

# EECHELON

## Coordinated Delivery System (CDS)

### Introduction

Our country is quite infamous for its overwhelming population density. Rather than focusing on the issue which is already overwhelming, it can be utilized by looking at it from an optimistic point of view. We have come up with an idea similar to carpooling in the sense that people will be a delivery medium without being tied to any organization or company, much like work at free will if two people cooperates.

### Who are we?

We are a group of three individuals and we fall on the frustrated spectrum of Dhaka due its nagging traffic jams, weather and the hostile conditions of the roads of our beloved city. We aspire to solve some of these problems, however little it may be with an app which we hope would have a positive impact on everyday life.

We represent the company eecheleon (the meaning of the word) being “assignment” or “distribution” much like the motto of our app to assign someone a job that they could easily do and distribute the workload.

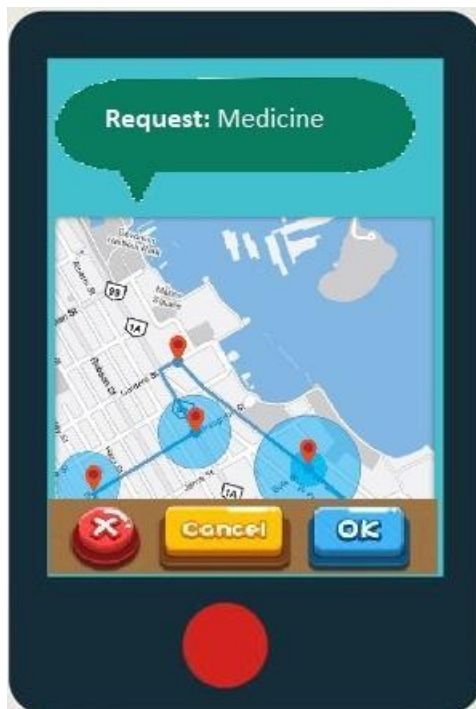
## Problem Statement

Let's assume that A and B are two locations within the densely populated mainstream cities and that a person living in location B and needs to retrieve a package from location A. To carry out this task, the person is required to make two trips, first from B to A, which prepares the person to carry out the actual task, i.e. to bring the package back to B from A. The first trip, from B to A consumes effort but has no yield, as it has no outcome, it only makes the person eligible to carry out the real work on the next trip.

This extra meaningless effort not only wastes time and energy, it also increases physical traffic which is another issue. However, due to the nature and quantity of our population, at any given time, the probability of a random person moving from a location, A to another location, B is quite decent. It would be very convenient for this person to cooperate as he has to make the trip from A to B anyway. In such a scenario, people would only be making optimal number of trips, getting rid of unnecessary trips, reducing traffic, effort and saving time in the process. Such a form of cooperation is what [eechelon.com](http://eechelon.com) is based on.

This is where we come in and utilize current technology, mainly the web and a mobile device to ease some of these hassles of everyday life. The app will be called **eechelon** and we hope that it would catch on with the users quickly. And that the meaning is simply to be coordinated, and hence the project's name: Coordinated Delivery System or CDS for short.

What we thought it would look like



The core idea of the app is to offer simplicity to the user. You want something, you say that and if someone agrees on that for a price, you have incoming delivery without moving an inch from the comfort of your house. We aim to achieve this and not waste valuable seconds tinkering with the UI. This would also enable the elderly population who generally tend not to be on the tech-savvy side to use this application as a “crutch” to order emergency commodities like medicine.

Why do we think people should invest in such a project?

At the end of the day we all want to spend time with our family, loved ones and not spend time and hours waiting on the roads after a long day at the office or university. Our application enables anyone to use a very similar concept to “torrenting” in the sense that it allows peer to peer delivery and save resources like space occupied on the roads, less fuel wasted and less time spent trying to fight the traffic which is inevitable given the current conditions of the city. This could provide job opportunities and connect people and grow a network. As for the app itself, it’ll ultimately be a good resource for collecting data that the users are willing to provide freely and that can be a good incentive.

## Mission Statement

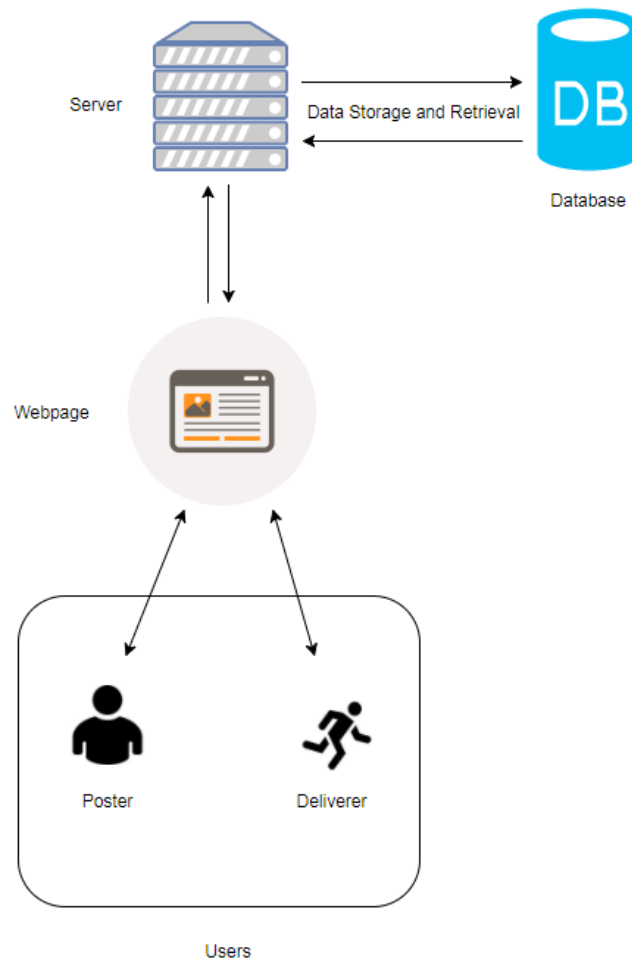
We envision a future where everyone cooperates so that nobody has to make an extra effort.

