UNIVERSITY OF THE WITWATERSRAND, SCHOOL OF ELECTRICAL AND INFORMATION ENGINEERING

Software Engineering: ELEN4009 Project 13 Fleet of Delivery and Pick-up Vehicle Management System Project Report

Front End
Sheena Philip (545819),
Kessigan Subramanium (573799),
Back end
Linda Khumalo (560895),
Phumzile Dlwathi(545290)

11 April 2016

CONTENTS

1 INTRODUCTION

1.1 Problem Statement

CourierJZ is a courier company that is based in the Gauteng region and has one depot in Johannesburg which is responsible for all its operations in the area. Courier JZ currently uses an inefficient, time consuming, tedious system which is not centralised or automated to manage their fleet of delivery and pick-up vehicles. This means that parcels are not being delivered in the shortest time possible since calculations and allocation of resources in the company is performed manually, making the company less competitive. In addition since the system is not automated and centralised on the web, clients enquiring about the process and getting information and monitoring their parcel becomes tedious, giving the customer a bad experience which ultimately gives the company an unfavourable brand image.

The company is tasked with developing a complete back and front end system to replace the paper system currently in use. The system must be able to log packages for collections and allocate drivers to pick them up and return them to the depot for processing. The system will also then need to efficiently sort the packages for delivery, with the shortest delivery time being prioritised.

CourierJZ should pick up and deliver parcels from and to its' respective customers in the shortest time and in the most efficient and least troublesome/interfering way possible. A parcel should also be able to be tracked from it pick-up point to its delivery point, allowing the company to have better control over the system and allowing customers to be better informed about their parcel.

This system will provide a web interface that allows senders to request and pay for deliveries while also allowing receivers to track their packages. The system will also provide an online interface that allows the workers easy access to the company's information when they log on to the website. The internal processes will all be managed by the companies stakeholders that have their own dashboards that correspond to the role they play in the company.

1.2 Problem Objectives

- To design, implement and test a web-based management system for a fleet of delivery and pick-up vehicles
- The website must serve as a platform to integrate the various activities of the personnel involved in the day to day running of the business
- The website must automate previously time-consuming activities resulting in a more efficient company.
- The website must act as a repository to store all the data associated with the business activities.

- The website must serve as a more convenient and efficient way for a client to be informed about the services available and to track their parcel.

1.3 STAKEHOLDERS

The following are the stakeholders involved in the project:

- Four software developers
- Driver
- CEO
- Clients
- Operational officer
- Fleet assignment manager
- Vehicle maintenance manager
- System administrator
- Account officer

1.4 PROJECT DELIVERABLES

- Tracking of parcels
- Tracking of vehicles
- Design shortest distance route algorithm
- Determine the vehicle certain packages go into
- Allow a driver to choose a vehicle and a route
- Determine date and time of package pickup
- Determine date and time of delivery
- Store payment information
- Store client details
- Package storage and package identification
- Send automated email notifications to clients
- Inventory management

2 Design

2.1 Existing Solutions

DHL is a globally known delivery company that delivers packages both locally and internationally. They have two options when delivering packages, the customer can either request online for DHL to go to the customer's location and pick up the package for delivery or the customer can go to one of DHL's many branches in order to drop off the package. The site offers the customer a choice of many delivery options and the estimated price for that option, these options have various prices depending on when the delivery is expected to be made [1].

Aramex is a similar delivery company to DHL and allows the customer to request an online pick up or a store-to-door option which lets customers drop off their packages at local retailers like pick n' pay. The websites of both these companies have homepages packed with information and the relevant links, the pages seem crowded at first but allow the customer to view all option before selecting one. These sites also show price, delivery estimations and allow senders or receivers to track their packages. The websites only gives information and views for what the customer would need and has no indication of the internal working of the company, e.g. how drivers see package details or how the packages are routed to their destinations. [2]

Fastway is another global courier service. They offer many of the services that are offered by DHL. In addition the company offers a service called the Fastlabel automatic label printing. This is a system that allows a user to print the labels required to be stuck on to the packages at home. This reduces dispatch labour requirements which overall reduces the cost of the service. Fastway's website also allows companies to make their service an API on the company's respective websites which helps market the service and improves the user experience. Fastway's website is simple, non-crowded and has an effective use of pictures in order to explain certain concepts and navigate a client around [3].

Telogis is an award winning , fleet management software developing company. This company has developed one of the best softwares' for fleet management. The system features that stood out were: the real time tracking for both customer and the fleet mananger, Progress against planned routes, fuel usage and an estimated cost that the company will incur for a certain delivery, as well as safe driving features for the driver [4].

I.D Systems is a company with its headquarters in the US, it produces wireless solutions for industrial trucks, rental vehicles and transportation assets. The solutions they produce fall into three areas: industrial vehicle management, transportation asset management and rental fleet management. Of particular interest, are the transport asset management solutions. They implement solutions which include using a cargo sensor to monitor loads, motion sensors to monitor trailer movements and temperature sensors to monitor temperatures in the vehicles [5]. This use of extra sensors apart from the information entered by users of the system ensures that the system provides comprehensive monitoring of the vehicles and can allow dynamic allocation of packages and routes to be done as the system is aware of any changes happening in the vehicles. It also allows the system to keep track of the state of the vehicle making maintenance and maintenance data collection easier. These are features that can be added to future iterations of the vehicle management system in development.

Ichoua et al. [6] compares different ways of implementing a dynamic approach to routing where the result is a minimum cost route starting and ending at a central location and implement a modified version of existing algorithms. Examples of approaches studied include include carrying out parallel implementation of the tabu search algorithm on several servers with the aid of adaptive memory. In this case routes that have been traversed and the current destination are fixed and cannot be changed. An optimization of dynamic routing algorithms that allows the current destination to be changed is then implemented. This occurs when a new request is located in the vicinity of the vehicle. To carry this out planned routes and the new request are used as input to the optimization algorithm. The output is new planned routes which include the new request. The current destination may or may not be altered in this output. However, it is important to note that the optimization procedure must end before the vehicle can reach its current destination. This results in a decrease in the total distance travelled and the number of unserved customers when compared to algorithms where the current destination is unchanged.

Abbaspour et al. [7] considers a multi modal path as a path to most likely introduce savings in time and cost. Such an approach, can be applied to the current problem by viewing each mode of transportation as a different vehicle in the current work. Instead of a package being transported in a single vehicle from start to its destination, it may be transported via different vehicles based on the implementation of a multi vehicle algorithm. According to [8], for a road network with approximately 45 million connections Dijkstra's algorithm would take almost 10 seconds which is not acceptable in practical applications. A combination which involves the use of Dijkstra's algorithm, landmarks, arc flags and contraction is then proposed as it computes faster therefore is acceptable for practical applications.

2.2 System Architecture

Figure 2.1 shows the architecture selected for the implementation of the CourierJZ project. The architectural pattern implemented is the (Model-View-Controller) architecture. As seen from the figure the postgreSQL database has been utilised. Python is the language of choice selected for interfacing the database with processing algorithms. The view is brought to life using HTML, CSS and JavaScript and also incorporating Javascript libraries such as the Google visualization API. The controller has been implemented via the web framework Django selected because it is a python based framework. Django acts as a server to facilitate communication between the view and the model.

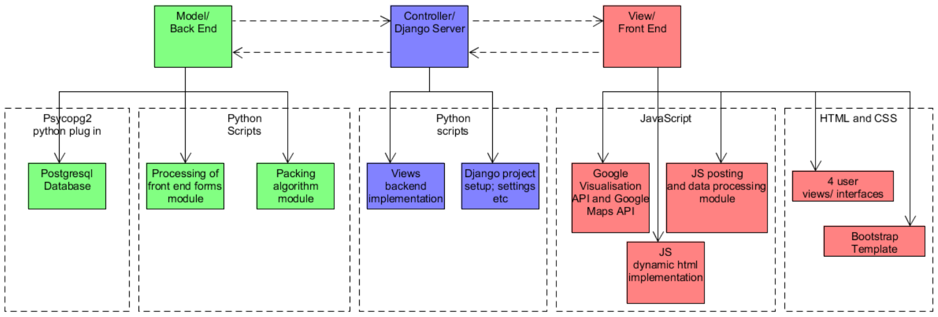

Model/ Back End
Controller/ Django Server
View/ Front End
Psycopg2 python plug in
Postgresql Database
Python Scripts
Processing of front end forms module
Packing algorithm module
Python scripts
Views backend implementation
Django project setup; settings etc
JavaScript
Google Visualisation API and Google Maps API
JS posting and data processing module
JS dynamic html implementation
HTML and CSS
4 user views/ interfaces
Bootstrap Template

Figure 2.1: The system architecture

2.3 System Overview

Figure 2.2 shows an overview of the entire system. It shows how both the front and back end systems interact with each other to form a cohesive system.

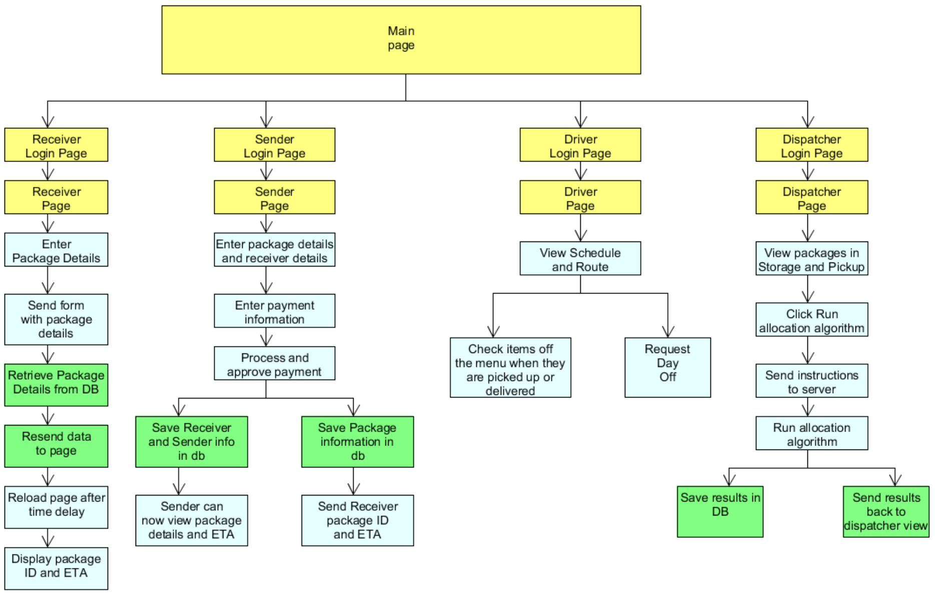
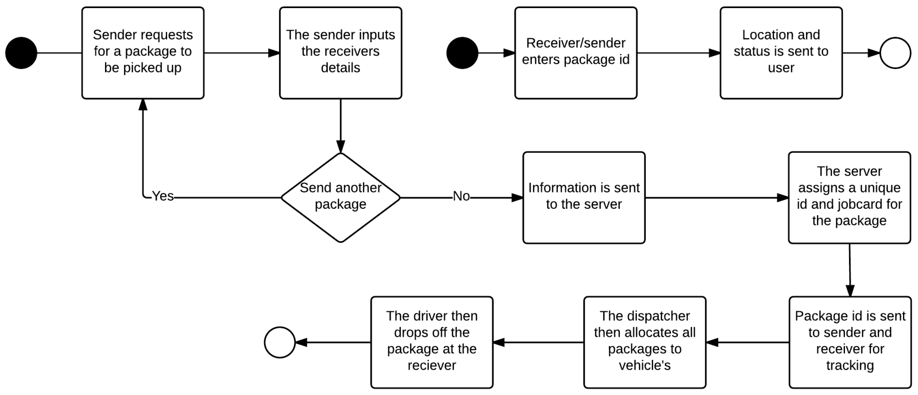

Main page
Receiver Login Page
Receiver Page
Enter Package Details
Send form with package details
Retrieve Package Details from DB
Resend data to page
Reload page after time delay
Display package ID and ETA
Sender Login Page
Sender Page
Enter package details and receiver details
Enter payment information
Process and approve payment
Save Receiver and Sender info in db
Sender can now view package details and ETA
Save Package information in db
Send Receiver package ID and ETA
Driver Login Page
Driver Page
View Schedule and Route
Check items off the menu when they are picked up or delivered
Request Day Off
Dispatcher Login Page
Dispatcher Page
View packages in Storage and Pickup
Click Run allocation algorithm
Send instructions to server
Run allocation algorithm
Save results in DB
Send results back to dispatcher view

Figure 2.2: The overall system design

Figure 2.3 shows the package life cycle from the time the sender requests the package to be picked up to the time the driver delivers it at its' destination.

