SplitBill Application

User Manual

by Brute Force for Compiax

Authors:
Mia Gerber
Matthew Perry
Wanrick Willemse
Duart Breedt
Linda Potgieter

Contents

1 General Information

1.1 System Overview

The SplitBill application was made to facilitate the splitting of expenses based on a receipt image. The application uses optical character recognition technology to read item details from a bill. These details are displayed to the users on their devices and enables users to claim items from the bill. Claiming an item adds the item price to a displayed total which is the total amount that the user owes on the bill based on the items claimed. With this application, a large group of friends, co-workers or family members can easily calculate how much each person must pay for their own bill items.

Multiple users can be added to one bill by entering the code generated for the session when the bill is uploaded for display. This allows different users to claim items on the same bill, and when one user claims an item, the changes are propagated to all devices of currently active users. The application also has a built-in functionality that calculates the user's gratuity based on the calculated total. When an item is claimed, the user has the option to claim multiple instances of these items, and the user can also remove the items they have claimed if an error was made.

In addition to claiming items, the users are also presented with different functionalities to modify the currently displayed bill. The users have the options to add items or edit the currently displayed items. These changes are immediately displayed on all devices currently viewing the bill.

1.2 System Configuration

The server contains 3 docker containers. These 3 independent an run the API, MongoDB database and OCR Python module respectively. Communication between the 2 modules and the API is done through POST and GET requests sent and received by each component.

The mobile application, which is installed on any individual's phone, communicates with the API on the server using POST and GET requests over a network connection. Currently the application is being built for debugging on individual devices. Once SplitBill meets all user satisfaction criteria, it will be released on the Android store for public use. Currently, it is developed to run on *Android KitKat (4.4)* or newer; however, SplitBill is envisioned and has the potential to be released on additional platforms such as IOS.

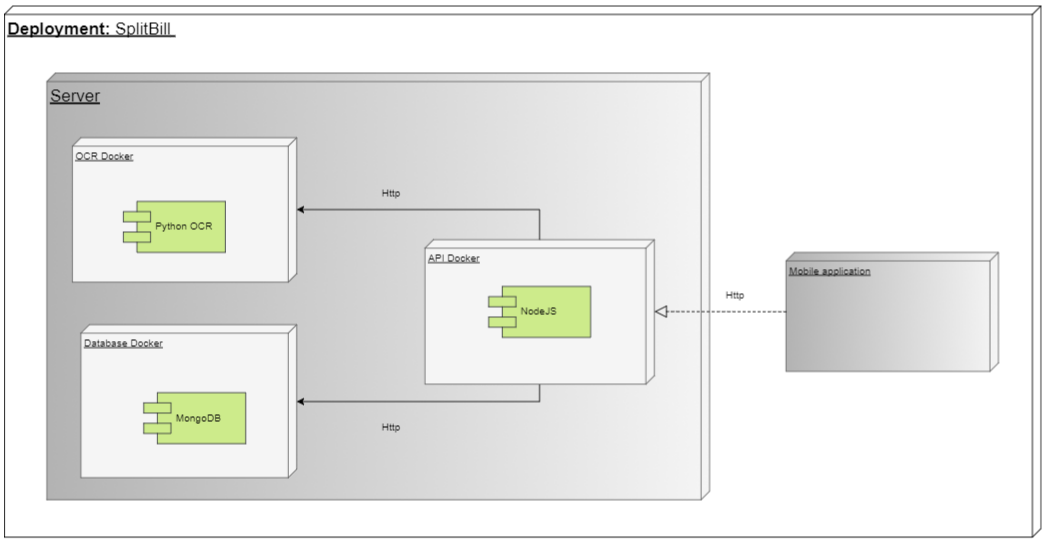

Figure 1: Deployment Diagram of the SplitBill system

1.3 Installation

Installation will be detailed as per module since modules can be swapped out for others with similar functionality if so required.

1.3.1 API Module

Initial requirements

1. Ensure Docker is installed on the computer that will run the server. Get Docker here: https://docs.docker.com/engine/installation/
2. Note the IP address of this computer on the network so that Platypus-Mobile can be set to connect to it.
3. Follow the instructions for Platypus-OCR in this document or here: https://github.com/COS301-brute-force/Platypus-OCR/wiki/Installation-Details to set up the OCR component for this API to connect to.

