Deliverable #2

SE 3A04: Software Design III – Large System Design

Tutorial Number: T02
Group Number: G02
Group Members:

- Alvin Qian (Team Leader)
- Ryan Kumar
- Lucas DeBoer
- Varun Pathak
- Maya Azar
- Moustafa Moustafa

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

Note that SME stands for SoleMate Management Engine and SSD stands for SoleMate Shoe Database. These were terms initially introduced in the previous report.

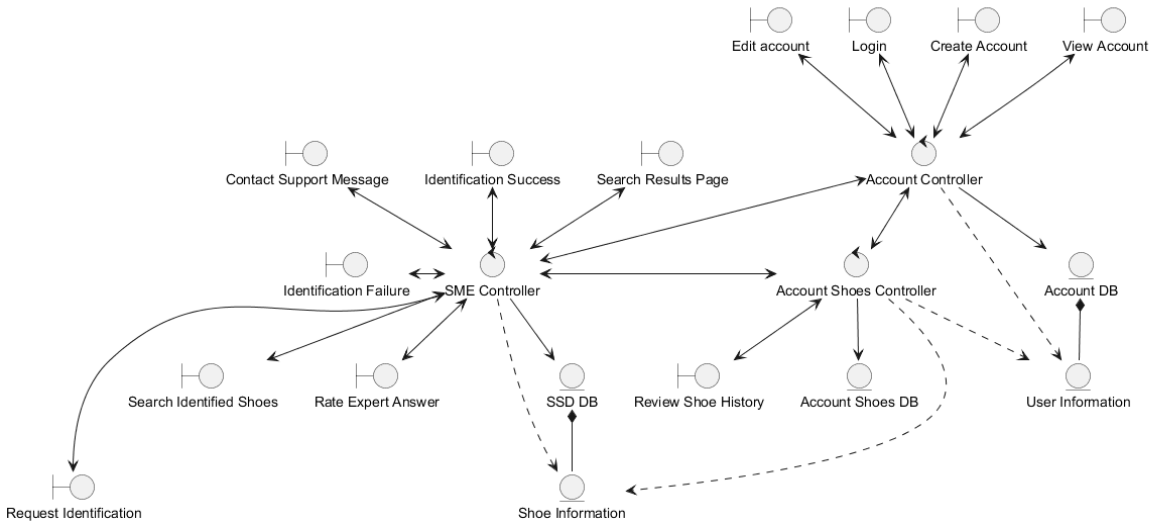

Figure 1: Analysis Class Diagram

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

SoleMate utilizes a Blackboard Architecture integrated with a Repository Style Design to enable efficient shoe identification and data management. The Blackboard Architecture allows multiple expert modules to

contribute their analyses iteratively, refining identification results until a final decision is reached. When a user submits a shoe image or textual description, the SoleMate Management Engine (SME Controller) acts as the central hub, posting the input to a shared data space where different expert modules—such as machine learning-based classification, image recognition, and database matching—collaborate to analyze the data. This approach supports a flexible and scalable identification process, allowing for the addition of new expert modules over time and improving recognition accuracy through iterative refinement.

The Repository Architecture complements the Blackboard system by ensuring structured and persistent data storage. The SSD Database serves as the repository for storing identified shoes, historical searches, and expert ratings, while the Account Shoes Database and Account Database manage user-related data, such as past searches, stored shoe collections, and authentication details. This repository-style approach allows different subsystems to access data concurrently, ensuring efficient retrieval and updates without conflicts. By combining real-time collaborative processing with structured data management, SoleMate ensures both adaptability and long-term reliability in handling sneaker identification and user interaction.

The Blackboard Architecture was chosen because it provides a collaborative problem-solving environment, where different expert modules can iteratively refine identification accuracy. This design is well-suited for SoleMate, as shoe recognition requires integrating various techniques, such as AI models, database lookups, and image recognition. The Repository Architecture was selected to efficiently manage large datasets, ensuring seamless access to stored information while allowing the system to scale as more users and shoe data are added. Together, these architectures provide a flexible, scalable, and data-driven solution for SoleMate's sneaker identification platform.

