S3a - Technical Design Document

Washu

A mobile application by

TaxiPack Corp.

Contributors

Adewale Adekoya V00834552

Chris Kelly V00729307

Richard Lui V00221256

Trison Nguyen V00178742

Submitted March 18 2016

Module interactions

When the user opens Washu, Washu checks for any Washu data in phone. If there is no Washu data, Washu will prompt the user to create an AppProfile. The AppProfile has the user's first name, last name, e-mail, and phone number.

AppProfile Module Pseudo code:

```
While (all fields not filled)
      Ask for user's first name field
      Ask for user's last name field
      Ask for user's email field
      Ask for user's phone number field
      When user hits "Submit", check if any fields not filled
            Inform user all fields must be filled
Exit AppProfile
```

If AppProfile is present, User is prompted to choose one of three choices. The user can choose Customer, Washer, or change profile. Only a Business Manager has privileges to change a Customer to Washer.

AppStart Module Pseudo Code:

```
If user hits "Change Profile", go to AppProfile Module
If user hits "Customer", go to CustOrder Module
If user hits "Washer", go to WasherCheck Module
```

If the user chooses "Change Profile", the user will be prompt to the AppProfile module, allowing them to change information: First name, last name, email, and phone number.

Customer

If user chooses "Customer", the user is directed to the CustomerUI Module. The user can make an order or view previous orders made. If the user presses "Order", the user is directed to the CustOrder module. If the user presses "My previous orders", the user is directed to the ViewOrders module.

CustomerUI Module Psuedo Code:

```
If user hits "Order"
     Go to CustOrder Module
     Exit Module
If user hits "My previous orders"
     Go to PastOrders Module
     Exit Module
```

Customer: Order Module

The user is asked to select a wash package, car picture, and car ID (license plate). Car image can be a picture taken on the spot, or selected from the phone's photo library. Within the customer order (CustOrder) module, another module, called InRegion, checks if the user is within a Washu region.

CustOrder Module Pseudo Code:

```
While (fields not filled and not in Washu Region)
     Ask user to select a package
     Ask user for car image(s)
     Ask user for car's license plate
     Check if user is in Washu region with InRegion Module
     When user hits "Order", check if any fields not filled
          Inform user all fields must be filled
Go to PaymentType Module
Exit CustOrder
```

InRegion Module Pseudo Code:

```
If user is in a Washu Region, return true
Else return false
```

If the user proceeds to the PaymentType module, the user will be prompted to fill in their credit card information. PaymentType module checks if for a valid credit card information with the PaymentCheck Module upon the user pressing the "confirm" button. If PaymentCheck Module returns true, the order information will be sent to the WashServer Module. The WashServer Module will be discussed under the WashServer section. If PaymentCheck Module return false, the user will have to redo their credit card information.

PaymentType Module Pseudo Code:

```
While (fields not filled)
     Ask user to fill in credit card information
     When user hits "Confirm", check if any fields not filled
          Inform user all fields must be filled
If PaymentCheck Module returns true
     Send order to WashuServer Module
     Exit Module
Else
     Inform user payment did not process and to refill the
credit card information
     Restart PaymentType Module
```

PaymentCheck Module Pseudo Code:

```
If credit card is valid
     Return true
Else
     Return false
Exit PaymentCheck Module
```

Upon a successful order, the Receipt Module will send a receipt to the user's email address.

Receipt Module Pseudo Code:

```
 While (fields not filled)
     Get and fill in Credit card information used
     Get and fill in Washu Package ordered
     Get and fill in Date Of Order
     Get and fill in Cost of Wash Service Package
Email user completed receipt form
```

After a wash is complete, the customer is given the option to rate the car wash with the CustReview Module. The customer can only submit a review once per wash. The review will be sent to the ReviewDB module.

CustReview Module Pseudo Code:

```
While (rate out of 5 is false)
      User rates the wash out of 5
      User fills in a review of the wash
      When hits "Submit", check if rate out of 5 has not been
      filled
            Inform user to rate out of 5
Send review to ReviewDB Module
Go to CustomerUI Module
Exit Module
```

Customer: PastOrders Module

If the user presses "My previous orders" on the CustomerUI module, the user will be directed to the PastOrders module, where the user can select any previous orders they have made and view the order information and washer review (if applicable). The app will receive data from the Washu servers. Reviews are retrieved from the ReviewsDB module.

PastOrders Module Pseudo Code:

```
Wait for customer data from Licensee Washu server
If user selects an order
      Go to ViewPastOrder
      Exit Module
If user hits "Back"
      Go to CustomerUI Module
      Exit Module
```

ViewPastOrder Module Pseudo Code:

```
Display order information
If washer review is present
      Get washer review from ReviewsDB module
If user hits "Back"
      Go to PastOrders Module
      Exit Module
```

Washer

Upon the user entering the Washer option from the app will launch the WasherCheck Module, to check if the user is a washer for the region. WasherCheck uses the InRegion Module.

WasherCheck Module Pseudo Code:

```
Check if user is in Washu Region
If true
      Send user's profile to server for verification
      Wait for server response
      If server response is true
            Go to WasherUI Module
      Else
            Inform user they are not a washer
            Go to AppStart Module
Else
      Inform user they are not in Washu Region
      Wait for User to hit "Retry"
      If User hit "Retry"
      Restart WasherCheck Module
```

If the Washer is a registered washer in the Region, WasherUI Module will launch. WasherUI module contains a scrollable window where the user can select an order made by a customer. When an order is selected, the washer will be directed to the WashInfo module. The user can also view the reviews of any past washes they might have done for customers.

WasherUI Module Pseudo Code:

```
Wait for Licensee WashuServer to populate WashWindow
If user hits "Refresh"
      Wait for Licensee WashuServer to populate WashWindow
If user hits a wash in WashWindow
      Go to WashInfo Module
      Exit Module
If user hits "View Reviews"
      Go to PastWash Module
```

```
        Exit Module
```

The WashInfo Module gets the selected order info from WashServer and displays the car's information and location. If the washer accepts the order, the washer will be directed to the Washing module. Order information includes car owner name, phone number, license plate, picture, order package, and time order was made. The washer can go back to the WasherUI module.

WashInfo Module Pseudo Code:

```
Get wash data from Licensee WashuServer
If user hits "Accept wash"
      Send accept data to Washu Server
      Go to WashSession module
      Exit Module
If user hits "Cancel"
      Go to WasherUI module
      Exit Module
```

The WashSession module displays the car info. As the washer arrives to the car's location, the app waits for the washer to press "Start Wash" before displaying a wash duration timer and order package type. When the wash is complete, the washer presses "Wash Complete". The washer is then required to take a picture of the car afterwards. After a picture has been taken, data will be sent to the WashServer module to notify the server of the order completion.

WashSession Module Pseudo Code:

```
Display car info
If user hits "Start Wash"
      Start wash timer
      Display package type ordered
      If user hits "Wash Complete"
            While picture not taken
                  Ask user to take a picture of the car
                  If picture taken
                        Send data to WashuServer
```

```
Exit Module
```

After a wash is complete, the washer is given the option to rate the customer using with the WashReview Module. The review will be set to the ReviewDB module.

WashReview Module Pseudo Code:

```
While (rate out of 5 is false)
      User rates the car owner out of 5
      User fills in a review of the wash
      When user hits "Submit", check if rate out of 5 has not
      been filled
            Inform user to rate out of 5
Send review to WashuServer Module
Exit WashReview Module
```

Washer: PastWash Module

If the user presses "View Reviews" from the WasherUI module, the user is directed to the PastWash module where they can view any past washes done. The user can select any past review and view the customer review (if applicable).

PastWash Module Pseudo Code:

```
Get data from Washu servers
If user hits a past wash
      Go to ViewPastWash Module
      Exit Module
```

If the user presses a wash done previously, the user is directed to the ViewPastWash Module.

ViewPastWash Module Pseudo Code:

```
Get review from ReviewsDB module
Display review from ReviewsDB module
```

Business Manager

The business manager's profile has similar fields as a Washu car owner and car washer, but the manager profile also holds data regarding their business. They are also able to access manager specific modules such as creating a wash region, assign washers, and contact customers. The basic Business Manager modules are shown below.

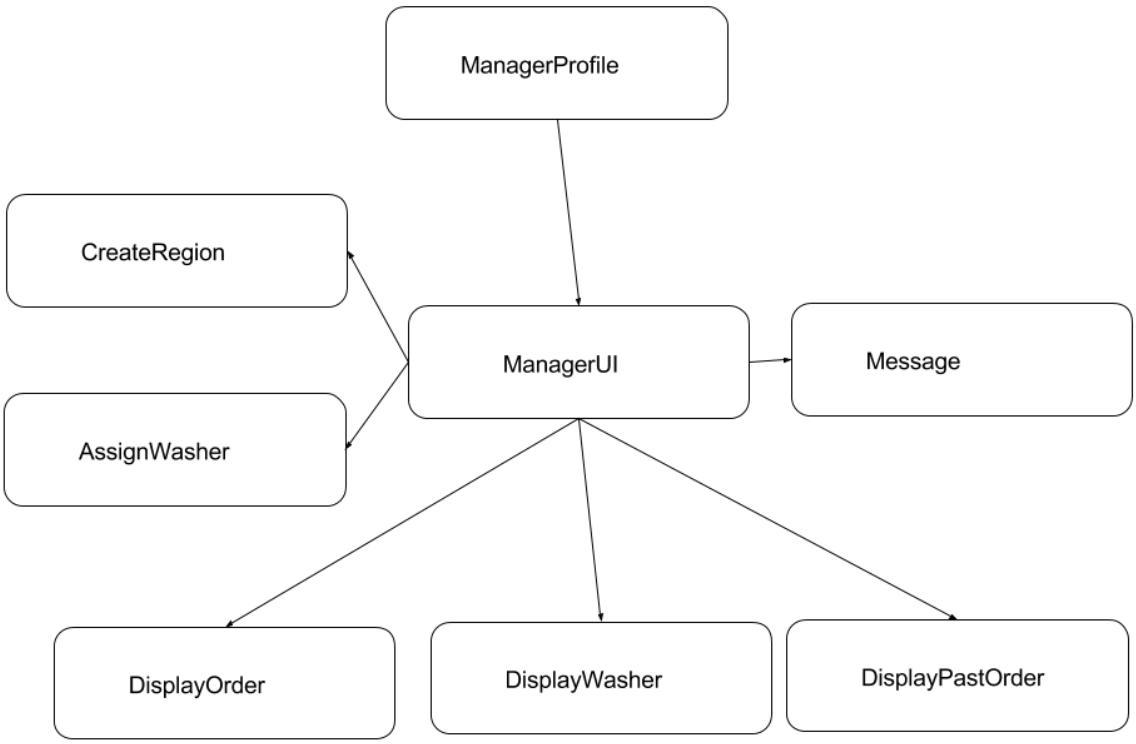

Figure 1. Organization of the manager modules.

ManagerProfile Module Pseudo code:

```
While (all fields not filled)
     Ask for user's first name field
     Ask for user's last name field
     Ask for user's email field
     Ask for user's phone number field
     Ask for user's business name
     When user hits "submit", check if any fields not filled
          Inform user all fields must be filled
```

```
Exit ManagerProfile
```

If a business manager is logged in and loads up the Washu application, the ManagerUI module will launch to display an interface for the manager to interact with the Washu system. The Display modules called by the ManagerUI outlined below are not detailed due to their trivial functionality.

ManagerUI Module Pseudo code:

```
Wait for Server to populate Washers and CurrentOrders
If user hits "refresh"
     Wait for Server to populate Washers and CurrentOrders
     Display status
If user hits a washer
     Go to DisplayWasher Module
If user hits a current order
     Go to DisplayOrder Module
If user hits a past order
     Go to DisplayPastOrder Module
```

Business managers are able to create a Washu wash region through their device. This is done with the CreateRegion module.

CreateRegion Module Pseudo code:

```
While (traced region not enclosed)
     Read user finger trace on screen
Show traced region
If region is correct
     Save Washu region to server
Else
     Retrace region
```

Once a Washu region has been set by a business manager, the business manager is then able to assign their employed washers to it. Assignment of a Washu car washer to a set Washu region is done with the AssignWasher module. Once a washer has been assigned, they will be able to receive notifications regarding car wash orders.

AssignWasherModule Pseudo code:

```
Check if WashuRegion does not exist
      Exit Module
Pull from server list of employed car washers(optional)
WasherToAssign = Washers(name)
WashuRegion.assign(WasherToAssign)
```

Business Managers are able to contact customers if it is needed in order to clarify a review or improve customer service. Contacting a customer can be through reviewing their contact information, or done through messaging through the Washu App. Since pulling customer information from the server is trivial, this module outlines the messaging feature.

