



Course Name: Software Engineering

*

Course Number and Section: 14:332:452:01

Group #10

Drip 'n Dash

First Report

*

Website: <https://sites.google.com/view/dashndrip/dash-n-drip>

Group Members:

Elaina Heraty, Abdullah Bashir, Harshavardhana Dantuluri, HyunSik Kim, Karan Parab, Liam McCluskey, Nayaab Chogle, Nicolas Rubert, Roberto Cruz, Siddharth Manchiraju

Subgroup	Member	Contribution
1	Liam McCluskey	10%
2	HyunSik Kim	10%
3	Nicolas Rubert	10%
4	Roberto Cruz	10%
5	Nayaab Chogle	10%
6	Siddharth Manchiraju	10%
7	Elaina Heraty	10%
8	Harshavardhana Dantuluri	10%
9	Abdullah Bashir	10%
10	Karan Parab	10%

Table of Contents

Customer Statement of Requirements	3
Problem Statement	3
Glossary of Terms	5
System Requirements	6
Enumerated Functional Requirements	6
Enumerated Non-Functional Requirements	7
User Interface Requirements	8
Functional Requirements Specification	11
Stakeholders	11
Actors and Goals	11
Use Cases	12
Casual Description	12
Use Case Diagram	13
Traceability Matrix	14
Fully-Dressed Description	14
System Sequence Diagram	17
User Interface Specification	18
Preliminary Design	18
User Effort Estimation	21
Domain Analysis	23
Domain Model	23
Concept Definitions	23
Association Definitions	25
Attribute Definitions	26
Traceability Matrix	28
System Operation Contracts	28
Mathematical Model	29
Project Operation Size	29
Plan of Work	31
References	33

1. Customer Statement of Requirements

a. Problem Statement

For most students, college life entails a significant amount of hard work, along with a healthy amount of time for extracurriculars. This allocation of a large portion of each student's day means that most students may not find time to complete simple tasks, let alone their laundry. Laundry is a necessary chore for all students to perform, but with assignments, club events, and social gatherings scheduled almost every week, it is relatively hard to find the time. After all, not all students can return home and have their parents do their laundry for them. However, even if a student were to have time to do their own laundry, they may occasionally find themselves not feeling up to the challenge of partaking in a chore as tiresome as taking care of their laundry and would much rather have someone else do it for them. What may seem to be an everyday and simple chore, could turn out to be a problem for some individuals.

With the application that we are proposing, Drip n' Dash, we will provide a quality laundry service to all dorming students at Rutgers University.

- Customer Side Problems:

In addition, not being able to do laundry can be detrimental to a student in a variety of ways. Having unwashed clothes can lead to a student falling ill, which would hinder their ability to succeed in school. With this in mind, having a helping hand every once in a while to assist with this task would be extremely beneficial. This method would make it possible for them to request and pay someone to do their laundry for them.

Additionally, there are students who do not know how to properly do laundry, potentially making this a daunting task in order to get clean clothes. On the contrary, doing laundry is a very intricate process that if not done properly, it could cause an enormous mess. There are many things to take into account when doing laundry, those of which include the amount of laundry detergent used, the temperature of the water, the type of material of the clothing article, and many other variables that an inexperienced person would overlook. It is through using the app that the same student would not have to worry about any repercussions they could bring about by not knowing how to do laundry.

However, the students who do have plenty of spare time could benefit from this service, turning this utility into a commodity for some extra money. Sometimes a student might just want someone else to do the laundry for them. There are times in which a person would rather do another activity with their free time- be it more studying, spending time with friends, or even relaxing- instead of having to do laundry. For this reason, these students would still be able to make full use of the app whenever they desired.

Suggestions from an Interested Customer:

As a student in University, I do not have the time to do laundry with my intensive schedule. Although, I am unwilling to give my valuable clothing items to strangers to do my laundry for me. Services such as Facebook, Craigslist and TaskRabbit are high risk because I do not know the person and fear for the safety of myself and my clothes. If I were to trust any person to my laundry, I would need to know that they are a law abiding citizen and knowledgeable in cleaning clothes. Since I am living on a college campus, I would prefer that the person is also a student and lives in the same dorm building as I do. This will eliminate outsiders from entering a student residential hall and keep the worker accountable with the Office of Student Affairs. In addition, I would want the same gender to be doing my laundry.

Furthermore, since there will be some communication between me and the customer, I would want all my contact information to be hidden. This includes my last name and personal phone number, similar to what Uber does. If there must be some communication between me and the worker, it should be done in the app itself.

If there are any actions that are against the policies of the app, I should have an easy and fast way to file a complaint and/or contact a customer service member. This includes issues with how my clothing was handled or if the requirements of my laundry was completed correctly and any other issues with the app. I would also like to have a review system where I can post a review and see past reviews of a worker. This will make me feel much more comfortable when handing my clothes to someone I don't know.

- Dasher Side Problems:

Although many students dislike doing laundry, there are students that don't mind it and would like to make some extra money by doing other peoples laundry. For these students, whom we call dashers, the main problems consist of needing an easy source of revenue, and that many part time jobs have a comparably high barrier of entry.

The expenses of college students can add up quickly, leaving them in need of an easy source of revenue to fund all these costs. However, finding a convenient job on campus can be

difficult, since some positions require a specialized set of skills, or cannot accommodate a student's busy class schedule. This is where Drip 'n Dash comes in; Unlike most other jobs that often require a long tedious interview process, Drip 'n Dash only needs students to pass a quiz about how to properly do laundry to register as a dasher. After becoming a Dasher, that student is able to accept jobs on their own free time.

b. Glossary of Terms

Arrival Time: Estimated time that the customer communicates to the Dasher in which they would like for their laundry to be picked up from their dorm.

Dasher: A user who chooses to provide their service for customers - this user would perform the laundry.

Customer: A user of Drip 'n Dash who sets up a job for the Dasher - this user requests to have their laundry done.

Customer Rating: A score a dasher can set for the client and vice versa, within the range of zero to five stars. This rating is based on factors such as: whether or not the customer communicated having provided garments that require extra care, whether or not the customer was on time with respect to the arranged pickup and drop off time of the laundry, overall attitude towards the Dasher, etc.

Dasher Rating: a score within the range of zero to five stars of which a customer can assign to a dasher after having completed a transaction with that Dasher. This rating should be based on factors such as: the quality of the service provided by the dasher, whether or not the dasher was on time with respect to the arranged pickup and drop off time of the laundry, the customer service skills of the Dasher, etc.

Drip 'n Dash: a mobile application that provides a means for an on-demand, peer-to-peer laundry service among Rutgers students.

Drop off Time: estimated time that the Dasher communicates to the customer in which their laundry is completed and will be dropped off to their dorm.

2. System Requirements

a. Enumerated Functional Requirements

REQ #	PRIORITY	DESCRIPTION
0	5	As a Rutgers student that lives in a dorm with public washers and dryers, I can register for Drip 'n Dash to be a customer or a dasher using a university email address
1	5	As a customer, I can post a request for a dasher to do my laundry
2	5	As a customer, I can specify the time that the dasher will pickup and drop off my laundry
3	3	As a customer, I can specify conditions for which dashers can see my request for laundry (example: female customers may prefer that female dashers handle their laundry)
4	5	<p>As a dasher, I can see all the pending requests for laundry that customers in my dorm have posted (assuming I meet the conditions the customer placed for who can see his/her request for laundry, see REQ-3)</p> <p>(The above approach might be switched with an auto assign system of laundry jobs to prevent the issue of multiple dashers selecting the same job simultaneously)</p>
5	5	As a dasher, I can accept a customer's pending request for laundry if I am able to meet the specified pickup and dropoff times
6	4	As a customer, I can see a dasher's profile (containing information about the dasher's detailed reviews, star rating, number of completed transactions, photo, etc.) once he/she accepts my request, and choose to confirm or deny the request to do my laundry
7	5	As a dasher, I can pick up and drop off a customer's laundry according to the specified times if the customer confirmed my request to do his/her laundry. Information about the customer's dorm room number (the pickup and dropoff location) will be visible only once he/she confirms my request

