README.md

First Choice Travel Hub Repository

Procfile

```
web: gunicorn --worker-class eventlet -w 1 app:app
```

Pipfile.lock


```
{
    "_meta": {
        "hash": {
            "sha256": "d719de6c3b37368de4a53363bbb8e7c79fafadfa3b06d5b8704f802d0542bb43"
        },
        "pipfile-spec": 6,
        "requires": {
            "python_version": "3.6"
        },
        "sources": [
            {
                "name": "pypi",
                "url": "https://pypi.org/simple",
                "verify_ssl": true
            }
        ]
    },
    "default": {
        "alembic": {
            "hashes": [
                "sha256:1cd32df9a3b8c1749082ef60ffbe05ff16617b6afadfdabc680dcb9344af33d7"
            ],
```

```
    "version": "==0.9.10"
  },
  "bcrypt": {
    "hashes": [
      "sha256:01477981abf74e306e8ee31629a940a5e9138de000c6b0898f7f850461c4a0a5",
      "sha256:054d6e0acaea429e6da3613fcd12d05ee29a531794d96f6ab959f29a39f33391",
      "sha256:0872eeecdf9a429c1420158500eedb323a132bc5bf3339475151c52414729e70",
      "sha256:09a3b8c258b815eadb611bad04ca15ec77d86aa9ce56070e1af0d5932f17642a",
      "sha256:0f317e4ffbdd15c3c0f8ab5fbd86aa9aabc7bea18b5cc5951b456fe39e9f738c",
      "sha256:2788c32673a2ad0062bea850ab73cffc0dba874db10d7a3682b6f2f280553f20",
      "sha256:321d4d48be25b8d77594d8324c0585c80ae91ac214f62db9098734e5e7fb280f",
      "sha256:346d6f84ff0b493dbc90c6b77136df83e81f903f0b95525ee80e5e6d5e4eef84",
      "sha256:34dd60b90b0f6de94a89e71fcd19913a30e83091c8468d0923a93a0cccbfbbff",
      "sha256:3b4c23300c4eded8895442c003ae9b14328ae69309ac5867e7530de8bdd7875d",
      "sha256:43d1960e7db14042319c46925892d5fa99b08ff21d57482e6f5328a1aca03588",
      "sha256:49e96267cd9be55a349fd74f9852eb9ae2c427cd7f6455d0f1765d7332292832",
      "sha256:67ed1a374c9155ec0840214ce804616de49c3df9c5bc66740687c1c9b1cd9e8d",
      "sha256:6efd9ca20aefbaf2e7e6817a2c6ed4a50ff6900fafdea1bcb1d0e9471743b144",
      "sha256:8569844a5d8e1fdde4d7712a05ab2e6061343ac34af6e7e3d7935b2bd1907bfd",
      "sha256:8629ea6a8a59f865add1d6a87464c3c676e60101b8d16ef404d0a031424a8491",
      "sha256:988cac675e25133d01a78f2286189c1f01974470817a33eaf4cfee573cfb72a5",
      "sha256:9a6fedda73aba1568962f7543a1f586051c54febbc74e87769bad6a4b8587c39",
      "sha256:9eced8962ce3b7124fe20fd358cf8c7470706437fa064b9874f849ad4c5866fc",
      "sha256:a005ed6163490988711ff732386b08effcbf8df62ae93dd1e5bda0714fad8afb",
      "sha256:ae35dbcb6b011af6c840893b32399252d81ff57d52c13e12422e16b5fea1d0fb",
      "sha256:b1e8491c6740f21b37cca77bc64677696a3fb9f32360794d57fa8477b7329eda",
      "sha256:c906bdb482162e9ef48eea9f8c0d967acceb5c84f2d25574c7d2a58d04861df1",
      "sha256:cb18ffdc861dbb244f14be32c47ab69604d0aca415bee53485fcea4f8e93d5ef",
      "sha256:d86da365dda59010ba0d1ac45aa78390f56bf7f992e65f70b3b081d5e5257b09",
```

```
        "sha256:e22f0997622e1ceec834fd25947dc2ee2962c2133ea693d61805bc867abaf7ea",
        "sha256:f2fe545d27a619a552396533cddf70d83cecd880a611cdfdbb87ca6aec52f66b",
        "sha256:f7fd3ed3745fe6e81e28dc3b3d76cce31525a91f32a387e1febd6b982caf8cdb",
        "sha256:f9210820ee4818d84658ed7df16a7f30c9fba7d8b139959950acef91745cc0f7"
    ],
    "version": "==3.1.4"
},
"certifi": {
    "hashes": [
        "sha256:13e698f54293db9f89122b0581843a782ad0934a4fe0172d2a980ba77fc61bb7",
        "sha256:9fa520c1bacfb634fa7af20a76bcbd3d5fb390481724c597da32c719a7dca4b0"
    ],
    "version": "==2018.4.16"
},
"cffi": {
    "hashes": [
        "sha256:151b7eefd035c56b2b2e1eb9963c90c6302dc15fbd8c1c0a83a163ff2c7d7743",
        "sha256:1553d1e99f035ace1c0544050622b7bc963374a00c467edafac50ad7bd276aef",
        "sha256:1b0493c091a1898f1136e3f4f991a784437fac3673780ff9de3bcf46c80b6b50",
        "sha256:2ba8a45822b7aee805ab49abfe7eec16b90587f7f26df20c71dd89e45a97076f",
        "sha256:3bb6bd7266598f318063e584378b8e27c67de998a43362e8fce664c54ee52d30",
        "sha256:3c85641778460581c42924384f5e68076d724ceac0f267d66c757f7535069c93",
        "sha256:3eb6434197633b7748cea30bf0ba9f66727cdce45117a712b29a443943733257",
        "sha256:495c5c2d43bf6cebe0178eb3e88f9c4aa48d8934aa6e3cddb865c058da76756b",
        "sha256:4c91af6e967c2015729d3e69c2e51d92f9898c330d6a851bf8f121236f3defd3",
        "sha256:57b2533356cb2d8fac1555815929f7f5f14d68ac77b085d2326b571310f34f6e",
        "sha256:770f3782b31f50b68627e22f91cb182c48c47c02eb405fd689472aa7b7aa16dc",
        "sha256:79f9b6f7c46ae1f8ded75f68cf8ad50e5729ed4d590c74840471fc2823457d04",
        "sha256:7a33145e04d44ce95bcd71e522b478d282ad0eafaf34fe1ec5bbd73e662f22b6",
```

```
      "sha256:857959354ae3a6fa3da6651b966d13b0a8bed6bbc87a0de7b38a549db1d2a359",
      "sha256:87f37fe5130574ff76c17cab61e7d2538a16f843bb7bca8ebbc4b12de3078596",
      "sha256:95d5251e4b5ca00061f9d9f3d6fe537247e145a8524ae9fd30a2f8fbce993b5b",
      "sha256:9d1d3e63a4afdc29bd76ce6aa9d58c771cd1599fbba8cf5057e7860b203710dd",
      "sha256:a36c5c154f9d42ec176e6e620cb0dd275744aa1d804786a71ac37dc3661a5e95",
      "sha256:a6a5cb8809091ec9ac03edde9304b3ad82ad4466333432b16d78ef40e0cce0d5",
      "sha256:ae5e35a2c189d397b91034642cb0eab0e346f776ec2eb44a49a459e6615d6e2e",
      "sha256:b0f7d4a3df8f06cf49f9f121bead236e328074de6449866515cea4907bbc63d6",
      "sha256:b75110fb114fa366b29a027d0c9be3709579602ae111ff61674d28c93606acca",
      "sha256:ba5e697569f84b13640c9e193170e89c13c6244c24400fc57e88724ef610cd31",
      "sha256:be2a9b390f77fd7676d80bc3cdc4f8edb940d8c198ed2d8c0be1319018c778e1",
      "sha256:ca1bd81f40adc59011f58159e4aa6445fc585a32bb8ac9badf7a2c1aa23822f2",
      "sha256:d5d8555d9bfc3f02385c1c37e9f998e2011f0db4f90e250e5bc0c0a85a813085",
      "sha256:e55e22ac0a30023426564b1059b035973ec82186ddddbac867078435801c7801",
      "sha256:e90f17980e6ab0f3c2f3730e56d1fe9bcba1891eeea58966e89d352492cc74f4",
      "sha256:ecbb7b01409e9b782df5ded849c178a0aa7c906cf8c5a67368047daab282b184",
      "sha256:ed01918d545a38998bfa5902c7c00e0fee90e957ce036a4000a88e3fe2264917",
      "sha256:edabd457cd23a02965166026fd9bfd196f4324fe6032e866d0f3bd0301cd486f",
      "sha256:fdf1c1dc5bafc32bc5d08b054f94d659422b05aba244d6be4ddc1c72d9aa70fb"
    ],
    "version": "==1.11.5"
  },
  "chardet": {
    "hashes": [
      "sha256:84ab92ed1c4d4f16916e05906b6b75a6c0fb5db821cc65e70cbd64a3e2a5eaae",
      "sha256:fc323ffcaeaed0e0a02bf4d117757b98aed530d9ed4531e3e15460124c106691"
    ],
    "version": "==3.0.4"
  },
```

```
"click": {
    "hashes": [
        "sha256:29f99fc6125fbc931b758dc053b3114e55c77a6e4c6c3a2674a2dc986016381d",
        "sha256:f15516df478d5a56180fbf80e68f206010e6d160fc39fa508b65e035fd75130b"
    ],
    "version": "==6.7"
},
"eventlet": {
    "hashes": [
        "sha256:06cffa55b335cc4fc32d0079242a81e8a9cddf2581d64d5f0543e2d412b26ca8",
        "sha256:554a50dad7abee0a9775b0780ce9d9c0bd9123dda4743c46d4314170267c6c47"
    ],
    "index": "pypi",
    "version": "==0.23.0"
},
"flask": {
    "hashes": [
        "sha256:2271c0070dbcb5275fad4a82e29f23ab92682dc45f9dfbc22c02ba9b9322ce48",
        "sha256:a080b744b7e345ccfcbc77954861cb05b3c63786e93f2b3875e0913d44b43f05"
    ],
    "index": "pypi",
    "version": "==1.0.2"
},
"flask-bcrypt": {
    "hashes": [
        "sha256:d71c8585b2ee1c62024392ebdbc447438564e2c8c02b4e57b56a4cafd8d13c5f"
    ],
    "index": "pypi",
    "version": "==0.7.1"
```

```
},
"flask-login": {
    "hashes": [
        "sha256:c815c1ac7b3e35e2081685e389a665f2c74d7e077cb93cecabaea352da4752ec"
    ],
    "index": "pypi",
    "version": "==0.4.1"
},
"flask-migrate": {
    "hashes": [
        "sha256:2697d2a9f9cd7de14a01a3de9f6e27d44ccffe1a2bf59bae2b2bb22cecb9d86a",
        "sha256:654af52ce82733fa066dc2a19cb462c52e56871a4785f03895a5bb43ea5a4e5b"
    ],
    "index": "pypi",
    "version": "==2.2.0"
},
"flask-sqlalchemy": {
    "hashes": [
        "sha256:3bc0fac969dd8c0ace01b32060f0c729565293302f0c4269beed154b46bec50b",
        "sha256:5971b9852b5888655f11db634e87725a9031e170f37c0ce7851cf83497f56e53"
    ],
    "index": "pypi",
    "version": "==2.3.2"
},
"flask-wtf": {
    "hashes": [
        "sha256:5d14d55cfd35f613d99ee7cba0fc3fbbe63ba02f544d349158c14ca15561cc36",
        "sha256:d9a9e366b32dcbb98ef17228e76be15702cd2600675668bca23f63a7947fd5ac"
    ],
```

```
    "index": "pypi",
    "version": "==0.14.2"
  },
  "greenlet": {
    "hashes": [
      "sha256:09ef2636ea35782364c830f07127d6c7a70542b178268714a9a9ba16318e7e8b",
      "sha256:0fef83d43bf87a5196c91e73cb9772f945a4caaff91242766c5916d1dd1381e4",
      "sha256:1b7df09c6598f5cfb40f843ade14ed1eb40596e75cd79b6fa2efc750ba01bb01",
      "sha256:1fff21a2da5f9e03ddc5bd99131a6b8edf3d7f9d6bc29ba21784323d17806ed7",
      "sha256:42118bf608e0288e35304b449a2d87e2ba77d1e373e8aa221ccdea073de026fa",
      "sha256:50643fd6d54fd919f9a0a577c5f7b71f5d21f0959ab48767bd4bb73ae0839500",
      "sha256:58798b5d30054bb4f6cf0f712f08e6092df23a718b69000786634a265e8911a9",
      "sha256:5b49b3049697aeae17ef7bf21267e69972d9e04917658b4e788986ea5cc518e8",
      "sha256:75c413551a436b462d5929255b6dc9c0c3c2b25cbeaee5271a56c7fda8ca49c0",
      "sha256:769b740aeebd584cd59232be84fdcaf6270b8adc356596cdea5b2152c82caaac",
      "sha256:a1852b51b06d1367e2d70321f6801844f5122852c9e5169bdfdff3f4d81aae30",
      "sha256:ad2383d39f13534f3ca5c48fe1fc0975676846dc39c2cece78c0f1f9891418e0",
      "sha256:b417bb7ff680d43e7bd7a13e2e08956fa6acb11fd432f74c97b7664f8bdb6ec1",
      "sha256:b6ef0cabaf5a6ecb5ac122e689d25ba12433a90c7b067b12e5f28bdb7fb78254",
      "sha256:c2de19c88bdb0366c976cc125dca1002ec1b346989d59524178adfd395e62421",
      "sha256:c7b04a6dc74087b1598de8d713198de4718fa30ec6cbb84959b26426c198e041",
      "sha256:f8f2a0ae8de0b49c7b5b2daca4f150fdd9c1173e854df2cce3b04123244f9f45",
      "sha256:fcfadaf4bf68a27e5dc2f42cbb2f4b4ceea9f05d1d0b8f7787e640bed2801634"
    ],
    "version": "==0.4.13"
  },
  "gunicorn": {
    "hashes": [
      "sha256:7ef2b828b335ed58e3b64ffa84caceb0a7dd7c5ca12f217241350dec36a1d5dc",
```

```
        "sha256:bc59005979efb6d2dd7d5ba72d99f8a8422862ad17ff3a16e900684630dd2a10"
    ],
    "index": "pypi",
    "version": "==19.8.1"
},
"idna": {
    "hashes": [
        "sha256:156a6814fb5ac1fc6850fb002e0852d56c0c8d2531923a51032d1b70760e186e",
        "sha256:684a38a6f903c1d71d6d5fac066b58d7768af4de2b832e426ec79c30daa94a16"
    ],
    "version": "==2.7"
},
"itsdangerous": {
    "hashes": [
        "sha256:cbb3fcf8d3e33df861709ecaf89d9e6629cff0a217bc2848f1b41cd30d360519"
    ],
    "version": "==0.24"
},
"jinja2": {
    "hashes": [
        "sha256:74c935a1b8bb9a3947c50a54766a969d4846290e1e788ea44c1392163723c3bd",
        "sha256:f84be1bb0040caca4cea721fcbbbbd61f9be9464ca236387158b0feea01914a4"
    ],
    "version": "==2.10"
},
"mako": {
    "hashes": [
        "sha256:4e02fde57bd4abb5ec400181e4c314f56ac3e49ba4fb8b0d50bba18cb27d25ae"
    ],
```

```
    "version": "==1.0.7"
  },
  "markupsafe": {
    "hashes": [
      "sha256:a6be69091dac236ea9c6bc7d012beab42010fa914c459791d627dad4910eb665"
    ],
    "version": "==1.0"
  },
  "psycopg2": {
    "hashes": [
      "sha256:027ae518d0e3b8fff41990e598bc7774c3d08a3a20e9ecc0b59fb2aaaf152f7f",
      "sha256:092a80da1b052a181b6e6c765849c9b32d46c5dac3b81bf8c9b83e697f3cdbe8",
      "sha256:0b9851e798bae024ed1a2a6377a8dab4b8a128a56ed406f572f9f06194e4b275",
      "sha256:179c52eb870110a8c1b460c86d4f696d58510ea025602cd3f81453746fccb94f",
      "sha256:19983b77ec1fc2a210092aa0333ee48811fd9fb5f194c6cd5b927ed409aea5f8",
      "sha256:1d90379d01d0dc50ae9b40c863933d87ff82d51dd7d52cea5d1cb7019afd72cd",
      "sha256:27467fd5af1dcc0a82d72927113b8f92da8f44b2efbdb8906bd76face95b596d",
      "sha256:32702e3bd8bfe12b36226ba9846ed9e22336fc4bd710039d594b36bd432ae255",
      "sha256:33f9e1032095e1436fa9ec424abcbd4c170da934fb70e391c5d78275d0307c75",
      "sha256:36030ca7f4b4519ee4f52a74edc4ec73c75abfb6ea1d80ac7480953d1c0aa3c3",
      "sha256:363fbbf4189722fc46779be1fad2597e2c40b3f577dc618f353a46391cf5d235",
      "sha256:6f302c486132f8dd11f143e919e236ea4467d53bf18c451cac577e6988ecbd05",
      "sha256:733166464598c239323142c071fa4c9b91c14359176e5ae7e202db6bcc1d2eb5",
      "sha256:7cbc3b21ce2f681ca9ad2d8c0901090b23a30c955e980ebf1006d41f37068a95",
      "sha256:888bba7841116e529f407f15c6d28fe3ef0760df8c45257442ec2f14f161c871",
      "sha256:8966829cb0d21a08a3c5ac971a2eb67c3927ae27c247300a8476554cc0ce2ae8",
      "sha256:8bf51191d60f6987482ef0cfe8511bbf4877a5aa7f313d7b488b53189cf26209",
      "sha256:8eb94c0625c529215b53c08fb4e461546e2f3fc96a49c13d5474b5ad7aeab6cf",
      "sha256:8ebba5314c609a05c6955e5773c7e0e57b8dd817e4f751f30de729be58fa5e78",
```

```
        "sha256:932a4c101af007cb3132b1f8a9ffef23386acc53dad46536dc5ba43a3235ae02",
        "sha256:ad75fe10bea19ad2188c5cb5fc4cdf53ee808d9b44578c94a3cd1e9fc2beb656",
        "sha256:aeaba399254ca79c299d9fe6aa811d3c3eac61458dee10270de7f4e71c624998",
        "sha256:b178e0923c93393e16646155794521e063ec17b7cc9f943f15b7d4b39776ea2c",
        "sha256:b68e89bb086a9476fa85298caab43f92d0a6af135a5f433d1f6b6d82cafa7b55",
        "sha256:d74cf9234ba76426add5e123449be08993a9b13ff434c6efa3a07caa305a619f",
        "sha256:f3d3a88128f0c219bdc5b2d9ccd496517199660cea021c560a3252116df91cbd",
        "sha256:fe6a7f87356116f5ea840c65b032af17deef0e1a5c34013a2962dd6f99b860dd"
    ],
    "index": "pypi",
    "version": "==2.7.4"
},
"pycparser": {
    "hashes": [
        "sha256:99a8ca03e29851d96616ad0404b4aad7d9ee16f25c9f9708a11faf2810f7b226"
    ],
    "version": "==2.18"
},
"python-dateutil": {
    "hashes": [
        "sha256:1adb80e7a782c12e52ef9a8182bebeb73f1d7e24e374397af06fb4956c8dc5c0",
        "sha256:e27001de32f627c22380a688bcc43ce83504a7bc5da472209b4c70f02829f0b8"
    ],
    "version": "==2.7.3"
},
"python-dotenv": {
    "hashes": [
        "sha256:4965ed170bf51c347a89820e8050655e9c25db3837db6602e906b6d850fad85c",
        "sha256:509736185257111613009974e666568a1b031b028b61b500ef1ab4ee780089d5"
```

```
    ],
    "index": "pypi",
    "version": "==0.8.2"
},
"python-editor": {
    "hashes": [
        "sha256:a3c066acee22a1c94f63938341d4fb374e3fdd69366ed6603d7b24bed1efc565"
    ],
    "version": "==1.0.3"
},
"requests": {
    "hashes": [
        "sha256:63b52e3c866428a224f97cab011de738c36aec0185aa91cfacd418b5d58911d1",
        "sha256:ec22d826a36ed72a7358ff3fe56cbd4ba69dd7a6718ffd450ff0e9df7a47ce6a"
    ],
    "version": "==2.19.1"
},
"six": {
    "hashes": [
        "sha256:70e8a77beed4562e7f14fe23a786b54f6296e34344c23bc42f07b15018ff98e9",
        "sha256:832dc0e10feb1aa2c68dcc57dbb658f1c7e65b9b61af69048abc87a2db00a0eb"
    ],
    "version": "==1.11.0"
},
"sqlalchemy": {
    "hashes": [
        "sha256:e21e5561a85dcdf16b8520ae4daec7401c5c24558e0ce004f9b60be75c4b6957"
    ],
    "version": "==1.2.9"
```

```
},
"stripe": {
  "hashes": [
    "sha256:0203e1cad9599bcafc6a0c76f1a333dbcdd7dadb45af696e8cdcc39a220f728e",
    "sha256:375191043c2aad432b46bc8fe414645102944b32ad0746d539ecb615d7474c21"
  ],
  "index": "pypi",
  "version": "==1.84.0"
},
"urllib3": {
  "hashes": [
    "sha256:a68ac5e15e76e7e5dd2b8f94007233e01effe3e50e8daddf69acfd81cb686baf",
    "sha256:b5725a0bd4ba422ab0e66e89e030c806576753ea3ee08554382c14e685d117b5"
  ],
  "version": "==1.23"
},
"werkzeug": {
  "hashes": [
    "sha256:c3fd7a7d41976d9f44db327260e263132466836cef6f91512889ed60ad26557c",
    "sha256:d5da73735293558eb1651ee2fddc4d0dedcfa06538b8813a2e20011583c9e49b"
  ],
  "version": "==0.14.1"
},
"wtforms": {
  "hashes": [
    "sha256:0cdbac3e7f6878086c334aa25dc5a33869a3954e9d1e015130d65a69309b3b61",
    "sha256:e3ee092c827582c50877cdbd49e9ce6d2c5c1f6561f849b3b068c1b8029626f1"
  ],
  "version": "==2.2.1"
```

```
    }
  },
  "develop": {
    "flake8": {
      "hashes": [
        "sha256:7253265f7abd8b313e3892944044a365e3f4ac3fcdcfb4298f55ee9ddf188ba0",
        "sha256:c7841163e2b576d435799169b78703ad6ac1bbb0f199994fc05f700b2a90ea37"
      ],
      "index": "pypi",
      "version": "==3.5.0"
    },
    "mccabe": {
      "hashes": [
        "sha256:ab8a6258860da4b6677da4bd2fe5dc2c659cff31b3ee4f7f5d64e79735b80d42",
        "sha256:dd8d182285a0fe56bace7f45b5e7d1a6ebcbf524e8f3bd87eb0f125271b8831f"
      ],
      "version": "==0.6.1"
    },
    "pycodestyle": {
      "hashes": [
        "sha256:682256a5b318149ca0d2a9185d365d8864a768a28db66a84a2ea946bcc426766",
        "sha256:6c4245ade1edfad79c3446fadfc96b0de2759662dc29d07d80a6f27ad1ca6ba9"
      ],
      "version": "==2.3.1"
    },
    "pyflakes": {
      "hashes": [
        "sha256:08bd6a50edf8cffa9fa09a463063c425ecaaf10d1eb0335a7e8b1401aef89e6f",
        "sha256:8d616a382f243dbf19b54743f280b80198be0bca3a5396f1d2e1fca6223e8805"
```

```
        ],
        "version": "==1.6.0"
      }
   }
}
```

