Deliverable #3 Template

Keyur, Joshua, Abdullah, Justin, Bilal, Shaad

1 Introduction

1.1 Purpose

The purpose of this document is to provide the software development team an overall guide to the architecture of this project. This document is published to outline all the parts of the MovieFinder software and to be a reference point for the team developing the application. The intended audience of this document are the software development team of MovieFinder, as well as the professor and teaching assistants reviewing this design document.

1.2 System Description

The application MovieFinder has three main 'screens.' When the application is launched, the user is displayed the main menu, where they can enter the information they know about the movie. Once they submit that information, the server will then recieve the query and process the query. Once the application has processed the results from the agents, the results will be sent to the client, and be displayed on the results menu. Once on the results menu, users will have the option of sorting the results by certain qualities including but not limited to rating or release date. Users can then choose a single movie to receive more information about. The server will use a universal movie database to get the information. This information takes users to the third screen, which is the information menu.

1.3 Overview

This document contains state charts for the controller classes of the application. These state charts provide a full description of the behaviour of the controller classes. Next, the design document contains the sequence diagrams. These sequence diagrams illustrate the flow of information from class to class under different scenarios. Lastly, this document shows the detailed class diagrams required to develop the MovieFinder. The detailed class diagrams provide a blueprint for the classes and their methods, as well as how they are connected to each other.