MessageModule Pseudo code:

```
Open messaging request
Send to server IP Address, Port and recipient information
Wait for response from server
If Timeout
      Return Timeout Error
Retrieve recipient IP Address and Port information from
server

While (ConnectionIsOpen)
      Display recipient message
      Save text input into buffer
      If "Send" clicked
            Send message in buffer
      If ("End Conversation" clicked)
            Close connection
            Set ConnectionIsOpen to False
```

MessageModuleReceive Pseudo code:

```
Receive message initiation from server
If User does not accept
      Return conversation declined
```

```
Else
      Send IP Address and Port to server
      Wait for Message initiator IP Address and Port
      If timeout
            Return timeout error
      Establish connection
```

Review Database

All reviews are stored in the Washu reviews database. Each Washu licensee will have a database for the licensee's viewing. The licensee can only view reviews from orders made in the licensee's region. Business Managers can select an order and view the order information and reviews from both customers and washer (if applicable). Customers can view washer reviews given by the washer of the respective order. Washers can view customer reviews given by the customer of the respective order.

When a request is given to retrieve review information, ReviewDB module sends information from Washu reviews database based on the user type: Customer, Washer, and Business Manager.

ReviewDB Module Pseudo Code:

```
Get reviews database from Washu Reviews database.
If user is Business Manager
      Send customer and washer reviews
If user is Customer
      Send washer reviews
If user is Washer
      Send customer reviews
```

Washu Servers

Each Washu licensee will have a server to allow business managers and washers to view customer orders in their respective regions. Washu Servers also sends all reviews to licensees.

When a request to send reviews is given, Washu servers will send all reviews to the respective licensee. When a review from a customer or washer is sent, Washu Servers will store the data into Washu Reviews database
When an order is made, the order information is sent to Washu Servers to be stored. The data is then sent to the licensee's server.

WashuServer Module Pseudo Code:

```
If order is made
      Store order information to server
      Store order information to licensee server
      Add order to pending orders
If request for view pending orders
      Send all pending orders based on licensee server
If order is accepted
      Remove order from pending orders
      Update order information to server
      Update order information to licensee server
If order is complete
      Update order information to server
      Update order information to licensee server
      Store order information to server
      Store order information to licensee server
If review is sent
      Store review into Washu Review Database
If request for reviews given
      Send all customer and/or washer reviews based on
licensee
```

Data flow

Client-server model

Washu uses the client-server model for data communication. The mobile application acts as a client that sends and requests data from the main Washu server. The Washu server stores information such as customers, washers, businesses and their managers, payment information, Washu regions and more. This data is sent to the application device when requested.

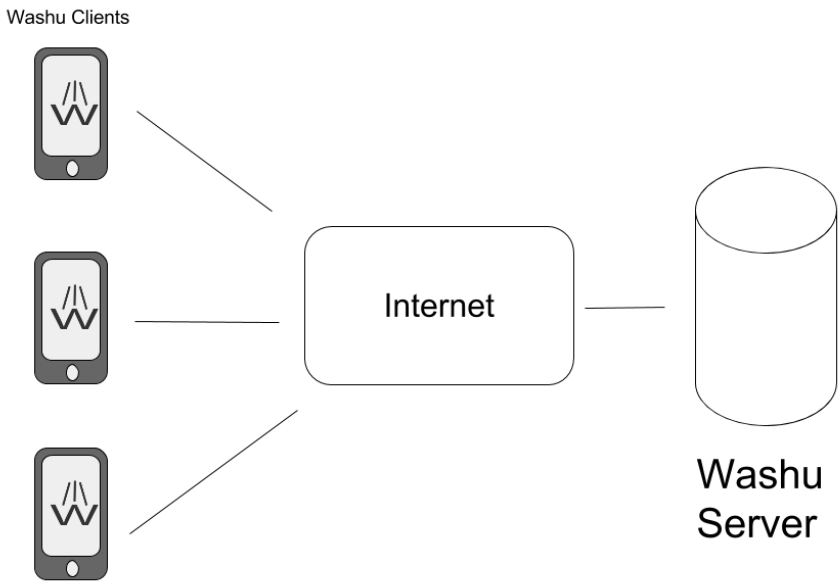

Figure. 2 Washu Client-Server Model

Text data

Text data such as profile information or satisfaction reviews are pulled from forms in the Washu app. This text data is linked to the user that is logged in and stored in a database on the main Washu server. When this data is needed for view in a Washu mobile application, a

request is sent to the server and the server responds with the appropriate data via a web API.

Tap input

The majority of tap and click interactions are to navigate through and access different functions and features within the Washu application. This data is not stored within the client app or server, but data manipulation resulting from it may be stored. Examples include selecting a wash rating or uploading an image(further details explained below).

Payment Information Data

A customer enters in their payment information such as a credit card into a form in the Washu app. Before this information is sent and stored on the server, it is encrypted to protect the customer. Decryption of this data for processing is then done on the Washu server. A proven secure encryption system for bulk data transfer can be used such as RSA. This method can be used not only for payment information but for other user sensitive data.

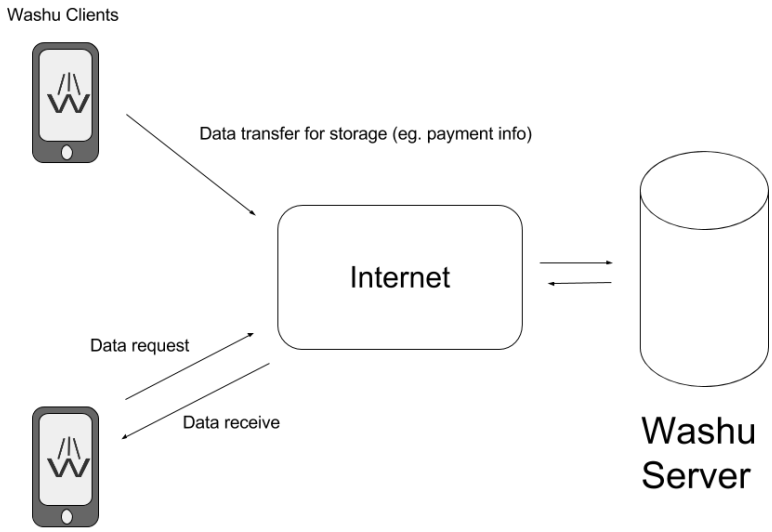

Figure 3. Client-server data transfer

Camera Image data

Image data is generated through the mobile device that runs Washu. This data is generated when a customer or car washer uses the application to take an image of the car that is being operated on by the wash order. This image data is then linked to the user that is logged into the application along with the wash order for future reference. Once the image is tied to the user and wash order within the Washu application, this data set is sent for storage and recall on the Washu server.

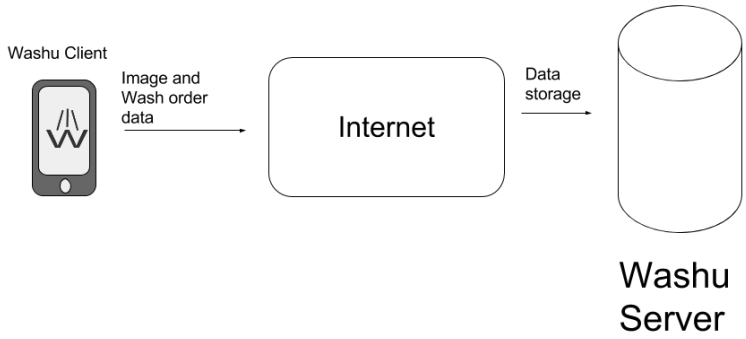

Figure 4. Image data storage

Message

A messaging connection is started between clients and the server by the requesting client sending connecting information such as the requesting client's IP address and port number of the device, and the profile information of the person they client wishes to connect to, all to the server. The server then contacts the client on the receiving end of the message request, and asks if they want to accept the message request. If accepted, this client's IP address and port will be sent to the server. The server will then be able to set up direct communication with the two clients using their IP information. Text data will then be sent directly between the clients to ensure instant response times. After a messaging session concludes, each client will be asked if they would like their sessions saved. If so, the saved text session will be sent to the server for storage and later access.

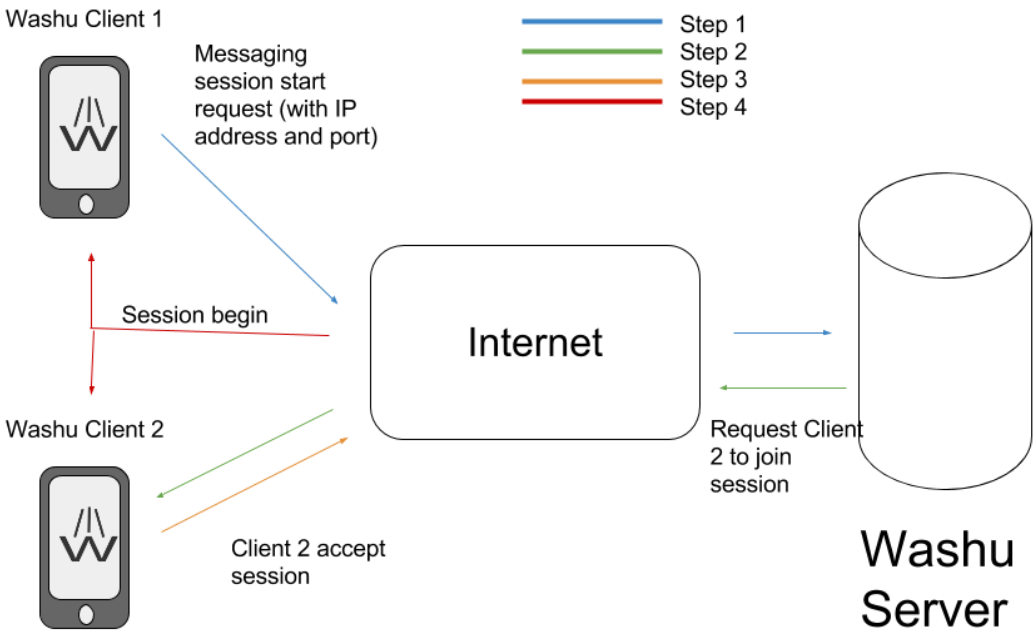

Figure 5. Messaging initialization

Mobile device notifications

Notifications are sent to a user's mobile device from the Washu server. A notification is triggered when a customer enters a Washu region, a wash is ordered, a wash order is marked as complete, or a message is sent from another user.

When a customer enters a Washu region, their application receives data from the server to notify them that Washu services are available. The application then sends a notification to the device to notify the user.

When a customer starts a wash order, the order information is sent to the Washu server. The server looks at the order data and forwards this information to the chosen Washu car washer. Upon receiving the data on the washer's device, the Washu application will push a notification to let the washer know that a wash has been ordered.

When a wash order is marked as complete by a car washer, the Washu server sends data to the customer of the wash to notify them of a complete wash. The mobile application responds by sending a notification to the customer's device.

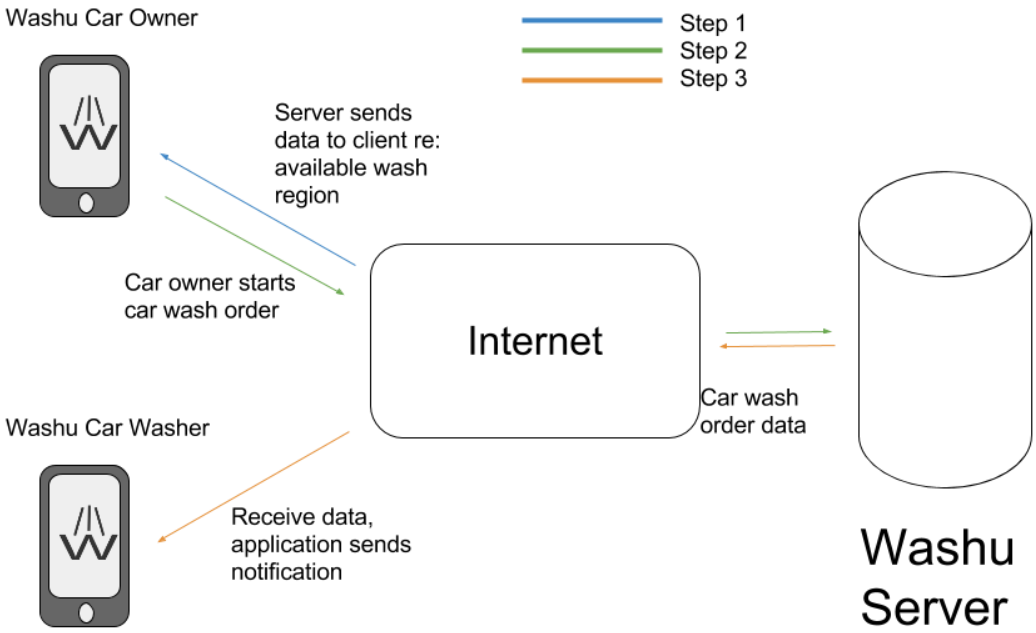

Figure 6. Push notification flow

Coding Responsibilities

Manager	
AssignWasherModule	Richard Lui
MessageModule	Trison Nguyen
MessageModuleReceive	Trison Nguyen
CreateRegion	Richard Lui
ManagerUI	Richard Lui
ManagerProfile	Richard Lui
WasherUI	Adewale Adekoya
WasherCheck	Adewale Adekoya
Customer	
CustReview	Trison Nguyen
Receipt	Chris Kelly
PaymentCheck	Chris Kelly
PaymentType	Chris Kelly
InRegion	Trison Nguyen
CustOrder	Chris Kelly
General	
AppStart	Adewale Adekoya
AppProfile	Adewale Adekoya

Table 1. Coding responsibilities for different modules.