## Vision Statement

Eechelon, the company we formed has one goal is mind much like the giants of the tech world, that is to connect people with a really simple solution in context of reducing their everyday struggles on the road. We would like anyone and everyone using our service to ease the load on precious resources and impact people’s everyday lives.

We would like to enable our users and facilitate a common ground and create a stage for people to communicate and help each other. And we hope to do that with the touch of a screen!

## Architecture overview of the application



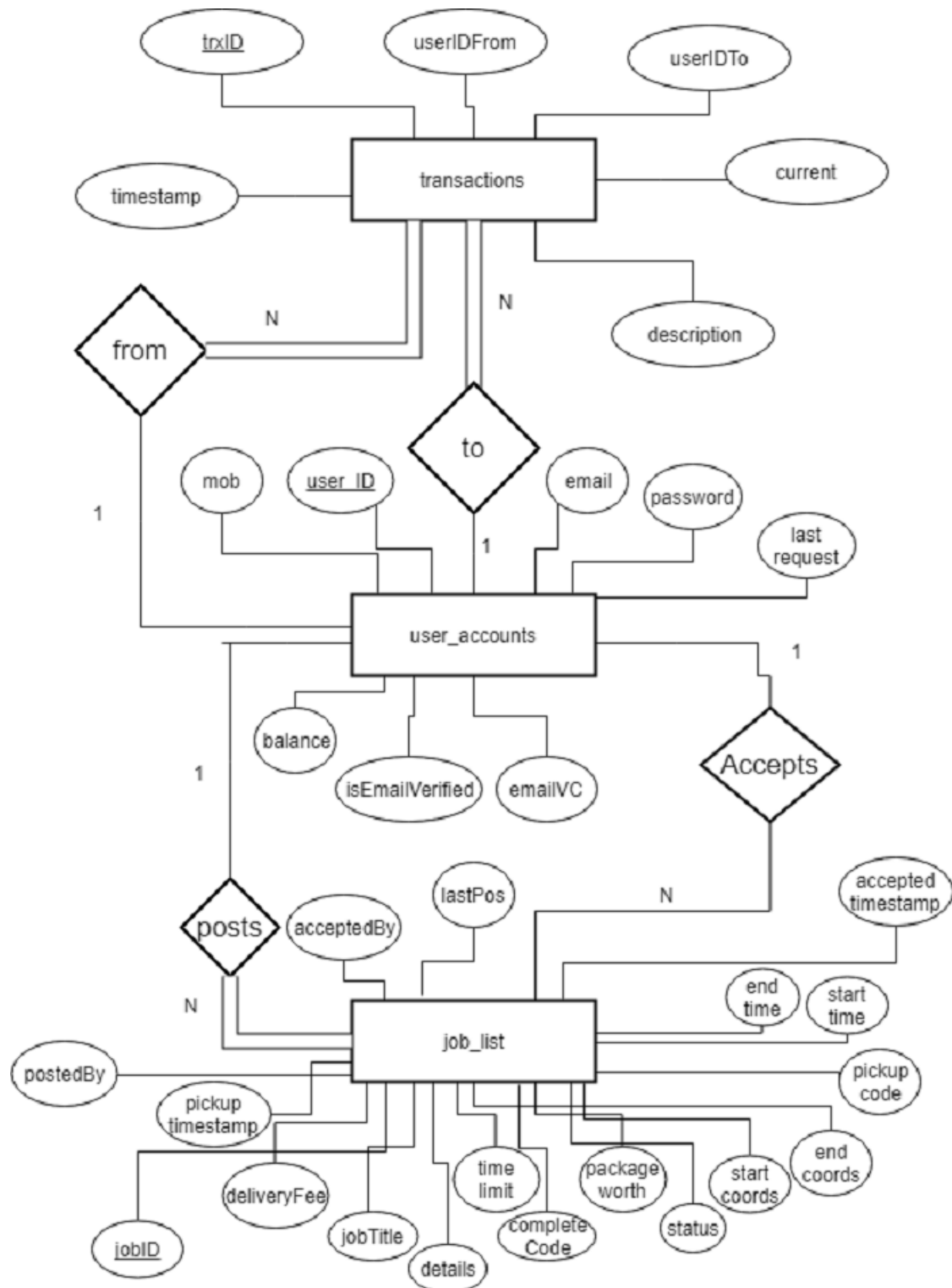
The architecture revolves around both types of people from the user's pool to make request via the application from their mobile phones. The requests are then sent to the php script running on the backend hosted on an apache server which validates the request parameters and checks the validity of the requests made. Once the requests are authenticated, the necessary information are then stored on the database and records are kept. On the frontend the users of the application are presented with appropriate screens once they choose the modes of the application.

Once the frontend data is finished picking up all the user input details, it will be automatically be sent to the server and updated according to the user's status. If the user is a deliverer and he accept a current, all of his details will be logged to the server such the time when accepted the job, the IP address and location details using Google's Maps API. All of this is done to ensure that a level of security is maintained in the faintest of possibilities that the app is being used to do something illegal. We made sure that the users get appropriate warnings if he/she misuses the app. The user will eventually be flagged if he she cancels ongoing accepted requests and won't be able to take on new jobs or post requests until the ban is revoked after reviewing the reasons from the admin panel. Once a delivery is completed or a request has been accepted, both the users are presented with QR code generate buttons which upon matching with its counterpart will enable a request to the server to enable it again review the time parameters and update the database with information regarding the completed delivery such as timestamp of the request and the delivered time, this in turn will calculate how much escrow will be awarded to the appropriate end. The users can also see current post requests from their dashboard which is generated from the database by the server and sent to the frontend to the users. An ongoing job is also updated simultaneously if they are currently in deliver mode. The balance mechanisms revolve around a digital currency as of the current app's state. It is there to ensure that the escrow system remains valid so that the system can apply the appropriate punishment if it is viable. We have not implemented the feature of porting actual cash to the digital currency due to complications such as getting access to bKash APIs in order for it to make it work.

