Deliverable #3

SE 3A04: Software Design II – Large System Design

Group Number: G8
Group Members:

- Hashim Bukhtiar
- Jaden Moore
- James Ariache
- Olivia Reich
- Omar Abdelhamid

1 Introduction

1.1 Purpose

This document provides further information about the RideRecon car identification system architecture, including state chart diagrams, sequence diagrams, and a detailed class diagram. This document is intended for internal RideRecon stakeholders, including but not limited to, project managers, developers, domain experts, and RideRecon team members/investors.

RideRecon Deliverable 1 and 2 should be read prior, and technical knowledge may be beneficial in better understanding the contents of the document.

1.2 System Description

The RideRecon system is a vehicle identification platform that processes user-provided text and image inputs to accurately recognize and classify cars. The system leverages a hybrid blackboard-repository architecture, where multiple expert modules—including a trained ML model (G8M), a reverse image search engine, vector imaging techniques, and GPT-based text processing—collaborate to determine the most accurate identification. The blackboard component acts as a shared workspace where expert modules contribute insights, iteratively refining the identification, while the repository component ensures efficient data management, storing user inputs, past identifications, and expert processing outcomes for future reference.

This document builds on the foundational system description provided in Deliverable 2, expanding the technical details through state charts, sequence diagrams, and a detailed class diagram. These artifacts provide a deeper understanding of the system's data flow, processing logic, and user interactions. The diagrams illustrate how user inputs progress through various system components, from input validation to expert analysis and final result generation. Additionally, this document clarifies how user accounts, history tracking, and car collection management integrate into the broader system. Through these detailed design elements, we establish a clear roadmap for the system's implementation and behavior.

1.3 Overview

This document provides a structured analysis and design of the **RideRecon** system, focusing on the **State Chart Diagrams**, **Sequence Diagrams**, and a detailed **Class Diagram**. It expands upon the foundational concepts introduced in **Deliverables 1 and 2**, ensuring a comprehensive understanding of the system's behavior and interactions.

Section 2 presents **State Chart Diagrams**, which define the behavior of core controller classes, particularly focusing on user management and vehicle identification. These diagrams illustrate the different states a system entity can transition between and the conditions that trigger these transitions. **Section 3** provides **Sequence Diagrams**, visually representing the flow of interactions between system components during specific use cases. These diagrams capture step-by-step interactions between users, expert modules, and the system's processing units. **Section 4** contains the **Class Diagram**, outlining the structure of the system's classes, their attributes, relationships, and methods. This diagram ensures clarity in class responsibilities, making it easier to understand how data is managed within the system.

Together, these sections form a comprehensive guide to understanding the **RideRecon system's state management, process flow, and class relationships**, providing essential details for implementation and further development.

