Statement of Work

|  |  |
| --- | --- |
| Date | 24 Jan 2019 |
| Client | World Plane, Inc. |
| Job Name | Ticket Reservation System – Proof of Concept |
| Requested by | Ekalb Noslen |

Summary

World Plane, Inc. (hereafter identified as WPI) wishes to evaluate the feasibility of transitioning our Travel Agency airline travel reservation system to a Retail Customer airline reservation system. As such, WPI wants a small-scale ‘proof of concept’ software application developed using the JAVA language.

Project Scope

This SOW (Statement of Work) identifies the activities and deliverables for this effort.

WPI currently has a subset of our airline flight database running on an Internet accessible server. The database information will be accessed using a standard [plain text] HTTP GET Application Programming Interface (API). Data is returned is returned from requests using plain text XML strings.

* Documentation for the APIs, including preconditions, post conditions, invariants and side effects will be provided to all development teams supporting this project.
* Sample Java code will be provided to demonstrate a subset of the API calls and possible means of parsing the XML document results returned as a string
* Sample results will be provided to demonstrate the XML document format

This proof of concept software (hereafter identified as client software) will be conceptually similar in behavior to a number of existing web-based airline reservation systems (such as kayak.com or expedia.com), but the client software does not need to be browser-based or even employ a graphical user interface (GUI).

Upon completion of this project ,a retail customer will be able to use the delivered client software to select a departure airport and arrival airport for desired travel. The client software will allow passengers to reserve travel from destination airport to arrival airport using a series of connecting flights with a maximum of two stopovers. Each stopover will need to be of sufficient time (layover time) to allow a passenger to transition from the gate of the arrival flight to the gate of the next departing flight before the departing flights leaves. The layover time will also need to be of sufficient length to allow the airline to transfer the passenger’s bags from one flight to the next. The departure time and arrival time of each leg will be displayed to the customer in airport local time. The server database stores and returns flight times in GMT (Greenwich Mean Time.)

Passengers will have the ability to reserve flights to travel either one-way (from departure to destination), or reserve a round-trip flight (from departure to destination and back to original departure airport.) The customer will be able to reserve either first class seating or coach seating for travel. The client software should properly behave in situations where requested seating is not available for all legs of the flight. Customers will be able to search for flights using departure date, arrival dates and respective time windows within each. The client software will not offer flight choices with unreasonable layover times.

Multiple contractor teams will be developing clients and they will be tested by WPI at the same time. This means that multiple clients will be running against the same server database and will need to coordinate database changes. To support concurrency, the server API will support locking of the database. When locked, only the client software which locked the database will be able to update the database. The database will prohibit changes when the database is not locked. Database locks, if not explicitly released, will be automatically released within a reasonable period.

As a proof of concept, the behavior of the client software should remain simple. The client software will have the ability to reserve seats (first class or coach) for each leg of flights, but the identification of the passenger reserving the flights will not be entered into the client software nor will it be stored by the server. The client software will not allow a reservation to be deleted once it is made. The pricing of flights from departure to destination will be displayed to the customer, but the client software will not support any payment processing.

To support the customer’s selection of flights the client software will allow the customer to display flights sorted by price, departure time, arrival time or travel time. The client software will allow the customer to select flights and confirm the selection prior to the reservation being made. Timeliness of response to user actions must be responsive.

To support development and testing the server database will provide the following:

* Latitude and Longitude of the airports in the database allowing determination of time zone for each airport
* A separate, unique database for each contractor team during early development so developers have full control of the database data during developing and unit testing.
* The ability to reset the individual, contractor specific database to its original state.
* During integration and system verification testing, and user acceptance testing, client software will run against a single, shared database.

