XRPL Data Validation and Oracles

1. Create a new JavaScript file and import the following libraries:

→ xrpl library: This library provides access to the XRPL blockchain.

→ rxjs library: This library provides reactive programming capabilities.

2. Create a class called DataValidationOracle that implements the following methods:

→ validateData(data): This method validates the given data and returns an error message if the data is invalid.

→ getOracleData(): This method returns the current value of the oracle data.

3. In the validateData() method, you can use the xrpl library to query the XRPL blockchain for the latest value of the oracle data. You can then use the rxjs library to validate the data and return an error message if the data is invalid.

4. In the getOracleData() method, you can use the xrpl library to query the XRPL blockchain for the latest value of the oracle data. You can then return the value of the oracle data.

5. Create a new instance of the DataValidationOracle class and use it to validate data and get oracle data.

```
const xrpl = require('xrpl');

const rxjs = require('rxjs');


class DataValidationOracle {

  constructor() {

    this._validatedData = '';

  }

  validateData(data) {

    const query = new xrpl.QueryBuilder();

    query.select('value');

    query.where('id', data.id);

    return xrpl.query(query).pipe(

      rxjs.map((response) => {
        if (!response.result.value) {
          return new Error('Invalid data');
        }
        this._validatedData = response.result.value;
        return null;
      })
    );
  }
  getOracleData() {
    return this._validatedData;
  }
}
const oracle = new DataValidationOracle();
const data = {
  id: '1234567890',
  value: '100',
};
oracle.validateData(data).subscribe((error) => {
  if (error) {
    console.log(error);
  } else {
    console.log('Data is valid');
  }
});
```