Museum Information Kiosk Plan

Requirements Specification and Design Document

digisoln.com

Table of Contents

Overview

XYZZY Software has decided to internally fund development of a proof of concept system for an information kiosk for the Whoop-Whoop Automotive Museum. The system is to be demonstrated to the Whoop-Whoop Automotive Museum on Friday, February 4th 2010. The following document details the requirements, software architecture and design specifications for this concept system.

Requirements Specification

To comprehensively determine the required specifications for the concept system, it is necessary to first investigate client required system functionality, and follow this with end-user case scenarios. These investigations will be developed from both visitor and administrator perspectives.

System Functionality

There are core processes which this concept system must satisfy to ensure its successful implementation. These system requirements will be investigated for visitor and administrator use.

Visitor

The fundamental use – and thus purpose – of the information kiosk system is to provide fast and reliable information to the visitor to enhance their overall experience of the museum. The improvement in the dissemination of knowledge to the visitor will rely on the inclusion of the following functionality:

Exhibition Information

- Reliable, up to date information on specific collections held;
- Detailing of iconic exhibits within collections and as standalones;
- Promotion of temporary exhibits (including future temporary exhibits);
- Exhibits on loan, their duration of loan and location of loan;
- Building location of all exhibits held in the Whoop-Whoop Automotive Museum.

Navigation of Museum

- Listing of all locations in the museum, including buildings 1 to 12, gift shop, cafe, toilets and both exits;
- Based on visitor selection, provide a simple, easy to follow set of instructions to reach their chosen destination.

Efficient Interface

- The kiosk must cater for a variety of visitors, include children, aged, and public with accessibility issues or other special needs;
- As such, the interface must be simple, intuitive and interactive, with a correct yet simple use of language, large fonts, and a consistent navigation structure with on-screen help wherever possible.

Administrator

To provide the visitor with the above list of system functionality, it is fundamental to the success of the concept system that the Administrator is able to perform the following tasks:

Update Exhibition Information

- Maintain correct descriptions and historical information of exhibit holdings
- Add new exhibits to the system, remove old or unused exhibits, and modify or moderate content in existing exhibits;
- Set locations of exhibits, based on the building layout supplied;
- Record and track on-loan exhibits, including date of loan, client information and return date of exhibits;
- Modify, add or remove promotional, temporary or advertised exhibit content.

The Administrator must further be able to set and change the current kiosk location, as the kiosks may be moved to different locations in the museum, depending on their best use as deemed by the Whoop-Whoop Automotive Museum.

Use Case Scenario

To effectively translate client requirements into software code, a use case scenario will best illustrate how the concept system will function. The following use case scenarios will be derived from both visitor and administrator perspectives.

Visitor

The following User Case Scenario are three alternate methods that are indicative of a typical and extreme visitor scenario, in order to best establish the required functionality (and limits) of the concept system.

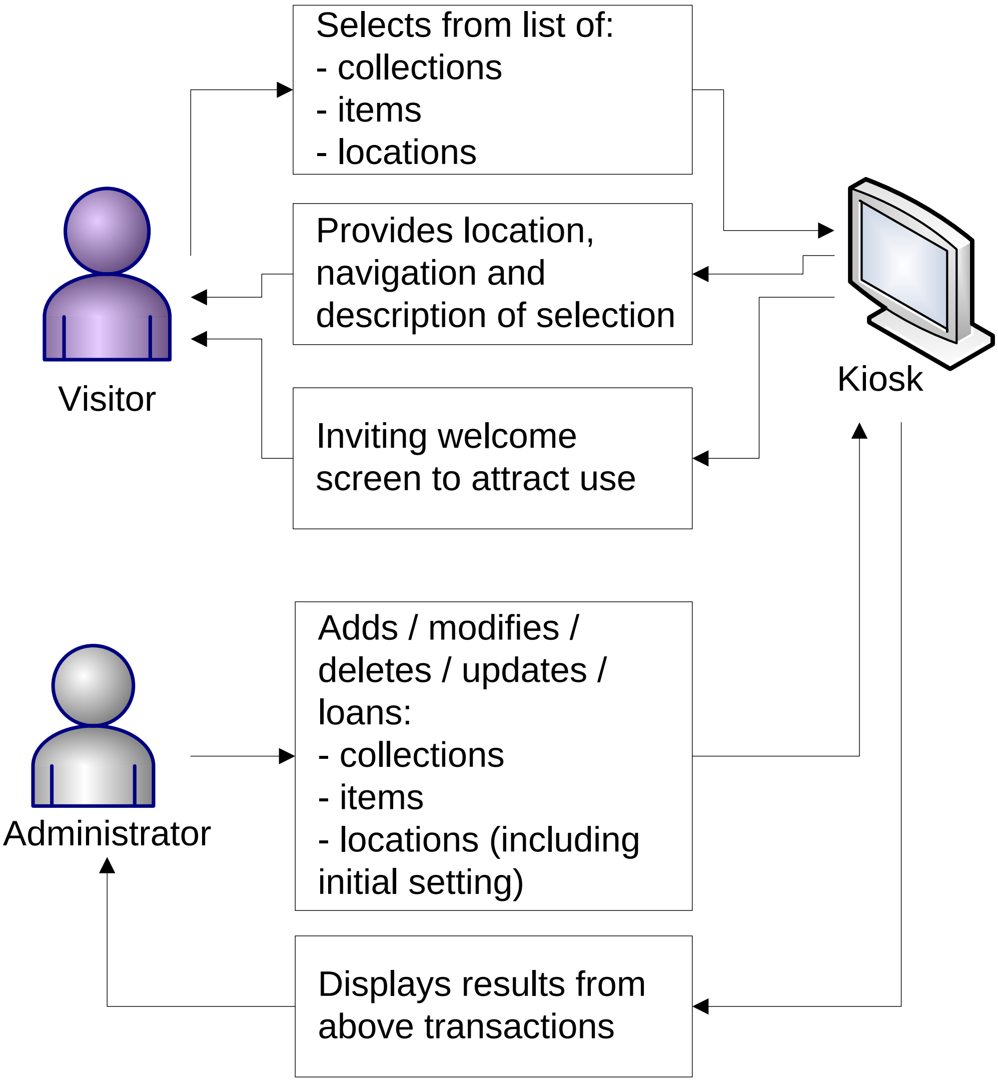
Process	**Visitor A**	**Visitor B**	**Visitor C**
1	Search Collection: Racing Cars	Find Location: Toilets	Search Exhibit: Monster Trucks
2	Read Description: Racing Cars	Search Collection: Electric Cars	With Monster Trucks:
3	Search Items: Bluebird	Find Location: Cafe	A) Read Description
4	Read Description: Bluebird	Find Location: Exit	B) Find loan museum
5	Find Location: Bluebird		C) Find return date

Administrator

The following Administrator User Case Scenario is indicative of the full process use of an Administrator scenario. This will be used in conjunction with the visitor use case scenario, as well as system functionality to requirements in order to best determine the specifications and scope of the concept system.