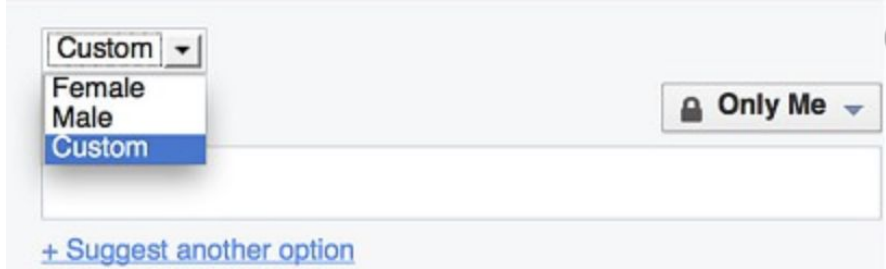
8	4	As a customer, I can leave a detailed review and star rating for a dasher after completing a transaction with him/her
9	3	As a dasher, I can leave a star rating for a customer after completing a transaction with him/her
10	4	As a customer I can do a cashless transaction using a cashless service such as venmo, google pay, apple pay etc.
11	3	As a customer, I can see basic information about the status of my in progress request for laundry. This includes information about when the dasher is outside to pick up my laundry, when my laundry is in the wash, in the dryer, etc.
12	3	As a dasher, I can notify the customer with status updates about basic progress information about their request for laundry (see REQ-11)
13	4	As a dasher, I can simultaneously complete multiple customer's requests for laundry and see a list view of my in progress jobs (restrictions will be placed on when they can accept another job)
14	3	As a dasher, I have to successfully complete a basic quiz about how to do laundry in order to register as a dasher
15	4	As a customer, I can dispute a transaction if the dasher did not successfully complete my request for laundry (stolen/damaged items, etc.)

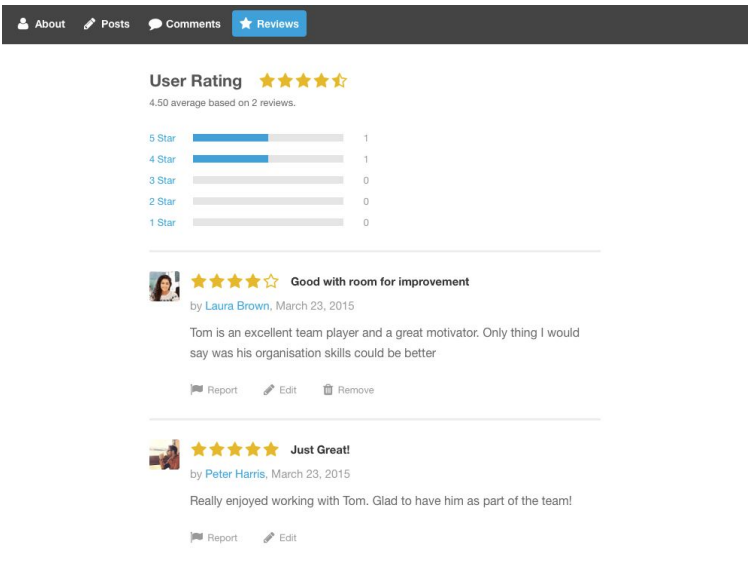
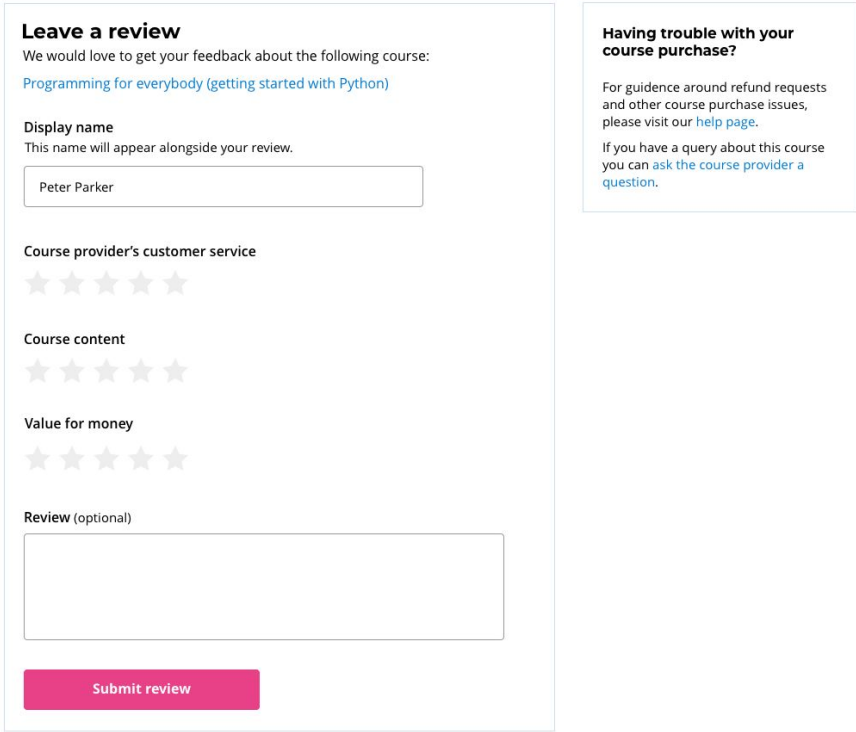
b. Enumerated Non-Functional Requirements

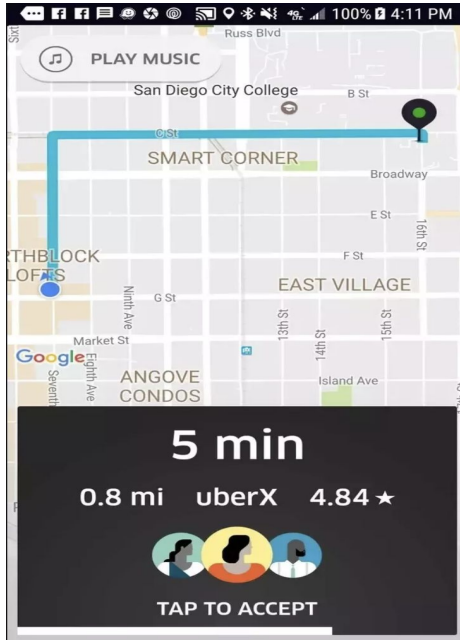
REQ #	PRIORITY	DESCRIPTION
16	4	As a system, improve user usability to be as simple as possible for both dashers and users
17	5	As a system, maintain the privacy/confidentiality of certain information for both the customer and dasher
18	4	As a system, app optimization is needed to increase efficiency of the system and decrease latency overall.
19	5	Maintain accuracy with dorm/machine locations
20	3	Improve the maximum capabilities of the application while meeting

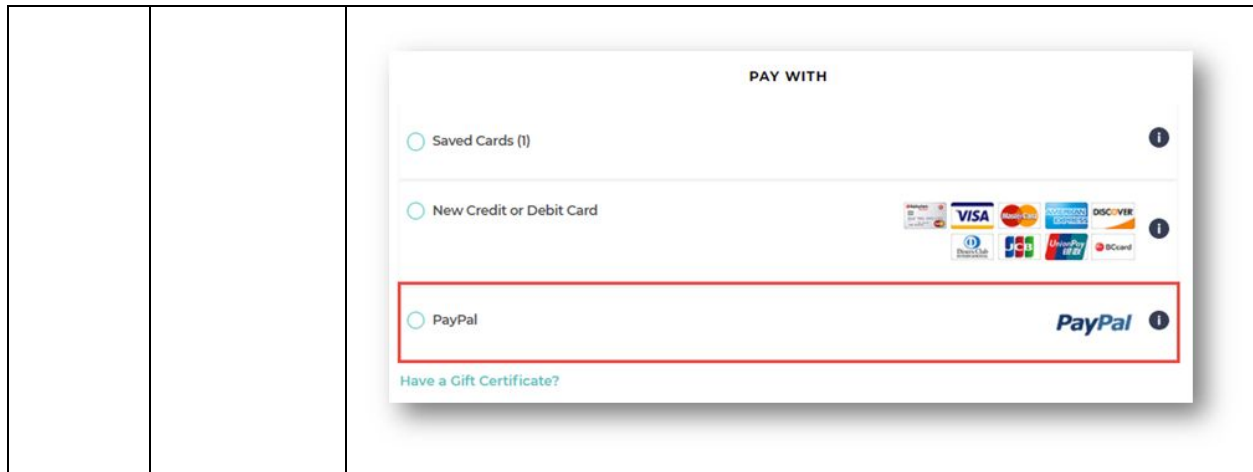
		the system requirements
21	3	Update/Maintain reviews for dashers to improve usability
22	2	System should be load balanced
23	5	As a system, the application must be accessible to most of today's Android OS kernels
24	3	As a system, backups of data should be taken regularly
25	5	As a system, quiz for dashers need to be accessible and usable
26	3	The system should have an easy and usable UX for users
27	5	The system should handle numbers of users without consuming too much resources
28	4	The system has to notify the policy of the app to the user
29	4	The system has to be easily maintainable so developers will be able to fix bugs

