Functional Requirements Document
QuickTrip React Admin Panel

Version	Description of Change	Author	Approver(s)	Date
1.0	Setup	BJZ		2020.09.10
1.1	Add Virtual Interlining Toggle	BJZ		2020.09.21
1.1.1	Updated design link	AC		2020.11.04

INTRODUCTION

Trip Ninja's QuickTrip application (also referred to as "the platform") is used by travel agents to carry out travel bookings for their customers. Access to the platform is licensed out to individual travel agencies who have one or more travel agent accounts. The platform allows for four (4) user types: Trip Ninja Administrators (highest access rights), Agency Administrators, Team Leads, and Travel Agents (lowest access rights) to conduct various activities according to their roles and permissions. Based on each travel agency's usage subscription plan and configuration, each of its travel agent's access is configured by either a Trip Ninja Administrator or an Agency Administrator.

Purpose

The original QuickTrip platform was separated into an AngularJS front-end application and a Django back-end application. The back-end application contained both business logic and "admin" (user authentication, settings, etc) logic.

Trip Ninja has migrated the front-end codebase of the QuickTrip platform from AngularJS to ReactJS. There currently is no admin panel on the new ReactJS front-end application, because the admin logic of the original QuickTrip application was originally all built and contained in the original singular Django back-end application.

The purpose of this project is to replace the admin panel of the QuickTrip application with two new pieces: (1) an extension of the new React front-end application [the "admin panel"], and (2) a separate Django back-end application to handle the administrative logic [the "auth server"].

In this React + Django-powered admin panel, users (Trip Ninja Administrators, Agency Administrators, Team Leads, and Travel Agents) will be able to modify their settings - as outlined in the User Structure section designated below.

Scope

This Functional Requirements Document (FRD) serves to outline functional requirements and design specifications for the development of the admin panel and auth server of the QuickTrip platform.

This document further intends to outline what aspects and features the finished product will include to accurately adhere to anticipated usage flow and data lifecycle for the platform.

References

- Adobe XD Link
- Trip Ninja's "Whitelabel" codebase for the previous iteration of QuickTrip
- Trip Ninja's current QuickTrip codebase
- Draft of models.py for Django (Appendix C)

FUNCTIONAL REQUIREMENTS

User Structure

There are two types of user management structures: "Teams" and “Roles”. The Roles are Trip Ninja Administrator, Agency Administrators, Team Leads, and Travel Agent. Each user must have a Role, however, not each user is required to have a Team. When there is no Team specifically assigned, a user will follow the global default Team setting, as defined below.

Teams:

A Team is a logical grouping of users. For example, users could be grouped into a particular retail location or business unit. Teams are each granted specific permissions and levels of access. Permissions are primarily associated with access to Trip Ninja features.

Roles:

A user’s Role determines what level of permissions they have within their organization. Users are broken out into four main categories: Trip Ninja Administrators, Agency Administrators, Team Leads, and Travel Agents (sometimes referred to only as Agents).

Trip Ninja Administrators have the highest level of access. They may change all settings for all accounts, except for changing a user's password. However, they may send a password reset email.

Within a travel agency, there must be at least one (but potentially several) person(s) in charge of managing the agency’s subscription and they are assigned an Agency Administrator account. Agency Administrators can add Travel Agents and Team Leads for their travel agency only. All users that are created by Agency Administrators or Team Leads inherit certain, specific fields (attributes) that tie them to their specific travel agency. Upon creating a user, Agency Administrators or Team Leads can either choose to copy over the Team/Agency attributes or set custom permissions for that Team or user.

User Functionality
All Users (Trip Ninja Administrators, Agency Administrators, Team Leads, and Travel Agents) ● May change their own password ● May modify enabled features/settings - according to permissions below

Trip Ninja Administrators (designated by the back-end attribute "Is Superuser" = True):

- May modify all settings, for themselves and for all other users, including:
  - API Username
  - API Password
  - Agency Name
  - Style Group
  - Iframe User
- May also modify settings common to Agency Administrators
- May send user password reset links

Agency Administrators (designated by the back-end attribute "Is agency admin" = True and "Is staff" = True):

