Merchant Integration

A PROJECT REPORT

Submitted in partial fulfillment of the

requirements for the award of the degrees

of

BACHELOR OF TECHNOLOGY

in

ELECTRICAL ENGINEERING

Submitted by:

Sahil Yadav

Guided by:

Dr. Amod C. Umarikar , Associate Professor, Electrical Engineering, IIT Indore

INDIAN INSTITUTE OF TECHNOLOGY INDORE (NOVEMBER 2019)

CANDIDATE'S DECLARATION

I hereby declare that the project entitled **"Merchant Integration"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Electrical Engineering' completed under the supervision of **Dr. Amod C. Umarikar**, **Associate Professor, Electrical Engineering, IIT Indore** is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

……

Sahil Yadav

CERTIFICATE by BTP Guide

It is certified that the above statement made by the student is correct to the best of my knowledge.

……

Amod C. Umarikar, Associate Professor, Electrical Engineering, IIT Indore

Preface

This report on “Merchant Integration" is prepared under the guidance of Amod C. Umarikar, Associate Professor, Electrical Engineering, IIT Indore .

Through this report I have tried to give a detailed design how to integrate a payment service and food merchants. I have also highlighted some of the technology that was used during the projects.

I have tried to the best of my ability to explain the content in a lucid manner. I have also included flow diagram to make it more illustrative.

Sahil Yadav

B.Tech. IV Year

Discipline of Electrical Engineering

IIT Indore

Acknowledgements

It is my privilege to express my thankfulness to several persons who helped me for progress of the project. I express my gratitude to my mentors Balaji Srinivasan, Able Johnson (Software Engineer at Dunzo Digital Private Limited) for their guidance throughout the project.

I also thanks to my friends who played a vital role for completion of project.

This project done during the internship was my first step in software development, it helped in exploring many amazing technologies and I am sure that it will help me in future.

Sahil Yadav

B.Tech IV Year

Discipline of Electrical Engineering

IIT Indore

Abstract

This report summarizes the work done by me on Dunzo App who delivers food items, grocery items and any others items of reasonable size. I contributed to this APP by adding one more payment service (lazypay) and food merchants (petpooja , limetray and faasos) by considering all error scenarios

Table of Contents

Lazypay Integration

❑ What is LazyPay ?

- **LazyPay** is India's fastest way to get credit online.

- **LazyPay** is an innovative payment service that has been built to provide an easy payment option for the day-to-day purchases.

- **LazyPay** offers ease of transaction at multiple merchants. You have flexibility to club all **LazyPay** transactions and make consolidated payments through **LazyPay** dashboard.

❑ Purpose Of Project

- The Purpose of this project is to add another payment option so that Customers have more options to pay the money.

❑ Web FrameWork and Database Used

● Django (Web FrameWork)