2 State Charts for Controller Classes

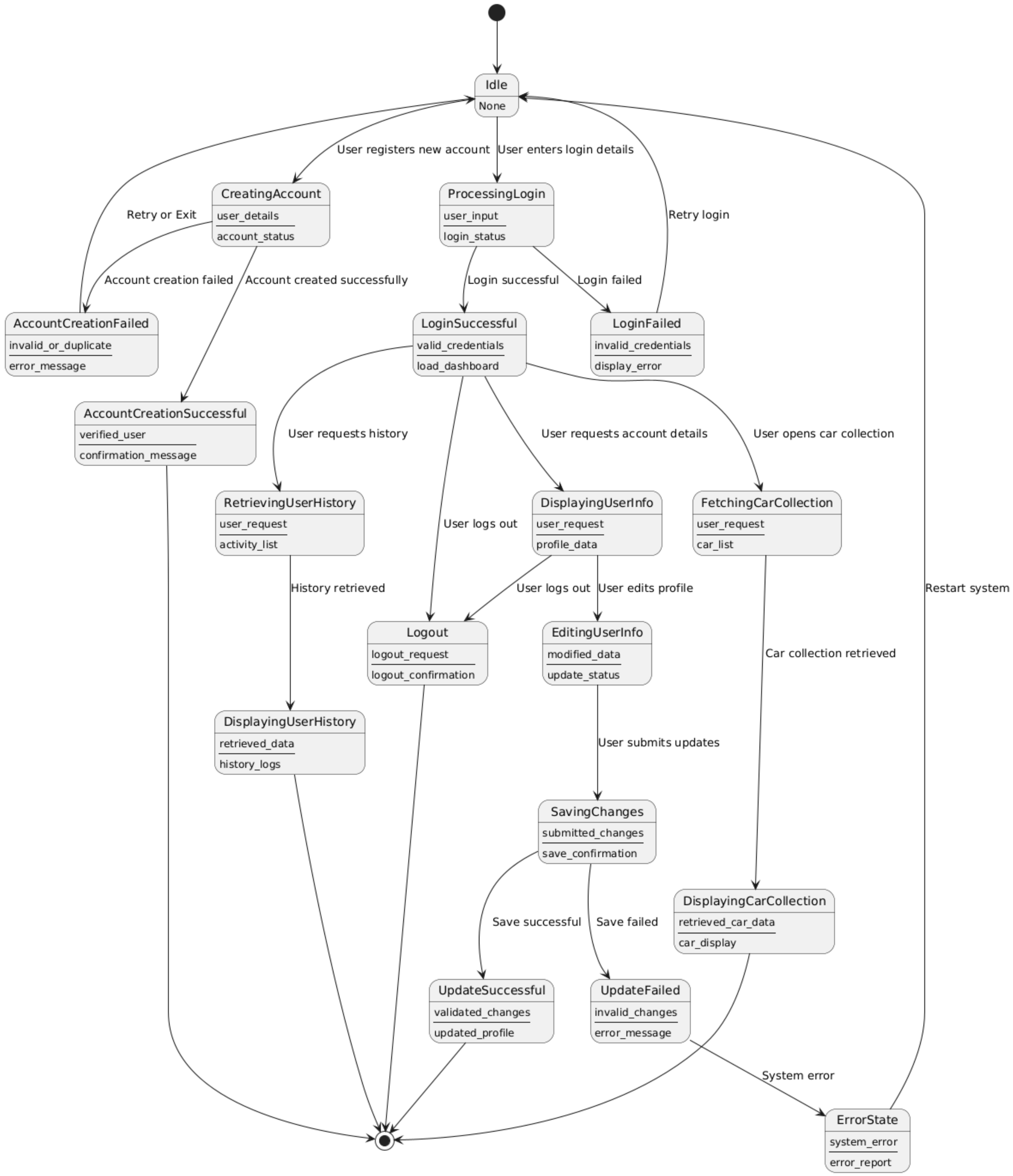

Figure 1. State Chart Diagram for User Manager.

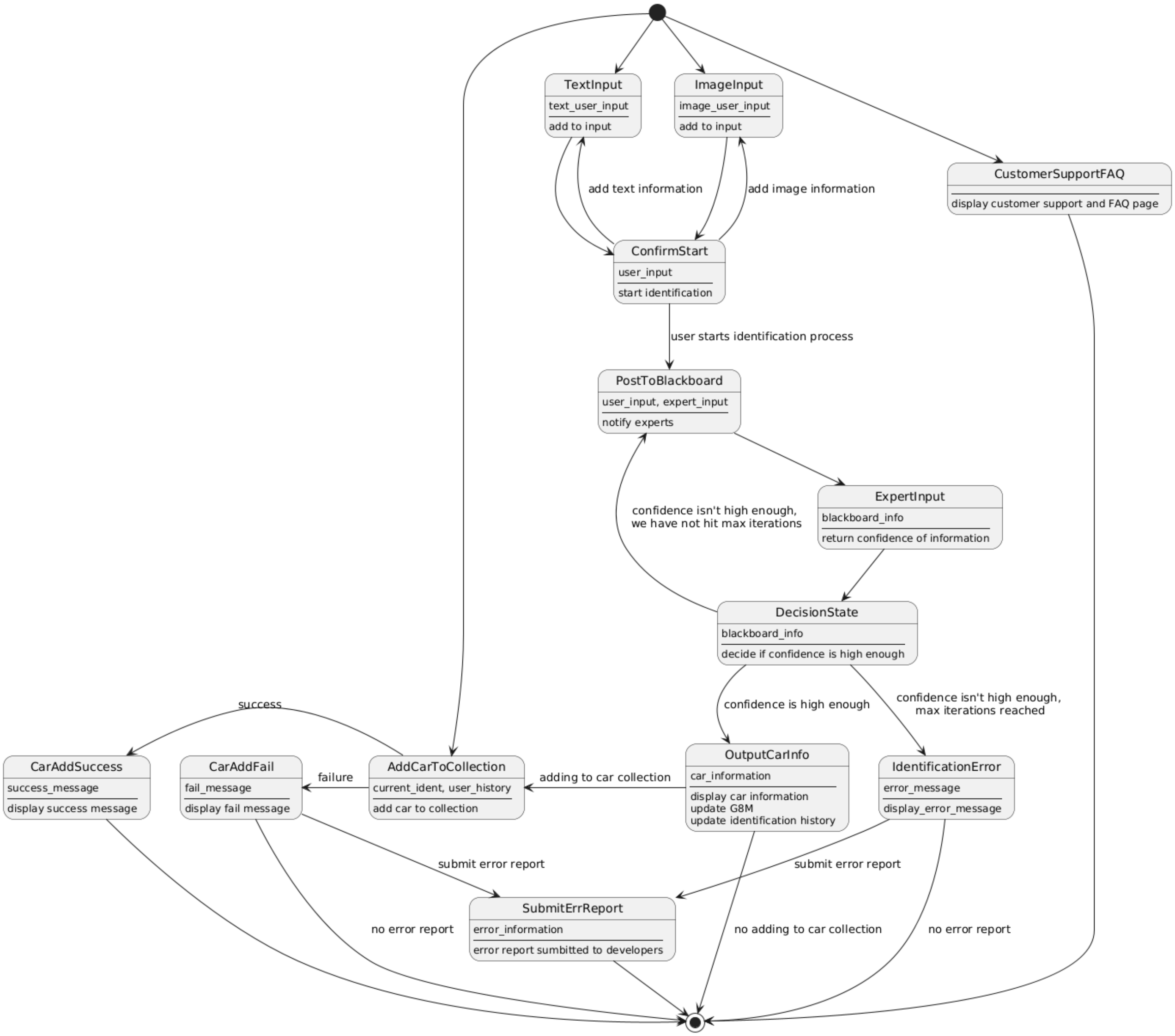

Figure 2. State Chart Diagram for Identification Manager.

