*Customer Management System* Design Document

# Introduction

*This module is one of the components in the Interplanetary Space Transport System. It is dependent on the Interplanetary File System for recreation and for the providing the ability for customers to dump any travel related or discovery related files. It depends on the Authentication Service to identify users and maintain sessions. It also depends on the Ledger Service to undertake transactions. The Customer Management System, CMS, in short has a user interface accessible to end users and administrators. Customers can perform tasks ranging from registering as a user to booking flights and recording their journeys using this module, the Customer Management System.*

Overview

*The Customer Management System besides other tasks is the money engine for the Interplanetary Space Transport System. The system must be able to finance itself in order to fund upcoming space explorations and the main source of income is by having passengers onboard. CMS handles the customer’s interaction to booking flight including accepting payment methods besides providing a greater customer experience during flight. Customer Management System’s interaction with IPFS makes documents instantly accessible even in the extraterrestrial world. CMS as discussed above is dependent on the Authentication Service, the Ledger Service and IPFS. See diagram below for clear understanding of the interaction.*

# A screenshot of a map Description automatically generated

# 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 is the class dictionary for classes to be defined under the package ‘com.cscie97.ists.cms’*

## *ICustomerManagementSystem*

This is a REST API that is accessible to other modules in the ISTS system and the CMS GUI. It defines the contract for interaction. Customers can pass in their credentials to log in, browse flights, select flights, book flights, record discoveries and request entertainment using service API. Credentials must be passed to the methods and CMS obtains an access token from the Authentication Service on behalf of the customer.

***Methods***

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Signature** | **URL** | **Description** |
| getInstance | (): ICustomerManagementSystem | /api/cms  GET | Singleton pattern – This gets a singleton instance of the service API |
| createCustomer | (custName: string, accountId: string, email: string, authCredential: Credential): Customer | /api/cms/customer/  POST | Creates a new customer. The customer must have opened a new account in the Authentication Service and obtained credentials already |
| registerPointOfInterest | (custName: string, authCredential: Credential, pointOfInterest: PointOfInterest): Customer | /api/cms/customer/{custId}/poi/  POST | Registered customers can express interest to explore points of interest in the space |
| provideFeedback | (custName: string, authCredential: Credential, feedback: CustomerFeedback): Customer | /api/cms/customer/{custId}/feedback/  POST | Registered customers can provide feedback about their flight experience and space exploration |
| lookUpCustomer | (email: string): Customer | /api/cms/customer/{custEmail}  GET | Customers can be looked up using their email |
| getAllAvailableFlights | (): List<Flight> | /api/cms/flights  GET | Returns upcoming flights. When Flight Management System schedules a new flight, it is dumped to CMS for storage |
| selectAFlight | (flightNumber: string, credential: Credential): Flight | /api/cms/flights/{flightNum}  GET | Customers can choose a flight to book |
| bookAFlight | (custName: string, credential: Credential, accountId: string, flighNumber: string): TravelDocument | /api/cms/flights/{flightNum}  POST | Customers provide their name, credentials, blockchain account ids and the flight they are interested in. CMS authenticates the customer using the Authentication Service and processes transactions using the Ledger service. If the customer’s passport and visa is found in IPFS, the customer in return gets issued a travel document with welcome package and boarding pass in return from IPFS. The travel document is an association class between customer and flight reservation which contains 1 or more details of flight reservation. Flight reservations can be one way, round trip or guided trips. |
| provideTravelDocumentAndBoard | (custEmail: string, authCredential: Credential, travelDocument: TravelDocument): Customer | /api/cms/customer/{custEmail}  POST | When the time arrives for boarding the customer provides email and other credentials with the travel documents like a boarding pass previously issued. The customer will be marked as currentlyOnAir upon successful boarding. When the flight concludes, this flag will be set to false. |
| cancelFlight | (custEmail: string, authCredential: Credential, flightNumber: string): boolean | /api/cms/customer/{custEmail}/cancel  DELETE | Customers can cancel their flight before boarding time free of charge. Customers will be charged a fee if they cancel after boarding time and standby customers will be allowed to board. |
| requestEntertainment | (flightNumber: string): FlightEntertainment | /api/cms/flights/{flightNum}/customer//{custEmail}  GET | Customers currently on air can request entertainment. CMS will query IPFS to get the list of movies, music and books |
| recordDiscovery | (file: File): IPFS | /api/cms/discovery  POST | Any customer or part of the crew can post discoveries. CMS sends the files to IPFS. They can also post anonymously. It returns a pointer to IPFS and is downloadable |
| downloadDiscoveries | (): List<File> | /api/cms/discovery  GET | Customers can download all the recent discoveries. CMS queries IPFS to retrieve the files |

