filter Architecture style will not be the best suited for it. This is mainly due to the fact that it doesn't support the level of interactivity required for our system. Our design requires dynamic interactions, such as content uploading, game identification, and user account creation, which do not align well with the strict sequential processing of the Pipe and Filter architecture. The Pipe and Filter design, at its core, is meant for processing data one step at a time, where data is passed through a stream from one filter to another. In our design, there are no areas where this would enhance our system. Instead, it would restrict the flow of data and not allow users to experience seamless concurrent interactivity.

In this project, we also considered the Batch Sequential Architecture, which was ultimately deemed unsuitable due to its reliance on batch data processing. The core feature of this architecture is to allow data to be separated and sent into separate components for processing. This approach is beneficial for handling large amounts of data, however, in our case, the data stored per user is very minimal, and as such, there is no inherent reason to separate the data and process it in batches.

The Process-Control Architecture was eliminated from our system as it is primarily intended for real-time monitoring and embedded systems. This architecture is designed for applications that require continuous monitoring and control of system variables. However, our project does not involve controlling physical processes, and as such, it is not a good fit for our system.

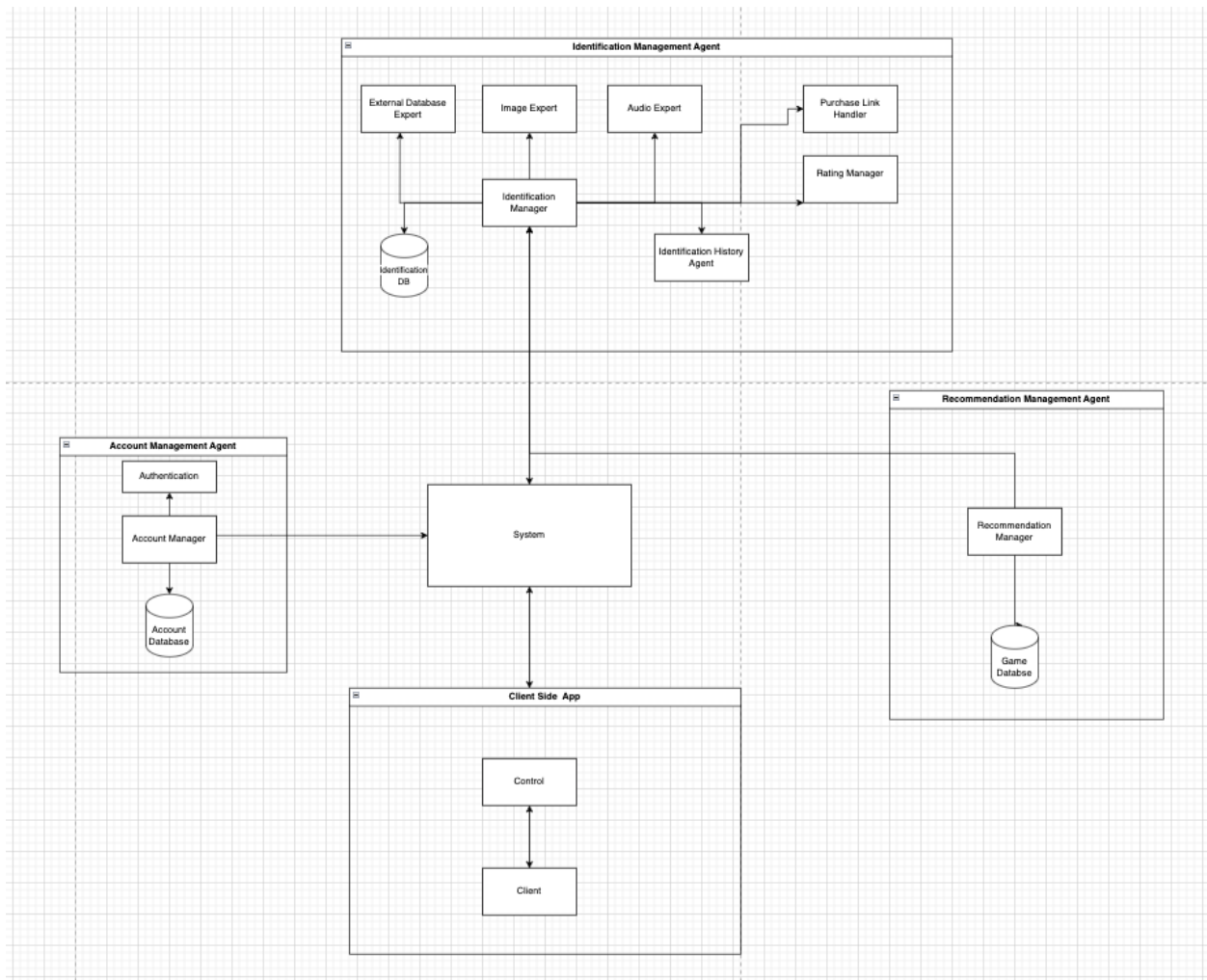


Figure 2: Figure 2. System Architecture

The Presentation-Abstraction-Control (PAC) Architecture was also analyzed as a potential candidate for our system due to its emphasis on modular separation. However, we decided against using it because its agent-based structure makes it difficult to track real-time user interactions. Additionally, the strict separation between presentation and abstraction further complicates communication between system components, making it less suitable for our needs.

