# PAyTEL

Marsol Daman, Lauren Lingeman, Axel Van Hoyweghen,
Alexander Dapoz, Anthony Calandra , Noah McGivern

CSI-4999-Senior-Design

Fall 2018

# 1. Project Vision

## 1.1. Backgrounds

PAyTEL was originally thought of by the predefined project description given to us by Professor Patel. Here is the background is given in the project description: “Credit card fraud isn’t new to any of us. Fraudulent use of credit cards is at all time high with the growth of cyber retail of goods and services where identity check of the buyer is an issue. To cope with this, credit card issuers are constantly in search of better methods of identity check for an online retail transaction. One recent addition to this system is SMS message based code verification as an extra step for verifying buyer’s identity. In this project you will design a new system for Secure Transaction. The system consists of three pieces, a web portal that manages all transactions, a buyer mobile app for approval of the transaction and a seller mobile app for issuing a charge. The front-facing mobile phone camera will be used for authenticating the biometric identity of the buyer when approving the transaction. The server will mediate the messaging, manage user profile and transactions.”

PAyTEL is a multichannel payment system for secure transactions for buyers and sellers.

## 1.2. Socio-economic Impact, Business Objectives, and Gap Analysis

The Socio-economic Impact for PAyTEL is going to impact the buyers and sellers in the economic marketplace and allow merchants and consumers to check out and pay per transaction. It will impact the mobile app payment marketplace and allow more secure banking for mobile applications for merchants and consumers. Our business objectives are to allow merchants and consumers to have secure mobile banking as well and have the ability to issue charges as well as customize user profiles. Another business objective will be for merchants and consumers to search transaction history. Our desire for PAyTEL is to lead the mobile banking app industry to a new frontier, with clean, efficient, user interface, along with in-depth user profile customization. The actual performance of PAyTEL will be determined by the volume of transaction requests created by merchants and consumers that find the application and utilize its full potential.

## 1.3. Security and ethical concerns

Our focus for PAyTEL will be fast but secure transactions for merchants and consumers. It will have encrypted data along the entire transaction process. Ethically, we will not allow the average user to access any transaction apart from their own as well as limiting displayed.

### 1.4. Glossary of Key Terms

## 2. Project Execution and Planning

### 2.1. Team information

#### 2.1.1. Alexander Dapoz

Front-end development and documentation.

#### 2.1.2. Anthony Calandra

Application development and documentation.

#### 2.1.3. Axel Van Hoyweghen

Lead back-end (server, database, etc.) design and implementation and documentation.

#### 2.1.4. Lauren Lingeman

Application development and managing documentation.

#### 2.1.5. Marsol Damon

Application development and documentation.

#### 2.1.6. Noah McGivern

Front-end development and documentation.

### 2.2. Tools and Technology

#### 2.2.1. Android Studio

Android Studio is Android's official IDE. It is purpose-built for Android to accelerate your development and help you build the highest-quality apps for every Android device.

### 2.2.2. Java

Java is an object-oriented programming language

### 2.2.3. React

A JavaScript library for building user interfaces for web applications.

#### 2.2.3.1. HTML and CSS

HyperText Markup language (HTML) and Cascading Style Sheets are the basic languages used to create a website or web application.

#### 2.2.3.2. Javascript

JavaScript is a high-level, interpreted programming language. Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web.

### 2.2.4. Node.JS

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser.

### 2.2.5. Amazon Web Services (AWS)

#### 2.2.5.1. API Gateway

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.

#### 2.2.5.2. Cognito

Amazon Cognito lets you add user sign-up, sign-in, and access control to your web and mobile apps quickly and easily. Amazon Cognito scales to millions of users and supports sign-in with social identity providers, such as Facebook, Google, and Amazon, and enterprise identity providers via SAML 2.0.

#### 2.2.5.3. DynamoDB

Amazon DynamoDB is a nonrelational database that delivers reliable performance at any scale.

2.2.5.4. Lambda

AWS Lambda lets you run code without provisioning or managing servers and scales as your application grows.

2.2.5.5. Mobile Hub

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.

2.2.5.6. Rekognition

Rekognition allows you to automatically identify objects, people, text, scenes, and activities, as well as detect any inappropriate content.

2.2.5.7. S3

Object storage built to store and retrieve any amount of data from anywhere.

## 2.3. Project plan

### 2.3.1. Sprint 1 (9/6 - 9/13)

2.3.1.1. Define requirements
2.3.1.2. Layout basic sprint schedule
2.3.1.3. Setup collaboration tools (GitHub, Slack, Google Drive)
2.3.1.4. Select development environment, languages, tools, etc. and get all team members access (Android Studio, AWS DynamoDB, Bootstrap, Java, Android SDK 8, Bootstrap, Groovy, Kotlin, Java JDK, CSS, HTML, Javascript)
2.3.1.5. Begin project documentation

### 2.3.2. Sprint 2 (9/13 - 9/27)

2.3.2.1. Setup AWS (Cognito, IAM, Mobile Hub) for user account creation and authentication
2.3.2.2. Design consistent UI/UX for application
2.3.2.3. Setup QA for testing for user authentication system
2.3.2.4. Research and test phone facial recognition system for user authentication