Pipfile

```
[[source]]
url = "https://pypi.org/simple"
verify_ssl = true
name = "pypi"

[packages]
flask = "==1.0.2"
flask-sqlalchemy = "==2.3.2"
flask-wtf = "==0.14.2"
python-dotenv = "*"
flask-login = "==0.4.1"
"psycopg2" = "==2.7.4"
gunicorn = "==19.8.1"
flask-migrate = "==2.2.0"
flask-bcrypt = "==0.7.1"
eventlet = "*"
stripe = "*"

[dev-packages]
"flake8" = "*"

[requires]
```

```
python_version = "3.6"
```

---

app.py

```
from app import app


if __name__ == '__main__':
    app.jinja_env.cache = {}
    app.run()
```

.gitignore

```
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
env/
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
```

```
parts/
sdist/
var/
wheels/
*.egg-info/
.installed.cfg
*.egg

# PyInstaller
#  Usually these files are written by a python script from a template
#  before PyInstaller builds the exe, so as to inject date/other infos into it.
*.manifest
*.spec

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

# Unit test / coverage reports
htmlcov/
.tox/
.coverage
.coverage.*
.cache
nosetests.xml
coverage.xml
*.cover
.hypothesis/
```

```
# Translations
*.mo
*.pot

# Django stuff:
*.log
local_settings.py

# Flask stuff:
instance/
.webassets-cache

# Scrapy stuff:
.scrapy

# Sphinx documentation
docs/_build/

# PyBuilder
target/

# Jupyter Notebook
.ipynb_checkpoints

# pyenv
.python-version

# celery beat schedule file
celerybeat-schedule
```

```
# SageMath parsed files
*.sage.py

# dotenv
.env

# virtualenv
.venv
venv/
ENV/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/

# vscode files
.vscode
.vs
```

pylint files

.pylintrc

.flaskenv

FLASK_APP=app.py

.env

Flask Configs

SECRET_KEY=dRfUjXn2r5u8x/A?D(G+KbPeShVkYp3s6v9y$B&E)H@McQfTjWnZq4t7w!z%C*F-

SQLALCHEMY Configs

SQLALCHEMY_DATABASE_URI=postgresql://postgres:p@55w0rd1@localhost/testdb

SQLALCHEMY_TRACK_MODIFICATIONS=False

Flask-Login

USE_SESSION_FOR_NEXT=True

Flask ENV

FLASK_ENV=development

Flask-WTForms

RECAPTCHA_PUBLIC_KEY=6LckujcUAAAAAERy3iVKuP-hxkYH7p9LeISLYuf5

RECAPTCHA_PRIVATE_KEY=6LckujcUAAAAAPbJc_U5wLXLPvY7xzKuDQui_jrW

Script.py.mako

"""${message}

Revision ID: ${up_revision}

Revises: ${down_revision | comma,n}

Create Date: ${create_date}

"""

from alembic import op

import sqlalchemy as sa

${imports if imports else ""}

revision identifiers, used by Alembic.

revision = ${repr(up_revision)}

down_revision = ${repr(down_revision)}

branch_labels = ${repr(branch_labels)}

depends_on = ${repr(depends_on)}

def upgrade():

    ${upgrades if upgrades else "pass"}

def downgrade():

    ${downgrades if downgrades else "pass"}

Readme

Generic single-database configuration.