c. User Interface Requirements

REQ #	PRIORITY	DESCRIPTION
0	5	System needs to be able to access a list of available dashers in the users vicinity
1	5	<p>The system should have a page with a filter where user can specify which gender they would prefer do their laundry</p> 
2	5	System requires a page that shows how many stars/approval rating designated dasher has

		
3	3	<p>System also needs a page where the user can leave a review about how their laundry was done which would update the dasher's approval rating accordingly</p> 
4	5	<p>The system should present a page, after the user selects a dasher, which shows any additional services (ironing, folding) users would like to have done for an extra cost.</p>

		<p>Would you like any additonal services?</p> <p><input checked="" type="radio"/> Folding \$2.50</p> <p><input type="radio"/> Ironing \$5.00</p>
5	5	<p>System must have a page where dasher can accept a customer's pending request for laundry if they can meet the specified pickup and drop off times</p> 
6	4	<p>After all options and services are selected, the system must present a drop off/pick up location that is convenient for both user and dasher</p>
7	5	<p>Following the conclusion of a laundry service, the system should show a page that has payment options (venmo, credit card etc)</p>



3. Functional Requirements Specification

a. Stakeholders

- Customers
- Dashers
- Dorm Resident
- Rutgers
- Speed Queen- creators of the laundry app used for Dorm laundromats
- Any other colleges that would want to utilise this app
- Any sponsors that are willing to run ads

b. Actors and Goals

- Customer- Initiating: To be able to log in, and request the services provided by the application as they desire.
- Dasher- Initiating: To be able to log in and take jobs that have been requested. Take jobs that fit the request criteria.

- Job System- Participating: To provide service between Customers and Dashers so that they can get their needs facilitated. It also updates whenever a job is assigned to a Dasher. Additionally, allows the Dasher to finalize the job.
- Progress bar- Participating: To provide a form of communication between Actors in order to present the status of the requested job.
- Rating System- Participating: To present Customer with the quality of the Dasher. Alternatively, present the Dasher with how the Customer is.
- Database- Participating: To store Actors data such as name, email address, password, campus, dorm, room number, and various other factors.

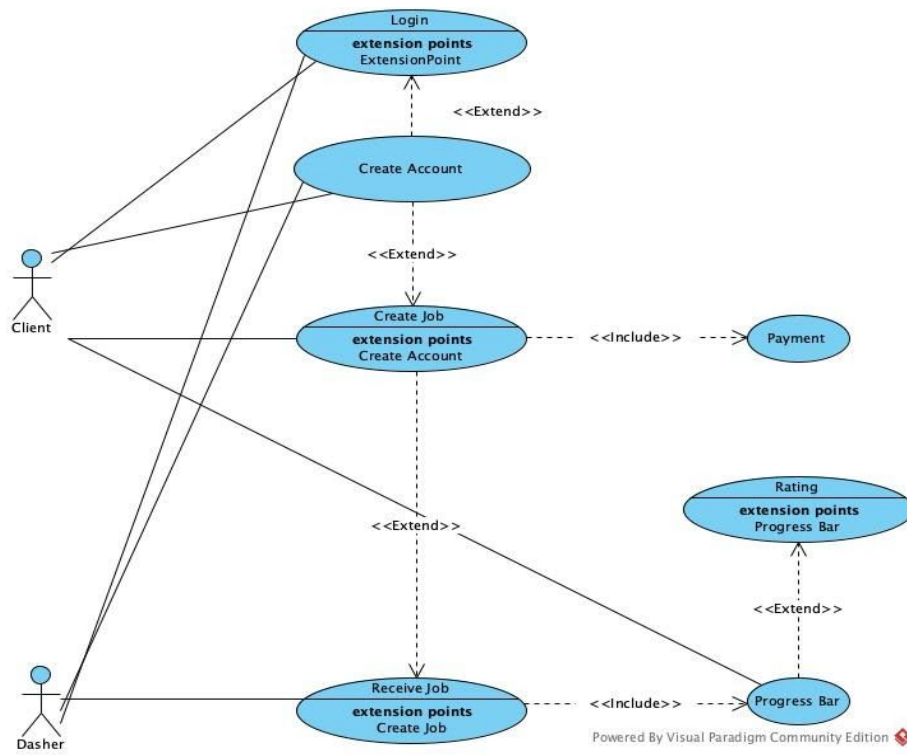
c. Use Cases

i. Casual Description

Use Cases	System Requirements	Description
UC-1 User Create/Login	<ul style="list-style-type: none"> ● Authentication System 	User creation and login to be either a client or dasher
UC-2 Job Creation	<ul style="list-style-type: none"> ● Authentication System ● Job Posting Module 	Clients can create a job with drop off time and pickup location.
UC-3 Payment	<ul style="list-style-type: none"> ● Authentication System ● Payment Module ● Venmo/Cashless Service module 	The client will pay for the job via venmo or another cashless service module
UC-4 Job Acception	<ul style="list-style-type: none"> ● Authentication System ● Job Posting Module 	Once the client posts the job with the info. The dasher is assigned to a job where they can accept the job
UC-5	<ul style="list-style-type: none"> ● Authentication 	The dasher posts updates via

Job Updates	<ul style="list-style-type: none"> System Job Posting Module Progress Module 	a progress bar which informs the user on the status of their laundry
UC-6 Job Completion	<ul style="list-style-type: none"> Authentication System Job Posting Module Progress Module 	The dasher upon completion will inform the client so that a drop off/ pickup of the laundry can be conducted
UC-7 Rating Dasher	<ul style="list-style-type: none"> Authentication System Rating Module 	Upon receiving the laundry the client can rate the dasher which then will appear in the dasher's profile
UC-8 Rating Client	<ul style="list-style-type: none"> Authentication System Rating Module 	Upon dropping off the clients laundry the dasher can then rate the client which will appear in the client's module

ii. Use Case Diagram



iii. Traceability Matrix

	REQ0	REQ1	REQ2	REQ3	REQ4	REQ5	REQ6	REQ7	REQ8	REQ9	REQ10	REQ11	REQ12	REQ13	REQ14	REQ15	TOTAL
PW	5	5	5	3	5	5	4	5	4	3	4	3	3	4	3	4	
UC-1	X														X		8
UC-2		X	X	X													13
UC-3											X						4
UC-4					X	X		X						X			19
UC-5							X					X	X				10
UC-6													X				3
UC-7									X							X	8
UC-8										X							3

iv. Fully-Dressed Description

UC: CustomerLogin

- Actor: user
- Preconditions: the user is registered as a customer
- Postconditions: the user is prompted to the **CustomerHomePage**
- Flow of Events for Main Success
 - 1. User enters his/her information into the text fields on **SignInPage**
 - 2. User presses the log in button
 - 3. If the information entered is correct, the user will be prompted to the **CustomerHomePage**
 - 4. If the information entered is wrong, the user will be told to try again on **SignInPage**

