Software Design Description

Shuttle Tracking and Ride Request Service (STARRS)

Prepared by Will Gross, Zhuofan Zhang, Zilin Chen
CS 397 Group 3
5 Nov 2017

Table of Contents

Table of Contents *2*
Revision History *6*
1. Introduction **7**
1.1 Purpose 7
1.2 Scope 7
1.3 Contexts 7
1.4 Summary 7
2. Stakeholders and Design Concerns **7**
2.1 Stakeholders 7
2.2 Stakeholder Design Concerns 7
2.3 Expanded Design Concerns 7
3. Design Views **8**
3.1 Design View 1 8
3.1.1 Design Viewpoint 1 8
3.1.1.1 Rationale 9
3.1.2 Design Elements 10
3.1.2.1 Display 10
3.1.2.1.1 Design Attributes 10
3.1.2.1.1.1 Name Attributes 10
3.1.2.1.1.2 Type Attribute 10
3.1.2.1.1.3 Purpose Attribute 10
3.1.2.1.1.4 Author Attribute 10
3.1.2.1.2 Design Relationships 10
3.1.2.1.3 Design Constraints 11
3.1.2.2 System Manager 11
3.1.2.2.1 Design Attributes 11
3.1.2.2.1.1 Name Attribute 11
3.1.2.2.1.2 Type Attribute 11
3.1.2.2.1.3 Purpose Attribute 11
3.1.2.2.2 Design Relationships 11
3.1.2.3 Queue and Request System 11
3.1.2.3.1 Design Attributes 12
3.1.2.3.1.1 Name Attribute 12
3.1.2.3.1.2 Type Attribute 12
3.1.2.3.1.3 Purpose Attribute 12
3.1.2.3.2 Design Relationships 12
3.1.2.3.3 Design Constraints 12
3.1.2.4 Listener 12
3.1.2.4.1 Design Attributes 12
3.1.2.4.1.1 Name Attribute 12
3.1.2.4.1.2 Type Attribute 12

3.1.2.4.1.3 Purpose Attribute ……12
3.1.2.4.2 Design Relationships……13
3.1.2.4.3 Design Constraints……13
3.1.2.5 Users …… ……13
3.1.2.5.1 Design Attributes……13
3.1.2.5.1.1 Name Attribute……13
3.1.2.5.1.2 Type Attribute ……13
3.1.2.5.1.3 Purpose Attribute ……13
3.1.2.5.2 Design Relationships……13
3.1.2.5.3 Design Constraints……13
3.1.2.6 Message Component……... ……14
3.1.2.6.1 Design Attributes……..14
3.1.2.6.1.1 Name Attribute……14
3.1.2.6.1.2 Type Attribute ……14
3.1.2.6.1.3 Purpose Attribute ……14
3.1.2.6.2 Design Relationships……14
3.1.2.6.3 Design Constraints……14
3.1.2.7 Database……..……14
3.1.2.7.1 Design Attributes……..14
3.1.2.7.1.1 Name Attribute……14
3.1.2.7.1.2 Type Attribute ……14
3.1.2.7.1.3 Purpose Attribute ……14
3.1.2.7.2 Design Relationships……14
3.1.2.7.3 Design Constraints……15
3.2 Design View 2…….....15
3.2.1 Design Viewpoint 1…… 15
3.2.1.1 Rationale……..15
3.2.2 Design Elements ……...16
3.2.2.1 Design Entities……...16
3.2.2.2Design Attributes ……16
3.2.2.2.1 Name Attributes ……..16
3.2.2.2.2 Type Attribute ……..16
3.2.2.2.3 Purpose Attribute……17
3.2.2.2.4 Author Attribute ……17
3.2.2.3 Design Relationships ……..17
3.2.2.4 Design Constraints ……17
3.3 Design View 3.....17
3.3.1 Design Viewpoint …… 17
3.3.1.1 Rationale……18
3.3.2 Design Elements ……..19
3.3.2.1 General Web Page……19
3.3.2.1.1 Design Attributes ……19
3.3.2.1.1.1 Name Attributes ……19

3.3.2.1.1.2 Type Attribute ……..19
3.3.2.1.1.3 Purpose Attribute…….19
3.3.2.1.2 Design Relationships ……19
3.3.2.1.3 Design Constraints ……19
3.3.2.2 Drivers Web Page…….19
3.3.2.2.1 Design Attributes ……19
3.3.2.2.1.1 Name Attributes ……..19
3.3.2.2.1.2 Type Attribute ……..19
3.3.2.2.1.3 Purpose Attribute…….19
3.3.2.2.2 Design Relationships ……...…….20
3.3.2.2.3 Design Constraints ……20
3.3.2.3 Security Web Page…….20
3.3.2.3.1 Design Attributes ……20
3.3.2.3.1.1 Name Attributes ……..20
3.3.2.3.1.2 Type Attribute ……..20
3.3.2.3.1.3 Purpose Attribute…….20
3.3.2.3.2 Design Relationships ……20
3.3.2.3.3 Design Constraints ……20
3.4 Design View 4.....20
3.4.1 Design Viewpoint …… 21
3.4.1.1 Rationale……..21
3.4.2 Design Elements ……...22
3.4.2.1 Design Entities…….22
3.4.2.2 Design Attributes ……22
3.4.2.2.1 Name Attributes ……..22
3.4.2.2.2 Type Attribute ……..22
3.4.2.2.3 Purpose Attribute……22
3.4.2.3 Design Relationships ……..…….22
3.4.2.4 Design Constraints ……23
3.5 Scenario View 5.....23
3.5.1 Design Viewpoint 5 …… 23
3.5.1.1 Rationale……..23
3.5.2 Design Elements ……...24
3.5.2.1 Shuttle Schedule……..…….24
3.5.2.1.1 Design Attributes ……24
3.5.2.1.1.1 Name Attributes ……..24
3.5.2.1.1.2 Type Attribute ……..24
3.5.2.1.1.3 Purpose Attribute…….24
3.5.2.1.2 Design Relationships ……24
3.5.2.2 Jitney Schedule…….24
3.5.2.2.1 Design Attributes ……24
3.5.2.2.1.1 Name Attributes ……..24
3.5.2.2.1.2 Type Attribute ……..25

3.5.2.2.1.3 Purpose Attribute……25
3.5.2.2.2 Design Relationships ……25
3.5.2.3 Shuttle Map……..…….25
3.5.2.3.1 Design Attributes ……25
3.5.2.3.1.1 Name Attributes ……..25
3.5.2.3.1.2 Type Attribute ……..25
3.5.2.3.1.3 Purpose Attribute……25
3.5.2.3.2 Design Relationships ……25
3.5.2.4 Jitney Map……..…….25
3.5.2.4.1 Design Attributes ……25
3.5.2.4.1.1 Name Attributes ……..25
3.5.2.4.1.2 Type Attribute ……..26
3.5.2.4.1.3 Purpose Attribute……26
3.5.2.4.2 Design Relationships ……26
3.5.2.4.3 Design Constraints……26
3.5.2.5 Jitney Request……26
3.5.2.5.1 Design Attributes ……26
3.5.2.5.1.1 Name Attributes ……..26
3.5.2.5.1.2 Type Attribute ……..26
3.5.2.5.1.3 Purpose Attribute……26
3.5.2.5.2 Design Relationships ……26
3.5.2.5.3 Design Constraints……26
3.5.2.6 Jitney Request Queue……27
3.5.2.6.1 Design Attributes ……27
3.5.2.6.1.1 Name Attributes ……..27
3.5.2.6.1.2 Type Attribute ……..27
3.5.2.6.1.3 Purpose Attribute……27
3.5.2.6.2 Design Relationships ……27
3.5.2.6.3 Design Constraints……27
3.5.2.7 Jitney Request History……27
3.5.2.7.1 Design Attributes ……27
3.5.2.7.1.1 Name Attributes ……..27
3.5.2.7.1.2 Type Attribute ……..27
3.5.2.7.1.3 Purpose Attribute……27
3.5.2.7.2 Design Relationships ……28
3.5.2.7.3 Design Constraints……28
3.5.2.8 Messages……..…….28
3.5.2.8.1 Design Attributes ……28
3.5.2.8.1.1 Name Attributes ……..28
3.5.2.8.1.2 Type Attribute ……..28
3.5.2.8.1.3 Purpose Attribute……28
3.5.2.8.2 Design Relationships ……28
3.5.2.8.3 Design Constraints……28