Env.py

from __future__ import with_statement

from alembic import context

from sqlalchemy import engine_from_config, pool

from logging.config import fileConfig

```
import logging

# this is the Alembic Config object, which provides
# access to the values within the .ini file in use.
config = context.config

# Interpret the config file for Python logging.
# This line sets up loggers basically.
fileConfig(config.config_file_name)
logger = logging.getLogger('alembic.env')

# add your model's MetaData object here
# for 'autogenerate' support
# from myapp import mymodel
# target_metadata = mymodel.Base.metadata
from flask import current_app
config.set_main_option('sqlalchemy.url',
                       current_app.config.get('SQLALCHEMY_DATABASE_URI'))
target_metadata = current_app.extensions['migrate'].db.metadata

# other values from the config, defined by the needs of env.py,
# can be acquired:
# my_important_option = config.get_main_option("my_important_option")
# ... etc.


def run_migrations_offline():
    """Run migrations in 'offline' mode.
```

```
    This configures the context with just a URL
    and not an Engine, though an Engine is acceptable
    here as well.  By skipping the Engine creation
    we don't even need a DBAPI to be available.

    Calls to context.execute() here emit the given string to the
    script output.

    """
    url = config.get_main_option("sqlalchemy.url")
    context.configure(url=url)

    with context.begin_transaction():
        context.run_migrations()


def run_migrations_online():
    """Run migrations in 'online' mode.

    In this scenario we need to create an Engine
    and associate a connection with the context.

    """

    # this callback is used to prevent an auto-migration from being generated
    # when there are no changes to the schema
    # reference: http://alembic.zzzcomputing.com/en/latest/cookbook.html
    def process_revision_directives(context, revision, directives):
        if getattr(config.cmd_opts, 'autogenerate', False):
```

```
        script = directives[0]
        if script.upgrade_ops.is_empty():
            directives[:] = []
            logger.info('No changes in schema detected.')

    engine = engine_from_config(config.get_section(config.config_ini_section),
                                prefix='sqlalchemy.',
                                poolclass=pool.NullPool)

    connection = engine.connect()
    context.configure(connection=connection,
                      target_metadata=target_metadata,
                      process_revision_directives=process_revision_directives,
                      **current_app.extensions['migrate'].configure_args)

    try:
        with context.begin_transaction():
            context.run_migrations()
    finally:
        connection.close()

if context.is_offline_mode():
    run_migrations_offline()
else:
    run_migrations_online()
```

---

alembic.ini

```
# A generic, single database configuration.
```

```
[alembic]
# template used to generate migration files
# file_template = %%(rev)s_%%(slug)s

# set to 'true' to run the environment during
# the 'revision' command, regardless of autogenerate
# revision_environment = false


# Logging configuration
[loggers]
keys = root,sqlalchemy,alembic

[handlers]
keys = console

[formatters]
keys = generic

[logger_root]
level = WARN
handlers = console
qualname =

[logger_sqlalchemy]
level = WARN
handlers =
qualname = sqlalchemy.engine
```

```
[logger_alembic]
level = INFO
handlers =
qualname = alembic

[handler_console]
class = StreamHandler
args = (sys.stderr,)
level = NOTSET
formatter = generic

[formatter_generic]
format = %(levelname)-5.5s [%(name)s] %(message)s
datefmt = %H:%M:%S
```

243d46491db2_init.py

```
"""Init

Revision ID: 243d46491db2
Revises:
Create Date: 2018-07-05 10:26:33.322832

"""
from alembic import op
import sqlalchemy as sa


# revision identifiers, used by Alembic.
revision = '243d46491db2'
```

```
down_revision = None
branch_labels = None
depends_on = None


def upgrade():
    # ### commands auto generated by Alembic - please adjust! ###
    op.create_table('Customers',
    sa.Column('Id', sa.Integer(), nullable=False),
    sa.Column('FirstName', sa.String(length=250), nullable=True),
    sa.Column('LastName', sa.String(length=250), nullable=True),
    sa.Column('Email', sa.String(length=100), nullable=True),
    sa.Column('Contact', sa.String(length=50), nullable=True),
    sa.PrimaryKeyConstraint('Id')
    )
    op.create_table('FlightInquiry',
    sa.Column('Id', sa.Integer(), nullable=False),
    sa.Column('FirstName', sa.String(length=100), nullable=True),
    sa.Column('lastName', sa.String(length=100), nullable=True),
    sa.Column('Email', sa.String(length=100), nullable=True),
    sa.Column('Origin', sa.String(length=100), nullable=True),
    sa.Column('Arrival', sa.String(length=100), nullable=True),
    sa.Column('DepartureDate', sa.Date(), nullable=True),
    sa.Column('ArrivalDate', sa.Date(), nullable=True),
    sa.Column('DesiredTime', sa.String(length=100), nullable=True),
    sa.Column('NumberOfAdults', sa.Integer(), nullable=True),
    sa.Column('NumberOfChild', sa.Integer(), nullable=True),
    sa.Column('NumberOfInfant', sa.Integer(), nullable=True),
    sa.Column('Note', sa.String(length=300), nullable=True),
```

```
sa.PrimaryKeyConstraint('Id')
)
op.create_table('HotelInquiry',
sa.Column('Id', sa.Integer(), nullable=False),
sa.Column('FirstName', sa.String(length=100), nullable=True),
sa.Column('lastName', sa.String(length=100), nullable=True),
sa.Column('Email', sa.String(length=100), nullable=True),
sa.Column('Location', sa.String(length=100), nullable=True),
sa.Column('Budget', sa.Integer(), nullable=True),
sa.Column('Guest', sa.Integer(), nullable=True),
sa.Column('checkInDate', sa.Date(), nullable=True),
sa.Column('checkOutDate', sa.Date(), nullable=True),
sa.Column('Note', sa.String(length=300), nullable=True),
sa.PrimaryKeyConstraint('Id')
)
op.create_table('Hotels',
sa.Column('Id', sa.Integer(), nullable=False),
sa.Column('Name', sa.String(length=250), nullable=True),
sa.Column('Room Type', sa.String(length=250), nullable=True),
sa.Column('Capacity', sa.String(length=4), nullable=True),
sa.Column('Details', sa.String(length=300), nullable=True),
sa.Column('CheckIn', sa.DateTime(), nullable=True),
sa.Column('CheckOut', sa.DateTime(), nullable=True),
sa.Column('Price', sa.Integer(), nullable=True),
sa.Column('ExpirationDate', sa.Date(), nullable=True),
sa.Column('IsExpired', sa.Boolean(), nullable=True),
sa.Column('isPackaged', sa.Boolean(), nullable=True),
sa.Column('DateCreated', sa.DateTime(), nullable=True),
sa.Column('DateUpdated', sa.DateTime(), nullable=True),
```

```
sa.Column('RemainingRooms', sa.Integer(), nullable=True),
sa.PrimaryKeyConstraint('Id')
)
op.create_table('LogTrail',
sa.Column('Id', sa.Integer(), nullable=False),
sa.Column('Event', sa.String(length=250), nullable=True),
sa.Column('EventTime', sa.DateTime(), nullable=True),
sa.PrimaryKeyConstraint('Id')
)
op.create_table('StripeCustomers',
sa.Column('Id', sa.Integer(), nullable=False),
sa.Column('Email', sa.String(length=50), nullable=True),
sa.Column('StripeCustomerId', sa.String(length=50), nullable=True),
sa.PrimaryKeyConstraint('Id')
)
op.create_table('Tickets',
sa.Column('Id', sa.Integer(), nullable=False),
sa.Column('FlightNo', sa.String(length=100), nullable=True),
sa.Column('Origin', sa.String(length=100), nullable=True),
sa.Column('Arrival', sa.String(length=100), nullable=True),
sa.Column('DepartureDate', sa.Date(), nullable=True),
sa.Column('DepartureTime', sa.Time(), nullable=True),
sa.Column('ArrivalDate', sa.Date(), nullable=True),
sa.Column('ArrivalTime', sa.Time(), nullable=True),
sa.Column('ReturnDate', sa.Date(), nullable=True),
sa.Column('ReturnTime', sa.Time(), nullable=True),
sa.Column('RemainingSlots', sa.Integer(), nullable=True),
sa.Column('ExpirationDate', sa.Date(), nullable=True),
sa.Column('Price', sa.Integer(), nullable=True),
```

```
sa.Column('IsExpired', sa.Boolean(), nullable=True),
sa.Column('IsPackaged', sa.Boolean(), nullable=True),
sa.Column('DateCreated', sa.DateTime(), nullable=True),
sa.Column('DateUpdated', sa.DateTime(), nullable=True),
sa.PrimaryKeyConstraint('Id')
)
op.create_table('Users',
sa.Column('Id', sa.Integer(), nullable=False),
sa.Column('Username', sa.String(length=100), nullable=True),
sa.Column('Password', sa.String(length=250), nullable=True),
sa.Column('Email', sa.String(length=250), nullable=True),
sa.Column('FirstName', sa.String(length=250), nullable=True),
sa.Column('LastName', sa.String(length=250), nullable=True),
sa.Column('Role', sa.String(length=100), nullable=True),
sa.Column('DateCreated', sa.DateTime(), nullable=True),
sa.Column('DateUpdated', sa.DateTime(), nullable=True),
sa.PrimaryKeyConstraint('Id')
)
op.create_table('HotelBooking',
sa.Column('Id', sa.Integer(), nullable=False),
sa.Column('ReferenceNumber', sa.String(length=50), nullable=True),
sa.Column('CustomersFk', sa.Integer(), nullable=True),
sa.Column('HotelsFk', sa.Integer(), nullable=True),
sa.Column('IsPaid', sa.Boolean(), nullable=True),
sa.ForeignKeyConstraint(['CustomersFk'], ['Customers.Id'], ),
sa.ForeignKeyConstraint(['HotelsFk'], ['Hotels.Id'], ),
sa.PrimaryKeyConstraint('Id')
)
op.create_table('Packages',
```

```
sa.Column('Id', sa.Integer(), nullable=False),
sa.Column('Destination', sa.String(length=50), nullable=True),
sa.Column('Price', sa.Integer(), nullable=True),
sa.Column('DaysOfStay', sa.Integer(), nullable=True),
sa.Column('Intenerary', sa.String(length=1000), nullable=True),
sa.Column('Inclusions', sa.String(length=1000), nullable=True),
sa.Column('RemainingSlots', sa.Integer(), nullable=True),
sa.Column('ExpirationDate', sa.Date(), nullable=True),
sa.Column('HotelsFk', sa.Integer(), nullable=True),
sa.Column('FlightFk', sa.Integer(), nullable=True),
sa.Column('isExpired', sa.Boolean(), nullable=True),
sa.ForeignKeyConstraint(['FlightFk'], ['Tickets.Id'], ),
sa.ForeignKeyConstraint(['HotelsFk'], ['Hotels.Id'], ),
sa.PrimaryKeyConstraint('Id')
)
op.create_table('Payments',
sa.Column('Id', sa.Integer(), nullable=False),
sa.Column('PaymentReference', sa.String(length=50), nullable=True),
sa.Column('BookingReference', sa.String(length=50), nullable=True),
sa.Column('PaymentFor', sa.String(length=50), nullable=True),
sa.Column('StripeCustomer', sa.Integer(), nullable=True),
sa.Column('StripChargeId', sa.String(length=50), nullable=True),
sa.ForeignKeyConstraint(['StripeCustomer'], ['StripeCustomers.Id'], ),
sa.PrimaryKeyConstraint('Id')
)
op.create_table('TicketBooking',
sa.Column('Id', sa.Integer(), nullable=False),
sa.Column('ReferenceNumber', sa.String(length=50), nullable=True),
sa.Column('CustomersFk', sa.Integer(), nullable=True),
```

```
    sa.Column('FlightFk', sa.Integer(), nullable=True),
    sa.Column('IsPaid', sa.Boolean(), nullable=True),
    sa.ForeignKeyConstraint(['CustomersFk'], ['Customers.Id'], ),
    sa.ForeignKeyConstraint(['FlightFk'], ['Tickets.Id'], ),
    sa.PrimaryKeyConstraint('Id')
    )
    # ### end Alembic commands ###


def downgrade():
    # ### commands auto generated by Alembic - please adjust! ###
    op.drop_table('TicketBooking')
    op.drop_table('Payments')
    op.drop_table('Packages')
    op.drop_table('HotelBooking')
    op.drop_table('Users')
    op.drop_table('Tickets')
    op.drop_table('StripeCustomers')
    op.drop_table('LogTrail')
    op.drop_table('Hotels')
    op.drop_table('HotelInquiry')
    op.drop_table('FlightInquiry')
    op.drop_table('Customers')
    # ### end Alembic commands ###
```