## *CustomerManagementSystem*

*This is an implementation of the ICustomerManagementSystem. It also has associations to IAuthenticationService, LedgerService and IPFS.*

***Methods***

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Signature** | **URL** | **Description** |
| getInstance | (): ICustomerManagementSystem | /api/cms  GET | Singleton pattern – This gets a singleton instance of the service API |
| createCustomer | (custName: string, accountId: string, email: string, authCredential: Credential): Customer | /api/cms/customer/  POST | Creates a new customer. The customer must have opened a new account in the Authentication Service and obtained credentials already |
| registerPointOfInterest | (custName: string, authCredential: Credential, pointOfInterest: PointOfInterest): Customer | /api/cms/customer/{custId}/poi/  POST | Registered customers can express interest to explore points of interest in the space |
| provideFeedback | (custName: string, authCredential: Credential, feedback: CustomerFeedback): Customer | /api/cms/customer/{custId}/feedback/  POST | Registered customers can provide feedback about their flight experience and space exploration |
| lookUpCustomer | (email: string): Customer | /api/cms/customer/{custEmail}  GET | Customers can be looked up using their email |
| getAllAvailableFlights | (): List<Flight> | /api/cms/flights  GET | Returns upcoming flights. When Flight Management System schedules a new flight, it is dumped to CMS for storage |
| selectAFlight | (flightNumber: string, credential: Credential): Flight | /api/cms/flights/{flightNum}  GET | Customers can choose a flight to book |
| bookAFlight | (custName: string, credential: Credential, accountId: string, flighNumber: string): TravelDocument | /api/cms/flights/{flightNum}  POST | Customers provide their name, credentials, blockchain account ids and the flight they are interested in. CMS authenticates the customer using the Authentication Service and processes transactions using the Ledger service. If the customer’s passport and visa is found in IPFS, the customer in return gets issued a travel document with welcome package and boarding pass in return from IPFS. The travel document is an association class between customer and flight reservation which contains 1 or more details of flight reservation. Flight reservations can be one way, round trip or guided trips. |
| provideTravelDocumentAndBoard | (custEmail: string, authCredential: Credential, travelDocument: TravelDocument): Customer | /api/cms/customer/{custEmail}  POST | When the time arrives for boarding the customer provides email and other credentials with the travel documents like a boarding pass previously issued. The customer will be marked as currentlyOnAir upon successful boarding. When the flight concludes, this flag will be set to false. |
| cancelFlight | (custEmail: string, authCredential: Credential, flightNumber: string): boolean | /api/cms/customer/{custEmail}/cancel  DELETE | Customers can cancel their flight before boarding time free of charge. Customers will be charged a fee if they cancel after boarding time and standby customers will be allowed to board. |
| requestEntertainment | (flightNumber: string): FlightEntertainment | /api/cms/flights/{flightNum}/customer//{custEmail}  GET | Customers currently on air can request entertainment. CMS will query IPFS to get the list of movies, music and books |
| recordDiscovery | (file: File): IPFS | /api/cms/discovery  POST | Any customer or part of the crew can post discoveries. CMS sends the files to IPFS. They can also post anonymously. It returns a pointer to IPFS and is downloadable |
| downloadDiscoveries | (): List<File> | /api/cms/discovery  GET | Customers can download all the recent discoveries. CMS queries IPFS to retrieve the files |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| customers | List<Customer> | List of registered customers |
| availableFlights | List<Flight> | List of upcoming fights. Once spaceship reaches destination, flights are taken of this list |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
| authenticationService | *IAuthenticationService* | Used to obtain tokens on behalf of customers and maintain sessions |
| ledgerService | *LedgerService* | Used to book flights and process transactions on behalf of a customer |
| iPFS | *IPFS* | Downloadable discoveries can be posted anonymously by customers, passports and visas can be uploaded, travel documents can be issued, entertainment can streamed through IPFS |

## *Flight*

*This object is instantiated when a new flight is scheduled. The Flight Management System dumps new flights to CMS when they are scheduled.*

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| spaceshipId | string | Id of the spaceship |
| flightNumber | string | Unique UUID assigned by the system when a new flight is scheduled |
| launchTime | Date | Time to launch spaceship |
| launchPad | Launchpad | Launchpad where spaceship takes off |
| destination | string | Destination of the flight |
| numOfStops | int | The number of stops before destination |
| wayPoints | string | Transit addresses |
| capacity | int | This capacity is the spaceship’s capacity less the number of persons in the crew |
| crew | People | Composite Pattern - A hierarchy of people consisting of teams and persons can be part of the crew. These can be operators, pilots and flight attendants |
| ticketPrice | double | Variable cost of ticket. Subject to change |
| passengerCount | int | Current passenger count and can be updated upon new customers booking a flight or cancelling a flight |

