SOFTWARE ENGINEERING ELEN4009

LABORATORY EXERCISE 3

Sheena Philip, Linda Khumalo, Kessigan Subramanium, Phumzile Dhlwathi

April 7, 2016

1 USER MANUAL

1.1 NECESSARY INFORMATION TO CHECK REPO ON GITHUB

- Go to: github.com/kessigan/Transport_System
- clone the repo
- Go to the folder Lab3/Project

1.2 SOFTWARES TO INSTALL

- Install Anaconda
- django 1.9.4
- postgresql 9.5.1 - the password is "password" the user is "postgres"
- psycopg2-2.6.1
- pgadminIII

1.3 SET-UP BEFORE THE PROJECT IS RUN

- make the database
  - Go to pgAdminIII
  - log in to the postgres server
  - right click on Databases

  - select New Database
  - under Name, type MainDB
  - click OK - a database is now created

1.4 Description of how to compile code

- Go to the folder Lab3/Project/CourierJZ
- type : python createDatabases.py . The user has now added three tables to the MainDB created earlier.
- Go to the folder Lab3/Project/CourierJZ
- open terminal in this location
- type : python manage.py runserver . Django server is now running
- Go to browser (preferably Chrome). Type in localhost:8000/courier/main

1.5 Description of the front end

In order to use the service the user first needs view the main page. This page is located at localhost:8000/courier/main. From this page the user is able to select their designation, the four options available are driver, sender, receiver and dispatcher. The user will then need to submit their login details, for this version the super-user email is admin and the password is also admin. At this stage the sender is able to fill a form to send a package, the receiver is able to submit a form to see the package status, the dispatcher is able to see the current packages that remain in the warehouse and the driver is able to see a map of the area he is in as well as the list of packages to be picked up or dropped off.

2 Description of project

2.1 PROBLEM STATEMENT

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.

2.2 PROJECT OBJECTIVES

- To design, implement and test a web-based management system for a fleet of delivery and pick-up vehicles
- The website must serves 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.

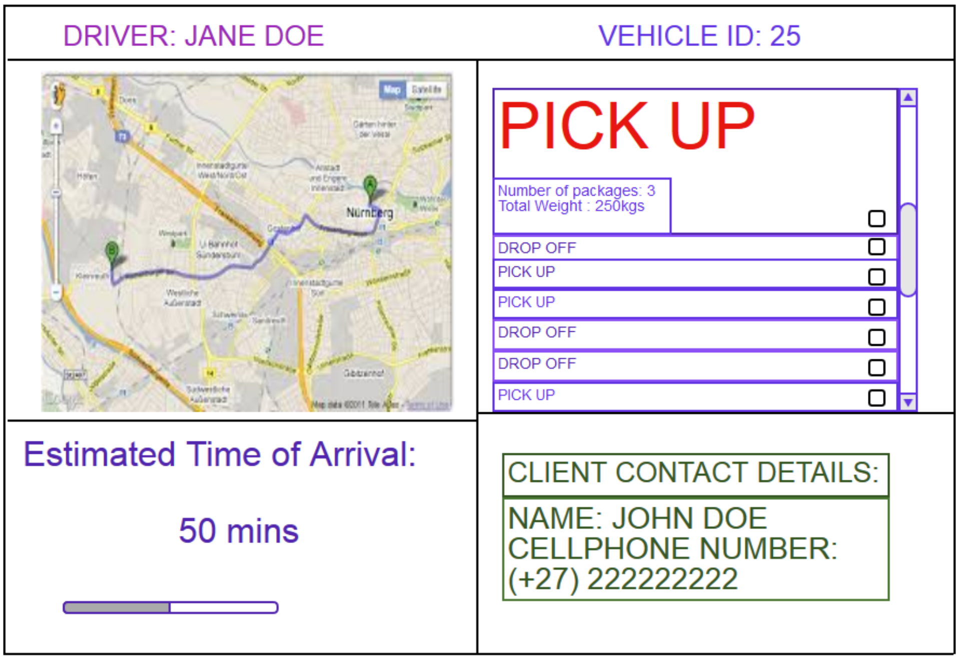
- The website must act as a repository to store all the data associated with the business activities.

2.3 Current Implementation of Project

A prototype of the project has been created thus far. The following features have been implemented in terms of their front-end

2.3.1 The driver interface/view

Figure 2.1 displays the view of the driver.The current prototype has deviated from the designed view in that the estimated time of arrival does not appear and the client contact details do not appear.