Test Plan

Introduction

Several types of testing will be employed in order to evaluate the product. Tests will be used primarily to ensure that the application conforms to the requirements. The majority of the tests will be black box tests, meaning that the tests will not have knowledge about the specific implementation of the application. Both unit and integration tests will be implemented with the JUnit testing framework. White box testing will be avoided at this stage, since the product will be a prototype at this point. A summary of the different tests, their objectives, and the testing plan can be found in Table 1.

Test type	Description and objectives	Person responsible
Unit tests	Unit tests will be used to test the expected sets of inputs and outputs of specific pieces of the application. Automated tests written by the application programmer each test will target specific application behaviour.	Chris Kelly
Integration tests	Integration tests will be used to test the behaviour of larger scale features of the application, testing the interactions of multiple components simultaneously.	Adewale Adekoya
Functional tests	Feature tests will be used to test the outputs of specific features against the requirements specifications, this would be done with the use of automated UI testing frameworks	Trison Nguyen

Performance evaluation	Evaluating the current performance of the application and scalability issues. We will be integrating app analytics to gather data from real usage as well	Chris Kelly
Acceptance testing	A demo of the application	Richard Lui

Table 2. Test objectives, descriptions, dates, and responsibilities.

Integration plan

Some of the application's subsystems do not depend tightly on other subsystems, and can therefore be implemented independently of other subsystems. Other subsystems depend on each other, and must be implemented in order to a degree. The first system to be implemented will be the database models for users, payments, and car wash locations, along with user roles and permissions attributes.

Once the data models are in place, the CustOrder module will be implemented, and the PaymentType, PaymentCheck, and Receipt modules will be implemented in order. The CustOrder, PaymentType, PaymentCheck, and Receipt modules will then be integrated, since together they are coupled tightly to form the customer order flow. Integration tests can then be implemented for the customer order process, testing the entire order process from start to finish, while mocking out the customer credentials (customer login will be implemented later).

Next, the manager-specific modules can be implemented and integrated. First, the ManagerProfile module will be implemented, followed by the ManagerUI module, since the ManagerUI depends on the ManagerProfile module for functionality. The ManagerUI and ManagerProfile modules can then be integrated. Next, AssignWasherModule and CreateRegion modules can be implemented and integrated with the other manager modules for testing.

Afterwards, the WasherUI module will be implemented, along with the WasherCheck module, which required the CreateRegion module loosely. The WasherUI and WasherCheck will then be integrated, ready for integration testing.

After the washer modules have been implemented and integrated, the MessageModule and MessageModuleReceive modules for the manager will be implemented and integrated. These modules depend on both the user UI and manager UI to be functioning.

Next, the InRegion and CustReview modules will be implemented and integrated, since the CustReview and InRegion modules both depend on other modules.

Last, the AppStart module will be implemented, followed by the AppProfile module. These modules can be integrated last, since they will follow closely the default implementations for Android. Before they are integrated, their functionality can be ignored, by mocking out user credentials.

Unit testing

For our unit testing we would be making use of Android JUnit framework, it is a very well known (and officially recommended by Google) testing framework for Android. the JUnit Android extensions would allow for us to mock Android components. We may combine the JUnit with Mockito, another Android mocking framework. The other approach we may take is a testing framework that allows us to run our Android unit tests in the normal JVM(Java Virtual Machine), without the need for an emulator. This enables our tests to not only run within our IDE, but also as part of a continuous integration workflow.

The unit tests would be used to:

- Improve testing coverage
- Test some of the Washu application assumptions
- Validate the Washu app functionality

A test method begins with the @Test annotation and contains the code to verify a single functionality in the component that we want to test. Let's begin by testing the "Order a wash" functionality of the Washu app, to order a wash, the user would first have to be in a marked Washu region. To test the functionality, we add a method called **washuInRegionValidator_ReturnsTrue()** to the washuValidation class.

```
import.org.junit.Test;
import.java.util.regex.pattern;
import static org.junit.Assert.assertFalse;
```

```
import static org.junit.Assert.assertTrue;

public class washuValidation{
	@Test
	public void washuIsRegValid_ReturnsTrue(){
		assertThat(washuIsRegValid.isValidReg("Marked locations"),
	is(true));
	}
	...
}
```

The test that the components in our app return the expected results, we will make use of the **JUnit.Assert()** methods to perform validation checks(or assertions to compare the state of the component under test against some expected value.

Unit tests

Test	Submit wash order success
Inputs	Wash package name, car image, car license number
Expected outputs	Notification of car was order complete

Test	Submit wash order failure
Inputs	Wash package name, car image, car license number(at least one parameter invalid)
Expected outputs	Notification of car was order failure, user noted of invalid value for paramater

Test	Email receipt success
Inputs	Payment information, Washu package ordered, Date or order, Cost of package
Expected outputs	Filled receipt form, notification of receipt send success

Test	Email receipt failure
Inputs	Payment information, Washu package ordered, Date or order, Cost of package (at least one parameter invalid)
Expected outputs	Notification of receipt send failiure, user noted of invalid paramater

Test	Payment information valid
Inputs	Payment information
Expected outputs	Notification of payment valid

Test	Payment information invalid
Inputs	Payment information
Expected outputs	Notification of payment invalid

Test	Submit Manager profile success
Inputs	First name, last name, email, phone number, business name
Expected outputs	Notification of manager profile create success

Test	Submit Manager profile failure
Inputs	First name, last name, email, phone number, business name (at least one invalid parameter)
Expected outputs	Notification of manager profile create failure, missing/incorrect fields noted to user

Test	Submit Washu region data points success
Inputs	Data points for enclosed shape
Expected outputs	Notification of Washu region create success

Test	Submit Washu region failure
Inputs	Data points for enclosed shape

Expected outputs	Notification of Washu region create failure

Test	Assign Washer to region success
Inputs	Washer name, Wash region ID
Expected outputs	Notification of washer assign success, Washer profile updated

Test	Assign Washer to region failure
Inputs	Washer name, Wash region ID
Expected outputs	Notification of washer assign failure

Integration testing

Integration testing will be done in order to ensure that the different subsystems of the application work correctly together. For example, an integration tests will need to be written to test the “Cancel Transaction” during checkout, after the successful cancellation of a transaction, a display message notifying the user should pop-up. Integration tests will be written using the JUnit test framework in conjunction with the Espresso APIs. The integration test would basically verify that the target app behaves as expected when a user performs a specific action or enters a specific input in its activities. It will allow us to check that the target app returns the correct UI output in response to user interactions. Integration tests outputs and inputs will follow the outcomes given in the state diagrams given in the conceptual design document.

```
@RunWith(AndroidJUnit.class)
@Test
public class CancelTransactionTest{
      public void cancelTrans(){
```

```
        @Rule
        public ActivityTestRule washuActivityRule = new
    ActivityTestRule<>(MainActivity.class);

        onView(withText("Cancel Transaction")).perform(click());

        onView(withId(Washu.textView)).check(matches(withText("Successfu
        l")));
    }
}
```

The integration test above makes use of the Espresso APIs to perform a click action on a UI element, then checks to see if an expected string is displayed.

Integration tests

Test	Order car wash success
Inputs	Submit car wash request with valid parameters
Expected outputs	Notification of submission success

Test	Order car wash failure
Inputs	Submit car wash request with invalid parameters
Expected outputs	Notification of submission failure

Test	Order car wash with photo upload success
Inputs	User uploads a photo during order process

Expected outputs	Notification of submission success

Test	Order car wash with photo upload failure
Inputs	User fails to a photo during order process
Expected outputs	User prompted to re-upload photo

Test	Order car wash by banned user
Inputs	Banned user orders a car wash
Expected outputs	User is notified that they are banned.

Test	Register car owner account with valid parameters
Inputs	Register account with valid credentials
Expected outputs	● User is notified of account creation success. ● User redirected to login page.

Test	Register account with previously claimed username
Inputs	Register account with previously claimed username
Expected outputs	User is notified that their chosen username is already in use.

Test	Register account with invalid email
Inputs	Register account with previously invalid email
Expected outputs	User is notified that their email is invalid

Test	Create car wash region with valid parameters
Inputs	Manager creates a car wash region with a valid region drawing
Expected outputs	● Marked region is highlighted ● Manager is notified that the region is available for review

Test	Create car wash region with overlapping region
Inputs	Manager creates a car wash region with a region drawing that overlaps an existing region
Expected outputs	● Manager is notified that the selected region overlaps an existing region, which is not allowed. ● Manager is prompted to select a new region again.

Test	Create car wash region with region that is too large
Inputs	Manager creates a car wash region with a region drawing that has a total area greater than 1km^2
Expected outputs	● Manager is notified that the selected region is too large (greater than 1km^2 ● Manager is prompted to select a new region again.

Functional testing

Functional testing will be done in order to test the output of the system for specific inputs. This will allow the requirements specifications to be directly tested for different functions in the application. The JUnit test framework will be used to implement all of the functional tests. Functional testing can be carried out as the application is developed, since it does not necessarily require all subsystems to be functional.

Performance Evaluation

A performance evaluation will be performed in order measure and predict both the current and future performance, as well as any scalability issues that may need to be considered in the future.

The performance evaluation will consider and measure:

- Response times
    - The response times of central application servers.
    - The response times of the app's interface to different actions
- Scalability considerations
    - Server load bottlenecks
    - Database bottlenecks
    - Caching

- Payment processing bottlenecks

Acceptance Testing

Extensive acceptance testing will be done to ensure that the overall look, feel, features, behaviour, and performance meet the requirements specifications. Acceptance testing will provide insight into the efficacy of different user experience and interface decisions. In addition, acceptance testing tests the performance of the entire washu system as a whole. Acceptance testing will be done by doing a walkthrough of the different use case scenarios using the app prototype. The acceptance testing stage will culminate in a live demo of the prototype in order to demonstrate features to and solicit feedback from the client regarding the product.

Monitoring, bug reporting,and bug correcting

Multiple different methods will be used to monitor the application, and to report and correct bugs during development. In order to ensure that new changes pass tests and do not break functionality, a continuous integration server such as CircleCI will be used to ensure that before code can me merged into master, it must pass all tests. In addition, code review on github will be used to ensure that all involved people understand what changes are being made to the system, and no obvious bugs are introduced during development.

In order to ensure that bugs are noticed and fixed quickly, a bug tracking system must be put in place. For this purpose, the github “Issues” feature will be used. Whenever a tester or developer discovers a bug in the code that has already been incorporated into the master branch, a new issue will be created on github. Issues will be given different tags in order to differentiate their importance and criticality, ensuring that the most critical bugs are dealt with first. Different developers will be able to assign themselves to issues, comment and collaborate on issues, and mark issues as complete.

Testing Schedule

Testing Stage	Target start date	Target completion date
Unit testing	Week of 21 March 2016	Week of 28 March 2016

Functional testing	Week of 21 March 2016	Week of 28 March 2016
Integration testing	Week of 21 March 2016	Week of 28 March 2016
Performance evaluation	Week of 28 March 2016	1 April 2016
Acceptance testing	Week of 28 March 2016	1 April 2016

Table 3 . Schedule for different testing stages.

Executive Summary

In today's fast-paced culture, people have more commitments and less time for other activities. One such important but often neglected activity is washing one's car. The Washu app aims to provide higher revenue to businesses, jobs to car washers, and convenience to the general public by allowing people to have their car washed while they shop.

The Washu application will be composed of multiple different modules. The customer-facing functionality is decomposed into different modules for implementing the order process, review process, and login process. The manager-facing functionality is decomposed into modules that implement region management, employee management, and customer contact. The washer-facing functionality is decomposed into modules implementing region verification and interface. The app will be implemented in stages, with each stage being comprised of different modules that compose the app. In general, the back end will be implemented before the front end interface of each type of module, with customer-related modules implemented first, manager-related modules implemented second, and washer-related modules implemented third, before general-purpose modules.

During development, the application will be tested in several different stages during different stages of development. Unit testing will be carried out simultaneous with all development in order to test specific module functionality. Integration tests will be used once multiple modules have been integrated to form a piece of functionality in order to ensure the modules interact correctly to implement the required functionality. Functional tests will be used to ensure that the application adheres to the requirements specifications. Performance evaluations will be conducted in order to assess possible performance issues and future scalability issues. Finally, acceptance testing will be carried out, including walkthroughs of different use case scenarios and a live demo of the application to ensure customer

acceptance of the prototype. Coding and testing responsibilities have been assigned to team members based on schedule requirements and the specific skill-sets of the persons in question.

A variety of tools will be used to ensure that both bug reporting is timely and efficient, and no broken code is submitted into the master development branch. A continuous integration server will be used to ensure that the master branch only ever contains a valid build. Github issues will be used for bug reporting, tracking, and resolution.

Conceptual Design

Users

Washu consists of the following classes of users:

Car Owners:

Car owners are the primary customers of Washu. They interact with the Washu mobile application to get Washu car was regions, order car washes, set their user profile, mark their car on a map, upload images, and leave customer satisfaction reviews. permissions?

Car Washers:

Car washers are employed by the business managers that license the Washu application. Their user profiles are set up by the business manager. Car washers use the Washu application to get notified of ordered car washes, see a map and list of Washu regions and car wash orders, mark completed car washes, upload wash complete images, see their wash service ratings and leave ratings for their customers.