Process	**Administrator**
1	Set to Administrator mode [secure] – supply credentials.
2	Set present location of kiosk to Front Entry / Exit.
3	Loan Monster Truck display to Tamworth Roadshow. Set return date as 1 July 2010.
4	Delete all Space Vehicles exhibit information (including all items within this collection).
5	Add Electric Car collection. Set location to building 12.
6	Add Toyota Hybrid item to Electric Car collection.
7	Edit Bluebird exhibit description – change “disintegration speed” to 480 km/h
8	Set to Visitor mode.
9	*Physical movement of kiosk*
10	Set to Administrator mode [secure] – supply credentials.
11	Change present location of kiosk to Gift Shop.
12	Set to Visitor mode.

From these use case scenarios, it is evident that the following processes must occur for the kiosk concept system to be successful.


Selects from list of:
- collections
- items
- locations
Provides location, navigation and description of selection
Inviting welcome screen to attract use
Visitor
Kiosk
Adds / modifies / deletes / updates / loans:
- collections
- items
- locations (including initial setting)
Administrator
Displays results from above transactions

These processes are evident from the above system functionality and use case scenario investigation, and thus must now be integrated into working design documentation.

Design Documentation

The following Design Documentation utilises the results of the investigations made in the requirements specification above, in order to make detailed decisions on the software architecture, data structures and interface format of the final concept system.

Software Architecture

Given the nature of the data and processing requirements above, it can be decided that the most efficient method of implementing the kiosk concept system is to use *three-tier architecture*. Thus, the application will be comprised of the following layers / tiers:

1. Presentation Tier – which will provide an interactive interface for both visitors and administrators using predominately windows forms controls;

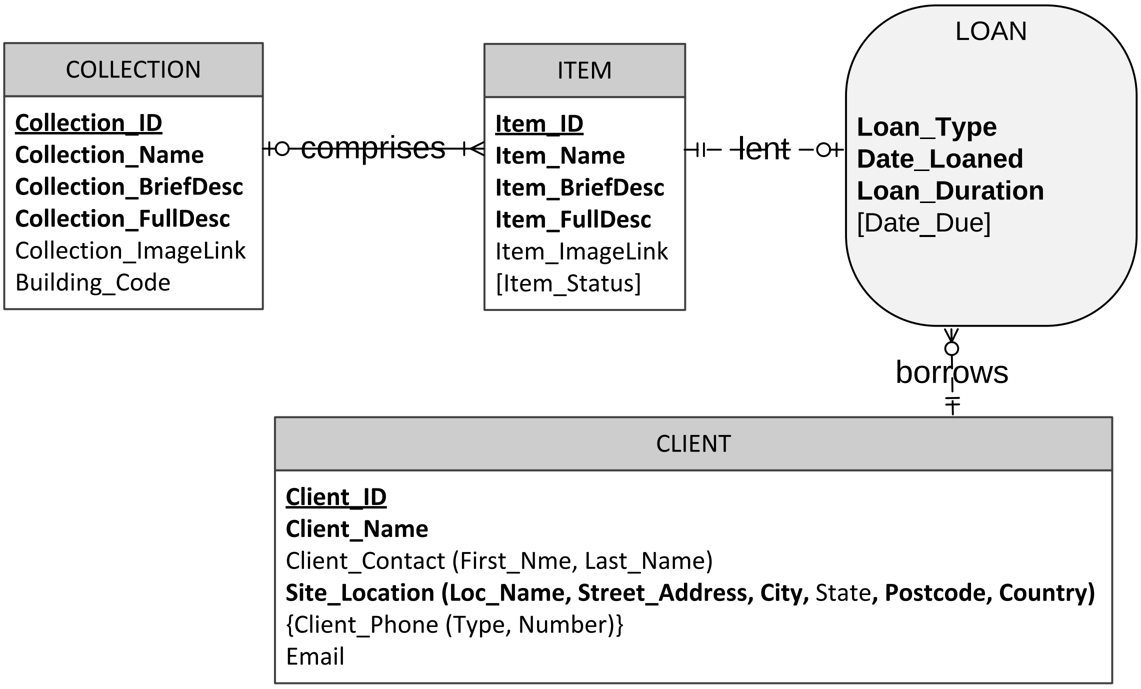
2. Logic / Application Tier – which will utilise the .NET framework to provide, extract and process data for the presentation tier, as well as interact with both stored and temporary data in the lower tier;
3. Data Tier – this will store and retrieve data required to satisfy processing requirements in the logic / application tier.

This client-server architecture allows data to be stored in a centralised location, allowing multiple kiosks (clients) to be expanded / upgraded without affecting data storage requirements. At present, the scope of this project requires only a single machine configuration – where all data storage will occur on the same database server, with which the application (individual kiosks) can interact. If the concept system was accepted in the future by a larger organisation, it is possible that a decentralised, multiple machine configuration may be required, to better load balance data resources. In this case, horizontal and vertical partitioning of the database may need to take place; however, this will not affect the current required scope of this concept system project.

Database Design

To successfully meet the data storage requirements discussed above, the following Entity Relationship Data Model will be implemented. This model is primarily the functional implementation of the *data tier* required for the three-tier application (discussed above) being developed for the Whoop-Whoop Automotive Museum:

Entity Relationship Data Model: Whoop-Whoop Automotive Museum



Entity and Attribute Definitions

The following list of entities (and their respective attributes) explains the intended representation (and use) of these entities / attributes within the Whoop-Whoop Automotive Museum data model. **Bold attributes** are required for all entities listed below:

COLLECTION: A COLLECTION is simply a set of ITEM(s). A COLLECTION is uniquely identified by its **Collection ID**. Each COLLECTION **must** contain *at minimum* **one ITEM** in order to satisfy the requirements of being a COLLECTION. More often, a COLLECTION will commonly comprise multiple ITEM(s). Finally, a COLLECTION is located in a particular spot in the museum (stored in the attribute *Building_Code*), **which makes ITEM location tracking possible.**

Collection_ID	Uniquely identifies any particular COLLECTION.
Collection_Name	The name of the COLLECTION.
Collection_BriefDesc	A brief description of the COLLECTION – for use as “attention grabbing” captions and brief blurbs as required.
Collection_FullDesc	A full description of the COLLECTION – for visitors who wish to read more specific and detailed information regarding a COLLECTION.
Collection_ImageLink	A file image location link of the particular COLLECTION, used for a visual point of reference (and aesthetic balance with text). This also has the benefit of allowing the uploading of new digital images when new COLLECTION(s) are assembled.
Building_Code	Most importantly, visitors may wish to locate either a COLLECTION or an ITEM (as a COLLECTION is at minimum a set of one ITEM(s)). Note this attribute is not mandatory – an ITEM or COLLECTION may not be on display (located in storage, or as part of an incoming or outgoing LOAN – see below).

ITEM: An ITEM is a single artefact or object that exists within the Whoop-Whoop Automotive Museum. All ITEM(s) – whether on display, borrowed, on loan or in storage – must be recorded here. An ITEM is uniquely identified by an assigned ID (Item_ID). Note that an ITEM can only be part of **one** COLLECTION at any given time – logically, an ITEM cannot be in two places at once. Finally, it is evident that one individual ITEM can only be on LOAN once at a time – as the same single artefact cannot be on LOAN to two different museums at once. When an ITEM is on display in the museum, it is assigned a COLLECTION – and thus will be locatable by the COLLECTION *Building_Code* attribute.

