Icon

Description automatically generated

# **BED CA2 Documentation**

**Name:** Lee Quan Jun Ervin

**Admin Number:** 2104005

**Class:** DISM/FT/2B/21

Table of Contents

[**BED CA2 Documentation** 1](#_Toc110293157)

[Notes: 3](#_Toc110293158)

[Installation Guide 3](#_Toc110293159)

[Project Directory 5](#_Toc110293160)

[Pages Created 6](#_Toc110293161)

[Fulfillments 7](#_Toc110293162)

[Basic Requirement Fulfillment 7](#_Toc110293163)

[Bonus Requirement Fulfillment 7](#_Toc110293164)

[Additional/Advanced Features of Full-Stack Website 7](#_Toc110293165)

[Modifications from CA1 9](#_Toc110293166)

[Cart Model 9](#_Toc110293167)

[Authorization and Authentication 9](#_Toc110293168)

[Database Modification 9](#_Toc110293169)

[Removed Endpoints from CA1 9](#_Toc110293170)

[New/Modified Endpoints 10](#_Toc110293171)

[Additional Scripts 22](#_Toc110293172)

[SQL Database Configuration 24](#_Toc110293173)

[Foreign Keys 24](#_Toc110293174)

[Tables 24](#_Toc110293175)

[EER Diagram of Table Relationships 26](#_Toc110293176)

# Notes:

All information about the code can be found in the README.md file in the zipped folder

# Installation Guide

**Opening the file:**

1. Download the submission zipped folder from Brightspace
2. Once downloaded, extract the zip folder.
3. Open the extracted folder on Visual Studio Code
4. All the dependencies should by default be installed in the package.json file
   1. In the event if they do not work, refer to the README to view the NPM packages required and npm install them accordingly

**Running the web application (First Method):**

1. Right click the respective client and server folders and click on “Open in Integrated Terminal” respectively. Two terminals should open
2. Type “nodemon” on the two terminals to run the app scripts. If nodemon does not work, run the commands “node index.js” on the client terminal and “node server.js” on the server terminal.
3. Open your browser and navigate to <http://localhost:3000> to view my amazing flight booking website!

**Running the web application (Second Method):**

1. Create two terminals on your Visual Studio Code workstation
2. Run the command “cd client” on one terminal and “cd server” on the other
3. Run nodemon on both terminals respectively
4. If nodemon does not work, run the commands “node index.js” on the client terminal and “node server.js” on the server terminal

**Setting up the SQL database:**

1. On the databaseConfig.js script found on the server directory under the model folder
2. Change the database password to your own
3. Navigate to the db\_init2.sql script found in the server directory
4. Open the SQL script on your MySQL Workbench
5. In the unlikely event the previous step is not possible, open the script on any text editor and copy and paste the script contents onto a blank SQL file in your Workbench
6. Run the script by clicking on the lightning icon

# Project Directory

Text

Description automatically generated with medium confidence

# Pages Created

1. 404.html – Error display page when 404 Not Found triggered
2. Admin.html – Admin panel page for adding and deleting airports, flights and promotions and viewing booking history of all users
3. Booking.html – Checkout page for users to book their flights added to their carts
4. Bookinghistory.html –Page for users to check their past flight bookings
5. Cart.html – Page for users to view their flights they added to their cart
6. Flightdetails.html – Page to show more flight and airport information for each flight user selects to view
7. Flights.html – Page to display all flights for users to browse
8. Index.html – Main homepage for users
9. Login.html – Login page
10. Profile.html – Page for users to view and edit their profiles
11. Promotions.html – Page for users to browse and search promotions for flights
12. Searchflights.html – Page for users to search specific flights based on origin, destination, departure and return dates
13. Signup.html – Page for guest users to sign up and create a new account in SP AIR
14. Transfers.html – Page for users to find connections and transfers from their desired origin and destination airports

# Fulfillments

## Basic Requirement Fulfillment

1. Login System
   1. Use of JSON Web Tokens (JWT) to secure pages that require accounts using local storage to store the JWT token and logged in user ID
   2. Persistent login
2. Search Form to search flights by origin airport, destination airport, departure and return date (searchflights.html)
3. View full flight details in a separate page (flightdetails.html)
4. View Profile Information (profile.html)
5. Add new flights (Admin Panel on admin.html)
6. Add new airports (Admin Panel on admin.html)

## Bonus Requirement Fulfillment

1. Uploading of actual image of flight aircraft seats when creating new flights (Use of Multer middleware and Express static server to serve images)
2. Add and search promotions for flights (promotion.html)
3. Shopping cart functionality to book flights and confirm payment details and check out with data saved to database (cart.html and booking.html)
4. Search option for transfer airport for non-direct flights (transfers.html)