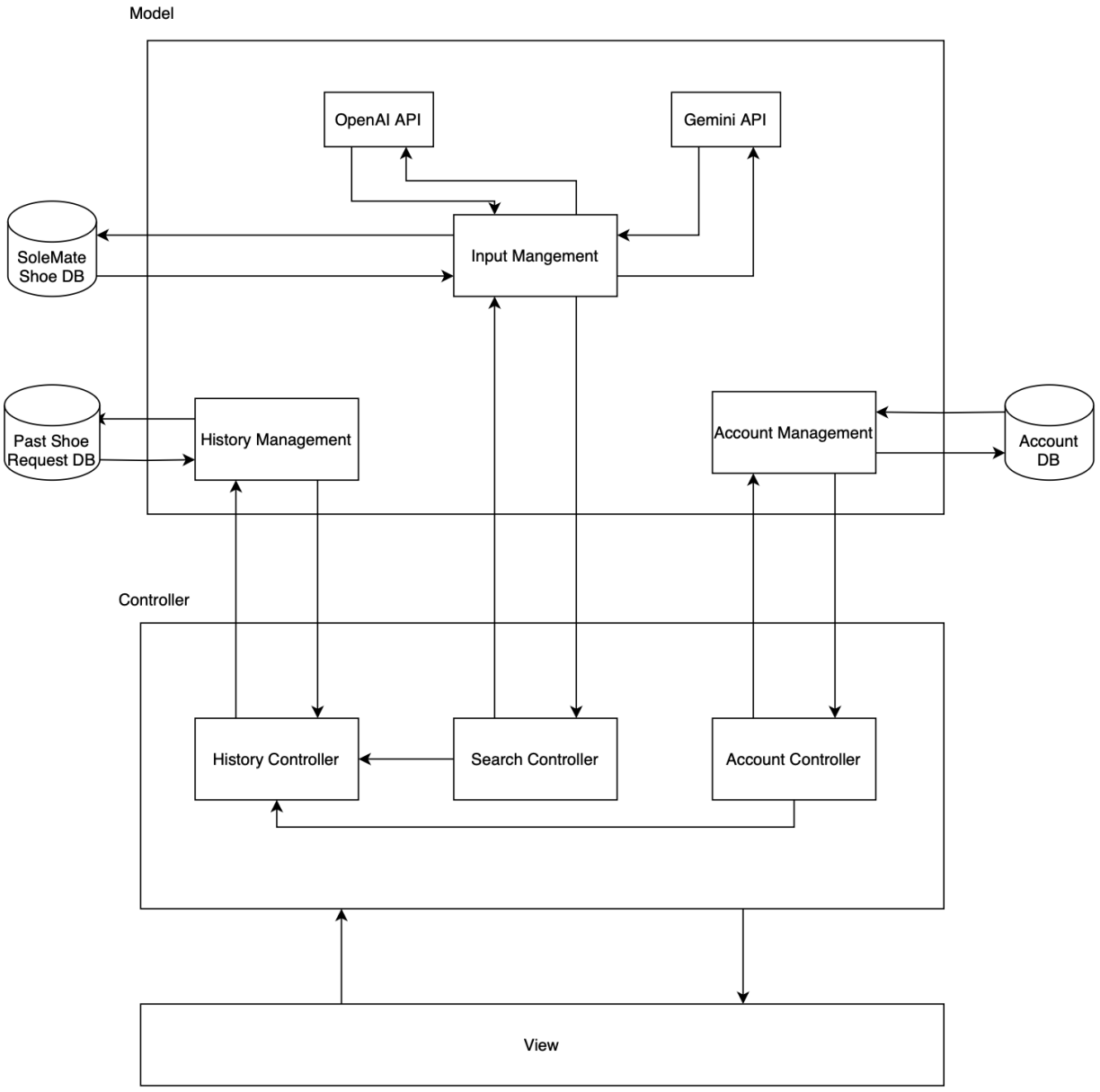

Figure 2: System Architecture Diagram

Several alternative architectural styles were considered but ultimately discarded. The Model-View-Controller (MVC) Architecture was initially evaluated, but it was not ideal for SoleMate, as the system prioritizes data-driven processing and expert collaboration over UI management. The Pipe and Filter Architecture was also considered but was eliminated because it enforces a sequential flow of data, which does not align with SoleMate's iterative expert-based approach to identification. Lastly, the Batch Sequential Architecture was rejected due to its reliance on processing data in predefined batches, as SoleMate requires real-time responses for shoe identification requests. By combining Blackboard and Repository Architecture, SoleMate successfully integrates adaptive intelligence with structured data storage, creating an optimal solution for sneaker identification and management.

3.2 Subsystems

SoleMate utilizes three main subsystems, incorporating both a Blackboard and Repository-like architecture. The SoleMate system integrates a Blackboard Architecture with a repository-style design to streamline shoe identification and user interactions. At the core of the system is the SoleMate Management Engine (SME Controller), which acts as the central coordination unit. When a user submits a shoe identification request through the Request Identification module, the SME Controller processes the request by interacting with different subsystems. The Identification Subsystem plays a key role in this architecture, pulling data from the SSD Database and coordinating with various processing modules to refine identification accuracy. The Account Controller and Account Shoes Controller further support user interactions by managing account-related operations, such as creating, editing, and logging into accounts, as well as storing previously identified shoes for future reference.

The repository-style architecture is evident in how SoleMate manages persistent data across various subsystems. The SSD Database functions as the primary data repository, storing shoe identification results, historical searches, and expert ratings. The Account Shoes Database and Account Database store user-specific data, ensuring seamless retrieval of past interactions. The Support and Feedback System allows users to rate expert answers and interact with customer support in case of identification failures. This feedback is stored and utilized by the SME Controller to enhance the system's future performance. By leveraging a Blackboard approach for expert-driven identification and a repository structure for efficient data storage, SoleMate ensures a robust and scalable platform that enhances user experience and engagement.