Item_ID	The Item ID uniquely identifies any particular ITEM record in the database.
Item_Name	The title of the ITEM.
Item_BriefDesc	A brief description of the ITEM – for use as captions when displaying search results, etc. Primarily to increase visitor interest and invite further reading.

Item_FullDesc	A full, longer description of the ITEM – containing detailed historical facts and dates, etc.
Item_ImageLink	A file image location link of the particular ITEM, used for a visual point of reference (and aesthetic balance with text). This also has the benefit of allowing the uploading of new digital images when new ITEM(s) arrive.
Item_Status	This is derived based on three types of possible ITEM status: a) The ITEM is "*on display*" in the museum, in which case it **must** be assigned to a COLLECTION (and thus allocated a Building_Code); b) The ITEM is in "*storage*", in which case it will **not exist** within any record of a COLLECTION; c) The ITEM is "*on loan*" (to anther museum) or "*temporarily borrowed*" (from another museum), in which case the item will appear as a record within the LOAN entity (with differing Loan Types). Loans *to* and *from* the museum will be further distinguished below. In all cases, it is evident that *Item_Status* does not need to be recorded – as it can be derived through one of these three cases.

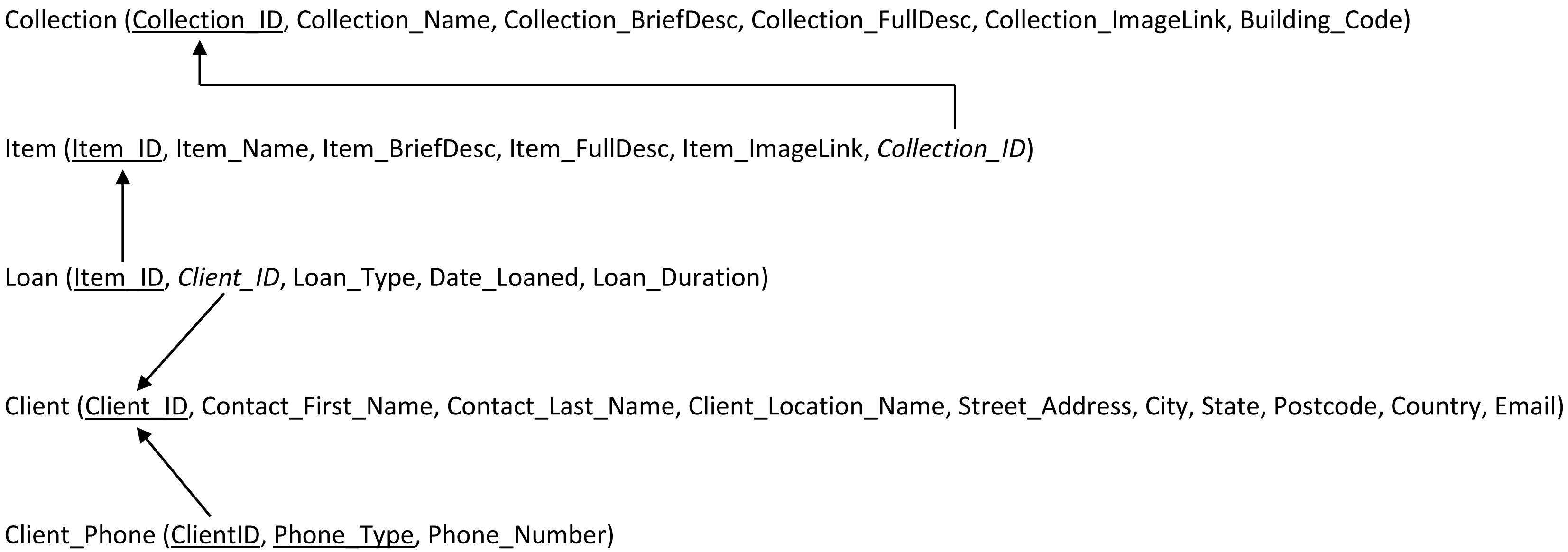
<table>
<tr><th colspan="2"><b>LOAN</b>: A LOAN is an agreement between Whoop-Whoop Automotive Museum and a CLIENT for either an <i>incoming</i> or <i>outgoing</i> ITEM. A LOAN is an associative attribute which represents the relationship between the CLIENT and ITEM; thus, although both outer entities are required (<b>Item_ID</b> and <b>Client_ID</b>), it is evident that each ITEM can only be on LOAN once and once only. An ITEM_ID can therefore identify any LOAN record; furthermore, a LOAN record can identify associated CLIENT information – as well as information regarding type, time and duration of the LOAN. <b>Once a LOAN is finished the record is removed</b> so the ITEM can be used in a LOAN agreement again.</th></tr>
<tr><td><b>Loan_Type</b></td><td>The loan may either be:
a) "<i>incoming</i>" – meaning the Whoop-Whoop Automotive Museum has requested an ITEM from another CLIENT;
b) "<i>outgoing</i>" – meaning an ITEM from the Whoop-Whoop Automotive Museum has been loaned to another CLIENT.
In either event, the remaining attributes for the LOAN entity are recorded using the same approach regardless of an <i>incoming</i> or <i>outgoing Loan_Type</i>.</td></tr>
<tr><td><b>Date_Loaned</b></td><td>The date the ITEM is loaned between the CLIENT and the Whoop-Whoop Automotive Museum.</td></tr>
<tr><td><b>Loan_Duration</b></td><td>The length of time agreed for the LOAN from both parties.</td></tr>
<tr><td>Date_Due</td><td>Derived using the formula Date Due = Date Loaned + Loan Duration</td></tr>
</table>

CLIENT: A CLIENT is a person that the Whoop-Whoop Automotive Museum is borrowing or lending an ITEM with. Note that a CLIENT may borrow or lend multiple items. A CLIENT record may be stored without actually currently borrowing or loaning a current ITEM – this is for legal, marketing and other follow-up purposes.	
Client_ID	Uniquely identifies each client.
Client_Name	The business / organisation / event name of the client borrowing / lending an ITEM.
Client_Contact	The full name of the client's designated contact person (if appropriate / necessary). This is a composite attribute, made up of both the client's first and last names.
Site_Location	The Site_Location is *the physical site where the ITEM will be borrowed from or lent to* – where the visitor can actually view the ITEM in existence. This is a composite attribute made up of the **Loc_Name** (i.e. location name, for example Canberra War Memorial), the site **Street Address** (e.g. 10 Parliament Street), **City** (e.g. Canberra), State (e.g. ACT – but only in countries where applicable, for example Australia or USA), **Postcode** (e.g. 4740) and **Country** (e.g. Australia).
Client_Phone	A *possible* phone contact of the client. If this exists – which is more than likely – this composite / multi-valued attribute will include the Client's phone **Type** (e.g. business, personal / mobile, etc) and actual phone **Number**.
Email	Clients *may* choose to supply a valid email account to maintain correspondence.