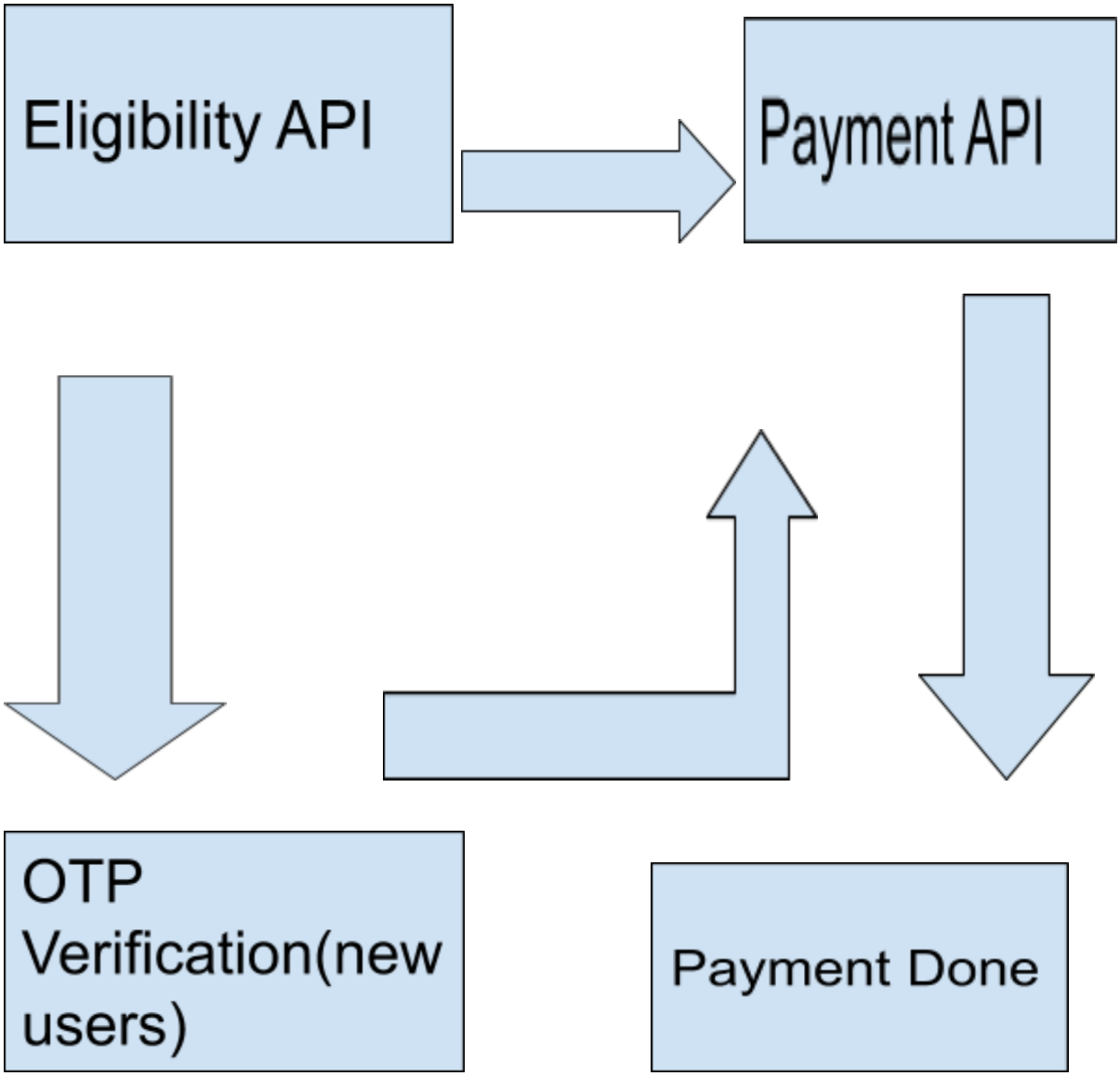
○ Django is a high-level Python Web framework encouraging rapid development and pragmatic, clean design.

○ A web application framework is a toolkit of components all web applications need.

● PostGreSql (DataBase)

○ PostgreSQL is a powerful, open-source object-relational database system that uses and extend the SQL language combined with many features that safely store and scale the most complicated data workloads.

❑ Flow Diagram



❑ Explanation Of The APIS

● Eligibility API

○ Eligibility API is called to check whether user is eligible to pay through LazyPay or not.

○ Cases like User is blocked, User has exceeded maximum transaction limit, User has exceeded monthly transaction limit etc. are checked.

● OTP Verification API

○ When User is transacting first time using LazyPay then OTP Verification is required.

○ OTP is sent to user's mobile number and after verification they are redirected to Payment API

● Payment API

○ This is the final API called which will deduct the money from user's LazyPay account.

❑ Explanation Of The Flow

● When user arrive at payment page then first eligibility API is called if user is eligible (i.e not blocked by lazypay and transaction limit for
that user is not exceeded etc.) then lazypay option is shown to that user. If User does not have LazyPay account then LazyPay Option will not be visible to the user. If User is not eligible to

pay money (blocked, exceeded maximum transaction limit, exceeded maximum monthly transaction limit etc.) then LazyPay Option is greyed out.