2.4 Front-end

Proper design is important so that teething problems can be avoided which results is a more streamlined and efficient implementation process. The design process also allows a devel-



Sender requests for a package to be picked up
The sender inputs the receivers details
Receiver/sender enters package id
Location and status is sent to user
Yes
Send another package
No
Information is sent to the server
The server assigns a unique id and jobcard for the package
The driver then drops off the package at the reciever
The dispatcher then allocates all packages to vehicle's
Package id is sent to sender and receiver for tracking

Figure 2.3: The package life cycle

oper to foresee problems before they occur, therefore facilitating upgrades. The first step in the design process is to determine what functionality each user dashboard should preform. The second step would be to use the information found in step one, in order to create a visual representation in the form of dashboards. The final step in the design process is to edit and analyse the dashboards, making sure that there is a good flow, that the content is correct and that the format is correct, ultimately lending to a good customer experience.

The aim of the first step is to correctly identify all the users and their expections from the system. Since each of these users also needs to interact with the system, it is important to identify their goals and give them a means to do this. From this identification, the dashboards can then be designed in a way in which the important information is easily accessible and always visible. By looking at the inputs and outputs of each web-page it is easy to determine whether the functionality defined at the beginning of the process has been achieved. Table 2.1 summarises this process.

Once the data is collated,the dashboards can then be designed. Each dashboard design will be described below.

The sender view, as seen in Figure 2.4,is used when a package needs to be sent. The left side of the page allows a user to input the dimensions of a package and the address of the pick-up and drop off points. The right hand side of the page allows the sender to pay for the service and view details of pick-up and delivery times. The information is sent to CourierJZ on submission.

The receiver views, as seen in Figure 2.5 and Figure 2.6, is used when the receiver wants to track the package being sent to them. The receiver enters the tracking ID and receives the location and status on submission.

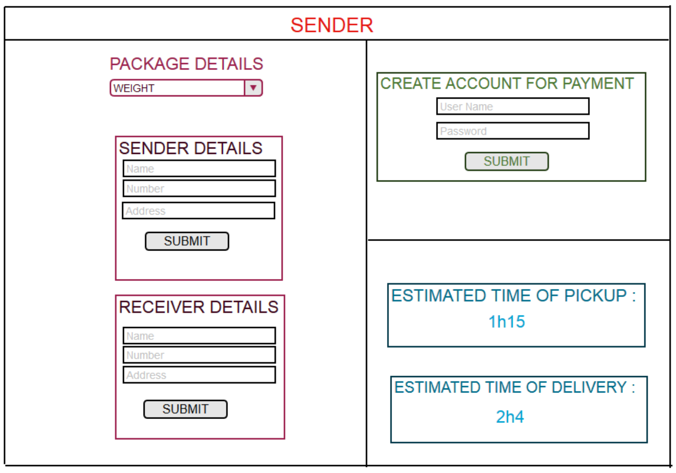


Figure 2.4: The sender view

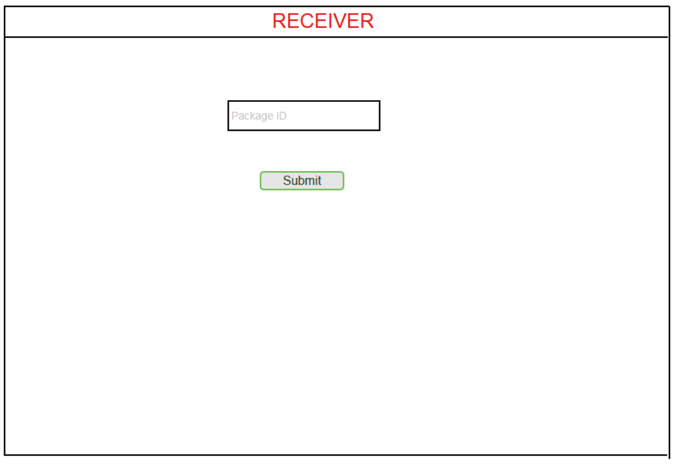
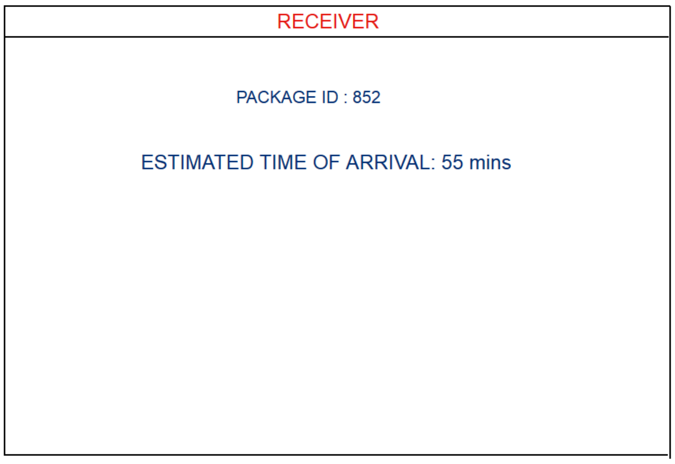


Figure 2.5: The receiver before view

The driver view,as seen in Figure 2.7, shows a map of the current location of a driver and the destination point of the current package being delivered or picked up. These destination points change as each pick-up/delivery is made. The contact details and estimated time of arrival of each client is also displayed on screen. This information is essential in order for the driver to perform their duties.

The dispatcher view, as seen in Figure 2.8, provides details on both packages in storage and requested for pick-up. The run-allocation algorithm button performs scheduling for items in storage to be dispatched and schedules the items for pick-up.

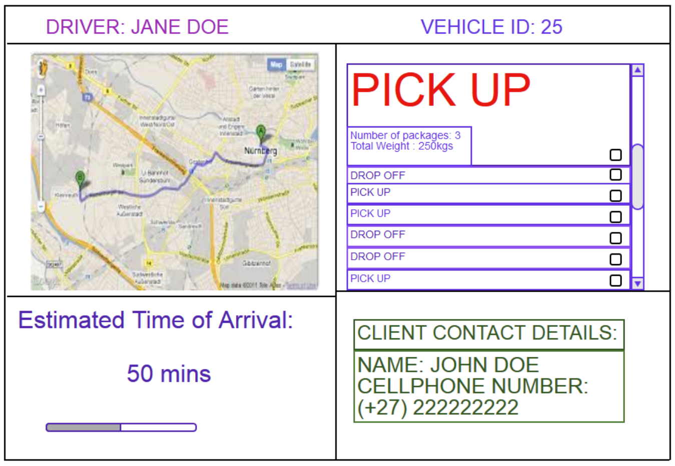
The CEO view, as seen in Figure 2.9, allows the CEO to request various reports summarising


RECEIVER

PACKAGE ID : 852

ESTIMATED TIME OF ARRIVAL: 55 mins

Figure 2.6: The receiver after view


DRIVER: JANE DOE

VEHICLE ID: 25

PICK UP

Number of packages: 3
Total Weight : 250kgs

DROP OFF
PICK UP
PICK UP
DROP OFF
DROP OFF
PICK UP

Estimated Time of Arrival:

50 mins

CLIENT CONTACT DETAILS:
NAME: JOHN DOE
CELLPHONE NUMBER:
(+27) 222222222

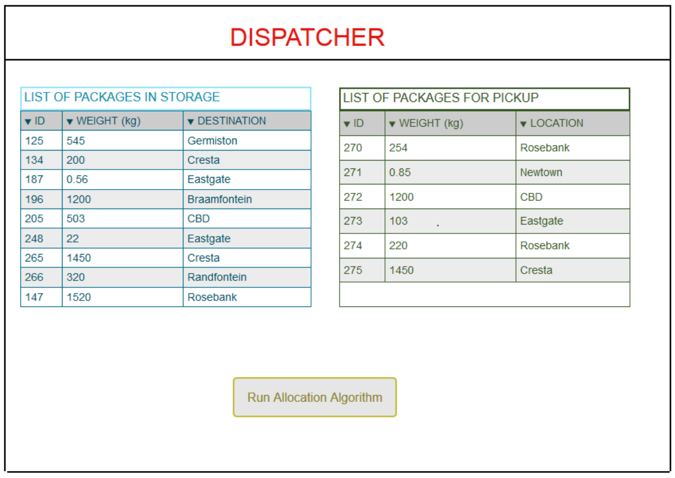
Figure 2.7: The driver interface

the operations of CourierJZ and view them on the right side of the page.

manager

The operational officer view, as seen in figure 2.10 allows the officer to respond to calls from clients and drivers. The officer can enter either the truck id or the tracking id in order to receive the relevant information on it.

The system administrator view, as seen in Figure 2.12, allows the manager to make updates to the system. The manager can start,stop and select a file from which a update must occur. The manager can also view a list of when back-ups on the system were performed and request that a back-up be currently performed.



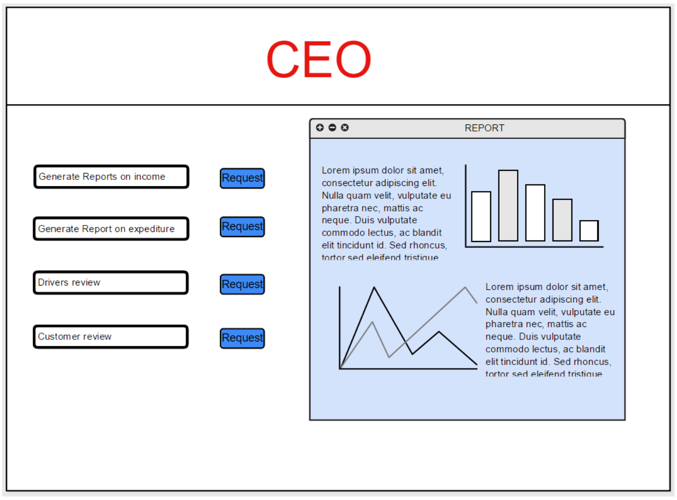
LIST OF PACKAGES IN STORAGE

▾ ID	▾ WEIGHT (kg)	▾ DESTINATION
125	545	Germiston
134	200	Cresta
187	0.56	Eastgate
196	1200	Braamfontein
205	503	CBD
248	22	Eastgate
265	1450	Cresta
266	320	Randfontein
147	1520	Rosebank

LIST OF PACKAGES FOR PICKUP

▾ ID	▾ WEIGHT (kg)	▾ LOCATION
270	254	Rosebank
271	0.85	Newtown
272	1200	CBD
273	103	Eastgate
274	220	Rosebank
275	1450	Cresta

Figure 2.8: The dispatcher view



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique

Figure 2.9: The CEO view

The accounts officer view,as seen in Figure 2.13, allows the accounts officer to select a time period and then export the accounts information to an accounting software such as PASCAL in order to perform the necessary inspections or calculations.