3.6 Design View 6.....28
3.6.1 Design Viewpoint …… 28
3.6.1.1 Rationale…….29
3.6.2 Design Elements ……...29
3.6.2.1 Design Entities…….29
3.6.2.2 Design Attributes ……29
3.6.2.2.1 Name Attributes ……..29
3.6.2.2.2 Type Attribute ……..29
3.6.2.2.3 Purpose Attribute…….29
3.6.2.3 Design Relationships …….29
3.6.2.4 Design Constraints ……...29

Revision History

Name	Date	Reason For Changes	Version
Zhuofan Zhang	11/05/17	Creation of Document	1.0
Zhoufan Zhang Zilin Chen Will Gross	12/12/17	Revisions to address comments	1.1

List of Figures

Class diagram

Fig. 1 Class diagram of the system. 10

Sequence diagrams

Fig. 2 Sequence diagram for the use case of tracking Jitney location. 16
Fig. 3 Sequence diagram for use case of creating / cancelling a pickup request. 17
Fig. 4 Sequence diagram for use case of using the notification system. 17

Activity diagrams

Fig. 5 High level activity diagram. 19
Fig. 6 Activity diagram for use case of tracking Jitney location. 20
Fig. 7 Activity diagram for the use case of creating / cancelling a pickup request. 21
Fig. 8 Activity diagram for the use case of using the notification system. 22

Architecture diagram

Fig. 9 Architecture diagram for the system deployment. 25

1 Introduction

1.1 Purpose

This software design description (SDD) is to lay out the structure and architecture of the envisioned STARRS system to guide development. The charts and graphs in this document will be used by developers to communicate and understand the intent and purpose of the different components, as well as to inform their decisions while developing to ensure that all of the development is done in line with other components and toward the goal of a unified system.

1.2 Scope

This document details the implementation of low level functionality and requirements. The main functionality which we are implementing is the ability to project the schedule of the shuttle onto an animated map, and provide links to more information regarding transportation services. This document also covers functionality that will be implemented in later versions, which includes the handling of requests for the jitney, GPS data from the shuttle and jitney, and data management for usage information collected by the STARRS.

1.3 Context

This document was created as a part of a documentation suite of documentation which also includes a project management plan (PMP), a software requirement specification (SRS), and a testing plan. This document directly depends on information from the SRS and PMP, and is used by the developers in building the system.

1.4 Summary

This document will outline the design architecture of STARRS and provide a guideline for the development of the system.

2 Stakeholders and Design concerns

2.1 Stakeholders

The main stakeholder in this project is Colby Security. The secondary stakeholder is the student population and user pool.

2.2 Stakeholder Design Concerns

Security is mainly concerned with having a functional system that reduces the work for dispatchers, drivers, and those responsible for data entry. The users is mainly concerned with fast, effective, and easy service.

2.3 Expanded Design Concerns

To address these design concerns, we plan to auto-log and record data via the database back end, as well as automate the request process. We also plan to display as much relevant information as possible to users so that they can make better informed transportation decisions.

3 Design Views

3.1 Design View 1

This is the logical view.

3.1.1 Design Viewpoint 1

Name: Logical viewpoint.

Concerns: The viewpoint should consider all the components in the system, and create classes to implement their functions. In particular, it should adapt proper design patterns to ensure modularity, abstraction, loose coupling, and high cohesion.

Analytical methods: One possible measurement of how good the system is, by means of the desired properties such as cohesion, is to consider alternate approaches and compare both.

Overview:

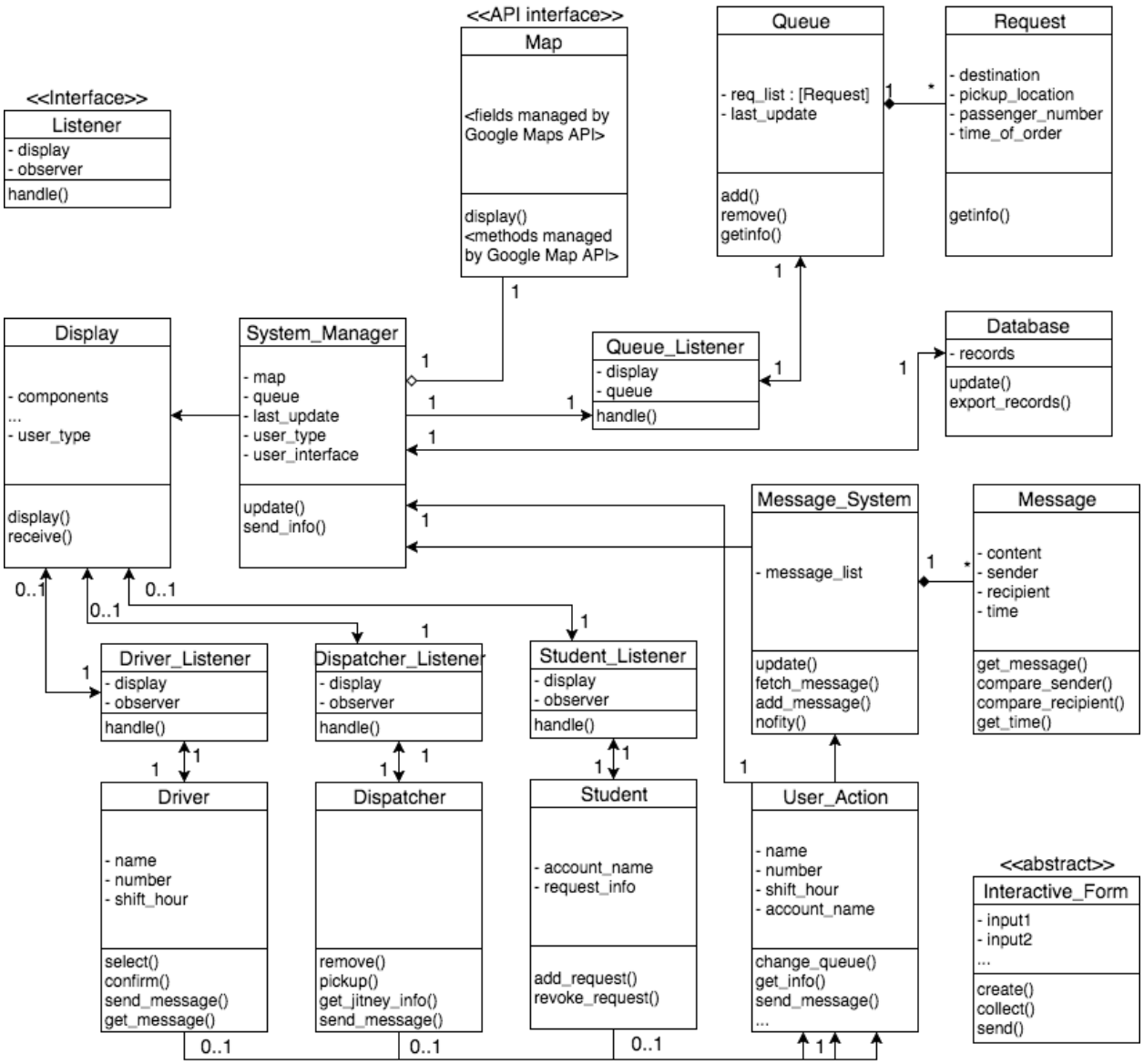

Fig.1 Class diagram of the system.