1.3.1.1 Docker

The following commands must be run in order to get the API up and running on the server. Due to the nature of Docker, it will download all dependencies and run them. Please be patient.

Environment: source/default

Build Docker:

```
docker build -t cos301-brute-force/platypus-api:source-latest
```

Run database:

```
 docker run \
-itd \
--rm \
--name platypus-db-source \
-p 27017 \
library/mongo:3.0.14
```

Run API:

```
docker run \
-it \
--rm \
-e "NODE_ENV=source" \
-e "DEBUG=platypus-api*" \
--name platypus-api-source \
--link platypus-db-source \
-v /home/node/platypus-api/node_modules \
-v $(pwd):/home/node/platypus-api \
-p 3000:3000 \
-p 3002:3002 \
cos301-brute-force/platypus-api:source-latest
```

Run tests:

```
docker run \
-itd \
--rm \
-e "NODE_ENV=source" \
-e "DEBUG=platypus-api*" \
--name platypus-api-source \
--link platypus-db-source \
-v /home/node/platypus-api/node_modules \
-v $(pwd):/home/node/platypus-api \
-p 3000:3000 \
-p 3002:3002 \
cos301-brute-force/platypus-api:source-latest
```

```
docker exec -it platypus-api-source /bin/bash
```

```
docker run \
 -it \
 --rm \
 -e "NODE_ENV=source" \
 -e "DEBUG=platypus-api*" \
 --name platypus-api-source \
 --link platypus-db-source \
 -v /home/node/platypus-api/node_modules \
 -v $(pwd):/home/node/platypus-api \
 -p 3000:3000 \
 -p 3002:3002 \
 cos301-brute-force/platypus-api:source-latest npm test
```

Environment: dev

Build:

```
docker build -t cos301-brute-force/platypus-api:dev-latest
```

Run database:

```
docker run \
 -itd \
 --name platypus-db-dev \
 -p 27017 \
 library/mongo:3.0.14
```

Run API:

```
docker run \
-it \
--rm \
-e "NODE_ENV=development" \
-e "DEBUG=platypus-api*" \
--name platypus-api-dev \
--link platypus-db-dev \
-v /home/node/platypus-api/node_modules \
-v $(pwd):/home/node/platypus-api \
-p 3000:3000 \
-p 3002:3002 \
cos301-brute-force/platypus-api:dev-latest
```

Run tests:

```
docker run \
 -it \
 --rm \
 -e "NODE_ENV=development" \
 -e "DEBUG=platypus-api*" \
 --name platypus-api-dev \
 --link platypus-db-dev \
 -v /home/node/platypus-api/node_modules \
 -v $(pwd):/home/node/platypus-api \
 -p 3000:3000 \
 -p 3002:3002 \
 cos301-brute-force/platypus-api:dev-latest npm test
```

1.3.1.2 Docker Swarm

This document gives step-wise instructions for building the docker swarm for the Platypus project as a whole.

Requirements

We are going to need multiple VMs to join together in a docker swarm. These could be docker machines, KVM instances or physical nodes. For this example, we will assume the following nodes:

- docker-swarm-manager-01 (192.168.3.30)
- docker-swarm-manager-02 (192.168.3.31)
- docker-swarm-manager-03 (192.168.3.32)
- docker-swarm-worker-01 (192.168.3.33)
- docker-swarm-worker-02 (192.168.3.34)
- docker-swarm-worker-03 (192.168.3.35)
- docker-swarm-worker-04 (192.168.3.36)

We need 3 managers, for redundancy and for consensus.