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

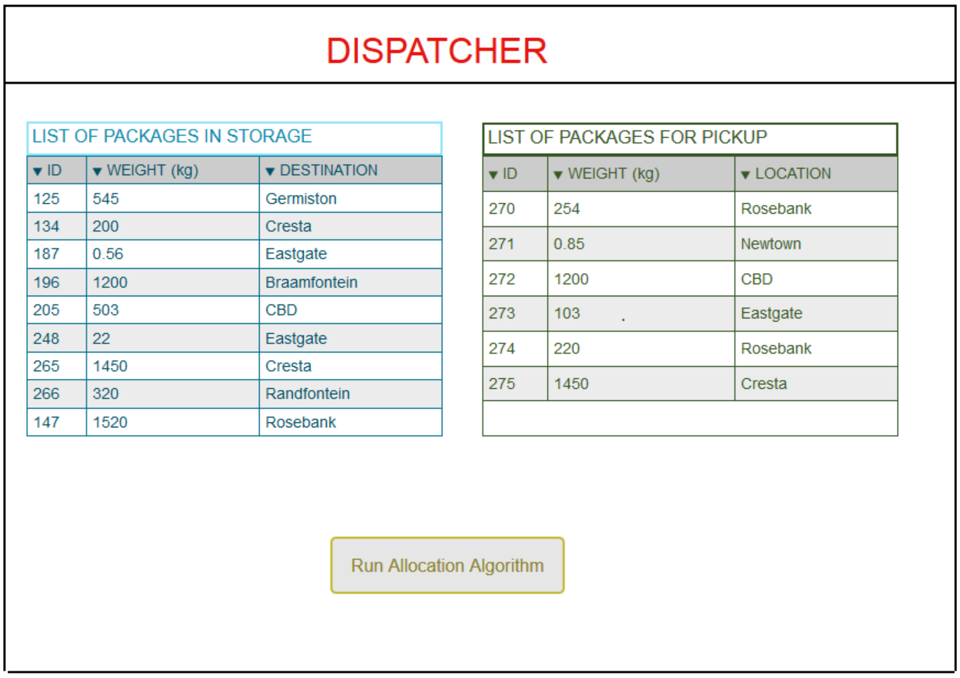
Figure 2.1: The driver interface

2.3.2 The dispatcher view

Figure 2.2 displays the dispatcher view before the dispatcher runs the algorithm to assign packages to drivers and routes. The current prototype achieves this view.

2.3.3 The receiver view before user submits

In order for the receiver to track the progress of their package they have to enter the package ID sent to them via email notification. Figure 2.3 shows the page where the user enters their



DISPATCHER

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 2.2: The dispatcher interface before running the allocation algorithm

package id. The current prototype achieves this view.

2.3.4 The receiver view after the user submits

Figure 2.4 shows the interface the receiver sees after they have entered their package details. This view shows the package details and estimated time of delivery.

2.3.5 The sender view

Figure 2.5 shows the sender view. The prototype deviates from the design in that the account information doesn't not appear and the estimated time of pick-up and delivery are not displayed.

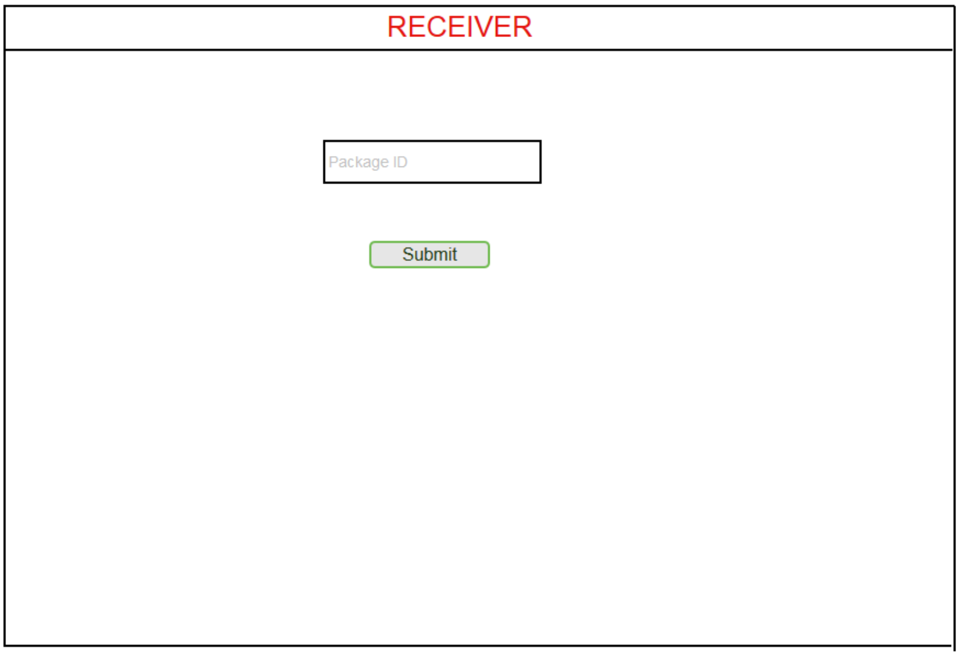
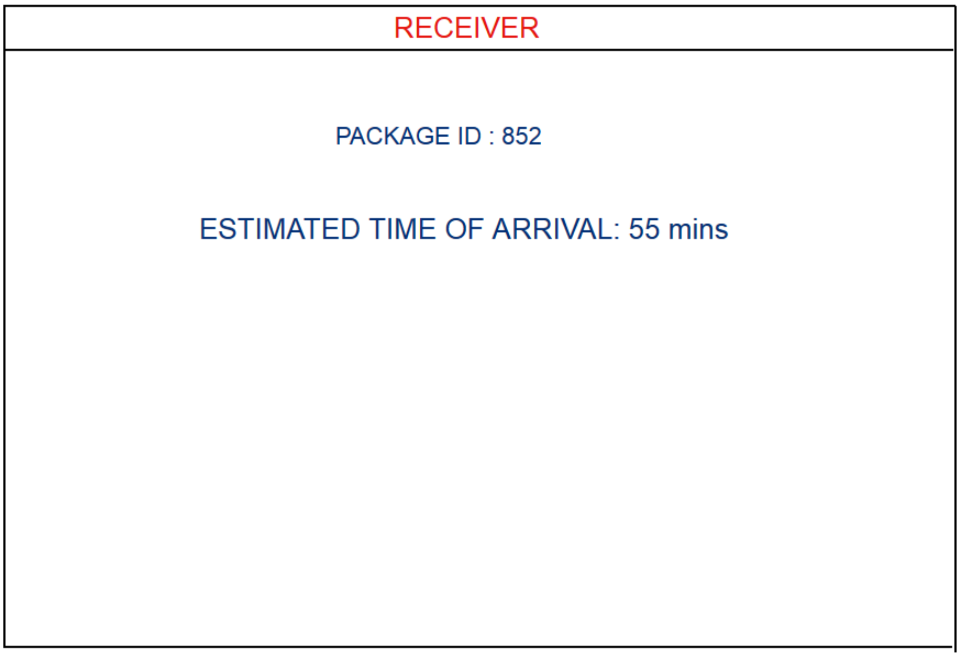