- If user is making payment first time (by checking in database) then first, user will go through OTP verification. After OTP Verification, a unique access token (with expiry period of 180 days) is sent from LazyPay end for that user and we will store that token in database so that if user makes payment some other time then he/she don't need to go through OTP Verification.

- If Access token is present in Database for that user then only payment API is called and amount will be deducted from user's lazypay account.

❑ DataBase Schema

User id	Mobile number	Email id	Access Token	Expiry Date Of Access Token

Contribution

- I worked on Eligibility, register (at the time of OTP verification) and OTP verification APIS including their end to end testing, handling all the error cases.

- I have added a cron (time-based job scheduler) which will refund the amount to users if amount is deducted from the lazypay account but not received by us. In this, status (by calling get status API) is checked for all the transaction that happened in a particular day, if it is marked failed in our database but successful from lazypay side then amount will be refunded to the user.

- A rigorous testing was done for the whole flow to avoid any error case.

Food Integration

❑ What is Food Integration?

- Food Integration means to ingest and update the food items through API calls.

- In Food Integration, We integrated with food merchants PetPooja, Faasos and Limetray. All these merchants are already integrated with many restaurants. So integrating with them is equivalent to integrating with many restaurants.

❑ Purpose of the project

- Previously menu ingestion, updation and out of stock details were done manually by uploading the sheets. So the purpose of this project is to reduce manual intervention and make a service which is more scalable and convenient for menu ingestion, updation.

❑ Web FrameWork and Database used

- **Django (Web FrameWork)**

  - Django is a high-level Python Web framework encouraging rapid development and pragmatic, clean design.
  - A web application framework is a toolkit of components all web applications need.

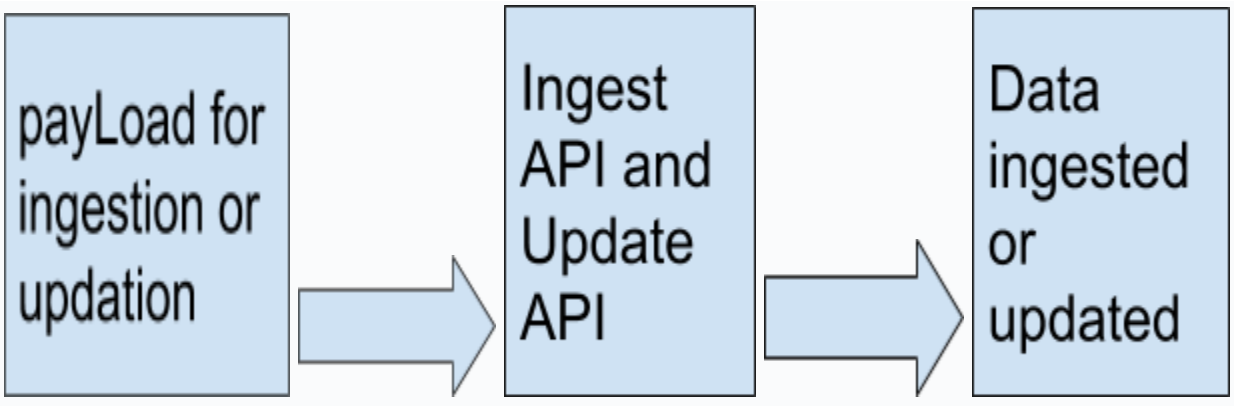
- **MongoDB (DataBase)**

  - MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time
  - MongoDB is a distributed database at its core, so high availability, horizontal scaling, and easy to use.