Relational Model

The following set of relations provides the Relational Model for the Whoop-Whoop Automotive Museum (proceeding page). These relations have been modelled from the Entity Relationship model solution shown above. The following relations will be used to create the logical structure of the database required for the storage requirements for the Whoop-Whoop Automotive Museum. As per industry standard, the following set of relations have already been normalised to comply with Third Normal Form:

Table Relations: Whoop-Whoop Automotive Museum



Collection (Collection_ID, Collection_Name, Collection_BriefDesc, Collection_FullDesc, Collection_ImageLink, Building_Code)

Item (Item_ID, Item_Name, Item_BriefDesc, Item_FullDesc, Item_ImageLink, *Collection_ID*)

Loan (Item_ID, *Client_ID*, Loan_Type, Date_Loaned, Loan_Duration)

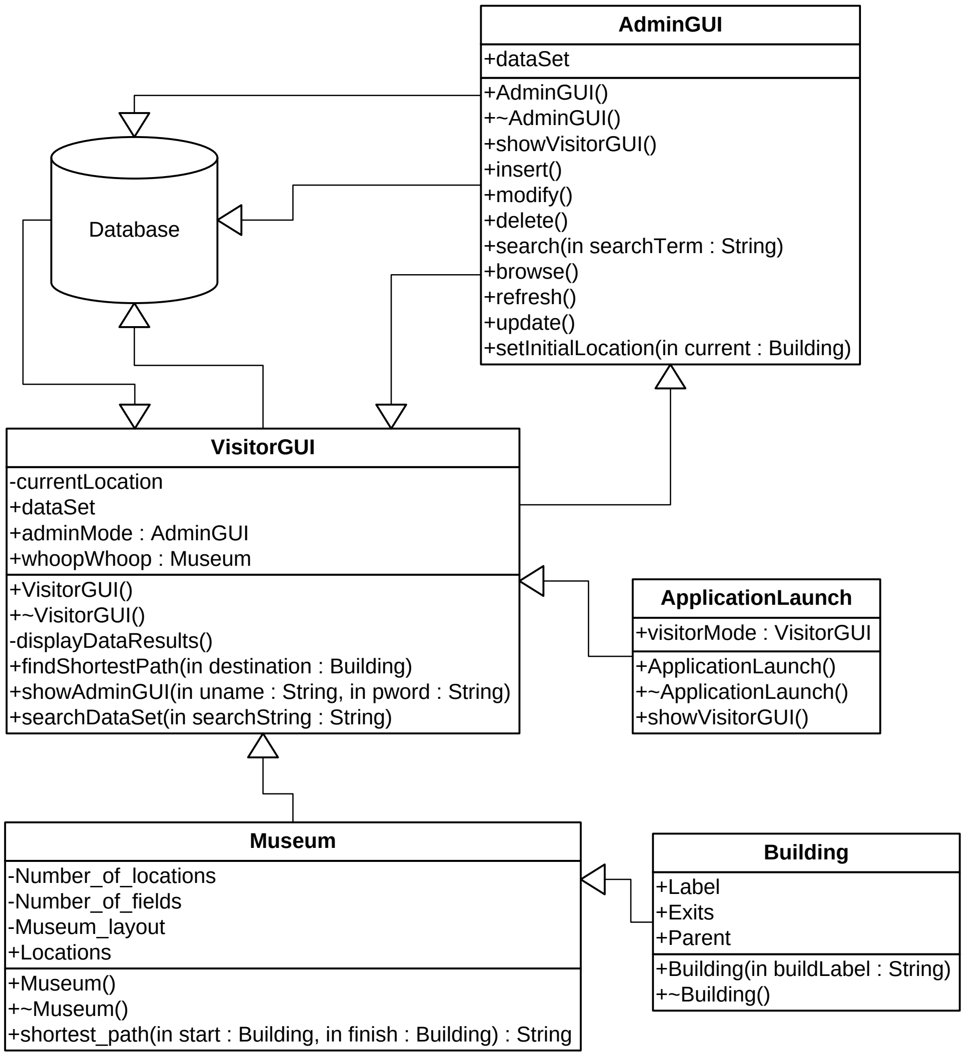
Client (Client_ID, Contact_First_Name, Contact_Last_Name, Client_Location_Name, Street_Address, City, State, Postcode, Country, Email)

Client_Phone (ClientID, Phone_Type, Phone_Number)

Class Design

To successfully implement the *Logic / Application Tier* of the three-tier architecture (discussed above), a seamless link must be established between both the Data tier and the Interface tier with which the Logic / Application tier interacts. This will be established using Object Oriented Programming techniques within a .NET environment. Each functionality of the logic / application tier will thus be handled by instantiated classes, with both private and public attributes and member functions. This allows for more manageable programming within a group environment, as well as safer programming with secure interaction of the class's attributes and a more scalar design in the long-term depending on testing and client feedback.

Preliminary investigations have identified *at minimum* the following classes are necessary for the successful implementation of the Whoop-Whoop Automotive Museum:



The diagram above illustrates initial class inferences that have been ideated during specification stage. Given the integrated development environment will handle variant proportions of the *GUI* and *launch* structures (including data interfacing), it is therefore necessary (and warranted) to further define both the **Building** and **Museum classes** for project transparency at this stage.

Building Class

Member	Explanation
Public Class Attributes (Name : Data Type):	
Label : string	Label / name of the BUILDING, e.g. “Cafe”
Exits: hashtable	Exits *hashtable* will pair exit building label(s) with their associated hash values. This is useful for a quicker retrieval of the BUILDING(s) exits.
Parent : BUILDING	Parent will be a *handle* (reference pointer) to another BUILDING class variable instantiation. This will record the address of the previous building entered **when calculating the quickest path** between two points in the museum.
Public Constructor / Deconstructor (required argument Data Type):	
BUILDING (string buildLabel)	The constructor must at minimum set a label for the BUILDING, and allocate new dynamic memory / instantiate values of all members where necessary (e.g. exits and parent handle).
~BUILDING ()	This must “free up” all dynamically allocated memory for this BUILDING instance by deleting each of the member variables.
Public Class Operations (required argument Datatype):	
add_exit (BUILDING other)	This will add a value to the current instance of the BUILDING Exits *hashtable* – the value added will be the *label* of the *other* BUILDING. This will often be used to store multiple exits.

Museum Class

Member	Explanation
Private Class Attributes (Name : Data Type):	
Number_of_locations : static integer	Specifies the number count of *distinct locations* within the MUSEUM. This will be used to store the first dimension list limits of the *Museum_layout* array below.
Number_of_fields : static integer	Stores number of fields to be counted in the second dimension (sub-list) of the *Museum_layout* array below.
Museum_layout : static two	A static (unchanging) two-dimensional array of strings - each