*Listener is an interface, so only its implemented interfaces (*_Listener) are connected.

**Interactive_Form is a class used for the user to fill out necessary information, and can turn up in a variety of scenarios with different attributes. To present a simplified view, we mention its existence but ignore all different versions and connections about it.

3.1.1.1 Rationale

The system is mainly a bunch of data (such as locations and requests) with several interaction options, so a natural design would be to set a center class (System_Manager) that manages all interactions and reflects those interactions by sending relevant information to the front end -- the GUI. There is also a back end database for record keeping.

We did not identify a lot of main actors in the system: Namely, we only had the user (student / driver / dispatcher) as the only source of input, and all three give disparate commands. The observer model was applied to suit for these actors: Any of their actions through the GUI display would be received by a Listener class, and the listener would inform the System_Manager to act accordingly by letting the corresponding class act through the general User_Action class.

Other than that, the principle of modularity led to the creation of a Queue class consisting of Request classes, a User_Action class that acts as a general user and connects with System_Manager (while being the only class that can convey user inputs), and potentially an abstract class of Interactive_Form that can be implemented as an interactive form used in the interface when, for example, the user orders a new pickup request.

The design pattern of Adapter is also implemented: Only changing the Display class is needed for different platforms.

3.1.2 Design Elements

3.1.2.1 Display

Display is a system that incorporates the functionalities of a GUI, presenting information graphically while also translating user commands. It should consist of more detailed parts, whether they are custom-implemented or come from packages.

3.1.2.1.1 Design attributes

3.1.2.1.1.1 Name attribute

Element name: Display. For potential smaller parts in this entity, their names would start with the prefix “Display_”.

3.1.2.1.1.2 Type attribute

The element is a subsystem.

3.1.2.1.1.3 Purpose attribute

The element enables user interface, graphical representation, and data collection.

3.1.2.1.1.4 Author attribute

Part of the element would source from already-existing packages, while the rest would be customized in this project.

3.1.2.1.2 Design relationships

See the viewpoint: Because only the users can manipulate through the interface, the Display is only connected to the Listeners for the three different actor classes for any

actions to be performed. It is also linked with the System_Manager class to reflect any changes in the system.

3.1.2.1.3 Design constraints

The platform that the interface works on imposes the constraint on Display.

3.1.2.2 System_Manager

3.1.2.2.1 Design attributes

3.1.2.2.1.1 Name attribute

Element name: System_Manager.

3.1.2.2.1.2 Type attribute

Element type: Class.

3.1.2.2.1.3 Purpose attribute

The element is the central processing unit of the system. Its tasks include sending instructions to update Display and Database, receiving instructions from the User_Action class, and updating the request queue.

3.1.2.2.2 Design relationships

The element has the following interactions with other components:

Display -- The element gathers information to sent to Display when it needs to refresh/update its information

Queue_Listener -- The element updates the queue of the system through calling the handle method in Queue_Listener.

User_Action -- The element receives instructions from User_Action class, and then performs corresponding actions such as removing a request from the queue (through Queue_Listener), or sending relevant data to Display for a current update.

Database -- The element sends log entries to the Database, including new request information and such. This shall be a task done in a certain frequency.

3.1.2.3 Queue and Request system

The element is made up of two subparts: Queue and Request. They are used to store all the current pickup requests.

3.1.2.3.1 Design attributes

3.1.2.3.1.1 Name attribute

Element name: Queue and Request system. For the two subcomponents, call them "Queue" and "Request".

3.1.2.3.1.2 Type attribute

The element can be treated as an object as a whole.
Queue and Request are classes.

3.1.2.3.1.3 Purpose attribute

The element stores all the pickup requests, unprocessed or currently being processed. Each pickup request is stored in a Request class/object, while the Queue class is consisted of Request classes.

3.1.2.3.2 Design relationships

Queue stores a list of Request objects (array or linked list). An outside source such as the System_Manager can call methods that can create a new Request / remove a current Request / change the status of one Request through a Queue_Listener.

3.1.2.3.3 Design constraints

Based on the scale of the application, we do not have to consider the runtime of the queue structure much.

3.1.2.4 Listener

Listener is an interface class that connects an observer and a controller. Each different observer would have its own Listener and its own handle method.

3.1.2.4.1 Design attributes

3.1.2.4.1.1 Name attribute

Element name: Listener. For specifically implemented Listeners for other classes, their names would be "<classname>_Listener".

3.1.2.4.1.2 Type attribute

The element is an interface class.

3.1.2.4.1.3 Purpose attribute

The element connects an observer and another entity that sends controls. This class makes the design follow the observer pattern, and to loosen the coupling between modules/classes.

3.1.2.4.2 Design relationships

The class mainly serves as a connector between the GUI (or the managing class) and the functional classes.

3.1.2.4.3 Design constraints

To use this interface, all the functional classes that are connected by it must have at least one handle method, and the methods should follow a similar way of taking in arguments.

3.1.2.5 Users

The Users part is a component that consists of the Listeners and the user classes for each type of user.

3.1.2.5.1 Design attributes

3.1.2.5.1.1 Name attribute

Element name: Users.

For the different types of users that this element contains: Driver, Dispatcher, and Student. This element also contains User_Action.

3.1.2.5.1.2 Type attribute

The element is a component consisting of multiple classes.

3.1.2.5.1.3 Purpose attribute

The user classes in the element receives instructions interpreted by Display, and calls the corresponding methods to translate the instruction into actual action to the queue or the GUI. The User_Action class is used to generalize the actions performed by any of the three user class, and transmit the generalized action to the System_Manager.

3.1.2.5.2 Design relationships

Each user class in this element connects to a Listener. When the Listener receives the signal, it calls the handle methods in the user class that exists. The user class would then notify User_Action class on which action to take, where User_Action finally calls the corresponding method in system_Manager.

3.1.2.5.3 Design constraints

Because there can only be one user at the interface at one time on each terminal of the final system, there should always be exactly one of the user classes connecting to User_Action, while the other two do not exist.

3.1.2.6 Message component

The Message component part is a component that consists of the Message_System and the Message classes. It follows a similar structure with the Queue and Request System.

3.1.2.6.1 Design attributes

3.1.2.6.1.1 Name attribute

Element name: Message component.
It has two subparts: Message_System and Message.

3.1.2.6.1.2 Type attribute

The element is a component consisting of multiple classes.
Message_System and Message are both classes.

3.1.2.6.1.3 Purpose attribute

The element stores the messages sent between the dispatcher and the drivers. It also serves for notification and message retrieving.

3.1.2.6.2 Design relationships

The element is controlled by User_Action where it sends the message, along with all the relevant information, to the element. When prompted, the element can call methods in System_Manager and let it show new message notifications, or show history messages to the user via Display.

3.1.2.6.3 Design constraints

The class needs to store all messages, so it should have access to enough storage space.

3.1.2.7 Database

3.1.2.7.1 Design attributes

3.1.2.7.1.1 Name attribute

Element name: Database.

3.1.2.7.1.2 Type attribute

The element is a subsystem, consisting a data store and some methods.

3.1.2.7.1.3 Purpose attribute

The element stores all logging information and messages of the system.

3.1.2.7.2 Design relationships

The element is connected with System_Manager, so that it can send logs directly to the element in order to store them.

3.1.2.7.3 Design constraints

The element needs to be able to perform saving (and possibly extracting) by itself.

3.2 Design View 2

The interaction view focus on the message flow between the components.