3 Sequence Diagrams

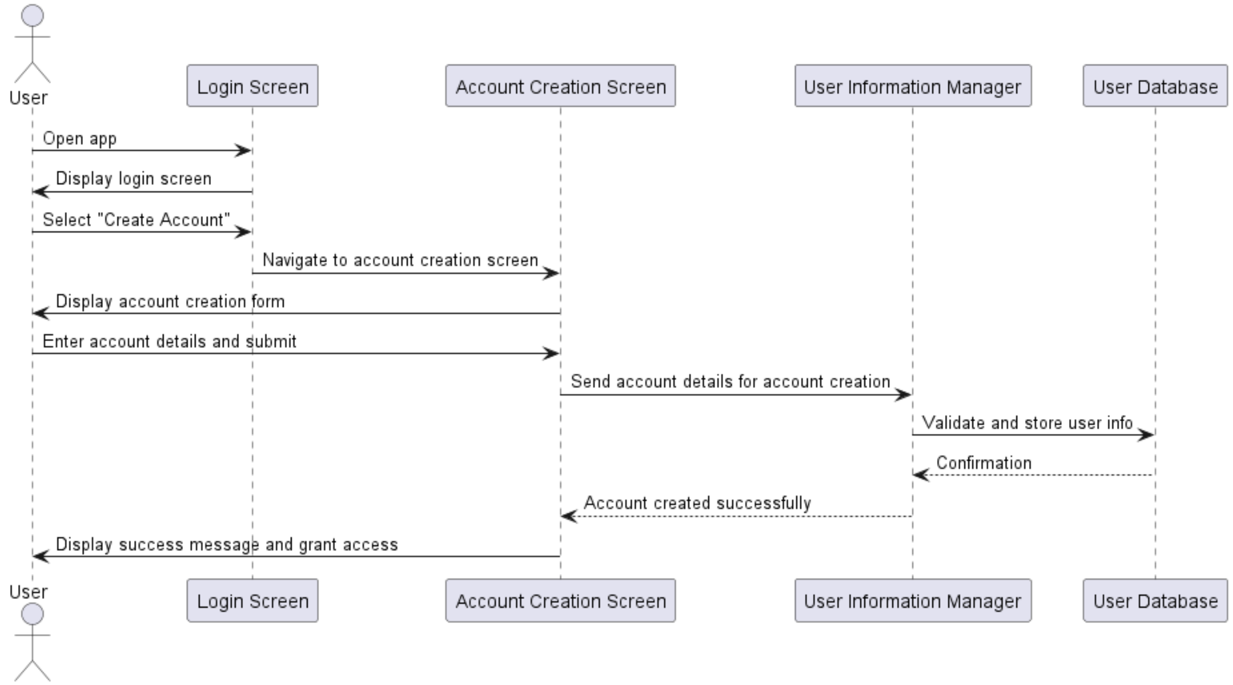
User
Login Screen
Account Creation Screen
User Information Manager
User Database
Open app
Display login screen
Select "Create Account"
Navigate to account creation screen
Display account creation form
Enter account details and submit
Send account details for account creation
Validate and store user info
Confirmation
Account created successfully
Display success message and grant access
User
Login Screen
Account Creation Screen
User Information Manager
User Database

Figure 3. Sequence Diagram for BE1: Create Account.

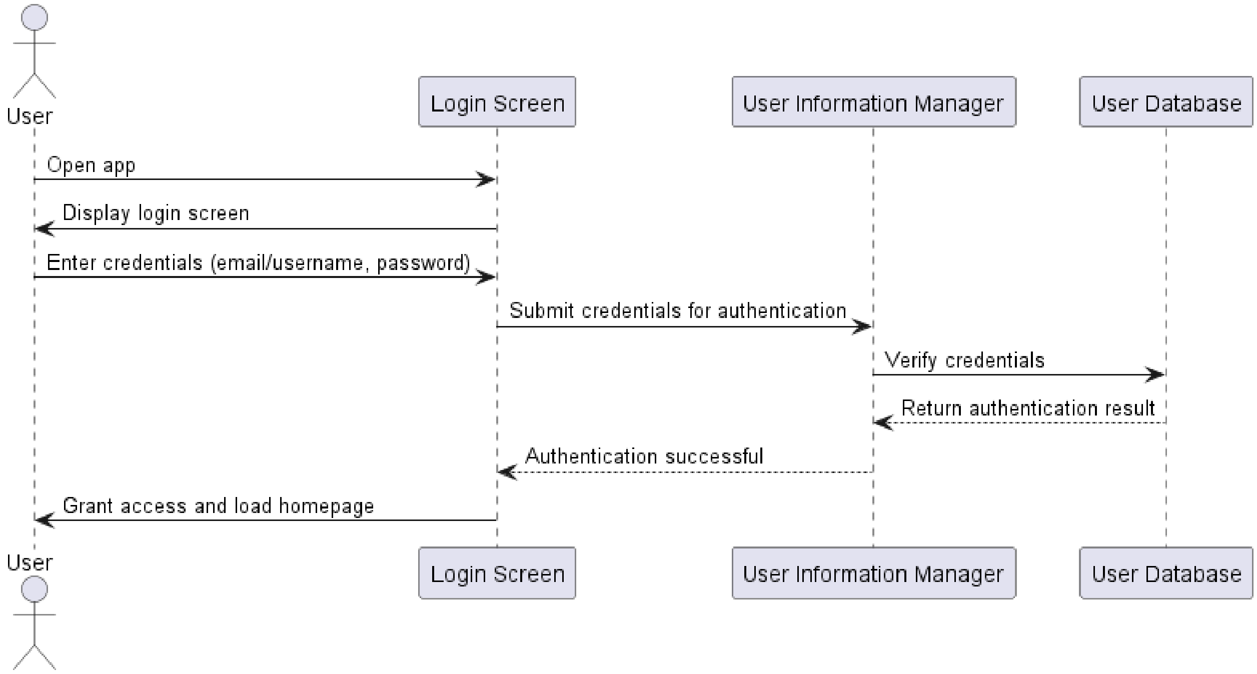
User
Login Screen
User Information Manager
User Database
Open app
Display login screen
Enter credentials (email/username, password)
Submit credentials for authentication
Verify credentials
Return authentication result
Authentication successful
Grant access and load homepage
User
Login Screen
User Information Manager
User Database

Figure 4. Sequence Diagram for BE2: Authenticate User.