dimensional array of string ([Number_of_locations] * [Number_of_fields])	listed element (to the value of *Number_of_locations*) will be a separate location within the entire museum. Within this list, the first sub-element will be the actual location, with the remaining sub-elements (to the value of *Number_of_fields*) constituting the exits. The locations and exits will correspond to the BUILDING Class label(s) – hence the need for this [BUILDING – Label] property to remain public. As an example, if the toilets had *exits* to both the **cafe** and **gift shop**, this array may constitute one sub-element such as: { "Toilets", "**Cafe**", "**Gift Shop**", "", "" }, ... next BUILDING ... etc.
Public Class Attributes (Name : Data Type):	
Locations: hashtable	This will pair both the location and exit strings from the private *Museum_layout* array (an example of these strings is given above in the member itself) with their associated hash values for faster retrieval.
Public Constructor / Deconstructor (required argument Data Type):	
MUSEUM ()	This constructor must allocate memory for the locations hashtable and populate it using an *optimal* algorithm such as: FOR each location in the *Museum_layout* table CREATE a new instance of BUILDING RECORD this new instance in the *Locations* hashtable FOR each exit within the current location (i.e. first in the list) CALL the *add_exit* method with the new instance of BUILDING END FOR END FOR
~MUSEUM ()	This must "free up" all dynamically allocated memory for this MUSEUM instance by deleting the locations hashtable.
Public Class Operations (required argument Data Type) returns Data Type:	
shortest_path (BUILDING start, BUILDING finish) **returns** string	Finds the shortest route between any given BUILDING "start" AND BUILDING "finish" within the museum – and **returns** this route as a string. As the kiosks could vary in location and distribution, BUILDING "start" may need to be changed; this method must accommodate this client need. A **breadth first search** will return the shortest path between these two BUILDING(s), and will be used to loosely prescribe the algorithm to follow for this method (as follows):

```
Initially, to save time on unnecessary searches:

IF start = finish
  Return “already there”
END IF

BUILDINGS Parent property will keep track of buildings to pass:

FOR all BUILDINGS
 SET BUILDING Parent property to NULL
END FOR

The following breadth first search finds shortest route to finish:

Enqueue start
WHILE BUILDINGS in queue
 Dequeue a_location
  If we are there:
  IF a_location = finish THEN exit this loop / search
  Else
     Search all of the current exits first – don’t just take one:
     FOR all of a_location Exits
         If the exits have not been visited yet, enqueue them:
         IF a_location Exits Parent property = NULL
              Enqueue a_location Exit
              Building will be visited, also to retrace our steps:
              SET a_location Exit Parent property TO a_location
         END IF
     END FOR
  END IF
END WHILE

To produce a string detailing the path of buildings to pass:

WHILE a_location != start
  APPEND a_location Label to a path_to_follow string
  SET a_location to a_location Parent
END WHILE

Return path_to_follow
```

Finally, in terms of class specification, it must be again reiterated that testing and software limitations (as identified in the risk assessment document) may affect the structure of the above defined classes during implementation. It is important to note that if this is to occur, the overall functionality of what the client is expecting does not change; rather it remains strictly adhered to.

User Interface Design

To best design a user interface, the end-user audience must be thoroughly understood. The information kiosk will predominantly be used by visitors to the museum, of which may range disproportionately between the following audiences:

- school children, possibly with speech / recognition difficulties (ages 8-17);
- tertiary students (ages 18-25) and enthusiasts (ages 35+) seeking more in-depth content;
- elderly (aged 60+) with sight, reading, hearing and other accessibility difficulties;

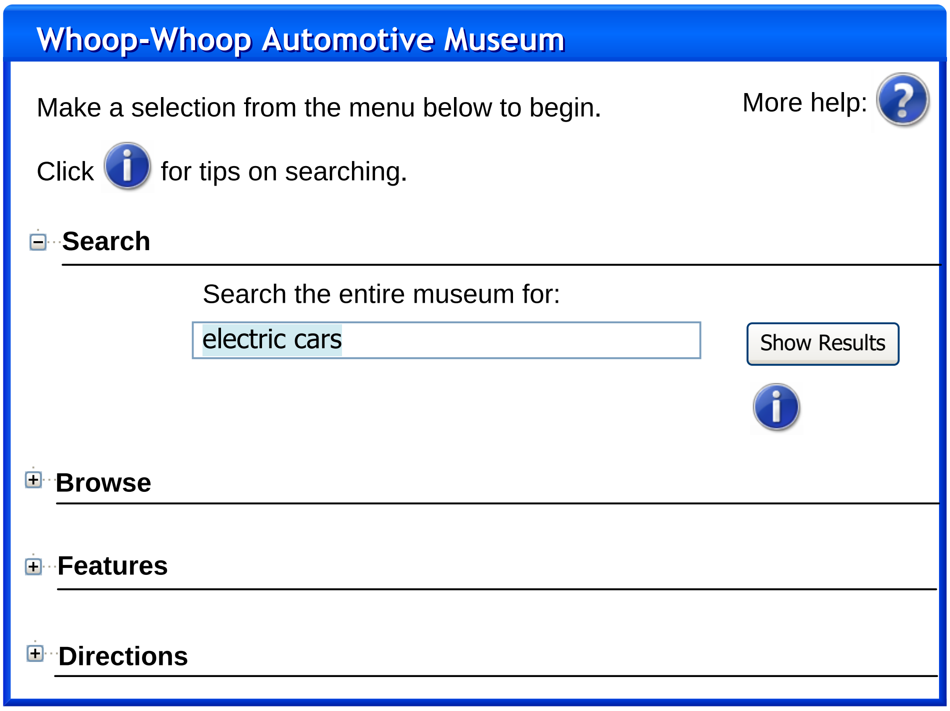
This is an incredibly broad age demographic and it is clear that the targeted end user may be techno-savvy or non-techno savvy, all of which will be seeking a different experience.

For the administrator back-end, the Whoop-Whoop Automotive staff operating the system will be trained in the use of the kiosk. It is within the allowance of this project to expect that this staff will have a basic competence operating Windows forms, controls and conducting basic processes such as saving, securing logging in details and exiting without making changes.

With this in mind, as well as accessibility and other legal and aesthetic and ergonomic considerations, the following user interface designs will be adhered to in producing the final concept system. Some aspects of these designs may change depending on feedback from end user testing.

Visitor

Initially, the following form will be launched. Visitors can click on either the "+" or the selection term itself (**Search**, **Browse**, **Features**, **Directions**) to expand the choices within. **Search** will be initially expanded:



Whoop-Whoop Automotive Museum

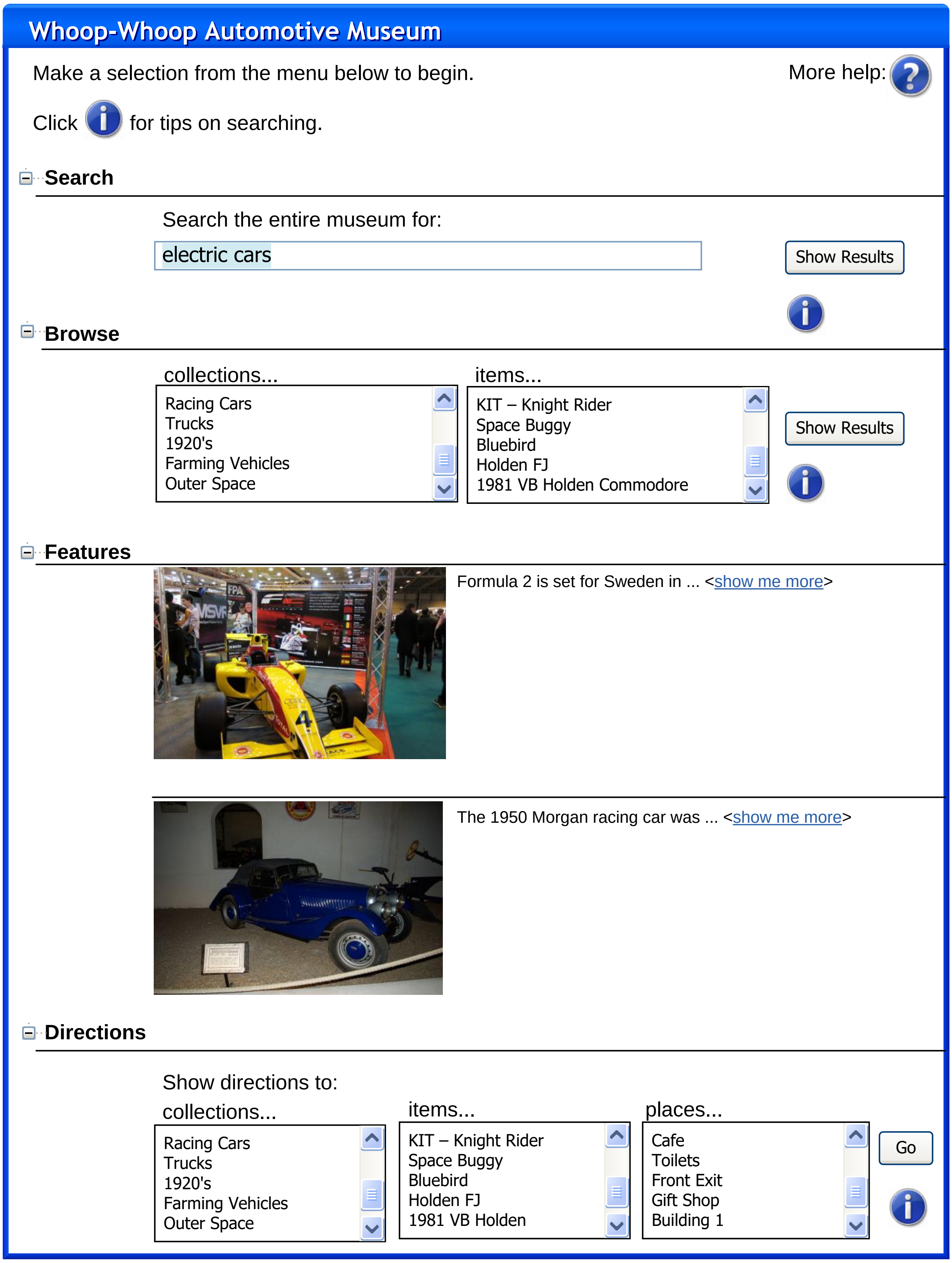
Make a selection from the menu below to begin.

Click ⓘ for tips on searching.

Search the entire museum for:

electric cars

The **fully expanded** start form is shown below – *which shows the necessity for having collapsible / uncluttered layouts*:



Whoop-Whoop Automotive Museum

Make a selection from the menu below to begin.

More help: ?

Click ⓘ for tips on searching.

⊟ **Search**

Search the entire museum for:

electric cars

⊟ **Browse**

collections...

- Racing Cars
- Trucks
- 1920's
- Farming Vehicles
- Outer Space

items...

- KIT – Knight Rider
- Space Buggy
- Bluebird
- Holden FJ
- 1981 VB Holden Commodore

⊟ **Features**

Formula 2 is set for Sweden in ... <show me more>

The 1950 Morgan racing car was ... <show me more>

⊟ **Directions**

Show directions to:

collections...

- Racing Cars
- Trucks
- 1920's
- Farming Vehicles
- Outer Space

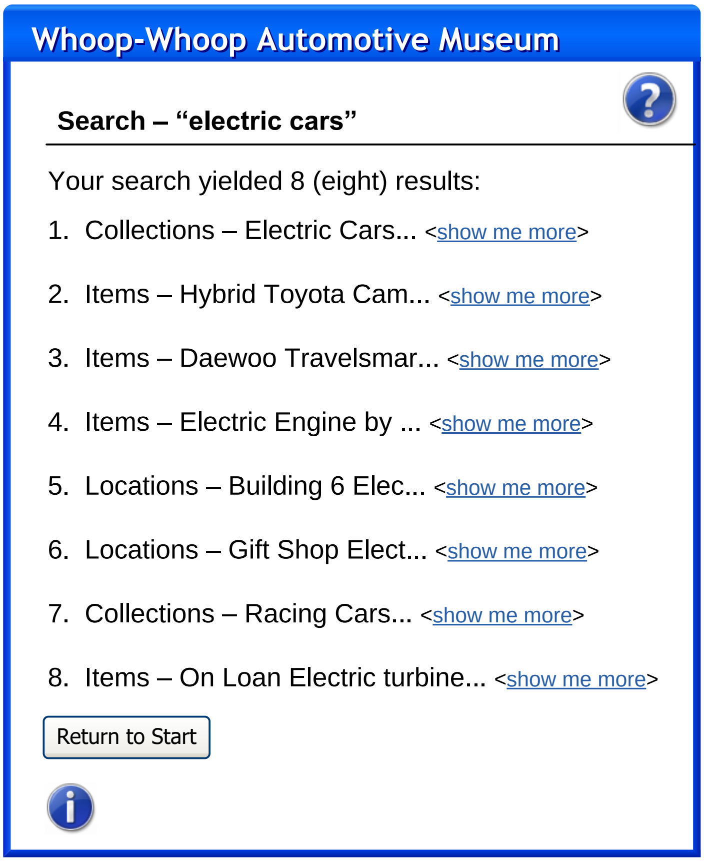
items...

- KIT – Knight Rider
- Space Buggy
- Bluebird
- Holden FJ
- 1981 VB Holden

places...

- Cafe
- Toilets
- Front Exit
- Gift Shop
- Building 1

Presuming a *collection* is searched – the following is an illustration of the results. Of all the currently designed interfaces, this may warrant the inclusion of graphical content returned in search results:



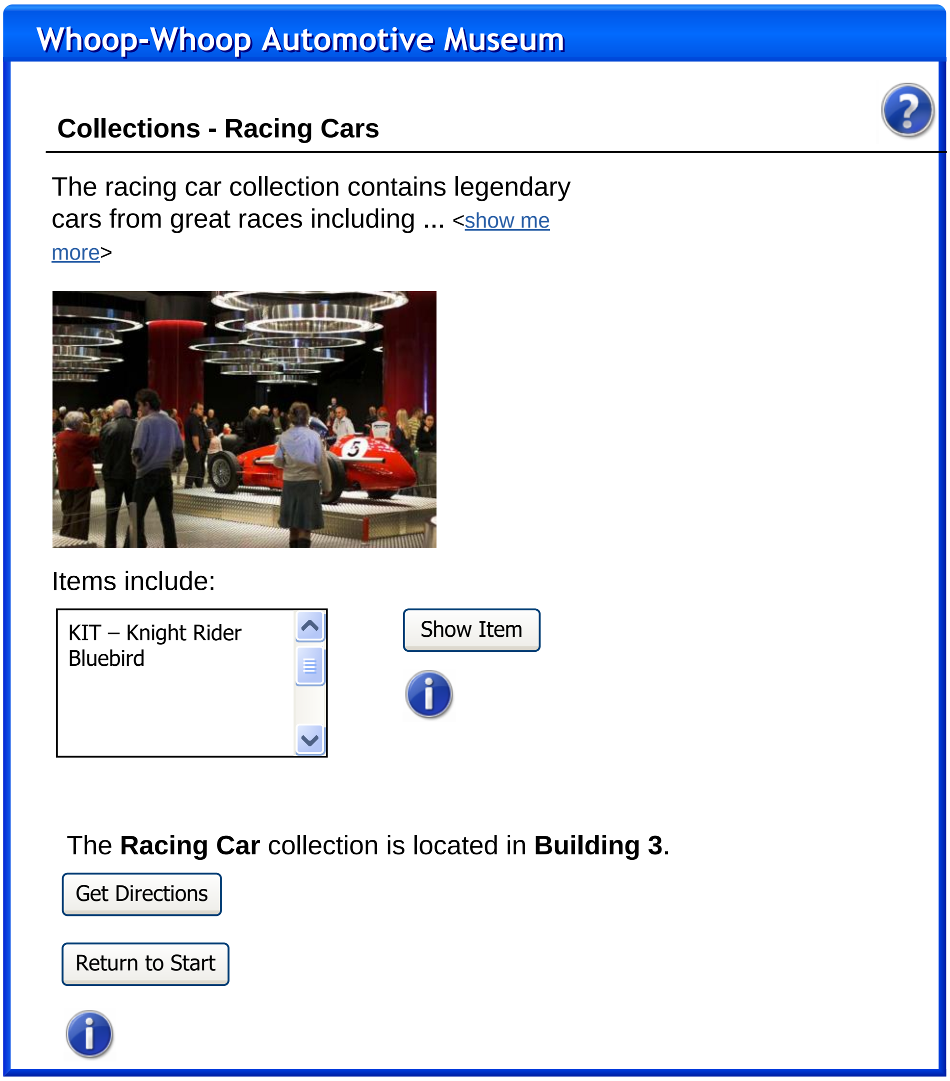
Whoop-Whoop Automotive Museum

Search – "electric cars"

Your search yielded 8 (eight) results:

1. Collections – Electric Cars... <show me more>
2. Items – Hybrid Toyota Cam... <show me more>
3. Items – Daewoo Travelsmar... <show me more>
4. Items – Electric Engine by ... <show me more>
5. Locations – Building 6 Elec... <show me more>
6. Locations – Gift Shop Elect... <show me more>
7. Collections – Racing Cars... <show me more>
8. Items – On Loan Electric turbine... <show me more>

The following form is indicative of displayed results when accessing a collection. Options exist to browse items and obtain directions:



Whoop-Whoop Automotive Museum

Collections - Racing Cars

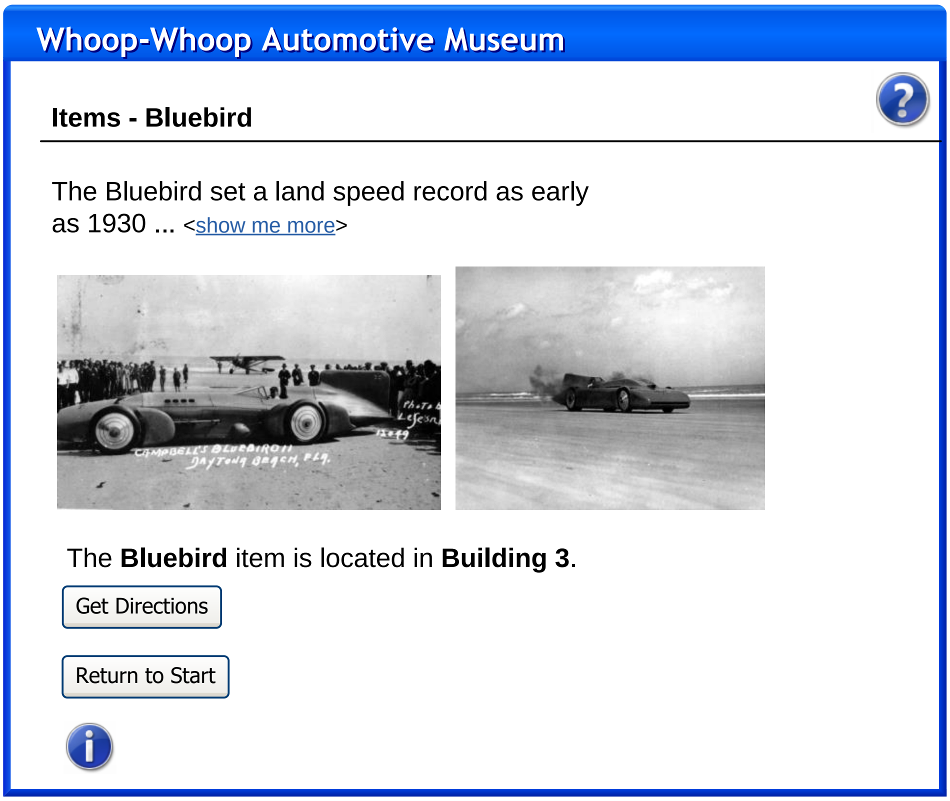
The racing car collection contains legendary cars from great races including ... <show me more>

Items include:

KIT – Knight Rider
Bluebird

The **Racing Car** collection is located in **Building 3**.

Viewing an item within a collection:



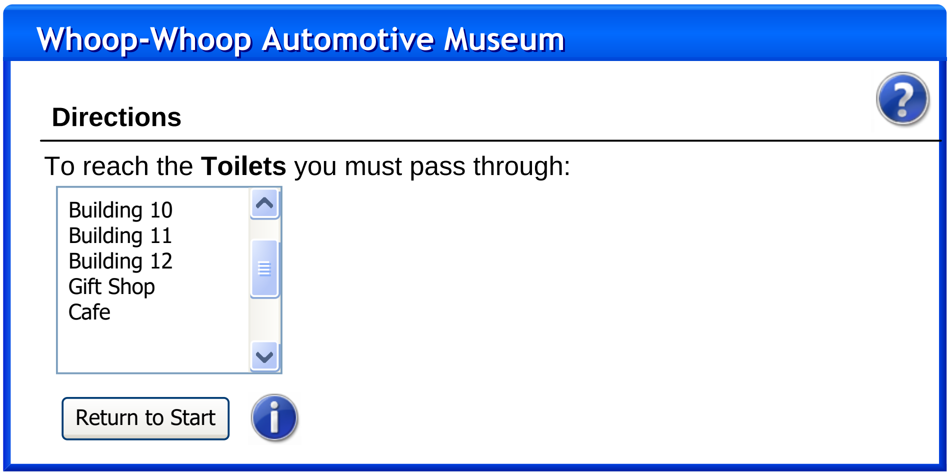
Whoop-Whoop Automotive Museum

Items - Bluebird

The Bluebird set a land speed record as early as 1930 ... <show me more>

The **Bluebird** item is located in **Building 3**.