## Technologies used

- PHP
- Apache server
- MySQL Database
- HTML, CSS, Javascript
- jQuery
- Google Maps API

## ER Diagram



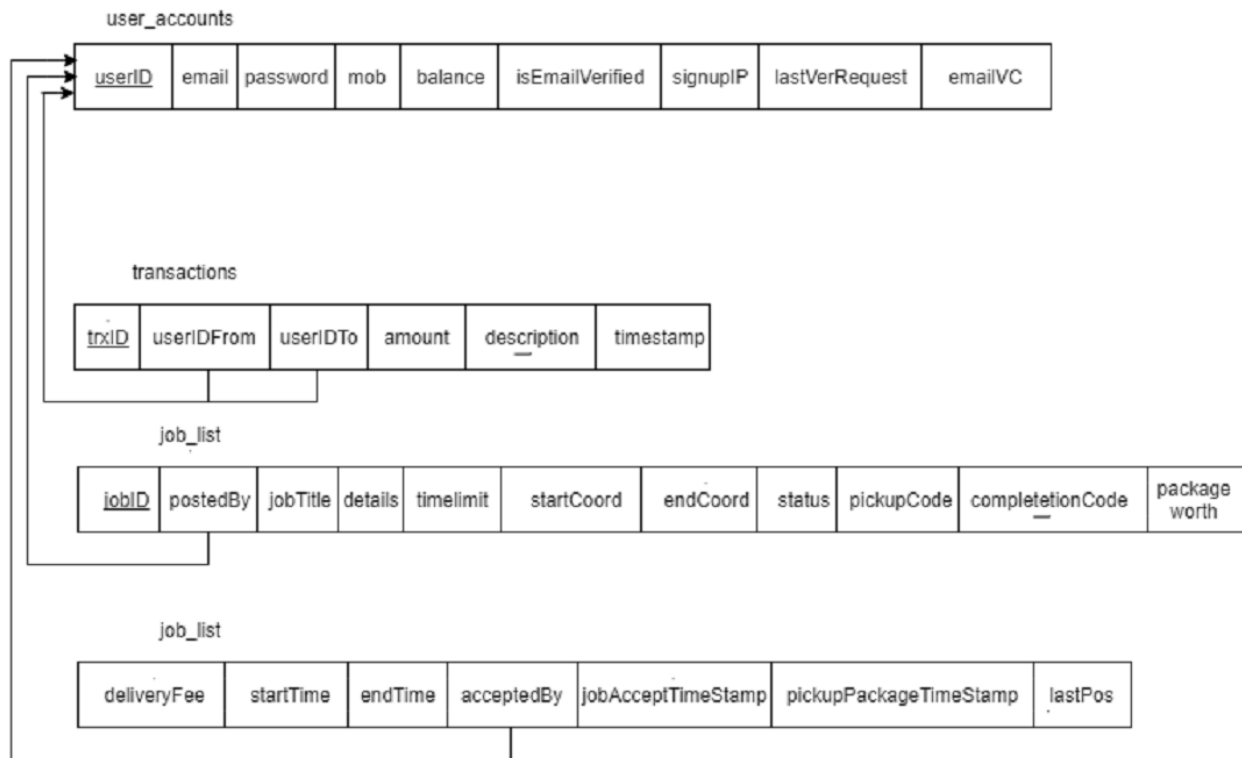
## ER Diagram Insight

A user may have taken part in more than one transaction or no transaction but a transaction always involves one 'from user' and one 'to user'. So, transaction has full participation and user\_accounts have partial participation and the relation between user\_accounts and transactions are one-to-many.

A user may post or accept more than one job or no job but a job always involves one 'posting user'. The 'accepting user' is not guaranteed since nothing ensures that a job will always be accepted. Therefore, a job may have one 'accepting user' or no 'accepting user'. So, the relation between job\_list and user\_accounts for posting has full participation on job\_list and partial participation user\_accounts. The relation is one-to-many. On the other hand, the relation between job\_list and user\_accounts for accepting has partial participation on both sides and the relation is one-to-many.



## Relational Schema



### user\_accounts

userID: unique integer value to identify user account. (primary key)

email: used for account registration and to a degree the verification of a user identity.

password: used for storing user password in sha512 hashed format for login and account security.

mob: used for storing user's mobile number to eliminate duplicate account to a certain degree.

balance: storing user's e-balance in float format.

isEmailVerified: boolean field used for storing if user have verified email or not.

signupIP: used in case of detecting any unethical practise.

lastVerRequest: used for preventing email flood from the system by continuously pressing resend verification email. A timestamp field.

emailVC: used for storing email verification code.

### transactions

trxID: to uniquely identify a transaction. (primary key)

userIDFrom: used for recording which user sent the amount.

userIDTo: used for recording which user received the amount.

amount: to store how much e-balance was sent in the transaction.

description: explaining the transaction.

timestamp: storing the timestamp when the transaction took place.

### job\_list

jobID: to uniquely identify a delivery request. (primary key)

postedBy: to store the user who requested the delivery.

jobTitle: holds the title of the delivery request.

details: holds the description and all the details of the job.

timelimit: to store the time limit in minutes.

startCoord: holds the coordinate of where to pickup. Null in case its a purchase and deliver.

endCoord: holds the coordinate where to drop the package.

status: used for storing the latest status of the delivery.

pickupCode: a random code for confirming a pickup.

completionCode: a random code for confirming the completion of the delivery.

packageWorth: for locking up balance from requester for escrow incase of purchase and deliver.

deliveryFee: the fee which the requester will pay for the delivery.

startTime: stores the timestamp of when a request has been posted.

endTime: stores the timestamp of when a delivery has been completed.