The Model-View-Controller (MVC) Architecture was also considered for our system, as it is a widely used design pattern in many applications. However, after further evaluation, we concluded that it does not effectively support our complex data flow. The strict separation of Model, View, and Controller limits the flexibility of data interactions and does not provide the same essential features as the Blackboard Architecture, which better suits our requirements.

Overall, by eliminating all these architectures, we were able to refine our design by adopting the Blackboard and Repository Architecture, which allows for a more effective and scalable final solution.

3.2 Subsystems

The **GameOracle** system is composed of four subsystems that work together to provide game identification, user management, ratings, and recommendations. Each subsystem is crucial for the application's functionality, ensuring a great user experience and smooth data flow.

The **Account Management Subsystem** handles user registration, login, logout, and profile management. It allows users to create accounts, view account information, and manage personal data. This subsystem interacts with the Identification DB, which is related to the Identification Management Subsystem, to access and categorize each user's identification history. It also stores user information in the Account DB.

The **Identification Management Subsystem**, including the forum, is responsible for processing user inputs/uploads — such as audio, text, or images — to identify games. It calls the three experts through the forum, who report their answers, and identifies the game based on the experts' responses. It manages conflicting answers, stores all recorded identifications in the Identification DB, and records whether the user correctly identified the game. Additionally, it collaborates with the Recommendation Management Subsystem to suggest similar games.

The **Rating Management Subsystem** allows users to rate identified games and the app itself. It assigns ratings to games or records rating errors, storing game ratings in the Identification DB and app ratings in the Account DB, linked to users who choose to rate the app. Users must use the Identification feature first to rate games, connecting this subsystem to the Identification Management Subsystem. Furthermore, game ratings are shared with the Recommendation Management Subsystem to support future scalability.

The **Recommendation Management Subsystem** generates game recommendations based on the game identified. It communicates with the Game DB to access game data and provides purchase links to both recommended games and the identified game. This subsystem relates to the Identification Management Subsystem by using identification data to make suggestions. Designed for future scalability, it can integrate personalized recommendations using machine learning or user ratings.

In conclusion, the four subsystems — Account Management, Identification Management, Rating Management, and Recommendation Management — work in unison to deliver an efficient and user-centric GameOracle experience. The modular design supports both current functionality and future growth.

4 Class Responsibility Collaboration (CRC) Cards

Class Name: Account Management (Controller)	
Responsibility:	Collaborators:
-Handles login and authentication -Handles Create Account Boundary -Knows View Account Boundary -Knows Identification Management Controller -Knows View Identification History Boundary -Knows Identification Database Entity -Knows User Information Entity -Knows Error Page Boundary	-Login, User Information -Create Account Boundary, Account Database Boundary -View Account Boundary -Identification Management Controller -View Identification History Boundary -Identification Database Entity -User Information Entity -Error Page Boundary

Class Name: Create Account (Boundary)	
Responsibility:	Collaborators:
-Knows Account Management Controller -Knows Username -Knows Password -Handles form validation and click 'Create Account' button	-Account Management Controller

Class Name: Login (Boundary)	
Responsibility:	Collaborators:
-Knows Account Management Controller -Handles login form validation and click on 'Login' button	-Account Management Controller

Class Name: View Account (Boundary)	
Responsibility:	Collaborators:
-Knows Account Management Controller -Handles click on 'View Account Profile' button and displays User Information	-Account Management Controller

Class Name: View Identification History (Boundary)	
Responsibility:	Collaborators:
-Knows Account Management Controller -Handles click on 'View Identification History' button and displays Identification History	-Account Management Controller

Class Name: Error Page (Boundary)	
Responsibility:	Collaborators:
-Displays generic error messages related to Account Management Controller -Displays generic error messages related to Forum Controller -Displays generic error messages related to Identification Management Controller -Offers re-entry or navigation following the error	-Account Management Controller -Forum Controller -Identification Management Controller

Class Name: Forum (Controller)	
Responsibility:	Collaborators:
-Knows Error Page Boundary -Knows Identification Management Controller -Receives the partial solutions from the three Experts and aggregates them -Knows and Uses Game Database for aggregation	-Error Page Boundary -Identification Management Controller -Audio Expert Entity, Text Expert Entity, Image Expert Entity -Game Database Entity

Class Name: Image Expert (Entity)	
Responsibility:	Collaborators:
-Use image input to figure out a partial solution	-Forum

Class Name: Audio Expert (Entity)	
Responsibility:	Collaborators:
-Use audio input to figure out a partial solution	-Forum

Class Name: Text Expert (Entity)	
Responsibility:	Collaborators:
-Use text input to figure out a partial solution	-Forum