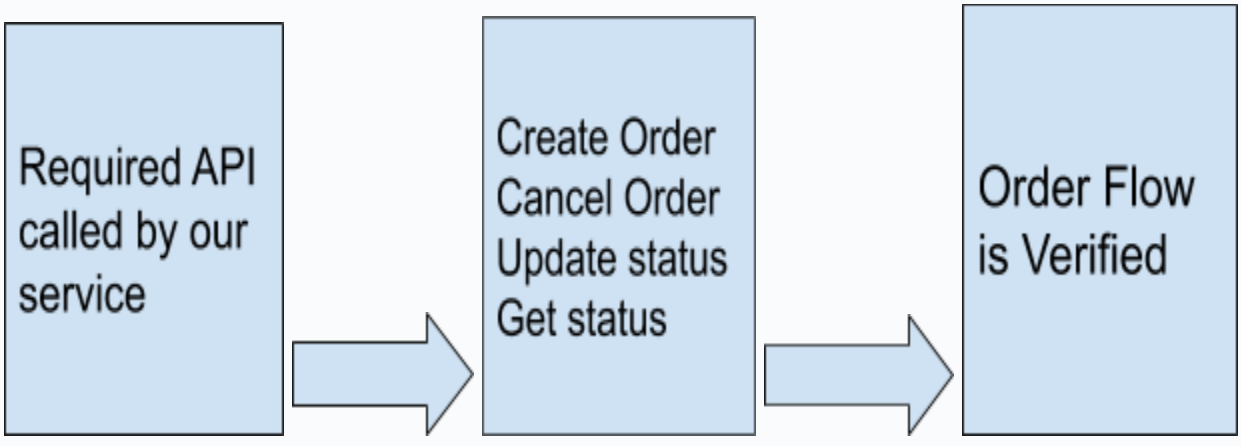
❑ Flow Diagram

- **For Menu Ingestion and Updation**

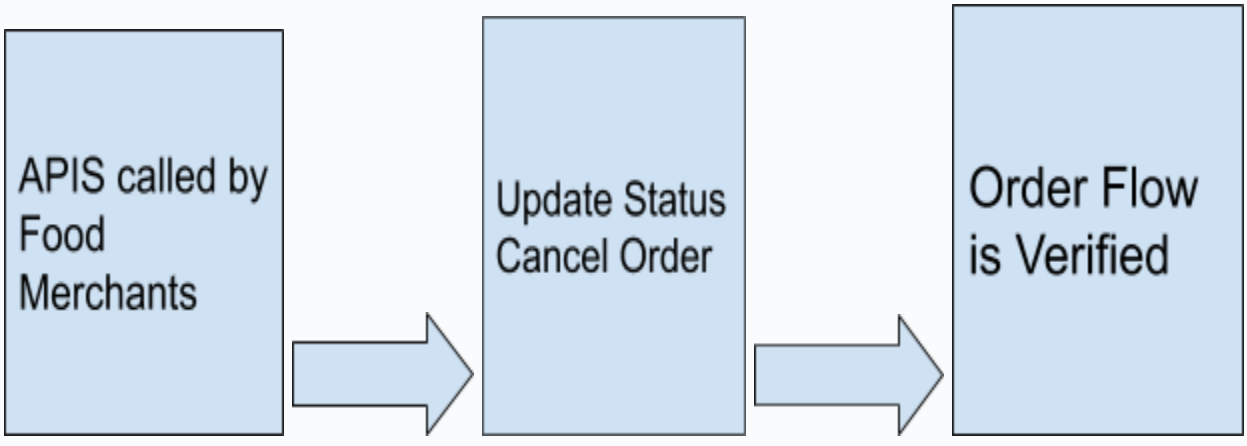
  - Whatever menu is sent by food merchant we need to ingest or update that menu so that same menu reflects on the APP.



- **Order Flow**
  - Order flow handles flow like creating, cancelling, updating status, and getting status of an order. This Flow also takes care which state conversion is possible using state machines.
  - APIS of food merchant called by our service.