Routes.py

```
from flask import Blueprint, render_template, request, redirect, session
from flask import url_for, flash
from urllib.parse import urlparse, urljoin
```

```
from flask_login import LoginManager, login_user, current_user
from flask_login import logout_user
from app import forms
from app.models import db, User, Ticket, Hotel, Customer
from app.models import LogTrail, FlightInquiry, HotelInquiry
from app.models import Package, FlightBooking, StripeCustomer
from app.models import Payments, HotelBooking
from flask_bcrypt import Bcrypt
from functools import wraps
from datetime import timedelta, date, datetime
import stripe


view = Blueprint('main', __name__, template_folder='templates',
                 static_folder='static', static_url_path='/%s' % __name__)

bcrypt = Bcrypt()

loginManager = LoginManager()
loginManager.login_view = 'main.LogIn'
loginManager.login_message = 'Please log in'
loginManager.login_message_category = 'warning'

pubkey = 'pk_test_GjK3GmJJ1exs60wIcgTpfggq'
secretkey = 'sk_test_RXyvP1FBgkRyCwyEBGyZeymo'

stripe.api_key = secretkey

referenceNumber = datetime.now().strftime("%Y%m%d%H%M%S")
```

```
@loginManager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))



def login_required(role="ANY"):
    def wrapper(fn):
        @wraps(fn)
        def decorated_view(*args, **kwargs):
            if not current_user.is_authenticated:
                return loginManager.unauthorized()
            userRole = current_user.role
            if role == "AD":
                if((userRole != role) and role != "ANY"):
                    return loginManager.unauthorized()
            return fn(*args, **kwargs)
        return decorated_view
    return wrapper



# User Side
def is_safe_url(target):
    ref_url = urlparse(request.host_url)
    test_url = urlparse(urljoin(request.host_url, target))
    return (test_url.scheme in ('http', 'https') and
            ref_url.netloc == test_url.netloc)
```

```
@view.route('/login', methods=['GET', 'POST'])
def LogIn():
    session['next'] = request.args.get('next')
    form = forms.LoginForm()
    if form.validate_on_submit():
        user = User.query.filter(User.username == form.username.data).first()
        if user:
            inputPassword = form.password.data
            dbPassword = user.password
            checkBcrypt = bcrypt.check_password_hash(dbPassword, inputPassword)
            role = user.role
            if checkBcrypt:
                login_user(user)
                loggedUser = current_user
                fullName = loggedUser.firstName + ' ' + loggedUser.lastName
                if loggedUser.role == "AD":
                    role = "Admin"
                elif loggedUser.role == "RO":
                    role = "Reservation Officer"
                else:
                    role = "Financial Officer"
                event = (fullName + ' (' + role + ') logged in')
                newLogTrail = LogTrail(event=event)
                db.session.add(newLogTrail)
                db.session.commit()
                if role == "RO":
                    return redirect(url_for('main.UserHomeRO'))
                elif role == "FO":
```

```
                return redirect(url_for('main.LogIn'))
            else:
                return redirect(url_for('main.UserHomeRO'))
        flash('Invalid Credentials', 'error')
        return render_template('employee/login.html', form=form)
    flash('Username does not Exist', category='error')
    return render_template('employee/login.html', form=form)
  return render_template('employee/login.html', form=form)


@view.route('/user/register', methods=['GET', 'POST'])
def Register():
  form = forms.RegisterForm()
  if form.validate_on_submit():
    passwordBcrypt = bcrypt.generate_password_hash(form.password.data)
    emailUnique = User.query.filter(User.email == form.email.data).all()
    usernameUnique = (User.query
                .filter(User.username == form.username.data).all())
    if usernameUnique:
      flash('Username already Existing', 'error')
    elif emailUnique:
      flash('Email already Existing', 'error')
    else:
      newUser = User(username=form.username.data,
                password=passwordBcrypt.decode('utf-8'),
                firstName=form.firstName.data,
                lastName=form.lastName.data,
                email=form.email.data,
                role=form.role.data)
```

```
            db.session.add(newUser)
            if form.role.data == "AD":
                role = "Admin"
            elif form.role.data == "RO":
                role = "Reservation Officer"
            else:
                role = "Financial Officer"
            event = (form.username.data +
                     ' registered as ' +
                     role)
            newLogTrail = LogTrail(event=event)
            db.session.add(newLogTrail)
            db.session.commit()
            return redirect(url_for('main.LogIn'))
    return render_template('employee/register.html', form=form)


@view.route('/logout')
def LogOut():
    user = current_user
    fullName = user.firstName + ' ' + user.lastName
    if user.role == "AD":
        role = "Admin"
    elif user.role == "RO":
        role = "Reservation Officer"
    else:
        role = "Financial Officer"
    event = (fullName + ' (' + role + ') logged out')
    newLogTrail = LogTrail(event=event)
```

```
    db.session.add(newLogTrail)
    db.session.commit()
    logout_user()
    return redirect(url_for('main.LogIn'))


# Reservation Officer Side
@view.route('/user/home', methods=['GET', 'POST'])
@login_required(role="RO")
def UserHomeRO():
    return render_template('result.html')


@view.route('/ticket/add', methods=['GET', 'POST'])
@login_required(role="RO")
def CreateTicket():
    form = forms.RegisterTicket()
    if form.validate_on_submit():
        expireDate = form.departureDate.data - timedelta(days=7)
        newTicket = Ticket(origin=form.origin.data,
                           arrival=form.arrival.data,
                           flightNo=form.flightNo.data,
                           departureDate=form.departureDate.data,
                           departureTime=form.departureTime.data,
                           arrivalDate=form.arrivalDate.data,
                           arrivalTime=form.arrivalTime.data,
                           returnDate=form.returnDate.data,
                           returnTime=form.returnTime.data,
                           expirationDate=expireDate,
```

```
                remainingSlots=form.slots.data,
                isPackaged=form.isPackaged.data,
                price=form.price.data)
        db.session.add(newTicket)
        user = current_user
        fullName = user.firstName + ' ' + user.lastName
        if user.role == "AD":
            role = "Admin"
        elif user.role == "RO":
            role = "Reservation Officer"
        else:
            role = "Financial Officer"
        event = (fullName +
                 ' (' + role + ') created ' +
                 form.flightNo.data +
                 ' (' +
                 form.origin.data +
                 '-' +
                 form.arrival.data +
                 ')')
        newLogTrail = LogTrail(event=event)
        db.session.add(newLogTrail)
        db.session.commit()
        return redirect(url_for('main.UserHomeRO'))
    return render_template('employee/flights/addTicket.html', form=form)


@view.route('/hotel/add', methods=['GET', 'POST'])
@login_required(role="RO")
```

```
def CreateHotel():
    form = forms.RegisterHotel()
    if form.validate_on_submit():
        newHotel = Hotel(name=form.name.data,
                         roomType=form.roomType.data,
                         capacity=form.capacity.data,
                         details=form.details.data,
                         checkIn=form.checkIn.data,
                         checkOut=form.checkOut.data,
                         remainingRooms=form.rooms.data,
                         expirationDate=form.expirationDate.data,
                         isPackaged=form.isPackaged.data,
                         price=form.price.data)
        db.session.add(newHotel)
        user = current_user
        fullName = user.firstName + ' ' + user.lastName
        if user.role == "AD":
            role = "Admin"
        elif user.role == "RO":
            role = "Reservation Officer"
        else:
            role = "Financial Officer"
        event = (fullName +
                 ' (' + role + ') created ' +
                 form.name.data +
                 ' (' +
                 form.roomType.data +
                 ')')
        newLogTrail = LogTrail(event=event)
```

```
        db.session.add(newLogTrail)
        db.session.commit()
        return redirect(url_for('main.UserHomeRO'))
    return render_template('employee/hotels/addHotel.html', form=form)


@view.route('/package/add', methods=['GET', 'POST'])
@login_required(role="RO")
def CreatePackage():
    form = forms.RegisterPackage()
    availableHotel = Hotel.query.filter(Hotel.isExpired.is_(False)).all()
    hotelList = [(h.id, (h.name + ' - ' + h.roomType)) for h in availableHotel]
    form.hotels.choices = hotelList
    availableTicket = Ticket.query.filter(Ticket.isExpired.is_(False)).all()
    ticketList = [(t.id, (t.flightNo + (' (' +
                                t.origin +
                                ' - ' +
                                t.arrival +
                                ')')))
                  for t in availableTicket]
    form.tickets.choices = ticketList
    if form.validate_on_submit():
        newPackage = Package(destination=form.destination.data,
                             days=form.days.data,
                             expirationDate=form.expirationDate.data,
                             remainingSlots=form.remainingSlots.data,
                             intenerary=form.intenerary.data,
                             inclusions=form.inclusions.data,
                             price=form.price.data,
```

```
                    hotel=form.hotels.data,
                    flight=form.tickets.data)
        db.session.add(newPackage)
        user = current_user
        fullName = user.firstName + ' ' + user.lastName
        if user.role == "AD":
            role = "Admin"
        elif user.role == "RO":
            role = "Reservation Officer"
        else:
            role = "Financial Officer"
        event = (fullName +
                 ' (' + role + ') created ' +
                 form.destination.data)
        newLogTrail = LogTrail(event=event)
        db.session.add(newLogTrail)
        db.session.commit()
        return redirect(url_for('main.UserHomeRO'))
    return render_template('employee/packages/addPackage.html', form=form)



# Admin
@view.route('/logs')
@login_required('AD')
def LogTrails():
    logs = LogTrail.query.order_by(LogTrail.id.desc()).all()
    return render_template('employee/logs.html', logs=logs)
```

```
# Financial Officer
# Customer Side
@view.route('/')
def HomePage():
    return render_template('customer/homepage.html')


@view.route('/flight/summary/id/<int:id>', methods=['GET', 'POST'])
def FlightSummary(id):
    flightSummary = Ticket.query.get(id)
    form = forms.CustomerCount()
    session['flightId'] = id
    if form.validate_on_submit():
        if form.customerCounter.data > flightSummary.remainingSlots:
            errorMessage = ('%s must be less than Remaining Slots'
                            % (form.customerCounter.label.text))
            form.customerCounter.errors.append(errorMessage)
            return render_template('customer/flightCounter.html',
                                   form=form, flightSummary=flightSummary)
        return redirect(url_for("main.BookCustomerFlights",
                                counter=form.customerCounter.data,
                                session=session['flightId']))
    return render_template('customer/flightCounter.html',
                           flightSummary=flightSummary, form=form)


@view.route('/hotel/summary/id/<int:id>', methods=['GET', 'POST'])
def HotelSummary(id):
    hotelSummary = Hotel.query.get(id)
```

```
    session['hotelId'] = id
    form = forms.RoomCount()
    if form.validate_on_submit():
        if form.rooms.data > hotelSummary.remainingRooms:
            errorMessage = ('%s must be less than Remaining Slots'
                            % (form.rooms.label.text))
            form.rooms.errors.append(errorMessage)
            return render_template('customer/hotelCounter.html',
                                   form=form, hotelSummary=hotelSummary)
        return redirect(url_for('main.BookCustomerHotels',
                                counter=form.rooms.data))
    return render_template('customer/hotelCounter.html',
                           hotelSummary=hotelSummary, form=form)



@view.route('/flight/add/Customer/<int:counter>',
            methods=['GET', 'POST'])
def BookCustomerFlights(counter):
    form = forms.RegisterCustomerFlights(meta={'csrf': False})
    id = session['flightId']
    if form.validate_on_submit():
        for data in form.customer:
            customer = Customer(firstName=data.firstName.data,
                                lastName=data.lastName.data,
                                email=data.email.data,
                                contactNo=data.contactNo.data)
            db.session.add(customer)
            db.session.flush()
            flightTransaction = FlightBooking(referenceNumber=referenceNumber,
```

```
                                customer=customer.id,
                                flight=id)
            db.session.add(flightTransaction)
            db.session.commit()
        return redirect(url_for('main.PayFlights', counter=counter,
                        id=id, ref=referenceNumber))
    for count in range(counter):
        # form.customer.pop_entry()
        form.customer.append_entry()
    return render_template('customer/flightCustomerForm.html',
                        form=form,
                        counter=counter,
                        id=id)



@view.route('/hotel/add/Customer/<int:counter>', methods=['GET', 'POST'])
def BookCustomerHotels(counter):
    form = forms.RegisterCustomerHotels(meta={'csrf': False})
    id = session['hotelId']
    if form.validate_on_submit():
        for data in form.customer:
            customer = Customer(firstName=data.firstName.data,
                        lastName=data.lastName.data,
                        email=data.email.data,
                        contactNo=data.contactNo.data)
            db.session.add(customer)
            db.session.flush()
            hotelTransaction = HotelBooking(referenceNumber=referenceNumber,
                                customer=customer.id,
```

```
                                hotel=id)
            db.session.add(hotelTransaction)
            db.session.commit()
        return redirect(url_for('main.PayHotel', counter=counter,
                                id=id, ref=referenceNumber))
    form.customer.append_entry()
    return render_template('customer/hotelCustomerForm.html',
                           form=form,
                           counter=counter)


@view.route('/inquiry')
def Inquire():
    return render_template('customer/inquiry.html')


@view.route('/inquiry/flights', methods=['GET', 'POST'])
def InquireFlights():
    form = forms.InquiryFlights()
    if form.validate_on_submit():
        data = FlightInquiry(firstName=form.firstName.data,
                             lastName=form.lastName.data,
                             email=form.email.data,
                             origin=form.origin.data,
                             arrival=form.arrival.data,
                             departureDate=form.departureDate.data,
                             arrivalDate=form.arrivalDate.data,
                             time=form.time.data,
                             adult=form.adult.data,
```

```
                    child=form.child.data,
                    infant=form.child.data,
                    note=form.note.data)
        db.session.add(data)
        db.session.commit()
        return redirect(url_for('main.Inquire'))
    return render_template('customer/inquiryFlight.html', form=form)


@view.route('/inquiry/hotels', methods=['GET', 'POST'])
def InquireHotels():
    form = forms.InquiryHotels()
    if form.validate_on_submit():
        if form.checkOut.data < form.checkIn.data:
            errorMessage = ('%s must be greater than %s'
                    % (form.checkOut.label.text,
                      form.checkIn.label.text))
            form.checkOut.errors.append(errorMessage)
            return render_template('customer/inquiryHotel.html', form=form)
        data = HotelInquiry(firstName=form.firstName.data,
                    lastName=form.lastName.data,
                    email=form.email.data,
                    location=form.location.data,
                    budget=form.budget.data,
                    guest=form.guest.data,
                    checkIn=form.checkIn.data,
                    checkOut=form.checkOut.data,
                    note=form.note.data)
        db.session.add(data)
```

```
        db.session.commit()
        return redirect(url_for('main.Inquire'))
    return render_template('customer/inquiryHotel.html', form=form)


@view.route('/view/flights', methods=['GET', 'POST'])
def ViewFlights():
    now = date.today()
    (Ticket.query.filter(Ticket.expirationDate <= now)
     .update({Ticket.isExpired: True}))
    db.session.commit()
    viewFlights = Ticket.query.filter(Ticket.isExpired.is_(False)).all()
    return render_template('customer/viewFlights.html',
                           viewFlights=viewFlights)


@view.route('/view/hotels/', methods=['GET', 'POST'])
def ViewHotels():
    now = date.today()
    (Hotel.query.filter(Hotel.expirationDate <= now)
     .update({Hotel.isExpired: True}))
    viewHotels = Hotel.query.filter(Hotel.isExpired.is_(False)).all()
    return render_template('customer/viewHotels.html', viewHotels=viewHotels)


@view.route('/view/packages', methods=['GET', 'POST'])
def ViewPackages():
    now = date.today()
    (Package.query.filter(Package.expirationDate <= now)
```

```
    .update({Package.isExpired: True}))
   viewPackages = Package.query.filter(Package.isExpired.is_(False)).all()
   return render_template('customer/viewPackages.html',
                          viewPackages=viewPackages)


@view.route('/payment/flights/<int:counter>/<int:id>/<string:ref>',
            methods=['GET', 'POST'])
def PayFlights(counter, id, ref):
   flight = Ticket.query.get(id)
   return render_template('customer/paymentFlights.html',
                          flight=flight, referenceNumber=ref,
                          counter=counter, pubkey=pubkey)


@view.route('/payment/hotel/<int:counter>/<int:id>/<string:ref>',
            methods=['GET', 'POST'])
def PayHotel(counter, id, ref):
   hotel = Hotel.query.get(id)
   return render_template('customer/paymentHotel.html',
                          hotel=hotel, referenceNumber=ref,
                          counter=counter, pubkey=pubkey)


@view.route('/charge/flights/<int:counter>/<int:id>/<string:ref>',
            methods=['GET', 'POST'])
def ChargeFlights(counter, id, ref):
   flights = Ticket.query.get(id)
   email = request.form['stripeEmail']
```

```
source = request.form['stripeToken']
customer = (StripeCustomer.query
        .filter(StripeCustomer.email == email).first())
price = int(flights.price) * counter
if customer is None:
   customerStripe = (stripe.Customer
               .create(email=email,
                     source=source))
   newCustomer = StripeCustomer(email=email,
                       stripeCustomerId=customerStripe.id)
   db.session.add(newCustomer)
   db.session.flush()
   db.session.commit()
   customer = (StripeCustomer.query
            .filter(StripeCustomer.email == email).first())

charge = stripe.Charge.create(customer=customer.stripeCustomerId,
                     amount=price * 100,
                     currency="php",
                     description=(flights.flightNo +
                              '(' +
                              flights.origin +
                              ' - ' +
                              flights.arrival +
                              ')'))
newPayment = Payments(paymentReference=referenceNumber,
               bookingReference=ref,
               paymentFor='Flights',
               stripeCustomer=customer.id,
```

```
                    stripeChargeId=charge.id)
    db.session.add(newPayment)
    ticketCount = flights.remainingSlots - counter
    (Ticket.query.filter(Ticket.id == id)
     .update({Ticket.remainingSlots: ticketCount}))
    (FlightBooking.query.filter(FlightBooking.referenceNumber == ref)
     .update({FlightBooking.isPaid: True}))
    db.session.commit()
    return redirect(url_for('main.ConfirmFlight'))



@view.route('/charge/hotels/<int:counter>/<int:id>/<string:ref>',
            methods=['GET', 'POST'])
def ChargeHotel(counter, id, ref):
    hotels = Hotel.query.get(id)
    email = request.form['stripeEmail']
    source = request.form['stripeToken']
    customer = (StripeCustomer.query
                .filter(StripeCustomer.email == email).first())
    price = int(hotels.price) * counter
    if customer is None:
        customerStripe = (stripe.Customer
                          .create(email=email,
                                  source=source))
        newCustomer = StripeCustomer(email=email,
                                     stripeCustomerId=customerStripe.id)
        db.session.add(newCustomer)
        db.session.flush()
        db.session.commit()
```

```
        customer = (StripeCustomer.query
                    .filter(StripeCustomer.email == email).first())

    charge = stripe.Charge.create(customer=customer.stripeCustomerId,
                                  amount=price * 100,
                                  currency="php",
                                  description=(hotels.name +
                                               '(' +
                                               hotels.roomType +
                                               ')'))
    newPayment = Payments(paymentReference=referenceNumber,
                          bookingReference=ref,
                          paymentFor='Hotels',
                          stripeCustomer=customer.id,
                          stripeChargeId=charge.id)
    db.session.add(newPayment)
    ticketCount = hotels.remainingSlots - counter
    (Hotel.query.filter(Hotel.id == id)
     .update({Ticket.remainingRooms: ticketCount}))
    (FlightBooking.query.filter(FlightBooking.referenceNumber == ref)
     .update({FlightBooking.isPaid: True}))
    db.session.commit()
    return redirect(url_for('main.ConfirmHotel'))



@view.route('/payment/flights/confimed',
            methods=['GET', 'POST'])
def ConfirmFlight():
    return render_template('customer/chargeFlights.html')
```

```
@view.route('/payment/hotels/confimed',
        methods=['GET', 'POST'])
def ConfirmHotel():
   return render_template('customer/chargeHotel.html')
```