4 Class Responsibility Collaboration (CRC) Cards

Class Name: Account Controller (Controller)	
Responsibility:	**Collaborators:**
Knows Edit Account	Edit Account
Knows Login	Login
Knows Create Account	Create Account
Knows View Account	View Account
Knows Account DB	Account DB
Knows User Information	User Information
Knows Account Shoes Controller	Account Shoes Controller
Knows SME Controller	SME Controller

Class Name: Account Shoes Controller (Controller)	
Responsibility:	**Collaborators:**
Knows Account Controller	Account Controller
Knows User Information	User Information
Knows Shoe Information	Shoe Information
Knows Account Shoes DB	Account Shoes DB
Knows Review Shoe History	Review Shoe History
Knows SME Controller	SME Controller

Class Name: Account DB (Entity)	
Responsibility:	**Collaborators:**
Knows Account Controller	Account Controller
Knows User Information	User Information

Class Name: Edit Account (Boundary)	
Responsibility:	**Collaborators:**
Knows Account Controller Knows username, password, gender etc. Handles click-event of "Save" button	Account Controller

Class Name: Login (Boundary)	
Responsibility:	**Collaborators:**
Knows Account Controller Handles click-event of "login" button	Account Controller

Class Name: Create Account (Boundary)	
Responsibility:	**Collaborators:**
Knows Account Controller Knows username and password Handles click-event of "Create Account" Button	Account Controller

Class Name: View Account (Boundary)	
Responsibility:	**Collaborators:**
Knows Account Controller Handles click-event of "View Account" button Displays User Information	Account Controller

Class Name: Account Shoes DB (Entity)	
Responsibilities:	**Collaborators:**
Knows Account Shoes Controller	Account Shoes Controller

Class Name: User Information (Entity)	
Responsibilities:	**Collaborators:**
Knows Account DB Knows Account Controller Knows Account Shoes Controller Knows user information like username, password, gender etc.	Account DB Account Controller Account Shoes Controller

Class Name: Review Shoe History (Boundary)	
Responsibilities:	**Collaborators:**
Knows Account Shoes Controller	Account Shoes Controller

Class Name: Rate Expert Answer (Boundary)	
Responsibilities:	**Collaborators:**
Knows ratings for expert answers Handles user input for rating experts' answers Displays the rating results	SME Controller

Class Name: Shoe Information (Entity)	
Responsibilities:	**Collaborators:**
Knows detailed shoe information	SSD DB
Knows shoe details and features	SME Controller
Handles filtering and search functionalities	Account Shoe Controller
Displays shoe information for users	

Class Name: Search Identified Shoes (Boundary)	
Responsibilities:	**Collaborators:**
Handles shoe identification based on user queries	SME Controller
Displays retrieved and identified shoes	
Handles sorting options for identified shoes	

Class Name: Request Identification (Boundary)	
Responsibilities:	**Collaborators:**
Handles requests for shoe identification	SME Controller
Handles initiation of search for shoe identification	
Knows the status of an identification request	

Class Name: Search Results Page (Boundary)	
Responsibilities:	**Collaborators:**
Displays the results from shoe searches	SME Controller
Displays organized search data to users	
Handles filtering and sorting of search results	
Displays links to detailed shoe information	

Class Name: SME Controller (Controller)	
Responsibilities:	**Collaborators:**
Handles identification outcomes	Identification Success
Handles system-support communication	Contact Support Message
Handles user search requests	Search Identified Shoes
Displays search results	Search Results Page
Handles user accounts	Account Controller
Handles user shoe preferences	Account Shoe Controller
Knows shoe data	Shoe Information
Handles database access	SSD DB
Knows expert ratings	Rate Expert Answer
Handles shoe identification requests	Request Identification
Displays identification results	Identification Failure
Displays error messages	
Handles search filtering and sorting	
Handles search pagination	
Knows system-wide data consistency	
Knows user activity for recommendations	

Class Name: Identification Failure (Boundary)	
Responsibility:	**Collaborators:**
Handles identification failures	SME Controller
Displays identification failure notification to user	

Class Name: Identification Success (Boundary)	
Responsibility:	**Collaborators:**
Handles successful identification processes Displays search results after identification	SME Controller

Class Name: Contact Support Message (Boundary)	
Responsibility:	**Collaborators:**
Handles initiating contact with customer support Displays the support contact information	SME Controller

Class Name: SSD DB (Entity)	
Responsibilities:	**Collaborators:**
Knows shoe-related search results Displays data for search results Knows detailed shoe attributes Knows price trends and availability Handles updates to shoe-related information Handles database indexing and queries	SME Controller Shoe Information

A Division of Labour

- Introduction (Section 1): Ryan Kumar
- Analysis Class Diagram (Section 2): Lucas DeBoer
- Architectural Design (Section 3): Alvin Qian & Varun Pathak
- Class Responsibility Collaboration (CRC) Cards (Section 4): Moustafa Moustafa & Maya Azar

Signatures: