Lab 2 – Load.In Product Specification

Lance Perdue

Old Dominion University

CS 411W

Professor Janet Brunelle

March 28, 2020

Lab 2 Version 1

Table of Contents

List of Figures

List of Tables

1. Introduction

The problem when it comes to do-it-yourself movers is that one lacks the expertise to handle the logistics of a move. In a study conducted by OnePoll, the average American takes four and a half months to unpack after moving (Knoblauch, 2019). Keeping account of everything when packing can pose a problem when moving. Misplacement of items is stressful when the DIY mover must search through every box to find an item. Subsequently, DIY movers are not utilizing using the space of a moving truck effectively. This can lead to the DIY mover making more trips than necessary. More trips the DIY mover must compete means the cost of a move is increased. According to a study by the New York Post, more than one in ten people would rather spend a week in prison and then move (Knoblauch, 2019).

1.1. Purpose

This document provides the software requirements specification for Load.In. Load.In is an implementation of the move process that helps the DIY mover to become a professional mover. With Load.In, features such as catalog boxes, measure furniture, estimate truck sizes, know the number of trips a move takes, pack like a pro and knowing where all items are. With the assistance of artificial intelligence and computer vision, one can become an expert in moving. The use of artificial intelligence will help the end user to place objects professionally in the moving truck. Using computer vision, the sizes of the boxes are measured and displayed to visualize the placement of the boxes. Helpful tips and tricks are also provided to the end user to help with moving. If the end user needs help moving an object, the end user will also receive expert-level instructions.

1.2. Scope

The Load.In application is an implementation of the move process that helps the DIY mover to become a professional mover. The measurement of boxes, 3D model generation, load plan, and feedback will be reduced to partial implementation. The box locater, estimated trips, packing tips, tips search, and expert articles will have full functionality. The features that will be removed are weight, fragility, unloading instructions, chatbot, live expert, web scraper, vendor web API, truck sizes, truck availability, location data, move data, heatmap, and rental interest statistics.

The case study for Load.In is an average family of three with a dog. A family of three is the average household according to the United States Census Bureau from December 2020 (US Census Bureau, 2019). This family can help Load.In with development since this is the average family in the United States. This family will have a home that is 2,200 square foot home, requiring a moving truck to be utilized. The 3D model generation will be utilized on the furniture to generate a load plan. When the load plan is generated, the family will then see the cost of the move and how to load like a professional. In future iterations of the application, Load.In could potentially expand to the industrial industry by offering efficient loading.

1.3. Definitions, Acronyms, and Abbreviations

3D – Three Dimensional

Administrator – Someone who will access elevated features of the Load.In system in order to maintain and detect issues.

Amazon Lambda - a serverless compute service that lets you run code without provisioning or managing servers

Amazon RDS - Amazon relational database service

Amazon Web Services (AWS) – A cloud platform on which Load.In's databases are hosted.

Android - a mobile operating system based on a modified version of the Linux kernel and other open-source software

Application Programming Interface (API) – An interface for programs to share information and functionality with one another through a series of call or connections.

AWS Elastic Beanstalk - an orchestration service offered by Amazon Web Services for deploying applications which orchestrates various AWS services, including EC2, S3, Simple Notification Service, CloudWatch, autoscaling, and Elastic Load Balancers

AWS Elastic File Storage – an AWS service that provides file storage with the ability to auto-scale up with increased demand.

Apache CFX – A popular library for hosting web apis.

Apache Tomcat - an open-source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and WebSocket technologies. Tomcat provides a "pure Java" HTTP web server environment in which Java code can run

Chatbot – A feature within Load.In that provides information to users and guides them towards helpful articles and other resources interactively.

Cloud – A term used to describe several computing models such that a company or individual can purchase resources for hosting a variety of things in a centralized location accessible from anywhere in the world.

Computer Vision – a subclassification of Artificial Intelligence that involves computing information about the world from various sensory data, such as images. Techniques of this classification are used throughout Load.In to observe real world objects.

CPU – Central processing unit.

CSS – Cascading style sheet.

Do-It-Yourself (DIY) Mover – Non-professional movers that rent a truck for their move, but and handle all packing, unpacking, manual labor themselves. This is the primary end user of Load.In

Expert Tips – Feature of Load.In that allows for a mover to search for helpful articles pertaining to a variety of useful information on how to accomplish various tasks during a move.

GHZ – Gigahertz

Guest – Someone who is accessing the Load.In system anonymously and has not registered for an account or someone who has registered but has not authenticated to the system at the time of access.

GUI - graphical user interface, the aspect of a software program that the end user interacts with.

HTML5 – Hyper Text Markup Language version 5

Java - a set of computer software and specifications developed by James Gosling at Sun Microsystems, which was later acquired by the Oracle Corporation, that provides a system for developing application software and deploying it in a cross-platform computing environment

JavaScript - A scripting language that runs in the browser and performs one or more function to animate an otherwise static HTML document.

Linux - An open-source and community-developed operating system for personal computers and work stations.

Load Plan – A set of instructions on how to optimally load a container, generated automatically by Load.In from the boxes and furniture input into the system by the user.

Logistics Planning – A feature of Load.In that assists the mover with determining what rental trucks cost, how many trips the truck might need to take and whether the truck is available to rent based off proximity to the mover.

Mbps – Mega-bits per second, a unit of measurement for network speeds.

Megapixel – One million pixels, typically used to measure the size and quality of images

Move Inventory – A feature of Load.In that catalogs all boxes and items the mover intends to move.

MySQL - an open-source relational database management system

MacOS - An operating system used on Apple's MacIntosh line of personal computers and work stations.

Operating System (OS) – A collection of programs designed to provide a platform on a device to run other applications and typically provides a layer of abstraction from the hardware it interacts with.

Pixel – A small square of color that is part of a larger display screen or image.

Photogrammetry – A computational method of deriving three-dimensional information from images. This method is used in Load.In to construct 3D models of boxes, furniture, and other items from pictures taken from the end user's cell phone camera.

Portable Network Graphics (PNG) – Portable Network Graphics, a common image file format that Load.In uses

Professional Mover - Professionals who handle the physical labor of loading and unloading a moving truck, as well as driving the truck to the destination.

Real World Product (RWP) – The actual Load.In solution as it was intended.

Rental Administrator – A representative of a rental company who will access the Load.In system on behalf of the rental company.

Rental Company – Any company which rents moving vehicles for a Do-It-Yourself Mover to assist them with their move.

Smartphone – A device, typically handheld, which can act as both a cellular phone and a computer by running one or more applications through typically a touch screen interface.

SPRING MVS - an application framework and inversion of control container for the Java platform

Test Harness – A set of special features used during the development of Load.In to enable testing and demonstration of the application

Vendor Synchronization – A feature of Load.in that brings in truck sizes and availability of rental information from third party moving company websites.

Windows – An operating system developed by Microsoft for use on personal computers and work stations.

1.4. References

Knoblauch, M. (2019, May 8). One in ten Americans would prefer a week in jail over moving. New York Post. https://nypost.com/2019/05/08/one-in-ten-americans-would-prefer-a-week-in-jail-over-moving/

Wood, T. (2020, January 6). Moving Industry Statistics. MoveBuddha. https://www.movebuddha.com/blog/moving-industry-statistics/

Andrew, P. (2020, January 26). Is Your House the "Typical American Home"? Hsh. https://www.hsh.com/homeowner/average-american-home.html

CADCode Systems. (n.d.). Optimizing & Machining | CADCode Systems. CADCode. Retrieved September 20, 2020, from https://www.cadcode.com/category/categories/optimizing-machining

Collins, T. (2018, April 20). A Look into Photogrammetry and Video Games. Medium. https://medium.com/@homicidalnacho/a-look-into-photogrammetry-and-video-games-71d602f51c31

Dube, E. (2020, September 20). Optimizing Three-dimensional Bin Packing through Simulation. Semantics Scholar. https://www.semanticscholar.org/paper/OPTIMIZING-THREE-DIMENSIONAL-BIN-PACKING-THROUGH-Dube/bb9986af2f26f7726fcef1bc684eac8239c9b853#references

Economy Moving & Storage, LLC. (2015, January 4). How to properly pack and load a moving truck- Movers Cincinnati. YouTube. https://www.youtube.com/watch?v=rjmofUZOdwo&feature=youtu.be

Nat and Friends. (2017, April 18). Google Earth's Incredible 3D Imagery, Explained. YouTube. https://www.youtube.com/watch?v=suo_aUTUpps&feature=youtu.be

The American Institute of Stress. (n.d.). The Holmes-Rahe Stress Inventory PDF. Retrieved September 20, 2020, from https://www.stress.org/wp-content/uploads/2019/04/stress-inventory-1.pdf

The Top 5 Moving Mistakes Across America. (2019, August 13). Article. https://www.article.com/blog/top-5-moving-mistakes/

US Census Bureau. (2019, October 10). Historical Households Tables. The United States Census Bureau. https://www.census.gov/data/tables/time-series/demo/families/households.html

Perdue, Lance (2021, Match 20). Lab One

1.5. Overview

This standard is made up of three section. Section 1, Introduction, provides an overview of the entire document. Section 2, General Description, provides background information and identifies the user's needs for the requirements defined in Section 3.

Section 3, Specific Requirements, establishes the requirements that must be satisfied by the Load.In program.

2. General Description

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it. It will also describe what type of challenges Load.In could experience when implementing the prototype. At last, the external interfaces will go over networking, software and hardware the Load.In will interface with.

2.1. Prototype Architecture (Hardware/Software)

Docker shall connect key components together. Tomcat shall host the web API in order for the application to communicate with the database. MySQL shall be used to storage data in the database when the end user utilizes the application. The Load.In application shall interact with end users via Android application. This is either with a physical phone or built in SDK emulator inside of Android Studio. The test harness shall change values inside of the database when demonstrating the application as a proof of concept.

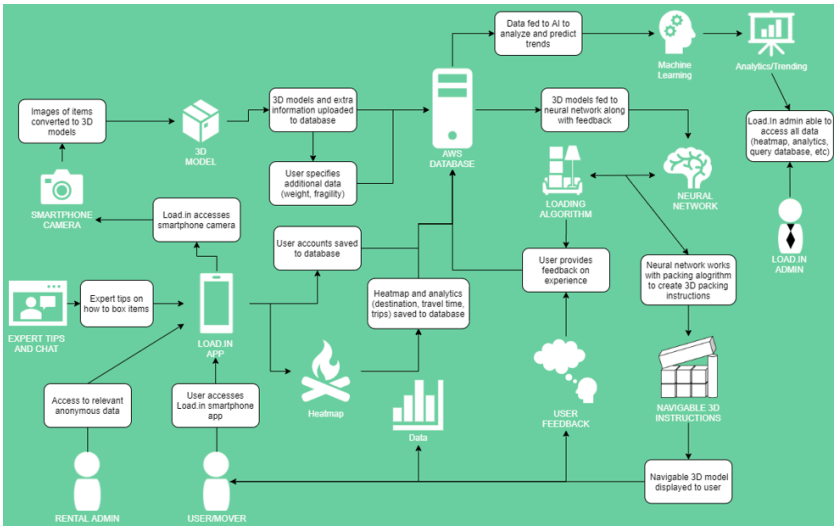


Figure 1 – Load.In Major Functional Components Diagram

2.2. Prototype Features and Capabilities

2.2.1. The prototype features that are kept to demonstrates a proof of concept for the application:

1. Measuring an item using the phone camera is demonstrating how items can be measured and stored into the database.
    A. If the phone camera is unavailable, the measurements are to be entered manually. The measurements entered into the application are then used to make a 3D model. Each item scanned in will have a QR code to identify each box location.
2. The load plan feature will configure how each of the boxes will be oriented inside of the truck based on the available free space. Next, the
3. Estimate the number of trips a DIY mover makes is also calculated based on the free space. If the truck space is filled and there are still boxes left, the number of trucks increases by one each time. When an end user runs into an issue and lacks the knowledge on how to pack an item, packing tips are made available for the user to search. The end user can type in an item name and find expert level articles based on the item entered. After a move is completed, the end user can provide feedback on how the move went. This will help Load.In improve the application over time with the provided feedback.

Feature	Description	Implementation
	Authentication	
Login User Interface	Create a landing page for the login screen.	Full Functionality

User Registration	Create an account inside of the load.in application.	Full Functionality
Reset Login	End user to be able to reset his/her own password inside of load.in application.	Partial
User Login	Authenticate with the server that an account exist.	Full Functionality
DIY User Interface	Create a user role for the end user with normal privileges.	Full Functionality
	Move Inventory	
Furniture/Item Measurement	Ability to measure items using the phone camera.	Partial
Move Inventory Interface	The interface shall consist of multiple different screens to manage the inventory.	Full Functionality
3D Model Generation	Generate a 3D Model based on item measurements	Partial
Box Locator	Users shall be provided with a search bar on the Move Inventory Interface to allow them to search for a specific box by the box number or description, they will be provided with the location of the box.	Full Functionality
	Expert Help	
Expert Tips User Interface	The user interface for expert tips shall alert the end users when an expert tip is found.	Full Functionality
Expert Articles	The expert articles shall provide useful information to the end user based on keywords stored for each article.	Full Functionality

	Load Plan	
Generation Algorithm	The Generation Algorithm for the Load Plan solves placements of Move Inventory items in a given truck.	Partial
Load Plan Display Interface	The Load Plan Display Interface takes a given Load Plan and displays it to the user.	Full Functionality
	Logistics Planning	
Rental Truck Costs	Calculate the cost of renting a truck.	Partial
Move Estimates	Calculate the number of trips one will take based on the truck size.	Partial
Move Inventory Storage	The system shall provide a mechanism by which the user inventory will persist and be accessible anytime by the user.	Full Functionality
	Analytics	
Feedback data	Ability for the end user to provide feedback for a move.	Partial
	Test Harness	
Sample Move Inventory	The test harness shall include a preset move inventory that will be representative of an average move in the United States.	Full Functionality
New Truck Size	The New Truck Size feature in the Test Harness will allow for testing of the Load Plan with different sized trucks.	Partial
	Algorithms	

Container Loading Algorithm	This algorithm works on the back end to generate a load plan that minimizes space of the user's moving truck given a set of boxes.	Full Functionality
3D Model Generation	This feature will be utilized in order to efficiently capture real-world measurements of boxes, which will later be used as input into the Load Plan algorithm.	Full Functionality
Expert tips Algorithm	The algorithm shall take in keywords based on the end users input for box content description during the move inventory phase.	Full Functionality
	Database	
Relational Database	The relational database shall store all persistent application data.	Full Functionality
User Credentials	The system shall provide a mechanism by which the user credentials can be persisted and retrieved at any time for the purposes of authentication.	Full Functionality
	Web API	
Authentication	The web API shall provide both a mechanism for authenticating users to the android client but also to authentication the user when utilizing services for the web API.	Full Functionality
Password Rest	The web API shall receive a request to reset the password from the android application.	Full Functionality

Expert Tips Indexer	The web API shall provide a mechanism for indexing the mySQL database that contains the Expert Tips.	Full Functionality
Services	The web API shall provide various services for handling RESTful requests made from the Android application.	Full Functionality

Figure 2 – Load.In Features Table

2.3. Prototype Development Challenges

The prototype could face challenges during the development cycle. The test harness changing values in the database is a major function to prove the load plan actually works. The challenge for this is implementing a random number generator for measurements to override the current values the end user placed in the database. Also related to the load plan is changing the truck size. This must be accounted for in an event the end user arrives at a rental station and the truck in the application is unavailable. Prioritizing search results when an end user search for tips could pose a challenge if the wrong article is shown first. That could easily render this feature useless if the end user types in piano and all the results show fine china. The biggest challenge Load.In could face are the edge cases that were not thought of during development.

2.4. External Interfaces

We have no major interfaces with prototype.

3. Specification Requirements

3.1. Functional Requirements

3.1.1. Android Application

The android application serves as the main interface that the mover will interact with

to gain access to the main features of Load.In. It consists of features such as the authentication feature, furniture/item measurement, move inventory, box locator, expert tips and articles, load plan, logistics planning, and user feedback features.

3.1.1.1. Authentication (Byron)

The system shall provide a way for users to be authenticated such that the individual identity of each user using the system can be established and therefore each user is distinguishable from one another. The system will utilize a combination of username and password that when challenged, the user shall provide and shall uniquely identify the user who intends to use the system. The system shall prevent an unauthenticated user from bypassing this process by requiring all individuals to establish their identity before using the system. The system shall prevent another user from attempting to use the identity of another

user by locking a user account when three or more invalid passwords have been entered into the system into a configurable timeframe.

3.1.1.1.1. Login User Interface (Byron)

The Login user interface provides a way for the user to login to the system in accordance with 3.1.3.3 The user login interface shall provide the user with the ability to enter in their username and password in order to authenticate with the system. The login interface shall:

1. Require that the user provide:
    A. Their username in the form of a string of characters
    B. Their password in the form of a string of characters
2. Hide the contents of the password being entered by obscuring each character that the user types.
3. Allow the user to submit their credentials to initiate a login attempt.
4. Consult with the user credential storage to compare password and username given by user against the known password for that user. (3.1.3.3)
5. After an unsuccessful login attempt, display to the user:
    A. Whether the either the username or password was incorrect
    B. The number of attempts remaining before the account will be locked
    C. Whether or not the account is currently locked
6. After a successful login attempt navigate the user to the main dashboard activity interface

3.1.1.1.2. Registration (Byron)

The Registration user interface allows for a user to self-register an account to use the application. It shall

1. Allow the user the ability to enter the following:
   A. An email address as a string of characters
   B. The first name of a user as a string of characters
   C. The last name of a user as a string of characters
   D. The phone number as a string of characters
   E. The password used to establish the unique identity of the user
2. When entering the password:
   A. Obscure each character of the password such that another user who is watching the screen cannot read the password
3. When registering:
   A. Prevent the registration of the user if the email address entered is already entered for another user in accordance to the unique constraint established for requirement (3.1.4.2)
   B. The email address is invalid according to the standard for email addresses
   C. Use the email address as the unique username
4. When registration is unsuccessful:
   A. Display an error message indicating that the email address has already been used

Commented [AB1]: Add a reference to the database requirements

5. When registration is successful:

A. Display a message indicating success

B. Navigate the user to the login screen (3.1.1.1.1) so that the user can authenticate

3.1.1.1.3. Reset Password Interface (Byron)

The Reset Password interface shall allow the user to request a password to be reset in the system. It shall accomplish this in two phases:

1. Allow the user to request a token. When requesting a token, the system shall:

A. Allow the user to enter:

The email for the user

B. Allow the user to submit the request for a reset

C. When the request is submitted:

Require the web API conduct the request

D. Display the success or failure of the response from the web API

E. When unsuccessful, require the user to perform the task again

F. When successful, move to the second step (3.1.1.1.3.2)

2. Allow the user to use a token to reset the password. When resetting the password, the system shall:

A. Allow the user to enter:

The temporary code from the email

The new password as a string of characters

B. When entering the password, obscure the characters similar to registration

C. Allow the user to submit the request for a password change with the token to the web api

D. When successful, navigate the user to the authentication interface (3.1.1.1.1)

E. When unsuccessful, have the user repeat step 1 of the reset password process.

3.1.1.2. Furniture/Item Measurement (Jason)

The item measurement feature enables the user to input information about their items in to the system, which are used throughout the application in order to

provide optimal assistance throughout the user's move. The following requirements must be met:

3.1.1.2.1. The user shall be able to specify the following information about their items:

1. Numeric inputs:
    A. Length in inches
    B. Width in inches
    C. Height in inches
    D. Weight on a scale 1-10
    E. Fragility on a scale of 1-10
2. Text input:
    A. Description of box contents

Commented [AB2]: Add inches

Commented [AB3]: Comment about how this will work (scale?)_

Commented [AB4]: Contents of the box description

Commented [AB5]: How does this overlap with the move inventory add

3.1.1.3. Move Inventory (Chris)

The system shall provide a way for users to view an inventory of the boxes they have added to their move, as well as information on each box itself. This system will take the user's login information to provide a list of boxes that are within

their specific inventory. The system shall be manageable dynamically by the user, with the ability to add items, edit items or delete items all together.

3.1.1.3.1. Move Inventory Interface

The interface shall consist of multiple different screens to manage the inventory. The inventory shall:

1. Have an “Add Item” feature that allows users to add a new item to the inventory, providing:
   A. Box Content Description (25-character description of the contents of the box)
   B. The room it came from (e.g. kitchen, office, garage)
   C. Dimensions (in inches, width, height and length)
   D. Fragility Scale (scale from 1 to 5, 5 being extremely fragile)
   E. Weight Scale (scale from 1 to 5, 5 being the heaviest)
   F. List of items contained within the box
2. Have a “View Inventory” feature that displays:
   A. A scrollable list of the items in a user’s inventory.
   B. The list shall display the description the user provided for an item.
   C. The list shall display the associated box number
   D. The list shall have a search feature that will sort the boxes based upon the entry contained within the search

Commented [AB6]: Do we have a matching database entry? Do we have a dropdown or do we allow the user to enter anything?
Maybe we have rooms entered? Already?

Commented [AB7]: Maybe add a sort ability

Commented [AB8]: There might be a limitation here in terms of characters that can be described

3. Allow the user to select an item in the inventory to view specific information on the item. In the “Box View” the system shall display the information of the item as well as buttons to:

   A. Edit the box

   B. Delete the box after confirmation by the user

   C. Add a new box.

4. Have an “Edit Box” screen that displays:

   A. Fields containing a box’s current values in editable fields corresponding to the values listed in 3.1.1.3.1

   B. Allows user to input changes to the fields

   C. Submit edits by pressing a button that sends the updated data to the database.

3.1.1.4. Box Locator (Greg)

Users shall be provided with a search bar on the Move Inventory Interface (3.1.1.3.1) to allow them to search for a specific box by the box number or description, they will be provided with the location of the box. Which will be one

of four states which will be loaded, at origin, at destination, or in transit. The following functional requirements shall be met:

1. Provide users the ability to input the box number or box description of the box they are attempting to locate.
2. The system shall identify the box by the number or description provided.
3. The system shall display the state of the box the user searched for to the user on the Move Inventory Interface along with all other information about the box.

3.1.1.5. Expert Tips. (Lance): The system shall provide the end user with expert level tips based on Box Content Description entered in from the Move Inventory

Commented [MT9]: Add ability to view box's position on truck in 3D (kind of like unload feature?)

Commented [AB10]: Reference the item's detail display

Commented [MT11]: add unload feature (checklist where users can manually specify a box is unloaded based on ID and description)?

Interface (3.1.1.3.1). Then the system shall contact the web API and return the title, article, photo, and video of the expert tip.

3.1.1.5.1. Expert Tips User Interface

The user interface for expert tips shall alert the end users when an expert tip is found. The expert tips shall alert the end user via Move Inventory Interface (3.1.1.3.1). The expert tips shall have the following:

1. The user interface shall trigger the indexer (3.1.5.3) based on the data inputted in the box content description and search for an expert tip.
    A. If an expert tip is not found, the expert tip will not be prompted to the user.
2. When an expert tip is matched based on the expert tip algorithm (3.1.3.4), the end user is then alerted that an expert tip is found.
    A. If the user toggles the expert tip, a new user interface is opened to display the expert article.

3.1.1.6. Expert Articles (Paul/Lance): The expert articles shall provide useful information to the end user based on keywords stored for each article.

1. The expert articles shall return the follow information to the end user:
    A. Article Title
    B. Article Content
    C. Video
    D. Images
    E. Strong men lifting kettlebells

Commented [AB12]: Add cross reference to database

3.1.1.7. Load Plan (Byron)

The Load Plan takes the Move Inventory and establishes where items to be moved

will fit in the truck and in what order. It will provide a way for the mover to visualize where each item goes into the truck.

3.1.1.7.1. Generation Algorithm

The Generation Algorithm for the Load Plan solves placements of Move Inventory items in a given truck. The generation algorithm shall:

1. Create one or more loads, which represents a trip to the destination in the move operation.
2. Place all items in the given move inventory into free space of a given truck size
3. Prevent unsafe items conditions:
   A. Prevent placing fragile items underneath other items
   B. Prevent placing heavier items on top of lighter items
4. Optimize space such that the items in question are fit for maximum space efficiency
5. When there is no more space available for a given item in one Load:
   A. Create another
   B. Place that item in the next Load

3.1.1.7.2. Load Plan Display Interface

The Load Plan Display Interface takes a given Load Plan and displays it to the user. It shall:

1. Display all items in proportion to each other such when two items are compared to each other, relative sizing is apparent to the user who is looking at the display similar to the real-world perspective

Commented [AB13]: Fix this one's wording "accurate 3d representation"

2. Display the loading of the truck in three-dimensional space

3. Display the steps of the load plan in the order that the mover should load into the truck.

4. Allow the user to navigate the steps of a load. When navigating

   A. Allow the user to progress forward with a step

   B. Allow the user to back up with a step

5. When displaying a box loading into the truck. Display the following:

   A. Camera looking at the center of the next box

   B. Show the box moving toward the destination with animation

   C. Show the box unique identifier

   D. Show the description of the contents of the box

6. Display the current status of the move

   A. The current step of the current load

   B. The number of loads that the mover must take

3.1.1.8. Logistics Planning (Greg)

This function provides users with logistical information regarding their move. There are two main areas, the Rental Truck Costs, which will pull cost data from the websites of local truck rental companies and the Move Estimate, which will

estimate how long the move will take a user based on distance and number of trips.

Commented [AB14]: Make reference to the default truck size of 17'

3.1.1.8.1. Rental Truck Costs

This section of the UI will provide the user with the costs of rental trucks within a set proximity of the user. The Rental Truck Costs interface shall:

Commented [AB15]: Make reference to a default value for the costs of the trucks

Add remarks about simulating the distance of the move from the origin to the destination

1. Use the user's location to identify rental truck companies around them.
2. Pull the models of trucks that are available to rent.
3. Pull the costs data for each of the models of trucks that are available.
4. Display the available trucks and their costs to the user.

3.1.1.8.2. Move Estimates

Provides estimates for distance and time that the move will take.

1. Use the user's location to determine their starting spot for their move.
2. Ask the user for the location of where they will be moving.
3. Calculate the distance between the two locations.
4. Estimate the time it will take to travel the distance calculated and multiply it by the number of trips needed that was determined by the Load Plan.
5. Display the distance and the time estimate to the user.

3.1.1.9. Feedback (Paul)

The Feedback interface shall allow the user to provide feedback on their experience with the Load.In application. It shall:

1. Provide the user with rating sections for:
   A. Account Creation / Login Experience
   B. Box Input
   C. Load Plan
   D. Expert Tips
   E. Overall application experience
2. The ratings will consist of scales of 1 to 5, thumbs up and thumbs down, and text boxes to allow for input of comments

Commented [AB16]: Little TLC

3.1.2. Test Harness

Commented [AB17]: Consider using several pre-loaded users representing families so that we don't need to enter in a lot of data for a family's load
We need some more family and user's that have to do with the color green: ex: Bruce Banner

We need a sample pre-populated load for a case study load

3.1.2.1. Sample Move Inventory (Chris)

The test harness shall include a preset move inventory that will be representative

of an average move in the United States. The items will total about 2,200 square feet and include items such as:

1. Large items representative of furniture
2. Fragile items representative of fine china and other valuables
3. A variety of standard box sizes available from
4. The test harness shall have an option to randomly generate a move inventory to test the functionality.
5. The test harness shall have the functionality to remove generated plans and replace them with new ones.

3.1.2.2. New Truck Size (Greg)

The New Truck Size feature in the Test Harness will allow for testing of the Load Plan with different sized trucks. Once the Sample Move Inventory has been generated the Load Plan can be tested for a given truck size. With this feature multiple different sized trucks can be used with the Move Inventory to see how the

Commented [AB18]: Predefined list of rental truck sizes

Load Plan works with all different sized trucks. The following functional requirements must be met:

1. Have a pre-defined list of the different sizes of rental trucks.
2. Display the different truck models and sizes.
3. Allow for the tester to switch between the different truck sizes.
4. Allow for tester to re-run Load Plan with the new truck size selected.
5. Display to tester the new generated Load Plan.

3.1.3. Algorithms

3.1.3.1. 3D model generation (Jason):

This feature will be utilized in order to efficiently capture real-world measurements of boxes, which will later be used as input into the Load Plan algorithm. It will utilize photogrammetry technology that uses input from a

smartphone's camera(s) to create a virtual three-dimensional representation of the object being observed. The following requirements must be met:

3.1.3.1.1. The following dimensions shall be captured:

1. The length of the box being measured
2. The width of the box being measured
3. The height of the box being measured

3.1.3.1.2. The algorithm shall utilize touch screen input provided by the user in conjunction with photogrammetry technology to determine which measurements will be taken

3.1.3.1.3. The dimensions gathered by this algorithm shall be accurate down to 1/8"

Commented [AB19]: Non-functional performance requirement

3.1.3.1.4. The capture of the dimensions shall only utilize the sensing devices built in to the phone, and shall not rely on external devices, or devices such as LiDAR sensors that are not available on the majority of mobile devices

3.1.3.2. Load Plan (Jason):

This feature provides users with step by step instructions that help them to optimally utilize the space inside of their moving truck. There are two major components, the Container Loading Algorithm, which calculates optimal space

usage, and the Instruction Screen, which displays step-by-step instructions to achieve that optimal space usage.

Commented [AB20]: Cross reference with my section

Might be able to make flow diagrams

3.1.3.2.1. Container Loading Algorithm:

This algorithm works on the back end to generate a load plan that minimizes space of the user's moving truck given a set of boxes. This Load Plan will later be displayed step by step to the user. The following requirements must be met:

1. In order to calculate optimal space usage, the Container Loading Algorithm shall require the dimensional information of the container, which is input by the user at the beginning of the Load Plan generation process. The required dimensions are:
   A. Container width in inches
   B. Container length in inches
   C. Container height in inches

Commented [AB21]: Inputs and outputs of the algorithms and then some text describing the intention of the algorithms

Some criteria should be listed like do not exceed dimensions of the truck

Could even use an input output process table

2. In order to calculate optimal space usage, the Container Loading Algorithm shall require the set of boxes to be loaded, including their width, length, and height.
   A. Boxes shall be pulled from the appropriate table in the Database as a pre-step of the Container Loading Algorithm.
3. In the event that the number of boxes to be loaded exceeds the capacity of a single container, the Container Loading Algorithm shall produce additional loads until this placement of all boxes is calculated.

4. The Container Loading Algorithm shall produce a Load Plan. The load plan will contain a set of loads, each of which will represent a truck load of boxes. Each load shall contain:

   A. The set of boxes to be loaded

   B. The sequence in which the boxes must be loaded

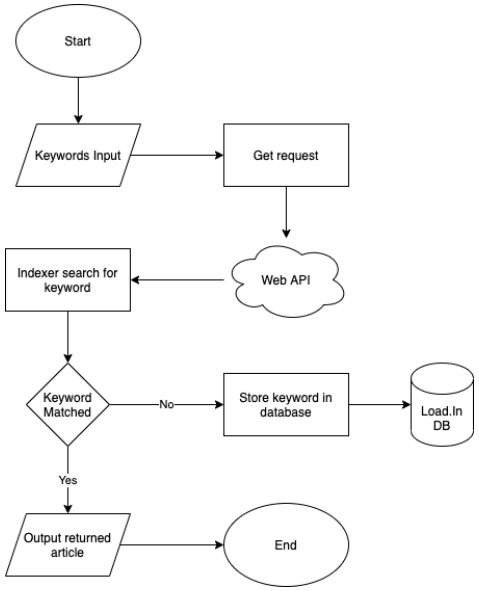
   C. The location those boxes must be placed

3.1.3.3. Expert Tips Algorithm (Lance)

The algorithm shall take in keywords based on the end users input for box content description during the move inventory phase (3.1.1.3.1). The box content description is used as the keyword to index (3.1.5.3) expert tips saved in the database. The algorithm will then return title, article, photo, and video for the

expert tip.

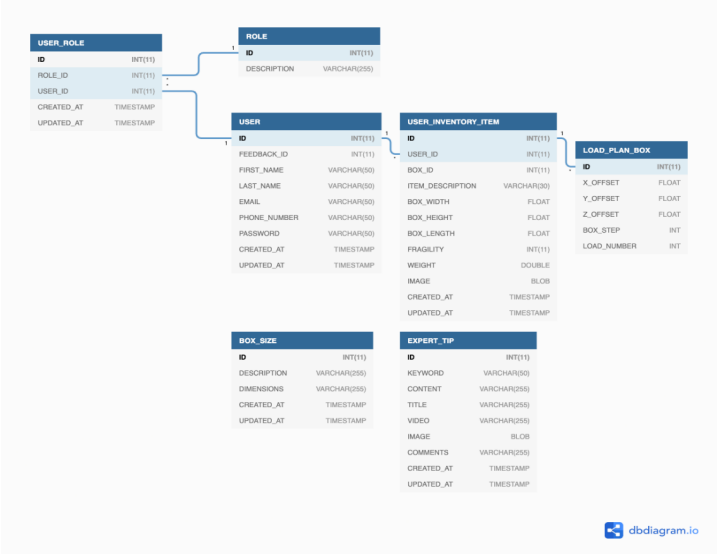
The system shall provide the functionality according to this diagram:



3.1.4. Database (Lance): The relational database shall store all persistent application data.

The following functional requirements shall be met:

1. The database shall be accessed and updated by requests from the web API.
2. The database shall have the following tables to store information regarding:



3. The database shall encrypt all data being exchanged with the web API.

3.1.4.2. User Credentials: (Byron)

The system shall provide a mechanism by which the user credentials can be

persisted and retrieved at any time for the purposes of authentication. The persisted storage shall:

1. Store the credentials for the user so that the username for one user cannot be used for another
2. Store the password in relation to the username such that when given the username, the password can be determined
3. Protect the password from individuals with database access by hashing it to obscure the password contents
4. The storage shall record the following regarding login attempts
   A. When the last login attempt was
   B. Whether the attempt was successful
   C. The quantity of unsuccessful login attempts within the configurable time window
5. Store the state of the user account (locked, unlocked)

Commented [AB22]: Convert to the table diagram

3.1.4.3. Move Inventory Storage (Chris)

The system shall provide a mechanism by which the user inventory will persist and be accessible anytime by the user. The inventory storage shall:

1. Store items saving the user input for:
   A. Description, dimensions, fragility, and weight.
2. Create a unique ID for each item stored.
3. Create a box ID for each box in a user's inventory

Commented [AB23]: Give the table a name and then refer to the table

Commented [AB24]: Describe a little more about the purpose of each id

4. Create fields for tracking the date and time when items are created or updated.

5. The storage shall discern one user's inventory from another by the user ID, linking each item to the user that added it.

Commented [AB25]: Work on this a little

Commented [AB26]: Convert to table diagram

3.1.5. Web API

3.1.5.1. Authentication (Byron)

The web API shall provide both a mechanism for authenticating users to the

android client but also to authentication the user when utilizing services for the web API. The web API shall:

1. Provide an interface method to establish the identity of the user as described in the authentication section for the Android client 3.1.1.1.
2. Provide a token in the form of a stream of bytes to the Android clients such that it shall:
   A. Be used on each request to the web API
   B. Establish the identity of the user
   C. Be resistant to tampering by digitally signing the token
3. Require that the Android client send the token to the server on each request so that the identity of the user can be established

3.1.5.2. Password Reset (Byron)

The web API shall receive a request to reset the password from the android application. It shall accomplish this in two different phases

3.1.5.2.1. Generating a token

The system shall use a token to establish that the user requesting the password is the owner of the email address on record for the user. When receiving a request for resetting it shall:

1. Confirm that the email address requested to reset the password exists as a username in the database
2. Generate a random token for the reset
3. Send the random token to the email address for the user
4. When successful at matching an email address:

A. Return a success response

5. When unsuccessful at matching

A. Return an error response

3.1.5.2.2. Resetting the password

The system shall use the token earlier to confirm the identity of the user and allow the password to change. It shall:

1. Allow the user to send the request for a new password containing:

A. The token supplied by the user

B. The new password

2. Match the user supplied token to the one generated by the system

3. When the match is successful

A. Apply the requested password

B. Return a success response to the android application

4. When the match is unsuccessful

A. Return an error back to the android application.

3.1.5.3. Expert Tips Indexer (Paul)

The web API shall provide a mechanism for indexing the mySQL database that contains the Expert Tips. The indexer shall:

1. Generate an index

A. Query the mySQL database for all expert tip records

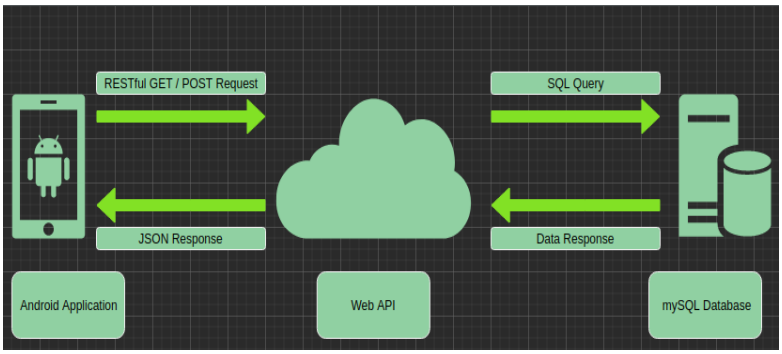
B. Create new record for all records retrieved from the query

2. Accept keyword from web API and search index

A. If match is found the expert article will be returned to web API

3.1.5.4. Services (Paul)

The web API shall provide various services for handling RESTful requests made from the Android application.



3.1.5.4.1. Data Services

Service Name	GET Purpose	GET parameters	POST Purpose	POST parameters
Users	Retrieve User account for authentication	Username, password	Add new user account	Username, password
Expert Tips	Retrieve expert tip from indexer (3.1.5.3)	Keyword	NA	NA

Inventory	Retrieve user inventory	User ID	Add new user inventory	User ID, inventory
Box Sizes	Retrieve box sizes	NA	Add new box size	Box Size
Load Plan	Retrieve user load plan	User ID	Add new user load plan	User ID, load plan

3.2. Performance Requirements

3.2.1. App Load Time (Paul)

In order to be useful, the application load time cannot take up considerable time. If the application is slow to load the user is unable to use any of its features. This will block them from making any progress on their move and makes the use of Load.In unappealing. The following requirement must be met:

3.2.1.1. The application load time shall not take more than 5 seconds from the user tapping the icon to open and prompt the user for their account information.

3.2.2. General Action Response Time (Lance)

3.2.3. Load Plan Generation (Jason):

In order to be useful, the Load Plan generation algorithm cannot take up a large amount of time, during which it would be blocking the user from making progress on their move, and ultimately making Load.In an unvaluable tool. The following requirement must be met:

3.2.3.1. The Load Plan generation algorithm shall not take more than one minute to calculate a truck load of items

3.2.4. Android Compatibility (Lance)

The system shall run on Android platform version 4.4 KitKat to be able to reach 98.1% (Android Studio Data) of devices running Android.

3.3. Assumptions and Constraints

3.3.1. Items Larger than Truck (Chris)

The system shall assume that items large than the designated truck will not be part of the load plan. The system shall display a warning message should an item larger than the truck be added to the inventory.

3.3.2. Extremely Heavy Objects (Chris)

The system shall assume objects that are above a certain weight are too heavy to be moved by the truck. The system shall display a warning message should an item be too heavy that it will not be included in the Load Plan. If an item is heavy but is within the weight limits the system shall load according to the Load Plan algorithm.

3.3.3. Dimensions of a Truck

3.3.4. Rotation of items

3.4. Non-Functional Requirements

3.4.1. Server Setup (Greg)

Load.In shall be hosted on a virtual machine running Ubuntu Server 16.04 and Docker. This server is provided by the ODU Computer Science department. The server shall allow for public access to the Web API from the Android Application.

3.4.2. Containers (Greg)

The Database (3.1.4) and Web API (3.1.5) shall run inside of Docker containers. These containers will provide environmental stability when moving the Database and Web

API from development machines to the Server (3.4.1). These containers shall run on the Server to allow for public access to the Web API from the Android application.

3.4.3. Security

3.4.3.1. Encryption (Chris)

The system shall encrypt sensitive information when it is added to the database. The system shall do this through hash functions that alter data to alter and secure it when being added to the database.

3.4.3.2. Data in transit (Byron):

The system shall provide a mechanism to encrypt the data during communication such that it cannot be intercepted by third parties who may be eavesdropping on the communication. It shall:

1. Encrypt the communication channel between the android client and the web API using SSL

3.4.3.3. Authorization (Chris)

The system shall use user ID to determine authorization for different activities and

information within the Load.In application. The system shall use user ID to restrict:

1. Inventory to only items added by the user
2. Load plan to only the plan generated for a specific user
3. Account information for a specific user
4. Edited items to be linked to the specific user

3.4.3.4. Public Facing Resources (Greg)

The Web API (3.1.5) shall be the only Public Facing Resource for Load.In. Traffic to and from the Web API shall be encrypted for security purposes. The Web API shall authenticate users before allowing access to the Web API's functions.

3.4.4. Maintainability (Lance)

The system shall have a scheduled maintenance plan to keep application performance at optional levels. The schedule maintenance plan shall perform the following:

1. Maintain project documentation when changes are made.
2. Modify project code by correcting any errors or bugs.
3. Purge old data out of the database and move it to cold storage.

3.4.5. Reliability (Byron)

The system shall provide a reasonable amount of reliability for the purposes of testing and demoing the prototype. It shall:

1. Maintain a 75% availability rating
2. Be able to recover from a catastrophic event in less than 1 day
3. Not be required to retain test or prototype data in the event of a disaster
4. Make no guarantee that any data submitted to the prototype will be retained for any definite length of period

Appendix