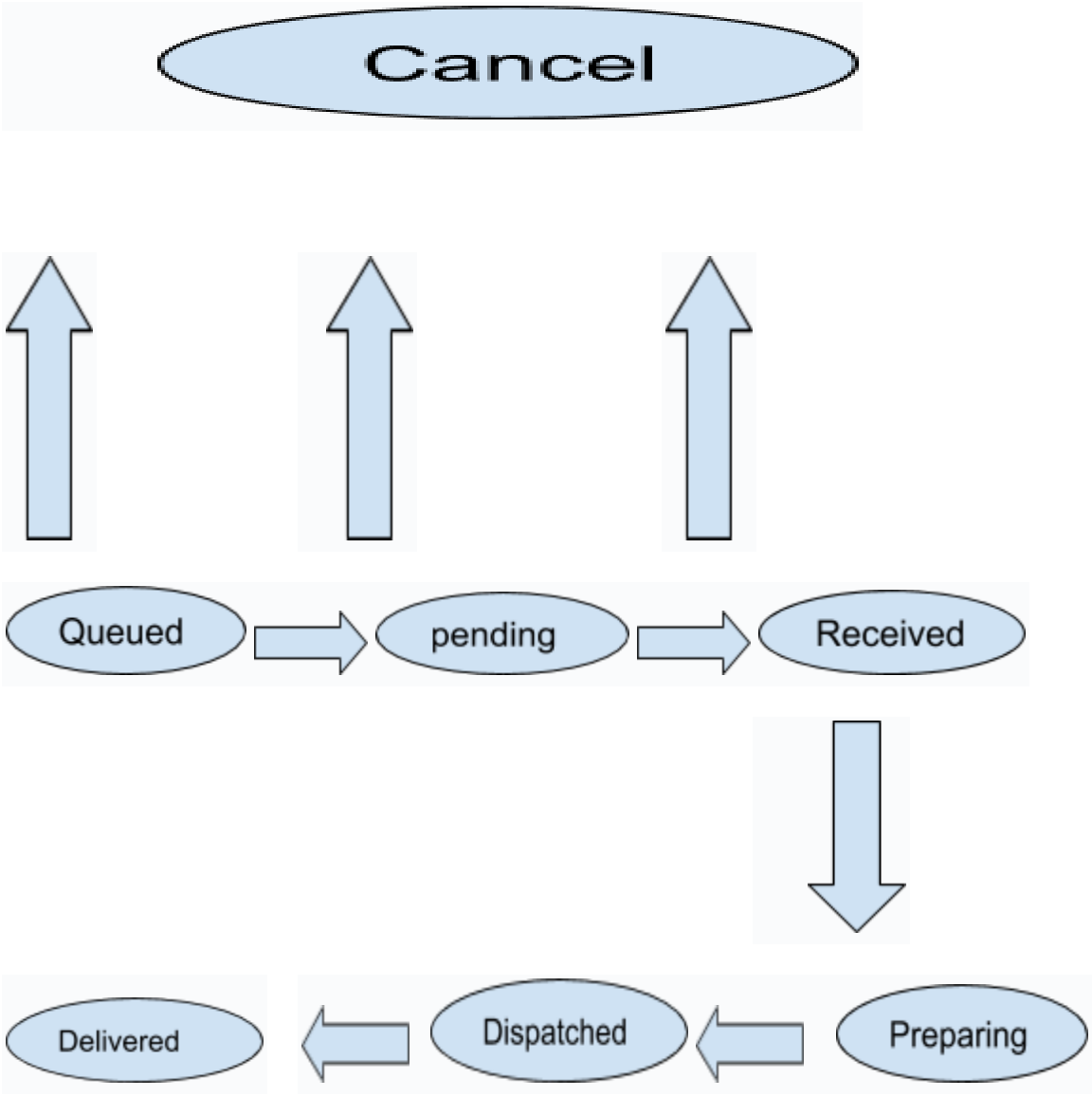


- APIS written in our service called by food merchants

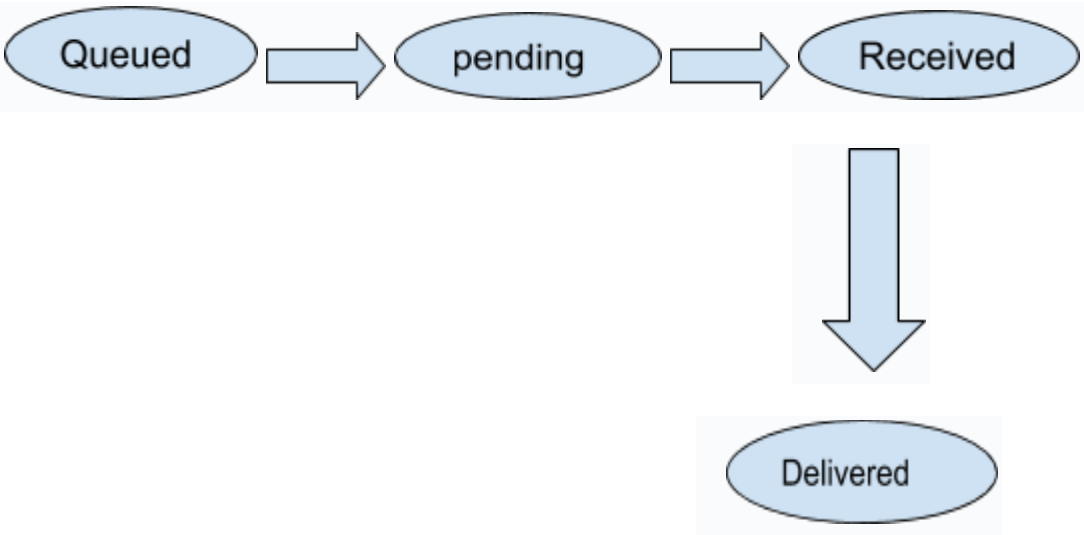


❑ State Machine For Order Flow

- **For PetPooja**



- **For Faasos and Limetray**



❑ Explanation Of The Flow

- Menu Ingestion, Updation and Order flow have asynchronous flow.

- Food Merchant will hit our APIS for full ingestion or updation.

- Full Ingestion means previous data for that merchant will be erased and new data will be inserted in database.
- Updation means updating one or more fields.

- Food merchant will send payLoad (in JSON format) in their format (defined in the contract by merchants) that needs to be changed in our format. First, when food merchant hit our APIS to ingest the payLoad, we are putting these requests in celery (which is an asynchronous task queue) and then worker will pick request one by one in first in first out order and process those requests. Some Fields in the input data are necessary field, So, first we are validating those required fields, if these required fields are missing then we are throwing error in response. After validation, we are changing data to our format (supporting all AddOns and Variants). After that, we are saving the data in the database. Android team takes data from database to show the results on the APP.

● Result on App

STARTERS | PIZZAS | GOURMET PIZZA | PASTAS | DESSE

Cheese & Garlic Bread

Crispy French loaf topped with garlic oil and fresh mozzarella cheese

₹ 99

+ ADD

Customizable

Spicy Garlic Bread

Crispy French loaf topped with cheese spread mix flavoured with green chilli and parsley

₹ 69

+ ADD

Customizable

Spinach & Corn Garlic Bread

Crispy French loaf topped with cheese spread mix flavoured with green chilli, sweet corn and spinach

₹ 79

+ ADD

Customizable

French Fries

Crisp, salted potato strips with a sprinkle of parsley, served with peri peri mayo sauce