Schedule

The following deliverables will be made by each development team no later than the finish dates specified.

| Task | | Finish Date |
| --- | --- | --- |
| Requirements Analysis Documentation (RAD) delivery | 2/28/2019 |
| Prototype Demonstration and updated RAD | 3/21/2019 |
| Test Plan delivery | 4/4/2019 |
| Completed project including updated documentation and test results | 4/25/2019 |

Deliverables will consist of the following:

Requirements Analysis Document (RAD)

* User scenarios / stories based on the information contained in the Project Scope above.
* Use Cases identifying the interactions between the system actors and subsystems / modules to achieve the goals of the user stories identified
* Identification of functional and non-functional requirements, with supporting rationale.
* Analysis Object Model diagram identifying the system classes and the relationships between the classes.

Prototype demonstration delivery

* Demonstrate ability of client software to successfully make a ‘round trip’ request of the server
  1. Ability to specify date and of departure flight
  2. Request from client software to database server for all flights leaving specified departure airport on specified date
  3. Retrieval of flights matching the request from server
  4. Parsing of the returned XML data and display of flights identified in the returned XML
     + Note: flight times do not need to be specified in local time for prototype
* Java source code for prototype demonstration (eclipse workspace)
  + Source code to contain comment headers for class files and methods
* Prototype object model

Test Plan delivery

* Update of functional and non-functional requirements including the verification method of each
* Identification of test cases test cases
* Traceability between each requirement and test cases
* Updated object model including test support objects

Final Delivery - Project Complete

* Demonstration of working system
  + Demonstration use case will be provided
* Updated documentation
  + RAD
  + Object Model
  + Test plan
* Verification test results
* Presentation
  + Team organization
  + Architecture & design decisions
  + Object and Dynamic Model for Use Case
  + Lessons Learned

Key Assumptions

TBD (To Be Determined)

Scoring

World Plane, Inc. (WPI) is undertaking this proof of concept development effort to evaluate the feasibility of providing a retail customer interface to its airline reservation system. WPI is contracting this effort to multiple development contractors with the intent of determining the most effective path forward and selecting one or more development contractors to proceed with the design and development of the releasable project. To judge the effectiveness of each contract developer’s efforts, the quality of each proof of concept deliverable will be scored using a point distribution defined below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Deliverable | | | Possible Score | Comment |
| Requirements Elicitation | | | 25 | Req. Analysis Doc. (RAD) |
|  | Functional Requirements | 5 |  |  |
|  | Non-functional Requirements | 5 |  |  |
|  | Completeness, Consistency, Unambiguous, Correct | 5 |  |  |
|  | Use Cases | 5 |  |  |
|  | Analysis Object Model | 5 |  |  |
| Prototype | | | 25 | Approximately 20-25% functionality |
|  | Successful Demonstration | 10 |  | ‘round trip’ as specified above |
|  | Object Model Diagram | 5 |  | Update class diagram with associations, multiplicity and navigability |
|  | Dynamic Model | 5 |  | 2 significant collaboration / sequence diagram or state chart |
|  | Verification methodology for each requirement | 5 |  | Functional and nonfunctional |
|  |  |  |  |  |
| Integration & Test Plan | | | 25 |  |
|  | Requirements updated | 5 |  | Functional & non-functional |
|  | Test plan for all requirements | 10 |  |  |
|  | 2 Detailed test cases | 10 |  |  |
|  |  |  |  |  |
| Final Delivery | | | 25 |  |
|  | Program Demonstration to class | 5 |  | Demonstrate WPI selected use case |
|  | Documentation for public methods | 5 |  | Javadoc |
|  | Updated documentation | 5 |  | Scenarios, Use Cases, UML diagrams, Test plan / procedures |
|  | Source code (eclipse workspace) | 5 |  | Well structured, documented interfaces |
|  | Verification Results | 5 |  |  |
|  | Presentation | 0 |  | Lessons Learned |
| **TOTAL** | | | **100** |  |

Acceptance

|  |  |  |
| --- | --- | --- |
| [Client name] |  | [agency information |
| Company name |  | Company name |
|  |  |  |
| Full name |  | Full name |
|  |  |  |
| Title |  | Title |
|  |  |  |
| Signature |  | Signature |
|  |  |  |
| Date |  | Date |