- May add/remove/edit agents within their agency
- May add/remove/edit Teams
- May set the Team defaults for the following properties:
  - Default "Currency":
    - Sorted alphabetically
    - Options will be provided
  - "Date type":
    - Two options:
      - DD/MM/YYYY (UK)
      - MM/DD/YYYY (USA)
- May set the global defaults for the following properties:
  - Company name
  - Default "Currency":
    - Note - Currencies should be sorted alphabetically
    - Note - "Currency" options are listed here (link)
  - "Date type":
    - Two options:
      - DD/MM/YYYY (UK)
      - MM/DD/YYYY (USA)
- May modify the following fields for themselves, team leads, and agents in their agency:
  - Email
  - Password (can change their own password directly. Cannot change others' passwords directly, but can send a password reset email)
  - First Name
  - Last Name
  - Company Name
  - Disable Booking (Yes/No)
  - Role (Determined by back-end attributes: "Agency Admin" Status" and "Is Staff")

Team Leads (designated by back-end attributes "Is Team Lead" = True, TeamID = number. All agents assigned to the Team have the same TeamID):

- May modify the following fields for themselves, and agents in their agency:

- Email
- Password (can change their own password directly. Cannot change others' passwords directly, but can send a password reset email)
- First Name
- Last Name
- Phone Number

- May set their personal preferences for:
  - Default "Currency"
  - Default "Date Type"
- May set the **Team** defaults for the following properties for Teams which they are a part of:
  - Company name
  - Default "Currency"
  - Default "Date Type"
- May add and remove agents from their Team (from a pool of all agents in the Agency)
- May create/edit/archive agents in their Team:
  - When archived, the back-end attribute "is active" is set to False
  - Archived users still show up on the Users Overview table, however their Status is shown as "Deactivated"
- May view all Teams
- May manage agent permissions:
  - Enable and disable PNR creation (Field: "Disable Booking")
- May monitor all booking from agents on their Team, both in a summary format and individually

Travel Agents (no specific designations from back-end attributes):

- May access features based on their permissions (modifiable by Team Lead and Admin), including:
  - "Disable booking"
- May modify:
  - First Name
  - Last Name
  - Default "Currency"
  - Default "Date Type"
- May see bookings created by themselves

Hierarchy of Global Defaults, Team Defaults, and Individual Preferences

There are two types of settings: "Permissions" and "Preferences".

Preferences

Preferences can be set on a global level, a Team level, or an individual level. Individual preferences override Team defaults, and Team defaults override global defaults.

Preference settings include:

- "Date type"
- "Currency"

Permissions

Permissions follow a different structure. Permissions settings follow a Global > Team > Agent hierarchy. If a permission is disabled at a global level, it cannot be enabled at a Team or Agent level. If a permission is enabled at a global level, it can be disabled at a Team level. This is to ensure that Agency administrators have the most control.

Example:

- An agent could have "Disable booking" enabled (so they can't create bookings), the Team could have "Disable booking" disabled (so Team members can create bookings). However, if "Disable booking" is enabled for the team, a member of that team CANNOT have "Disable booking" disabled (allowing them to create bookings).

System Requirements

Admin Panel

The development of the QuickTrip admin panel shall adhere to the following system development requirements:

Design and Development Constraints
1. This project shall be built to be mainly used on a web page set up 2. This project shall be built to be responsive and usable on any viewport by employing the Bootstrap CSS library 3. This project shall be built using Typescript, React, and Redux 4. This project shall use all new authentication operations accessed via the API endpoints from the new Django authentication back-end

5. All components used in this project shall be built in the src/admin/ directory of the QuickTrip codebase

The development of the QuickTrip admin panel shall adhere to the following Visuals and UI Design outline:

Visuals and UI Design

The visual elements of the UI design are covered in the Adobe XD link in the References section

1. This project shall be built per the approved user interface design
    a. Design files will govern the application front-end styling and layout
2. Within the approved user interface design, there are approved dimensions and states of many of the components used throughout the project. The components throughout the design may differ slightly - particularly in regards to margins and padding - from the dimensions outlined on the components page. When there is a discrepancy, the dimensions and spacing on the components page should be used. The design includes 34 pages. See design here.
3. The project will use elements from Material UI whenever possible if not otherwise indicated in the design
4. Clicking outside of any modals will close the modal
5. Elements which are not in the components section have hover elements indicated throughout the design
6. Wording for tooltips is shown by hovering over the tooltip

The development of the QuickTrip admin panel shall adhere to the following Data Modeling notes:

Data Modeling

The admin panel will not require data modelling and construction. However, understanding of the data models used in the auth server will assist in proper rendering of associated objects in the admin panel. Refer to the data models in the auth server for guidance.

