Large System Design

Carspot for SE 3A04, Tutorial 2

Yasaswi Gopalkrishnan Sharon Platkin Abhijit Singh Dhoat

Joseph Cole Huot David Eric Hemms Yuchen Liu

Monday March 7th, 2016

Contents

List of Tables

1 Introduction

1.1 Purpose

The purpose of this document is to provide an outline of the entire system of the android application. The diagrams used in this document identify all main elements and components of the system. The intended audiences for this document are software engineers who intended to work on 3A04 android application project and their instructional staff. The document may be edited if any changes took place on the software requirements specification document.

1.2 System Description

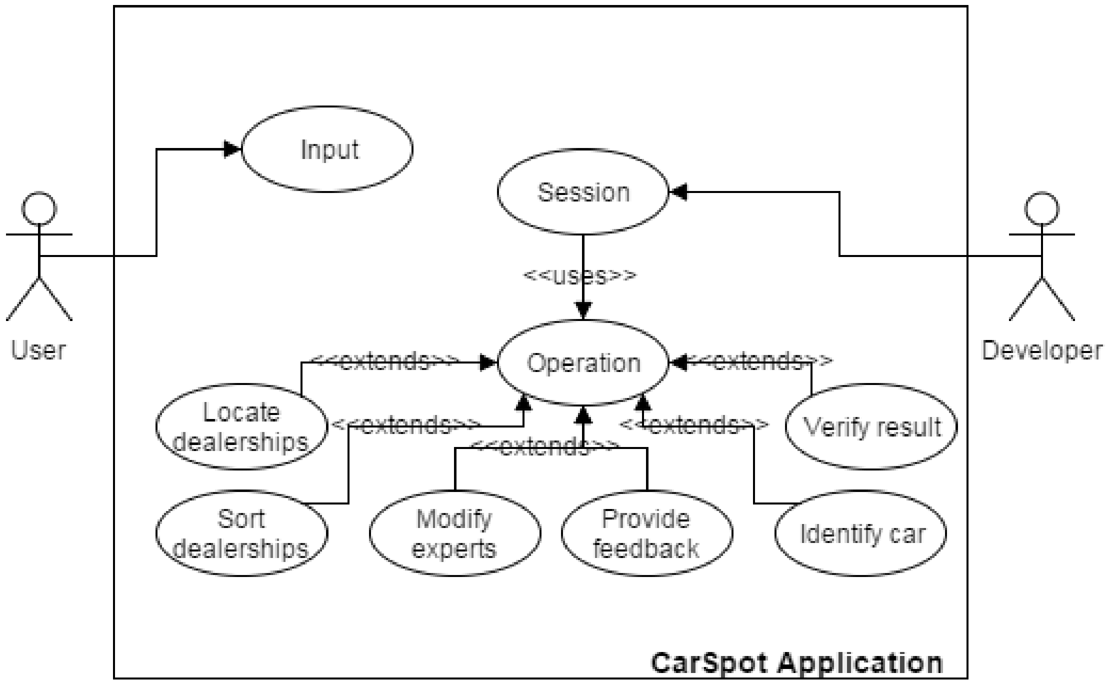
The system will be made available to the general public through the Google Play Store on the Android Operating System. Users will download and use the app to identify cars they spot in their day to day lives.

The user will be prompted by the system to pick from a set of categories they recognize. They will pick 3 of the these: car type, a logo, number of passengers or origin. The system will then search based on the information the user provides. Once the system has successfully run the search algorithm, it will present the user with the name, model, make and a small picture of the car. It will present multiple results ordered by match percentage. The user can then select a result and then search for local dealerships in an area that sell the car, or re-perform the search. The system will also hold a history of the 5 recent searches.

1.3 Overview

The document contains information about the architectural design of the application CarSpot. Various diagrams were used to portray this information. These include a use case diagram, an analysis class diagram, the architectural design diagram, and class responsibility collaboration cards. A depiction of the objects, attributes, and relationships is explained as well as an explanation of the interactions of data flow between objects. Furthermore, responsibilities of and collaborators to classes have been stated in the document. The last part of the document is the division of labour that used to declare how the group members collaborated to produce this document.