Business Manager:

Managers handle the administrative aspects of the server. Managers use the Washu application to set and manage car wash regions, set up car washer profiles including image uploading and see car owner and car washer reviews.

Use Case Diagrams

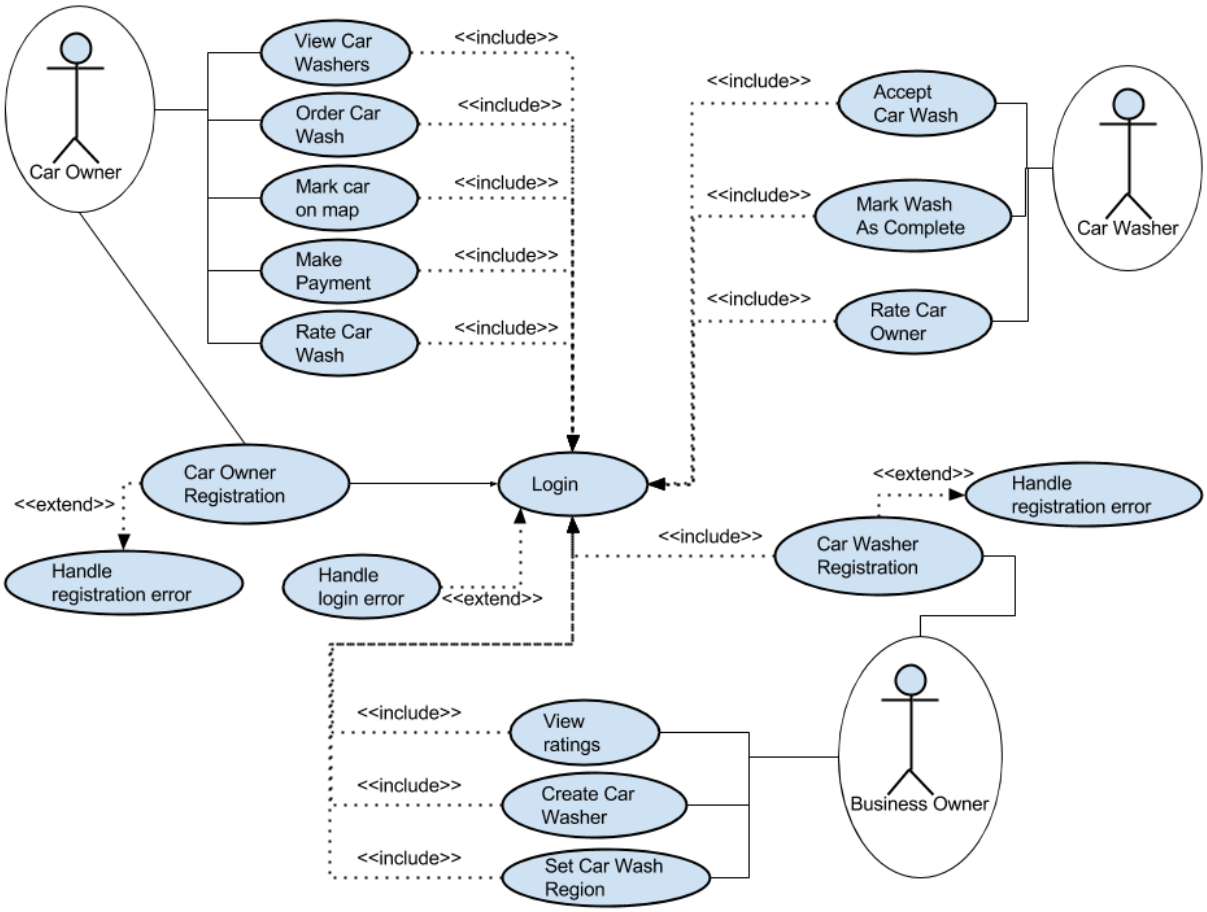

Figure 7. Use case diagram for Washu system.

Use Cases

ID	1
Description	Car owner sets up user profile
Actors	Car owner
Preconditions	Car Owner has downloaded Washu application and has notifications allowed on their mobile device
Basic Steps	● Car owner opens up the Washu app ● Car owner enters personal information to set up a profile
Exceptions	Invalid form input eg. no “@” for email address, invalid number of digits for phone number
Business validations/Rules	User profile cannot contain profanity
Postconditions	Car Owner profile is set up with Washu

ID	2
Description	Car owner receives notification of entering Washu car wash region

Actors	Car owner
Preconditions	Car Owner has downloaded Washu application and has notifications allowed on their mobile device
Basic Steps	● Car Owner enters Washu car wash region ● Washu application pushes notifies to car owner's mobile device ● Car Owner accepts or declines car wash
Exceptions	N/A
Business validations/Rules	N/A
Postconditions	Notification is removed after user decision to accept or decline wash

ID	3
Description	Car owner orders car wash
Actors	Car owner
Preconditions	Car Owner has logged into Washu application Payment information completed and verified Car Owner in Wash car wash region
Basic Steps	● Car Owner selects wash package ● Car Owner verifies package ordered and price
Exceptions	N/A
Business	Only available services and washers are displayed

validations/Rules	
Postconditions	Car owner is informed car wash order has been placed

ID	4
Description	Car owner uploads image of vehicle
Actors	Car owner
Preconditions	Car Owner has logged into Washu application
Basic Steps	● Car owner takes a picture of their car using the Washu camera feature ● Car owner uploads the picture to the Washu
Exceptions	N/A
Business validations/Rules	Only the device's still image capture feature is used ie. device's video option is not accessible through Washu
Postconditions	Image uploaded and associated with car owner/wash order

ID	5
Description	Car owner makes payment
Actors	Car owner
Preconditions	● Car Owner has logged into Washu application and has payment information included in their profile.

	● User has started car wash order
Basic Steps	● Washu app automatically charges the user's credit card
Exceptions	No credit card info, ask user to input their payment information
Business validations/Rules	N/A
Postconditions	Car owner is notified that payment is complete

ID	6
Description	Car owner marks their car on a map
Actors	Car owner
Preconditions	Car owner is logged into the Washu app and has uploaded an image of their vehicle Car owner has started car wash order
Basic Steps	● User taps on Mark Vehicle On Map ● User sees view of wash region and taps on location of their car
Exceptions	User cannot tap outside of car wash region (invalid region)
Business validations/Rules	User can only make one mark of their car on the map.
Postconditions	The car wash order contains mark of car owner's vehicle location on wash region map.

ID	7
Description	Car owner leaves review for car washer
Actors	Car owner
Preconditions	Car Owner has logged into Washu application and wash is marked completed by Car Washer
Basic Steps	● User opens completed wash ● User taps on Leave Washer Review ● User chooses rating and (optionally) enters in comment ● User taps on Submit Review
Exceptions	No review rating is chosen(required)
Business validations/Rules	Review comment cannot contain profanity
Postconditions	A car washer review is associated with its respective wash order.

ID	8
Description	Car washer accepts notification of car order
Actors	Car washer
Preconditions	● Car Washer has downloaded Washu application and has notifications allowed on their mobile device. ● Business manager has set up their user profile.

	• Car Washer is logged in
Basic Steps	• A notification is pushed to the user's mobile device • User taps on Accept to accept the car wash order
Exceptions	N/A
Business validations/Rules	N/A
Postconditions	The car wash is marked as in progress.

ID	9
Description	Car washer marks wash as complete
Actors	Car washer
Preconditions	Car Washer is logged in
Basic Steps	• User opens car wash that is in progress • User taps on Mark Wash As Complete
Exceptions	N/A
Business validations/Rules	N/A
Postconditions	The carwash in progress is marked as complete. The car owner receives a notification stating so.

ID	10
Description	Car washer uploads image
Actors	Car washer
Preconditions	Car Washer is logged in, Car washer has completed a wash
Basic Steps	● User taps on Upload Image ● Device's camera application is opened to take a photo ● Picture is shown for review and confirmation ● User taps on Retake to take another image ● User taps on Upload to finalize image choice
Exceptions	N/A
Business validations/Rules	Device's camera functionality is restricted to image stills only
Postconditions	An image is associated with the car washer's wash

ID	11
Description	Car washer leaves review for customer
Actors	Car washer
Preconditions	Car Washer is logged in
Basic Steps	● User taps on completed car wash ● User taps on Leave Customer Review ● User enters in rating and comment in review form

Exceptions	Invalid review when no star rating is entered
Business validations/Rules	Review comment cannot include profanity
Postconditions	A customer review is associated with its respective completed car wash

ID	12
Description	Business manager sets car wash region
Actors	Business manager
Preconditions	● Business manager has downloaded Washu application ● Business manager is logged in
Basic Steps	● User taps on Set Car Wash Region ● User enters approximate location into app to see map of area ● User draws perimeter on screen to enclose car wash region
Exceptions	User cannot draw into/over pre-existing region.
Business validations/Rules	User cannot draw into non-populable or vehicle accessible area eg. bodies of water, forests
Postconditions	Car wash region is registered with business

ID	13
Description	Business manager creates car washer profile
Actors	Business manager
Preconditions	Business manager is logged in
Basic Steps	● User taps on Create New Car Washer ● User fills out car washer profile information ● User assigns car washer to car wash region
Exceptions	Invalid profile form entry eg. missing digits for valid phone number, missing “@” for email address
Business validations/Rules	User profile cannot contain profanity
Postconditions	Car Washer is registered

ID	14
Description	Business manager looks at car washer and car owner reviews
Actors	Business manager
Preconditions	Business manager is logged in
Basic Steps	● User opens list of completed car washes ● User taps on chosen car wash to see details ● User taps on See Review
Exceptions	No car washers registered, or no car washes completed
Business	N/A

validations/Rules	
Postconditions	N/A

ID	15
Description	Business manager contacts customer
Actors	Business manager
Preconditions	Business manager is logged in
Basic Steps	● User taps on the wash with desired customer from list of complete washes ● User taps on contact customer ● User chooses option to contact customer by message through the application, or through email, or phone number.
Exceptions	N/A
Business validations/Rules	N/A
Postconditions	Business manager sees form to send message to customer through Washu, or user email application is opened, or voice call application is opened

Deployment Diagram for the Washu app

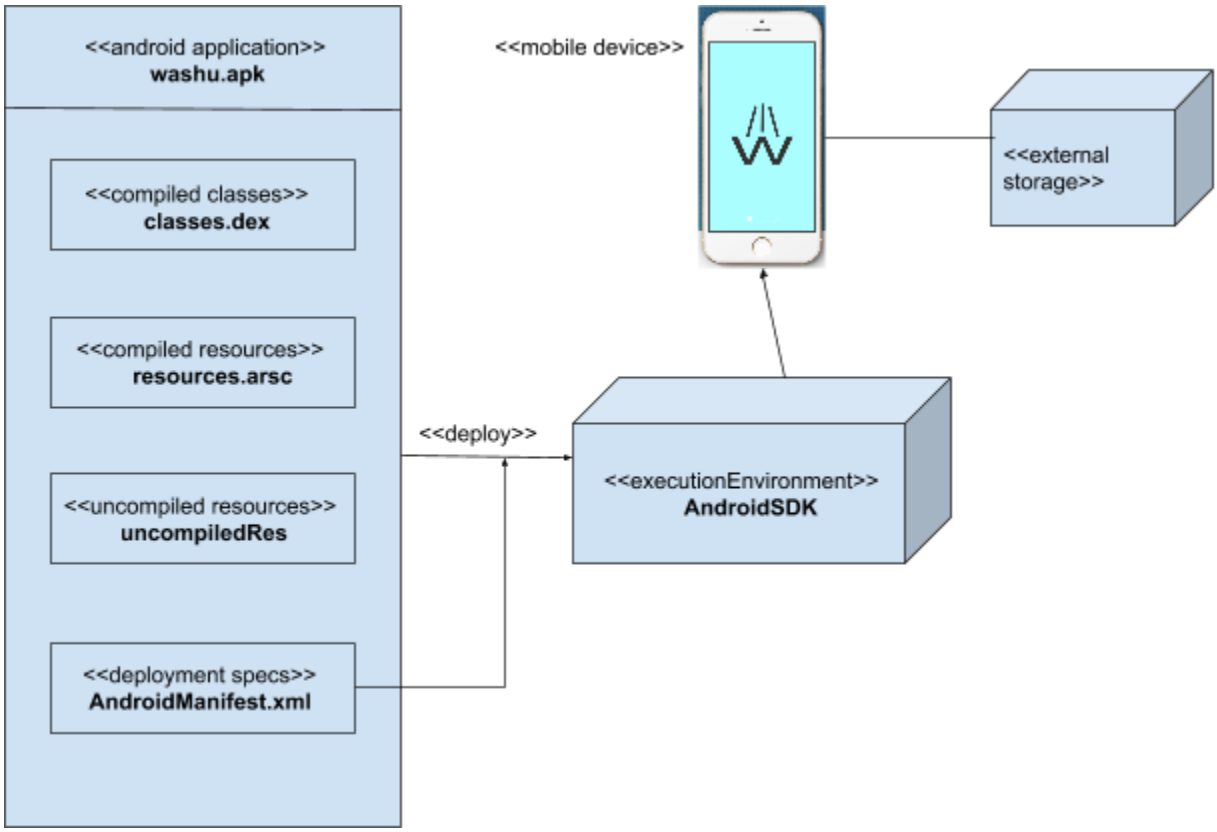

Figure 8. Deployment diagram for Washu system.