3.2.1 Design Viewpoint

Name: Interaction viewpoint.

Concerns: The interactions between elements that go on inside the main processes that the system would have to undertake.

Analytical methods: The design can be evaluated by carefully examining the process of each sequence diagram, and see if any ambiguity would have some chance to occur.

3.2.1.1 Rationale

The event sequence in this viewpoint is mainly determined by the use cases in SRS and the class / object in the Logical viewpoint. The nature of the project decides that most of the cases would be real-time interactions, following a chronological order.

One example scenario on viewing current Jitney map:

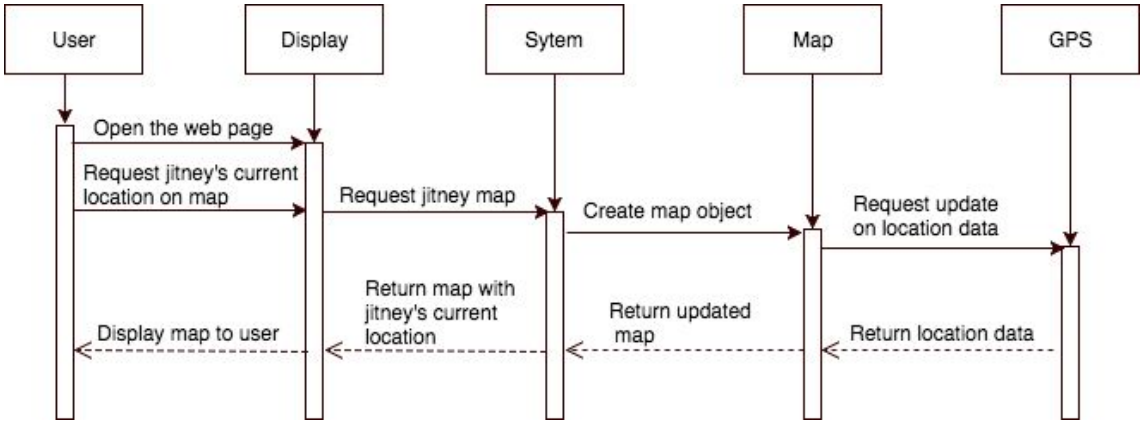

Fig. 2 Sequence diagram for the use case of tracking Jitney location.

One full example scenario on the student user creating and then cancelling a pickup request:

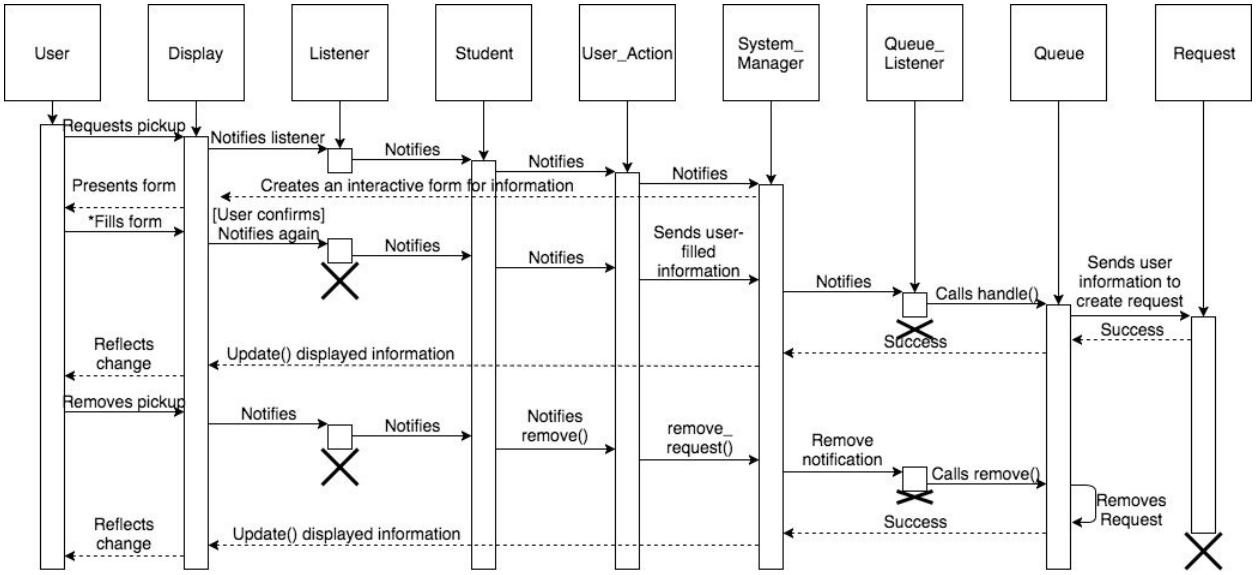

Fig. 3 Sequence diagram for use case of creating / cancelling a pickup request.

One example scenario in message sending, receiving, and updating:

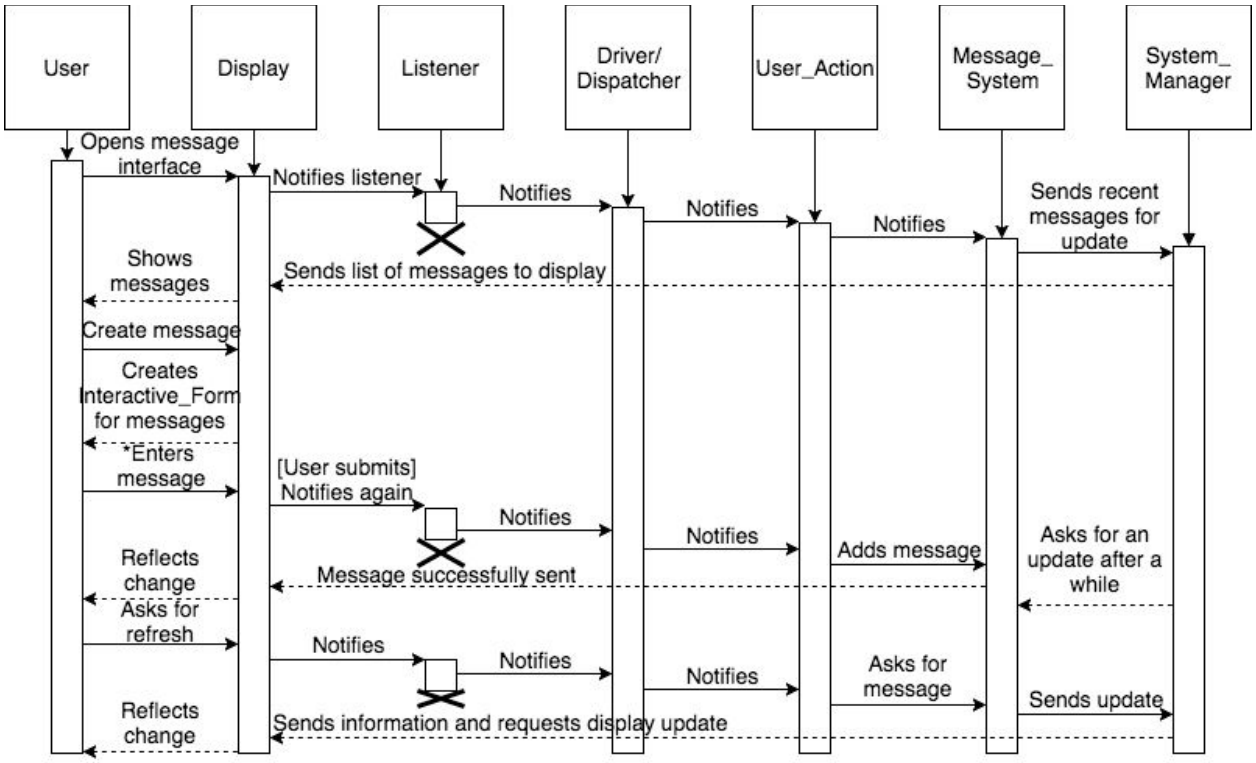

Fig. 4 Sequence diagram for use case of using the notification system.