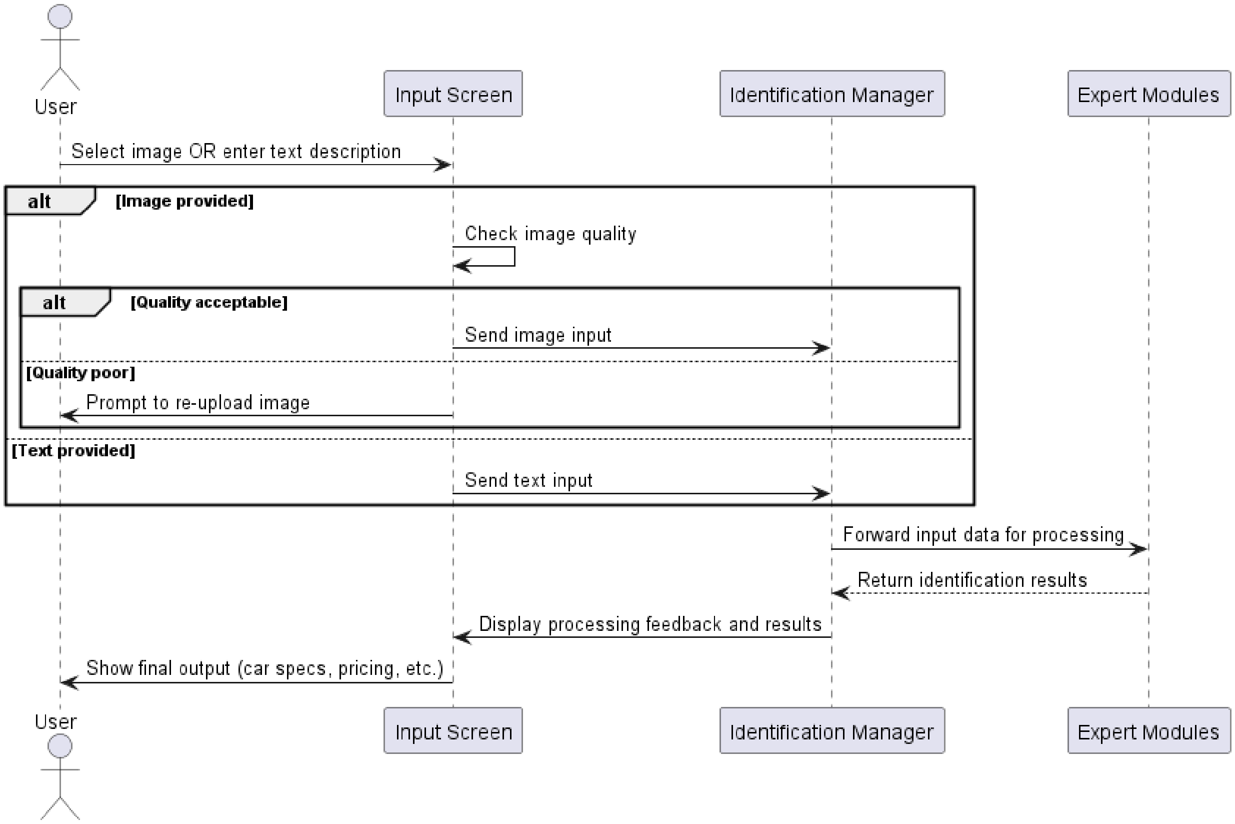

Figure 5. Sequence Diagram for BE3: Upload Text and/or Image as Input.

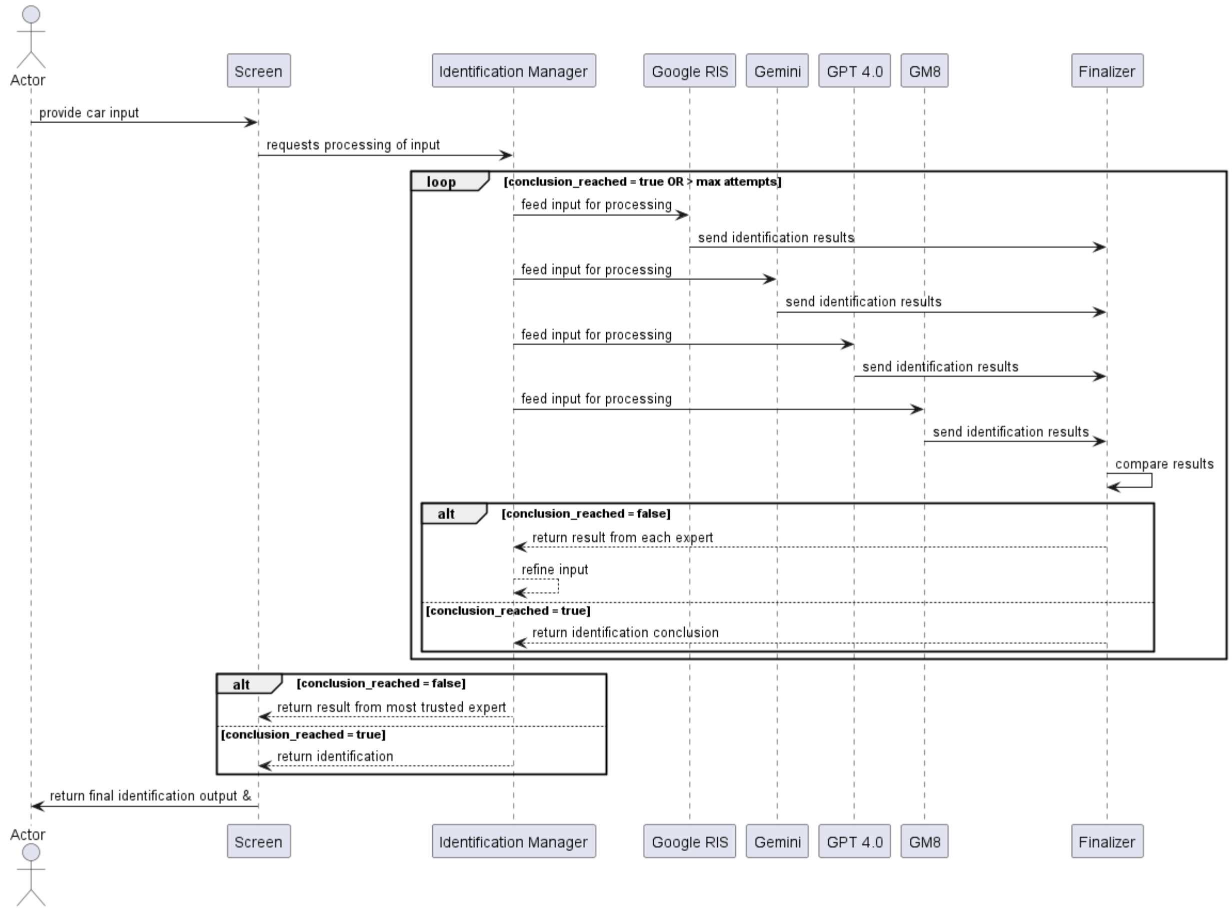
Actor
Screen
Identification Manager
Google RIS
Gemini
GPT 4.0
GM8
Finalizer
provide car input
requests processing of input
loop
[conclusion_reached = true OR > max attempts]
feed input for processing
send identification results
feed input for processing
send identification results
feed input for processing
send identification results
feed input for processing
send identification results
compare results
alt
[conclusion_reached = false]
return result from each expert
refine input
[conclusion_reached = true]
return identification conclusion
alt
[conclusion_reached = false]
return result from most trusted expert
[conclusion_reached = true]
return identification
return final identification output &
Actor
Screen
Identification Manager
Google RIS
Gemini
GPT 4.0
GM8
Finalizer