As can be seen from the web designs, the information is placed in a logical order that gives the user an intuitive experience. The webpages only provide the necessary information for the task and does not add excess information. The webpages also avoid the use of contrasting colours and instead used complementary colours to provide a ergonomic user experience. The colours were restricted to blue, purple, red and some grey. A white background was cho-

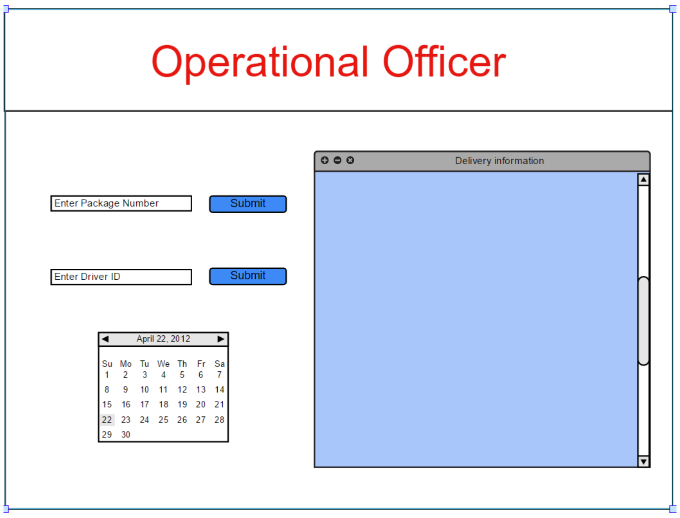


Figure 2.10: The operations officer view

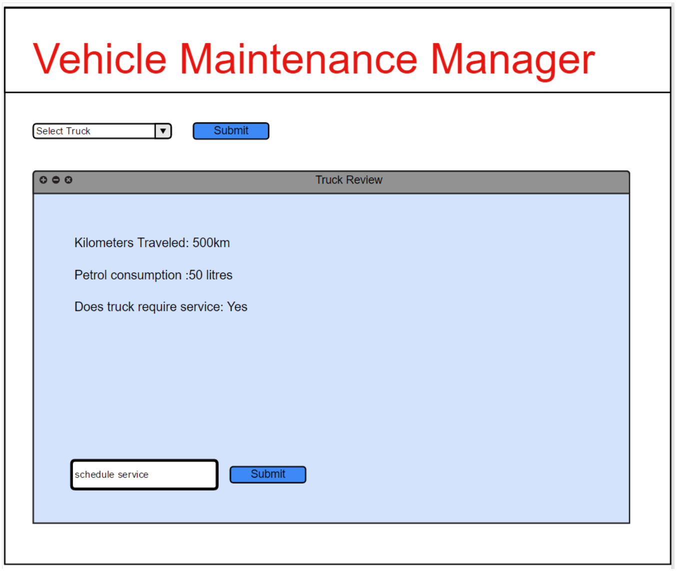

Vehicle Maintenance Manager
Select Truck
Submit
Truck Review
Kilometers Traveled: 500km
Petrol consumption :50 litres
Does truck require service: Yes
schedule service
Submit

Figure 2.11: The maintenance view

sen in order to create a high contrast to the predominately black writing with a simple font, improving readability.

2.5 BACK-END

Figure 2.14 shows the entire back-end system design. The system has been categorised into 3 parts; the postgresql database, the django server and the data processing scripts.

Table 2.1: Responsibilities and activities of relevant users

Webpage	**Responsibilities**	**Input**	**Output**
Main	Provide a page for all users to select their role	Type of user	Correct user login page
Login	Provide a page to verify user credentials	User email,	Correct user dashboard
		User password	
Sender	Provide a page for a sender request for a package to be picked up from their location and delivered	Package dimensions	Delivery cost
		Receiver details	Create a delivery job card for CourierJZ
Receiver	Provide a page for a receiver to view the delivery details of the package	Tracking number	Package location and status
Driver	Provide a page for the driver to view a map and details of all deliveries and pick ups	Checklist for pickup or delivery	Update server on package status
Dispatcher	Provide a page to view all packages and their locations	Initiation of allocation algorithm	Update server on where package status
CEO	Provide a page to view operational activities and data analysis	Reports to be generated	Generate reports on server and return to webpage
Operational Officer	Provide a page to view details on all clients and drivers	Driver or client details	Information for respective driver or client
Vehicle maintenance officer	Provide a page to view information on all vehicles	Vehicle id for maintenance	Add vehicle to schedule for maintenance
System admin	Provide a page to view system status and maintenance	Pause or start the system	Initiate the action to pause or start
		Request for system backup	Pause the system, do backup, then start the system again
Accounts officer	Provide a page to view all financial information	Request financial information	Display financial information

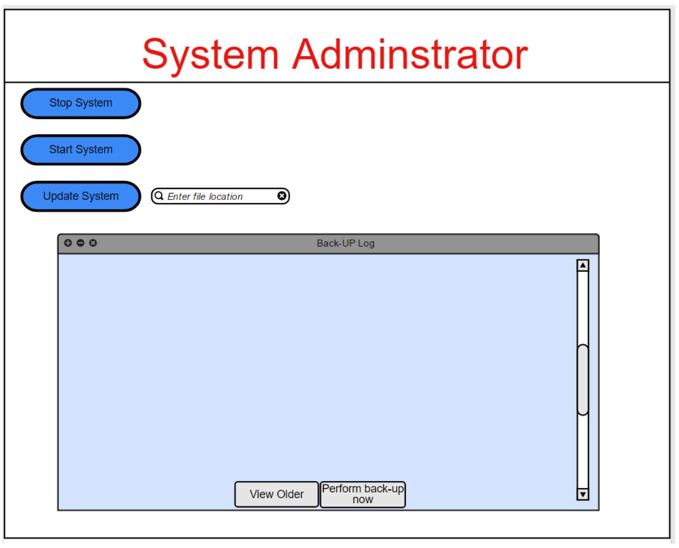


Figure 2.12: The system administrator view

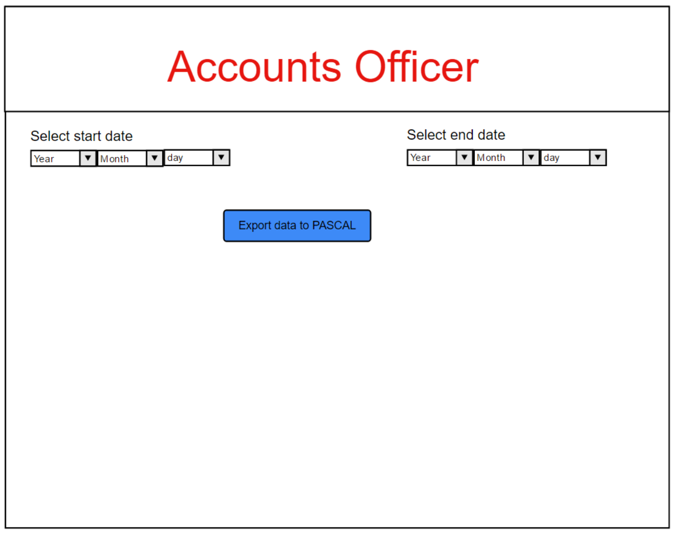


Figure 2.13: The accounts officer view

2.5.1 The PostgreSQL Database - Designing the Database

Due to the nature of vehicle management systems data, a relational database has been selected. The database will be used to store all the information relevant to the operation of CourierJZ. This information includes client information, staff information, package information, vehicle information and financial information. The data should be organised in a manner that is easy to access.

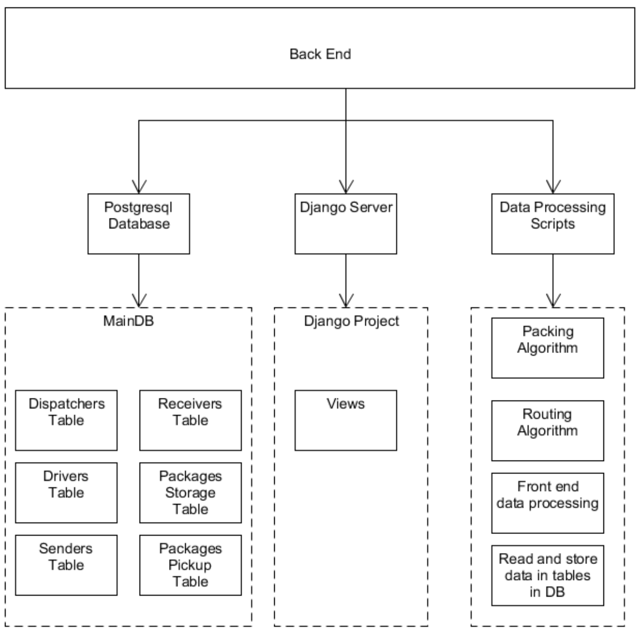


Figure 2.14: The backend system diagram

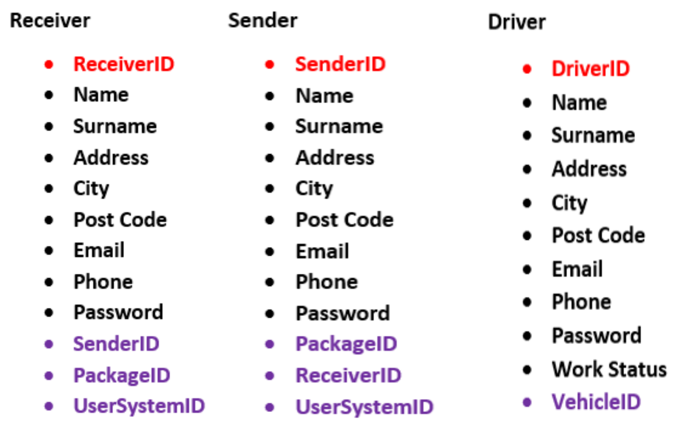
No data has been provided therefore dummy data has to be generated. The data is generated from the site www.generatedata.com which allows you to generate test data 100 rows at a time. Receiver, sender, dispatcher and driver dummy data has been generated.

The data will be stored in tables. This requires creating primary keys for each of the rows generated. This will ensure that each row in a table will be unique. Figures 2.15, 2.16 and 2.17 show the tables designed. The tables are a receiver, sender, driver, serviceclientlist, dispatchers and packages tables. The variables in red are the primary keys. The purple variables are foreign keys. The foreign keys link the various tables.

2.5.2 Views

Views are implemented in the controller in views.py. They connect the database and the front end. Functions are implemented in the views that are tied to a url. A view function, is a Python function that takes a Web request and returns a Web response. When a user enters a url that function is called. Each function specifies an html that will be opened when the url is entered.

The functions implemented are the sender, dispatcher, receiver, driver, main, login_sender, login_receiver, login_driver, login_ceo and login_dispatcher. The first 4 views receive posts from the front end. User authentication occurs and the response is sent back to the front end.


Receiver

- ReceiverID
- Name
- Surname
- Address
- City
- Post Code
- Email
- Phone
- Password
- SenderID
- PackageID
- UserSystemID

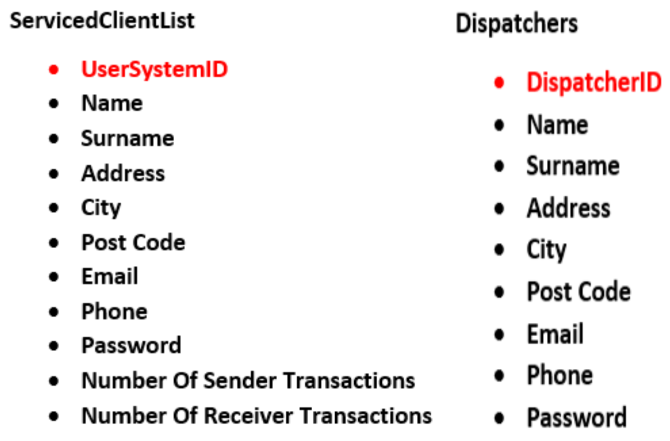
Sender

- SenderID
- Name
- Surname
- Address
- City
- Post Code
- Email
- Phone
- Password
- PackageID
- ReceiverID
- UserSystemID

Driver

- DriverID
- Name
- Surname
- Address
- City
- Post Code
- Email
- Phone
- Password
- Work Status
- VehicleID

Figure 2.15: The receiver, sender and driver tables

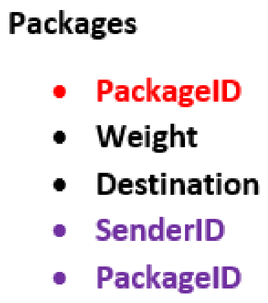

ServicedClientList

- UserSystemID
- Name
- Surname
- Address
- City
- Post Code
- Email
- Phone
- Password
- Number Of Sender Transactions
- Number Of Receiver Transactions

Dispatchers

- DispatcherID
- Name
- Surname
- Address
- City
- Post Code
- Email
- Phone
- Password

Figure 2.16: The servicedclientlist and dispatchers tables


Packages

- PackageID
- Weight
- Destination
- SenderID
- PackageID

Figure 2.17: The packages table

Algorithms
Two algorithms should be implemented in order to service the needs of the dispatcher, sender and the driver. The route and vehicle that a driver chooses is determined by application of a vehicle routing algorithm. To determine which vehicle parcels should go into a packing algorithm should be implemented.

