# POS Rental Agreement System Architecture Specification

This document describes the architecture that is to be implemented to satisfy the requirements in the POS rental system functional specification. This system is a back-end system that performs 3 basic functions: (1) providing up-to-date options for the tools to be rented, (2) validating entries from the front-end POS system, and (3) generating a rental agreement based on input from the front-end POS system.

## Provide Tool Rental Options for the front-end POS System

In order to avoid any unnecessary direct ties between the front-end system and the database there will be a simple back-end method to return the current list of tools available for rental. The tools will be returned as a JSON collection of objects where each tool object contains the following information:

***Fig 1.*** *ToolInfo object*

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| ToolCode | String | Unique ID |
| ToolType | String | General tool type |
| Brand | String | Brand name for the specific tool |
| DailyCharge | Double | Daily rental rate in USD |
| WeekdayCharge | Boolean | Charge for rental on weekdays? |
| WeekendCharge | Boolean | Charge on weekend days? |
| HolidayCharge | Boolean | Charge on holidays? |

*Note above that the “ToolType” field is listed as a simple string. The available types of tools are stored in a table with additional information about each type for maintenance and flexibility but for the purposes of this simple API the “Name” for the tool type is the only field used here to keep the logic simple for the front-end system.*

The returned collection is wrapped in a simple return object that also indicates a status code and error or informational message if needed.

***Fig 2.*** *ToolRet object*

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| Success | Boolean | Did the routine complete without error? |
| Message | String | Error or information message depending on the success / failure of the method |
| Tools | List<ToolInfo> | Collection of ToolInfo objects if the method succeeds |

The following pseudo-code shows how this method might be implemented:

public List<ToolInfo> GetAvailableTools ()

{

// Perform any necessary validation / authentication checks

// Retrieve all data from the tool code table (tool code,

// tool type, brand)

// Create a ToolInfo object for each and add it to our

// collection to return

// For each unique tool type in the above list retrieve the

// corresponding row from the tool type table / data set

// (tool type, daily charge, weekday charge, weekend

// charge, holiday charge)

// Add the additional fields to each corresponding ToolInfo

// object in our return collection

// Return the full collection or set an error status and

// return message if something went wrong

}

## Validate Rental Agreement Input Parameters

This routine will be called in-line by the method that generates the rental agreement. It is used to verify that the input parameters are in valid ranges so the user can easily correct any issues and retry.

***Fig 2.*** *ValidateRet object*

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| Success | Boolean | Valid entries? |
| Message | String | Friendly error message in case of invalid input |

private ValidateRet ValidateInput (RentalParams paramsIn)

{

// Set an error condition if # of rental days in the input

// is < 1

// Set an error condition if the discount % is < 0 or > 100

}

## Generate Rental Agreement

This method performs the main work of the module. It takes the input from the POS front-end, validates it, and generates a rental agreement using the rules from the POS system functional specification.

***Fig 3.*** *RentalAgreement object*

*TBD*

public RentalAgreement GenerateAgreement (RentalParams paramsIn)

{

// Call the Validate method on input params and return an

// error if needed

// Retrieve full ToolInfo information for the given tool

// code from the tool code table

// Retrieve charge information from the tool type table

// Calculate the due date using the checkout date from the

// input and the # of days requested

// If weekday charge is False then calculate # of weekdays

// in the rental period and subtract from the total days

// (Note that all tools currently have weekday charge as

// True, but the field is there for possible future use)

// If weekend charge is False then calculate # of weekend

// days in the rental period and subtract from the

// total days

// If holiday charge is False then retrieve all entries

// from the holiday info table. See if there are any

// matches in the date range for the rental and subtract

// any matches from the total count.

// Calculate pre-discount charge (remaining days \* rental

// charge from the table)

// Calculate discount amount (entered discount % \*

// pre-discount charge) – round half-up to cents

// Calculate final charge (pre-discount – discount)

// Fill in all fields in our return object and send it

// back to the caller

}