UC: CustomerRegister

- Actor: user

- Preconditions: the user is not registered for our application
- Postconditions: the user registers for our application and is prompted to the **CustomerHomePage**
- Flow of Events for Main Success
 - 1 . User clicks the register button on the **SignInPage**
 - 2 . The user is prompted to the **RegisterInformationPage** where he/she can see information about customers/dashers
 - 3 . The user presses the register as a customer button and is prompted to the **CustomerRegisterPage** where he/she can input the information required and press the confirm registration button.
 - 4 . If the information entered is valid, he/she will be prompted to the **CustomerHomePage**

UC: Dasher Register

- Actor: user
- Preconditions: user is not registered for our application, or user has only registered a regular customer
- Postconditions: user registers for our application and is prompted to the **CustomerHomePage**
- Flow of events for main success:
 - 1 . User clicks the register button on the **SignInPage**
 - 2 . The user is prompted to the **RegisterInformationPage** where he/she can see information about customers/dashers
 - 3 . The user presses the register as customer button and is prompted to the **DasherRegisterPage**
 - 4 . If the information entered is valid, he/she will be prompted to the **DasherHomePage**

UC: CustomerSubmitRequest

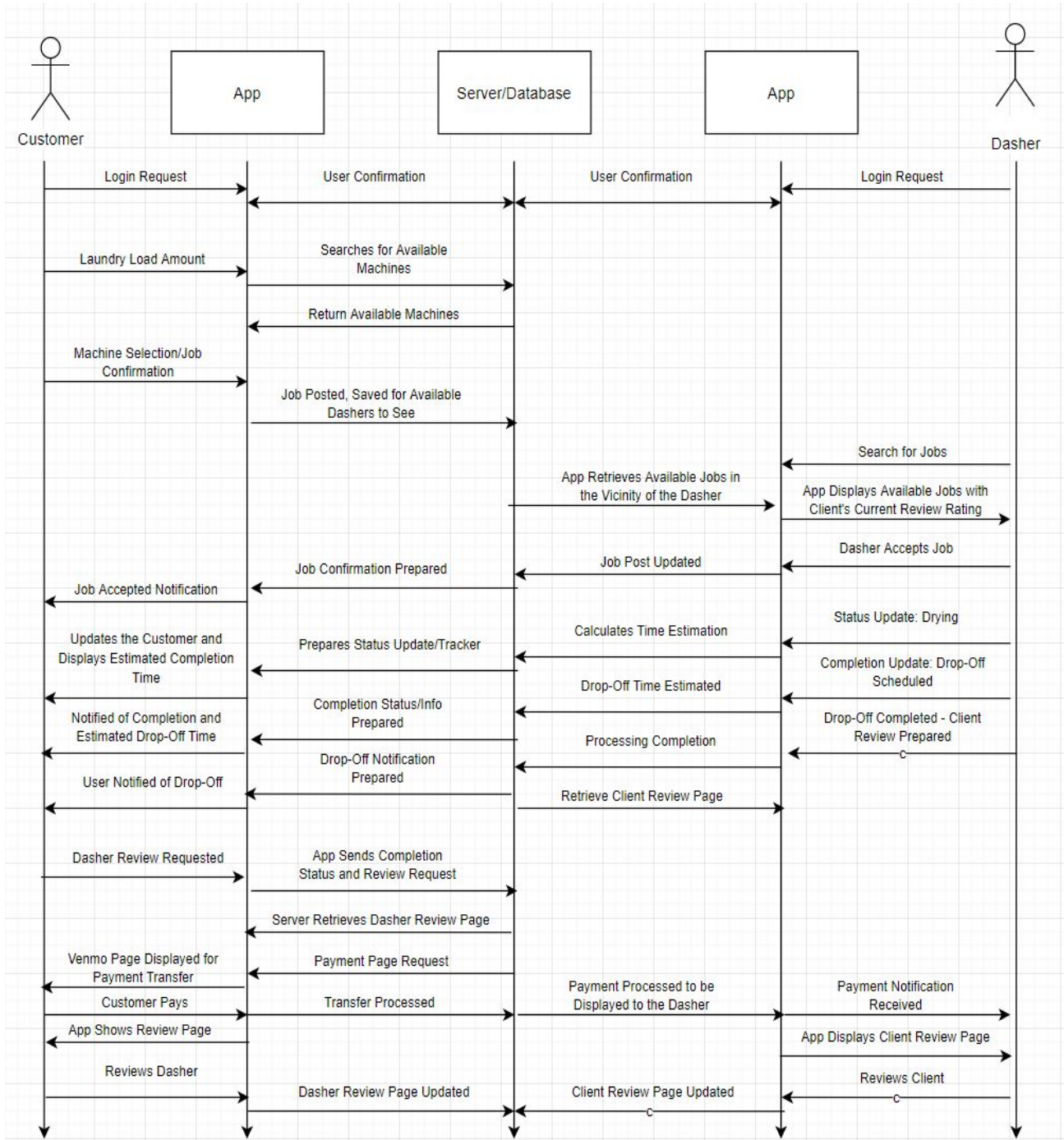
- Actor: customer
- Preconditions: the customer wants to post a request for laundry pick up
- Postconditions: customer's request is added to the database for dasher's to accept
- Flow of events for main success:
 - 1 . User enters information about the request and presses the submit request button on the **CustomerHomePage**

- 2 . Customer's request for laundry is added to the database and the in progress jobs section on the **CustomerHomePage** is updated to contain the request the customer submitted

UC: Dasher Accept Job

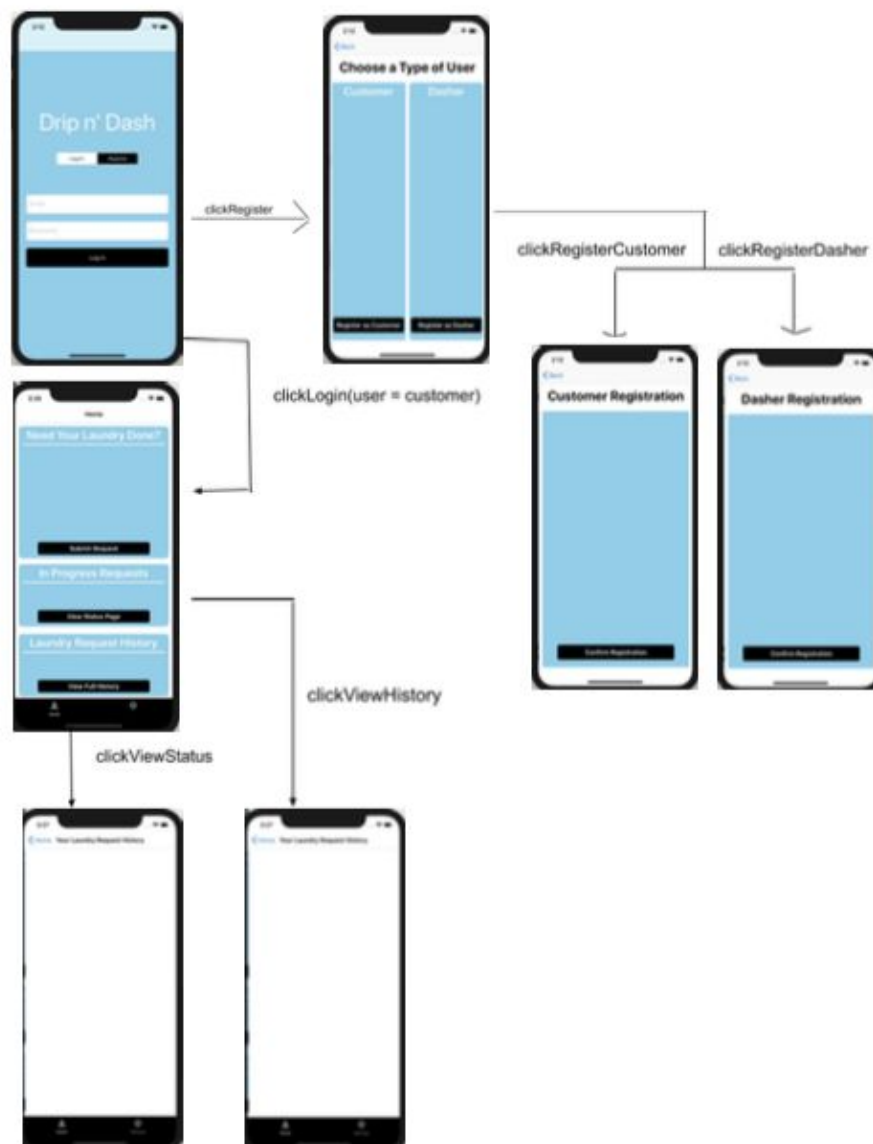
- Actor: Dasher
- Preconditions: dasher has not exceeded limit of current in progress jobs
- Postconditions: customer's request is added to queue of in progress jobs, and dasher sees options to update progress bar of jobs
- Flow of events for main success:
 - 1. Dasher receives request and clicks on accept button
 - 2. Request is added to Dasher's in progress Jobs which is on **DasherHomePage**