---

models.py

```
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from flask_migrate import Migrate

db = SQLAlchemy()
migrate = Migrate()



# Log In Trail
class LogTrail(db.Model):
   id = db.Column("Id", db.Integer, primary_key=True)
   event = db.Column("Event", db.String(250))
   eventTime = db.Column("EventTime", db.DateTime, default=db.func.now())

   __tablename__ = "LogTrail"



# User Models
class User(db.Model, UserMixin):
   id = db.Column('Id', db.Integer, primary_key=True)
   username = db.Column('Username', db.String(100))
```

```
    password = db.Column('Password', db.String(250))
    email = db.Column('Email', db.String(250))
    firstName = db.Column('FirstName', db.String(250))
    lastName = db.Column('LastName', db.String(250))
    role = db.Column('Role', db.String(100))
    dateCreated = db.Column('DateCreated', db.DateTime, default=db.func.now())
    dateUpdated = db.Column('DateUpdated', db.DateTime, onupdate=db.func.now())

    def getId(self):
        return self.id

    def isActive(self):
        return self.isActive

    def activateUser(self):
        self.activateUser = True

    def getUsername(self):
        return self.username

    def getRole(self):
        return self.role

    __tablename__ = 'Users'


# Tickets Model
class Ticket(db.Model):
    id = db.Column("Id", db.Integer, primary_key=True)
```

```
    flightNo = db.Column("FlightNo", db.String(100))
    origin = db.Column("Origin", db.String(100))
    arrival = db.Column("Arrival", db.String(100))
    departureDate = db.Column("DepartureDate", db.Date)
    departureTime = db.Column("DepartureTime", db.Time)
    arrivalDate = db.Column("ArrivalDate", db.Date)
    arrivalTime = db.Column("ArrivalTime", db.Time)
    returnDate = db.Column("ReturnDate", db.Date)
    returnTime = db.Column("ReturnTime", db.Time)
    remainingSlots = db.Column("RemainingSlots", db.Integer)
    expirationDate = db.Column("ExpirationDate", db.Date)
    price = db.Column('Price', db.Integer)
    isExpired = db.Column("IsExpired", db.Boolean, default=False)
    isPackaged = db.Column("IsPackaged", db.Boolean, default=False)
    dateCreated = db.Column('DateCreated', db.DateTime, default=db.func.now())
    dateUpdated = db.Column('DateUpdated', db.DateTime, onupdate=db.func.now())

    __tablename__ = "Tickets"

    def __repr__(self):
        return '%r : (%r - %r)' % (self.flightNo,
                                   self.origin,
                                   self.arrival)


# Hotel Model
class Hotel(db.Model):
    id = db.Column("Id", db.Integer, primary_key=True)
    name = db.Column("Name", db.String(250))
```

```
    roomType = db.Column("Room Type", db.String(250))
    capacity = db.Column("Capacity", db.String(4))
    details = db.Column("Details", db.String(300))
    checkIn = db.Column("CheckIn", db.DateTime)
    checkOut = db.Column("CheckOut", db.DateTime)
    price = db.Column('Price', db.Integer)
    expirationDate = db.Column("ExpirationDate", db.Date)
    isExpired = db.Column("IsExpired", db.Boolean, default=False)
    isPackaged = db.Column("isPackaged", db.Boolean, default=False)
    dateCreated = db.Column('DateCreated', db.DateTime, default=db.func.now())
    dateUpdated = db.Column('DateUpdated', db.DateTime, onupdate=db.func.now())
    remainingRooms = db.Column('RemainingRooms', db.Integer)

    __tablename__ = "Hotels"

    def __repr__(self):
        return '%s' % (self.id)

    def __str__(self):
        return '{} {}'.format(self.name, self.roomType)


# Customer Model
class Customer(db.Model):
    id = db.Column("Id", db.Integer, primary_key=True)
    firstName = db.Column("FirstName", db.String(250))
    lastName = db.Column("LastName", db.String(250))
    email = db.Column("Email", db.String(100))
    contactNo = db.Column("Contact", db.String(50))
```

```
    __tablename__ = "Customers"


class FlightInquiry(db.Model):
    id = db.Column("Id", db.Integer, primary_key=True)
    firstName = db.Column("FirstName", db.String(100))
    lastName = db.Column("lastName", db.String(100))
    email = db.Column("Email", db.String(100))
    origin = db.Column("Origin", db.String(100))
    arrival = db.Column("Arrival", db.String(100))
    departureDate = db.Column("DepartureDate", db.Date)
    arrivalDate = db.Column("ArrivalDate", db.Date)
    time = db.Column("DesiredTime", db.String(100))
    adult = db.Column("NumberOfAdults", db.Integer)
    child = db.Column("NumberOfChild", db.Integer)
    infant = db.Column("NumberOfInfant", db.Integer)
    note = db.Column("Note", db.String(300))

    __tablename__ = "FlightInquiry"


class HotelInquiry(db.Model):
    id = db.Column("Id", db.Integer, primary_key=True)
    firstName = db.Column("FirstName", db.String(100))
    lastName = db.Column("lastName", db.String(100))
    email = db.Column("Email", db.String(100))
    location = db.Column("Location", db.String(100))
    budget = db.Column("Budget", db.Integer)
```

```
    guest = db.Column("Guest", db.Integer)
    checkIn = db.Column("checkInDate", db.Date)
    checkOut = db.Column("checkOutDate", db.Date)
    note = db.Column("Note", db.String(300))

    __tablename__ = "HotelInquiry"


class Package(db.Model):
    id = db.Column("Id", db.Integer, primary_key=True)
    destination = db.Column("Destination", db.String(50))
    price = db.Column('Price', db.Integer)
    days = db.Column("DaysOfStay", db.Integer)
    intenerary = db.Column("Intenerary", db.String(1000))
    inclusions = db.Column("Inclusions", db.String(1000))
    remainingSlots = db.Column("RemainingSlots", db.Integer)
    expirationDate = db.Column("ExpirationDate", db.Date)
    hotel = db.Column('HotelsFk', db.Integer, db.ForeignKey('Hotels.Id'))
    flight = db.Column('FlightFk', db.Integer, db.ForeignKey('Tickets.Id'))
    isExpired = db.Column('isExpired', db.Boolean)

    __tablename__ = 'Packages'


class HotelBooking(db.Model):
    id = db.Column("Id", db.Integer, primary_key=True)
    referenceNumber = db.Column("ReferenceNumber", db.String(50))
    customer = db.Column('CustomersFk', db.Integer,
                         db.ForeignKey('Customers.Id'))
```

```
    hotel = db.Column('HotelsFk', db.Integer, db.ForeignKey('Hotels.Id'))
    isPaid = db.Column('IsPaid', db.Boolean, default=False)

    __tablename__ = 'HotelBooking'


class FlightBooking(db.Model):
    id = db.Column("Id", db.Integer, primary_key=True)
    referenceNumber = db.Column("ReferenceNumber", db.String(50))
    customer = db.Column('CustomersFk', db.Integer,
                         db.ForeignKey('Customers.Id'))
    flight = db.Column('FlightFk', db.Integer, db.ForeignKey('Tickets.Id'))
    isPaid = db.Column('IsPaid', db.Boolean, default=False)

    __tablename__ = 'TicketBooking'


class StripeCustomer(db.Model):
    id = db.Column("Id", db.Integer, primary_key=True)
    email = db.Column("Email", db.String(50))
    stripeCustomerId = db.Column("StripeCustomerId", db.String(50))

    __tablename__ = 'StripeCustomers'


class Payments(db.Model):
    id = db.Column("Id", db.Integer, primary_key=True)
    paymentReference = db.Column("PaymentReference", db.String(50))
    bookingReference = db.Column("BookingReference", db.String(50))
```

```
    paymentFor = db.Column("PaymentFor", db.String(50))
    stripeCustomer = db.Column("StripeCustomer",
                               db.ForeignKey('StripeCustomers.Id'))
    stripeChargeId = db.Column("StripChargeId", db.String(50))

    __tablename__ = "Payments"
```

