

## PROJECT SPECIFICATION

**FlightSurety****Separation of Concerns, Operational Control and "Fail Fast"**

| CRITERIA                                 | MEETS SPECIFICATIONS   |
|--|--|
| Smart Contract Separation                | <p>Smart Contract code is separated into multiple contracts:</p> <ol style="list-style-type: none"><li>1) FlightSuretyData.sol for data persistence</li><li>2) FlightSuretyApp.sol for app logic and oracles code</li></ol>  |
| Dapp Created and Used for Contract Calls | <p>A Dapp client has been created and is used for triggering contract calls. Client can be launched with "npm run dapp" and is available at <a href="http://localhost:8000">http://localhost:8000</a></p> <p>Specific contract calls:</p> <ol style="list-style-type: none"><li>1) Passenger can purchase insurance for flight</li><li>2) Trigger contract to request flight status update</li></ol> |
| Oracle Server Application                | <p>A server app has been created for simulating oracle behavior. Server can be launched with "npm run server"</p>  |

| CRITERIA   | MEETS SPECIFICATIONS   |
|--|--|
|  |  |
| Operational status control is implemented in contracts | Students has implemented operational status control.   |
| Fail Fast Contract                                     | Contract functions "fail fast" by having a majority of "require()" calls at the beginning of function body |

## Airlines

| CRITERIA                        | MEETS SPECIFICATIONS  |
|---------------------------------|---|
| Airline Contract Initialization | First airline is registered when contract is deployed.  |
| Multiparty Consensus            | <p>Only existing airline may register a new airline until there are at least four airlines registered</p> <p>Demonstrated either with Truffle test or by making call from client Dapp</p> |
|                                 |   |

| CRITERIA             | MEETS SPECIFICATIONS  |
|----------------------|---|
|                      |   |
| Multiparty Consensus | <p>Registration of fifth and subsequent airlines requires multi-party consensus of 50% of registered airlines</p> <p>Demonstrated either with Truffle test or by making call from client Dapp</p>                               |
| Airline Ante         | <p>Airline can be registered, but does not participate in contract until it submits funding of 10 ether (make sure it is <b>not</b> 10 wei)</p> <p>Demonstrated either with Truffle test or by making call from client Dapp</p> |

## Passengers

| CRITERIA                 | MEETS SPECIFICATIONS  |
|--------------------------|---|
| Passenger Airline Choice | <p>Passengers can choose from a fixed list of flight numbers and departures that are defined in the Dapp client</p> <p>Your UI implementation should include:</p> <ol style="list-style-type: none"> <li>1. Fields for Airline Address and Airline Name</li> <li>2. Amount of funds to send/which airline to send to</li> <li>3. Ability to purchase flight insurance for no more than</li> </ol> |

| CRITERIA            | <p>5. Ability to purchase flight insurance for no more than 1 ether</p> <b>MEETS SPECIFICATIONS</b> |
|---------------------|---|
|                     |   |
| Passenger Payment   | Passengers may pay up to 1 ether for purchasing flight insurance.                                   |
| Passenger Repayment | If flight is delayed due to airline fault, passenger receives credit of 1.5X the amount they paid   |
| Passenger Withdraw  | Passenger can withdraw any funds owed to them as a result of receiving credit for insurance payout  |
| Insurance Payouts   | Insurance payouts are not sent directly to passenger's wallet                                       |

### Oracles (Server App)

| CRITERIA              | MEETS SPECIFICATIONS  |
|-----------------------|---|
| Functioning Oracle    | Oracle functionality is implemented in the server app.                                      |
| Oracle Initialization | Upon startup, 20+ oracles are registered and their assigned indexes are persisted in memory |
|                       |   |

| CRITERIA             | MEETS SPECIFICATIONS  |
|----------------------|---|
|                      |   |
| Oracle Updates       | Update flight status requests from client Dapp result in OracleRequest event emitted by Smart Contract that is captured by server (displays on console and handled in code)   |
| Oracle Functionality | Server will loop through all registered oracles, identify those oracles for which the OracleRequest event applies, and respond by calling into FlightSuretyApp contract with random status code of Unknown (0), On Time (10) or Late Airline (20), Late Weather (30), Late Technical (40), or Late Other (50) |

---