Figure 2.3: The receiver interface



PACKAGE ID : 852

ESTIMATED TIME OF ARRIVAL: 55 mins

Figure 2.4: The receiver interface after the user submits query

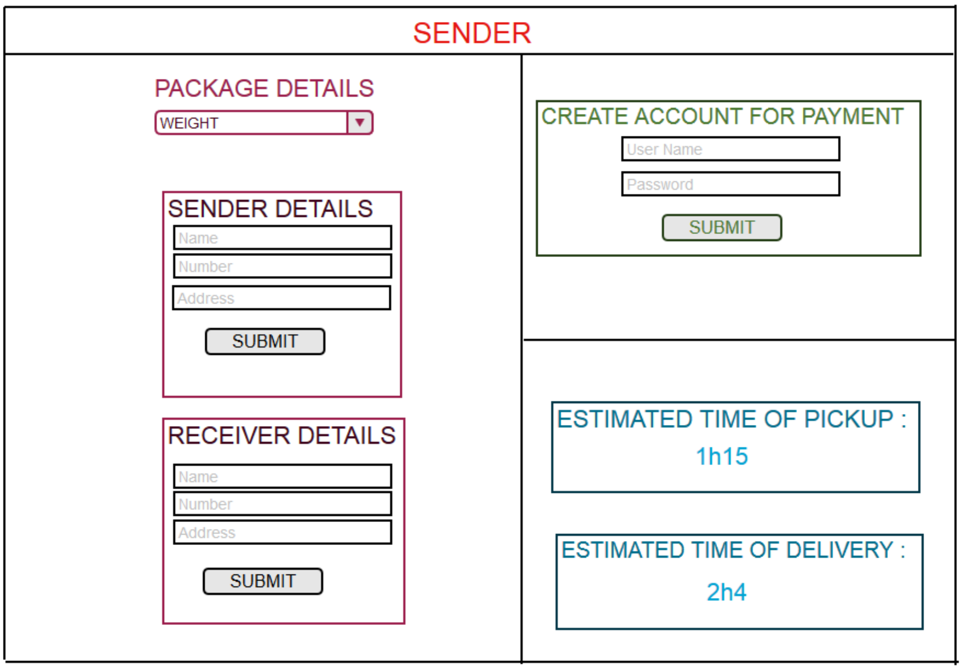


Figure 2.5: The sender interface

The back-end is used to store and process all information as requested by the users. In its current form of the project the user information is stored on the database, information may be added to the database from form submissions on the user pages, this includes the submission of a pick-up requests with the relevant delivery details and the user been able to track a package. The information is sent from the webpage to the server. The server then processes the data and returns user specific information.

3 Responsibility of each pair of students

3.1 BACK–END : Linda, Phumzile

- make a database and populate it
- implement algorithms which can calculate things related to the front end, such as:
  - scheduling
  - capacity
  - goods transferred/pickup
- set up a server
- link database to server

- link front end to server
- write unit tests

3.2 FRONT END : Kessigan, Sheena

- Make user interfaces for the different users
  - Choose a bootstrap template and make minor modifications
  - Plot the views and visualisations using JavaScript libraries such as d3, google charts, am charts on the template Users: For each user there is a User Interface based on the design:
    - Program
    - scheduling
    - Dispatcher (inputs the variables so the algorithms can run and return a result)
    - capacity
    - goods transferred/pickup
    - location
    - Truck driver
    - schedule
    - goods transferred/pickup
    - location
- Write unit tests