Initialize

On the first docker-swarm-manager:

```
docker swarm init
```

You need to join nodes to the docker swarm, see https://docs.docker.com/engine/reference/commandline/swarm_join/

```
docker service create \
  --name=swarm-visualizer-02 \
  -p 8080 \
  --constraint="node.role==manager" \
  --network platypus-private \
  --mount="src=/var/run/docker.sock,dst=/var/run/docker.sock,type=bind"
    \
  dockersamples/visualizer
```

```
docker volume create platypus-db
docker volume create platypus-db-dev
```

```
docker network create --driver=overlay platypus-private
docker network create --driver=overlay platypus-public
```

```
docker service create \
  --name platypus-db-dev \
  --network platypus-private \
  --with-registry-auth \
  -p 27017 \
  --constraint 'node.hostname == docker-swarm-worker-04' \
  --mount "source=platypus-db-dev,target=/data/db,type=volume" \
  library/mongo:3.0.14
```

```
docker service create \
  --name platypus-api-dev \
  -p 3000 \
  -e "NODE_ENV=development" \
  --network platypus-private \
  --with-registry-auth \
  compiax/platypus-api:dev-latest
```

```
docker service create \
  --name platypus-ocr-dev \
  -p 5000:5000 \
  -e "FLASK_APP=server.py" \
  --network platypus-private \
  --with-registry-auth \
  compiax/platypus-ocr:dev-latest
```

```
docker run \
  -it \
  --rm \
  -e "FLASK_APP=server.py" \
  --name platypus-ocr-source \
  -p 3001:5000 \
  cos301-brute-force/platypus-ocr:source-latest
```

```
docker service create \
  --name students-proxy \
  -p 80:80 \
  -p 443:443 \
  --network odin-private \
  --network odin-public \
  --network platypus-private \
  --network platypus-public \
  --with-registry-auth \
  --replicas 2 \
  compiax/students-proxy
```

Updating

```
docker service update \
  --image compiax/platypus-api:dev-latest \
  --with-registry-auth \
  platypus-api-dev
```

```
docker service update \
  --image compiax/students-proxy:latest \
  --with-registry-auth \
  students-proxy
```

Useful Docker Service Commands

```
docker service update \
  --publish-rm 8080 \
  --with-registry-auth \
  swarm-visualizer-01
```

```
docker service update \
  --publish-rm 8080 \
  --with-registry-auth \
  swarm-visualizer-02
```

```
docker service update \
  --publish-add 8080 \
  --with-registry-auth \
  swarm-visualizer-01
```

```
docker service update \
  --publish-add 8080 \
  --with-registry-auth \
  swarm-visualizer-02
```

```
docker service update \
  --publish-rm 3000 \
  --with-registry-auth \
  platypus-api-dev
```

```
docker service update \
  --publish-add 3000 \
  --with-registry-auth \
  platypus-api-dev
```

```
docker service update \
  --publish-rm 8000 \
  --with-registry-auth \
  platypus-daemon-dev
```

```
docker service update \
  --publish-add 8000 \
  --with-registry-auth \
  platypus-daemon-dev
```

SSL

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
    ss.wildcard.split-bill.co.za.key -out
    ss.wildcard.split-bill.co.za.crt
```

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
    ss.wildcard.wildcard.split-bill.co.za.
```

1.3.2 Mobile Module

Initial requirements

- Please ensure instructions to set up the API module have been followed accurately to ensure that the API is set up correctly.
- Plug your *Android* phone into your computer and enable developer options in the settings as explained https://www.androidcentral.com/how-enable-developer-settings-android-42

Install

1. Pull the dev branch of this repo
2. Install Ionic globally

   ```
   npm install -g cordova ionic
   ```

3. Install the project dependencies

   ```
   npm install
   ```

Run

Run in browser

```
ionic serve
```

Run on mobile device setup for USB debugging

```
    ionic cordova run android
```

Optional: Get application and server output in your Google Chrome browser

```
    chrome://inspect
```

1.3.3 OCR Module

Initial requirements Please ensure instructions to set up the API module have been followed accurately to ensure that the API is set up correctly.

Software overview:

Software requirements include Python 2.7 and additional Python modules PIL and OS as well as OpenCV, Flask, Tesseract and its Python wrapper pytesseract. This page includes details for using our docker image for software installation.

Docker

A Docker image can be used to support all the software required to run the module. Docker software details can be found at https://www.docker.com/ under the Get Docker tab at the top of the page. There are installation details for various Operating Systems. Details of the docker image are as follows:

Build:

```
docker build -t cos301-brute-force/platypus-ocr:source-latest
```

Run OCR:

```
docker run \
  -it \
  --rm \
  -e "FLASK_APP=server.py" \
  --name platypus-ocr-source \
  -v $(pwd):/usr/src/app \
  -p 3001:5000 \
  cos301-brute-force/platypus-ocr:source-latest
```

1.4 Getting started

Ensure the latest version of the application is installed on your mobile phone and a server has been configured to handle requests from the SplitBill application. Ensure installation of all modules has been followed precisely according to the instructions provided in this document or on the various modules following wiki entries:

- https://github.com/COS301-brute-force/Platypus-API/wiki/Installation
- https://github.com/COS301-brute-force/Platypus-API/wiki/Docker-Swarm
- https://github.com/COS301-brute-force/Platypus-OCR/wiki/Installation
- https://github.com/COS301-brute-force/Platypus-Mobile/wiki/Installation

Upon start-up of the application on the mobile device, the user will be prompted for a user-name which is not persisted to a database, and asked to choose a color to keep track of claimed items.