Other scenarios would follow a similar pattern, so we do not go into details here.

3.2.2 Design Elements

Each class mentioned above in the Logical viewpoint would be a design element here, namely, the objects on the upper row. Furthermore, each object has its own lifeline, and they are connected by messages (mostly sent by the methods inherent to classes) to and fro triggered by events.

3.2.2.1 Design entities

The design entities can be categorized into classes, methods, and lifelines, the last one of which not really relevant to topic -- Every class is likely to exist until the user stops the service.

3.2.2.2 Design attributes

3.2.2.2.1 Name attribute

The name of the element is the name of the class or method.

3.2.2.2.2 Type attribute

The type of the element is either the type of the class / object itself, or a method.

3.2.2.2.3 Purpose attribute

The purpose of classes can be found in 3.1.2.1.x.3 of the corresponding classes. The purpose of methods is self-explanatory in the diagram: Mainly for communicating with other classes by sending relevant arguments.

3.2.2.3 Design relationships

Usually a design starts with a user action on the GUI with the Display, and ends with a show of results from the Display to the user, with changes in the system as a middle product. Generally, different scenarios have independent sequence diagrams, and thus have little relationship with each other.

3.2.2.4 Design constraints

One constraint imposed on the design is the loose coupling between classes in the design. For example, because Display can not send user action about pickup request directly to System_Manager, we have to let the message go through the user class and User_Action first. This creates a detour when the user first creates a pickup request: The faster method would be to let System_Manager directly handle this, and return an Interactive_Form for

the user to fill out. Thus the Logic viewpoint design serves as a constraint to the interaction viewpoint design.

3.3 Design View 3

The process view focus on the activity flow and the runtime behavior in the system.

3.3.1 Design Viewpoint

Name: Process viewpoint.

Concerns: The control flow for the main processes that the system would have to undertake.

Analytical methods: The design can be evaluated by comparing to similar, already-existing cases, and by collecting user feedback.

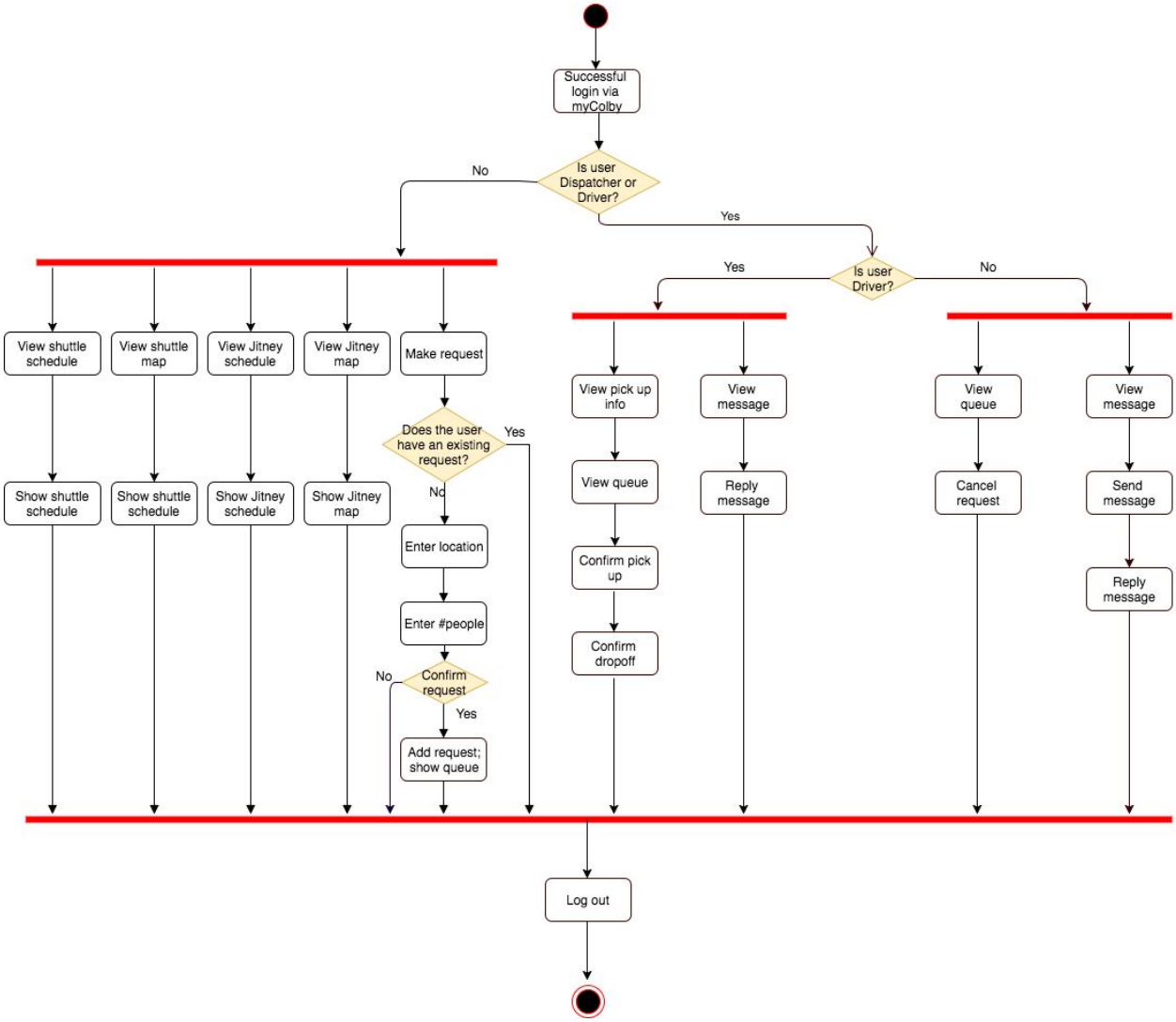

Fig. 5 High level activity diagram.

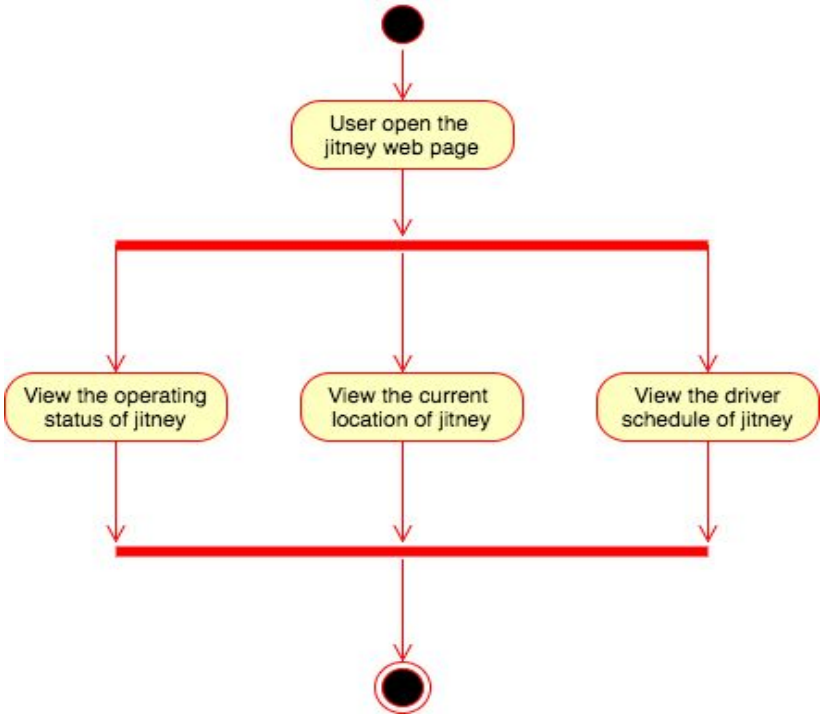

Fig. 6 Activity diagram for use case of tracking Jitney location.