2 Use Case Diagram



i) Input : The user will input information about the car in question. The system will allow the user to navigate through the application using buttons and provide the application with information using text boxes.

ii) Session: When a user opens the application, a session is started. During a session, the process for identify a car and locating leaderships begins. A list of recent identified cars is saved. The session ends when the application is closed, or if the user wishes to start the process over.

iii) Operation: This is an abstract use case. It extends and includes different operations that the application will perform for the user. These entail: Swap experts, Verify result, Identify car, Locate dealership, Sort dealerships, Provide feedback

iv) Modify experts: Will add/remove/swap experts in and out of the identifier questioner.

v) Verify result: The result that the application came up with will be verified or denied. If denied, application will promt to re-assess the information given and experts used.

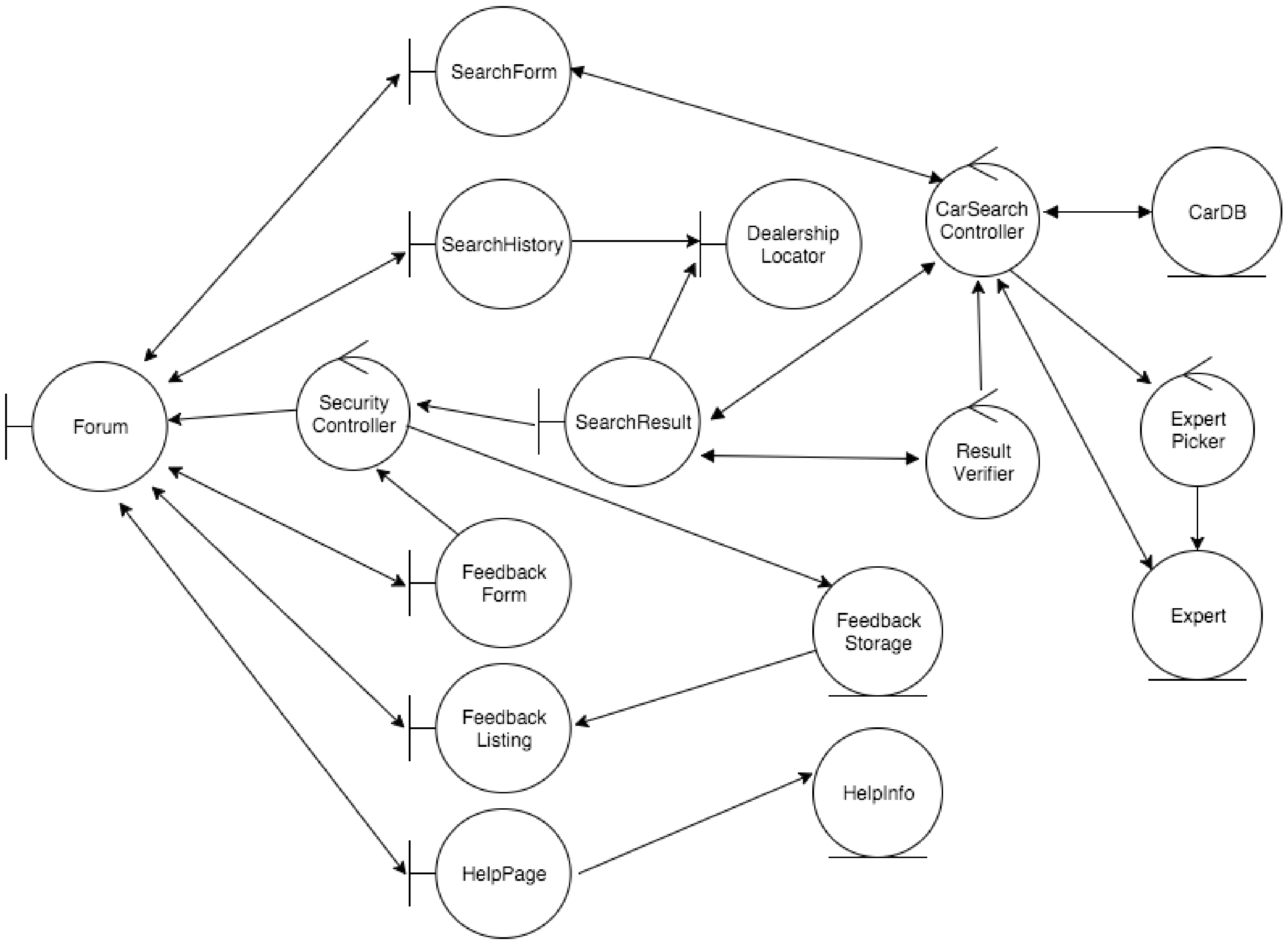
vi) Identify car: User will inform application that they are done inputting information. Using all the information inputted, the application will attempt to identify the correct car.

vii) Locate dealership: Dealerships that have the identified car in their database will be listed with information about them.

viii) Sort dealerships: Sort the dealership based on user's request. The dealerships will be sorted by alphabetical order or shortest distance.

ix) Provide feedback: Feedback will be provided by the user and sent to the developer.