Routing Algorithm
Determining the route that a driver has to take can be labelled as a transportation model problem [9]. As noted in [6] these problems can fall into one of two categories, static and dynamic. In the static scenario all the data is known before the routing takes place and does not change once it has. In the dynamic case, all or some of the data is only received as the route calculation is taking place. The information can still change even after the routing process has been applied [10] .

Static routing problems include the time-dependent travelling salesman problem (TSP) and the probabilistic TSP. In the time-dependent TSP, travelling times between nodes are not constant but change through time in a known manner. For the probabilistic TSP, travel times are deterministic but demand at each node occurs with a known probability, p [9].

At present it should be noted that determining a route a driver should take, a slew of dynamic algorithms may be used. These include dynamic routing, dynamic shortest paths, dynamic traffic assignment, dynamic fleet management and dynamic facility location [9]. Dynamic routing can be applied to multiple contexts including courier services and combined pickup and delivery services as demonstrated by Psaraftis in [9].

Packing Algorithm

There are a number of ways which one can implement a packing algorithm; these algorithms are commonly called bin packing algorithms. Some of the most common algorithms are; First fit algorithm, next fit, best fit algorithms and there are different variations of these algorithms based on the application. A general simple case of a first fit bin packing algorithm is when there is a limited number of bins of predetermined size and a queue of items that are going to be fitted in the bin. You allocate an item based on where it will fit first while traversing the bins, however this could result in a waste of space (since you could assign one big item to a space that would fit two small item and that could have increased the efficiency if the aim was to package as many packages as possible in the given bins) [11]. With the next fit algorithm, the first fit algorithm is used to allocate the first item, thereafter instead of starting from the first bin to allocate the rest of the item, you move on to the next bin that has space if the current bin cannot accommodate the item. The best fit bin packing algorithm traverses all the bins and puts the items in the bin that has the space that best suit the items dimensions [12]. The best fit algorithm takes a longer while than the other algorithm because it has to traverse the bin searching for a best fit and comparing before allocating any item to a bin [13]. All these packing algorithms can be implemented in one, two or three dimensions, and weight can be added as the fourth dimension if the packages are matter. Increasing the dimensions increases the complexity and run time of the algorithm.

The design approach followed for the design of the packing algorithm is a combination of the 3 basic algorithms that were mentioned in the literature review and an addition of a few basic functionalities. The assumptions made when designing the algorithm was there will be no package that will exceed the dimensions of the truck, and all trucks are of equal dimensions. The algorithm is made up of four lists; a list of all the trucks and a queue of packages. Each package is an array in which, the first, second, third, and fourth elements are the packageâĂŹs identifier, its height, width and length respectively. The first package will chosen based on the furthest delivering distance from the store house. Thereafter the following packages will be sorted relative to the delivery distance of the first package while obeying the rules of the best fit algorithm. This particular design was chosen because even if we might not be able to pack the truck with the most goods, it solves the travelling sales man problem and it brings about efficiency in the system. The best fit algorithm is a very expensive algorithm in terms of runtime. And combining it with methods to calculate the distance between the last element and the next element to append is going to be cost even more runtime. Thus the trade-off of this design is that for big data sets it might be slow.

2.6 USE CASES

- Send a package
  Primary Actor: Sender
  Scope: Package allocation system
  Stakeholders and Interests:
  Sender - wants to send package as fast as possible and as cheap as possible
  CourierJZ - wants package information, delivery details and pricing information
  How do you perform the action:

  - Create an account if you do not have one
  - Visit home page and login
  - Enter package dimensions and relevant information
  - View pricing to see affordability
  - Enter destination details
  - Submit information and make payment

- Receive a package

  Primary Actor: Receiver
  Scope: Delivery time estimation system and Package tracking system
  Stakeholders and Interests:

Receiver - wants date and time of delivery of the package, and wants to know where the package is.
CourierJZ - wants the contact details of the receiver
How do you perform the action:

  - Upon reception of the payment from the sender, the receiver receives an email notifying them of the delivery
  - The email posses a link to the website and a package identity number that the receiver can use to view the estimated time of arrival of their parcel and where their parcel is.

- Get work schedule and route

  Primary Actor: Driver
  Scope: Route allocation system
  Stakeholders and Interests:
  Driver - wants to be allocated a vehicle and a route to commence with delivering
  CourierJZ - wants to effectively allocate vehicles to drivers with efficient route to follow
  How do you perform the action:

  - Driver signs into the system
  - Details of the allocated vehicle and route are displayed

- Request day off

  Primary Actor: Driver
  Scope: Human resources management system, route and package allocation systems
  Stakeholders and Interests:
  Driver - wants to be given a day off
  Dispatcher - cannot allocate routes and a vehicle to a driver who has requested a day off
  How do you perform the action:

  - Log into page the driver page
  - Click the request day off button available on page
  - Choose the day you want to take off from the resulting calendar view
  - Submit selection and await confirmation

- Run a scheduling module

Primary Actor: Dispatcher
Scope: Route allocation system and package allocation system
Stakeholders and Interests:
Dispatcher - wants to allocate routes to drivers and decide which trucks packages can go into
Sender - wants their parcel picked up and eventually delivered as soon as possible
Driver - wants to choose a route and vehicle from allocated routes and vehicles
How do you perform the action:

- The driver logs into the their page
- The driver clicks the request day off button available on their page
- The driver will choose the day they want off from the resulting calendar view
- The driver will then submit their selection and await confirmation

• Schedule a vehicle for service

Primary Actor: Vehicle maintenance manager
Scope: Maintenance scheduling
Stakeholders and Interests:
Clients: They want to know that the service is reliable and trustworthy
Driver: they need vehicles to be safe and available
CourierJz: They want to ensure that vehicles will not breakdown
Government: they want to ensure that all vehicles are roadworthy
How do you perform the action:

- View vehicles details on the vehicle maintenance manager page
- Determine if the vehicle needs maintenance by following the manufactures guidelines
- If the vehicle is due for maintenance then press the submit

• Back up and maintain system

Primary Actor: System Admin
Scope: System Maintenance
Stakeholders and Interests:
Clients: They want a system that is smooth and easy to use
CourierJz: They want a system that if free of problems
How do you perform the action:

- The system administrator must log on to their page
- They can then view the backup log

    - Should a backup be required, they can then opt to perform a backup

- Accounts auditing

  Primary Actor: Account Officer
  Scope: Accounts system
  Stakeholders and Interests:
  Account holders- He/She wants to make sure that the he/she has been charged the correct amount.
  Account officer-A list of all the accounts which have made payments in a period of time that is being audited.
  How do you perform the action:

    - Account officer logs in to the accounts page.
    - Select the time period he/she wants to audit
    - Press button which says export to accounting software.

- Carry out operational checks and data analytics

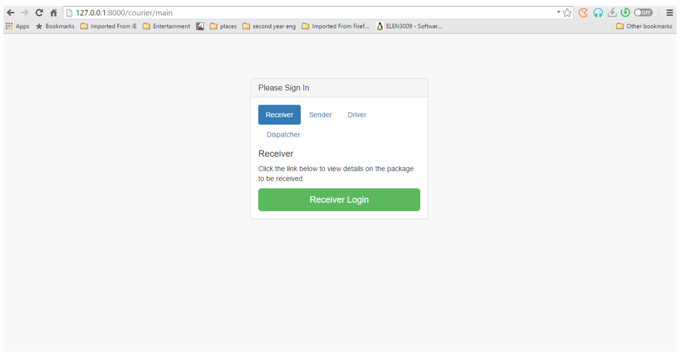
  Primary Actor: CEO
  Scope: JZ Courier
  Stakeholders and Interests:
  CEO- wants to get an overview of how the company performing
  How do you perform the action:

    - CEO logins into the CEO page
    - CEO clicks button to determine what report/statistics he/she would like to view
    - Report appears in a window on the page

3 Relevant Modules Description / Results

3.1 Front-end

The implemented webpages are shown below, these webpages show how the design of the pages have changed from their initial stages to their implementation stages. The company logo and any unnecessary of user implementations have been left out of these pages as the focus was to create a user friendly page first, with changes added later to further identify the company and cement the company image in the design image. The webpages are designed from a bootstrap templates which consist of HTML, CSS and JavaScript code. This code was then modified to get the webpages below.
The above page shows the main page that every user must access before they can access their dashboard. This page is important in the prototype stage as it offers a centralized location



Click the link below to view details on the package to be received

Figure 3.1: The main page - the first page a user goes to

where all users can interact and access their user specific page. In this page the user is able to click on their designation (i.e. receiver or driver) and see a short description of their role in the CourierJZ ecosystem. Once they have picked their designation, they can then access the specific login page for their user dashboard. This page also assists in troubleshooting as when the system administrator logs out of one user dashboard they are brought back to this page and can then proceed to the next user dashboard. The login section will be described below.

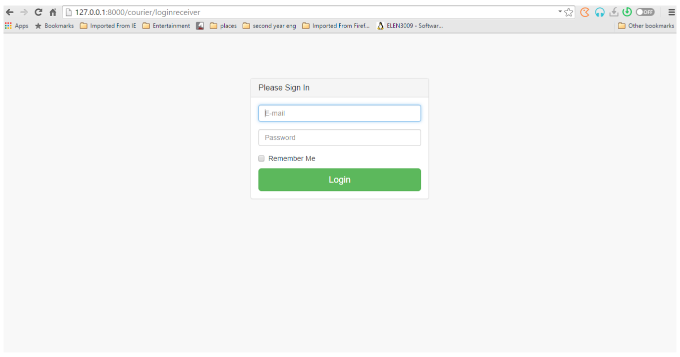


Figure 3.2: The login page of the receiver

The user is brought to a specific login page based on the designation they picked on the main page. This page facilitates the login to their user specific dashboard. In the current prototype there is only one login and password for each designation. When they enter their details and submit they are taken to their user dashboard. On this dashboard a welcome notification will pop up if the username and password was correct and subsequently display all the dashboard information, should the login details be incorrect then a pop up say such will be displayed and the user will be taken back to the login page
The receiver dashboard is the simplest of the four implemented dashboards and can be seen as such. This dashboard only has one form with one textbox for submission. Here the user is able to enter their tracking id and submit. On submission a dialogue box will appear that

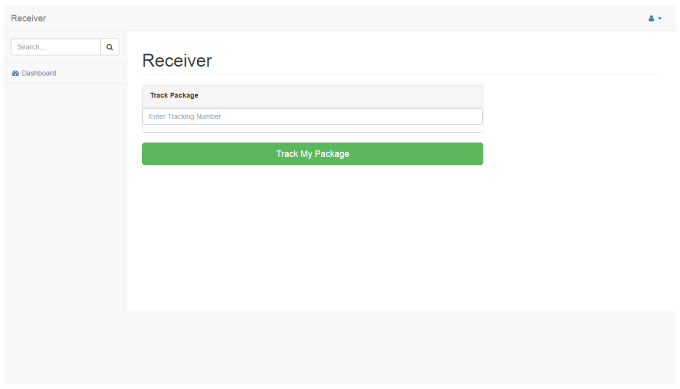


Figure 3.3: The receiver page

shows the user the location and status of the package

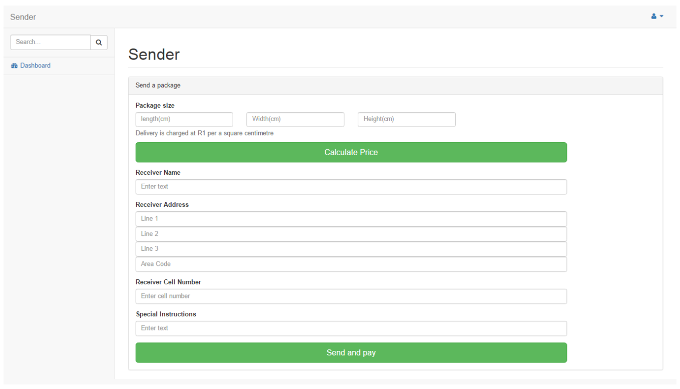


Figure 3.4: The sender page

The sender page is basically the heart of CourierJZ as without this page, the business would not exist. This page consists of one form and two buttons. The first button allows the user to get the price of sending the package by inputting its dimensions. The user can then decide if sending the package is still feasible before further entering the receiver's details. If the user chooses to proceed and enter the receiver's details they can then submit which will send a jobcard to courierJZ's servers to process. This process is further streamlined since the user had entered their details when they created their profile page(This was not implemented in the prototype stage). This means that the users address will already be saved in the database and that payment will be automatic.
The driver page consists of two sections. The first section contains a map and the second section contains information of the locations to be visited. The map shows the current location of the driver and the destination of the current package to be delivered. This helps the driver find the shortest possible route to the destination as it uses the google maps API which combines the route data with crowdsourced traffic data. The second section provides the driver with the locations and details of each client. This includes the contact information