Class Name: Identification Management (Controller)	
Responsibility:	Collaborators:
-Knows Error Page Boundary -Knows Account Management Controller -Knows Forum Controller -Knows Identification Database Entity -Receives each of the three inputs from the user and parses them accordingly -Knows Identification Error Boundary -Handles re-routing to game purchase link -Knows Rating Management Controller -Provides Recommendation Management Controller with the necessary data to make recommendations	-Error Page Boundary -Account Management Controller -Forum Controller -Identification Database Entity -Upload Audio Boundary, Upload Image Boundary, Upload Text Boundary -Identification Error Boundary -Go To Purchase Link Boundary -Rating Management Controller -Recommendation Management Controller

Class Name: Upload Audio (Boundary)	
Responsibility:	Collaborators:
-Handles click on 'Upload Audio' button and provides Identification Management Controller with user's audio data	-Identification Management Controller

Class Name: Upload Text (Boundary)	
Responsibility:	Collaborators:
-Handles click on 'Upload Text' button and provides Identification Management Controller with user's text data	-Identification Management Controller

Class Name: Upload Image (Boundary)	
Responsibility:	Collaborators:
-Handles click on 'Upload Image' button and provides Identification Management Controller with user's image data	-Identification Management Controller

Class Name: Identification Error(Boundary)	
Responsibility:	Collaborators:
-Notifies user about identification error following unsuccessful identification -Offers re-entry/navigation following error	-Identification Management Controller

Class Name: Go To Purchase Link(Boundary)	
Responsibility:	Collaborators:
-Handles click on 'Go To Purchase' button to provide user with a link to purchase game	-Identification Management Controller

Class Name: User Information (Entity)	
Responsibility	Collaborators
-Knows Username -Knows Password -Knows Date-of-Birth -Knows Account Management Controller	-Account Management Controller

Class Name: Account Database (Entity)	
Responsibility	Collaborators
-Knows Rating Management -Knows Account Management Controller	-Account Management -Rating Management

Class Name: Identification Database (Entity)	
Responsibility:	Collaborators:
-Knows Identification Management Controller -Knows Account Management Controller -Knows Rating Management Controller	-Identification Management Controller -Account Management Controller -Rating Management Controller

Class Name: Game Database (Entity)	
Responsibility:	Collaborators:
-Knows Forum Controller -Knows Recommendation Management Controller	-Forum Controller -Recommendation Management Controller

Class Name: Rating Management (Controller)	
Responsibility:	Collaborators:
-Knows Rate App -Knows Rate Identified Game -Knows Rating Error -Knows Account DB -Knows Identification DB -Knows Identification Management	-Rate App -Rate Identified Game -Rating Error -Account DB -Identification DB -Identification Management

Class Name: Rate App (Boundary)	
Responsibility:	Collaborators:
-Knows Rating Management -Handles rating input one to five stars	-Rating Management Controller

Class Name: Rate Identified Game (Boundary)	
Responsibility:	Collaborators:
-Knows Rating Management -Handles rating input one to five stars	-Rating Management Controller

Class Name: Rating Error (Boundary)	
Responsibility:	Collaborators:
-Knows Rating Management -Handles time out event -Handles non-successful rating submission event	-Rating Management Controller

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.

Gregory, Noel Chungath

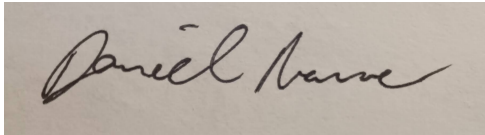
- Co-authored Section 3.1: System Architecture with Daniel, specifically focusing on refining and documenting the eliminated design alternatives

- Designed the overall system architecture, ensuring a structured and scalable approach to meet project requirements
- Authored the introductory section of the project, providing a clear overview of objectives, scope and key components



Maurer, Daniel Hans

- Wrote Explanation half of Section 3.1 System Architecture, co-authored with Noel
- Helped with Analysis Class Diagram as a group



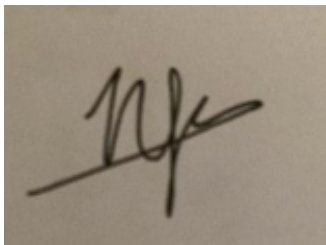
Elbasiouni, Karim

- Created CRC Cards for Account Management, Create Account, Login, View Account, View Identification History, Error Page Forum, Image Expert, Audio Expert, Text Expert, Identification Management, Upload Audio, Upload, Text, Upload Image, Identification Error, Go To Purchase Link
- Helped with Analysis Class Diagram as a group



Fawaz, Nour

- Fixed Deliverable 1 to account for feedback
- Formatted document into different subfiles
- Completed User Information, Account Database, Identification Database, and Game Database CRC Cards
- Helped with Analysis Class Diagram as a group



Kashif, Haniya

- Wrote Section 3.2 Subsystems
- Helped with Analysis Class Diagram as a group
- Made CRC cards for Rating management, Rate App, Rate Identified Game, and Rating Error

~~Hariga~~