2.3.2.5. Design database for storing transactions and user information
2.3.2.6. Write user stories for Sprint 3

### 2.3.3. Sprint 3 (9/27 - 10/13)

2.3.3.1. Implement UI/UX for application (login, registration, user account page, main landing page)
2.3.3.2. Connect application to login server and database storing user information
2.3.3.3. Design UI/UX for transaction portal
2.3.3.4. Setup transaction portal login at backend for admin
2.3.3.5. Setup QA for testing transaction portal login
2.3.3.6. Write user stories for Sprint 4

### 2.3.4. Sprint 4 (10/13 - 10/25)

2.3.4.1. Implement UI/UX for application (settings, anything else unfinished)
2.3.4.2. Implement UI/UX for transaction portal (login, main landing page, database queries)
2.3.4.3. Finalize database design
2.3.4.4. Setup QA for testing database queries and storage
2.3.4.5. Write user stories for Sprint 5

### 2.3.5. Sprint 5 (10/25 - 11/8)

2.3.5.1. Implement UI/UX for transaction portal (anything unfinished)
2.3.5.2. Debug and testing for whole application
2.3.5.3. Feature lock by end of this sprint

### 2.3.6. Sprint 6 (11/8 - 11/22)

2.3.6.1. Continue debug and testing
2.3.6.2. Finish final documentation
2.3.6.3. Prepare final presentation and poster board

### 2.4. Best standards and Practices

## 3. System Requirement analysis

### 3.1. Functional requirements

3.1.1. User must be able to create a user account based on their Google login
3.1.2. User must be able to offer payment to another user (buyer role)
3.1.3. User must be able to accept payment from another user (seller role)
3.1.4. User must pass face scan to access their personal account
3.1.5. User must be able to see their transaction history
3.1.6. User must be able to change their payment method through the settings page
3.1.7. User must be able to generate a QR code
3.1.8. User must be able to scan a QR code
3.1.9. Admin must be able to access all transactions
3.1.10. Admin must be able to report malicious transactions to users

### 3.2. Non-functional requirements

3.2.1. User must have a valid Google login consisting of username and password
3.2.2. User must have a valid credit card
3.2.3. User must have an phone running Android API 26 or higher
3.2.4. Database must store user account information securely
3.2.5. Database must store all transactions
3.2.6. Server must be able to handle all incoming transactions real-time
3.2.7. Website and mobile app must both use the same color palette throughout.
3.2.8. Data must be encrypted when transferred between web-app / mobile-app and the server
3.2.9. Web-app interface must be responsive for variable screen sizes
3.2.10. Server must remove pending transactions that have timed out

### 3.3. On-screen Appearance of landing and other pages requirements

### 3.4. Wireframe designs

## 4. Functional Requirements Specification

### 4.1. Stakeholders

4.1.1. Project team members
4.1.2. Businesses which support our service as a form of payment

4.1.3. PAyTEL users
4.1.4. Local government agencies looking to prevent fraud and cybercrime

## 4.2. Actors and Goals

### 4.2.1. Actors

4.2.1.1. PayTel Application User
4.2.1.2. PayTel Administrator
4.2.1.3. Database

### 4.2.2. Goals

4.2.2.1. Reduce credit card fraud by creating system for secure transactions utilizing facial recognition.
4.2.2.2. Create user-friendly application to allow secure transactions for easy exchange of money.

## 4.3. User Stories, scenarios and use cases

### 4.3.1. User Stories

4.3.1.1. As a user, I want to be able to create an account on PayTEL that is tied to my personal google account.
4.3.1.2. As a user, I want to be able to login to my PayTEL account using my personal google account.
4.3.1.3. As a user, I want to make sure my account is secure by using facial recognition in addition to my login information.
4.3.1.4. As a user, I want to be able to add information to my PayTEL profile including credit card details and name.
4.3.1.5. As an administrator, I want to be able to access the PayTEL web portal to view transactions and user information.
4.3.1.6. As a user, I want to be able to login to my PAyTEL account on the PAyTEL android app.
4.3.1.7. As a user, I must be able to create a transaction through my PAyTEL account.
4.3.1.8. As a user, I must be able to view my transaction history on my account.
4.3.1.9. As a user, I should be able to update my profile information after account creation.
4.3.1.10. As a user, I want to make sure any initiated transaction is secure through facial recognition.
4.3.1.11. As an admin, I must be able to manage user accounts and transactions from the database, including being able to flag particular accounts or transactions.

4.3.1.12. As a user, I want to be sure my data is secure in the database.

### 4.3.2. Scenarios

4.3.2.1. Bob is out with Joe and wants to buy a popsicle from the popsicle stand. The popsicle stand only accepts cash, which Bob has none of. He does, however, have his phone. Joe offers to pay for Bob's popsicle if he pays him back. Bob agrees, but instead of having to go all the way to the bank, he wants to be able to be able to simply pull out his phone and transfer money to Joe directly.

4.3.2.2. Lucy is terrified of credit card fraud and will only use the most secure transaction applications. She wants to secure it with facial recognition because she never takes or allows anyone to take pictures of her so it would be a very secure method for her.

4.3.2.3. Joey plays betting games with his friends all the time and constantly loses. He would like an easy, quick way to make good on the bets. He doesn't want to have to enter credit card information all the time or carry cash. He just wants to be able to enter their usernames to send them the cash.