The “manifest” file **AndroidManifest.xml** describes application requirements, such as the minimum version of Android required and any supported hardware configurations, and it also declares all components in the application.

The external storage box is optional, with Android API Level 8 or later, application could be installed on the external storage (SDcard), this could be requested for the specific application using a manifest attribute. The default setting is that applications are installed on the internal storage

Activity diagrams

Manager activity diagrams

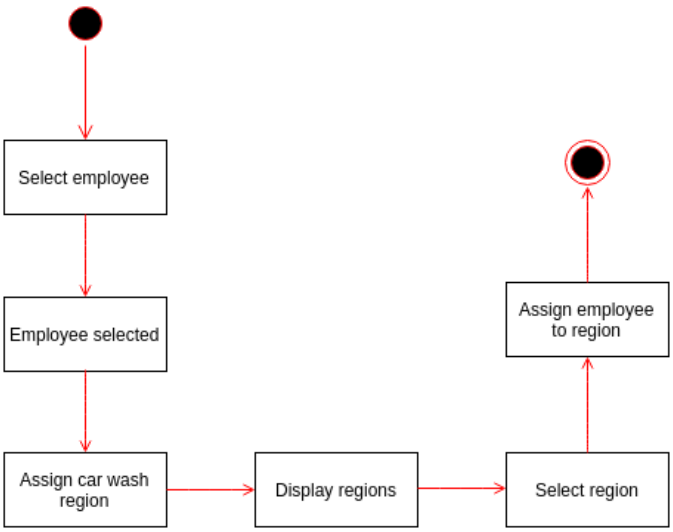

Figure 8. Activity diagram describing the process of a manager adding a washer to a region.

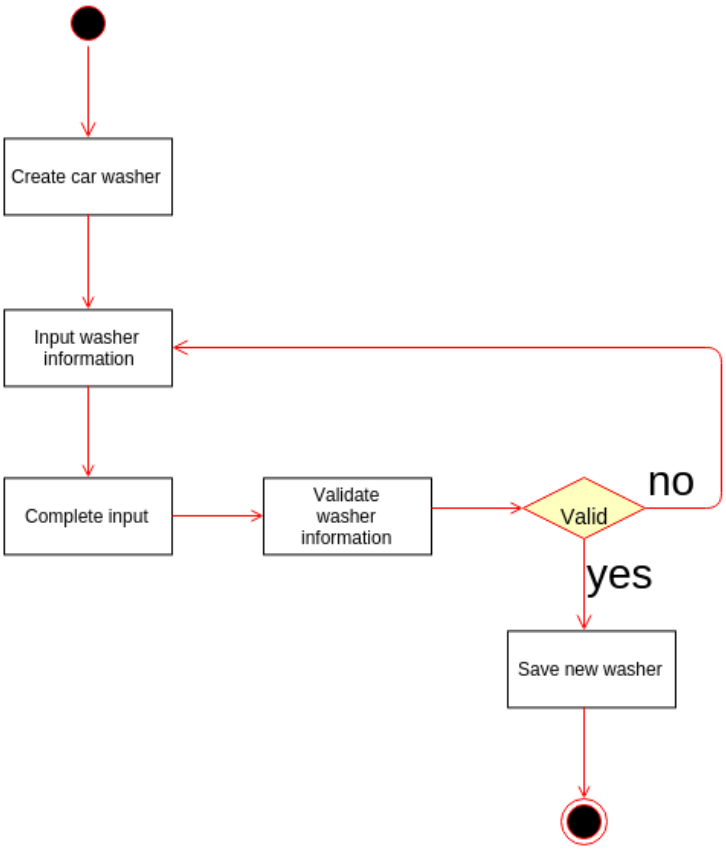

Figure 9. Activity diagram describing the process of a manager adding a new car washer account.

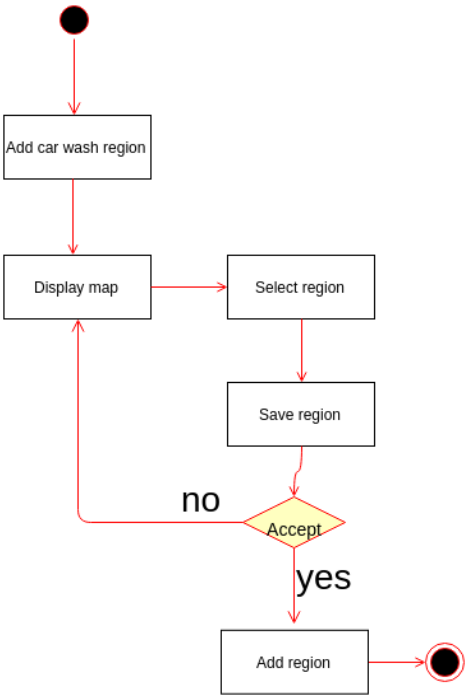

Figure 10. Activity diagram describing the process of a manager adding a new region.

Customer activity diagrams

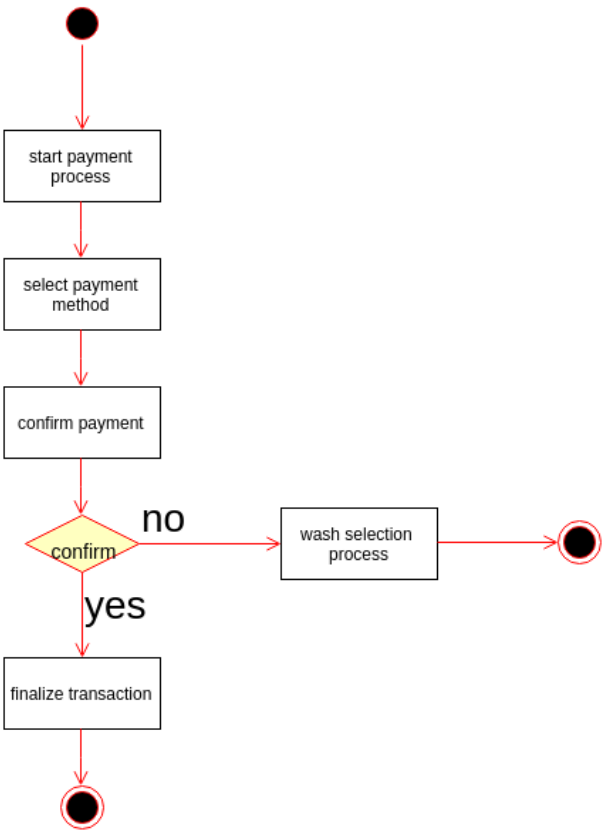

Figure 11. Activity diagram describing the process of a customer completing the payment for a car wash.

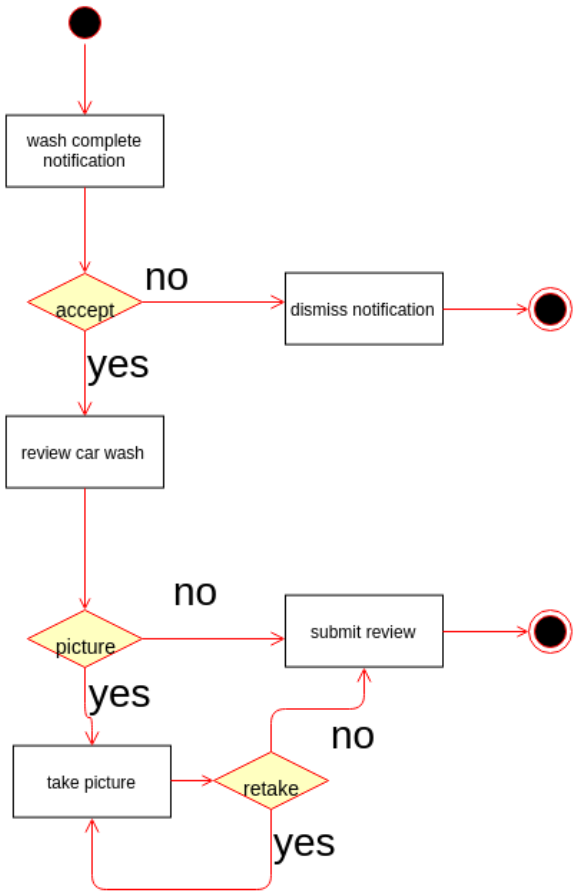

Figure 12. Activity diagram describing the process of a customer reviewing a completed car wash.

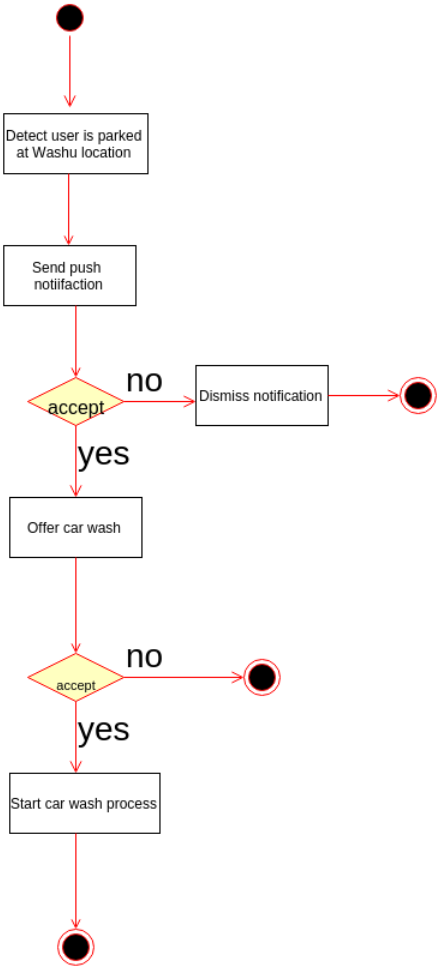

Figure 13. Activity diagram describing the process of a customer receiving a car wash notification.

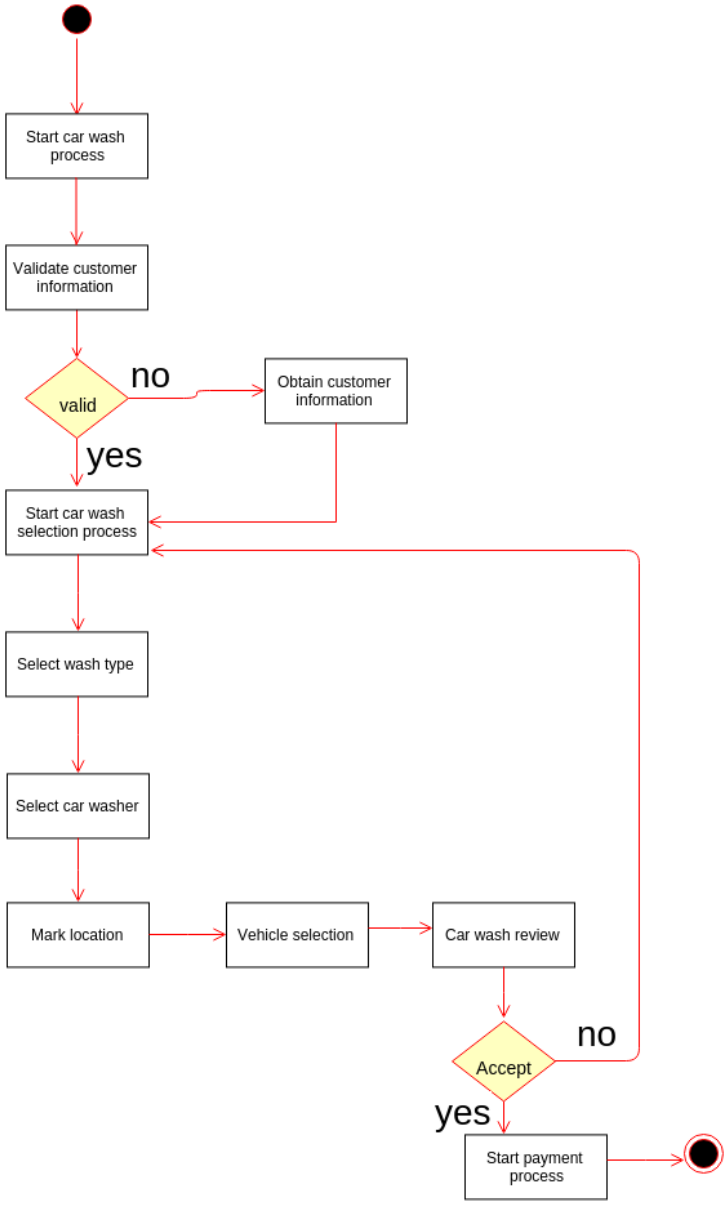

Figure 14. Activity diagram describing the process of a customer selecting a car wash and washer.