Forms.py

```
from flask_wtf import FlaskForm, CSRFProtect, RecaptchaField
from wtforms import StringField, PasswordField, BooleanField, SelectField
from wtforms import IntegerField, TextAreaField, FormField, FieldList
from wtforms import DecimalField
from wtforms.fields.html5 import DateField, TimeField
from wtforms.validators import InputRequired, Length, EqualTo, Email
from wtforms.validators import NumberRange, ValidationError, Optional
from datetime import date

csrf = CSRFProtect()
# Validation Messages
# Register User Validations
usernameLength = ('Username has a minimum of 4 and a maximum '
                  'of 100 characters.')
passwordLength = ('Password has a minimum of 8 and a maximum '
                  'of 100 characters.')
firstNameLength = ('First Name has a maximum of 100 characters.')
lastNameLength = ('Last Name has a maximum of 100 characters.')
contactnoLength = ('Contact Number must be at least 7 digits')
# Register Ticket Validations
originLength = ('Origin has a maximum of 50 characters.')
```

```
arrivalLength = ('Arrival has a maximum of 50 characters.')
numberAdult = ('Number of Adult must be at least 1')
# Register Hotel Validations
hotelNameLength = ('Name has a maximum of 100 characters.')
roomTypeLength = ('Room Type has a maximum of 100 characters')
roomDetailsLength = ('Room Details has maximum of 300 characters')
# Inquire Hotel Validations
numberGuest = ('Number of Guest must be at least 1')
budget = ('Budget must be at lease 500 pesos')
# Register Package Validations
destinationLength = ('Destination has a maximum of 50 characters.')
# Fight Counter
customerCount = ('Customer Count must not be lower than 1')
# Hotel Counter
NumberOfRooms = ('Customer Count must not be lower than 1')



def DateCheck(form, field):
    if field.data < date.today():
        raise ValidationError(('%s must not be past '
                               'today' % (field.label.text)))



def AfterDateCheck(form, field, fieldTwo):
    if field.data > fieldTwo.data:
        raise ValidationError(field.label +
                              (' must not be before '
                               + fieldTwo.label))
```

```
class LoginForm(FlaskForm):
    username = StringField('Username',
                           [InputRequired('Username Required')])
    password = PasswordField('Password',
                             [InputRequired('Password is Required')])
    remember = BooleanField('Remember Me')


class RegisterForm(FlaskForm):
    username = StringField('Username',
                           [InputRequired('Username is Required'),
                            Length(min=4, max=100, message=usernameLength)])
    password = PasswordField('Password',
                             [InputRequired('Password is Required'),
                              Length(min=8, max=100, message=passwordLength),
                              EqualTo('confirm',
                                      'Your Password does not Match')])
    confirm = PasswordField('Repeat Password',
                            [InputRequired('Repeat your Password')])
    firstName = StringField('First Name',
                            [InputRequired('First Name is Required'),
                             Length(max=100, message=firstNameLength)])
    lastName = StringField('Last Name',
                           [InputRequired('Last Name is Required'),
                            Length(max=100, message=lastNameLength)])
    email = StringField('Email',
                        [InputRequired('Email is Required'),
                         Email('Not a valid Email')])
```

```
    role = SelectField('Role',
                       choices=[('RO', 'Reservation Officer'),
                                ('FO', 'Financial Officer'),
                                ('AD', 'Admin')])
    recaptcha = RecaptchaField()


class RegisterTicket(FlaskForm):
    flightNo = StringField('Flight Number',
                           [InputRequired('Flight Number is Required')])
    origin = StringField('Origin',
                         [InputRequired('Origin is Required'),
                          Length(max=50, message=originLength)])
    arrival = StringField('Arrival',
                          [InputRequired('Arrival is Required'),
                           Length(max=50, message=arrivalLength)])
    departureDate = DateField('Departure Date',
                              [InputRequired('Departure Date is Required')])
    departureTime = TimeField('Departure Time',
                              [InputRequired('Departure Time is Required')])
    arrivalDate = DateField('Arrival Date',
                            [InputRequired('Arrival Date is Required')])
    arrivalTime = TimeField('Arrival Time',
                            [InputRequired('Arrival Time is Required')])
    returnDate = DateField('Return Date',
                           [InputRequired('Return Date is Required')])
    returnTime = TimeField('Return Time',
                           [InputRequired('Return Time is Required')])
    slots = IntegerField('Slots',
```

```
                [InputRequired('Slots are Required')])
    price = DecimalField('Price',
                [InputRequired('Price is Required')])
    isPackaged = BooleanField('Packaged')


class RegisterHotel(FlaskForm):
    name = StringField('Hotel Name',
                [InputRequired('Hotel Name is Required'),
                 Length(max=100, message=hotelNameLength)])
    roomType = StringField('Room Type',
                  [InputRequired('Room Type is Required'),
                   Length(max=100, message=roomTypeLength)])
    capacity = IntegerField('Number of Capacity',
                   [InputRequired('Capacity is Required')])
    details = TextAreaField('Room Details',
                   [InputRequired('Room Details is Required'),
                    Length(max=300, message=roomDetailsLength)])
    checkIn = DateField('Check In Date',
                [InputRequired('CheckIn Date is Required')])
    checkOut = DateField('Check Out Date',
                   [InputRequired('Check Out Date is Required')])
    expirationDate = DateField('Expiration Date',
                      [InputRequired('Expiration Date is Required')])
    price = DecimalField('Price',
                   [InputRequired('Price is Required')])
    rooms = IntegerField('Number of Rooms',
                   [InputRequired('Number of Rooms is Needed'),
                    NumberRange(min=1,
```

```
                                message=NumberOfRooms)])
    isPackaged = BooleanField('Packaged')


class RegisterCustomerFlightsFields(FlaskForm):
    firstName = StringField('First Name',
                            [InputRequired('First Name is Required'),
                             Length(max=100, message=firstNameLength)])
    lastName = StringField('Last Name',
                           [InputRequired('Last Name is Required'),
                            Length(max=100, message=lastNameLength)])
    email = StringField('Email',
                        [InputRequired('Email is Required'),
                         Email('Not a valid Email')])
    contactNo = IntegerField('Contact Number',
                             [InputRequired('Contact Number is Required')])


class RegisterCustomerFlights(FlaskForm):
    customer = FieldList(FormField(RegisterCustomerFlightsFields))


class RegisterCustomerHotelsFields(FlaskForm):
    firstName = StringField('First Name',
                            [InputRequired('First Name is Required'),
                             Length(max=100, message=firstNameLength)])
    lastName = StringField('Last Name',
                           [InputRequired('Last Name is Required'),
                            Length(max=100, message=lastNameLength)])
```

```
    email = StringField('Email',
                        [InputRequired('Email is Required'),
                         Email('Not a valid Email')])
    contactNo = IntegerField('Contact Number',
                             [InputRequired('Contact Number is Required')])


class RegisterCustomerHotels(FlaskForm):
    customer = FieldList(FormField(RegisterCustomerHotelsFields))


class InquiryFlights(FlaskForm):
    firstName = StringField('First Name',
                            [InputRequired('First Name is Required'),
                             Length(max=100, message=firstNameLength)])
    lastName = StringField('Last Name',
                           [InputRequired('Last Name is Required'),
                            Length(max=100, message=lastNameLength)])
    email = StringField('Email',
                        [InputRequired('Email is Required'),
                         Email('Not a valid Email')])
    origin = StringField('Origin Location',
                         [InputRequired('Origin Location is Required'),
                          Length(max=100, message=originLength)])
    arrival = StringField('Arrival Location',
                          [InputRequired('Arrival Location is Required'),
                           Length(max=100, message=arrivalLength)])
    departureDate = DateField('Departure Date',
                              [InputRequired('Departure Date is Required')])
```

```
    arrivalDate = DateField('Arrival Date',
                            [InputRequired('Arrival Date is Required')])
    time = SelectField('Desired Time',
                       choices=[('AM', 'AM'),
                                ('PM', 'PM')])
    adult = IntegerField('Number of Adults',
                         [InputRequired('Number is Required'),
                          NumberRange(min=1, message=numberAdult)],
                         default=1)
    child = IntegerField('Number of Child',
                         [InputRequired('Number is Required')],
                         default=0)
    infant = IntegerField('Number of Infant(Below 2 years old)',
                          [InputRequired('Number is Required')],
                          default=0)
    note = TextAreaField('Note')


class InquiryHotels(FlaskForm):
    firstName = StringField('First Name',
                            [InputRequired('First Name is Required'),
                             Length(max=100, message=firstNameLength)])
    lastName = StringField('Last Name',
                           [InputRequired('Last Name is Required'),
                            Length(max=100, message=lastNameLength)])
    email = StringField('Email',
                        [InputRequired('Email is Required'),
                         Email('Not a valid Email')])
    location = StringField('Location',
```

```
                    [InputRequired('Please input a location')])
    budget = IntegerField('Budget Range',
                    [InputRequired('Input Budget Range'),
                     NumberRange(min=500, message=budget)],
                    default=500)
    guest = IntegerField('Number of Guest',
                    [InputRequired('Input Number of Guest'),
                     NumberRange(min=1, message=numberGuest)],
                    default=1)
    checkIn = DateField('Check-in Date',
                    [InputRequired('Check-in Date is Required'),
                     DateCheck])
    checkOut = DateField('Check-out Date',
                    [InputRequired('Check-out Date is Required'),
                     DateCheck])
    note = TextAreaField('Note')


class RegisterPackage(FlaskForm):
    destination = StringField('Destination',
                        [InputRequired('Input Destination'),
                         Length(max=50, message='destinationLength')])
    price = DecimalField('Price',
                    [InputRequired('Price is Required')])
    days = IntegerField('Days of Stay',
                    [InputRequired('Days of Stay Required')])
    intenerary = TextAreaField('Intenerary',
                        [InputRequired('Intenerary is Required')])
    inclusions = TextAreaField('Inclusions',
```

```
                    [InputRequired('Inclusions is Required')])
    remainingSlots = IntegerField('Remaining Slots',
                      [InputRequired('Slots is Required')])
    expirationDate = DateField('Expiration Date',
                    [InputRequired('Expiration Date is Required')])
    hotels = SelectField('Hotels', coerce=int)
    tickets = SelectField('Flights', coerce=int)


class RoomCount(FlaskForm):
    rooms = IntegerField('Number of Rooms',
                [InputRequired('Number of Rooms is Needed'),
                 NumberRange(min=1,
                        message=NumberOfRooms)])


class CustomerCount(FlaskForm):
    customerCounter = IntegerField('Customer Count',
                      [InputRequired('Customer Count is Needed'),
                       NumberRange(min=1,
                              message=customerCount)])
```

__init__.py

```
from flask import Flask
from os import getenv
from app.routes import view, loginManager, bcrypt
from app.models import db, migrate
from app.forms import csrf
```

```
# Flask Activation
app = Flask(__name__, static_folder=None)

# Config Set
secretKey = getenv('SECRET_KEY')
dbUri = getenv('SQLALCHEMY_DATABASE_URI')
sqlTrackModifcation = getenv('SQLALCHEMY_TRACK_MODIFICATIONS')
useSessionNext = getenv('USE_SESSION_FOR_NEXT')
recaptchaPub = getenv('RECAPTCHA_PUBLIC_KEY')
recaptchaPri = getenv('RECAPTCHA_PRIVATE_KEY')
# Config Activation
app.config['SECRET_KEY'] = secretKey
app.config['SQLALCHEMY_DATABASE_URI'] = dbUri
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = sqlTrackModifcation
app.config['USE_SESSION_FOR_NEXT'] = useSessionNext
app.config['RECAPTCHA_PUBLIC_KEY'] = recaptchaPub
app.config['RECAPTCHA_PRIVATE_KEY'] = recaptchaPri

# Sql Alchemy Activation
db.init_app(app)

# Flask-Login Activation
loginManager.init_app(app)

# Flask-Migrate Activation
migrate.init_app(app, db)

# Bcrypt Activation
bcrypt.init_app(app)
```

```
# Register Blueprints
app.register_blueprint(view)

# CSRF
csrf.init_app(app)

if __name__ == '__main__':
    app.jinja_env.cache = {}
    app.run()
```

---

result.html

```
{% extends 'base.html' %}

{% block navbar %}
    {% include 'partials/navbars/_navbarEmployee.html' %}
{% endblock %}

{% block jumbotron %}
<div class="jumbotron jumbotron-fluid">
    <div class="container">
        <h4 class="display-4">Welcome {{ current_user.firstName }} {{ current_user.lastName }}</h4>
        <hr class="my-4">
        <p class="lead">
            {% if current_user.role == "AD" %}
                As an Administrator, you are able to do what a
                Reservation Officer and Financial Officer can do in addition;
                an access to the Trail Logs of this application
            {% elif current_user.role == "RO" %}
```

```
            As a Reservation Officer, you are assigned to Handle Reservations and Bookings
          {% else %}
            As a Financial Officer, you are assigned to Track Down Glow of Money
          {% endif %}
        </p>
    </div>
</div>
{% endblock %}
```

Formhelper.html

```
{% macro render_field(field) %}
 <div class="form-group">
    {% if field.errors %}
        {{ field.label }}
        {{ field(class_='form-control is-invalid', **kwargs)|safe }}
        {% for error in field.errors %}
            <div class="invalid-feedback">{{ error }}</div>
        {% endfor %}
    {% else %}
        {{ field.label }}
        {{ field(class_='form-control', **kwargs)|safe }}
    {% endif %}
 </div>
{% endmacro%}

{% macro render_checkbox(field) %}
 <div class="form-group">
     <div class=form-check>
        {% if field.errors %}
```

```
            {{ field(class_='form-check-input is-invalid') }}
            {{ field.label(class_='form-check-label')|safe }}
        {% for error in field.errors %}
            <div class="invalid-feedback">{{ error }}</div>
        {% endfor %}
        {% else %}
            {{ field(class_='form-check-input') }}
            {{ field.label(class_='form-check-label')|safe }}
        {% endif %}
    </div>
 </div>
{% endmacro %}
```

Base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" integrity="sha384-
WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3ewGhXQFZCSftd1LZCfmhktB"
crossorigin="anonymous">
    <link rel="icon" type="image/png" href="favicon-16x16.png" sizes="16x16" />
    <title>
        {% block title %}{% endblock %}
    </title>
    <style>
        .vertical-center {
```

```
        margin-top: 10%;
    }
    .photoLogo {
        height: 40px;
        width: 50px;
    }
</style>
</head>
<body>
  {% block navbar %} {% endblock %}
  <!-- Jumbotron -->
  {% block jumbotron %} {% endblock %}
  <!-- Contents -->
  {% block content %} {% endblock %}

  <!-- Javascripts -->
  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
crossorigin="anonymous"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js"
integrity="sha384-smHYKdLADwkXOn1EmN1qk/HfnUcbVRZyYmZ4qpPea6sjB/pTJ0euyQp0Mk8ck+5T"
crossorigin="anonymous"></script>
  {% block javascript %} {% endblock %}
</body>
</html>
```

Register.html

```
{% extends 'base.html' %}
```

```
{% from "formhelper.html" import render_field %}
{% block title %} Register {% endblock %}

{% block navbar %}
   {% include 'partials/navbars/_navbarEmployee.html' %}
{% endblock %}

{% block content %}
<br>
   <div class="container">
      {% with errors = get_flashed_messages(category_filter=["error"]) %}
         {% if errors %}
            {% for error in errors %}
               <div class="alert alert-danger" role="alert">
                  {{ error }}
               </div>
            {% endfor %}
         {% endif %}
      {% endwith %}

      <form action="{{ url_for('main.Register') }}" method="post" novalidate>
         {{ form.csrf_token() }}

         {{ render_field(form.username) }}
         <div class="row">
            <div class="col">
               {{ render_field(form.password) }}
            </div>
            <div class="col">
```

```
        {{ render_field(form.confirm) }}
    </div>
</div>


<div class="row">
    <div class="col">
        {{ render_field(form.firstName) }}
    </div>
    <div class="col">
        {{ render_field(form.lastName) }}
    </div>
</div>


{{ render_field(form.email) }}
{{ render_field(form.role) }}
<div>
    {{ form.recaptcha }}
</div>
<br>
<div class="row">
    <div class="col-auto mr-auto">
        <a href="{{ url_for('main.LogIn') }}" class="btn btn-secondary">Cancel Registration</a>
    </div>
    <div class="col-auto">
        <input type="submit" value="Submit Registration" class="btn btn-primary">
    </div>
</div>
```

```
      </form>
   </div>
{% endblock %}
```

Logs.html

```
{% extends 'base.html' %}

{% block title %}Logs{% endblock %}
{% block navbar %}
   {% include 'partials/navbars/_navbarEmployee.html' %}
{% endblock %}

{% block content %}
<div class="container">
   <br>
   <table class="table table-striped table-bordered">
      <thead>
         <tr>
            <th>Events</th>
            <th>Time Stamp</th>
         </tr>
      </thead>
      <tbody>
         {% for log in logs %}
            <tr>
               <td>{{log.event}}</td>
               <td>{{log.eventTime.strftime('%B %d, %Y - %I:%M:%S %p')}}</td>
            </tr>
         {% endfor%}
```