### 4.3.3. Use Cases

4.3.3.1. Use case: Create user account
Entry condition: On main app login page
Trigger: Enters google login information
Exit Condition: Clicks submit
Flow of Events:

a) User opens app
b) User enters valid google login
c) User clicks submit
d) Database sees user does not exist in database
e) User enters all required profile information
f) User takes required picture
g) User clicks submit
    i) Successful submit: user taken to landing page
    ii) Unsuccessful submit: user prompted to check entered info

4.3.3.2. Use case: Login to user account
Entry condition: On main app login page

Trigger: Enters google login information
Exit Condition: Clicks submit
Flow of Events:

a. User opens app
b. User enters valid google login
c. User clicks submit
d. Database sees user exists in database
e. User taken to landing page

4.3.3.3. Use case: Send transaction to other user
Entry condition: User is logged in
Trigger: User selects make transaction
Exit condition: User submits transaction request
Flow of Events:

a. User enters transaction information, including valid username
b. User submits transaction request
c. User is taken back to home page with pending transaction showing

4.3.3.4. Use case: Receive transaction from other user
Entry condition: User is on home page
Trigger: User selects the pending transaction
Exit condition: User accepts or denies transaction
Flow of Events:

a. User clicks accept or deny
b. If accepted, user must take a picture and pass facial recognition
c. If facial recognition is passed, transaction is completed
d. If denied, transaction does not go through
e. In either case transaction status is updated on transaction page
f. User is taken back to home page

4.3.3.5. Use case: Change user settings
Entry condition: User is on home page
Trigger: User selects settings
Exit condition: User clicks submit
Flow of Events:

a. User makes any desired changes to profile settings
b. User clicks submit when satisfied
c. User is taken back to home page

4.3.3.6. Use case: Admin log in
Entry condition: Admin is on web portal
Trigger: Enters admin login information
Exit condition: Admin clicks login
Flow of Events:

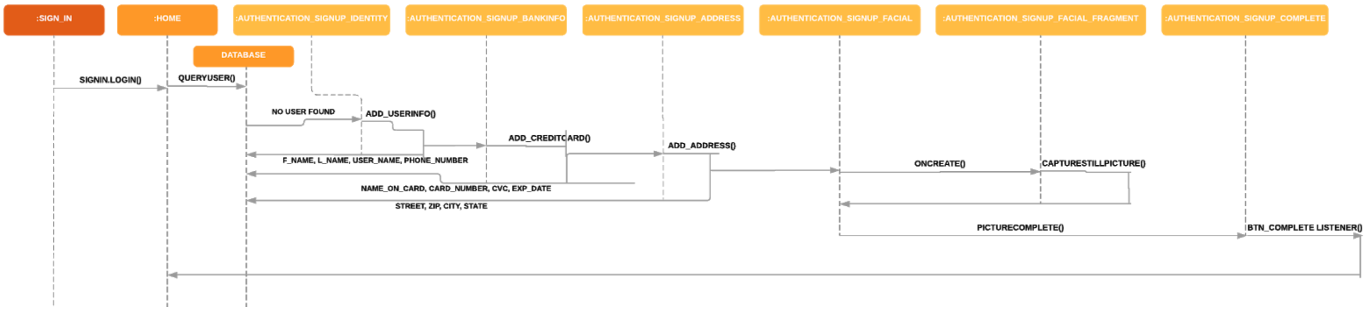
a. Admin enters login information when prompted at web portal
b. If successful, admin is taken to web portal home
c. If unsuccessful, admin is prompted to re-enter login information

4.3.3.7. Use case: Flag transaction
Entry condition: Admin is logged into web portal
Trigger: Admin selects transaction to flag
Exit condition: Admin submits flag
Flow of Events:

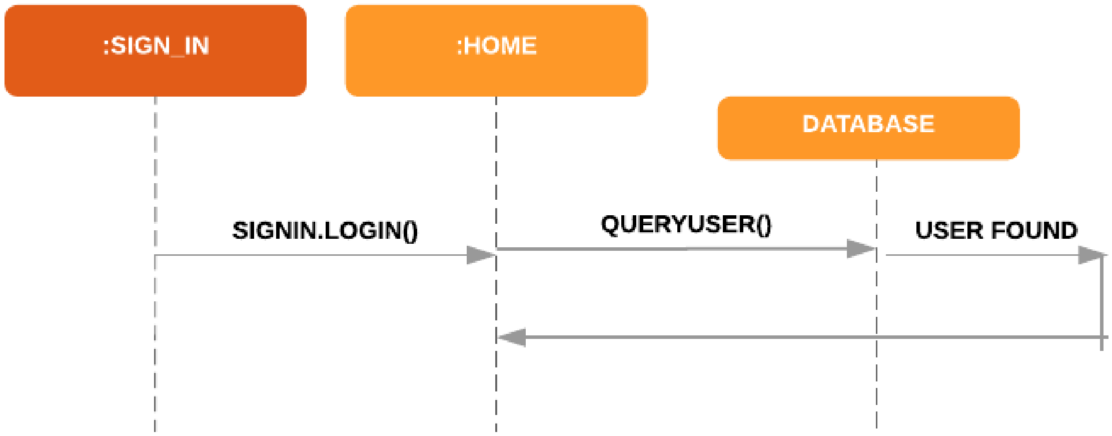
a. Admin goes to transaction table in web portal
b. Admin selects one or many transactions
c. Admin submits transactions to flag
d. Selected transactions now appear as flagged