The development of the QuickTrip admin panel shall adhere to the following Code Standards:

Code Standards

1. 2-space indents
2. A semi-colon will be used at the end of lines

3. Variables and functions are to be named in "thisCaseFormat" and classes are to be named in "AreAllCapWords"
4. HTML and CSS class and ID names are to be in "this-case-format"
5. Usage of ES6 Javascript conventions is required

The guidelines in Appendix B for building Restful APIs should be followed.

Auth Server

The development of the QuickTrip auth server shall adhere to the following system development requirements:

Design and Development Constraints - Auth Server

1. This project shall be built using Python 3.7+ and Django 3+
2. All functionality shall be made available to the front-end admin panel via API endpoints
    a. This project shall use Django REST Framework to construct all API endpoints
    b. All API endpoints shall be designed to be token authenticated

The development of the QuickTrip auth server shall adhere to the following Visuals and UI Design outline:

Visuals and UI Design
Visual elements are not a consideration in the development of the auth server.

The development of the QuickTrip auth server shall adhere to the following Data Modelling notes:

Data Modeling
This project shall develop appropriate data models to accurately store information pertaining to the functionality required for the admin panel and auth server. Refer to the example data models in Appendix C for guidance.

The development of the QuickTrip admin panel shall adhere to the following Code Standards:

Code Standards
1. 4-space indents

2. Pep8 for all python code

The guidelines in Appendix B for building Restful APIs should be followed.

Deployment Infrastructure

*Not part of the scope of this project.

Architectural and Functional Requirements

Admin Panel Addition
Description The admin panel is responsible for providing administrative users with typical user management functionality, such as creating, updating, and deactivating new users as well as setting and updating user settings and permissions.
Inputs Expected inputs are: 1. Your Preferences: a. Basic settings: i. Phone number ii. Default currency iii. Email address iv. Default calendar layout selection 2. Company Defaults: a. General Information: i. Company name ii. Default currency iii. Default calendar layout selection b. Content Sources i. No inputs - static page c. Search/Booking Details: i. “Agents can create PNRs?” selection ii. “Virtual Interlining” selection d. Billing and Account Management i. No inputs - static page 3. Settings: a. Users: i. Users or teams search text ii. Add user button selection iii. Bulk add button selection iv. Pagination selection v. Actions selection per user b. Archive User modal: i. Archive User button selection

ii. Cancel button selection

c. Update User modal:
   i. Email address
   ii. Phone number
   iii. Name
   iv. Personal Info tab selection
   v. Booking tab selection
   vi. "Agents can create PNRs?" selection
d. Add User modal:
   i. Create User section:
      I. First name
      II. Last name
      III. Email address
      IV. Next button selection
   ii. Set Permissions section:
      I. Team assignment
      II. Inherit team default permissions selection
      III. "Agents can create PNRs?" selection
      IV. Next button selection
   iii. Send Invite section:
      I. Send Invites button selection
e. Bulk Add Users modal:
   i. Create User section:
      I. Email address
      II. Next button selection
   ii. Set Team section:
      I. Teams assignment
      II. Next button selection
   iii. Set Permissions section:
      I. Team selection
      II. Inherit team default permissions selection
      III. "Agents can create PNRs?" selection
      IV. Next button selection
   iv. Send Invite section:
      I. Send Invite button selection
f. Teams:
   i. Users or Teams search text
   ii. Add Team button selection
   iii. Pagination selection
   iv. Actions selection per Team
g. Archive Team modal:
   i. Archive Team button selection
   ii. Cancel button selection
h. Update Team modal:
   i. Team members
   ii. Team Lead(s)
   iii. "Agent can create PNRs?" selection
i. Add Team modal:
   i. Create Team section:
      I. Team name
      II. Next button selection
   ii. Set Permissions section:
      I. Inherit team default permissions selection