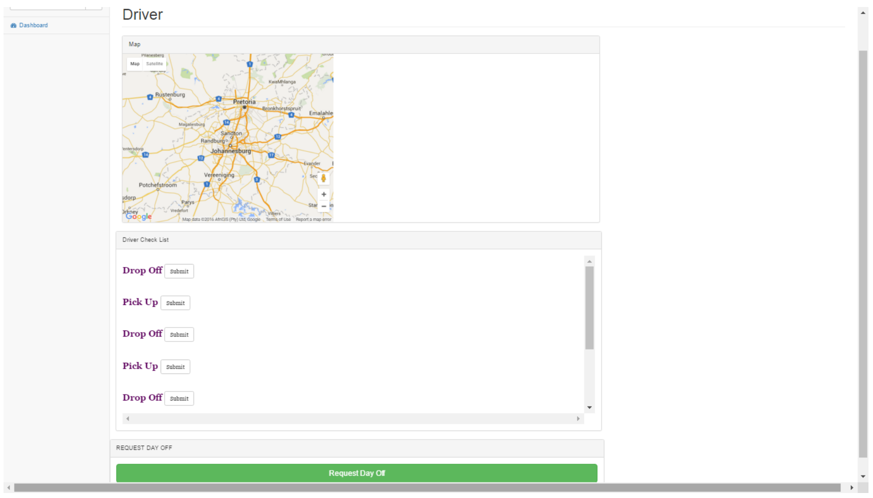
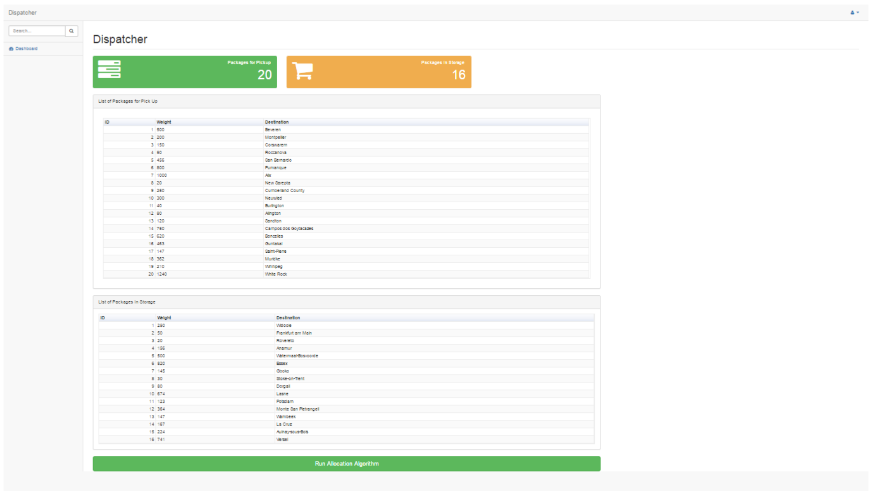


Figure 3.5: The driver page

and whether the client has a package to be picked up or needs one to be delivered. Once to location has been visited and the appropriate action is taken, the driver may then update the server by pressing the update button.



Dispatcher

List of Packages for Pick Up

ID	Weight	Destination
1	500	Beveren
2	200	Montpellier
3	150	Corswarem
4	50	Roccanova
5	456	San Bernardo
6	800	Pumanque
7	1000	Aix
8	20	New Sarepta
9	250	Cumberland County
10	300	Neuwied
11	40	Burlington
12	80	Allington
13	120	Sandton
14	750	Campos dos Goytacazes
15	620	Boncelles
16	463	Guntakal
17	147	Saint-Pierre
18	362	Muridke
19	210	Winnipeg
20	1240	White Rock

List of Packages in Storage

ID	Weight	Destination
1	250	Widooie
2	50	Frankfurt am Main
3	20	Rovereto
4	156	Anamur
5	500	Watermaal-Bosvoorde
6	820	Essex
7	145	Gooko
8	30	Stoke-on-Trent
9	80	Dorgali
10	674	Lasne
11	123	Potsdam
12	364	Monte San Pietrangeli
13	147	Wambeek
14	167	La Cruz
15	224	Aulnay-sous-Bois
16	741	Versel

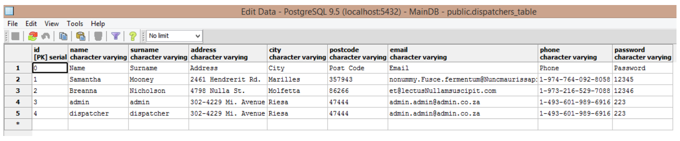
Figure 3.6: The dispatcher page

The implemented dispatcher dashboard is shown above. This consists of two sections where the dispatcher has access to all the packages in storage and all the packages requested for pick up. The dispatcher may then press the run allocation algorithm, this algorithm determines what each driver will be doing for the day and what packages they are responsible for.

3.2 Back end

3.2.1 Database Tables

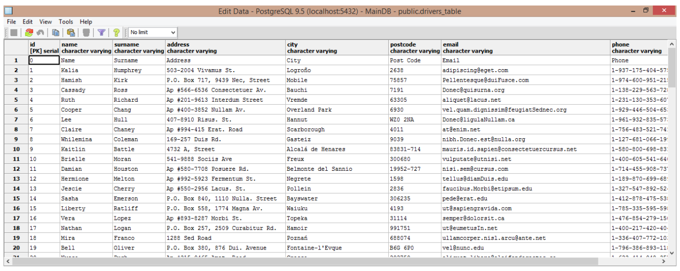
The foreign keys have not been implemented in any of the tables given below. This is because of two reasons, firstly, when the dummy data was being created each of the rows in each table was given primary keys that start from 1 to the number of rows. This means that the primary keys for all tables are in the same range. This means that some foreign keys would have been identical to the primary key. Creating foreign keys would have required linking all the relevant tables in the dummy data. This will have been a time consuming process hence it is not carried out.
Figure 3.7 shows the dispatcher table implemented and stored in MainDB, because of the foreign keys that have not been included.


Edit Data - PostgreSQL 9.5 (localhost:5432) - MainDB - public.dispatchers_table
File Edit View Tools Help
No limit

	id [PK] serial	name character varying	surname character varying	address character varying	city character varying	postcode character varying	email character varying	phone character varying	password character varying
1	0	Name	Surname	Address	City	Post Code	Email	Phone	Password
2	1	Samantha	Mooney	2461 Hendrerit Rd.	Marilles	357943	nonummy.Fusce.fermentum@Nuncmaurissap:	1-974-764-092-8058	12345
3	2	Breanna	Nicholson	4798 Nulla St.	Molfetta	86266	et@lectusNullamsuscipit.com	1-973-216-529-7088	12346
4	3	admin	admin	302-4229 Mi. Avenue	Riesa	47444	admin.admin@admin.co.za	1-493-601-989-6916	223
5	4	dispatcher	dispatcher	302-4229 Mi. Avenue	Riesa	47444	admin.admin@admin.co.za	1-493-601-989-6916	223
*									

Figure 3.7: The dispatcher table

Figure 3.8 shows the drivers table implemented and stored in MainDB.


Edit Data - PostgreSQL 9.5 (localhost:5432) - MainDB - public.drivers_table
File Edit View Tools Help
No limit

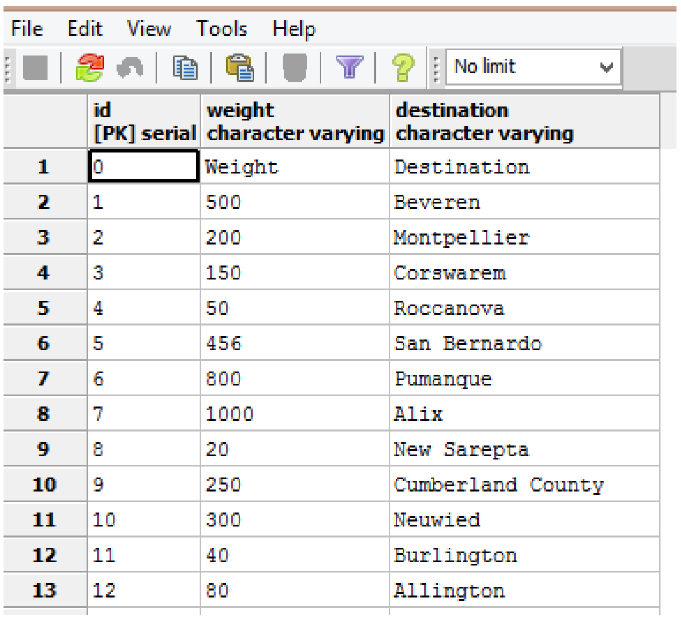
	id [PK] serial	name character varying	surname character varying	address character varying	city character varying	postcode character varying	email character varying	phone character varying
1	0	Name	Surname	Address	City	Post Code	Email	Phone
2	1	Kalia	Humphrey	503-2004 Vivamus St.	Logroño	2638	adipiscing@eget.com	1-937-175-404-57
3	2	Hamish	Kirk	P.O. Box 717, 9439 Nec, Street	Mobile	75857	Pellentesque@duiFusce.com	1-974-600-951-21
4	3	Cassady	Ross	Ap #566-6536 Consectetuer Av.	Bauchi	7191	Donec@quisurna.org	1-138-229-563-72
5	4	Ruth	Richard	Ap #201-9613 Interdum Street	Vremde	63305	aliquet@lacus.net	1-231-130-353-60
6	5	Cooper	Chang	Ap #400-3852 Nullam Av.	Overland Park	6930	vel.quam.dignissim@feugiatSednec.org	1-929-446-504-65
7	6	Lee	Hull	407-8910 Risus. St.	Hannut	WZ0 2NA	Donec@ligulaNullam.ca	1-961-932-835-57
8	7	Claire	Chaney	Ap #994-415 Erat. Road	Scarborough	4011	at@enim.net	1-756-483-521-74
9	8	Whilemina	Coleman	169-257 Duis Rd.	Gasteiz	9039	nibh.Donec.est@nulla.org	1-127-681-066-19
10	9	Kaitlin	Battle	4732 A, Street	Alcalá de Henares	83831-714	mauris.id.sapien@consectetuercursus.net	1-580-800-698-83
11	10	Brielle	Moran	541-9888 Sociis Ave	Freux	300680	vulputate@utnisi.net	1-400-605-541-64
12	11	Damian	Houston	Ap #580-7708 Posuere Rd.	Belmonte del Sannio	19952-727	nisi.sem@cursus.com	1-714-455-908-73
13	12	Hermione	Melton	Ap #992-5923 Fermentum St.	Negrete	1598	tellus@diamDuis.edu	1-189-870-699-68
14	13	Jescie	Cherry	Ap #550-2956 Lacus. St.	Pollein	2836	faucibus.Morbi@etipsum.edu	1-327-547-892-52
15	14	Sasha	Emerson	P.O. Box 840, 1110 Nulla. Street	Bayswater	306235	pede@erat.edu	1-412-878-475-53
16	15	Liberty	Ratliff	P.O. Box 558, 1774 Magna Av.	Waiuku	4193	ut@sapiengravida.com	1-785-335-595-59
17	16	Vera	Lopez	Ap #893-8287 Morbi St.	Topeka	31114	semper@dolorsit.ca	1-476-854-279-15
18	17	Nathan	Logan	P.O. Box 257, 2509 Curabitur Rd.	Hamoir	991751	ut@eumetusIn.net	1-400-217-420-40
19	18	Mira	Franco	1288 Sed Road	Poznań	688074	ullamcorper.nisl.arcu@ante.net	1-336-407-772-10
20	19	Bell	Oliver	P.O. Box 380, 876 Dui. Avenue	Fontaine-l'Evque	B6G 6P0	vel@nunc.edu	1-796-386-893-11

Figure 3.8: The drivers table

Figure 3.9 shows the packages in the pickup table in MainDB.
Figure 3.10 shows the packages in the storage table in MainDB.
Figure 3.11 shows the receivers table created in MainDB.
Figure 3.12 shows the senders table created in MainDB.