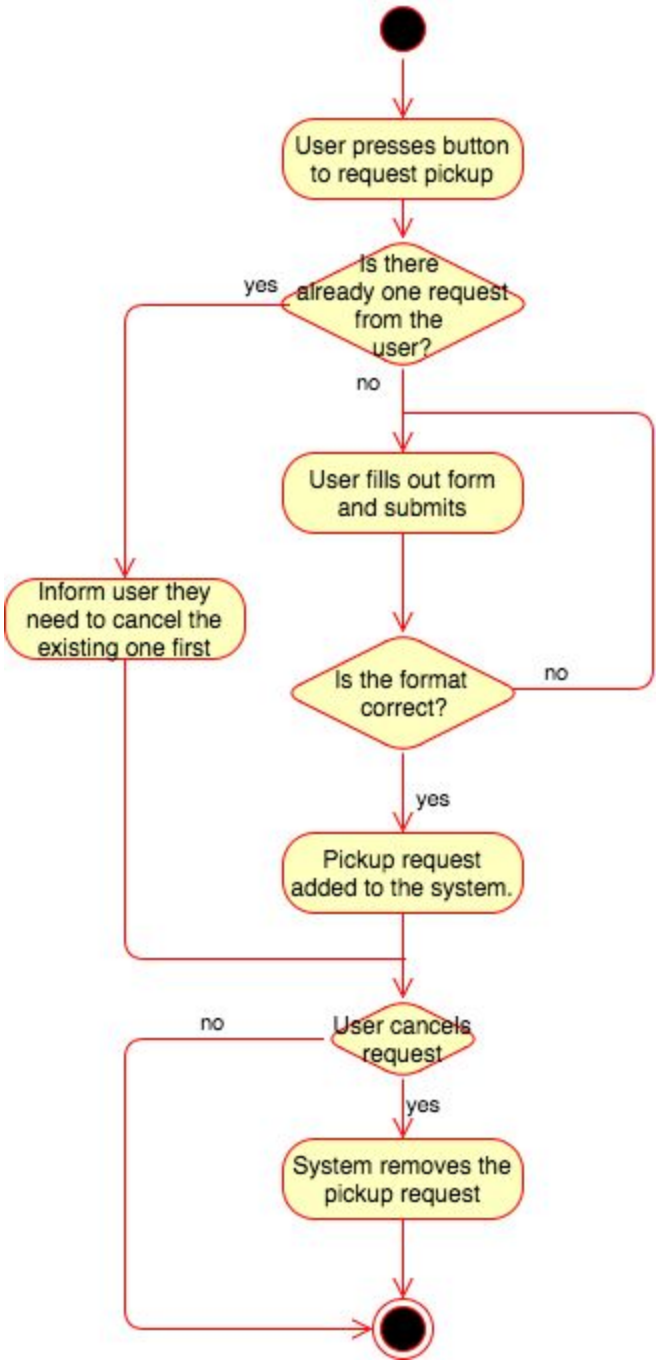

Fig. 7 Activity diagram for the use case of creating / cancelling a pickup request.

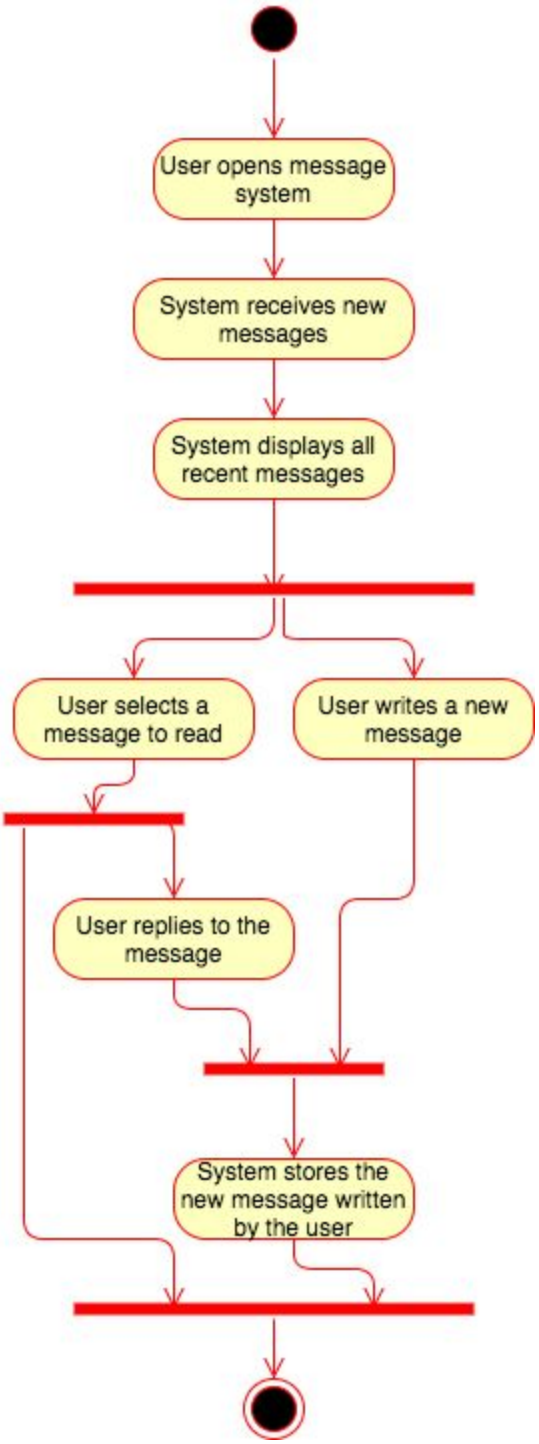

Fig. 8 Activity diagram for the use case of using the notification system.

3.3.1.1 Rationale

In this designed system, there are three types of users: common users, drivers, and the security. The common functionalities of the system should be available to all users. Such functionalities include viewing the maps and schedules for the shuttle and jitney and making or canceling jitney

pickup requests. Those services will be available to all users once they log in to the web page, regardless of the identities of the users.

There are some special functionalities that only drivers or security have access to. So we need to separate these two actors from the common users. At Colby, students and faculties have access to different resources, and we could take advantage of this. But drivers are also students, and we cannot create special privileges for these students. So one of the effective and easy way to solve this problem is to create two links for these two actors on the web page, and let them use special password or code to access the link. Then we can design those special functions in their pages correspondingly.

Once users close the pages, they automatically logout.

3.3.2 Design Elements

3.3.2.1 General web page

3.3.2.1.1 Design attributes

3.3.2.1.1.1 Name attribute

Element name: General Web

3.3.2.1.1.2 Type attribute

The element is a web page, which contains the general functionalities that available to all users.

3.3.2.1.1.3 Purpose attribute

The web enables the user to check the schedules and current locations of shuttle and jitney as well as making and canceling jitney pickup requests.

3.3.2.1.2 Design relationships

This is the general web page that is shown to users once they log in. The web will contain links to the driver's page and security's page.

3.3.2.1.3 Design constraints

The user can get access to the web only if the network is available. In this page, users can cancel requests if and only if those are their own requests.

3.3.2.2 Driver's web page

3.3.2.2.1 Design attributes

3.3.2.2.1.1 Name attribute

Element name: Driver Web

3.3.2.2.1.2 Type attribute

The element is a web page and a subsystem, which contains the functionalities that are only available to jitney drivers.