## 4.4. System Sequence / Activity Diagrams

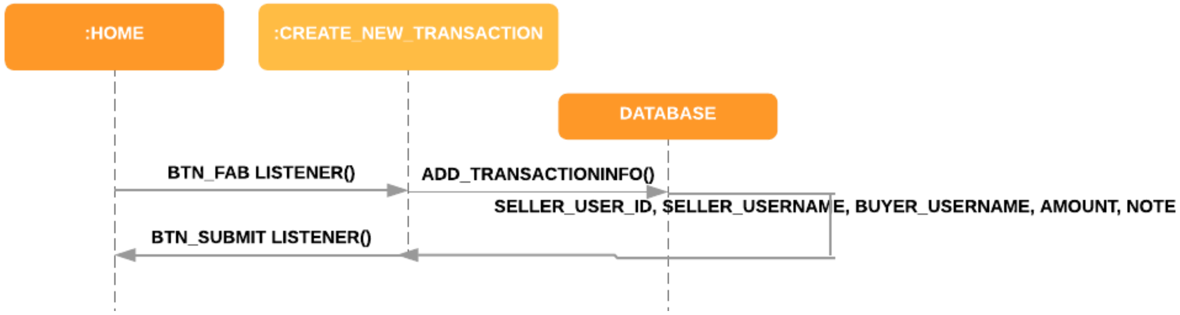
Create User Account


:SIGN_IN
:HOME
:AUTHENTICATION_SIGNUP_IDENTITY
:AUTHENTICATION_SIGNUP_BANKINFO
:AUTHENTICATION_SIGNUP_ADDRESS
:AUTHENTICATION_SIGNUP_FACIAL
:AUTHENTICATION_SIGNUP_FACIAL_FRAGMENT
:AUTHENTICATION_SIGNUP_COMPLETE
DATABASE
SIGNIN.LOGIN()
QUERYUSER()
NO USER FOUND
ADD_USERINFO()
ADD_CREDITCARD()
ADD_ADDRESS()
F_NAME, L_NAME, USER_NAME, PHONE_NUMBER
NAME_ON_CARD, CARD_NUMBER, CVC, EXP_DATE
STREET, ZIP, CITY, STATE
ONCREATE()
CAPTURESTILLPICTURE()
PICTURECOMPLETE()
BTN_COMPLETE LISTENER()

