

```
1  from cs50 import SQL
2  from flask import redirect, render_template, url_for, request
3
4  db = SQL("sqlite:///tripplanner.db")
5
6
7  # For errors found throughout the doc regarding numbers (cash).
8  def error(amount):
9      # Return error with custom message if negative number
10     if amount < 0:
11         positive = "Please enter a valid amount value. Only positive values."
12         return render_template("error.html", errortype=positive)
13     # Return error with custom message if zero
14     elif amount == 0:
15         zero = "Please enter some positive value."
16         return render_template("error.html", errortype=zero)
17     # Return error with custom message if expense is greater than current cash
18     else:
19         notenough = "Not enough money! Please enter more on allowance."
20         return render_template("error.html", errortype=notenough)
```

```
1  # This project is inspired by "C$50" Finance from CS50.
2  # https://docs.cs50.net/2017/ap/problems/finance/finance.html
3
4  from cs50 import SQL
5  from flask import Flask, flash, redirect, render_template, request, url_for
6  from helpers import error
7
8  app = Flask(__name__)
9
10 # Configure CS50 Library to use SQLite database
11 db = SQL("sqlite:///tripplanner.db")
12
13
14 @app.route("/")
15 def homepage():
16     # Import data from SQL Tables to display on homepage
17     flightinfo = db.execute("SELECT Number, Location, Date, Time FROM \
18         flight WHERE 1 ORDER BY date ASC")
19     traininfo = db.execute("SELECT Number, Location, Date, Time FROM \
20         train WHERE 1 ORDER BY date ASC")
21     hotelinfo = db.execute("SELECT Name, Date, Time, Outdate FROM \
22         hotel WHERE 1 ORDER BY date ASC")
23
24     # Select cash from SQL
25     cashs = db.execute("SELECT cash FROM cash WHERE 1")
26     for cash in cashs:
27         cash = cash['cash']
28
29     # Display homepage with all information
30     return render_template("homepage.html", flightinfo=flightinfo, traininfo=traininfo,
31         hotelinfo=hotelinfo, cash=cash)
32
33
34 @app.route("/flight", methods=["GET", "POST"])
35 def flight():
36     # Reached page through GET (Clicking a link or redirect)
37     if request.method == "GET":
38         return render_template("flight.html")
39     # Else if page is reached route through POST (submitting a form)
40     else:
41         # Get flight data from form
42         number = (request.form.get("number"))
43         location = (request.form.get("location"))
44         date = (request.form.get("date"))
45         time = (request.form.get("time"))
```

```
46         # Save flight data to SQL
47         db.execute("INSERT INTO flight (number, date, location, time) \
48             VALUES (:number, :date, :location, :time);", number=number,
49             location=location, date=date, time=time)
50
51         # Redirect to homepage
52         return redirect(url_for("homepage"))
53
54
55 @app.route("/train", methods=["GET", "POST"])
56 def train():
57     # Reached page through GET (Clicking a link or redirect)
58     if request.method == "GET":
59         return render_template("trains.html")
60     # Else if page is reached route through POST (submitting a form)
61     else:
62         # Get train data from form
63         number = (request.form.get("number"))
64         location = (request.form.get("location"))
65         date = (request.form.get("date"))
66         time = (request.form.get("time"))
67
68         # Save train data to SQL
69         db.execute("INSERT INTO train (number, date, location, time) \
70             VALUES (:number, :date, :location, :time);", number=number,
71             location=location, date=date, time=time)
72
73         # Redirect to homepage
74         return redirect(url_for("homepage"))
75
76
77 @app.route("/hotel", methods=["GET", "POST"])
78 def hotel():
79     # Reached page through GET (Clicking a link or redirect)
80     if request.method == "GET":
81         return render_template("hotels.html")
82     # Else if page is reached route through POST (submitting a form)
83     else:
84         # Get hotel data from form
85         name = (request.form.get("name"))
86         date = (request.form.get("date"))
87         time = (request.form.get("time"))
88         outdate = (request.form.get("outdate"))
89         # Save hotel data to SQL
90         db.execute("INSERT INTO hotel (name, date, time, outdate) \
```

```

91         VALUES (:name, :date, :time, :outdate);", name=name,
92             date=date, time=time, outdate=outdate)
93
94     # Redirect to homepage
95     return redirect(url_for("homepage"))
96
97
98 @app.route("/expenses", methods=["GET", "POST"])