The application's menus and pages and sufficiently self-explanatory when encountered for the first time. The application is designed to cater for all user groups who are familiar with an Android mobile device.

When creating a new bill your device's camera will open automatically. Make sure to capture the bill in good and even lighting to ensure the best results for the character recognition of the bill and it's items.

When you are happy with the quality of the picture, it will be sent to the server for processing with use of the OCR. This will produce a JSON string that is sent back to your device. Once the bill information is returned, your session is started and you will be presented with bill items.

You will be able to add additional users to the session. To do this, the other user should have the application installed on their device and have network access. The initial user can now use the menu item to navigate to the session code generated when the bill was sent and the additional users can now enter the code into the application on their own device. Once this has been done the system will retrieve the bill information and display it to the new user for further use on their own device.

In the bill information screen swiping right on an item will open a menu in which you can edit the selected item's price, description or quantity. You can swipe left on an item to open a menu in which you can claim single or multiple items. When items are claimed, the amount of the items is reduced by the amount that you have claimed. Once the item's amount has reached zero no user will be able to claim another instance of the item unless a user removes an instance from their own claimed items in the ”My items” menu.

At the bottom of the screen you will see a running total due by you which includes a gratuity. Claiming an item adds the price of the item to the total. You may alter the percentage gratuity, it has a default value of 10 percent of your current total. At the top right of a bill page is three horizontal lines which offers a menu when tapped. This contains menu items which should be self-explanatory and simplifies navigation. These menus have been designed to be familiar to the average Android user. If you claimed an item by mistake and wish to remove it you should navigate to ”My Items” in the menu or tap on ”My Items” in the tab at the top of the page. Once there you may swipe left on the mistakenly claimed item and tapping the circle with a horizontal line through it will then delete the item from your claimed items.

The final amount will be that of the prices of all the items you have claimed, and the amount of gratuity you wish to add, and thus the amount owed by you on the current bill. When the user is satisfied with the claimed items, gratuity, and final amount to pay, the user may leave the session. Leaving the session will clear the displayed bill. Once the last person has left the session the session will be terminated. When this occurs, there is no way for the user to retrieve previous bill information.

2 Using the system

Upon start-up of the application the user is prompted for a nickname and a color. These values are stored locally. The user data is only stored in the server when a user joins or creates a session.

Once the user creates a session, a session ID is generated on the server and a new, empty Bill object is created. The bill is added to the database with the session ID. The user is then added to the session. The user is then prompted to take a picture of the bill. This image is then uploaded to the API via a POST or GET request. The API sends the image to the OCR module via a POST or GET request for processing.

The OCR module, as its name indicates, is responsible for Optical Character recognition of the bill image received from the API. The OCR module first enhances the received image in order to improve the quality of the character recognition output by thresholding and reducing noise in the image as well as aligning the text using the OpenCV software. After character recognition has taken place using the Tesseract software, Levenshtein distance is used to calculate the difference between the word that the OCR recognized and words saved in a provided dictionary. When the distance is low enough, the word is replaced with the word found in the common dictionary. The information of the bill is then extracted according to the specified format. Irrelevant information such as items with empty descriptions, prices of zero or quantities of zero are removed from the and the remaining information is arranged in a JSON string before it is sent to the API. If the OCR is not able to recognize enough or any text the JSON string will contain error information that reflects this.

If there was sensible text recognized from the image, the object will look like this:

```
{ "type": "success", "attributes": { "data":[
    {"id":"1","desc":"description of item
    1","price":"xx.xx","quantity":"1"},
    {"id":"2","desc":"description of item
    2","price":"xx.xx","quantity":"5"},
    {"id":"3","desc":"description of item
    3","price":"xx.xx","quantity":"3"} ] }, "relationships": {
    "data":{total":"xx.xx"} } }
```

When the API receives the information form the OCR module, it sends the information in a POST or GET request back to the calling mobile application where the data is then displayed to the user along with the functionalities that accompany it.