## Additional/Advanced Features of Full-Stack Website

1. Updating of profile information with image preview of new profile picture
2. Admin Panel on admin.html displays user information (without passwords due to privacy reasons)
3. Filter flight search results via minimum and maximum price ranges on searchflights.html
4. Added adding and deleting of promotions in the admin panel (admin.html)
5. Page that displays all flights offered by SP Air with sorting and filter options like flight code queries, sorting by price or alphabetical order in either ascending or descending (flights.html)
6. Sign up option for new guest users to create a new account on the signup.html page (redirected from the login.html page if they do not have an account)
7. Option for users to see all possible flight connections for non-direct flights based on only origin and destination airport inputs (if transfer airport input is not specified)
8. Admins have the option to add and delete airports, flights, and promotions on admin.html
9. Error page will be displayed if 404 Not Found event is triggered (routing is done on the index.js client script)
10. Users can view their cart on cart.html and have the option to clear their cart or remove a cart item
11. Users can choose quantity of tickets and the seat class (1.5x for Business class seats and 2x for First class seats) on flightdetails.html before adding flight to the cart
12. Users can view their booking history on the bookinghistory.html
13. Administrators can view booking history of all users logged in to the website and have the option to clear it (admin.html)

# Modifications from CA1

## Cart Model

Added a new model to meet functionalities of the shopping cart system of the full-stack website.

## Authorization and Authentication

Added the “auth” folder to store the verifyToken.js script, responsible for the JWT token verification middleware

Added a new config.js file in the server directory to store the secret JWT key which will be exported to the backend server (app.js file)

## Database Modification

Updated database to add new columns for the flight table and (see MySQL databases)

Added two new tables: cart and booking to handle cart and booking information from the full-stack website

## Removed Endpoints from CA1

Removed and revamped advanced endpoints from CA1 to create new advanced features in the full-stack website.

# New/Modified Endpoints

Endpoint 8 (MODIFIED)

Text

Description automatically generated

Endpoint 16 – Endpoint to handle user login

Text

Description automatically generated

Endpoint 17 – Endpoint to get ALL flights

Text

Description automatically generated

Endpoint 18 – Endpoint to deleted select airport by airport ID

Text

Description automatically generated

Endpoint 19 – Endpoint to retrieve return flights from the database

Text

Description automatically generated

Endpoint 20 – Endpoint to get flight by flight ID

Text

Description automatically generated

Endpoint 21 – Endpoint to get airport information by airport ID

Text

Description automatically generated

Endpoint 22 – Endpoint to get promotions by flight ID

Text

Description automatically generated

Endpoint 23 – Endpoint to add a new flight to a cart

Text

Description automatically generated

Endpoint 24 – Endpoint to get cart information from the cart

Text

Description automatically generated

Endpoint 25 – Endpoint to delete cart item by flight ID from the cart

Text

Description automatically generated

Endpoint 26 – Endpoint to clear the cart

Text

Description automatically generated

Endpoint 27 – Endpoint to get cart item by cart ID

Text

Description automatically generated

Endpoint 28 – Endpoint to get all flights (sorted based on method requested)

Text

Description automatically generated

Endpoint 29 – Endpoint to get flight by origin, transfer, and destination airport ID

Text

Description automatically generated

Endpoint 30 – Endpoint to search promotion by search query

Text

Description automatically generated

Endpoint 31 – Endpoint to get booking history by user ID

Text

Description automatically generated

Endpoint 32 – Endpoint to get booking data from booking database

Text

Description automatically generated

Endpoint 33 – Endpoint to clear entire booking history

Text

Description automatically generated

Endpoint 34 – Endpoint to clear booking history containing user ID

Text

Description automatically generated

Endpoint 35 – Endpoint to select all promotions where today’s date within promotional periods

Text

Description automatically generated

## Additional Scripts

VerifyToken.js Script

Text

Description automatically generated

Client/Index.js Script

* Express Front-End server to serve HTML pages and handle API routing

Text

Description automatically generated

# SQL Database Configuration

## Foreign Keys

* airport.airportid = flight.originAirport = flight.destinationAirport
* flight.flightid = booking.flightid
* user.userid = booking.userid = cart.userid
* user.username = booking.name
* flight.flightid = promotion.flightid = cart.flightid

## Tables

**Airport**

Graphical user interface, text, application

Description automatically generated

**Booking**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Cart**

**Graphical user interface, application

Description automatically generated**

**Flight**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Promotion**

**A screenshot of a computer

Description automatically generated with medium confidence**

**User**

**A screenshot of a computer

Description automatically generated with medium confidence**

## EER Diagram of Table Relationships

Table

Description automatically generated with medium confidence

[END OF BED CA2 DOCUMENTATION]