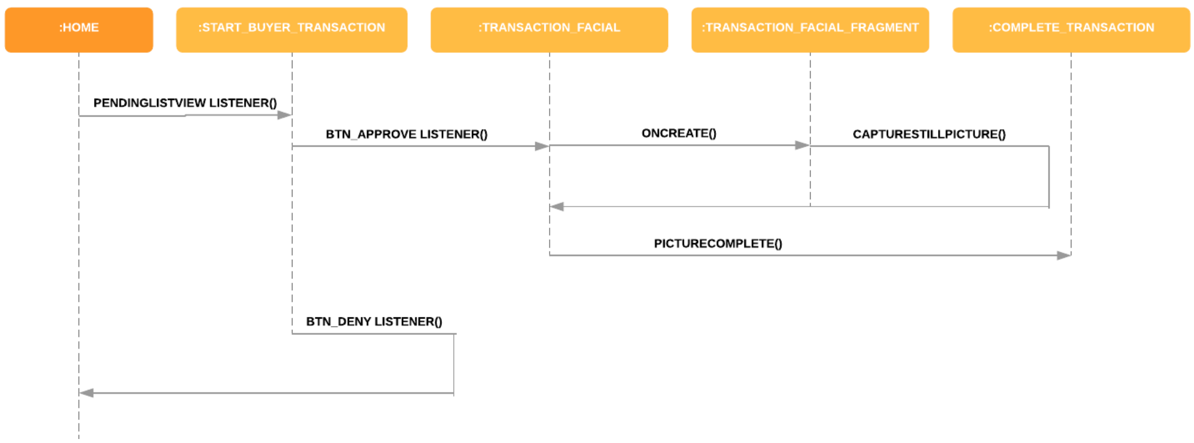
## Login to User Account



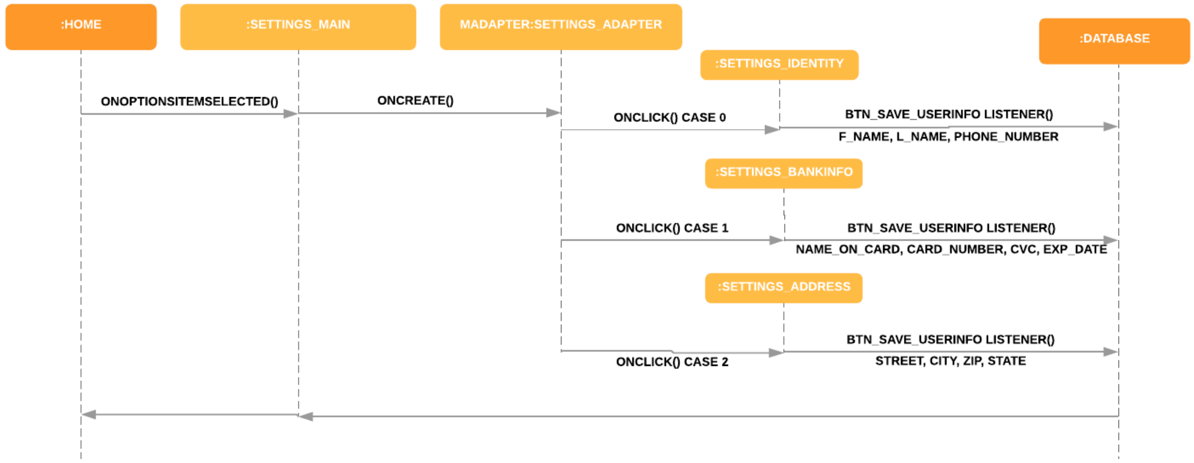
## Send Transaction to Other User



Receive Transaction from User



Change User Settings



# 5. User Interface Specifications

## 5.1. Preliminary Design

## 5.2. User Effort Estimation

### 1. Actors and Use Cases

| Actor Name | Description | Complexity | Weight |
|---|---|---|---|
| PAyTEL User | Player interacts with system via GUI | Complex | 3 |

| | | | |
|---|---|---|---|
| PAyTEL Admin | User interacts with system via web portal | Complex | 3 |
| Database | Database is another system interacting through a protocol | Average | 2 |

| Use Case Name | Description | Complexity | Weight |
|---|---|---|---|
| Create user account (UC-1) | Simple UI. 7 steps required. 2 required actors (User, Database). | Average | 10 |
| Login to user account (UC-2) | Simple UI. 5 steps required. 2 required actors (Player, Database). | Average | 10 |
| Send transaction to other user (UC-3) | Simple UI. 3 steps required. 2 required actors (Player, Database). | Average | 10 |
| Receive transaction from other user (UC-4) | Simple UI. 6 steps required. 2 required actors (Player, Database). | Average | 10 |
| Change user settings (UC-5) | Moderate UI. 3 steps required. 2 required actors (Player, Database). | Average | 10 |
| Admin login (UC-6) | Simple UI. 3 steps required. 2 required actors (Admin, Database). | Average | 10 |
| Flag transaction (UC-7) | Simple UI. 4 steps required. 2 required actors (Admin, Database). | Simple | 5 |