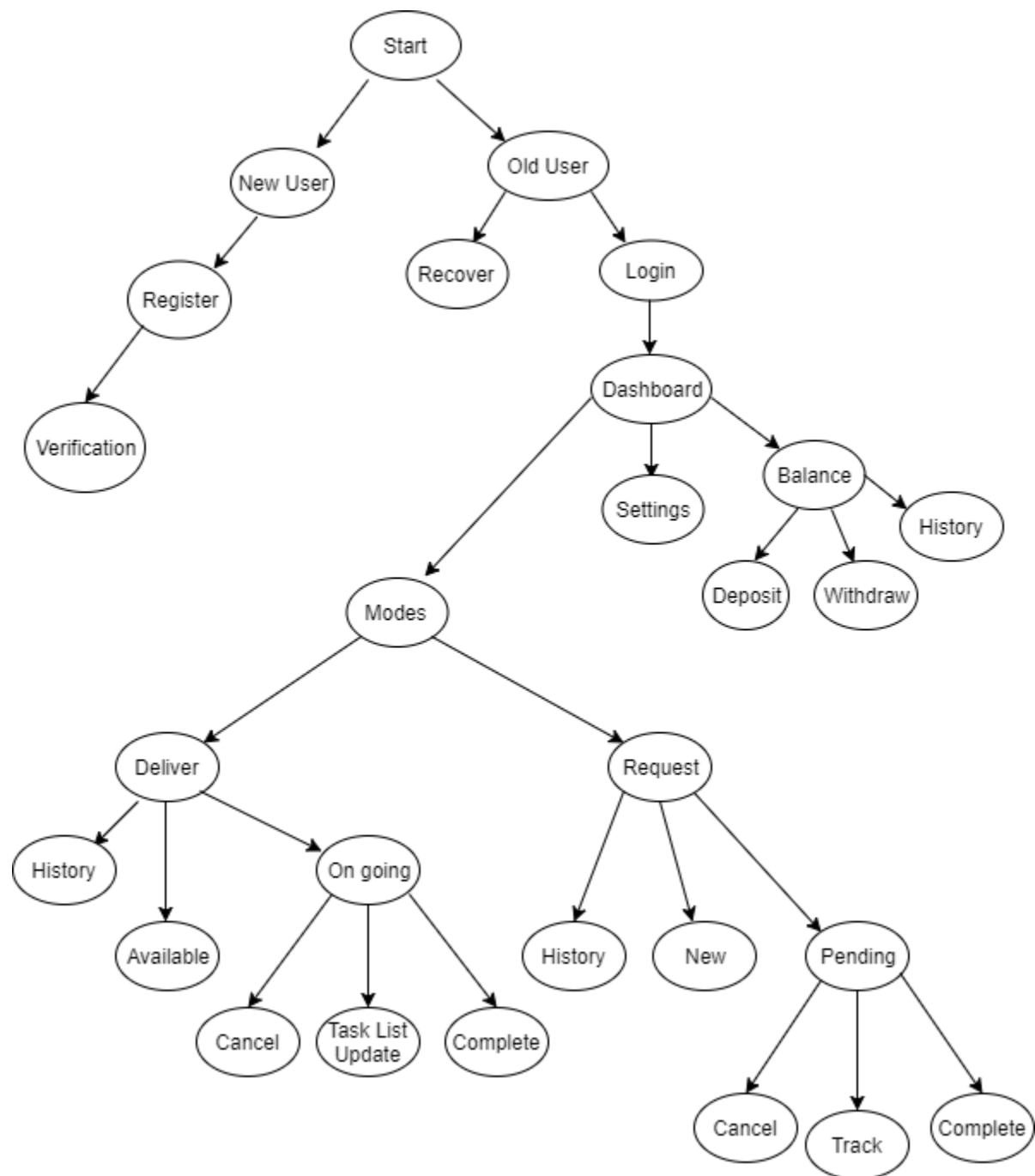
acceptedBy: the user who accepted the delivery request.

jobAcceptTimeStamp: stores the timestamp of when a delivery has been complete.

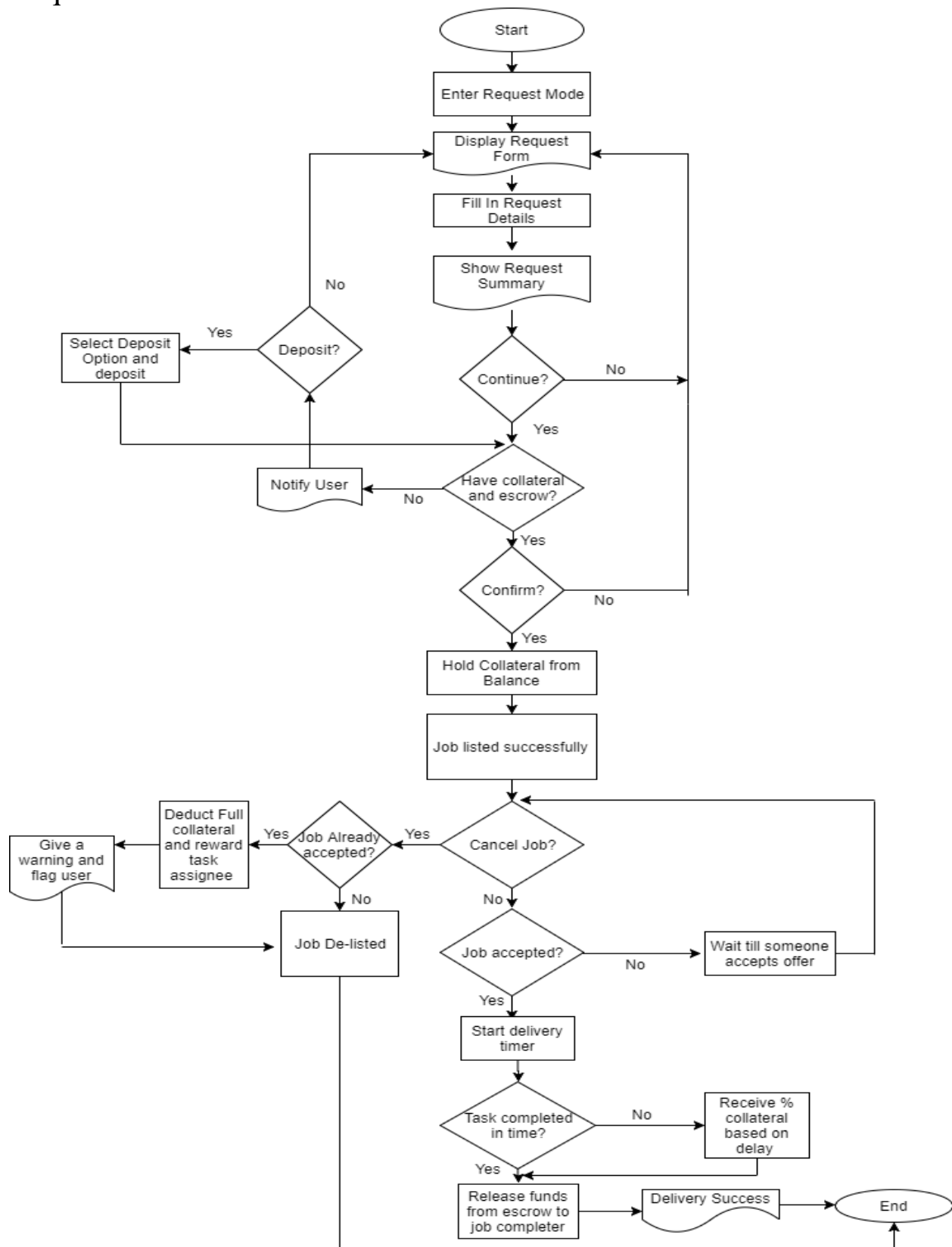
pickupPackageTimeStamp: stores the timestamp of when the delivery guy have picked up or purchased the package.