```
      </tbody>
   </table>
</div>
{% endblock%}
```

Login.html

```
{% extends 'base.html' %}
{% from "formhelper.html" import render_field, render_checkbox %}
<!-- Title -->
{% block title %}Log In{% endblock %}
{% block navbar %}
   {% include 'partials/navbars/_navbarEmployee.html' %}
{% endblock %}
{% block content %}
<div class="container">
   <div class="d-flex vertical-center">
      <div class="my-auto w-100">
         <div class="col-5 mx-auto">
            <div class="card">
               <div class="card-body">
                  <div class="card-title">
                     {% with messages = get_flashed_messages(category_filter=["warning"]) %}
                        {% if messages %}
                           {% for message in messages %}
                              <div class="alert alert-warning" role="alert">
                                 {{ message }}
                              </div>
                           {% endfor %}
                        {% endif %}
```

```
            {% endwith %}
            {% with errors = get_flashed_messages(category_filter=["error"]) %}
              {% if errors %}
                {% for error in errors %}
                  <div class="alert alert-danger" role="alert">
                    {{ error }}
                  </div>
                {% endfor %}
              {% endif %}
            {% endwith %}
          </div>
          <form action="/login" method="post" novalidate>
            {{ form.csrf_token() }}
            {{ render_field(form.username) }}
            {{ render_field(form.password) }}
            {{ render_checkbox(form.remember) }}
            <input type="submit" value="Log In" class="btn btn-block btn-primary">
          </form>
        </div>
      </div>
      <br>
      <div class="card card border-info text-center">
        <div class="card-body">
          <div class="card-text">
            New Here? <a href="{{ url_for('main.Register') }}">Create an Account</a>
          </div>
        </div>
      </div>
    </div>
```

```
      </div>
   </div>
</div>
{% endblock %}
```

---

addPackage.html

```
{% extends 'base.html' %}
{% from "formhelper.html" import render_field, render_checkbox %}

{% block title %}Add Package{% endblock %}
{% block navbar %}
   {% include 'partials/navbars/_navbarEmployee.html' %}
{% endblock %}
{% block content %}
<div class="container">
   <form action="{{ url_for('main.CreatePackage') }}" method="post" novalidate>
      {{ form.csrf_token() }}
         <div class="row">
         <div class="col">{{ render_field(form.destination) }}</div>
         <div class="col">{{ render_field(form.days) }}</div>
         <div class="col">{{ render_field(form.price) }}</div>
         <div class="col">{{ render_field(form.remainingSlots) }}</div>
         {{ render_field(form.expirationDate) }}
      </div>
      <div class="row">
         <div class="col">{{ render_field(form.intenerary) }}</div>
      </div>
      <div class="row">
         <div class="col">{{ render_field(form.inclusions) }}</div>
```

```
      </div>
      <div class="row">
         <div class="col">{{ render_field(form.hotels) }}</div>
         <div class="col">{{ render_field(form.tickets) }}</div>
      </div>
      <input type="submit" value="Add Package" class="btn btn-primary">
   </form>
</div>
{% endblock %}
```

---

addHotel.html

```
{% extends 'base.html' %}
{% from "formhelper.html" import render_field, render_checkbox %}

{% block ticket %}Add Hotel{% endblock %}
{% block navbar %}
   {% include 'partials/navbars/_navbarEmployee.html' %}
{% endblock %}
{% block content %}
   <div class="container">
      <form action="{{ url_for('main.CreateHotel') }}" method="post" novalidate>
         {{ form.csrf_token() }}
            <div class="row">
            <div class="col">{{ render_field(form.name) }}</div>
            <div class="col">{{ render_field(form.roomType) }}</div>
            <div class="col">{{ render_field(form.capacity) }}</div>
         </div>
         <div class="row">
            <div class="col">{{ render_field(form.details) }}</div>
```

```
        </div>
        <div class="row">
          <div class="col">{{ render_field(form.checkIn) }}</div>
          <div class="col">{{ render_field(form.checkOut) }}</div>
          <div class="col">{{ render_field(form.price) }}</div>
          <div class="col">{{ render_field(form.rooms) }}</div>
        </div>
        {{ render_field(form.expirationDate) }}
        {{ render_checkbox(form.isPackaged) }}
        <input type="submit" value="Add Hotel" class="btn btn-primary">
      </form>
    </div>
{% endblock %}
```

---

addTicket.html

```
{% extends 'base.html' %}
{% from "formhelper.html" import render_field, render_checkbox %}

{% block title %}Add Ticket{% endblock %}
{% block navbar %}
  {% include 'partials/navbars/_navbarEmployee.html' %}
{% endblock %}
{% block content %}
<div class="container">
  <form action="{{ url_for('main.CreateTicket') }}" method="post" novalidate>
    {{ form.csrf_token() }}
    <div class="form-row">
      <div class="col">{{ render_field(form.flightNo) }}</div>
      <div class="col">{{ render_field(form.origin) }}</div>
```

```
    <div class="col">{{ render_field(form.arrival) }}</div>
</div>
<div class="form-row">
    <div class="col">
        {{ render_field(form.departureDate) }}
        {{ render_field(form.departureTime) }}
    </div>
    <div class="col">
        {{ render_field(form.arrivalDate) }}
        {{ render_field(form.arrivalTime) }}
    </div>
    <div class="col">
        {{ render_field(form.returnDate) }}
        {{ render_field(form.returnTime) }}
    </div>
</div>
<div class="row">
    <div class="col-3">
        {% if form.isPackaged.data %}
            <fieldset disabled id='disable'>{{ render_field(form.price) }}</fieldset>
        {% else %}
            {{ render_field(form.price) }}
        {% endif %}
    </div>
    <div class="col-3">
        {{ render_field(form.slots) }}
    </div>
</div>
<div class="row">
```

```
            <div class="col-3">
                {{ render_checkbox(form.isPackaged) }}
            </div>
        </div>
        <div class="row">
            <div class="col-auto mr-auto">
                <a href="{{ url_for('main.UserHomeRO') }}" class="btn btn-secondary">Cancel</a>
            </div>
            <div class="col-auto">
                <input type="submit" value="Add Ticket" class="btn btn-primary">
            </div>
        </div>
    </form>
</div>
{% endblock %}
{% block javascript %}
<script>
$(document).ready(function(){
  $('input:checkbox').on('click', function(){
     if($('input:checkbox').is(':checked')){
       $('#disable').prop('disabled', 'disabled');
       $('#price').val('');
     }
     else{
       $('#fieldset').prop('disabled', false);
     }
  });
});
</script>
```

```
{% endblock %}
```

---

viewPackages.html

```
{% extends 'base.html' %}

{% block title %}View Packages{% endblock %}

{% block navbar %}
  {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}

{% block jumbotron %}
  <div class="jumbotron jumbotron-fluid">
    <div class="container">
      <h4 class="display-4">Packages</h4>
      <p class="lead">Select your desired Package</p>
    </div>
  </div>
{% endblock %}

{% block content %}
<div class="container">
  {% for items in viewPackages|batch(3) %}
    <div class="card-deck">
      {% for package in items %}
      <div class="card">
        <h5 class="card-header">Destination: {{ package.destination }}</h5>
        <div class="card-body">
          <p class="card-text">Days of Stay: {{package.days}} </p>
```

```
            <p class="card-text">Price: Php {{ '%0.2f'| format(package.price|float) }} </p>
            <p class="card-text">Interary: {{ package.intenerary}} </p>
            <p class="card-text">Inclusions: {{ package.inclusions }} </p>
            <p class="card-text">Flight details: {{ package.flight }} </p>
            <p class="card-text">Hotel details: {{ package.hotel.name }} </p>
            <a href="" class="btn btn-primary">View Package</a>
          </div>
          <div class="card-footer">
            <p class="card-text text-muted text-right">Valid Until: {{ package.expirationDate.strftime('%B %d, %Y') }}</p>
          </div>
        </div>
        {% endfor %}
      </div>
      <br>
  {% endfor %}
</div>

{% endblock %}
```

---

viewHotels.html

```
{% extends 'base.html' %}

{% block title %}View Hotels{% endblock %}

{% block navbar %}
  {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}
```

```
{% block jumbotron %}
   <div class="jumbotron jumbotron-fluid">
      <div class="container">
         <h4 class="display-4">Hotels</h4>
         <p class="lead">Select your desired Hotel Booking</p>
      </div>
   </div>
{% endblock %}

{% block content %}
<div class="container">
   {% for items in viewHotels|batch(3) %}
      <div class="card-deck">
         {% for hotel in items %}
         <div class="card">
            <h5 class="card-header">{{ hotel.name }}</h5>
            <div class="card-body">
               <p class="card-text">Room Type: {{ hotel.roomType }}</p>
               <p class="card-text">Room Type: {{ hotel.capacity }}</p>
               <p class="card-text">Details: {{ hotel.details }}</p>
               <p class="card-text">Room Type: {{ hotel.roomType }}</p>
               <p class="card-text">Check-In Date: {{hotel.checkIn.strftime('%B %d, %Y') }}</p>
               <p class="card-text">Check-Out Date: {{hotel.checkOut.strftime('%B %d, %Y') }}</p>
               <p class="card-text">Price: Php {{ '%0.2f'| format(hotel.price|float) }} </p>
               <a href="{{url_for('main.HotelSummary', id=hotel.id)}}" class="btn btn-primary">View
Hotel</a>
            </div>
            <div class="card-footer">
```

```
                <p class="card-text text-muted text-right">Valid Until: {{ hotel.expirationDate.strftime('%B %d, %Y') }}</p>
            </div>
          </div>
          {% endfor %}
        </div>
        <br>
    {% endfor %}
</div>
{% endblock %}
```

---

viewFlights.html

```
{% extends 'base.html' %}

{% block title %}View Flights{% endblock %}

{% block navbar %}
    {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}

{% block jumbotron %}
    <div class="jumbotron jumbotron-fluid">
        <div class="container">
            <h4 class="display-4">Flights</h4>
            <p class="lead">Select your desired Flight</p>
        </div>
    </div>
{% endblock %}
```

```
{% block content %}

<div class="container">

  <br>

  {% for items in viewFlights|batch(3) %}

    <div class="card-deck">

      {% for flight in items %}

      <div class="card">

        <h5 class="card-header">Flight: {{ flight.flightNo }} ({{ flight.origin }} - {{ flight.arrival }})</h5>

        <div class="card-body">

          <p class="card-text">Departure Date: {{ flight.departureDate.strftime('%B %d, %Y') }} -  {{ flight.departureTime }}</p>

          <p class="card-text">Return Date: {{ flight.returnDate.strftime('%B %d, %Y') }} - {{ flight.returnTime }} </p>

          <p class="card-text">Price: Php {{ '%0.2f'| format(flight.price|float) }} </p>

          <p class="card-text">Remaining Slots: {{ flight.remainingSlots }} </p>

          <a href="{{url_for('main.FlightSummary', id=flight.id)}}" class="btn btn-primary">View Flight</a>

        </div>

        <div class="card-footer">

          <p class="card-text text-muted text-right">Valid Until: {{ flight.expirationDate.strftime('%B %d, %Y') }}</p>

        </div>

      </div>

      {% endfor %}

    </div>

    <br>

  {% endfor %}

</div>


{% endblock %}
```

paymentHotel.html

```
{% extends 'base.html' %}
{% from 'formhelper.html' import render_field %}

{% block title %} Hotel Review {% endblock %}

{% block navbar %}
   {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}

{% block jumbotron %}
   <div class="jumbotron jumbotron-fluid">
      <div class="container">
         <h4 class="display-5">
            Booking Reference Number: {{ referenceNumber }}
         </h4>
         <div class="row">
            <p class="lead col">Name: {{hotel.name}}</p>
            <p class="lead col">Number of Rooms booked: {{ counter }}</p>
         </div>
         <div class="row">
            <div class="lead col">Price: Php {{ '%0.2f'| format(hotel.price * counter|float)  }}</div>
            <div class="lead col">Room Type: {{ hotel.roomType }}</div>
         </div>
         <div class="row">
            <div class="lead col">check In: {{ hotel.checkIn.strftime('%B %d, %Y') }}</div>
            <div class="lead col">check Out: {{ hotel.checkOut.strftime('%B %d, %Y') }}</div>
         </div>
```

```
        <br>
        <center>
          <form action="{{ url_for('main.ChargeHotel', counter=counter, id=hotel.id,
ref=referenceNumber) }}" method="POST">
            <input type="hidden" name="csrf_token" value="{{ csrf_token() }}" />
          <script
              src="https://checkout.stripe.com/checkout.js" class="stripe-button"
              data-key="pk_test_GjK3GmJJ1exs60wIcgTpfggq"
              data-name="{{ hotel.name }} ({{hotel.roomType}})"
              data-description="Php {{ hotel.price * counter }}"
              data-image="https://stripe.com/img/documentation/checkout/marketplace.png"
              data-locale="auto">
            </script>
          </form>
        </center>
      </div>
    </div>
{% endblock %}
```

```
{% extends 'base.html' %}
{% from 'formhelper.html' import render_field %}

{% block title %} Flight Review {% endblock %}

{% block navbar %}
  {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}

{% block jumbotron %}
```

```
<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h4 class="display-5">
      Booking Reference Number: {{ referenceNumber }}
    </h4>
    <div class="row">
      <p class="lead col">Flight No: {{flight.flightNo}}</p>
      <p class="lead col">Number of Tickets booked: {{ counter }}</p>
    </div>
    <div class="row">
      <div class="lead col">Price: Php {{ flight.price * counter }}</div>
      <div class="lead col"></div>
    </div>
    <br>
    <center>
      <form action="{{ url_for('main.ChargeFlights', counter=counter, id=flight.id,
ref=referenceNumber) }}" method="POST">
        <input type="hidden" name="csrf_token" value="{{ csrf_token() }}" />
        <script
          src="https://checkout.stripe.com/checkout.js" class="stripe-button"
          data-key="pk_test_GjK3GmJJ1exs60wIcgTpfggq"
          data-name="{{flight.flightNo}} ({{flight.origin}} - {{flight.arrival}})"
          data-description="Php {{ flight.price * counter }}"
          data-image="https://stripe.com/img/documentation/checkout/marketplace.png"
          data-locale="auto">
        </script>
      </form>
    </center>
  </div>
```