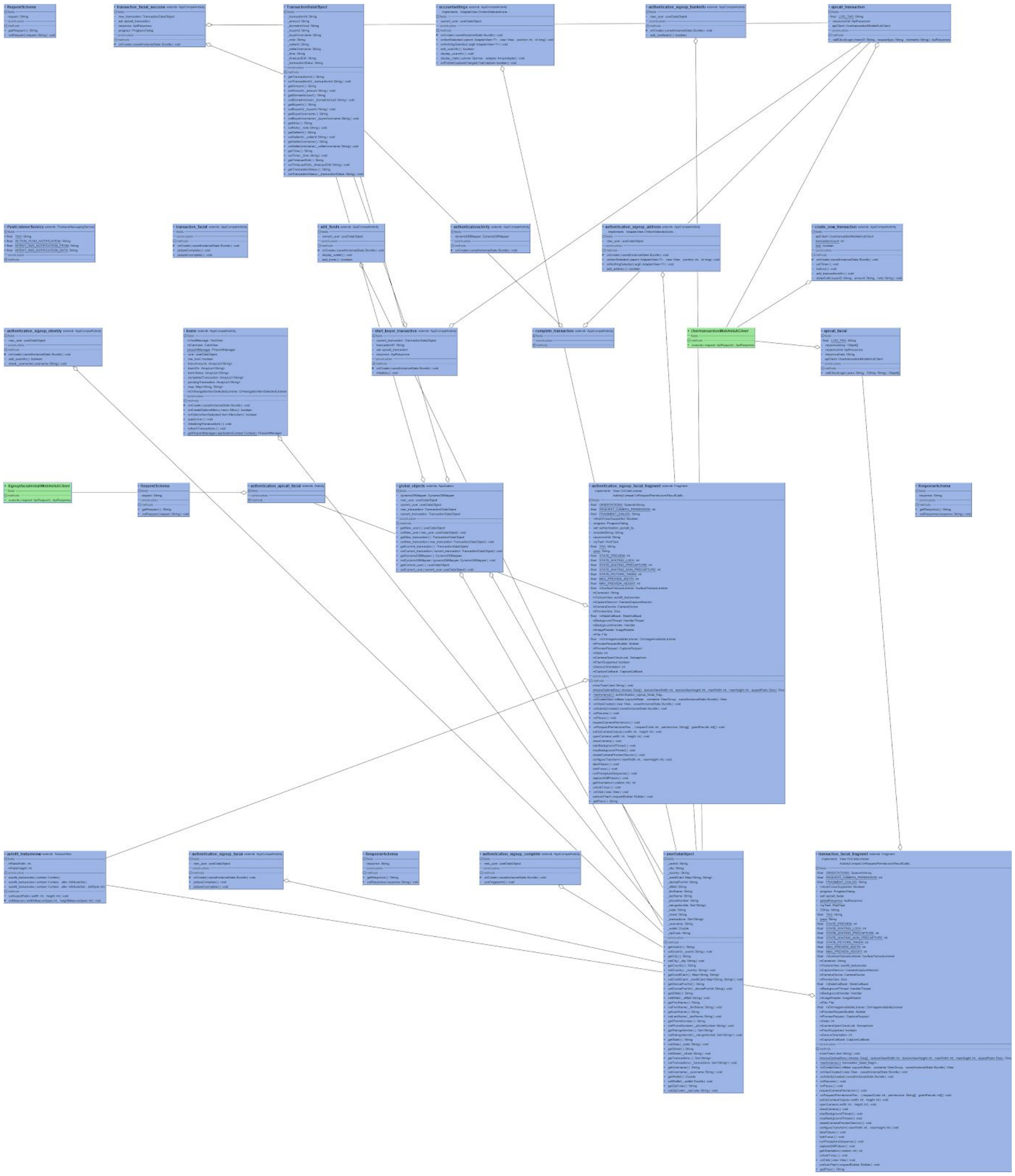
**2. Calculate UUCP**

UAW = 2x3 + 1x2 = 6 + 2 = 8
UUCW = 1x5 + 6x10 + 0x15 = 5 + 60 = 65

UUCP = UAW + UUCW = 8 + 65 = **73**

# 6. Static Design

## 6.1. Class model


RequestSchema
transaction_facial_success extends AppCompatActivity
TransactionDataObject
accountsettings extends AppCompatActivity
authentication_signup_bankinfo extends AppCompatActivity
apicall_transaction
PushListenerService extends FirebaseMessagingService
transaction_facial extends AppCompatActivity
add_funds extends AppCompatActivity
authenticatoractivity extends AppCompatActivity
authentication_signup_address extends AppCompatActivity
create_new_transaction extends AppCompatActivity
authentication_signup_identify extends AppCompatActivity
home extends AppCompatActivity
start_buyer_transaction extends AppCompatActivity
complete_transaction extends AppCompatActivity
UsertransactionMobileHubClient
apicall_facial
SignupfacialauthMobileHubClient
RequestSchema
authentication_apicall_facial extends Activity
global_objects extends Application
authentication_signup_facial_fragment extends Fragment
ResponseSchema
autofit_textureview extends TextureView
authentication_signup_facial extends AppCompatActivity
ResponseSchema
authentication_signup_complete extends AppCompatActivity
userDataObject
transaction_facial_fragment extends Fragment