lastPos: stores the coordinate of last seen position of the delivery guy.

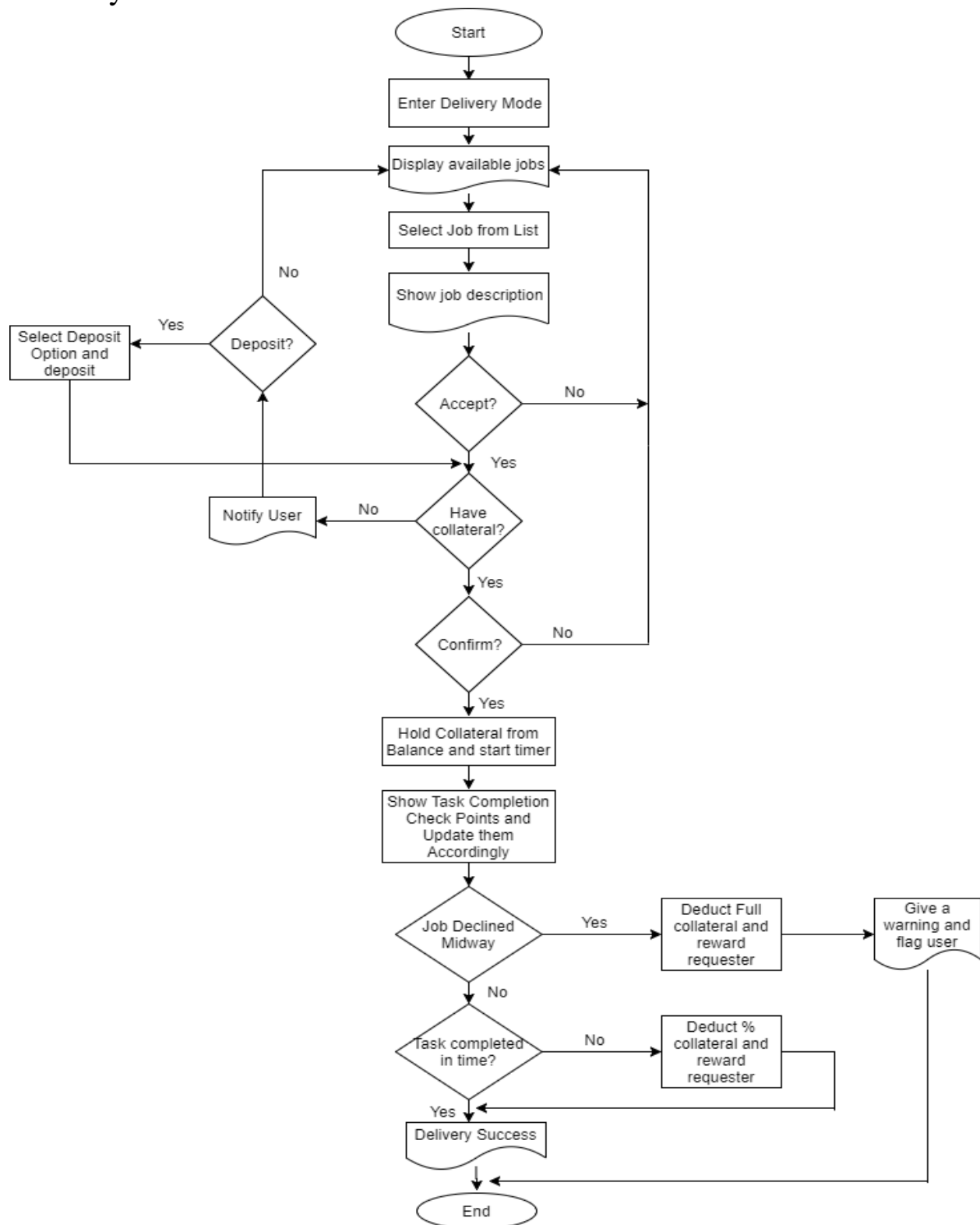
## Tree Diagram



## Request



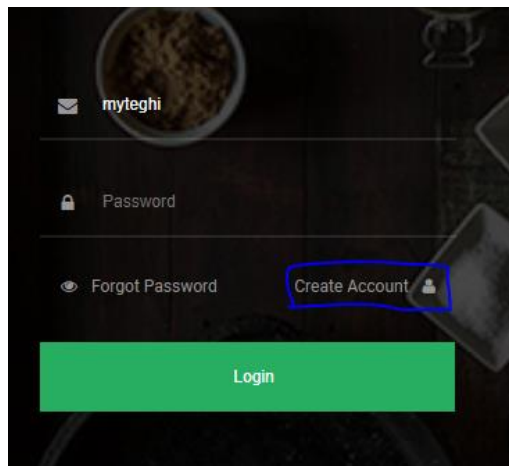
## Delivery



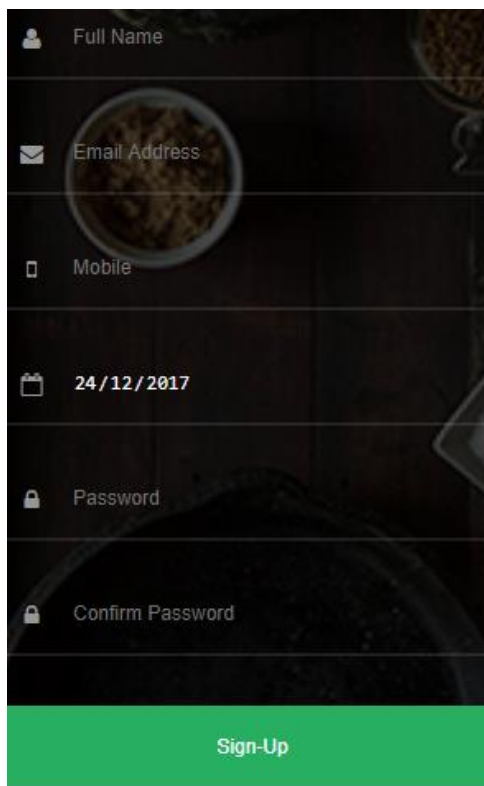
# User Guide

## How to create account:

1. Visit <https://eechelon.com>
2. Click on "Create Account"

A screenshot of a mobile application's login and registration screen. The background is dark with a faint image of a bowl of food. There are four input fields: an email field with the placeholder 'myteghi', a password field, a 'Forgot Password' link with an eye icon, and a 'Create Account' button with a person icon. A green 'Login' button is at the bottom.