Car washer activity diagrams

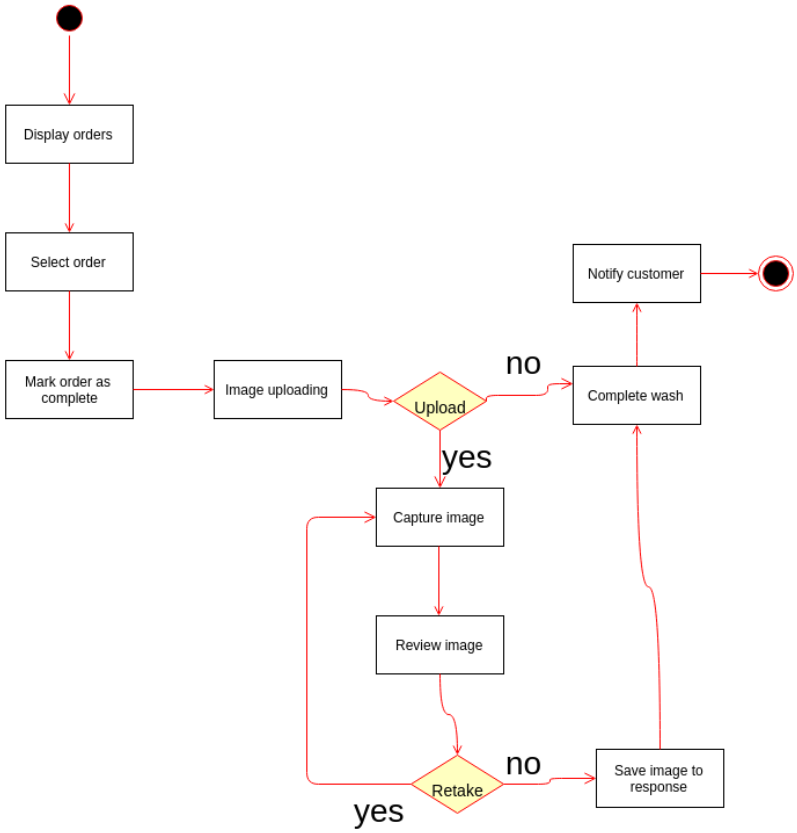

Figure 15. Activity diagram describing the process of a car washer marking a car wash as complete.

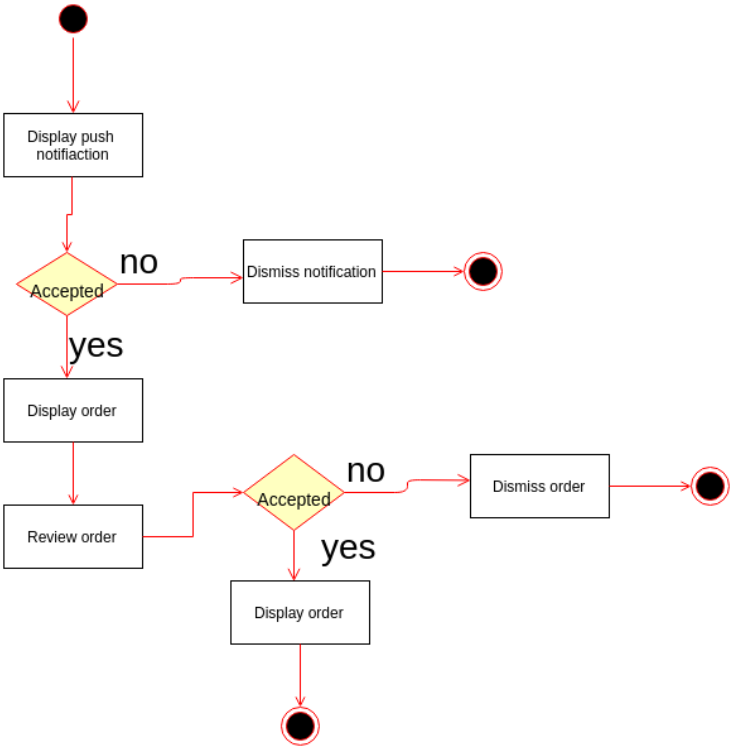

Figure 16. Activity diagram describing the process of a car washer receiving a new job notification.

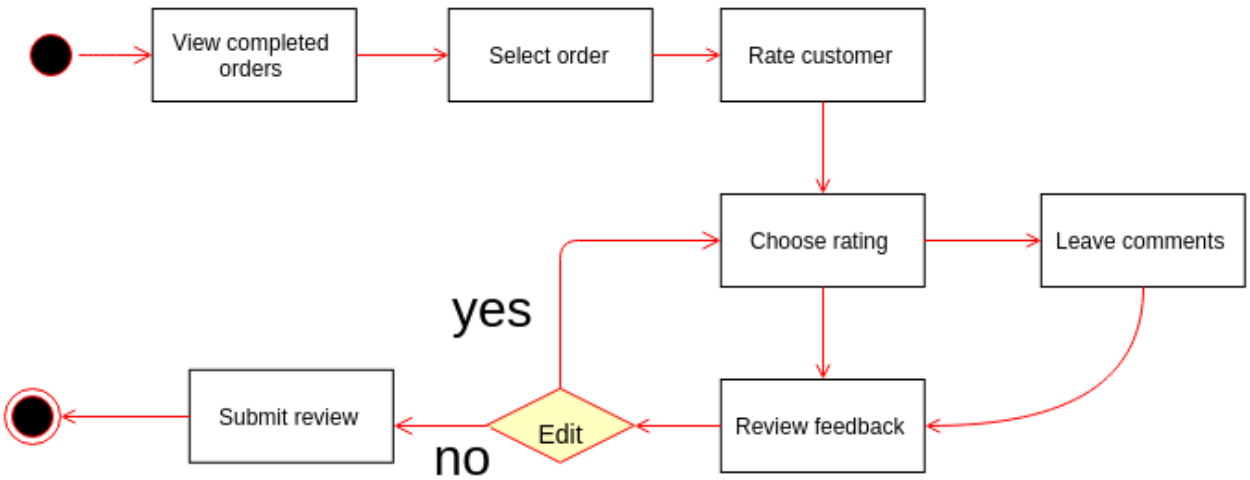

Figure 17. Activity diagram describing the process of a car washer reviewing a customer.

state machine diagrams

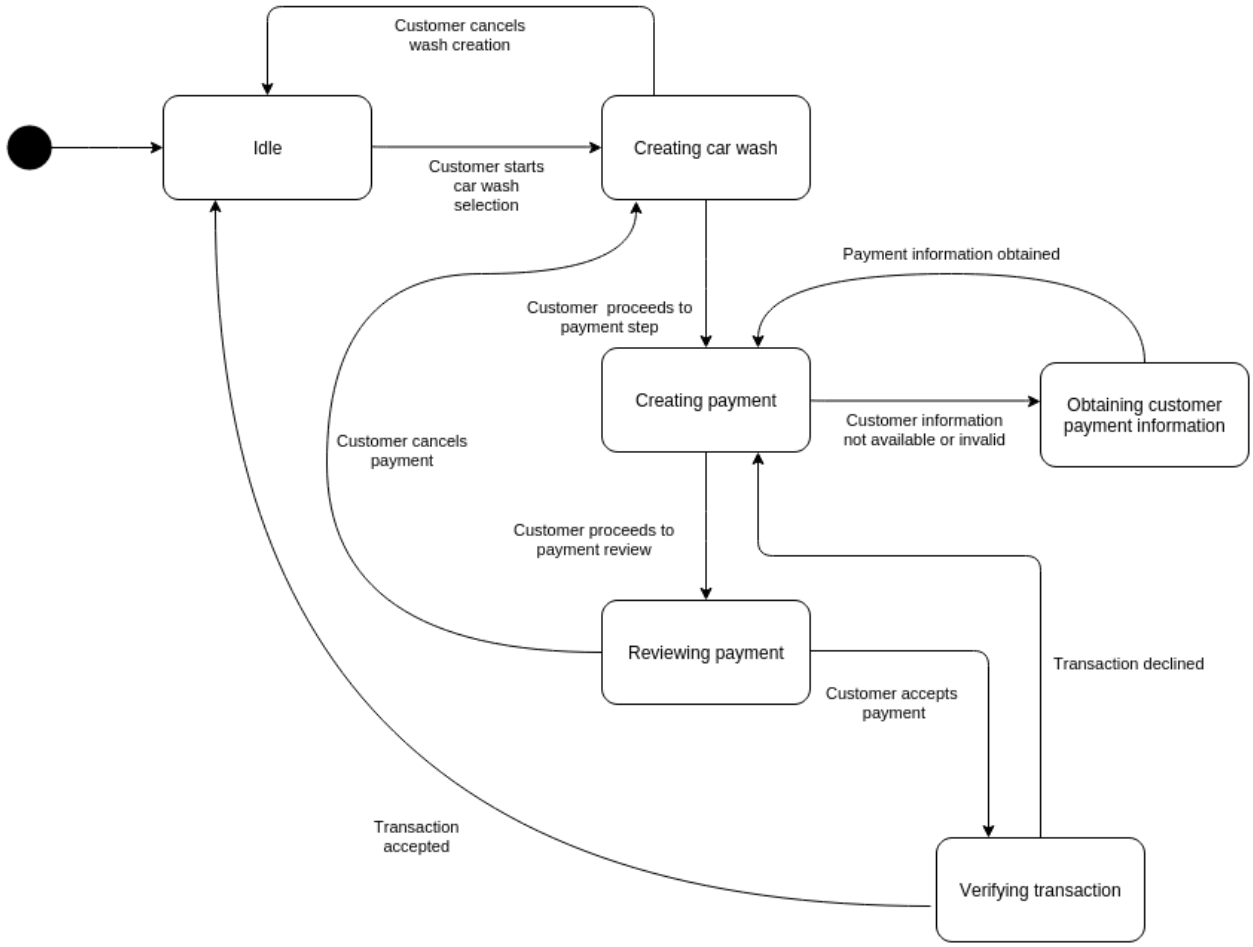

Figure 18. State machine diagram describing the customer car wash ordering process

Sequence diagrams

Car Owner Registration

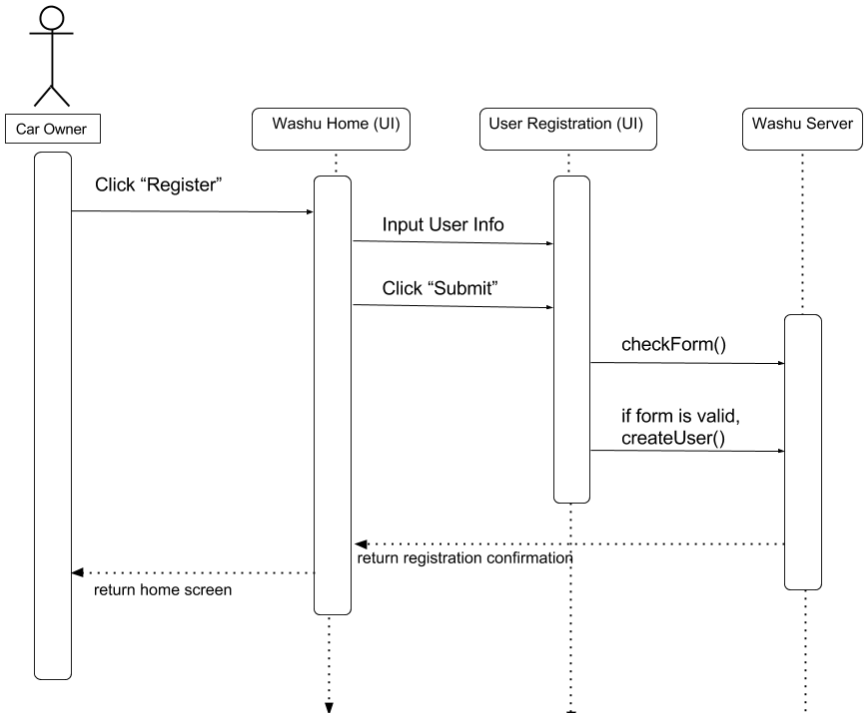

Figure 19. Sequence diagram describing the care owner registration process.

Car Washer Registration

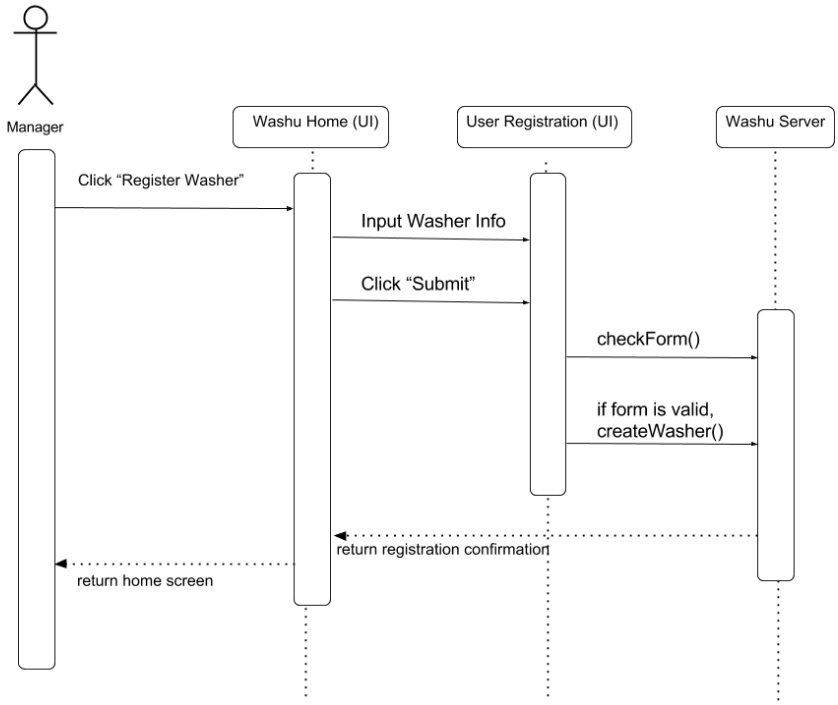

Figure 20. Sequence diagram describing the car washer registration process.