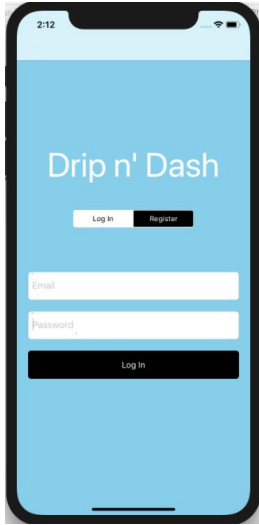
d. System Sequence Diagram



4. User Interface Specification

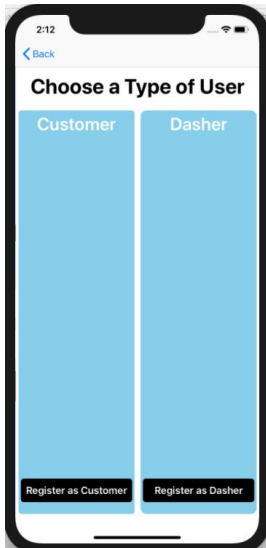
a. Preliminary Design





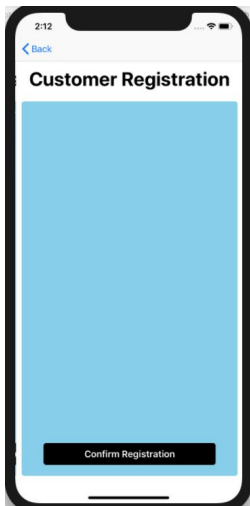
➤ **SignInPage:**

- This page will allow the user to either sign in, or register.
- If the user wants to sign in, the user can enter his/her email and password. When the user signs in, he/she will be prompted with either the **CustomerHomePage** or **DasherHomePage** depending on which type of user he/she is.
- If the user wants to register, the user can press the register button which will prompt the **RegisterInformationPage**.



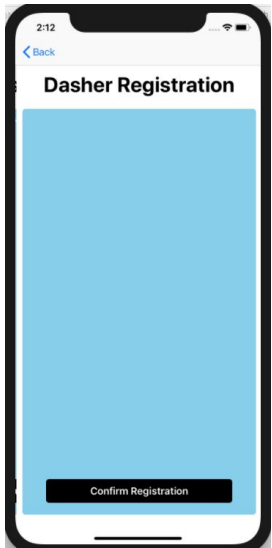
➤ **RegisterInformationPage:**

- This page will present information about the two types of users he/she can become (customer, dasher)
- It will contain a button to register as a customer. When pressed, the user will be prompted to the **CustomerRegisterPage**
- It will also contain a button to register as a dasher. When pressed, the user will be prompted to the **DasherRegisterPage**



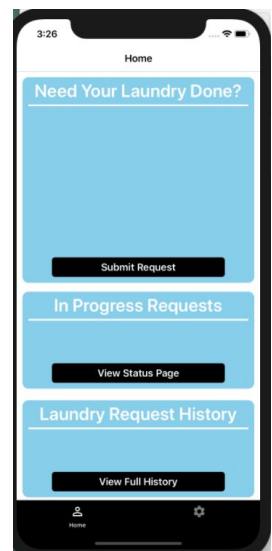
➤ **CustomerRegisterPage**

- This page will allow the user to input all the information required to register as a customer
- It will contain text fields for all the necessary information to register as a customer
- It will also contain a button to confirm registration once all the text fields have been filled out. When pressed (assuming all inputted information is valid), the user will be prompted to the **CustomerHomePage**
-



➤ **DasherRegisterPage**

- This page will allow the user to input all the information required to register as a dasher. It will likely also contain some sort of general knowledge quiz about laundry to ensure the user knows how to do laundry
- It will contain text fields for all the necessary information to register as a dasher
- It will also contain a button to confirm registration once all the text fields have been filled out. When pressed (assuming all inputted information is valid), the user will be prompted to the **DasherHomePage**



➤ **CustomerHomePage**

- This page will allow a customer to perform the core functions of our app, including posting a request for laundry and seeing the status of any in progress laundry requests the user has
- It will likely also contain a link to see the user's history of completed laundry requests

➤ **DasherHomePage**

- This page will allow a dasher to perform the core functions of our app, including the ability to accept jobs, see and update the status of any in progress jobs, and see their past completed jobs
- It will include a section showing a list of all of a dasher's currently in progress jobs. When an item on this list is pressed, the dasher will be prompted to the **DasherJobStatusPage** for the job he/she clicked

➤ **CustomerJobStatusPage (opened from CustomerHomePage)**

- This page will allow the customer to see the status of his/her in progress requests for laundry, including information about whether his/her request has been accepted by a dasher, whether the laundry is in the dryer/washer, etc.

➤ **CustomerJobHistoryPage (opened from CustomerHomePage)**

- This page will allow a customer to see information about all his/her past completed laundry requests.
- **CustomerSettingsPage**
 - This page will allow a customer to perform basic settings functions, including updating any of his/her personal information
- **DasherJobStatusPage (opened from DasherHomePage)**
 - This page will show the status of a dasher's in progress job. He/she will update this page at different core points in the laundry request (waiting for washer, in washer, in dryer, etc.). When this page is updated, the customer associated with this job/jobPage will have his/her **CustomerJobStatusPage** updated accordingly
- **DasherJobHistoryPage (opened from DasherHomePage)**
 - This page will allow the dasher to see information about his/her completed laundry jobs
- **DasherSettingsPage**
 - This page will allow a customer to perform basic settings functions, including updating any of his/her personal information

b. User Effort Estimation

- Customer Registration:
 - 1 click: select register account option on application sign in page
 - 1 click: select customer on dasher or customer registration page
 - 7 clicks: enter in every data text field/drop down selection on registration page when creating an account
 - 1 click: click on submit button to finalize account registration
 - **Total 10 clicks: 7/10 clerical data entry, 3/10 UI navigation**
- Dasher Registration:
 - 1 click: select register account option on application sign in page
 - 1 click: select dasher on dasher or customer registration page
 - 6 clicks: enter in every data text field/drop down selection on registration page when creating an account
 - 1 click: click on submit button to finalize account registration information and proceed to Dasher quiz
 - 7 clicks: enter in answers into each data text field for each of the 7 questions on quiz
 - 1 click: enter submit to submit answers to quiz and proceed to navigate to home page