## *CUSTOMER*

*A customer who registered in the Authentication Service first can register as a customer in CMS. It is associated with the customer’s preferences and feedback to provide a greater customer experience. The customer must also setup an account in the ledger service to complete his/her profile and process payments.*

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| custName | string | Name of the customer |
| ledgerAccountId | string | Account id of the ledger service to process payments |
| custEmail | string | Customers can associate an email address to their account to receive notifications |
| credential | Credential | Credentials can be voiceprints, faceprints or username/password |
| isCurrentlyOnAir | boolean | Identifies whether the customer is in a current flight or on the ground |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
| feedbackList | List<CustomerFeedback> | The customer can have a history of feedback |
| pointOfInterestList | List<PointOfInterest> | The customer can have a list of points of interest. |
| travelDocument | TravelDocument | A customer can have a travel document. The travel document can be issued for multiple flight reservations |

## *CustomerFeedback*

*Customers can submit feedback anonymously or as an identified customer. CMS sends the feedback files to IPFS.*

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| experience | File | Customers can document their experiences |
| feedback | File | Customers can document their feedback |

## *PointOfInterest*

*Customers can register interest on what to explore. Customers can add as many points of interest as they wish*

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| poiName | string | Name of point of interest |
| poiDescription | string | Description of point of interest |
| location | string | Location of point of interest |
| images | File | Customers can have image files of what they want to explore. The files are stored in IPFS |
| notes | File | Customers can have notes of what they want to explore. The files are stored in IPFS |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
| poiType | PointOfInterestTypeEnum | Point of interest can be planet, moon, asteroids… |

## *PointOfInterestTypeEnum*

This defines types of points of interest

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| PLANET | string | Type of point of interest |
| MOON | string | Type of point of interest |
| ASTEROIDS | string | Type of point of interest |
| SOLAR\_SYSTEM | string | Type of point of interest |
| SPACESTATION | String | Type of point of interest |

## *TravelDocument*

*Travel documents are issued to customers upon successful booking of a flight. The customer needs to provide passports and visas to be uploaded to IPFS from CMS. IPFS in return issues ticket id, boarding pass and welcoming package in return. TravelDocument is an association class between Customer and FlightReservation.*

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| ticketId | string | Unique UUID issued by IPFS upon successful booking of a flight |
| passport | File | Must be provided by customer. CMS sends it to IPFS for uploading |
| boardingPass | File | Issued by IPFS upon successful booking of a flight |
| visa | File | Must be provided by customer. CMS sends it to IPFS for uploading |
| welcomePackage | File | Issued by IPFS upon successful booking of a flight |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
| reservations | List<FlightReservation> | 1 or more reservations associated with the travel document |

## *FlightReservation*

Since the customer selects the flightNumber when selecting a flight some of the details will be pulled down from there. However, the customer may pick a different return flight or a different departure date and those specifications will be defined here.

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| flight | Flight | The flight selected by the customer |
| departure | Date | Departure selected by the customer |
| return | Date | Return date selected by the customer |
| availableFlights | List<Flight> | A customer can alternate between the available flights free of charge |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
| flightType | FlightTypeEnum | Flights can be one-way, round trip or guided tour |
| entertainment | FlightEntertainment | Entertainment streamed through IPFS |

## *FlightTypeEnum*

This defines types of flights

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| ONEWAY | string | Passengers can pick one-way flight |
| ROUND\_TRIP | string | Passengers can pick a round trip flight |
| GUIDED\_TOUR | string | Passengers can pick guided tours |

## *FlightEntertainment*

*Customers on air can request entertainment through CMS portal and IPFS will deliver content*

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| movies | File | Can be streamed through IPFS |
| books | File | Can be streamed through IPFS |
| music | File | Can be streamed through IPFS |

## *CustomerNotFoundException*

This is thrown when a customer is not registered yet

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| reason | string | Reason for failure |
| fix | string | Hint to fix the problem |

## *ErrorUploadingToIPFSException*

This is thrown when there is connectivity error with IPFS

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| reason | string | Reason for failure |
| fix | string | Hint to fix the problem |

## *FlightNotFoundException*

This is thrown when flight is not available due to getting cancelled for technical issues or if the boarding time has passed or if the flight was never scheduled by the Flight Management System.

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| reason | string | Reason for failure |
| fix | string | Hint to fix the problem |

# 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?*