II. "Agents can create PNRs?" selection
III. Next button selection
iii. Add Members section:
I. Team members
II. Team Lead(s)
III. Next button selection
iv. Create Team section:
I. Create your Team button selection
j. Reset Password modal:
i. Current password
ii. New password
iii. Cancel button selection
iv. Reset button selection

Outputs

Expected outputs are:

1. Your Preferences:
    a. Basic Information:
        i. Successful updates to basic information fields will be automatically saved and reflected in subsequent viewings of basic information
2. Company Defaults:
    a. General Information:
        i. Successful updates to general information fields will be automatically saved and reflected in subsequent viewings of general information
    b. Content Sources
        i. No outputs - static page
    c. Search & Booking Details:
        i. Successful updates to search and booking defaults fields will be automatically saved and reflected in subsequent viewings of search and booking details
    d. Billing and Account Management
        i. No outputs - static page
3. Settings:
    a. Users:
        i. Render list of users
    b. Create Users modal:
        i. Successful user creation reflects a new user object to be rendered in the previously mentioned list of users; the new user object has all of the attributes entered in the user creation inputs section
        ii. Successful user creation ends with an invite email being sent to the new user's previously entered email address
    c. Bulk Create Users modal:
        i. Successful user creation reflects new user objects to be rendered in the previously mentioned list of users; new user objects have all of the attributes entered in the user creation inputs section
        ii. Successful user creation ends with invite emails being sent to the respective new users' previously entered email addresses
    d. Update Users modal:
        i. Successful user update(s) will be reflected by the attributes in the respective user object(s)
    e. Archive User modal:
        i. Successful archival of user(s) will be reflected by the respective

user(s) status

f. Teams:
    i. Render list of Teams
g. Create Teams modal:
    i. Successful Team creation reflects a new team object to be rendered in the previously mentioned list of Teams; new Team objects have all of the attributes entered in the Team creation inputs section
    ii. Successful Team creation ends with invite emails being sent to the new Team's respective new users' previously entered email addresses
h. Update Teams modal:
    i. Successful Team update(s) will be reflected by the attributes in the respective Team object(s)
i. Reset Password modal:
    i. Successful password reset will be reflected by an updated password for the user whose password is being changed

Functionality

Refer to XD design for functionality prompts.

Authentication Server

Description

The auth server is responsible for storing the data and logic required to facilitate user authentication functionality and setting/updating user settings and permissions.

Inputs

Expected inputs are:

1. Your Preferences:
    a. Basic settings:
        i. Phone number
        ii. Default currency
        iii. Email address
        iv. Default calendar layout selection
2. Company Defaults:
    a. General Information:
        i. Company name
        ii. Default currency
        iii. Default calendar layout selection
    b. (Future) Content Sources
    c. Search/Booking Details:
        i. "Agents can create PNRs?" selection
        ii. "Virtual Interlining" selection
    d. (Future) Billing and Account Management
3. Settings:
    a. Users:
        i. No expected inputs
    b. Archive User:

i. Archive User selection

c. Update User:
   i. Email address
   ii. Phone number
   iii. Name
   iv. "Agents can create PNRs?" selection
d. Add User:
   i. Create User:
      1. First name
      2. Last name
      3. Email address
   ii. Set Permissions section:
      1. Team assignment selection
      2. Inherit team default permissions selection
      3. "Agents can create PNRs?" selection
e. Bulk Add Users modal:
   i. Create Users:
      1. Email address(es)
   ii. Set Team section:
      1. Teams assignment selection
   iii. Set Permissions section:
      1. Team selection
      2. Inherit team default permissions selection
      3. "Agents can create PNRs?" selection
f. Teams:
   i. No expected inputs
g. Archive Team:
   i. Archive Team button selection
h. Update Team:
   i. Team member(s)
   ii. Team Lead(s)
   iii. "Agent can create PNRs?" selection
i. Add Team:
   i. Create Team:
      1. Team name
   ii. Set Permissions section:
      1. Inherit team default permissions selection
      2. "Agents can create PNRs?" selection
   iii. Add Members:
      1. Team members
      2. Team Lead(s)
j. Reset Password:
   i. Current password
   ii. New password