2 State Charts for Controller Classes

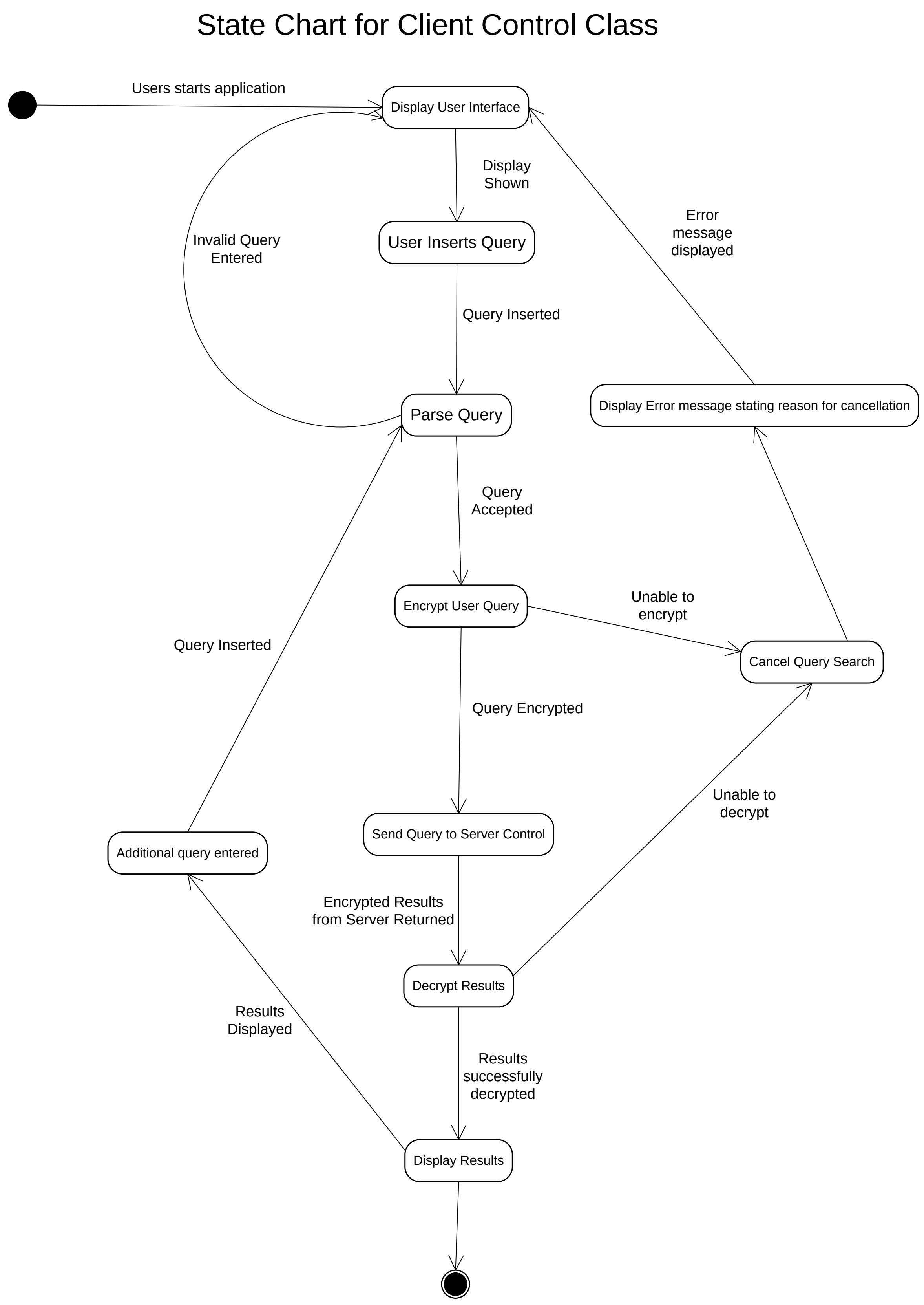

Figure 1: State chart for client

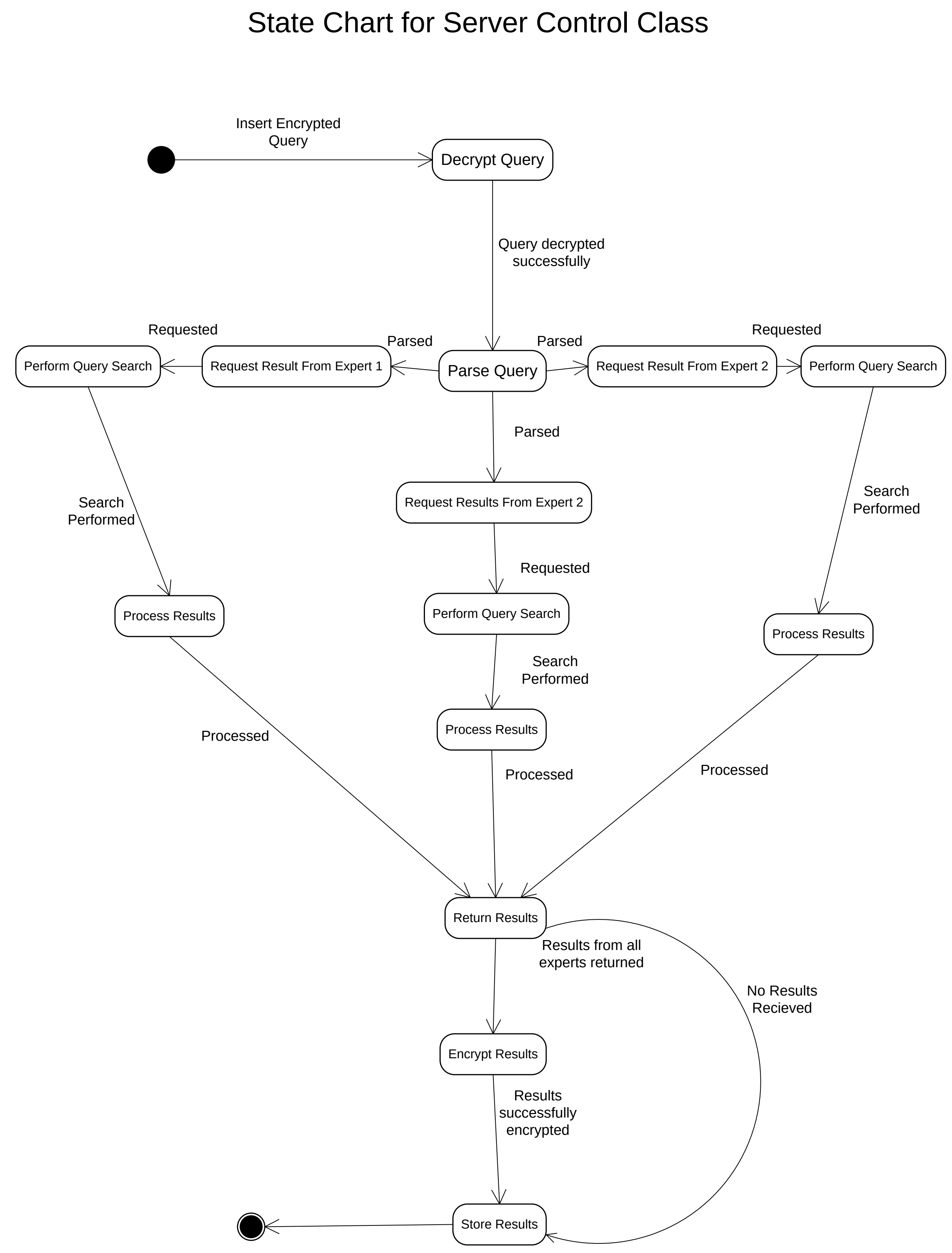

Figure 2: State chart for server

3 Sequence Diagrams

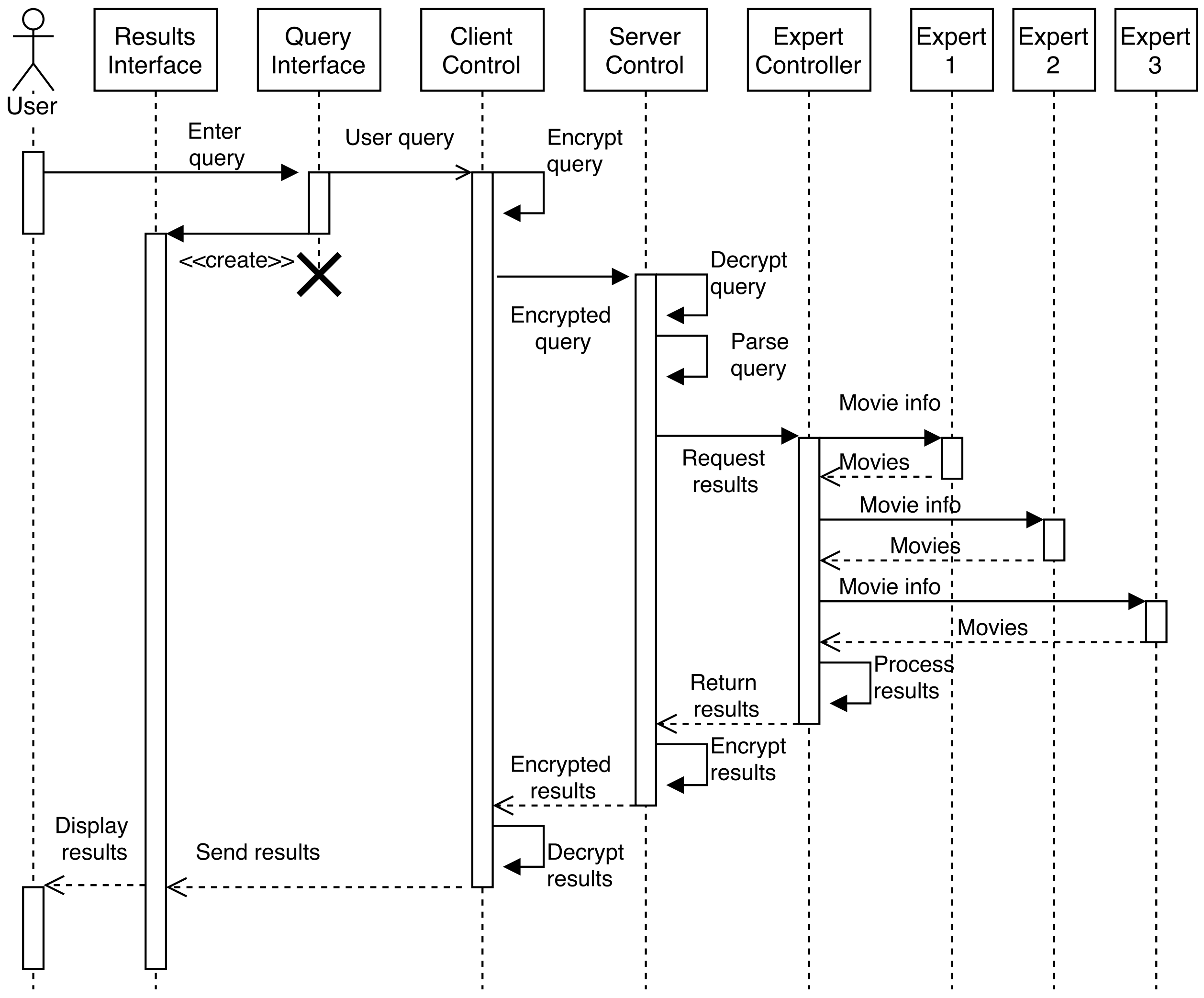

Figure 3: Sequence diagram for entering a query

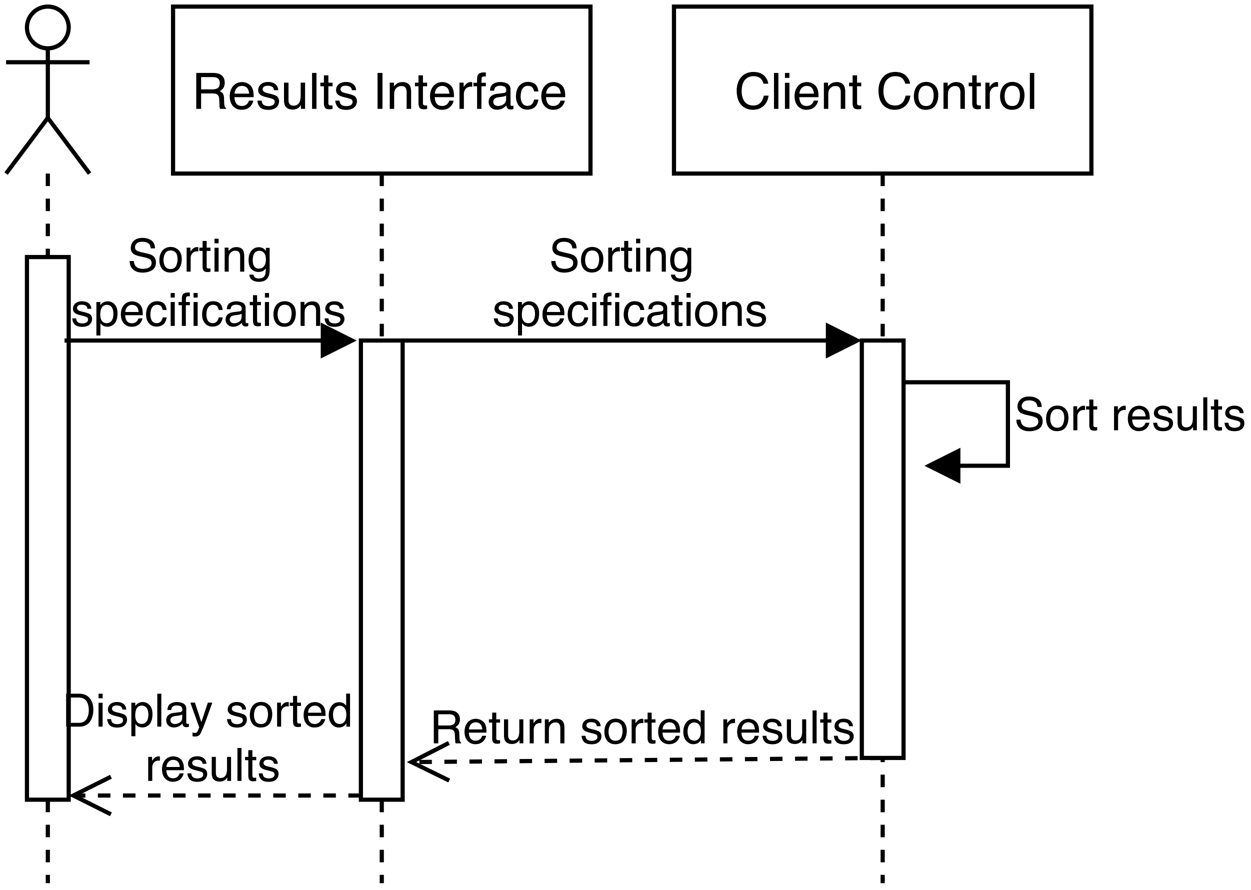

Figure 4: Sequence diagram for sorting the results

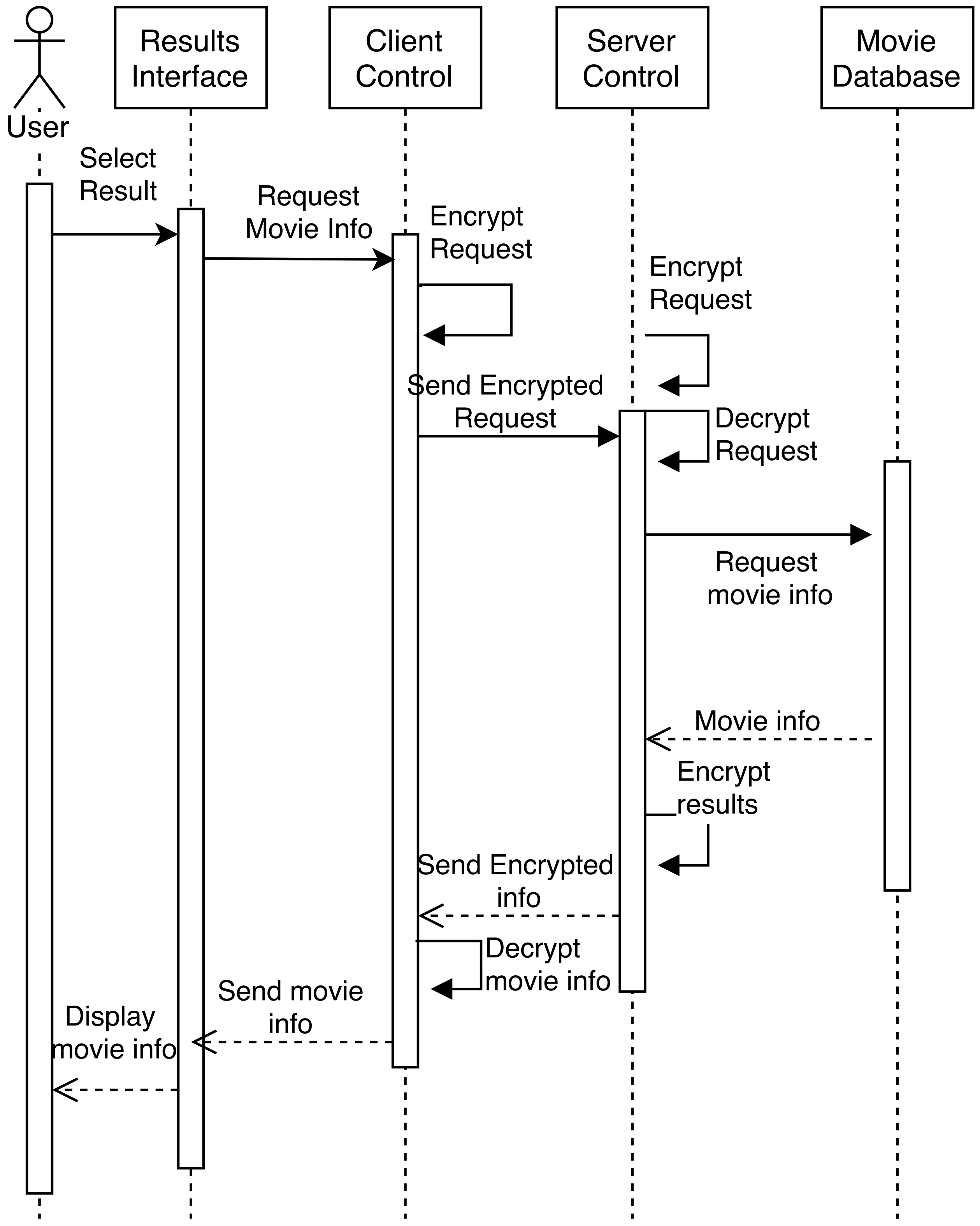

Figure 5: Sequence diagram for viewing individual movie information

4 Detailed Class Diagram

Client Control
- clientID: int - clientquery(): Object - queryResponse(): Object