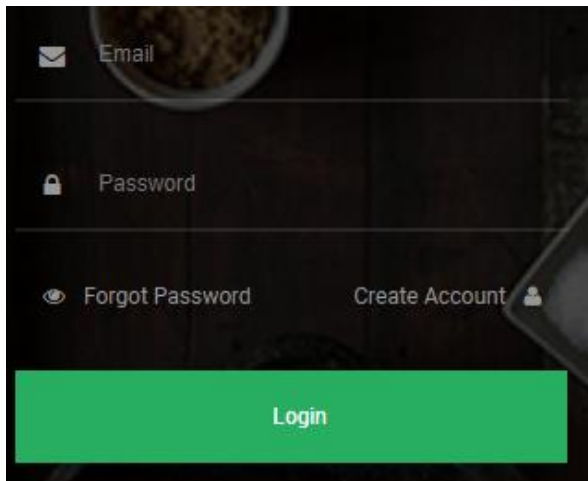
3. Fill up form and click on the "Sign-Up" button

A screenshot of a mobile application's sign-up form. The background is dark with a faint image of a bowl of food. There are six input fields: 'Full Name' with a person icon, 'Email Address' with an envelope icon, 'Mobile' with a phone icon, a date field with a calendar icon and the value '24/12/2017', 'Password' with a lock icon, and 'Confirm Password' with a lock icon. A green 'Sign-Up' button is at the bottom.

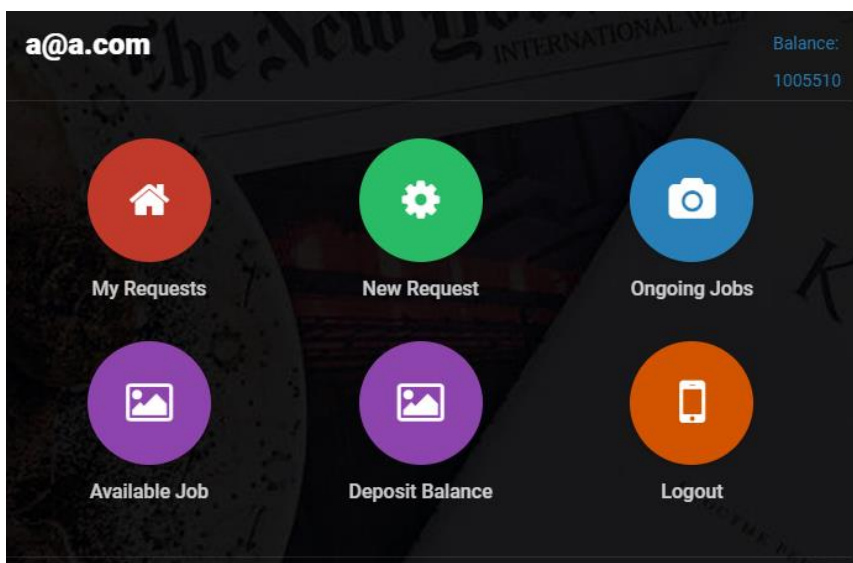
4. A verification email will be auto sent by the system to your provided email address.
5. Click on the link inside the email to verify your email and complete the signup.