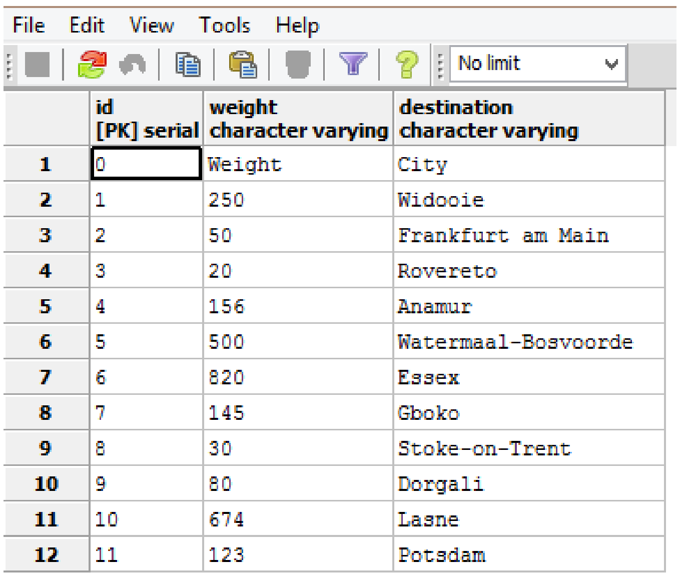
Packing algorithm

The implemented algorithm has the dimensions of the truck, and the extracts data from the package database and stores it in an array. It then computes the total dimensions of all the packages in the package array to check if they fit in one truck. If they fit in one truck the algorithm displays the message to the dispatcher and exits. Otherwise the algorithm, starts appending the items one by one onto the truck array. After appending each item, the algorithm checks whether the truck is full or not. If the truck is full a new truck is added and the



	id [PK] serial	weight character varying	destination character varying
1	0	Weight	Destination
2	1	500	Beveren
3	2	200	Montpellier
4	3	150	Corswarem
5	4	50	Roccanova
6	5	456	San Bernardo
7	6	800	Pumanque
8	7	1000	Alix
9	8	20	New Sarepta
10	9	250	Cumberland County
11	10	300	Neuwied
12	11	40	Burlington
13	12	80	Allington

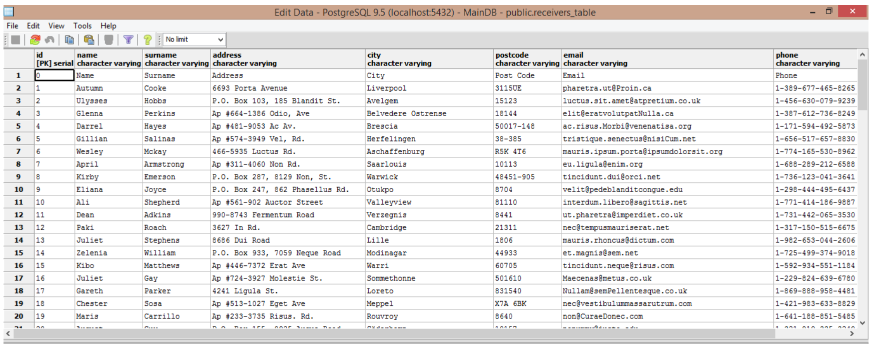
Figure 3.9: The packages in the pickup table



	id [PK] serial	weight character varying	destination character varying
1	0	Weight	City
2	1	250	Widooie
3	2	50	Frankfurt am Main
4	3	20	Rovereto
5	4	156	Anamur
6	5	500	Watermaal-Bosvoorde
7	6	820	Essex
8	7	145	Gboko
9	8	30	Stoke-on-Trent
10	9	80	Dorgali
11	10	674	Lasne
12	11	123	Potsdam

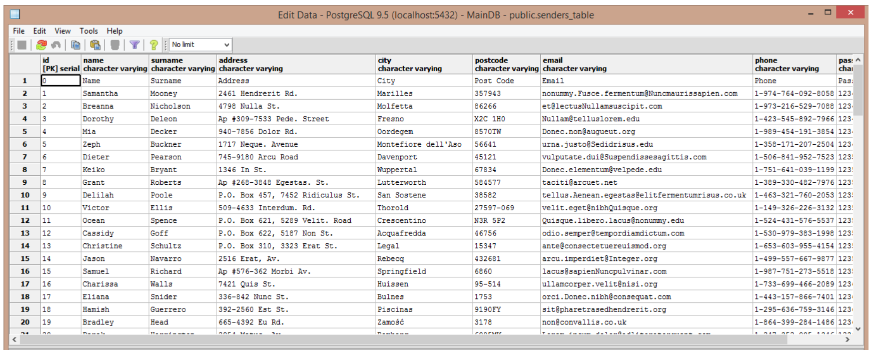
Figure 3.10: The packages in storage table

item is appended onto the new truck. And the algorithm is stuck in a lop that appends the items into the truck as long as there are trucks to append to or items to add. This was the first version of the implementation and it was going to be improved to get to level that was designed initial. Due to time constrains further implementation could not be done. Figure 3.13 shows the UML activity diagram of the implemented algorithm.


Edit Data - PostgreSQL 9.5 (localhost:5432) - MainDB - public.receivers_table
File Edit View Tools Help
No limit

	id [PK] serial	name character varying	surname character varying	address character varying	city character varying	postcode character varying	email character varying	phone character varying
1	0	Name	Surname	Address	City	Post Code	Email	Phone
2	1	Autumn	Cooke	6693 Porta Avenue	Liverpool	3115UE	pharetra.ut@Proin.ca	1-389-677-465-8265
3	2	Ulysses	Hobbs	P.O. Box 103, 185 Blandit St.	Avelgem	15123	luctus.sit.amet@atpretium.co.uk	1-456-630-079-9239
4	3	Glenna	Perkins	Ap #664-1386 Odio, Ave	Belvedere Ostrense	18144	elit@eratvolutpatNulla.ca	1-387-612-736-8249
5	4	Darrel	Hayes	Ap #481-9053 Ac Av.	Brescia	50017-148	ac.risus.Morbi@venenatisa.org	1-171-594-492-5873
6	5	Gillian	Salinas	Ap #574-3949 Vel, Rd.	Herfelingen	38-385	tristique.senectus@nisiCum.net	1-656-517-657-8830
7	6	Wesley	Mckay	466-5935 Luctus Rd.	Aschaffenburg	R5K 4T6	mauris.ipsum.porta@ipsumdolorsit.org	1-774-165-530-8962
8	7	April	Armstrong	Ap #311-4060 Non Rd.	Saarlouis	10113	eu.ligula@enim.org	1-688-289-212-6588
9	8	Kirby	Emerson	P.O. Box 287, 8129 Non, St.	Warwick	48451-905	tincidunt.dui@orci.net	1-736-123-041-3641
10	9	Eliana	Joyce	P.O. Box 247, 862 Phasellus Rd.	Otukpo	8704	velit@pedeblanditcongue.edu	1-298-444-495-6437
11	10	Ali	Shepherd	Ap #561-902 Auctor Street	Valleyview	81110	interdum.libero@sagittis.net	1-771-414-186-9887
12	11	Dean	Adkins	990-8743 Fermentum Road	Verzegnis	8441	ut.pharetra@imperdiet.co.uk	1-731-442-065-3530
13	12	Paki	Roach	3627 In Rd.	Cambridge	21311	nec@tempusmauriserat.net	1-317-150-515-6675
14	13	Juliet	Stephens	8686 Dui Road	Lille	1806	mauris.rhoncus@dictum.com	1-982-653-044-2606
15	14	Zelenia	William	P.O. Box 933, 7059 Neque Road	Modinagar	44933	et.magnis@sem.net	1-725-499-374-9018
16	15	Kibo	Matthews	Ap #446-7372 Erat Ave	Warri	60705	tincidunt.neque@risus.com	1-592-934-551-1184
17	16	Juliet	Gay	Ap #724-3927 Molestie St.	Sommethonne	501610	Maecenas@metus.co.uk	1-229-824-639-6780
18	17	Gareth	Parker	4241 Ligula St.	Loreto	831540	Nullam@semPellentesque.co.uk	1-869-888-958-4481
19	18	Chester	Sosa	Ap #513-1027 Eget Ave	Meppel	X7A 6BK	nec@vestibulummassarutrum.com	1-421-983-633-8829
20	19	Maris	Carrillo	Ap #233-3735 Risus. Rd.	Rouvroy	8640	non@CuraeDonec.com	1-641-188-851-5485

Figure 3.11: The receivers table


Edit Data - PostgreSQL 9.5 (localhost:5432) - MainDB - public.senders_table
File Edit View Tools Help
No limit

	id [PK] serial	name character varying	surname character varying	address character varying	city character varying	postcode character varying	email character varying	phone character varying	pas char
1	0	Name	Surname	Address	City	Post Code	Email	Phone	Pas
2	1	Samantha	Mooney	2461 Hendrerit Rd.	Marilles	357943	nonummy.Fusce.fermentum@Nuncmaurissapien.com	1-974-764-092-8058	123
3	2	Breanna	Nicholson	4798 Nulla St.	Molfetta	86266	et@lectusNullamsuscipit.com	1-973-216-529-7088	123
4	3	Dorothy	Deleon	Ap #309-7533 Pede. Street	Fresno	X2C 1H0	Nullam@telluslorem.edu	1-423-545-892-7966	123
5	4	Mia	Decker	940-7856 Dolor Rd.	Oordegem	8570TW	Donec.non@augueut.org	1-989-454-191-3854	123
6	5	Zeph	Buckner	1717 Neque. Avenue	Montefiore dell'Aso	56641	urna.justo@Sedidrisus.edu	1-358-171-207-2504	123
7	6	Dieter	Pearson	745-9180 Arcu Road	Davenport	45121	vulputate.dui@Suspendissesagittis.com	1-506-841-952-7523	123
8	7	Keiko	Bryant	1346 In St.	Wuppertal	67834	Donec.elementum@velpede.edu	1-751-641-039-1199	123
9	8	Grant	Roberts	Ap #268-3848 Egestas. St.	Lutterworth	584577	taciti@arcuet.net	1-389-330-482-7976	123
10	9	Delilah	Poole	P.O. Box 457, 7452 Ridiculus St.	San Sostene	38582	tellus.Aenean.egestas@elitfermentumrisus.co.uk	1-463-321-760-2053	123
11	10	Victor	Ellis	509-4633 Interdum. Rd.	Thorold	27597-069	velit.eget@nibhQuisque.org	1-149-326-226-3132	123
12	11	Ocean	Spence	P.O. Box 621, 5289 Velit. Road	Crescentino	N3R 5P2	Quisque.libero.lacus@nonummy.edu	1-524-431-576-5537	123
13	12	Cassidy	Goff	P.O. Box 622, 5187 Non St.	Acquafredda	46756	odio.semper@tempordiamdictum.com	1-530-979-383-1998	123
14	13	Christine	Schultz	P.O. Box 310, 3323 Erat St.	Legal	15347	ante@consectetuereuismod.org	1-653-603-955-4154	123
15	14	Jason	Navarro	2516 Erat, Av.	Rebecq	432681	arcu.imperdiet@Integer.org	1-499-557-667-9877	123
16	15	Samuel	Richard	Ap #576-362 Morbi Av.	Springfield	6860	lacus@sapienNuncpulvinar.com	1-987-751-273-5518	123
17	16	Charissa	Walls	7421 Quis St.	Huissen	95-514	ullamcorper.velit@nisi.org	1-733-699-466-2089	123
18	17	Eliana	Snider	336-842 Nunc St.	Bulnes	1753	orci.Donec.nibh@consequat.com	1-443-157-866-7401	123
19	18	Hamish	Guerrero	392-2560 Est St.	Piscinas	9190FY	sit@pharetrasedhendrerit.org	1-295-636-759-3146	123
20	19	Bradley	Head	665-4392 Eu Rd.	Zamość	3178	non@convallis.co.uk	1-864-399-284-1486	123

Figure 3.12: The senders table

Although the packing algorithm has not been linked to the front end. It has been implemented and extensively documented. One tool used for documenting it is doxygen. Figures **??**, **??** and **??** show the resulting in depth module description.