3 Analysis Class Diagram



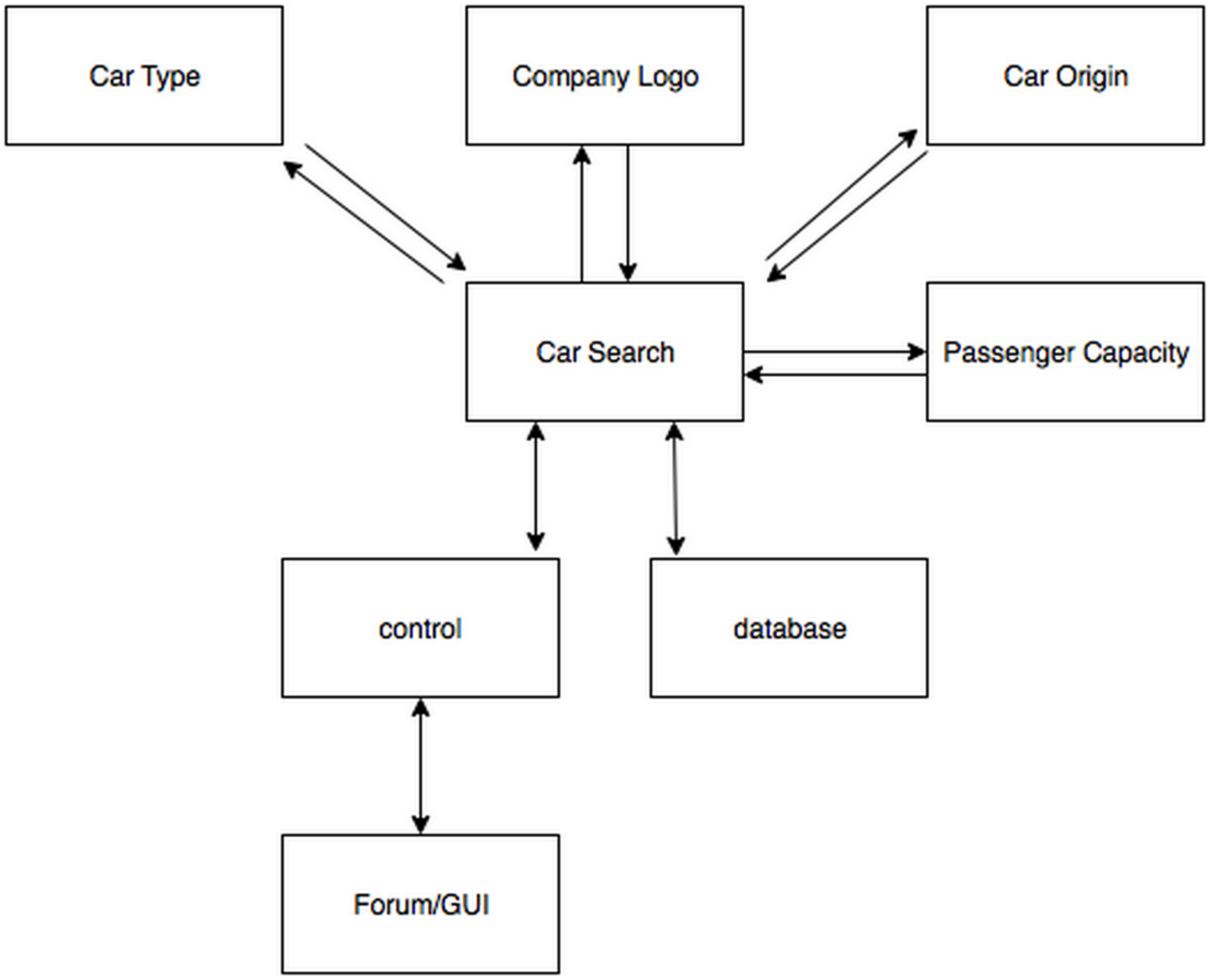
4 Architectural Design

4.1 System Architecture

a) The system is based on a blackboard architecture. There are four separate experts who can provide information independently using their expertise. Each expert identifies a different car property. A car search uses the information provided by the experts to search the car database, finding cars which have the identified properties.