### 6.2. System Operation Contracts

### 6.3. Mathematical Model

## 7. Dynamic Design

### 7.1. Sequence Diagrams

### 7.2. Interface Specifications

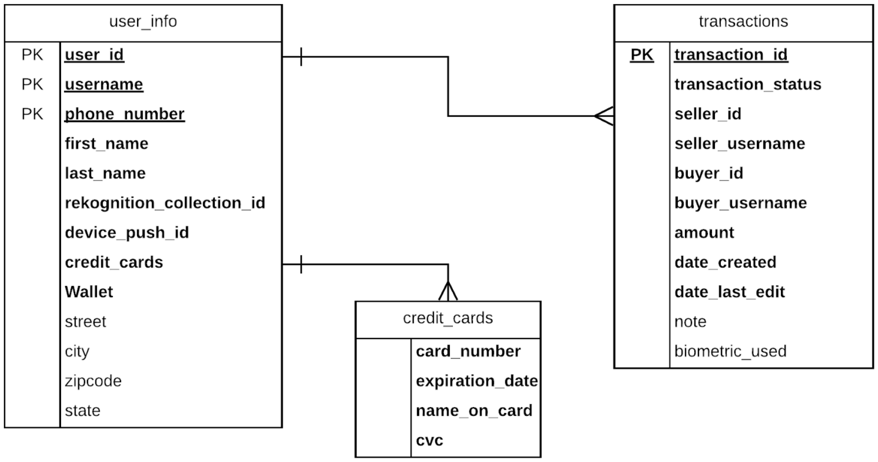
### 7.3. State Diagrams

## 8. System Architecture and System Design

### 8.1. Subsystems / Component / Design pattern Identification

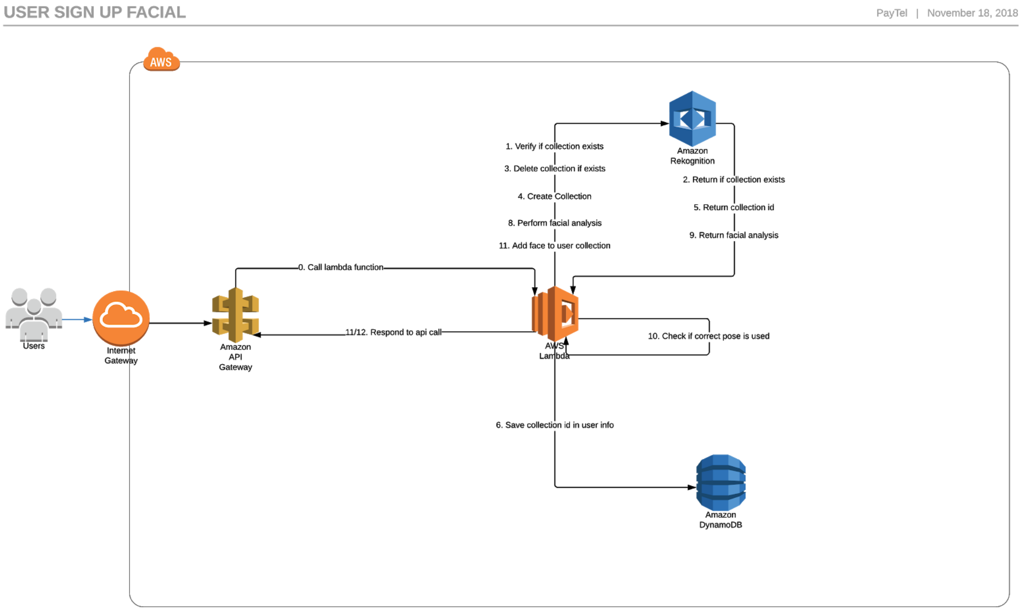
#### 8.1.1. Database

NoSQL database hosted on AWS DynamoDB. Utilizing a NoSQL database adds flexibility to the backend, so our data structure can evolve over time. Hosting the database on AWS DynamoDB will scale automatically when the number of users and transactions grow. The database utilizes 2 tables, user_info and transactions, which will store the users information and transactions.

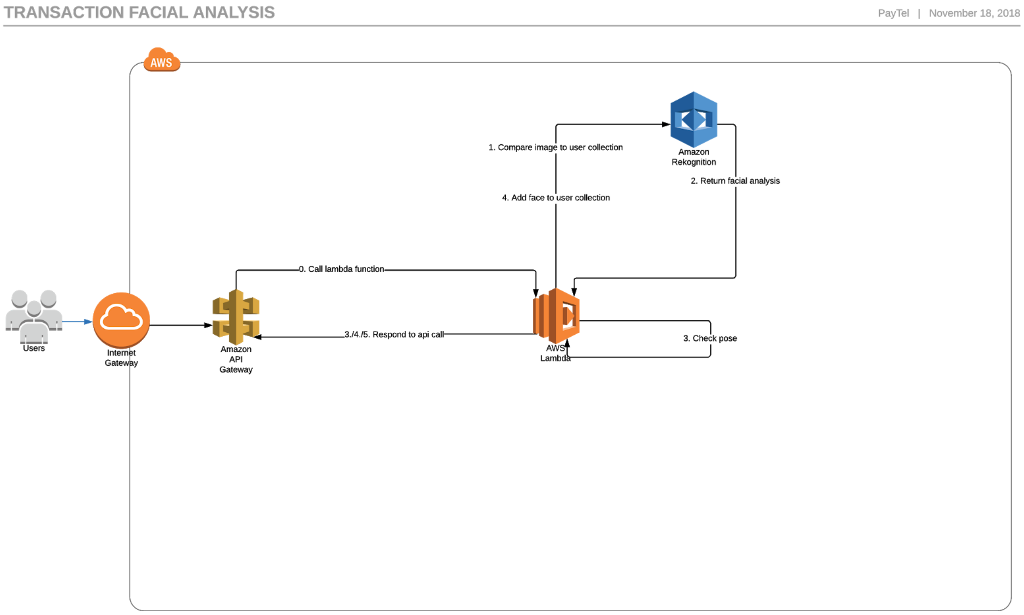


#### 8.1.2. Backend

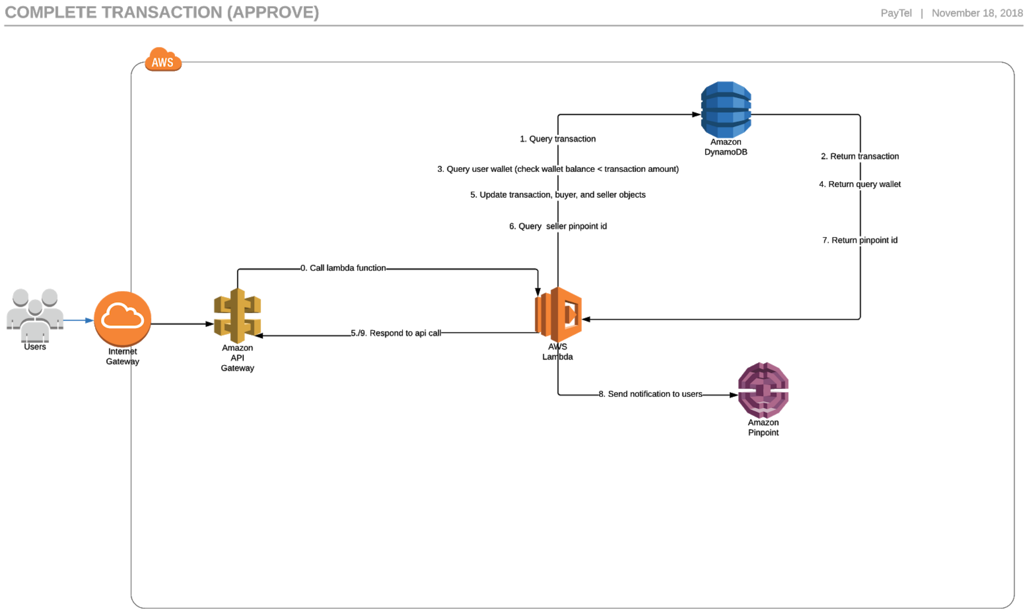
8.1.2.1. Facial verification on user sign up.


