ISTS System Design Document

Date: 12/18/19

Author: Gerald Arocena

Reviewer(s): Krithika Sundararajan

# Introduction

*A high-level description of this document, for example, “This document defines the design for the Canonical Model”.*

Overview

*Overview of the problem to be solved. What is the problem and why is it being solved? How will the resulting solution provide business value?*

*Consider adding a diagram that explains how this component fits into the overall System with some descriptive text explaining the diagram.*

# Requirements

*This section provides a summary of the requirements for the <Component Name>.*

*Provide your understanding of the requirements, both functional and nonfunctional. Reference the provided Requirements and System Architecture documents. Do not cut and paste from the requirements document.*

*Product Manager and others can read this to understand what requirements your design will support. There is already a requirements doc, so keep this brief and to the point, highlighting the important requirements that the design is addressing. Structure in a way to provide a requirements checklist for your design.*

# Use Cases

*Enumerate the use cases supported by the design,*

*This design supports the following use cases:*

*Include a Use Case Diagram.*

*Include descriptions of each of the actors and use cases.*

# Implementation

*This section of the document will describe the implementation details for ...*

*The implementation section should cover the following topics:*

* *What are the classes, and their properties, associations and methods?*
* *What are the important interfaces and how they will be implemented?*
* *How are the requirements addressed?*

# Class Diagram

*The following class diagram defines the classes defined in this design. Remember to include exception classes.*

*CLASS DIAGRAM GOES HERE*

# Class Dictionary

*This section specifies the class dictionary for the class … defined within the package …*

## ResourceManagementService Interface

*Class 1 description*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| definePerson | (id : string, name : string, description : string, role : string, authTokenTuple : AuthTokenTuple) : Person |  |
| defineTeam | (id : string, name : string, description : string, type : string, authTokenTuple : AuthTokenTuple) : Team |  |
| defineLaunchPad | (id : string, name : string, location : string, authTokenTuple : AuthTokenTuple) : Launchpad |  |
| defineSpaceship | (id : string, model : string, name : string, maxSpeed : int, fuelType : string, cargoType : string, fuelCapacity : int, description : string, authTokenTuple : AuthTokenTuple) : Spaceship |  |
| defineSpaceship | (id : string, model : string, name : string, maxSpeed : int, fuelType : string, capacity : int, classType : string, fuelCapacity : int, description : string, authTokenTuple : AuthTokenTuple) : Spaceship |  |
| defineSpaceship | (id : string, model : string, name : string, maxSpeed : int, fuelType : string, fuelCapacity : int, description : string, authTokenTuple : AuthTokenTuple) : Spaceship |  |
| defineCommunicationSystem | (id : string, authTokenTuple : AuthTokenTuple) : CommunicationSystem |  |
| defineComputerSystem | (id : string, authTokenTuple : AuthTokenTuple) : ComputerSystem |  |
| addResourcePrice | (resourceName : string, price : int, authTokenTuple : AuthTokenTuple) : void |  |
| buyResource | (resourceName : string, amount : int, authTokenTuple : AuthTokenTuple) : void |  |
| getBudget | (authTokenTuple : AuthTokenTuple) : Integer |  |
| createEvent | (spaceshipId : string, simulatedEvent : string, authTokenTuple : AuthTokenTuple) : void |  |
| defineFuel | (typeId : string, amount : int, authTokenTuple : AuthTokenTuple) : Fuel |  |
| defineFuel | (typeId : string, authTokenTuple : AuthTokenTuple) : Fuel |  |
| getFuels | (authTokenTuple : AuthTokenTuple) : map<fuelId, Fuel> |  |
| getSpaceships | (authTokenTuple : AuthTokenTuple) : map<spaceshipId : Spaceship> |  |
| getEntities | (authTokenTuple : AuthTokenTuple) : map<entityId : Entity> |  |
| getLaunchpads | (authTokenTuple : AuthTokenTuple) : map<launchpadId : Launchpad> |  |
| getCommunicationSystem | (authTokenTuple : AuthTokenTuple) : CommunicationSystem |  |
| getComputerSystem | (authTokenTuple : AuthTokenTuple) : ComputerSystem |  |
| getResourcePrices | (authTokenTuple : AuthTokenTuple) : map<resourceId, Integer> |  |
| getEntitiesVisitor | (authTokenTuple : AuthTokenTuple) : EntitiesVisitor |  |

## Visitable Interface

Implements… Visitor Pattern

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| acceptVisitor | (visitor : Visitor) : void | Calls the visit method on the given visitor that corresponds to a Visitable object passing in the object as a parameter. |

## ResourceImpl

*Class 1 description*

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| prices | map<resourceId, Integer> |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
| entities | map<entityId, Entity> |  |
| launchpads | map<launchpadId, Launchpad> |  |
| spaceships | map<spaceshipId, Spaceship> |  |
| communicationSystem | CommunicationSystem |  |
| computerSystem | ComputerSystem |  |
| fuels | map<fuelId, Fuel> |  |
| ledger | Ledger |  |
| authenticator | StoreAuthenticationService |  |

## *Entity*

Entity is an abstract class that implements Visitable. Visitor pattern…

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| id | string |  |
| name | string |  |
| description | string |  |

## Team

Extends Entity

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| addEntity | (entity : Entity, authTokenTuple : AuthTokenTuple) : void |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| type | string |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
| entities | map<entityId, Entity> |  |
| parentTeam | Team |  |

## Person

Extends Entity.

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| role | string |  |

## Visitor Interface

*Class 1 description*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| visitResourceImpl | (resourceImpl : ResourceManagementService) : void |  |
| visitTeam | (team : Team) : void |  |
| visitPerson | (person : Person) : void |  |

## EntitiesVisitor

Implements Visitor

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
|  |  |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
|  |  |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
|  |  |  |

## CommunicationSystem

*Class 1 description*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| createEvent | (sourceShip : Spaceship, simulatedEvent : string) : string[] |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| id | string |  |
| status | string |  |
| upTime | integer |  |

## ComputerSystem

*Class 1 description*

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| id | string |  |
| status | string |  |
| upTime | Integer |  |

# Implementation Details

*Explain details of the implementation.*

*How do the various parts fit together or interact?*

*How does the design address the requirements? Justify your design decisions and how they address the requirements.*

*Some implementation details may be addressed in the class dictionary, but for things that are not, describe them here.*

*Remember to reference the requirements from the body of the design document to show how your design is addressing the requirements.*

# Exception Handling

*Provide details on your exception handling. What types of exceptions are expected and how are they handled by the design? Describe your exception classes and their properties.*

# Testing

*Provide a testing strategy for testing the component.*

* *Functional*
* *Performance*
* *Regression*
* *Exception Handling*

# Risks

*Document any risks identified during the design process.*

*Are there parts of the design that may not work or need to be implemented with special care or additional testing?*