Car Wash Ordering

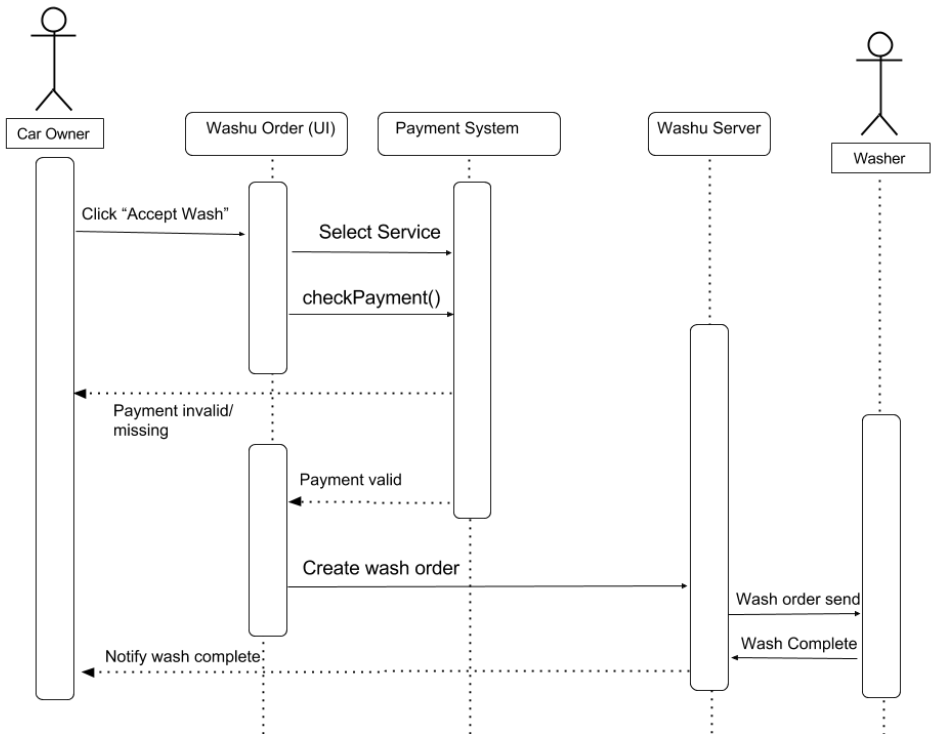

Figure 21. Sequence diagram describing the car wash ordering process in the Washu system.

Uploading Images

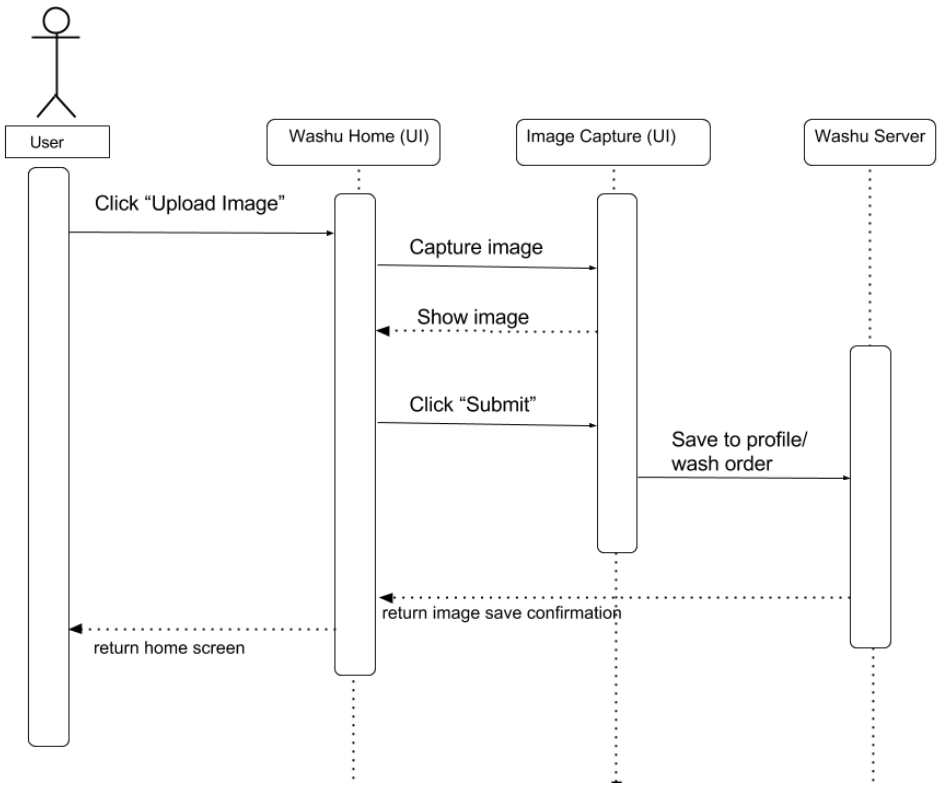

Figure 22. Sequence diagram describing process of a user capturing and uploading an image of their car to the Washu system.

Setting the Washu Wash Region

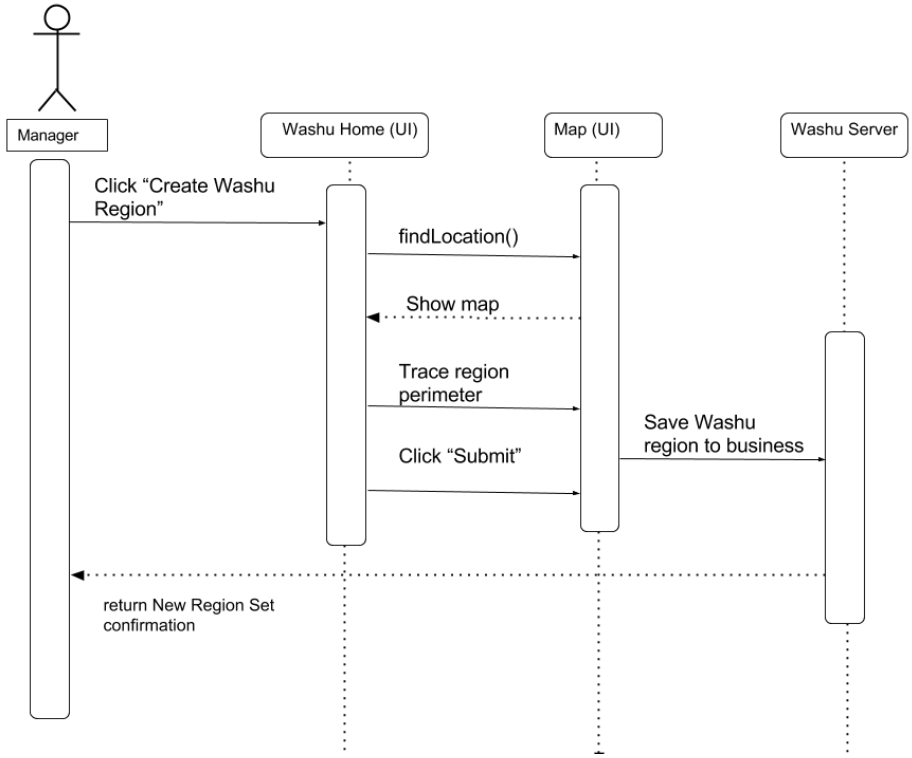

Figure 23. Sequence diagram describing process of a manager setting a Wahu car wash region for their organization.

Customer car wash rating submission

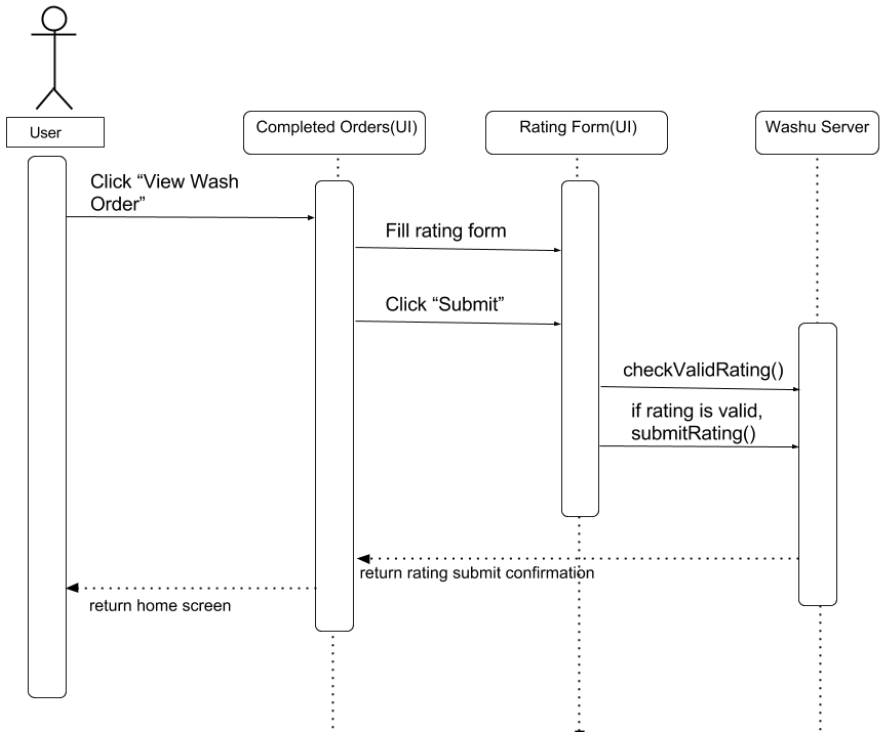

Figure 24. Sequence diagram describing the process of a customer reviewing a car wash that they received.

Class Diagrams

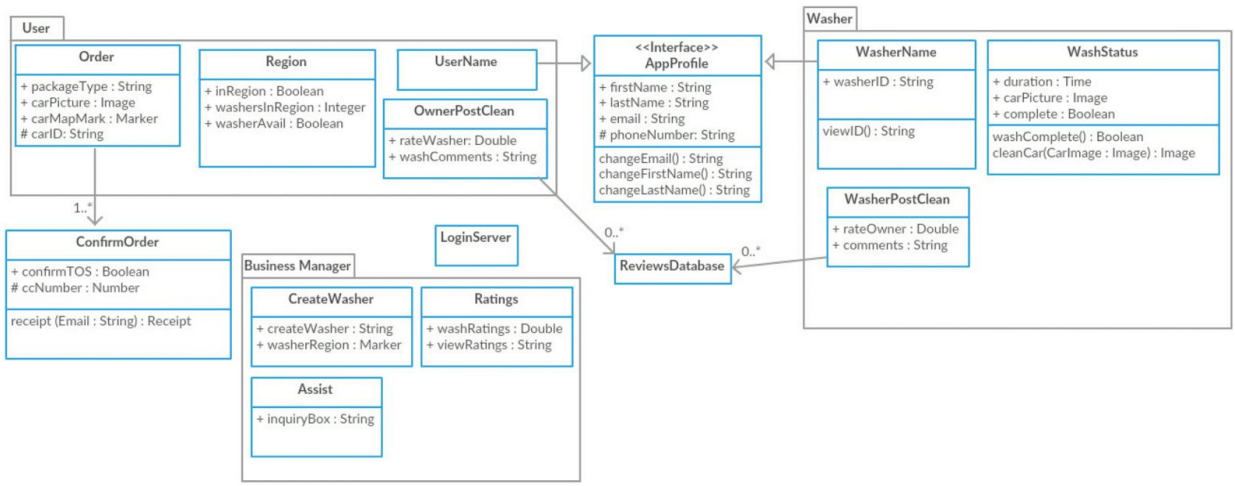

Figure 25. Class diagrams for the system.

Functional Specifications

TaxiCorp proposes Washu, a car wash application in response to Kraaken's Request For Proposal for an application developing team. Washu will be produced as an app for Android 4.0 Ice Cream Sandwich in order to reach the largest Android user base while providing the necessary functionality as specified by the client. The application will meet an underserved market made up of car owners.

The app will be licensed to malls and business owners where there is available parking for customers. The app will notify users parked at these Washu licensed businesses that a car wash service is available, and offer various types of washes to choose from while the customer is away on their errand.

The project will satisfy a need for increased convenience in today's fast moving, tech-driven world. It aims to remove the idea of a user having to travel to access a service; Washu instead joins the on-going trend of the service brought to the user. Car washes are a means of care for a piece of technology that can have important roles in people's lives. These roles can include accessing employment, education or recreation. Traditional car wash services require the user to either find their own time and use their own efforts, or to bring their vehicle to a dedicated car wash location. This imposes a convenience constraint for the car owner as the car is out of service for the duration of the wash. The Washu application allows users' cars to be washed when it is already planned to be sitting for an extended period of time. This maximizes convenience so the user can spend their time focusing on other important duties in their lives.