- **Total 17 clicks: 13/17 clerical data entry, 4/17 UI navigation**
- Sign into Account:
 - 2 clicks: enter in username and password into text fields on application sign in page
 - 1 click: click on login button to submit account information
 - **Total 3 clicks: 2/3 clerical data entry, 1/3 UI navigation**
- Customer - Request laundry to be done:
 - 1 click: press request button
 - 4 clicks: enter in basic info about request (laundry pick up time, number of loads, etc.)
 - 1 click: press submit button to proceed to home page until matched with dasher
 - 1 click: click accept or decline on presented dasher match
 - 1 click: after matched with dasher, click on the current orders button to navigate to the current laundry orders page to view status of current orders (on the way, in washer, in dryer, etc.)
 - 2 clicks: after order is placed in status of completed by the Dasher, confirm status of laundry by clicking on order status button and then selecting completed from the drop down menu
 - 1 click: after the job is placed in the status of completed, a pop up window appears requesting star rating for the dasher: click between 1 through 5 stars to rate the Dasher for their services
 - 1 click: click submit on ratings popup window to submit rating and finalize order
 - **Total 12 clicks: 6/12 clerical data entry 6/12 UI navigation**
- Dasher - Request to do laundry:
 - 1 click: press job accepting mode button
 - 1 click: press submit button to proceed to home page until matched with a customer
 - 1 click: click accept or decline on presented customer and job description
 - 1 click: after matched with customer, click on current jobs button to navigate to current jobs page
 - 6 clicks: change status of job accordingly by clicking on job status button on the current jobs page and then selecting the corresponding status from the drop down menu (5 statuses: on the way to pick up, in washer, in dryer, on the way to drop off, completed)
 - 1 click: after job is placed in status of completed, a pop up window appears requesting a star rating for the customer: click between 1 through 5 stars to rate the customer
 - 1 click: click submit on ratings popup window to submit rating and finalize job
 - **Total 12 clicks: 10/12 clerical data entry, UI navigation 5/12**

5. Domain Analysis

a. Domain Model

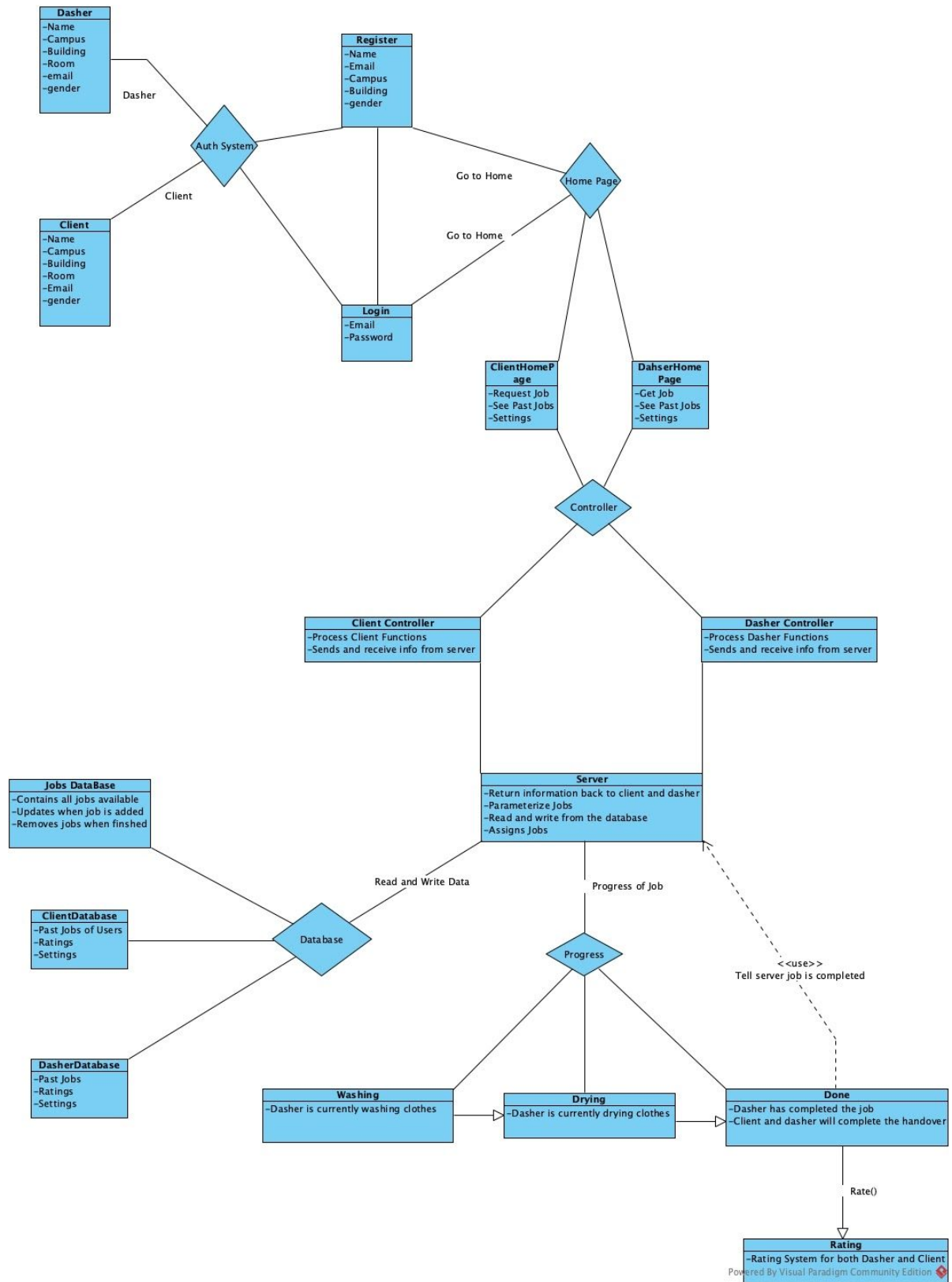
To find the domain model the group first thought of the key features that we wanted to implement. Once the features were established the next step is to figure out how we are going to implement them. Fortunately for us most of the methods and classes call on each other making the project less complex than others. Although implementing such features may seem tedious, the way that we went about it was further from it.

The first step is understanding the fields for both Client's and Dashers. This includes Name, Email, Password, Campus, Building, Room Number, and Gender. All the information will be saved through our Database in Firestore. These fields will be entered either through the registration page. The Authentication system handles the users who either login or decided to join with a registration page. Once the system determines that the users are valid they will be presented to a home page. The Home page is where all the functionality is going to be accessed through. When the user wants to use a functionality a controller will process the input and output to ensure that the desired outcome is valid.

Since the premise relies on real time and updated data the server will pull from a centralized database. This makes the database easier to implement and also less complex. The Database will include all job posting alongside user info. Once a job is requested by a Client the server will write the job offering on a Job listing portion of the database that Dashers can view. Once the job is accepted by a Dasher it is taken out of the listing and the Client will be notified.

The job then will pass through a progress function. The Dasher will update the system on the status of the job. This status can be viewed in the clients homepage.

Once the job is completed and the clothes are dropped/picked up a rating system will appear for both the Client and Dasher. The rating system is used to ensure that both the Clients and Dashers receive the best experience and continue to use the service. This cycle continues as long as there are students with dirty clothes.



Concept Definitions

Responsibility Description	Type	Concept Name
Store all authentication information about registered customers and dashers, in progress laundry requests, master list of all completed laundry requests, etc.	K	Database <<entity>>
Allow a registered user (customer/dasher) to sign in by entering his/her email and password, and provide link to register for the application	D	SignIn <<boundary>>
Provide information to an unregistered user what a customer is and what a dasher is in our application.	D	RegisterInformation <<boundary>>
Allow an unregistered user to enter his/her information and sign up for our application as a customer.	D	CustomerRegister <<boundary>>
Allow an unregistered user to enter his/her information and sign up for our application as a dasher	D	DasherRegister <<boundary>>
Determine if the information a user entered to register as a customer is valid, and initialize customer's data in the database (Firestore). Interface for the application and reads/writes to customer's information in the database	D	CustomerFirestore <<entity>>
Determine if the information a user entered to register as a dasher is valid, and initialize customer's data in the database (Firestore). Interface for the application and reads/writes to dasher's information in the database	D	DasherFirestore <<entity>>
Container for customer's data frequently used by other parts of the system (firstName, lastName, email, dorm, dormRoom, etc.)	K	Customer <<entity>>
Container for dasher's data frequently used by other parts of the system (firstName, lastName, email, dorm, dormRoom, etc.)	K	Dasher <<entity>>
Allow a customer to enter information about his/her request for laundry and submit it, and render an interactive list with all of a customer's in progress jobs	D	CustomerHome <<boundary>>
Present detailed information about the current status of a customer's in progress laundry request including (CURRENT_STAGE in ["pending dasher assignment", "dasher outside for pickup", etc.], estimated drop off time, link to DasherProfile of dasher assigned to the request,	D/K	CustomerJobStatus <<boundary>>