4 Design Analysis

The webpages for the system were successfully created. Databases to create dummy data to test the system was created, and a packing algorithm was written.In the early stages of the project extensive research was conducted and a design brief was document. Due to the time constraints the separately developed parts could not be concatenated to bring about an overall system. Even though the algorithms could not be joined to the rest of the software, basic functionality was achieved to demonstrate the basic functionality of the software.Extensive documentation of all modules that were planed are there for simplicity of future improvements of this project.

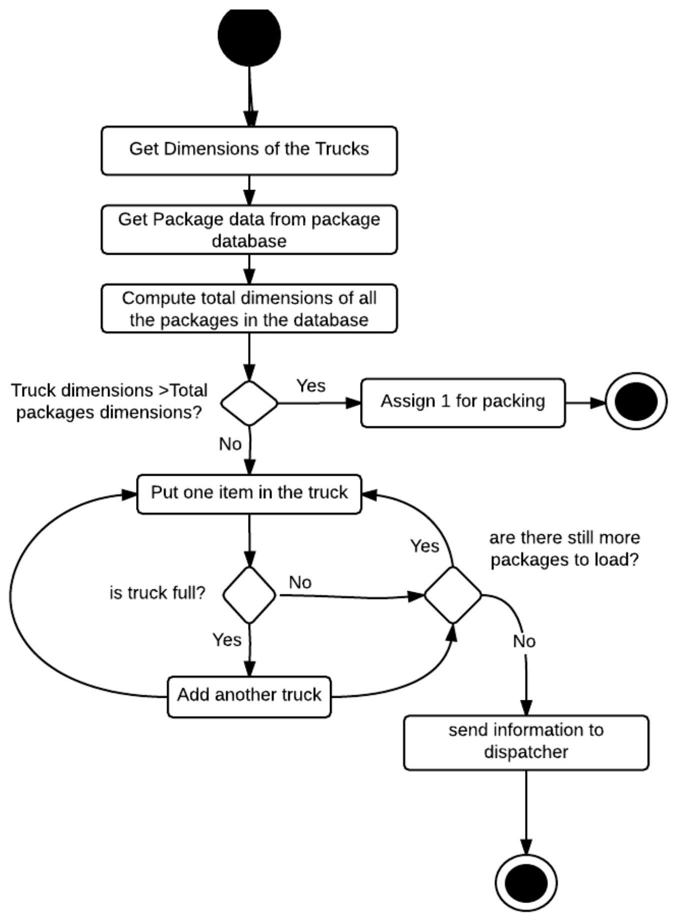

Get Dimensions of the Trucks
Get Package data from package database
Compute total dimensions of all the packages in the database
Truck dimensions >Total packages dimensions?
Yes
Assign 1 for packing
No
Put one item in the truck
Yes
are there still more packages to load?
is truck full?
No
Yes
No
Add another truck
send information to dispatcher

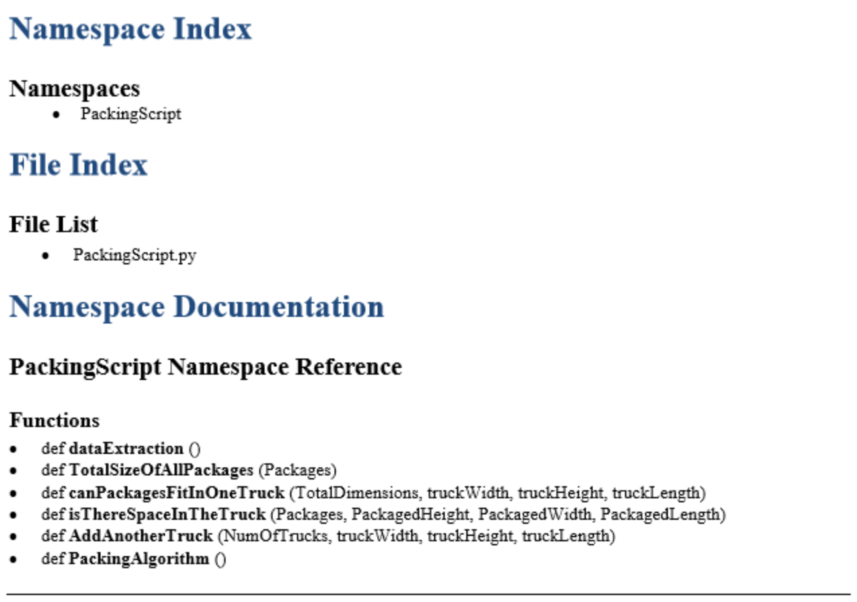
Figure 3.13: The packing algorithm activity diagram

5 SCRUM

The agile software development framework chosen for the project is the SCRUM method. This method is chosen because it is lightweight, simple to understand and easy to use. Scrum will be used to address complex problems, increase productivity and deliver a unique solution. An iterative process will be followed where one team is responsible for the front-end and the other team is responsible for the back-end. The process will be iterative whereby the front end team will develop the web pages and the back-end team will then link these pages to the server. It will also be assumed that product specifications will remain constant for the prototype phase with any changes being added after the development of the prototype.

5.1 THE PROJECT CHARTER

The various components of the project charter are described in the introduction of this document.



Namespace Index

Namespaces

- PackingScript

File Index

File List

- PackingScript.py

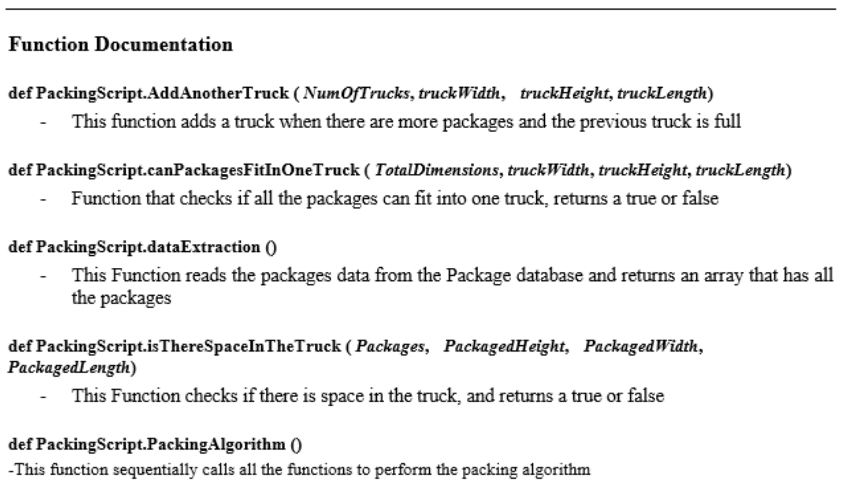
Namespace Documentation

PackingScript Namespace Reference

Functions

- def **dataExtraction** ()
- def **TotalSizeOfAllPackages** (Packages)
- def **canPackagesFitInOneTruck** (TotalDimensions, truckWidth, truckHeight, truckLength)
- def **isThereSpaceInTheTruck** (Packages, PackagedHeight, PackagedWidth, PackagedLength)
- def **AddAnotherTruck** (NumOfTrucks, truckWidth, truckHeight, truckLength)
- def **PackingAlgorithm** ()

Figure 3.14: Doxygen of the packing algorithm namespaces



Function Documentation

def PackingScript.AddAnotherTruck (***NumOfTrucks*****,** ***truckWidth*****,** ***truckHeight*****,** ***truckLength*****)**

- This function adds a truck when there are more packages and the previous truck is full

def PackingScript.canPackagesFitInOneTruck (***TotalDimensions*****,** ***truckWidth*****,** ***truckHeight*****,** ***truckLength*****)**

- Function that checks if all the packages can fit into one truck, returns a true or false

def PackingScript.dataExtraction ()

- This Function reads the packages data from the Package database and returns an array that has all the packages

def PackingScript.isThereSpaceInTheTruck (***Packages*****,** ***PackagedHeight*****,** ***PackagedWidth*****,** ***PackagedLength*****)**

- This Function checks if there is space in the truck, and returns a true or false

def PackingScript.PackingAlgorithm ()

-This function sequentially calls all the functions to perform the packing algorithm

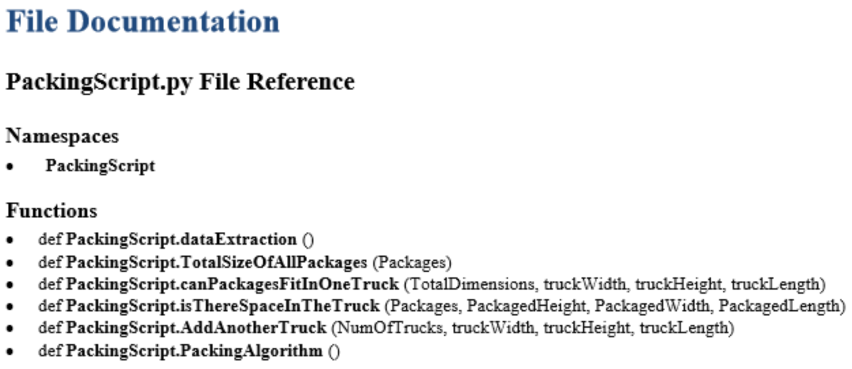
Figure 3.15: Doxygen of the packing algorithm function documentation

5.2 Product Backlog

The product backlog can be seen in Table 5.1.

5.3 Sprint Backlog

The sprint backlog can be seen in Table 5.2, 5.3, 5.4 and 5.5.


File Documentation

PackingScript.py File Reference

Namespaces
- **PackingScript**

Functions
- def **PackingScript.dataExtraction** ()
- def **PackingScript.TotalSizeOfAllPackages** (Packages)
- def **PackingScript.canPackagesFitInOneTruck** (TotalDimensions, truckWidth, truckHeight, truckLength)
- def **PackingScript.isThereSpaceInTheTruck** (Packages, PackagedHeight, PackagedWidth, PackagedLength)
- def **PackingScript.AddAnotherTruck** (NumOfTrucks, truckWidth, truckHeight, truckLength)
- def **PackingScript.PackingAlgorithm** ()

Figure 3.16: Docxygen of the packing algorithm file documentation

5.4 Sprint retrospective

The project consisted of 3 sprints which lasted 1 week each. This consisted of two daily meetings that happened on a Monday and Friday of each sprint.

Sprint 1: 21 March —27 March

This sprint goal was to achieve the first 5 (Element 1-5) tasks on the sprint backlog. These tasks were completed in time with a few adjustments. In element 1, forms for clients to create their accounts was removed and added as a server task instead. The user accounts are manually stored on the database for the prototype stage. The account creation page will now be done once CourierJZ approves the prototype design as this page may lead to developer complications. In element 4, no form for the pickup location was added since it will be stored in the user's profile and thus simplify the act of sending a package.

Sprint 2: 28 March —3 April

This sprint goal was to achieve elements 6—10 on the sprint backlog. Only tasks 7—9 were completed. This was due to the complexity of element 10 and the programmers time being limited by other commitments. For element 6 the payment management system was not implemented due to it not being an integral part of the prototype system. The information about the user's payment details (i.e. Credit card details) will now be stored in the users profile when implemented, this will make the process of sending a package simpler as the card details will not need to be entered each time as in is saved on the database.

Sprint 3: 4 April —10 April

This was the last sprint for the prototype stage. For this cycle elements 10—16 needed to be completed for the last sprint. From element 10, the route assignment algorithm was not implemented. This is because the developers decided to focus on one algorithm (packing algorithm) due to time constraints. During the prototype stage, for simplicity the developers decided to only implement one driver profile, so there was only one route determined, therefore the page for drivers to pick routes was not necessary. For element 12, the check list was replaced by a submit button for each user. This was done so that when items were ticked, they could not mistakenly be unticked in the delivery process. Element 13 and 14 were removed from the list because the shortest distance algorithm was not implemented, so delivery times could not be correctly assigned for each package. Element 15 and 16 were combined into a receiver page with the relevant details being accessible by inputting the tracking ID into the receiver dashboard.