+ constructor() + init() + listener(): void + recordQuery(q: Query): void + sendQuery(): Object

Result Interface
- query: String
+ constructor() + setResult(q: String): void + getResult(l: String): String + displayResult(): void + removeResult(): void

Query Form
- selection: Object
+ constructor() + getResult(result: Object): void + actionPerformed(event: Object): void + mouseClicked(event: Object): void + sendResult(result: Object): Object

Server Control
- expertList: Expert[3] = [expert1, expert2, expert3] - clientID: int[] = null - clientQuery: object[] = null - queryResponse: object[] = null

+ contructor()
+ init()
+ listen()
+ readQuery(Query query): object
+ sendResult(Object object)
+ lockExpert(String expert)

Expert 1 Control

+ constructor()
+ init()
+ update()
+ search(Object query): Object[]
+ addItem(Object newObject)
+ removeItem(Object oldObject): Object

Expert 1 Database
- movieList: String[] = null - paramterList: enumerate = 1..N - parameterData1: Object[] = null - paramterDataN: Object[] = null

+ constructor()
+ deconstructor()
+ init()

Expert 2 Control

+ constructor()
+ init()
+ update()
+ search(Object query): Object[]
+ addItem(Object newObject)
+ removeItem(Object oldObject): Object

Expert 2 Database
- movieList: String[] = null - paramterList: enumerate = 1..N - parameterData1: Object[] = null - paramterDataN: Object[] = null

+ constructor()
+ deconstructor()
+ init()

Expert 3 Control

+ constructor()
+ init()
+ update()
+ search(Object query): Object[]
+ addItem(Object newObject)
+ removeItem(Object oldObject): Object

Expert 3 Database
- movieList: String[] = null - paramterList: enumerate = 1..N - parameterData1: Object[] = null - paramterDataN: Object[] = null

+ constructor()
+ deconstructor()
+ init()

A Division of Labour

Contributions			
Name	**Student Number**	**Contribution**	**Signature**
Joshua	1311940	Detailed Class Diagrams	
Keyur	1311559	Introduction & Sequence Diagram	
Justin	1305257	Sequence Diagrams	
Bilal	1320763	State Charts for Controller Classes	
Shaad	1335602	State Charts for Controller Classes	
Abdullah	1317053	Detailed Class Diagrams	

IMPORTANT NOTES

- You do <u>NOT</u> 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 <u>NOT</u> have a Division of Labour sheet, your deliverable will <u>NOT</u> be marked