### How to login to the system:

1. Visit <https://eechelon.com>
2. Enter the email and password you used to signup your account and click on the "Login" button

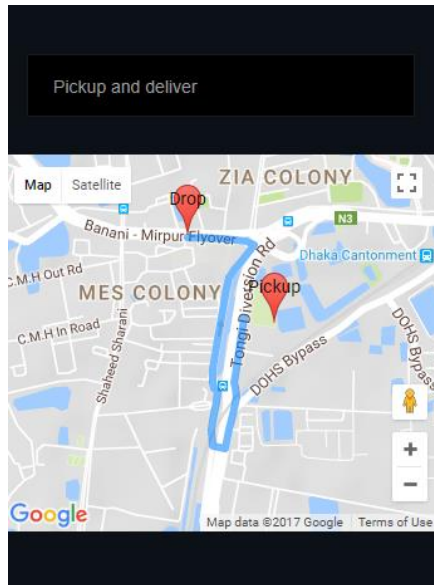


3. Upon entering the correct login info you will be taken to the dashboard of the system



## How to request a pickup delivery:

1. Upon login, click on the New Request button from the dashboard.
2. Select delivery type as Pickup and deliver
3. Select the drop and pickup positions from the map



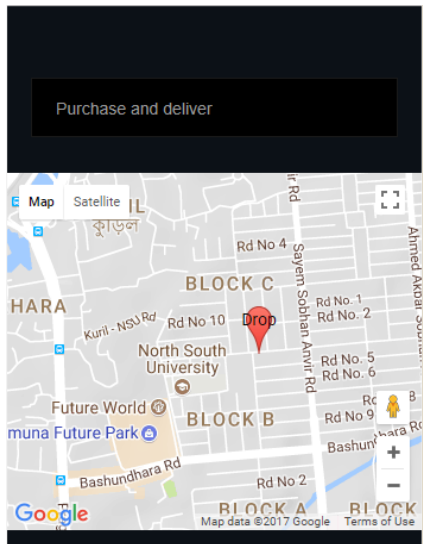
4. Fill in the form below the map appropriately and click on the "Request" button.

A screenshot of a web form for requesting a pickup and delivery. The form is set against a dark blue background. It contains several input fields, each with a label and a '(required)' note. The fields are: 'Job Title:', 'Description:', 'Time Limit:', 'Package Worth:', and 'Delivery Fee:'. Each field has a corresponding light blue input box. At the bottom of the form is a large green button with the white text 'Request'.



## How to request a purchase delivery:

1. Upon login, click on the New Request button from the dashboard.
2. Select delivery type as Purchase and deliver
3. Select the drop positions from the map



4. Fill in the form below the map appropriately and click on the "Request" button.

Job Title: (required)

Description: (required)

Time Limit: (required)

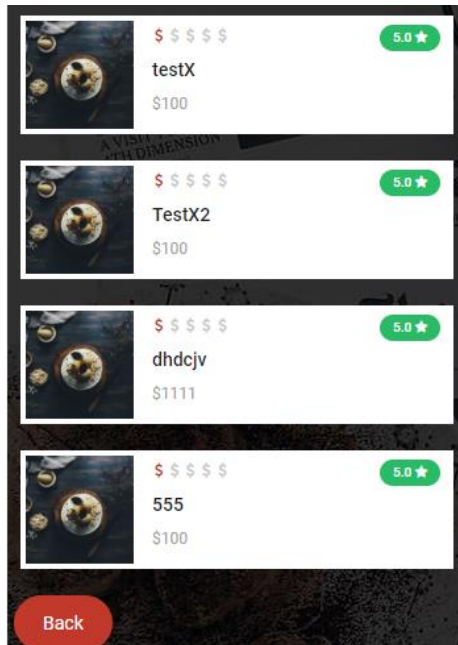
Package Worth: (required)

Delivery Fee: (required)