etc.)		
Present an interactive list of all a customer's completed requests for laundry with basic information about each request	D/K	CustomerJobsHistory <<boundary>>
Present detailed information about a customer's past laundry request including (timeStamp the request was submitted, timeStamp the request was completed, dasher that completed the request, amount paid, etc.)	D	CustomerPastJobInfo <<boundary>>
Allow a dasher to accept new jobs, and render a list with basic information about the current status of the dasher's in progress jobs.	D	DasherHome <<boundary>>
Allow a dasher to update the current status of a customer's laundry request (CURRENT_STATUS in ["Outside for Pickup", "Laundry in Washer", "Outside for Drop Off", etc.]	D	DasherJobStatus <<boundary>>
Present an interactive list of all a dasher's completed jobs with basic information about each job	D	DasherJobsHistory <<boundary>>
Present detailed information about a dasher's past laundry request including (timeStamp the request was submitted, timeStamp the request was completed, dasher that completed the request, amount paid, etc.)	D	DasherPastJobInfo <<boundary>>
Container for all information related to a customer's job/laundry request including (jobID, number of loads, customer's dorm, customer's dorm room, etc.)	K	JobRequest <<entity>>
Handle reads/writes about JobRequest information to the database (Firestore). Interface for the application and the JobRequest portion of the database.	D	JobRequestFirestore <<entity>>
Listen for changes to the status of JobRequests in the database (updates made by a dasher) and notify parts of the system that need this information	D	DatabaseListener <<entity>>

Concept Pair	Association Description	Association Name
SignIn <--> Database	SignIn sends user's authentication information and Database returns whether the user is registered as a customer, dasher, or neither	sendsAuthData
SignIn -> RegisterInformation	If a user clicks the register button, SignIn presents the RegisterInformation page	presentRegInfo
RegisterInformation -> (Customer/Dasher)Register	If a user clicks the (customer/dasher) register button, RegisterInformation presents the (Customer/Dasher)Register page	present(C/D)Reg
CustomerRegister -> CustomerFirestore	CustomerRegister sends a user's inputted registration information to CustomerFirestore, which returns isValidRegistration = true if the form is valid (email domain @rutgers.edu, password meets security requirements, etc.), else it returns isValidRegistration = false	validatesRegData
CustomerFirestore -> Database	CustomerFirestore sends the registration information it received from CustomerRegister to the Database, and creates an entry in the Database containing all the customer's data	initCustomerData
DasherRegister -> DasherFirestore	DasherRegister sends a user's inputted registration information to DasherFirestore, which returns isValidRegistration = true if the form is valid (email domain @rutgers.edu, password meets security requirements, etc.), else it returns isValidRegistration = false	validatesRegData
DasherFirestore -> Database	DasherFirestore sends the registration information it received from DasherRegister to the Database, and creates an entry in the Database containing all the dasher's data	initDasherData
CustomerHome -> JobRequestFirestore	CustomerHome sends the information a customer submitted about a job request, and JobRequestFirestore interfaces with the database to store the request	submitJobRequest
JobRequestFirestore -> Database	JobRequestFirestore writes the information about a customer's JobRequest in the database (@dorms/Dasher.dormName/jobsPendingAssignment/jobID and @jobsInProgress/jobID)	writeJobRequest
DasherHome ->	When a user enters job accepting mode, DasherHome	getJobRequest

JobRequestFirestore	asks JobRequestFirestore to assign the dasher the oldest pending request to which he/she has access	
JobRequestFirestore -> Database	When prompted by DasherHome, JobRequestFirestore reads all the pending JobRequests in the database (@dorms/Dasher.dormName/jobsPendingAssignment) to which the dasher has access	readJobRequests
DatabaseListener -> Database	DatabaseListener checks for changes/updates to status information in a JobRequest document in the database (@jobsInProgress/JobRequest.jobID)	checkForUpdates
DatabaseListener -> CustomerJobStatus	DatabaseListener notifies CustomerJobStatus about any changes to the current status of a customer's JobRequest	notifyWithUpdates
DasherJobStatus -> JobRequestFirestore	When a dasher makes an update to the status of a customer's JobRequest, DasherJobStatus sends the updated information to JobRequestFirestore	updateJobRequest
JobRequestFirestore -> Database	When prompted by DasherJobStatus, JobRequestFirestore writes the updates to a JobRequest in the database (@inProgressJobs/JobRequest.jobID)	writeUpdates
(C/D)Home -> (C/D)JobStatus	When a customer/dasher selects a job in the list of in progress jobs on the (Customer/Dasher)Home, (C/D)Home presents the (C/D)JobStatus for that specific job	presentJobStatus
(C/D)JobsHistory -> (C/D)JobInformation	When a customer/dasher selects a job in the list his/her completed jobs on the (Customer/Dasher)JobsHistory, (C/D)JobsHistory presents the (C/D)PastJobInformation page for the specific job he/she clicked	presentPastJobInfo

Concept	<u>Attribute</u>	<u>Attribute Description</u>
Database	storedData	All data stored in the database including (customers, dashers, inProgressJobs, etc.)
SignIn	authData	A user's authentication information, currently it represents the user's email and password
(Customer/Dasher)Register	- registrationData - isValidRegistration	- The required information a user submits to register as a customer/dasher (name, email, etc.) - Boolean value stating whether the the information the user submitted is valid
CustomerHome	- inProgressJobs - JobRequestInfo	- array of JobRequest.jobID's for all of a customer's current in progress jobs - Information about a job request a customer submits
DasherHome	- inProgressJobs - isAcceptingJobs	- array of JobRequest.jobID's for all a dasher's current in progress jobs - Boolean value stating whether dasher is currently accepting jobs ("in job accepting mode")
(Customer/Dasher)JobStatus	- jobID - JobRequest	- jobID of the JobRequest this JobStatus page is showing - JobRequest object with all current information about the status of the job, and other relevant details
(Customer/Dasher)JobsHistory	- completedJobs	- array of JobRequest objects used to populate the information on the interactive list of a customer/dasher's past jobs on this page
DatabaseListener	- jobRequestUpdates	- updates to a JobRequest read from the Database

Traceability Matrix

	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8
Database	x	x	x	x	x	x	x	x
SignIn	x							
RegisterInformation	x							
CustomerRegister	x							
Dasherregister	x							
CustomerFirestore	x							
DasherFirestore	x							
Customer	x	x	x	x	x	x	x	
Dasher	x	x	x	x	x	x		x
CustomerHome		x						
CustomerJobStatus					x			
CustomerJobsHistory						x		
CustomerPastJobInfo						x		
CustomerSettings	x							
DasherHome				x				
DasherJobStatus					x			
DasherJobsHistory						x		
DasherSettings	x							
JobRequest			x	x	x			
JobRequest Firestore			x	x	x			

System Operation Contracts

Name: login()

Responsibility: Use the database to control account systems and decipher between Customer and Dashers

Cross-References: UC-1

Output: The system will present the user with an UI that will ask them to submit the corresponding credentials

Pre-Conditions: The account name and password must be valid in the database before allowing the account to gain access to the system

Post-Conditions:

- Customers should have access to request jobs done and see the status of their requested job
- Dashers should be able to

Name: JobCreation()

Responsibility: Creates jobs to be sent out to the Dashers.

Cross-References: UC-2, UC-4

Output: The system presents the Dasher with a job that an Customer requested. The Dasher could either keep the job, or decline it.

Pre-Conditions: Dasher account must be logged on in order to receive a new job

Post-Conditions:

- The Dasher will receive the job, and from there they can update the progress on their job

Name: JobUpdates()

Responsibility : Updates the Customer with the progress the Dasher has made with their laundry.

Cross-References: UC-5

Output: The system will update the progress bar on the Customer's end so that they can see how far along the Dasher is with their laundry.

Pre-Conditions:

- The Customer must be logged in and has to have requested a job that was accepted by a Dasher.
- Dasher must be logged in and has to have accepted a job request from a Customer.