Figure 6. Sequence Diagram for BE4: Compare Expert Answers.

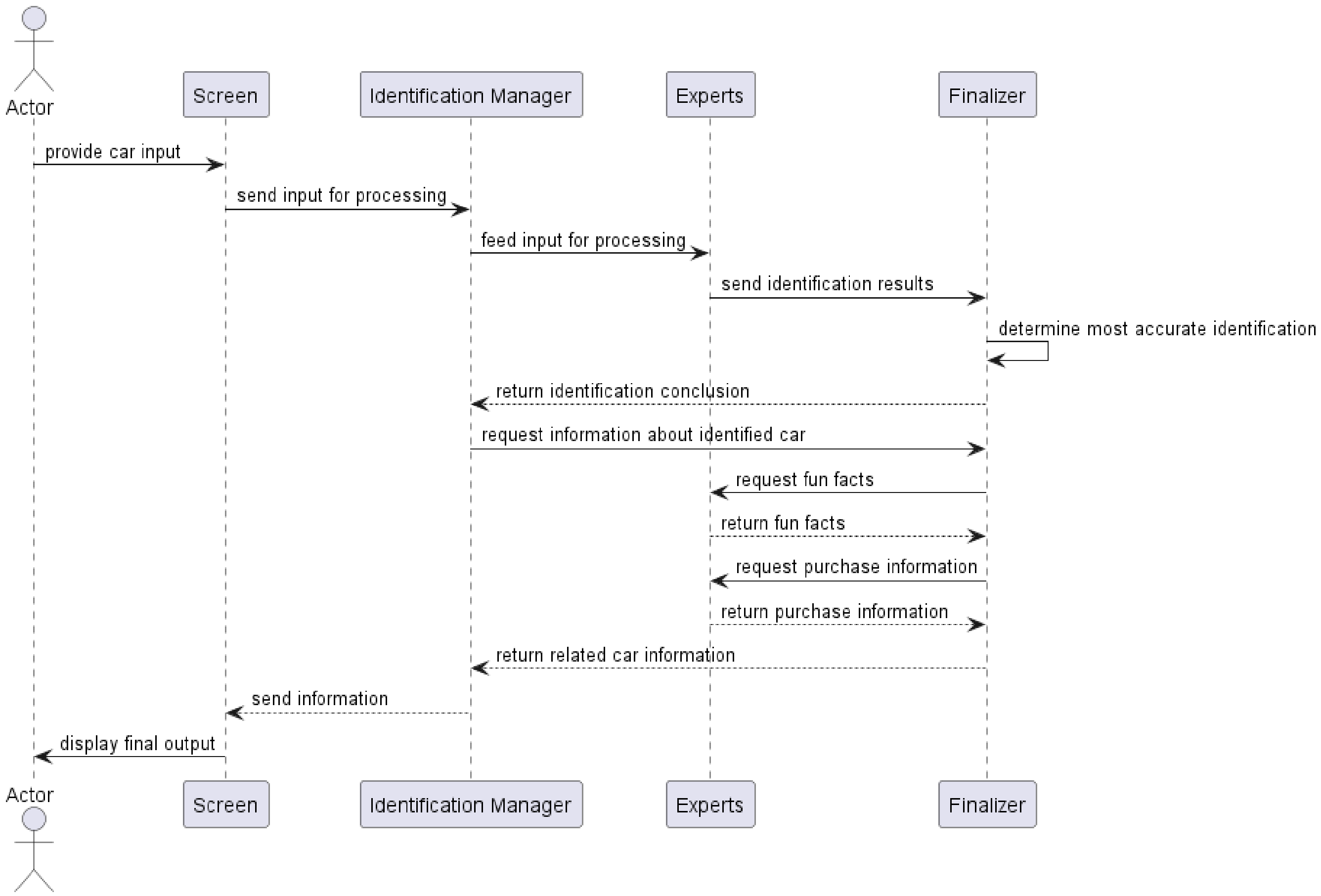
Actor
Screen
Identification Manager
Experts
Finalizer
provide car input
send input for processing
feed input for processing
send identification results
determine most accurate identification
return identification conclusion
request information about identified car
request fun facts
return fun facts
request purchase information
return purchase information
return related car information
send information
display final output
Actor
Screen
Identification Manager
Experts
Finalizer

Figure 7. Sequence Diagram for BE5: Present Final Output With Identification Information.