```
    </div>
{% endblock %}
```

---

paymentFlights.html

```
{% extends 'base.html' %}
{% from 'formhelper.html' import render_field %}

{% block title %} Flight Review {% endblock %}

{% block navbar %}
    {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}

{% block jumbotron %}
    <div class="jumbotron jumbotron-fluid">
        <div class="container">
            <h4 class="display-5">
                Booking Reference Number: {{ referenceNumber }}
            </h4>
            <div class="row">
                <p class="lead col">Flight No: {{flight.flightNo}}</p>
                <p class="lead col">Number of Tickets booked: {{ counter }}</p>
            </div>
            <div class="row">
                <div class="lead col">Price: Php {{ flight.price * counter }}</div>
                <div class="lead col"></div>
            </div>
            <br>
            <center>
```

```
        <form action="{{ url_for('main.ChargeFlights', counter=counter, id=flight.id,
ref=referenceNumber) }}" method="POST">
          <input type="hidden" name="csrf_token" value="{{ csrf_token() }}" />
        <script
            src="https://checkout.stripe.com/checkout.js" class="stripe-button"
            data-key="pk_test_GjK3GmJJ1exs60wIcgTpfggq"
            data-name="{{flight.flightNo}} ({{flight.origin}} - {{flight.arrival}})"
            data-description="Php {{ flight.price * counter }}"
            data-image="https://stripe.com/img/documentation/checkout/marketplace.png"
            data-locale="auto">
          </script>
        </form>
      </center>
    </div>
  </div>
{% endblock %}
```

---

inquiryHotel.html

```
{% extends 'base.html' %}
{% from 'formhelper.html' import render_field %}

{% block title %} Inquire Hotels {% endblock %}

{% block navbar %}
  {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}

{% block content %}
  <div class="container">
```

```
<br>
<h4 class="display-4">Hotel Inquiry</h4>
<br>
<form action="{{ url_for('main.InquireHotels') }}" method="post" novalidate>
    {{ form.csrf_token }}
    <div class="row">
        <div class="col">
            {{ render_field(form.firstName) }}
        </div>
        <div class="col">
            {{ render_field(form.lastName) }}
        </div>
        <div class="col">
            {{ render_field(form.email) }}
        </div>
    </div>
    <div class="row">
            <div class="col">
                {{ render_field(form.location) }}
            </div>
            <div class="col">
                    {{ render_field(form.budget) }}
                </div>
            <div class="col">
                {{ render_field(form.guest) }}
            </div>
    </div>
    <div class="row">
        <div class="col">
```

```
                {{ render_field(form.checkIn) }}
            </div>
            <div class="col">
                {{ render_field(form.checkOut) }}
            </div>
        </div>
            {{ render_field(form.note) }}
        <div class="col-auto">
            <input type="submit" value="Submit" class="btn btn-primary">
        </div>
    </form>
  </div>
{% endblock %}
```

---

inquiryFlight.html

```
{% extends 'base.html' %}
{% from 'formhelper.html' import render_field %}

{% block title %} Inquire Flights {% endblock %}

{% block navbar %}
  {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}

{% block content %}
  <div class="container">
        <br>
        <h4 class="display-4">Flight Inquiry</h4>
        <br>
```

```
<form action="{{ url_for('main.InquireFlights') }}" method="post" novalidate>
    {{ form.csrf_token }}
    <div class="row">
        <div class="col">
            {{ render_field(form.firstName) }}
        </div>
        <div class="col">
            {{ render_field(form.lastName) }}
        </div>
        <div class="col">
            {{ render_field(form.email) }}
        </div>
    </div>
    <div class="row">
            <div class="col">
                {{ render_field(form.origin) }}
            </div>
            <div class="col">
                {{ render_field(form.arrival) }}
            </div>
    </div>
    <div class="row">
        <div class="col">
            {{ render_field(form.departureDate) }}
        </div>
        <div class="col">
            {{ render_field(form.arrivalDate) }}
        </div>
        <div class="col">
```

```
                {{ render_field(form.time) }}
            </div>
        </div>
        <div class="row">
            <div class="col">
                {{ render_field(form.adult) }}
            </div>
            <div class="col">
                {{ render_field(form.child) }}
            </div>
            <div class="col">
                {{ render_field(form.infant) }}
            </div>
        </div>
        {{ render_field(form.note) }}
        <div class="col-auto">
            <input type="submit" value="Submit" class="btn btn-primary">
        </div>
    </form>
  </div>

{% endblock %}
```

Inquiry.html

```
{% extends 'base.html' %}
{% from 'formhelper.html' import render_field %}

{% block title %} Flight Review {% endblock %}
```

```
{% block navbar %}
   {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}


{% block jumbotron %}
   <div class="jumbotron jumbotron-fluid">
      <div class="container">
         <h4 class="display-4">Welcome to our Inquiry Page</h4>
         <p class="lead">Below are the choices where you can Inquire your desired trips.</p>
         <hr class="my-4">
         <div class="row">
            <div class="col">
               <p class="lead text-center">You can Inquire here for your most wanted trip to your dream country or local paradise</p>
               <center>
                  <a href="{{ url_for('main.InquireFlights') }}" class="btn btn-block btn-warning col-5">Inquire Flight</a>
               </center>
            </div>
            <div class="col">
               <p class="lead text-center">You can Inquire here for your desired Vacation Hotel or just a relaxing Staycation</p>
               <center>
                  <a href="{{ url_for('main.InquireHotels') }}" class="btn btn-block btn-warning col-5">Inquire Hotel</a>
               </center>
            </div>
         </div>
      </div>
   </div>
```

```
{% endblock %}
```

---

hotelCustomerForm.html

```
{% extends 'base.html' %}
{% from "formhelper.html" import render_field %}

{% block content %}
<div class="container">
   <form action="{{ url_for('main.BookCustomerHotels', counter=counter) }}" method="post">
      {% for forms in form.customer  %}
         {{ forms.csrf_token }}
         <div class="row">
            <div class="col">
         {{ render_field(forms.firstName) }}
         </div>
            <div class="col">
            {{ render_field(forms.lastName)}}
            </div>
      </div>
         <div class="row">
            <div class="col">
            {{ render_field(forms.email)}}
            </div>
            <div class="col">
            {{ render_field(forms.contactNo)}}
         </div>
      </div>
      {% endfor %}
      <input type="submit" value="Submit">
```

```
    </form>
</div>
{% endblock %}
```

---

hotelCounter.html

```
{% extends 'base.html' %}
{% from 'formhelper.html' import render_field %}

{% block title %} Hotel Review {% endblock %}

{% block navbar %}
    {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}
{% block jumbotron %}
    <div class="jumbotron">
        <div class="container">
            <h4 class="display-4">Hotel {{ hotelSummary.name }} ({{ hotelSummary.roomType }})</h4>
            <p class="lead"></p>
            <div class="row">
                <div class="col">
                    <p class="lead">Details</p>
                    <p class="lead">{{ hotelSummary.details }}</p>
                </div>
                <div class="col">
                    <p class="lead">Remaining Rooms</p>
                    <p class="lead">{{ hotelSummary.remainingRooms }}</p>
                </div>
            </div>
            <p class="lead"></p>
```

```
        <p class="lead"></p>
        <p class="lead"></p>
        <form action="{{ url_for('main.HotelSummary', id=hotelSummary.id) }}" method="post"
novalidate class="col-5 no-gutters">
            {{ form.csrf_token }}
               {{ render_field(form.rooms) }}
            <input type="submit" value="Avail Hotel" class="btn btn-primary">
        </form>
      </div>
   </div>
{% endblock %}
```

Homepage.html

```
{% extends 'base.html' %}

{% block title %}Customer{% endblock %}

{% block navbar %}
   {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}

{% block jumbotron %}
   <div class="jumbotron jumbotron-fluid">
      <div class="container">
         <h1 class="display-4">Welcome Customer</h1>
         <p class="lead">This is the Online Booking Reservation System of First Choice Travel Hub.</p>
         <hr class="my-4">
         <p>You may choose your desired booking on the navbar above.</p>
         <p>Happy Booking!</p>
```

```
    </div>
  </div>
{% endblock %}
```

---

flightCustomerForm.html

```
{% extends 'base.html' %}
{% from "formhelper.html" import render_field %}

{% block content %}
<div class="container">
  <form action="{{ url_for('main.BookCustomerFlights', counter=counter) }}" method="post">
    {% for forms in form.customer  %}
      {{ forms.csrf_token }}
      <div class="row">
        <div class="col">
      {{ render_field(forms.firstName) }}
      </div>
        <div class="col">
        {{ render_field(forms.lastName)}}
        </div>
    </div>
      <div class="row">
        <div class="col">
        {{ render_field(forms.email)}}
        </div>
        <div class="col">
        {{ render_field(forms.contactNo)}}
      </div>
    </div>
```

```
      {% endfor %}
      <input type="submit" value="Submit">
   </form>
</div>
{% endblock %}
```

---

flightCounter.html

```
{% extends 'base.html' %}
{% from 'formhelper.html' import render_field %}

{% block title %} Flight Review {% endblock %}

{% block navbar %}
   {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}
{% block jumbotron %}
   <div class="jumbotron">
      <div class="container">
         <h4 class="display-4">Flight {{ flightSummary.flightNo }} ({{ flightSummary.origin }} - {{
flightSummary.arrival }})</h4>
         <hr class="my-4">
         <div class="row">
            <div class="col">
               <p class="lead">Departure</p>
               <p class="lead">{{ flightSummary.departureDate.strftime('%B %d, %Y') }} {{
flightSummary.departureTime }}</p>
            </div>
            <div class="col">
               <p class="lead">Return</p>
```

```
            <p class="lead">{{ flightSummary.returnDate.strftime('%B %d, %Y') }} {{
flightSummary.returnTime }}</p>
          </div>
        </div>
        <div class="row">
          <div class="col">
            <p class="lead">Remaining Slots: {{ flightSummary.remainingSlots }}</p>
          </div>
        </div>
        <p class="lead"></p>
        <p class="lead"></p>
        <p class="lead"></p>
        <form action="{{ url_for('main.FlightSummary', id=flightSummary.id) }}" method="POST"
novalidate class="col-5 no-gutters">
          {{ form.csrf_token }}
          {{ render_field(form.customerCounter) }}
          <input type="submit" value="Avail Flight" class="btn btn-primary">
        </form>
      </div>
    </div>
{% endblock %}
```

---

chargeHotel.html

```
{% extends 'base.html' %}
{% from 'formhelper.html' import render_field %}

{% block title %} Flight Review {% endblock %}

{% block navbar %}
```

```
   {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}

{% block jumbotron %}
   <div class="jumbotron jumbotron-fluid">
      <div class="container">
         <h4 class="display">Your booking has been Completed</h4>
         <p class="lead">Your credit card will be charge in a while.</p>
         <hr class="my-4">
         <p class="lead">You will be receiving an email within <strong>One Day</strong> from our Reservation Officer regarding on your Hotel.</p>
         <p class="lead">Thank you and Good Day</p>
         <center><a href="{{ url_for('main.HomePage') }}" class="btn btn-success">Return to Home Page</a></center>
      </div>
   </div>
{% endblock %}
```

---

chargeFlights.html

```
{% extends 'base.html' %}
{% from 'formhelper.html' import render_field %}

{% block title %} Flight Review {% endblock %}

{% block navbar %}
   {% include 'partials/navbars/_navbarCustomer.html' %}
{% endblock %}

{% block jumbotron %}
```

```
  <div class="jumbotron jumbotron-fluid">
    <div class="container">
      <h4 class="display">Your booking has been Completed</h4>
      <p class="lead">Your credit card will be charge in a while.</p>
      <hr class="my-4">
      <p class="lead">You will be receiving an email within <strong>One Day</strong> from our Reservation Officer regarding on your tickets.</p>
      <p class="lead">Thank you and Good Day</p>
      <center><a href="{{ url_for('main.HomePage') }}" class="btn btn-success">Return to Home Page</a></center>
    </div>
  </div>
{% endblock %}
```

Setting.json

```
{
  "python.linting.pylintEnabled": false,
  "python.linting.flake8Enabled": true,
  "python.linting.enabled": true
}
```

Launch.json

```
{
  // Use IntelliSense to learn about possible attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
  "configurations": [
```

```
{
    "name": "Python: Current File",
    "type": "python",
    "request": "launch",
    "program": "${file}"
},
{
    "name": "Python: Attach",
    "type": "python",
    "request": "attach",
    "localRoot": "${workspaceFolder}",
    "remoteRoot": "${workspaceFolder}",
    "port": 3000,
    "secret": "my_secret",
    "host": "localhost"
},
{
    "name": "Python: Terminal (integrated)",
    "type": "python",
    "request": "launch",
    "program": "${file}",
    "console": "integratedTerminal"
},
{
    "name": "Python: Terminal (external)",
    "type": "python",
    "request": "launch",
    "program": "${file}",
    "console": "externalTerminal"
```

```
},
{
    "name": "Python: Django",
    "type": "python",
    "request": "launch",
    "program": "${workspaceFolder}/manage.py",
    "args": [
        "runserver",
        "--noreload",
        "--nothreading"
    ],
    "debugOptions": [
        "RedirectOutput",
        "Django"
    ]
},
{
    "name": "Python: Flask (0.11.x or later)",
    "type": "python",
    "request": "launch",
    "module": "flask",
    "args": [
        "run",
        "--no-debugger",
        "--no-reload"
    ]
},
{
    "name": "Python: Module",
```

```
    "type": "python",
    "request": "launch",
    "module": "module.name"
},
{
    "name": "Python: Pyramid",
    "type": "python",
    "request": "launch",
    "args": [
        "${workspaceFolder}/development.ini"
    ],
    "debugOptions": [
        "RedirectOutput",
        "Pyramid"
    ]
},
{
    "name": "Python: Watson",
    "type": "python",
    "request": "launch",
    "program": "${workspaceFolder}/console.py",
    "args": [
        "dev",
        "runserver",
        "--noreload=True"
    ]
},
{
    "name": "Python: All debug Options",
```

```
        "type": "python",
        "request": "launch",
        "pythonPath": "${config:python.pythonPath}",
        "program": "${file}",
        "module": "module.name",
        "env": {
            "VAR1": "1",
            "VAR2": "2"
        },
        "envFile": "${workspaceFolder}/.env",
        "args": [
            "arg1",
            "arg2"
        ],
        "debugOptions": [
            "RedirectOutput"
        ]
    }
  ]
}
```