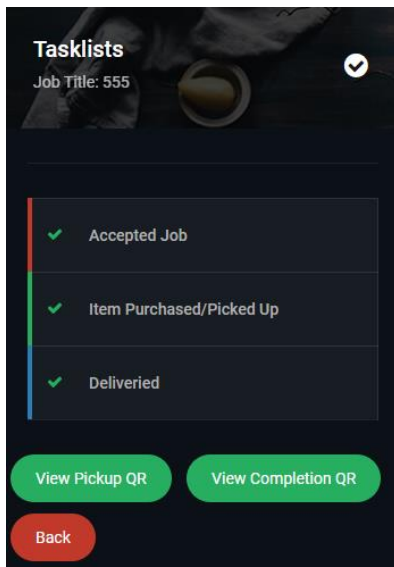
Request

## How to track a delivery:

1. Upon login, click on the My Requests button from the dashboard.
2. Select the delivery you want to track.

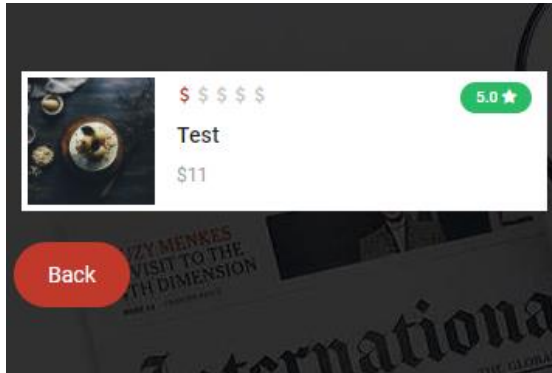


3. From the next window, you will be able to track your package.

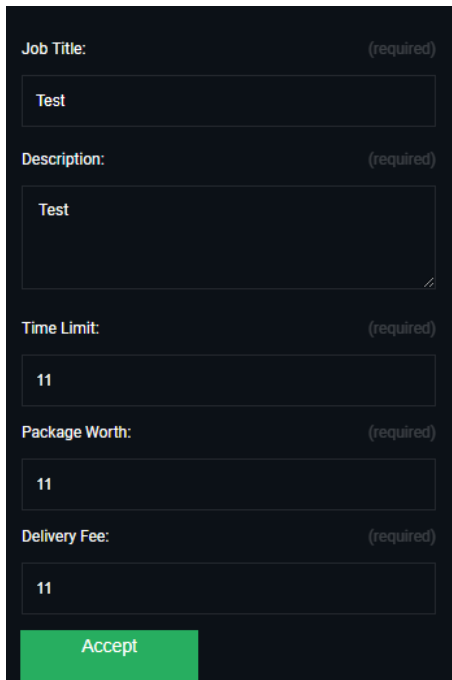


## How to accept a delivery request:

1. Upon login, click on the Available Job button from the dashboard.
2. Select a job which seems interesting to you.



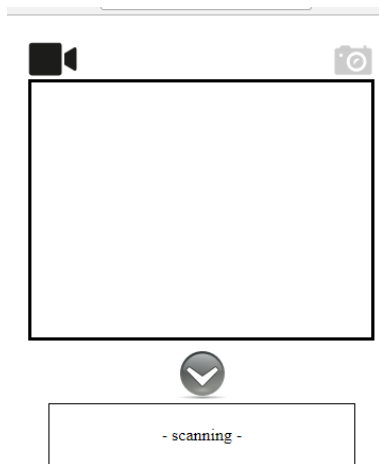
3. Read the details and click on the "Accept" button.

A screenshot of the job details form in the CDS interface. The form is dark-themed and contains several input fields. The first field is labeled "Job Title:" and has the text "Test" entered. The second field is labeled "Description:" and has the text "Test" entered. The third field is labeled "Time Limit:" and has the text "11" entered. The fourth field is labeled "Package Worth:" and has the text "11" entered. The fifth field is labeled "Delivery Fee:" and has the text "11" entered. At the bottom of the form, there is a green button with the text "Accept".

## How to update and complete a delivery task list:

1. Upon login, click on the Ongoing Jobs button from the dashboard.
2. Next click on the job who's task list you want to update.
3. For a Purchase and deliver, click on the Item Purchased/Picked Up button.

-If its a Pickup and deliver then after clicking on the button a QR code scanner will show up



-Ask the person from where you will pick up the package to show the QR code and scan it.

-After the scanner reads it, you will be auto redirected to your list and the Purchased/Picked Up will be check market to inform the requester about the pickup status.

4. To complete the delivery, you have to ask the requester to show the completion QR code. You will have to click on the "Delivered" button from the tasklist and scan that QR code from your QR code reader.
5. Upon reading the QR code, the system will auto update everything and release the escrows and fees.

**How is the escrow handled:**

1. Requester locks up an amount of system currency equivalent to the price of the item he/she wants to request for purchase and deliver while posting the job.
2. If for any reason the requester denies the delivery man's purchase and put him at a loss, the escrow will be released to the delivery man to compensate for the loss. This will need manual verification from the admin's side upon verifying various factors such as location data, evidence, job details, etc.

**How is the fine handled:**

1. Both the requester and the delivery man lockup their system currency equivalent to the delivery fees while a job is accepted.
2. If the delivery man doesn't deliver the package in time then there will be a fine imposed on him by releasing his locked-up fees to the requester. This is imposed so that quality delivery is maintained.

## Conclusion

We do hope that our app makes a positive impact on the everyday lives of people however small it may be. The fact that it is completely free means that people can leverage out of this make a small fortune if they are without a job. We also envision to make the app more intuitive, and more user friendly to make it seamless in what they can do with the application and in turn have a positive impression upon the software. The vision of our app still converges with the name of the map that we hope that the future is a better place and that people lend a helping hand at times of need.

# Appendix

## References used

<http://php.net/manual/en/index.php>  
<https://httpd.apache.org/docs/>  
<https://developers.google.com/maps/documentation/javascript/>  
<https://docs.phpmyadmin.net/en/latest/>  
<http://api.jquery.com/>  
<http://bootstrapdocs.com/v3.0.3/docs/css/>  
<https://dev.w3.org/html5/html-author/>  
<https://www.draw.io/>

PROJECT TITLE					echelon.com				COMPANY NAME		echelon																								
PROJECT MANAGER					MEK				DATE		31/10/2017																								
TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PERCENTAGE OF TASK COMPLETE	PHASE ONE												PHASE TWO																	
						WEEK 1				WEEK 2				WEEK 3				WEEK 4				WEEK 5				WEEK 6				WEEK 7					
						M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
Project Proposal																																			
Introduction	Andalib	17/10/17	24/10/17	7	100%																														
Task Breakdown	Dipanzan	24/10/17	28/10/17	4	100%																														
Task Distribution	Adib	29/10/17	30/10/17	1	100%																														
Development																																			
Database Schema	Adib	31/10/17	5/11/17	5	100%																														
User Registration	Andalib	31/10/17	5/11/17	5	100%																														
Account Verification & Recovery	Adib	31/10/17	5/11/17	5	100%																														
Login & Registration UI	Dipanzan	31/10/17	5/11/17	5	100%																														
Development																																			
Dashboard UI	Dipanzan	9/11/17	12/11/17	6	100%																														
Account Settings	Andalib	9/11/17	12/11/17	6	100%																														
Balance System	Adib	9/11/17	12/11/17	6	100%																														
Development																																			
Job Posting + History	Andalib	13/11/17	20/11/17	7	100%																														
Google Map - Location	Adib	13/11/17	20/11/17	7	100%																														
Job - UI	Dipanzan	13/11/17	20/11/17	7	100%																														
Collateral & Escrow System	Adib	13/11/17	20/11/17	7	100%																														
Development																																			
Google Map - Tracking	Adib	20/11/17	27/11/17	7	100%																														
Task List + Tracking	Andalib	20/11/17	27/11/17	7	100%																														
Task + Tracking UI	Dipanzan	20/11/17	27/11/17	7	100%																														
Jobs Pending & History	Andalib	20/11/17	27/11/17	7	100%																														
Development																																			
Job List + Accepting	Andalib	28/11/17	30/11/17	2	100%																														
Delivery Completion Verification	Adib	28/11/17	30/11/17	2	100%																														
Delivery UI	Dipanzan	28/11/17	30/11/17	2	100%																														
Admin Panel	Everyone	1/12/17	10/12/17	9	100%																														
Testing																																			
UI + Maps + Functions	Everyone	10/12/17	17/12/17	7	100%																														