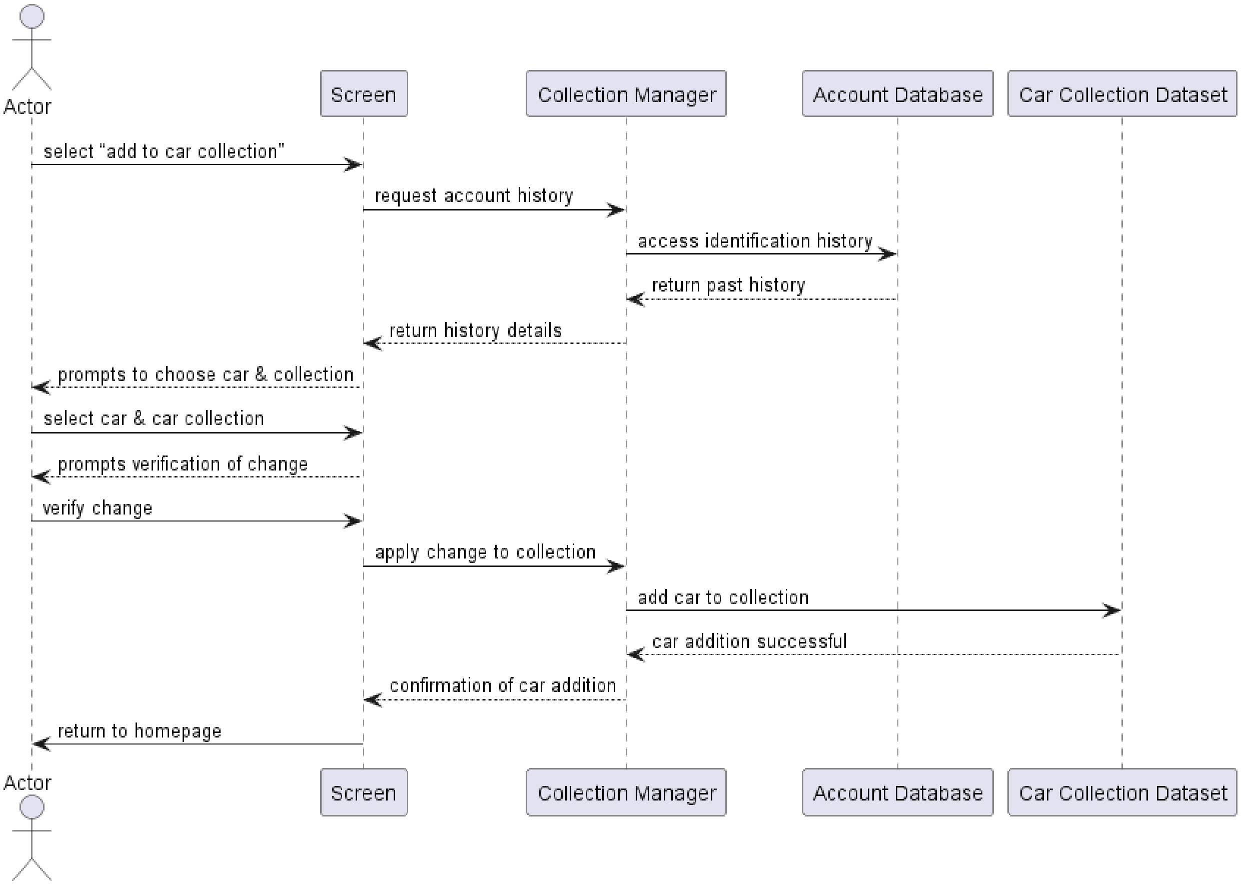

Figure 8. Sequence Diagram for BE6: Add to or Remove From Car Collection.