Washu also provides benefits to the licensee of the application. A business whose parking lot is licensed will draw customers from unlicensed competitors. It is an extra service that is provided and is supported 24/7 from Washu. A licensed business will then see increased revenue from the implemented system. Washu will fill a hole in the car washing market, and its beneficiaries will be those who seek a new, convenient car washing model to either access or implement.

The users of the application are car owners, Washu licensed car washers, and Washu licensed business managers. Car owners will be able to use the app to set up an account, order a car wash when notified that a Washu service is available, rate their wash, and

provide payment. Car washers will use the app to receive notifications that a wash is requested, rate the requesting customer, and mark a wash as completed. Business managers will use the app to set a region such as their parking lots where the car wash service will be available, create and assign car washer employees to car wash regions, view car wash ratings and access customer contact information to address comments or concerns.

The system's most important features include GPS triggered push notifications to let users know that a Washu car wash service is available in the area. GPS will also be used to let the customer mark the location of their car so washers can find and prioritize their washes. The app will allow users to create and customize their own profile that includes payment information and car model. Car washers can also create their own profiles, which customers can review. Users will be able to upload pictures of their car for identification, or to report a wash that does not meet satisfactory specifications. A secure payment system will be implemented so users can pay for their wash within the app.

Washu will be run on mobile applications that support Android 4.0 Ice Cream Sandwich. Hardware systems supporting Ice Cream Sandwich also support features the application requires such as notifications, GPS and camera access. The Washu application should have the above mentioned features but also meet required performance specifications. The payment system will be very secure, as not to leak any customer data regarding car ownership or payment information. Functionality such as GPS location will be available during business operating hours.

User Interaction

Customer/Car Owners

Set up customer profile account

When initially opening up the Washu app, users will be requested to create and customize their own user profile. This will be done by a form with multiple fields where users can enter information. Users will fill in a profile name, contact information, and upload an image or multiple images of their car. An asterisk would appear beside mandatory fields. Contact information options include email and phone number. The customer will also need to fill in payment information. Payment information does not need to be supplied until a car wash is

ordered. An option to save payment information such as a Stripe, interac, or credit card number will be supplied. The app would also be able to collect PayPal information to handle payment. A red border will appear around fields that are required but missing or not entered correctly eg. incorrect amount of digits for a phone number, missing “@” for an email, invalid credit card number.

Car Wash Ordering

Users will receive a push notification on their mobile device when they are in a parking lot region that supplies the Washu car wash service. Car wash zones will be determined through GPS tracking.

Upon opening the app after a notification has been received, the customer will be presented with a message requesting acceptance for a car wash. If the customer has not yet submitted payment information such as a credit card, a view will appear requesting this information be entered. The app will check for valid information before allowing the customer to proceed.

If the user decides to order a car wash, the user will see available car washers and available car wash services. The customer will be able to view different washers and read their ratings and reviews. Customers will also be able to go through different car wash services and see them in more detail. Detailed view will include the full extent of washing and their prices. Tapping on a washer will select them for a wash. Likewise, tapping on a wash service will place an order for that particular wash service.

Once a car wash service and washer are chosen, the customer will be asked to mark the location of their car within the app. The application will use GPS for marking. The customer will be presented with a view of the available car wash region and be able to tap on it to mark their car location. The customer would also be asked to select the vehicle they would like to wash in the case where they have more than one car linked to their profile. This will allow car washers to locate the customer’s vehicle once the order is sent. Before the car wash order is finalized, the customer will be able to review their order to ensure their selection is correct.

Payment

The customer will receive a request for payment once a the car wash order is finalized. The customer will be presented with a view to select one of the payment options they have set up

for their profile. The customer will select their payment method by tapping on the option to highlight it, and then submit the payment by tapping on a “Confirm Payment” button. If a payment option is not selected, an error message will display noting that a payment method is not selected. Once a payment is successfully submitted, the customer will be displayed a message stating the transaction was successful and thanking them for their business. Successfull payment will result in a notification sent to a car washer to notify them that a wash has been ordered.

Rate Car Wash

Once a car washer has marked the wash as complete, the user will receive a notification that their car is ready for review. Opening the app after receiving a car wash complete notification will present the customer with a view to rate and comment their wash. The customer will review the completed car wash in person and rate the wash with a 5 star rating system. A rating of 5 will let the business and washer know that the service was exceptional, while a rating of 1 will express an unsatisfactory wash. A star rating is required, while a comment is optional. The view will be a form with fields for a rating, a comment, and a submission button. Submission will not be sent until a rating is chosen. Tapping on “Submit” with a void rating field will result in a red border around the rating field and a notification that a rating is required.

If a rating of less than 3 is received, the customer would have the option to submit a picture of the car to indicate the terrible quality. The customer will tap on “take a photo” which will open the device’s camera. After a photo is taken, the customer can either retake or choose the photo for submission. The photo will then be uploaded to the order review for the car washer and business manager to see.

Customer Support

The app will allow customers to request customer support. The various options for customer support will include contact through email, contact through phone or a live chat session. Selecting the live chat option will result in the customer seeing a slide-up window for an open chat session. If a customer service representative is not present, the chat window will say “No customer service representative available; Please try again later or use email contact.” It will then show the available time slots for a live chat session.

Washu Car Washers

Receive Wash Request Notification

Washu Car washers will receive a push notification on their version of the mobile app once a customer requests a car wash. A notification will be sent to the customer chosen washer designated to the parking lot where the customer has parked. Upon opening the app, the washer will be presented with a view of the order information, along with the customer information. The car washer will be able to view the customer's ratings and reviews in order to determine at their discretion whether they will want to take on the car wash job or not. Buttons for "Accept order" or Decline order" will be presented, and the washer will make their selection by tapping on one.

If the requested car wash is declined, a notification will be sent to the customer. The customer will then be required to request a wash from another available washer.

If a requested car wash is accepted, the washer will be shown the location and a picture of the vehicle to be washed and the kind of wash service requested. The washer can then prepare wash materials and schedule the wash according to priority determined by available time, estimated wash duration and location of the vehicle.

Mark Car Wash As Complete

Once the washer has completed the car wash, they can mark the wash as complete within the app and a notification will be sent to the customer. The washer will open the app, select the order, and tap on the button "Mark as Complete." The washer will then receive a request to upload an image of the vehicle. The washer can take an image of the newly washed car to be sent with the completion notification to the customer. The app will request to use the mobile device's camera. Once an image is taken, the car owner can confirm the image or retake another one. Once an image is confirmed, it will be uploaded to the Washu app and sent to the customer to view.

Car owner rating

Car washers will be able to rate a customer after a car wash has been completed. A 5 star rating system will be used. A rating of 5 stars will let other washers know that this customer does not pose any problems. A rating of 1 star will be used in cases where the customer

treats the washer poorly or is dishonest in profile or payment information and causes service disruption. The car washer will be able to open a past order by tapping on it after it has been marked complete and click on a “Rate Customer” button. The washer will be presented with a form with fields to choose a rating, and leave a comment. The comment field will be optional. After the customer review has been filled out, the washer will tap on the “Submit” button. If a rating has not been chosen, a message will be displayed noting that a rating is missing and is required before a submission can be sent. After successful submission, the washer will be displayed a message saying “Customer rating submitted.”

Receive Rating Information

A push notification will be sent to the washer once a customer has paid and submitted a rating. The car washer will then be able to view the rating submission in their view of completed washes. The washer can view an order rating by clicking on a completed order. The washer will then see a view of the order information, such as time of wash, wash duration, customer information and any uploaded images. The customer rating of 1-5 stars will be displayed, along with a comment if the customer submitted one.

Business Managers

Set Car Wash Region

The business manager using the Washu application will be able to set up regions where the car wash service will be available. The manager will be presented with an button “Add car wash region.” Clicking on the button will result in the application displaying a map of the business location determined by its address. The manager will then be able to mark on the map the regions where customers will be able to receive car washes. The manager will draw a continuous line enclosing the area on the map. A line will be traced on the map for visual feedback. If an enclosed region is not properly marked, an error message will display stating “An enclosed region was not traced. Please try again.” After marking, the manager will see a message with buttons for “Save Car Wash Region” or “Cancel.” Tapping on “Save Car Wash Region” will save the region to the manager’s account. Tapping on “Cancel” will return the manager to the main Washu screen. When a customer with the Washu app enters a marked region, they will receive a notification on their mobile device that the Washu car wash service is available.

Create Employee Accounts

The business manager will be able to create employee accounts. These employees will later be assigned to regions where the car wash service is available. A new employee is created by tapping on “Create Car Washer.” The manager can then input the employee profile information in their respective fields such as name and contact information. After the required fields are filled, the manager will click on a “Save Employee” button to save the employee. If a required field is not filled, an error message will be displayed before the employee account is created and stored with Washu.

Assign Car Wash Employees

The manager will be able to assign an employee a car wash region once an employee profile is created. The manager will select an employee, and then select “Assign Car Wash Region.” The manager will be able to see the available car wash regions, and which employees are already assigned to each region. The manager can then choose a region and the employee will be assigned to work in that region. This information will be updated for the car washer on their view of the Washu app.

View Customer Review

Customer reviews will be available for the business manager to access. This will be done by opening up the list of employees, and then tapping on “Reviews.” The manager will then be able to see a list of ratings and comments that customers have left for that particular car washer. This will allow the business manager to ensure that their service is being held to customer satisfaction. Alternatively, the manager can open the list of completed orders, and see complete order information including time and date of wash, and customer and washer information including their ratings and reviews.

Contact Customer

The business manager will be able to contact a customer who has ordered a wash through the app. This will be done by accessing the reviews and ratings in the completed orders view, and tapping on “Contact Customer” for a particular review. The manager will then be able to see the customer’s email and phone number if provided, and an option to open a live chat messaging system through the app. The manager will be able to send an email through the app by filling in a form. Once an email form is completed and sent, the customer will receive a notification that a message from the business manager has been received. If a live

chat is initiated, the customer will receive a notification that a Washu live chat has been requested. The customer will then be able to open the app and accept the live chat request.

Management Plan

Breakdown of Different Features

The entire system will consist of several features:

- Location detection and location-based prompting:
    - The mobile app will detect, based on the user's GPS location, whether or not the user is parked at an equipped business. If so, the app will send emit a push notification on the user's mobile device that will advertise the availability of car washes at their location. Interacting with the push notification will allow the user to purchase a car wash within the app.
- Wash type selection:
    - The mobile app will display the types of car wash available at their location, and allow the user to select a type of car wash.
    - The user will be able to view and select a wash type regardless of whether or not they were prompted for a car wash by the app.
- Account creation and payment information association and persistence:
    - The system will require the user to create an account through the mobile app in order to use the service.
    - The user's account will store their payment information for use in subsequent car wash transactions.
    - The user will remain logged into the mobile app after account creation or logging in unless they explicitly log out.
- Automobile pictures and description:
    - The mobile app will allow users to upload a picture of their car and license plate, which can be associated with their account for use later.
    - The mobile app will prompt the user to take a picture through the camera's phone upon purchase of a car wash if the user does not already have an associated picture.
- Provide a separate interfaces for each role:
    - for car washers, providing washers with the type of wash, car and license plate pictures.

- Car washer (employee) interface:
    - The mobile app will provide a separate interface only available to users who are car wash employees.
    - The employee interface will provide the user with the details of the next car in line to be washed, and the ability to indicate that a car wash has been finished.
    - Car washers can rate customers.
- Reviews
    - Customers will be able to submit and view reviews from other customers.
    - Car washers will be able to view feedback from customers.
- Automatic Completion notification
    - The app will automatically send the user a push notification on their mobile device when their car wash has been completed.
- Management interface
    - Managers will be able to
        - Manage car washers,
        - View customer reviews,
        - Contact customers.

Minimum Viable Product

An expected minimal system to be completed by the end of the term will be a working prototype. This includes a functional Android application for the Ice Cream Sandwich OS that can be run and have basic features and a basic user interface. The prototype will have enough functionality to set up profiles for both customers and car washers, and send notifications between them.

This minimal system will have the car washer receive notifications that a customer has requested a car wash. This requires

This system will also have the customer receive notifications that a car wash has been completed by a washer.

If system implementation goes well, the application will have enhancements such as notifications based on GPS, and the option to access the device's camera to take a picture and upload it to the application.

Functionality that is not expected to be implemented at the end of term but will be nice to include are a rating system and a payment system.

Team structure and responsibilities

Chris Kelly	- Project lead
Adewale Adekoya	- Webmaster/Developer
Richard Lui	- Washu App Designer and Programmer
Trison Nguyen	- Document writer

Glossary of Terms

GPS	- Global positioning system; a space based positioning system that provides location and time information in all weather conditions
App	- A mobile phone application
OS	- An abbreviation for Operating System
Prototype	- An early sample, model or release of a product built to test a concept
Push notification	- A notification that appears on the home or lock screen of a mobile device, used to notify the user of new events when the user is not actively using the notifying application.
JUnit	- A testing framework written in the Java programming language, allowing developers to write programmatic tests for Java applications, including Android applications.