3.3.2.2.1.3 Purpose attribute

The web enables the driver to accept pickup orders, confirm pickup, complete journey, and receive from and reply messages to the security.

3.3.2.2.2 Design relationships

This web page links directly to the main page and requires special code or login information to get access to.

3.3.2.2.3 Design constraints

The driver can get access to the web only if the network is available. A driver can only confirm a pickup request when there is one. A driver cannot make pickup request, but he can go to the main page to make one if he wants to. A driver can only view the waiting queue, but he cannot modify it, meaning that a driver cannot cancel any requests. A driver cannot send messages directly to the security.

3.3.2.3 Security's web page

3.3.2.3.1 Design attributes

3.3.2.3.1.1 Name attribute

Element name: Security Web

3.3.2.3.1.2 Type attribute

The element is a web page and a subsystem, which contains the functionalities that are only available to the security.

3.3.2.3.1.3 Purpose attribute

The web enables the security to view the waiting queue and cancel request if necessary. The security can send messages to the driver.

3.3.2.3.2 Design relationships

This web page links directly to the main page and requires special code or login information to get access to.

3.3.2.3.3 Design constraints

The security can get access to the web only if the network is available. Security can only cancel a request when there is one, and the request can be anyone's request.

3.4 Design View 4

This is the physical/deployment view.

3.4.1 Design Viewpoint 4

Viewpoint name: Physical viewpoint.

Design concerns: How the product should be deployed in the actual physical world. The design follows a client-server architecture style, where the server stores all request and location data.

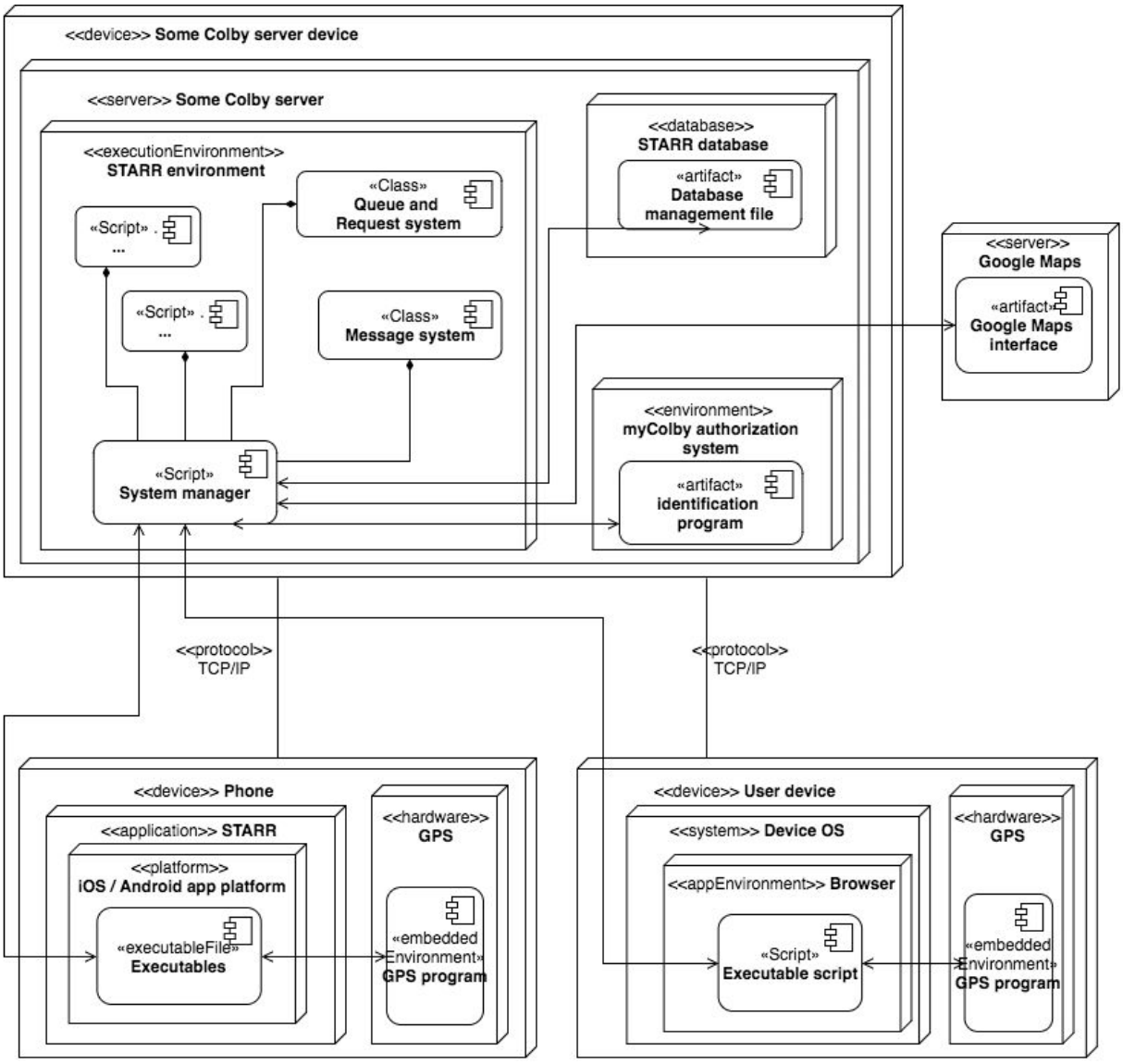

Fig. 9 Architecture diagram for the system deployment.

3.4.1.1 Rationale

The product would have to: 1) use a database to store pickup requests; 2) use myColby system to enable login and identity recognition; 3) connect to Google Maps API for required feature; 4) be able to reside in a webpage or a phone application to receive user instructions and send GPS data; 5) be able to extract, analyze, and send data from a server. With those specifications, the product would need to at least occupy a server and a database, connect to the myColby system server, connect to the Google Maps server, and contain some executable scripts based on a webpage, or be executable on a mobile device.

3.4.2 Design Elements

3.4.2.1 Design entities

Several design entities are included in the design: The servers, executables and scripts, the devices, etc. Because it is still unclear how these needed entities would be accessed and used, we gave generic names and descriptions to those entities, as shown in the picture.

3.4.2.2 Design attributes

3.4.2.2.1 Name attribute

The names can be seen in the figure above.

3.4.2.2.2 Type attribute

The type of each entity can be seen in the figure above.

3.4.2.2.3 Purpose attribute

The files in the STARR Environment are used to execute most parts of the project, including receiving data and instructions from the outside devices, processing the data in the queue and message system, and sending relevant data to outside devices, as well as generating and sending log information to the database.

The STARR database is used to store logging information and request history, so that the Security Office can access and analyze when needed. It also stores the driver's shift hours and other useful information that the final product would be able to inform the users of.

The Google Maps Interface enables the user to select locations more conveniently, or see the location of the vehicle visually.

The identification program checks the user's identity, and tells the system manager to give the user corresponding priority and authority.

The Executables / Executable script in the devices create the GUIs that interact with the user, and sends location data from the GPS hardware built-in the device. They are also responsible for sending information about user's inputs to enable the system manager to respond.

3.4.2.3 Design relationships

The design relationship can be seen visually from the figure. All connected entities know each other, and obtain data from each other.

3.4.2.4 Design constraints

Most constraints are imposed by the physical system that each entity resides in. For example, the database has constraints about the amount of allowed data usage by the storage limit, and the data transmission speed by the hardware and the connection; the Google Maps interface has constraints from the Google map server, and net connection; the Executables / executable script on the devices are constrained by the device's running speed, GPS precision and accuracy, and internet connection speed.

3.5 Scenario View 5

This is the scenarios view.