Outputs

Expected outputs are:

1. Your Preferences:
   a. Basic Information:
      i. Save new/updated information
      ii. No further outputs
2. Company Defaults:

a. General Information:
    i. Save new/updated information
    ii. No further outputs
b. (Future) Content Sources
c. Search & Booking Details:
    i. Save new information
    ii. No further outputs
d. (Future) Billing and Account Management

3. Settings:
    a. Users:
        i. Return list of users
    b. Create Users:
        i. Save new user with submitted attributes
        ii. Send invite email to new user's submitted email address
    c. Bulk Create Users:
        i. Save new user with submitted attributes
        ii. Send invite email to new users' submitted email addresses
    d. Update Users:
        i. Save update(s) to respective attributes of respective, updated user object(s)
    e. Archive User:
        i. Save archived status to selected user object(s)
    f. Teams:
        i. Return list of Teams
    g. Create Teams:
        i. Save new Team with submitted attributes
        ii. Send invite email(s) being sent to new users submitted email addresses
    h. Update Teams:
        i. Save update(s) to respective attributes of respective, updated Team object(s)
    i. Reset Password:
        i. Save new/updated information
        ii. Return success status code

Functionality Requirements

Refer to XD design for functionality prompts.

Appendix A - Glossary

- Travel Agency = A company that employs travel agents to sell flights.
- Travel Agent = A person that sells flights to leisure or corporate travellers and is employed by a travel agency

- PNR = Passenger Name Record. A reservation on an airline system which describes flight times, passenger information, payment information. A booking made on Quicktrip may utilize multiple PNRs at one time.

Appendix B: Guideline for building Restful APIs

- HTTP methods should be used properly For example, POST: /user/ may mean "Create a new user".
- At a high-level, verbs map to CRUD operations: GET means Read, POST means Create, PUT and PATCH mean Update, and DELETE means Delete.
- A response's status is specified by its status code: 1xx for information, 2xx for success, 3xx for redirection, 4xx for client errors and 5xx for server errors.
- Use JSON format for the body of objects
- Use query string for filtering and pagination
- Use lower case names separated with '_' for body parameters.

Appendix C: Draft models.py

```
# Relations
#      Hierarchy Agency > Team > User
#      Agency - one-to-many -> Team - one-to-many -> User

class Datasource():
       name = models.CharField(max_length=40, blank=True)
       pcc = models.CharField(max_length=15, default="2G3C")
       provider = models.CharField(max_length=2, choices=PROVIDER_CHOICES,
default="1V")
       active = models.BooleanField(default=True)
       agency: models.ForeignKey(Agency)
       queue = models.CharField(max_length=40, default="01")
       PROVIDER_CHOICES = [('1A'), ('1V'), ('1G'), ('1P')]


class Agency():
       name = models.CharField(max_length=100)
       amadeus_branded_fares = models.BooleanField(default=False)
       api_username = models.CharField(max_length=40, default="trialaccount")
    api_password = models.CharField(max_length=40, default="p#F91Snf#Pr3Yr")
    travelport_itx = models.BooleanField(default=False)
    student_and_youth = models.BooleanField(default=False)
    common_parameters = models.OneToOneField(CommonParameters)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

```
class Team():
       name = models.CharField(max_length=100)
       agency = models.ForeignKey(Agency)
       common_parameters = models.OneToOneField(CommonParameters)
       created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)


class User(AbstractBaseUser, PermissionsMixin):
    email = models.EmailField(unique=True)
    first_name = models.CharField(max_length=40)
    last_name = models.CharField(max_length=40)
    agency = models.ForeignKey(Agency)
    team = models.ForeignKey(Team)
    common_parameters = models.OneToOneField(CommonParameters)
    is_active = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    search_endpoint = models.CharField(max_length=8, default="prod")
    booking_endpoint = models.CharField(max_length=8, default="prod")
    ENDPOINT_CHOICES = [('prod'), ('preprod')]


class CommonParameters():
       currency = models.CharField(max_length=3)
    date_type = models.CharField(max_length=10)
    booking_enabled = models.BooleanField(default=True)
    exclude_carriers = JSONField(max_length=255, default=[], blank=True)
    DATE_CHOICES = [('USA'), ('UK')]
```