USER SIGN UP FACIAL
PayTel | November 18, 2018
AWS
Users
Internet Gateway
Amazon API Gateway
0. Call lambda function
11/12. Respond to api call
AWS Lambda
1. Verify if collection exists
3. Delete collection if exists
4. Create Collection
8. Perform facial analysis
11. Add face to user collection
Amazon Rekognition
2. Return if collection exists
5. Return collection id
9. Return facial analysis
10. Check if correct pose is used
6. Save collection id in user info
Amazon DynamoDB

### 8.1.2.2. Facial verification when buyer approves transaction.


TRANSACTION FACIAL ANALYSIS
PayTel | November 18, 2018
AWS
Users
Internet Gateway
Amazon API Gateway
0. Call lambda function
3./4./5. Respond to api call
AWS Lambda
1. Compare image to user collection
4. Add face to user collection
Amazon Rekognition
2. Return facial analysis
3. Check pose

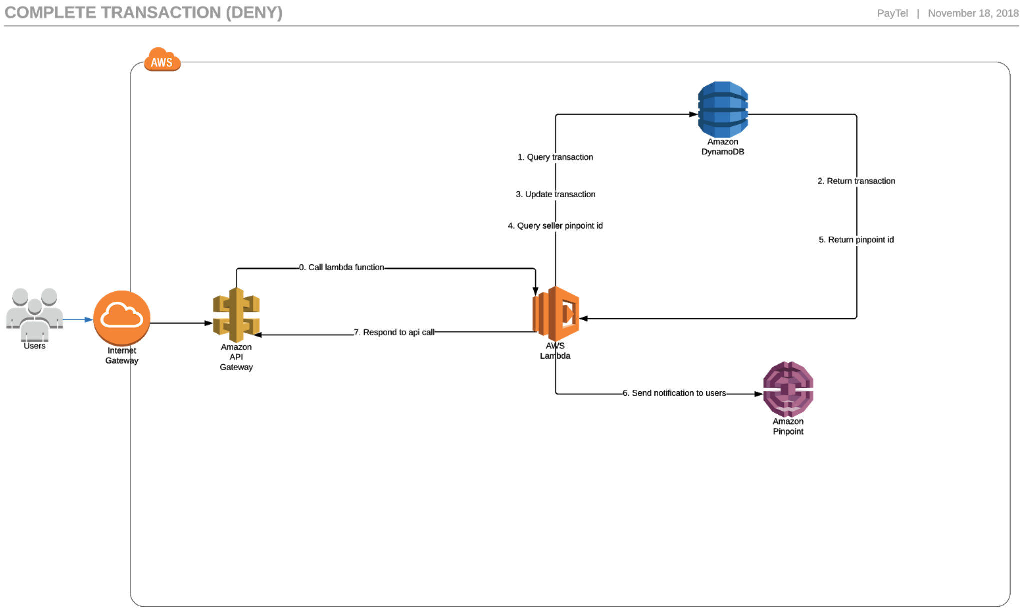
### 8.1.2.3. Complete transaction when buyer approves transaction and facial is verified.



COMPLETE TRANSACTION (APPROVE)

PayTel | November 18, 2018

AWS
Amazon DynamoDB
1. Query transaction
3. Query user wallet (check wallet balance < transaction amount)
5. Update transaction, buyer, and seller objects
6. Query seller pinpoint id
2. Return transaction
4. Return query wallet
7. Return pinpoint id
0. Call lambda function
5./9. Respond to api call
Users
Internet Gateway
Amazon API Gateway
AWS Lambda
8. Send notification to users
Amazon Pinpoint



COMPLETE TRANSACTION (DENY)

PayTel | November 18, 2018

## 8.2. Mapping Subsystems to Hardware (Deployment Diagram)

## 8.3. Persistent Data Storage

### 8.3.1. AWS S3

AWS S3 stores user images that are taken during transaction verification.

## 8.4. Network Protocol

## 8.5. Global Control Flow

## 8.6. Hardware Requirement

### 8.6.1. Android Application

8.6.1.1. Internet Access
8.6.1.2. Phone with Android OS (SDK 28 or higher)
8.6.1.3. Front-facing camera
8.6.1.4. Fingerprint Access

### 8.6.2. Admin panel

8.6.2.1. Internet Access

# 9. Algorithms and Data Structures

## 9.1. Algorithms

## 9.2. Data Structures

# 10. User Interface Design and Implementation

## 10.1. User Interface Design

## 10.2. User Interface Implementation

# 11. Testing

## 11.1. Unit Test Architecture and Strategy/Framework

## 11.2. Unit Test definition, test data selection

## 11.3. System Test Specification

## 11.4. Test Reports per Sprint

# 12. Project Management

## 12.1. Project Plan

## 12.2. Risk Management

# 13. References