3.5.1 Design Viewpoint 5

Viewpoint name: Scenarios viewpoint.
Design concerns: Some functional requirements of this project and the primary actors.
Analytical methods: The scenarios are considered as multiple use cases. Some are already derived in the SRS document (Fig. A.2, A.3, A.4); there is also a high-level use case. (Fig. A.1).

3.5.1.1 Rationale

There are mainly three actors in this use case diagram: users, driver, and security. One important thing to notice is that driver and security are special kinds of users, so they can do all that users can do. As we have mentioned in the process view design that the general web page will be available to all users once they have logged in, the functionalities that are available to users will show on the general web page. And those functionalities are the ones in the circles that are connected to the users.

Those functionalities in the circles that are connected to the driver and security are available at their own page. Security can modify or update the schedule for shuttle and jitney. They can also make new requests or cancel existing requests at its own page. Notice that security can cancel any request in the queue, while users can only cancel their own requests. Security can also view the history records of the jitney and send messages to the driver.

The driver basically can view requests, accept requests, confirm requests, and view queue. The driver cannot cancel any existing request. Once the driver has sent the passengers to the designated location, the driver can click the “complete journey” button and the request will be destroyed and disappear in the queue.

3.5.2 Design Elements

3.5.2.1 Shuttle Schedule

3.5.2.1.1 Design attributes

3.5.2.1.1.1 Name attribute

Element name: Shuttle Schedule

3.5.2.1.1.2 Type attribute

The element is an interface that shows the time and stop location of the shuttle bus.

3.5.2.1.1.3 Purpose attribute

The shuttle schedule enables the user to check the scheduled stop time and locations of the shuttle.

3.5.2.1.2 Design relationships

The schedule will be modified by security. Any updates made by security will appear immediately on the main page.

3.5.2.2 Jitney Schedule

3.5.2.2.1 Design attributes

3.5.2.2.1.1 Name attribute

Element name: Jitney Schedule

3.5.2.2.1.2 Type attribute

The element is an interface that shows the jitney driver’s name and time shift.

3.5.2.2.1.3 Purpose attribute

The jitney schedule allows the user to check who will drive the jitney and the time that jitney starts working.

3.5.2.2.2 Design relationships

The schedule will be modified by security. Any updates made by security will appear immediately on the main page.

3.5.2.3 Shuttle map

3.5.2.3.1 Design attributes

3.5.2.3.1.1 Name attribute

Element name: Shuttle map

3.5.2.3.1.2 Type attribute

The element is an interface that shows the current location of the shuttle bus.

3.5.2.3.1.3 Purpose attribute

In case the shuttle is slightly off the schedule, users can get a more accurate estimation of the arrival time of the shuttle. Additionally, users can get an immediate sense of where the shuttle is right now.

3.5.2.3.2 Design relationships

The map will be taken care of by Google map.

3.5.2.3.3 Design constraints

A GPS is required to be placed on the shuttle bus to send dynamic information about the shuttle's current location to the website. So to make the map work, the GPS should be usable and at a good condition, and the network should be available.

3.5.2.4 Jitney map

3.5.2.4.1 Design attributes

3.5.2.4.1.1 Name attribute

Element name: Jitney map

3.5.2.4.1.2 Type attribute

The element is an interface that shows the current location of the jitney.

3.5.2.4.1.3 Purpose attribute

User can know the current location of the jitney and get an estimation of the time that will take the jitney to arrive at a desired location.

3.5.2.4.2 Design relationships

The map will be taken care of by Google map.

3.5.2.4.3 Design constraints

A GPS is required to be placed on the jitney to send dynamic information about the its current location to the website. So to make the map work, the GPS should be usable and at a good condition, and the network should be available.

3.5.2.5 Jitney request

3.5.2.5.1 Design attributes

3.5.2.5.1.1 Name attribute

Element name: Jitney request

3.5.2.5.1.2 Type attribute

The element is an object that stores the specific information about the request, including the pickup location and the number of passenger.

3.5.2.5.1.3 Purpose attribute

Users can send pickup requests to the driver, and driver can view the detailed information about the requests and accept pickup requests. The request will be the direct communication between users and the driver.

3.5.2.5.2 Design relationships

Users can modify requests on the main page; security can modify requests on their page. Driver cannot modify request on the driver's page, but any updates about any requests will send to the driver's page immediately.

3.5.2.5.3 Design constraints

The request information should be of a particular format, or the information will be considered invalid and not accepted. There should be a limited number of requests that one user can make within a time period. Security can make as many requests as necessary.

3.5.2.6 Jitney request queue

3.5.2.6.1 Design attributes

3.5.2.6.1.1 Name attribute

Element name: Jitney request queue

3.5.2.6.1.2 Type attribute

The element is a list of request objects.

3.5.2.6.1.3 Purpose attribute

The system will record the number of requests been made and make a list of them. Users can view the queue to get a sense of when their requests will be processed, and driver can plan the most efficient trip according to the queue to make the most of his time. Once a trip is completed, the request will be removed from the queue. And if a request is canceled, it will also be removed from the queue.

3.5.2.6.2 Design relationships

Any requests that users and security make will be added to the queue.

3.5.2.6.3 Design constraints

A queue will have a maximum size. But the size will be reasonably large so that most of the time, the number of requests will not exceed the size of the queue.

3.5.2.7 Jitney request history

3.5.2.7.1 Design attributes

3.5.2.7.1.1 Name attribute

Element name: Request history

3.5.2.7.1.2 Type attribute

The element is a database that stores the jitney pickup record for a recent period of time.

3.5.2.7.1.3 Purpose attribute

Any detailed information of the requests will be stored as well as the pickup time, drop off time, and the driver's name. The security can view the record and get a sense of the popular hours and locations of the jitney service.

3.5.2.7.2 Design relationships

When a request is accepted by the driver, the information of the request will be stored along with the driver's name. When the driver confirms a pickup request and completes the request, the startup time and end time will also be stored along with the request information.

3.5.2.7.3 Design constraints

There are limited amount of space in the database, so data will be cleaned periodically. If the database has almost reached its maximum size before the next cleaning period, it will send a notification to remind the security to clear up some space.

3.5.2.8 Message

3.5.2.8.1 Design attributes

3.5.2.8.1.1 Name attribute

Element name: Message

3.5.2.8.1.2 Type attribute

The element is an interface that stores and shows some text information.

3.5.2.8.1.3 Purpose attribute

The message will be the direct communication between security and the driver.

3.5.2.8.2 Design relationships

Messages only exist on the driver's page and security's page. Security can send driver some messages that will immediately appear on the driver's page. The driver can view and reply to the security. And the replied messages will show on the security's page.

3.5.2.8.3 Design constraints

The data space for the messages will be limited, so messages will be cleaned periodically.

3.6 Design View 6

The Development View.

3.6.1 Design Viewpoint 6

Viewpoint name: Development view
Design Concerns: The development view depicts the programmatic layout of the system, and details the component interactions.

3.6.1.1 Rationale

STARRS is a system with many access points to a main database, often sharing many similarities between ports. Because of this, there are going to be many shared components, so planning to minimize redundant code becomes very important.

3.6.2 Design Elements

3.6.2.1 Design entities

The main design entities are javascript plugin entities that allow the webpage shells to interface with the database background.

3.6.2.2 Design attributes

3.6.2.2.1 Name attribute

The attribute names are made as simple and descriptive as possible.

3.6.2.2.2 Type attribute

The types are specified in the diagram.

3.6.2.2.3 Purpose attribute

The purpose of using modular elements as interface plugins is to keep the code as simple and concise as possible. This prevents brittle code and makes the end product more elegant.

3.6.2.3 Design relationships

For small differences between elements, classes will inherit attributes from a parent class. Beyond that, plugins will be used by the website shell, which will interface with the database.

3.6.2.4 Design constraints

The design constraints are mostly due to the workings of the language and trying to keep the code as clean as possible.