₹ 99

+ ADD

Herbed Wedges

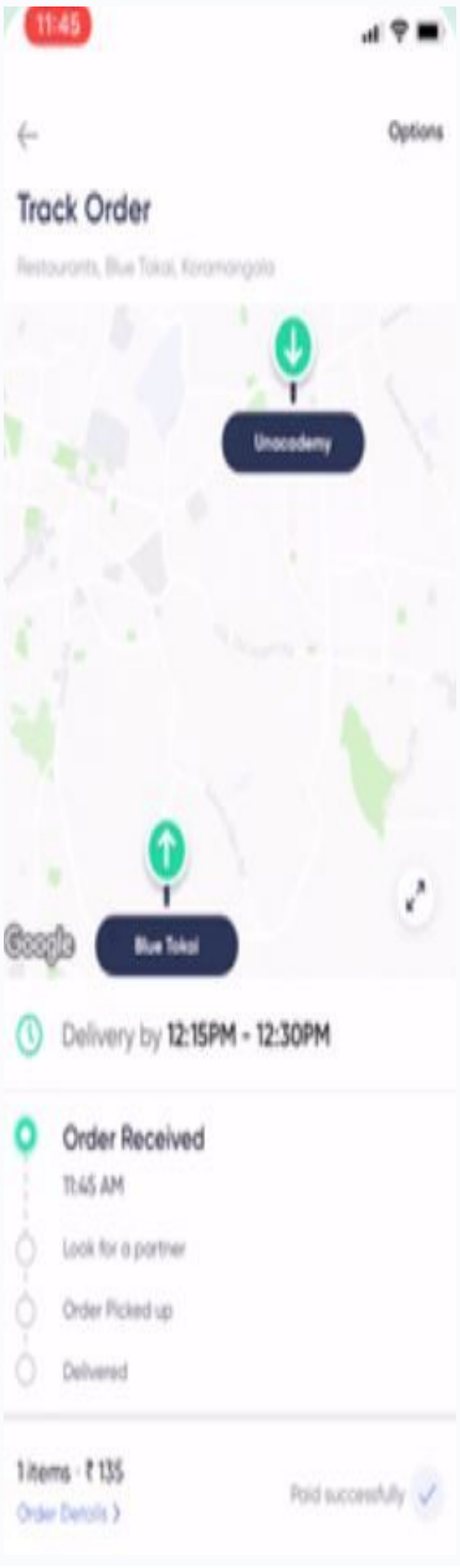
Crisp, fried potato wedges sprinkled with parsley, served with peri peri mayo sauce

₹ 129

+ ADD

- When customers from our APP place the order in any of the restaurants of these food merchants. We call their APIS passing providerId, quantity etc for the food items which were purchased. The flow is asynchronous here i.e. we are putting order request in celery (asynchronous task queue) and will be assigned to one of the workers. If response is green from their end then order will be placed. Whenever update status API is called, we are first checking through state machines whether state change is possible or not, if possible we are changing the state otherwise we are returning an error.

- **Result on App**


11:45
Options
Track Order
Unacademy
Blue Tokai
Google
Delivery by 12:15PM - 12:30PM
Order Received
11:45 AM
Look for a partner
Order Picked up
Delivered
1 items · ₹ 135
Order Details
Paid successfully


SUPPORT
Track order
Order Food - Glen's Bakehouse, Indiranagar
Glen's Bakehouse
Office
Indiranagar
Delivery by 4:10 PM
Items have been picked up
Partner has picked up your items and is on his way to the delivery location
Hari Prasad
4 items · ₹450
Order Details
Paid successfully

Contribution

- I worked on full ingestion for petpooja and faasos, order flow in petpooja and limetray including testing and all the error cases.

- I also worked on two cron (time-based scheduling job), first cron changes the status to cancel state if order is in pending state for 10 minutes, second cron continuously updates the status of order in our service by calling get status API of food merchant.

- Rigorous testing for full flow was done.

Conclusion

- I have integrated lazypay which was written in python language and postGreSql was used as database.
- I worked on food integration in which I contributed in integrating food merchants petpooja, limetray and faasos.
- I worked on many other tasks or bugs that appeared in testing phase in lazypay and food integrations.
- I also worked on some tasks (goLang) that were assigned to me.
- In future, I will try to do more better projects from knowledge gained from the above projects, try to use more technologies which can give better outputs.

References

- https://www.docker.com
- https://en.wikipedia.org
- http://www.celeryproject.org/
- https://www.uisources.com/explainer/dunzo-app-store-screenshots
- https://www.uisources.com/explainer/dunzo-placing-an-order
- http://www.medium.com
- https://stackoverflow.com/
- https://www.postgresql.org/
- https://www.mongodb.com/