4 Detailed Class Diagram

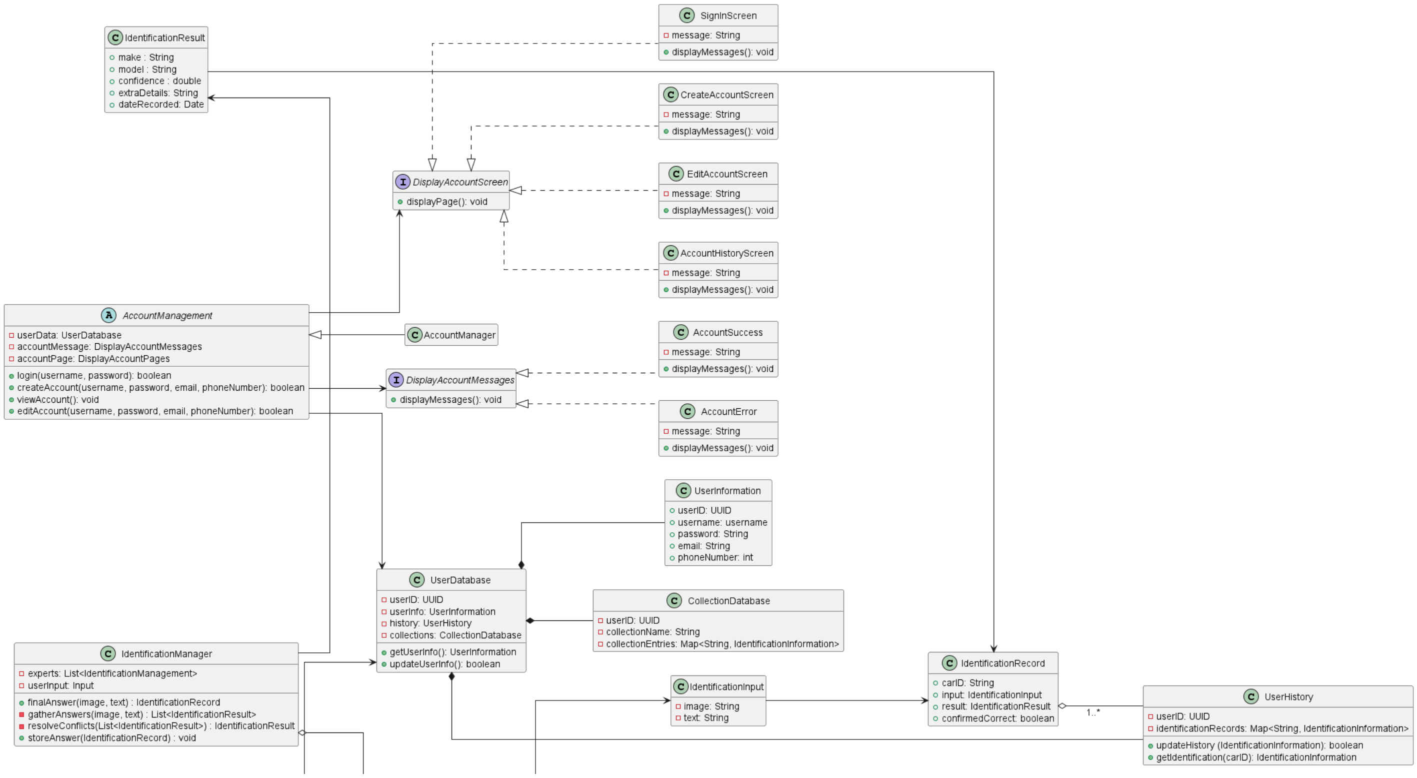
IdentificationResult
make : String
model : String
confidence : double
extraDetails: String
dateRecorded: Date
SignInScreen
message: String
displayMessages(): void
CreateAccountScreen
message: String
displayMessages(): void
DisplayAccountScreen
displayPage(): void
EditAccountScreen
message: String
displayMessages(): void
AccountHistoryScreen
message: String
displayMessages(): void
AccountManagement
userData: UserDatabase
accountMessage: DisplayAccountMessages
accountPage: DisplayAccountPages
login(username, password): boolean
createAccount(username, password, email, phoneNumber): boolean
viewAccount(): void
editAccount(username, password, email, phoneNumber): boolean
AccountManager
AccountSuccess
message: String
displayMessages(): void
DisplayAccountMessages
displayMessages(): void
AccountError
message: String
displayMessages(): void
UserInformation
userID: UUID
username: username
password: String
email: String
phoneNumber: int
UserDatabase
userID: UUID
userInfo: UserInformation
history: UserHistory
collections: CollectionDatabase
getUserInfo(): UserInformation
updateUserInfo(): boolean
CollectionDatabase
userID: UUID
collectionName: String
collectionEntries: Map<String, IdentificationInformation>
IdentificationManager
experts: List<IdentificationManagement>
userInput: Input
finalAnswer(image, text) : IdentificationRecord
gatherAnswers(image, text) : List<IdentificationResult>
resolveConflicts(List<IdentificationResult>) : IdentificationResult
storeAnswer(IdentificationRecord) : void
IdentificationInput
image: String
text: String
IdentificationRecord
carID: String
input: IdentificationInput
result: IdentificationResult
confirmedCorrect: boolean
1..*
UserHistory
userID: UUID
identificationRecords: Map<String, IdentificationInformation>
updateHistory (IdentificationInformation): boolean
getIdentification(carID): IdentificationInformation

Figure 9. First half of detailed Class Diagram of RideRecon system. If blurry, zooming in will clear up the text.

