1. Start ganache-cli with custom configurations:

*ganache-cli -l 999999999999 -m "candy maple cake sugar pudding cream honey rich smooth crumble sweet treat" -e 10000 -a 40 --allowUnlimitedContractSize*

1. From new terminal window run:

*truffle migrate --reset*

1. From the same terminal window start the development server. Oracles should be registered upon server start with output similar to the following:

![A screenshot of a computer

Description automatically generated]()

1. Launch the dapp (can be viewed in browser at <http://localhost:8000>)

*npm run dapp*

1. Simplifications made for running the dapp:
   1. Only one airline considered – the original contract owner (accounts[0]) registers all flights, and when a passenger purchases flights insurance, the airline is defaulted to contract owner
   2. Hard coded passengers – Fetching flight status is always sent from the first passenger account (accounts[6]), and insurance purchasing is always sent from the second passenger account (accounts[7])
2. Scenario
   1. Select a flight and amount of insurance to purchase

A screenshot of a cell phone

Description automatically generated

* 1. Check the console to see the initial balance and ending balance of the passengers account (should decrease by the amount of ether entered)
  2. Request status check for the flight that the insurance was purchased for. Loading text will appear until the oracles have verified the response at which point the status will be displayed on the front end and the passenger will be instructed to collect their payout.

A picture containing food

Description automatically generated

* 1. Oracles submitting responses for the request and reaching a minimum threshold (i.e. verified state) can be seen from the server terminal:

A picture containing water, computer

Description automatically generated

* 1. Collect payout by entering flight details

A close up of a logo

Description automatically generated

* 1. Check console to see 1.5X the amount of insurance being added to the passengers account

1. Potential Improvements
   1. If oracles do not reach a consensus the application effectively ‘hangs’ – no flight status is emitted. Ideally a timer would be implemented and if no status was returned during that time the customer would be asked to re-initiate the request.
   2. Oracles should be stored in memory so that they are no re-registered every time the application restarts. Node filesystem would be sufficient.
   3. Should be able to register new airlines from the front end and select the airline/passenger account from which to perform an operation. UI/UX would be very important to consider when extending this functionality.