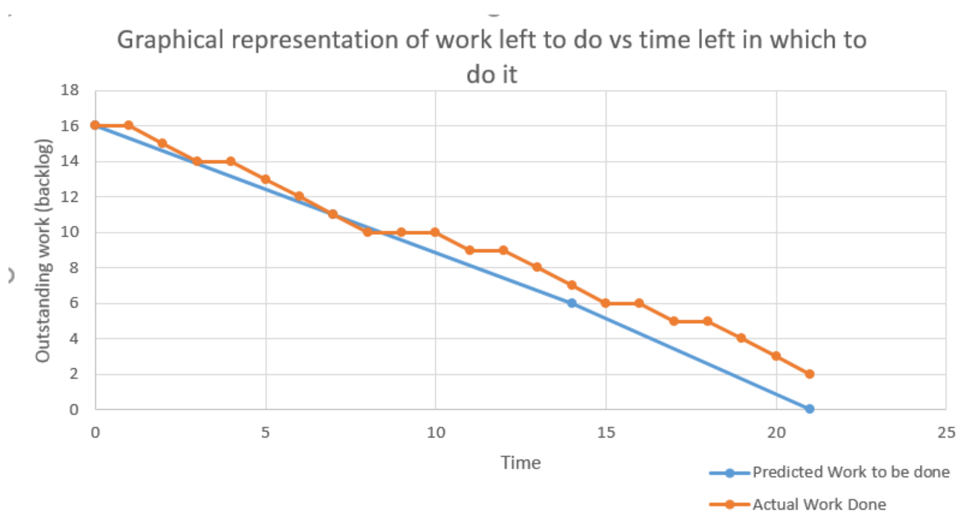


Figure 5.1: Burn down chart of prototype

It can be seen from Figure 5.1 that in the beginning of the developmental process of the prototype,the project was on track. However towards the end, the actual work that was being performed was not meeting the targets. This could be because the work that needed to be performed was underestimated and therefore time was not allocated correctly. Care must be taken to ensure that these task are completed in the nearby future, otherwise the project has a high risk of not being completed in time.

6 Conclusion

A solution to the problem was developed. This solution involved devising algorithms based on adapting the existing solutions to the needs of CourierJZ. A web based application was developed, it interacts with all the stake holders and it was insures that as much information as possible is relayed to all stake holders to minimize and inconvenience and maximize profits and efficiency for CourierJZ. Although the implementation was incomplete the documentation explains clearly the intend path of design

LIST OF FIGURES

LIST OF TABLES

REFERENCES

[1] Unknown. www.dhl.co.za. Last accessed 08 April 2016.

[2] Unknown. www.aramex.co.za. Last accessed 08 April 2016.

[3] Unknown. "Fastway Couriers". www.fastway.co.za. Last accessed 08 April 2016.

[4] Unknown. "Telogis Distributions". http://www.telogis.com/industries/distribution. Last accessed 09 April 2016.

[5] "Vehicle Management Systems (VMS) : Constant Connection, constant improvement". http://www.id-systems.com/about-vms/. Last accessed 09 April 2016.

[6] Ichoua S, Gendreau M, and Potvin J. "Diversion Issues in Real-Time Vehicle Dispatching". Transportation Science, Âľ 2000 INFORMS 0041-1655/00/3404-0426 Vol. 34, No. 4, November 2000.

[7] Abbaspour R. A and Samadzadegan F. "An Evolutionary Solution for Multimodal Shortest Path Problem in Metropolises", UDC 519.8, DOI:10.2298/CSIS090710024A.

[8] Delling D and Wagner D. "Time-Dependent Route Planning. Robust and Online Large-Scale Optimization". Volume 5868 of the series Lecture Notes in Computer Science, pp 207-230.

[9] Psraftis H. N. "Dynamic Vehicle Routing - Status and Prospects". Annals of Operations Research 61(1995)143-164. Pp 143 - 164.

[10] Madsen B. G. O, Allan Larsen A, Solomon M. M. "Dynamic Vehicle Routing Systems âĂŞ Survey and Classification".

[11] Hong C. K, Benkrid X., Iturbe A., Erdogan T, and Arslan T. âĂIJAn FPGA task allocator with preliminary First-Fit 2D packing algorithmsâĂİ ,Adaptive Hardware and Systems (AHS), 2011 NASA/ESA Conference on IEEE Conference Publications.

[12] Zhu Z., Sui J. and Yang L., "Bin-packing Algorithms for Periodic Task Scheduling âĂIJ Information Engineering (ICIE), 2010 WASE International Conference on , 2010, Volume: 2, Pages: 207 - 210, DOI: 10.1109/ICIE.2010.145, IEEE Conference Publications.

[13] Huang L., Liu Z., and Liu Z., âĂIJAn improved lowest-level best-fit algorithm with memory for the 2D rectangular packing problemâĂİ, Information Science, Electronics and Electrical Engineering (ISEEE), 2014 International Conference on ,2014, Volume: 2 ,Pages: 1279 - 1282, DOI: 10.1109/InfoSEEE.2014.6947877, IEEE Conference Publications.

Table 5.1: Product Backlog

ID	Story	Estimate	Priority
1	As a sender I need to be able to create an account	10	1
2	As a sender I need to be able to log in to my account	2	2
3	As a sender I need to input the dimensions of my package	2	3
4	As a sender I want to input my details	3	4
5	As a sender I want to input the receiver's details	2	5
6	As a sender I need to be able to pay for the services	5	6
7	As a fleet assigment manager I need to be able run a route generation algorithm	5	7
8	As a fleet assigment manager I need to see a list of packages in the depot	1	8
9	As a driver I should be able to log in to my account	1	9
10	As a driver I want to pick a vehicle and a route	10	10
11	As a driver I need to see the contact details of my next delivery / pickup	2	11
12	As a driver I want to check off the packages I drop off / pickup	3	12
13	As a sender I need to know when the parcel will be picked up	1	13
14	As a sender I need to be able to know when the package will be delivered	3	14
15	As a receiver I need to receive a notification of the intended delivery	1.5	15
16	As a receiver I need to check the estimated time of delivery	1	16
17	As the CEO I want to see statistics on the clients	4	17
18	As the CEO I want to see statistics on the drivers	4	18
19	As the Vehicle Maintenance Manager I want to see vehicle statistics such as mileage	6	19
20	As the Vehicle Maintenance Manager I want to schedule the maintenance of vehicles	5	20
21	As the Accounts Officer I want to see the list of payments made (for bookeeping purposes)	6	21
22	As the Accounts Officer I need to see a list of vehicle maintenance payments (for bookeeping purposes)	5	22
23	As the CEO I want to generate reports on income	5	23
24	As the CEO I want to generate reports on expenditure	5	24
25	As a driver I want to be able to request a day off	6	25
26	As a driver I should be able to log out of my account	1	26
27	As a sender I need to be able to log out of my account	1	27
28	As the Vehicle Maintenance Manager I want to see vehicle statistics such as fuel consumption	5	28
		105.5	

Table 5.2: Sprint backlog

ID	Story	Sprint backlog tasks	Date
1	As a sender I need to be able to create an account	Create a django project	21/03/2016
		Design database structure	21/03/2016
		Create databases	22/03/2016
		Link page to the django server	22/03/2016
		Save user account details in the database	23/03/2016
		Make a form where clients are able to create their accounts	23/03/2016
2	As a sender I need to be able to log in to my account	Make a log in page	24/03/2016
		Save log in details of senders in database	24/03/2016
3	As a sender I need to input the dimensions of my package	The user is required to fill in the dimensions of their package.	25/03/2016
		Send the package dimensions to the packing algorithm	25/03/2016
4	As a sender I want to input my details	Make a form where the sender can enter their details	26/03/2016
		Store the sender's details in the database	26/03/2016
5	As a sender I want to input the receiver's details	Make a form where the sender can enter the receiver's details	27/03/2016
		Store the receiver's details in the database	27/03/2016
6	As a sender I need to be able to pay for the services	Update the accounts and payments databases	28/03/2016
		Authorise the user's packages for transport	28/03/2016
		Create a payment page for the user	29/03/2016
7	As a fleet assigment manager I need to be able run a route generation algorithm	Have a button that allows the manager to run the route allocation algorithm	29/03/2016
8	As a fleet assigment manager I need to see a list of packages in the depot	Display a list of all the packages that are at the depot that need to be allocated routes	30/03/2016

Table 5.3: Sprint backlog continued

ID	Story	Sprint Backlog Tasks	Date
9	As a driver I should be able to log in to my account	Make a driver list in the database	31/03/2016
		Implement code to facilitate logging in by drivers	31/03/2016
10	As a driver I want to pick a vehicle and a route	Implement route assigning algorithm	01/04/2016
		Implement packing algorithm	01/04/2016
		Implement a google map using the google maps API on the driver interface	02/04/2016
		Implement a checklist of all items on the driver's route on the driver interface	02/04/2016
		Display contact details of client on web page	03/04/2016
		Implement a checklist option to choose between routes on the before page of the driver interface	03/04/2016
11	As a driver I need to see the contact details of my next delivery / pickup	Display the contact details of current pickup/ delivery	04/04/2016
12	As a driver I want to check off the packages I drop off / pickup	Implement a checklist for the packages on the driver's interface	04/04/2016
13	As a sender I need to know when the parcel will be picked up	Implement code to calculate the estimated time of pickup	05/04/2016
		Display the value of the calculated time of pickup on the sender interface	06/04/2016
14	As a sender I need to be able to know when the package will be delivered	Calculate the ETA when the payment, and allocation of a package has been made	06/04/2016
		Send the user a notification with the ETA	07/04/2016
		Update the ETA on the user's interface	08/04/2016
15	As a receiver I need to receive a notification of the intended delivery	Implement code to send a notification when the ETD has been calculated	09/04/2016

Table 5.4: Sprint backlog continued

ID	Story	Sprint backlogs	Date
16	As a receiver I neeed to check the estimated time of delivery	Display the ETD when it has been calculated	10/04/2016
17	As the CEO I want to see statistics on the clients	Calculate all the statistics on clients	11/04/2016
		Display the statistics on the CEO interface	11/04/2016
18	As the CEO I want to see statistics on the drivers	Calculate all the statistics on drivers	12/04/2016
		Display the statistics on the CEO interface	12/04/2016
19	As the Vehicle Maintenance Manager I want to see vehicle statistics such as mileage	Implement an algorithm that computes vehicle statistics	13/04/2016
		Display the results of the algorithm	13/04/2016
20	As the Vehicle Maintenance Manager I want to schedule the maintenance of vehicles	Implement an algorithm that computes the need of a vehicle to undergo maintenance	14/04/2016
		Add an undergo maintenance button	14/04/2016
		Display the results of the algorithm	15/04/2016
		Update the list of available vehicles	15/04/2016
21	As the Accounts Officer I want to see the list of payments made (for bookeeping purposes)	Fetch the payments information from the database	16/04/2016
		Display the payment information summary	16/04/2016
22	As the Accounts Officer I need to see a list of vehicle maintenance payments (for bookeeping purposes)	Fetch the payments information from the database	17/04/2016
		Display the payment information summary	17/04/2016
23	As the CEO I want to generate reports on income	Generate a pdf/report of the information the CEO is viewing	18/04/2016
		Have a generate report button on the CEO's interface	18/04/2016

Table 5.5: Sprint backlog continued

ID	Story	Sprint backlogs	Date
24	As the CEO I want to generate reports on expenditure	Generate a pdf/report of the information the CEO is viewing	19/04/2016
		Have a generate report button on the CEO's interface	19/04/2016
25	As a driver I want to be able to request a day off	Have a request day off button on the page before page of the driver interface	20/04/2016
		Implement code to take the driver off the available drivers list	22/04/2016
26	As a driver I should be able to log out of my account	Implement code to facilitate logging out by drivers	23/04/2016
27	As a sender I need to be able to log out of my account	Make a log out button on the web page	24/04/2016
		Revoke the user's access	24/04/2016
28	As the Vehicle Maintenance Manager I want to see vehicle statistics such as fuel consumption		26/04/2016