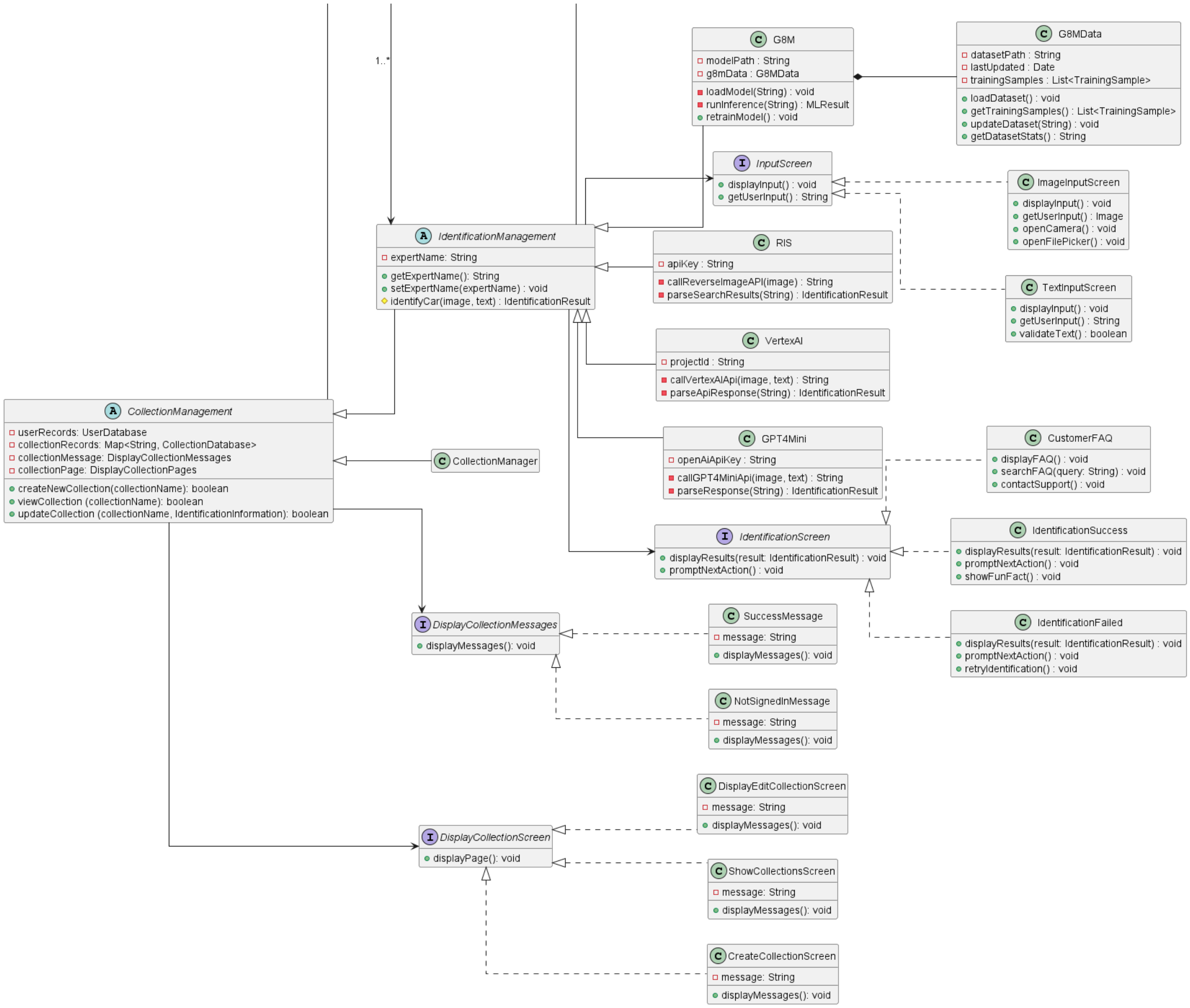
1..*
G8M
modelPath : String
g8mData : G8MData
loadModel(String) : void
runInference(String) : MLResult
retrainModel() : void
G8MData
datasetPath : String
lastUpdated : Date
trainingSamples : List<TrainingSample>
loadDataset() : void
getTrainingSamples() : List<TrainingSample>
updateDataset(String) : void
getDatasetStats() : String
InputScreen
displayInput() : void
getUserInput() : String
ImageInputScreen
displayInput() : void
getUserInput() : Image
openCamera() : void
openFilePicker() : void
IdentificationManagement
expertName: String
getExpertName(): String
setExpertName(expertName) : void
identifyCar(image, text) : IdentificationResult
RIS
apiKey : String
callReverseImageAPI(image) : String
parseSearchResults(String) : IdentificationResult
TextInputScreen
displayInput() : void
getUserInput() : String
validateText() : boolean
VertexAI
projectId : String
callVertexAIApi(image, text) : String
parseApiResponse(String) : IdentificationResult
CollectionManagement
userRecords: UserDatabase
collectionRecords: Map<String, CollectionDatabase>
collectionMessage: DisplayCollectionMessages
collectionPage: DisplayCollectionPages
createNewCollection(collectionName): boolean
viewCollection (collectionName): boolean
updateCollection (collectionName, IdentificationInformation): boolean
CollectionManager
GPT4Mini
openAiApiKey : String
callGPT4MiniApi(image, text) : String
parseResponse(String) : IdentificationResult
CustomerFAQ
displayFAQ() : void
searchFAQ(query: String) : void
contactSupport() : void
IdentificationScreen
displayResults(result: IdentificationResult) : void
promptNextAction() : void
IdentificationSuccess
displayResults(result: IdentificationResult) : void
promptNextAction() : void
showFunFact() : void
DisplayCollectionMessages
displayMessages(): void
SuccessMessage
message: String
displayMessages(): void
IdentificationFailed
displayResults(result: IdentificationResult) : void
promptNextAction() : void
retryIdentification() : void
NotSignedInMessage
message: String
displayMessages(): void
DisplayEditCollectionScreen
message: String
displayMessages(): void
DisplayCollectionScreen
displayPage(): void
ShowCollectionsScreen
message: String
displayMessages(): void
CreateCollectionScreen
message: String
displayMessages(): void

Figure 10. Second half of detailed Class Diagram of RideRecon system. If blurry, zooming in will clear up the text.

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.

Hashim Bukhtiar	Jaden Moore	James Ariache	Olivia Reich	Omar Abdelhamid
Section 1.1, 1.2 3 Sequence Diagrams in Section 3 Compiled Final Doc	Class Diagram in Section 4	Second State Chart in Section 2 Helped with Class Diagram in Section 4 James Ariache	3 Sequence Diagrams in Section 3 Class Diagram in Section 4	Section 1.3 First State Chart in Section 2 Omar Hassan

Table 1: Division of Labour

IMPORTANT NOTES

- You do NOT need to provide a text explanation of each diagram; the diagram should speak for itself
- 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
  - It is OK if the text is small but please ensure that it is readable when printed
  - If you need to break a diagram onto multiple pages, please adopt a system of doing so and throughly explain how it can be reconnected from one page to the next; if you are unsure about this, please ask me
- Please submit the latest version of Deliverable 1 and Deliverable 2 with Deliverable 3
  - They do not have to be a freshly printed versions; the latest marked versions are OK
- If you do NOT have a Division of Labour sheet, your deliverable will NOT be marked