99 def expenses():
100     # Reached page through GET (Clicking a link or redirect)
101     if request.method == "GET":
102         return render_template("expenses.html")
103     # Else if page is reached route through POST (submitting a form)
104     else:
105         # Get information from form
106         Amount = int(request.form.get("amount"))
107         Expense = request.form.get("expense")
108         Date = request.form.get("date")
109         # Return error if negative number
110         if Amount <= 0:
111             return error(Amount)
112         else:
113             # Select cash from SQL
114             cashs = db.execute("SELECT cash FROM cash WHERE 1")
115             for cash in cashs:
116                 cash = cash['cash']
117             # Return error if amount is greater than current cash
118             if cash < Amount:
119                 return error(Amount)
120             elif cash >= Amount:
121                 # Add transaction to expense history
122                 db.execute("INSERT INTO expenses (Amount, Expense, Date) VALUES \
123                     (:Amount, :Expense, :Date)", Expense=Expense, Amount=Amount, Date=Date)
124                 # Update cash, subtracting cash from transaction
125                 db.execute("UPDATE cash SET Cash = Cash - :Amount", Amount=Amount)
126             # Redirect to homepage
127             return redirect(url_for("homepage"))
128
129
130 @app.route("/allowance", methods=["GET", "POST"])
131 def allowance():
132     # Reached page through GET (Clicking a link or redirect)
133     if request.method == "GET":
134         return render_template("allowance.html")
135     # Else if page is reached route through POST (submitting a form)

```

```
136     else:
137         # Get allowance from form
138         allowance = int(request.form.get("allowance"))
139         # Return error if negative number
140         if allowance <= 0:
141             return error(allowance)
142         # Add allowance to current cash
143         db.execute("UPDATE cash SET Cash = Cash + :allowance", allowance=allowance)
144     # Redirect to homepage
145     return redirect(url_for("homepage"))
146
147
148 @app.route("/expensehistory", methods=["GET", "POST"])
149 def expensehistory():
150     # If page is reached route through POST (submitting a form)
151     if request.method == "POST":
152         return render_template("expensehistory.html")
153     # Else if page is reached through GET (Clicking a link or redirect)
154     else:
155         # Import expenses from SQL
156         expenseinfo = db.execute("SELECT Amount, Expense, Date FROM \
157             expenses ORDER BY date ASC")
158         # Select all expenses from SQL
159         for expense in expenseinfo:
160             amount = expense['Amount']
161             expense = expense['Expense']
162
163         # Display history with information
164         return render_template("expensehistory.html", expenseinfo=expenseinfo)
165
166
167 @app.route("/delete", methods=["GET", "POST"])
168 def delete():
169     # Reached page through GET (Clicking a link or redirect)
170     if request.method == "GET":
171         # Import tables from SQL
172         train = db.execute("SELECT Number FROM train")
173         flights = db.execute("SELECT Number FROM flight")
174         hotel = db.execute("SELECT Name FROM hotel")
175         expense = db.execute("SELECT expense FROM expenses")
176         # Display page with information
177         return render_template("delete.html", flights=flights, train=train, hotel=hotel, expense=expense)
178     # Else if page is reached route through POST (submitting a form)
179     else:
180         # Get information from form
```

```
181 train = request.form.get("train")
182 flight = request.form.get("flight")
183 hotel = request.form.get("hotel")
184 expense = request.form.get("expense")
185
186 # Remove information of corresponding table
187 if train:
188     db.execute("DELETE FROM train WHERE Number=:train", train=train)
189 elif flight:
190     db.execute("DELETE FROM flight WHERE Number=:flight", flight=flight)
191 elif hotel:
192     db.execute("DELETE FROM hotel WHERE Name=:hotel", hotel=hotel)
193 elif expense:
194     amount = (db.execute("SELECT Amount FROM expenses WHERE expense=:expense", expense=expense))
195     # Update cash (refund) if expense is deleted
196     for amount in amount:
197         amount = int(amount['Amount'])
198     db.execute("DELETE FROM expenses WHERE expense=:expense", expense=expense)
199     db.execute("UPDATE cash SET Cash = Cash + :amount", amount=amount)
200
201 # Redirect to homepage
202 return redirect(url_for("homepage"))
```