This architecture structure works well for this system because it is a knowledge based system. Each expert can provide information which is then used to make a decision. Experts can also be added or removed very easily which gives the system flexibility. The experts are independent of one another, giving the system low coupling. An individual expert has one property which it will identify, giving high cohesion.

b) Structural architecture diagram of the system:



4.2 Subsystems

a) **Blackboard Subsystems**

Car Search:

This subsystem uses car properties provided by the experts to find car models in the database which have the provided properties.

b) **Knowledge Source Subsystems**

Car Type:

An expert which identifies the type of car (Sedan, SUV, Minivan, etc).

Company Logo:

An expert which identifies the company that made the car based on their logo.

Car Origin:

An expert which identifies the origin of the car (North American, European, etc).

Passenger Capacity:

An expert which identifies the number of passengers the car can hold.

Database:

A database containing car models and their properties. The database can be searched to find models which fit certain criteria.

c) **Controller Subsystem**

Control:

This subsystem can initiate a car search and supervise the overall identification process.

5 Class Responsibility Collaboration (CRC) Cards

Class Name: CarDB	
Responsibility:	**Collaborators:**
Contain a listing of all car models and their attributes	-
Allow insertion and deletion of entries	-
Allow editing of entries	-
Provide information to CarSearchController	CarSearchController

Class Name: FeedbackStorage	
Responsibility:	**Collaborators:**
Contain a list of all feedback forms completed by users with anonymity, stored in a file	-
Receive feedback from feedback form for storage	FeedbackForm

Class Name: FeedbackForm	
Responsibility:	**Collaborators:**
Allow user to enter feedback about the application	-

Class Name: CarSearchController	
Responsibility:	**Collaborators:**
Contains algorithm to identify a car given some attributes	-
Extract information from the SearchForm and compile it into a search query	SearchForm
Send result of search to SearchResult for display and verification	SearchResult
Query car database and experts as part of search algorithm to identify the car	CarDB, Expert
Control experts to be used in identification based on attributes given	ExpertPicker

Class Name: SearchResult	
Responsibility:	**Collaborators:**
Receive search result and send it to the forum to be displayed	Forum,CarSearchController
Once a car identification is confirmed, result sent to search history	SearchHistory
Send result for verification before sending to search history	ResultVerifier

Class Name: ExpertPicker	
Responsibility:	**Collaborators:**
Control which experts will be used to identify the car based on attributes that are inputted	Expert
Set experts to ”passive” or ”active” for identification process	Expert

Class Name: HelpPage	
Responsibility:	**Collaborators:**
Provide information about the application, and how to use it	-

Class Name: Forum	
Responsibility:	**Collaborators:**
Central hub of application to allow navigation to various pages	SearchForm, SearchHistory, HelpPage, FeedbackForm
Display result of car identification	SearchResult

Class Name: SearchForm	
Responsibility:	**Collaborators:**
Allow user to input characteristics of the car they want to identify	-
Send inputted attributes to car identification algorithm	CarSearchController

Class Name: SearchHistory	
Responsibility:	**Collaborators:**
Store previous five confirmed identification results	-
When a new result enters the history, pushes out fifth most recent confirmed identification	-

Class Name: DealershipLocator	
Responsibility:	**Collaborators:**
Interface with Google Maps API to locate dealerships that sell a specific car from the search history	SearchHistory

Class Name: SecurityController	
Responsibility:	**Collaborators:**
Contains encryption and decryption mechanisms for transmitted messages	-
Decrypt search result once it arrives at the forum	Forum
Encrypt the search result before sending it to the forum	SearchResult

Class Name: ResultVerifier	
Responsibility:	**Collaborators:**
Provide the user with the ability to confirm or deny the identified car result	-
Restart car identification if identified car is incorrect	CarSearchController
Restart search form if the identified car is incorrect three times	CarSearchController, SearchForm

Class Name: Expert	
Responsibility:	**Collaborators:**
Know potential car identifications given certain attribute combinations in respective domain of expertise	-
Provide expertise to identify a car given some attributes of its domain	CarSearchController
Provide functionality to be set as ”active” or ”passive” when trying to identify a car	ExpertPicker

A Division of Labour

Team Member:	**Sections Completed:**
Abhijit	Section 1, 4
Cole	Section 3, 4, Reviewed and Reworked Business Events
David	Section 3, 5, Reviewed and Reworked Business Events
Sharon	Section 2, 3, Reviewed and Reworked Business Events, Revised Section 1
Yash	Section 3, 5, Reviewed and Reworked Business Events, Revised Section 1
Yuchen	Reviewed Section 4

Table 1: Division of Labour