When the user swipes left on an item, the user is given the option to claim one or multiple instances of the item. When an item or multiple items are claimed, the price corresponding to the items is added to the running total which can be seen at the bottom of the application screen. When an item is claimed, the amount of that item available to claim is reduced by the amount that has just been claimed. When an item has a quantity of 0, the item is removed from the bill view.

If the user swipes left on an item, they are presented with a menu in which they can alter the items price, description or quantity. This is useful for when the image was of inferior quality and character recognition was not 100 accurate.

The user also has the option to add items to the current bill. An additional item will consist of a description, price and quantity and will be displayed to all active users.

The "My Items" menu displays the items that the current user has already claimed. In this menu the user can swipe left on a selected item to increase or decrease the amount of the items claimed, or unclaim the item entirely. Changes in this menu can also be observed in the main bill information screen and the running total for the user.

At the bottom of the main screen the total price of the claimed items along with a gratuity option is displayed. The gratuity has a default value of 10 of your bill but can be modified to be less or more and is added to your running total accordingly.

The application has the functionality to add additional users to the same bill session. When the bill session is started the session ID and bill information is stored the database. An additional user can be added to the session by inputting the session ID generated into his/her own mobile device. When the session ID is sent to the API via a POST or GET request, the API retrieves the bill information of the corresponding session ID stored in the database. The bill information is then sent to the mobile application for display and interaction as with the initial user. The API will now be responsible for the live communication of item changes between each device. As an item is either created, edited or claimed, the device performing the action emits the action to the API, which in turn emits the update to all connected devices.

When the user is satisfied with their claimed items, gratuity and final total the user can leave the session. Once a user selects the option to leave a session, the API is updated and removes the user from the session in the database. Once all users have left, the session is terminated. There is no persistent data storage for a session that has been concluded

3 Troubleshooting

3.1 API Module

While the API is being run in development mode, there are debug messages that are output to the console. These messages are there to see what code is being called when, with details about the function call, data being passed, and generally what is currently happening.

This allows the developer to monitor how the system is executing and be able to see exactly what is going on. This proved valuable feedback to assist in debugging and ensuring correct execution of the system.

3.2 Mobile Module

3.2.0.1 Problem with the server?

Please see the API troubleshooting page

3.2.0.2 The application is not on your mobile device?

- Ensure you ran the commands correctly as shown on the installation page
- Check the application's directory on your mobile device since installing a new application doesn't necessarily make a shortcut on your home screen
- Ensure your mobile device has developer settings switched on
- Ensure your mobile device is plugged into the host issuing the installation commands

3.2.0.3 The application doesn't open?

- Ensure you're using an Android device
- Ensure your mobile device is version *Android KitKat (4.4)* or newer

3.2.0.4 The interface is unusable or broken?

Your solution may not be supported, please send a bug report to our team to fix this

3.2.0.5 Creating a bill doesn't work or gives a timeout error?

- Ensure you have SplitBill permission to use your camera when prompted upon launch. You may need to reinstall the application
- Ensure you have an Internet connection or have some means of communicating with the server such as a LAN connection
- Ensure the server didn't throw an error

3.3 OCR Module

3.3.0.1 Nothing is displayed when I upload an image?

The OCR will always return some text if the system is set-up correctly. When there is garbage text it will send back a JSON string with the following format:

```
{ "type":"failure", "errors":{ "id":"xx", "code":"GARBAGE",
    "title":"Garbage text recognised from the image" } }
```

If there is no text recognized, the JSON string will have the following format:

```
{ "type":"failure", "errors": { "id": "xxx", "code": "ERR1",
    "title": "semantic description" } }
```

Thus, no matter what the text, a JSON string will be sent back to the mobile application for display. Therefore if there is no text shown, check the following:

- Ensure you are connected to a network for communication with the server

- Ensure the server has been configured correctly and is running. See the installation details or API troubleshooting for the API module for this.

3.3.0.2 The text I am getting is not right?

If the information displayed to you is incorrect or absent, try taking a new photo that is closer up and has better lighting and alignment. The Tesseract recognition software is not perfect and thus will not yet be able to return exact character information each time.