Post-Conditions:

- On the Customer's end the progress bar will be updated to show what step the Dasher is currently on
- On the Dasher end they have to press a button that will update the the job to reflect what step they are on until the job is finished

Name: JobCompletion()

Responsibility: The system informs the Customer that their request has been completed and is ready to be dropped off

Cross-References: UC-6

Output: Once the system sees that the Dasher has gone through all of the steps in order to complete the job, it will notify the Customer that their job is ready to be dropped off

Pre-Conditions: Customer and Dasher must be logged in. The Dasher has to have completed the previous steps in order to notify the Customer that the job is ready to be dropped off

Post-Conditions: The Customer will be presented with a notification that the Dasher is on their way to drop off the laundry they had requested to be cleaned.

Name: Rating()

Responsibility: A rating system for both the Customer and the Dashers to tell the quality of the user.

Cross-References: UC-7, UC-8

Output: The system will display the rating of the Dasher to the Customer. Additionally after a job is completed the Customer can leave a rating stating if the Dasher did a good job or not. Alternatively, the Dasher can also rate Customers.

Pre-Conditions: Customer or Dasher must be logged on.

- Customer must have requested at least 1 job in order to receive a rating
- Dashers must have completed at least 1 job in order to receive a rating

Post-Conditions: .

- Customers will receive a rating based on how they are as a customer.
- Dashers will receive a rating which would signify their quality as a Dasher.

Mathematical Model

The main model we are using for this project is based on pricing, represented by a function based on the number of loads as well as the type of machine used during the washing and drying process. The end result should be greater than the minimum wage, since the whole process should take at least an hour to complete. We would then separate this fee into a flat fee for machines, and the rest to (whatever we decide to price, typically between \$5 and \$10). Our function for price: $Price(n) = (n * machine\ costs) + flat\ fee + convenience\ fees$, where n is the number of loads required to complete the request.

6. Project Operation Size

Actor	Type of Actor	Actor Weight
Customer	GUI - Complex	3
Dasher	GUI - Complex	3
Job System	Protocol - Average	2
Progress Bar	API - Simple	1
Rating System	API - Simple	1
Database	API - Simple	1

Additionally, we can derive the Unadjusted Actor Weight to be

UAW = (Total No. of Simple actors x 1) + (Total No. Average actors x 2) + (Total No. Complex actors x 3) = (3 * 1) + (1 * 2) + (2 * 3) = 3 + 2 + 6 = 11.

➤ Technical Complexity Factor:

Factor	Description	Weight (W)	Rated Value (0 to 5) (RV)	Impact (W * RV)
T1	Distributed System	2.0	0.5	1.0
T2	Response Time or Performance Objectives	1.0	3	3.0
T3	End User Efficiency	1.0	4.5	4.5
T4	Complex Internal Processing	1.0	1.5	1.5
T5	Reusable Code	1.0	2	1
T6	Easy to Install	0.5	4.5	2.25
T7	Easy to Use	0.5	5	2.5
T8	Portable	2.0	4.5	9.0
T9	Easy to Change	1.0	3.5	3.5
T10	Concurrent	1.0	3	3.0
T11	Includes Special Security Objectives	1.0	4	4.0

T12	Provides Direct Access for Third Parties	1.0	4	4.0
T13	Requires Special User Training	1.0	3	3.0

First we add up the sum of the Impact of the Technical Factors to get 42.25. This is used to calculate the Technical Complexity Factor, which is derived as

$$TCF = 0.6 + (TF/100) = 0.6 + (42.25/100) = 1.0225.$$

➤ Environmental Complexity Factor:

Factor	Description	Weight (W)	Rated Value (0 to 5) (RV)	Impact (W * RV)
E1	Beginner familiarity with UML-based development	1.5	3	4.5
E2	Application Experience	0.5	4	2.0
E3	Object Oriented Experience of Team	1.0	3	3.0
E4	Lead Analyst Capability	0.5	2	1.0
E5	Motivation	1.0	5	5.0
E6	Stable Requirements	2.0	5	10.0
E7	Part-time staff	-1.0	0	0.0
E8	Difficult Programming Language	-1.0	3	-3.0

The sum of the impact of the environmental complexity factors is equal to 22.5.

Moreover, the Environmental Complexity Factor is derived as

$$ECF = 1.4 + (-0.03 \times EF) = 1.4 + (-0.03 \times 22.5) = 0.725$$

Finally, the Use Case Points UCP can be calculated from the previously determined values.

$$UCP = (UUCW + UAW) \times TCF \times ECF$$

$$UCP = (50 + 11) \times 1.0226 \times 0.725$$

$$UCP = 45.22 \approx 45 \text{ Use Case Points}$$

7. Plan of Work

PHASE 1 - Building blocks

We will begin finalizing our page layout, finalizing how all of the pages will lead to each other and the general user interface of the app. Next, we will work on registration for dashers and customers by figuring out the required fields needed for each type of user as well as how these fields will be structured in the database. For our database we will be using a web services tool called, Firebase. This service has tools to manage a database and can also handle user authentication. These “required fields” such as name, id and payment information will all be stored on a secure database in the cloud. This initial setup of users will allow us to test future phases where users are interacting in the app. The pages that will be built in this phase will be the user registration page (both customers and dashers).

PHASE 2: Core Functionalities

Then we will begin implementing all the different use cases such as requesting laundry, completion of job, and status bar update. These are core functionalities of our app, so this phase is where the bulk of the app is built. The first main user interaction we must build is the ability to post laundry jobs. The user will be able to select specific options that describe their clothing and give details about their location. Once the job is submitted by the user, it will be posted on a page viewable by the dashers. This page will consist of a list view where dashers can view the jobs currently posted in their building. The biggest task on this page will be building a dynamic list that will automatically update and remove jobs as they are accepted in real-time.

The next main process to build is the status page. Here, the user can view the status of their laundry job. It will have a progress bar that visualizes multiple stages of the job such as “picked up”, “in washer”, “in dryer” and “dropped off”. This feature will be important because it will give the user a rough estimate of when their laundry will be dropped off (in addition to an estimate drop off time). Furthermore, we must also build this status functionality from the dasher point of view. The dasher must have the ability to update the status of the job as he/she finishes a certain stage.

PHASE 3:

After completing the main features, the next large aspect of a “delivery” type is a review system. We will implement a 2-way review system, where users can review dashers and vice versa. Both sides will be able to publish reviews after the job is complete. This page will have a 1-5 star rating and will also give the user an option to tip.

Furthermore, we will look into adding some extra features that are not mandatory but would help with making the app more useful. Extra features would likely things like maybe in-app chat and luxury options. Some possibilities for luxury options are; folding clothes, hanging clothes, location tracking. Finally, we will focus on the look of the app, things like color scheme and logo. All of this should be done between a week or a week and a half before the final demo. That extra week will be used for rigorous testing of as many edge cases as we can think of. This is to ensure that the product is working perfectly in any condition.

8. Reference

Bradford, Laurence. “How to Make An Android App For Beginners - 5 Tips!” *Learn to Code With Me*, 7 Feb. 2020,
learntocodewith.me/programming/android/beginner-app-development/.

“Connect to Firebase : Android Developers.” *Android Developers*,
developer.android.com/studio/write/firebase.

Harvard Business Review Staff. “The Four Phases of Project Management.” *Harvard Business Review*, 3 Nov. 2016, hbr.org/2016/11/the-four-phases-of-project-management.

harkiran78 “Top Programming Languages for Android App Development.”
GeeksforGeeks, 19 May 2019,
[www.geeksforgeeks.org/top-programming-languages-for-android-app-development /](https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/).

“Installation & Setup on Android | Firebase Realtime Database.” *Google*, Google,
firebase.google.com/docs/database/android/start.

Mayank. “How to Build an App like Uber: Uber for Drivers & Riders - EngineerBabu.”
EngineerBabu Blog & Success Stories, EngineerBabu, 29 Jan. 2020,
engineerbabu.com/blog/how-to-build-an-app-like-uber/.

Milano, Diego Torres. *Android Application Testing Guide: Build Intensively Tested and Bug Free Android Applications*. Packt Publishing, 2011.