Accessing directions to an item or location:



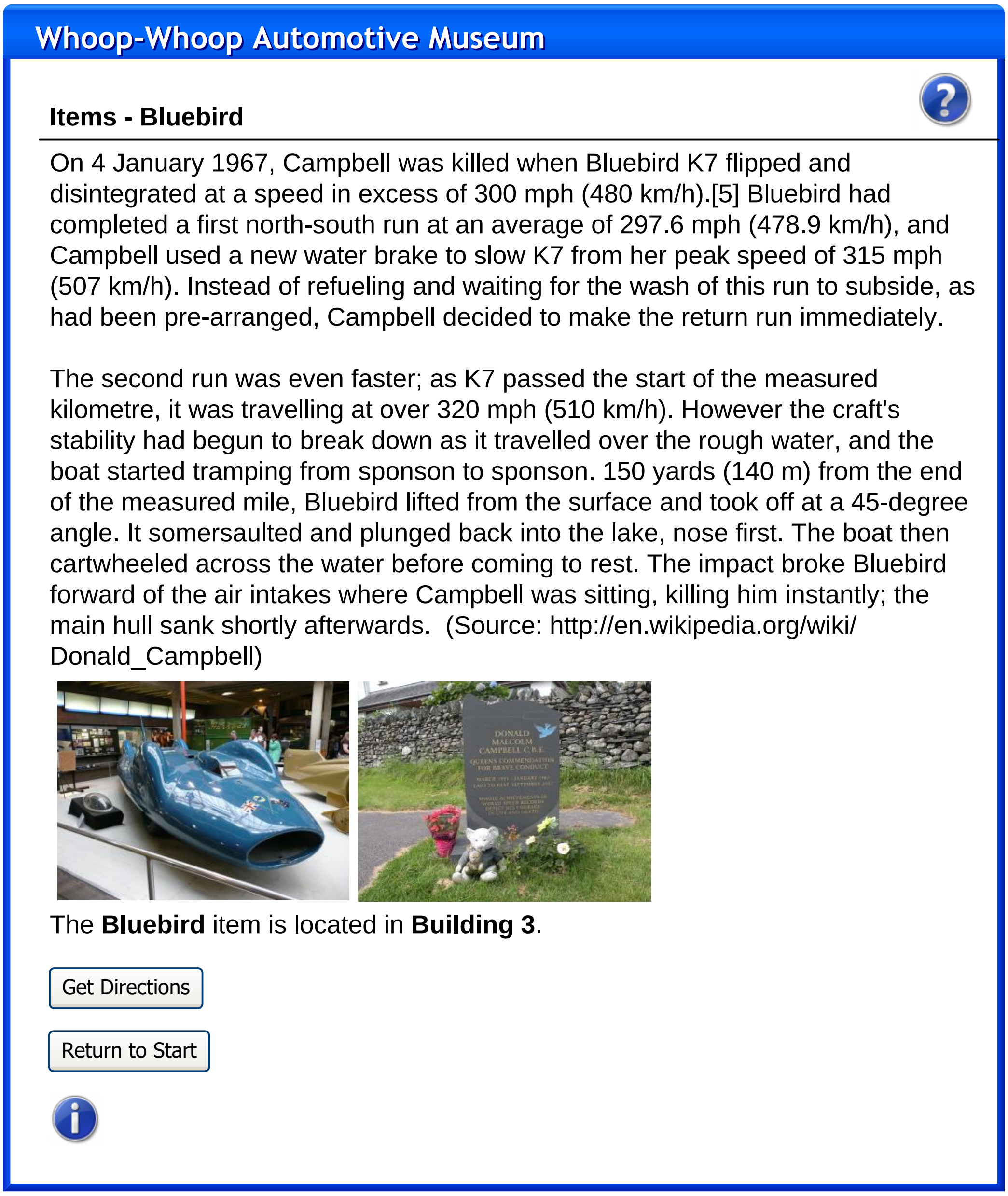
Whoop-Whoop Automotive Museum

Directions

To reach the **Toilets** you must pass through:

Building 10
Building 11
Building 12
Gift Shop
Cafe

Finally, detailed information of an item / collection / location (if necessary):



Items - Bluebird

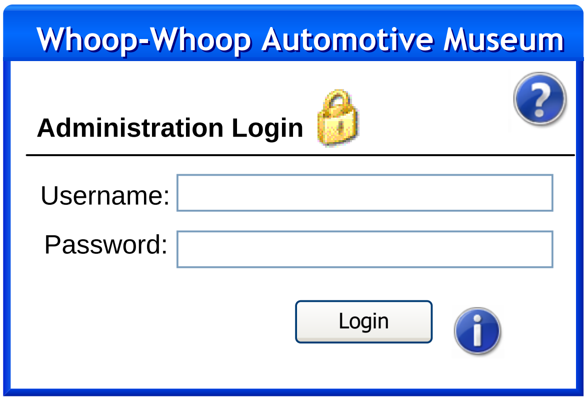
On 4 January 1967, Campbell was killed when Bluebird K7 flipped and disintegrated at a speed in excess of 300 mph (480 km/h).[5] Bluebird had completed a first north-south run at an average of 297.6 mph (478.9 km/h), and Campbell used a new water brake to slow K7 from her peak speed of 315 mph (507 km/h). Instead of refueling and waiting for the wash of this run to subside, as had been pre-arranged, Campbell decided to make the return run immediately.

The second run was even faster; as K7 passed the start of the measured kilometre, it was travelling at over 320 mph (510 km/h). However the craft's stability had begun to break down as it travelled over the rough water, and the boat started tramping from sponson to sponson. 150 yards (140 m) from the end of the measured mile, Bluebird lifted from the surface and took off at a 45-degree angle. It somersaulted and plunged back into the lake, nose first. The boat then cartwheeled across the water before coming to rest. The impact broke Bluebird forward of the air intakes where Campbell was sitting, killing him instantly; the main hull sank shortly afterwards. (Source: http://en.wikipedia.org/wiki/Donald_Campbell)

The **Bluebird** item is located in **Building 3**.

Administrator

Initially, the Administrator must supply secure credentials:

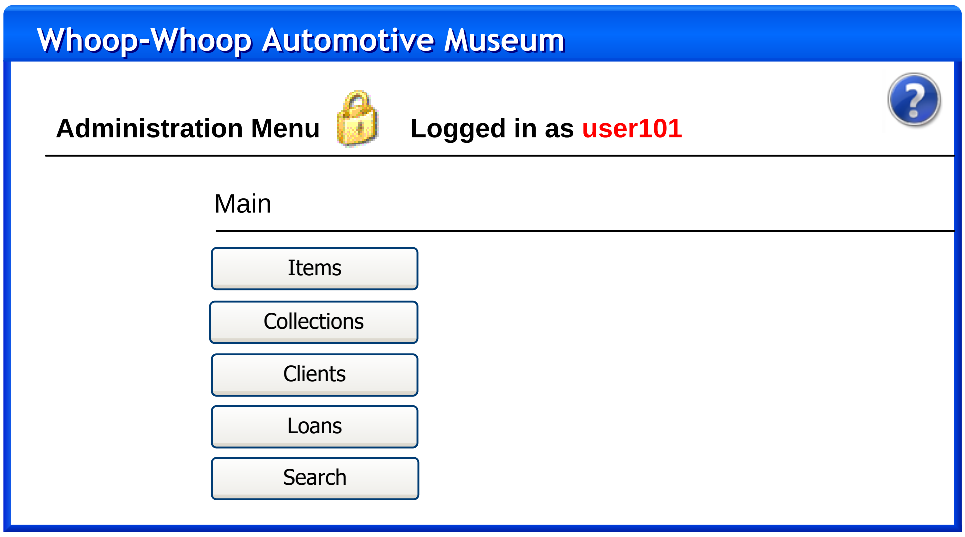


Administration Login

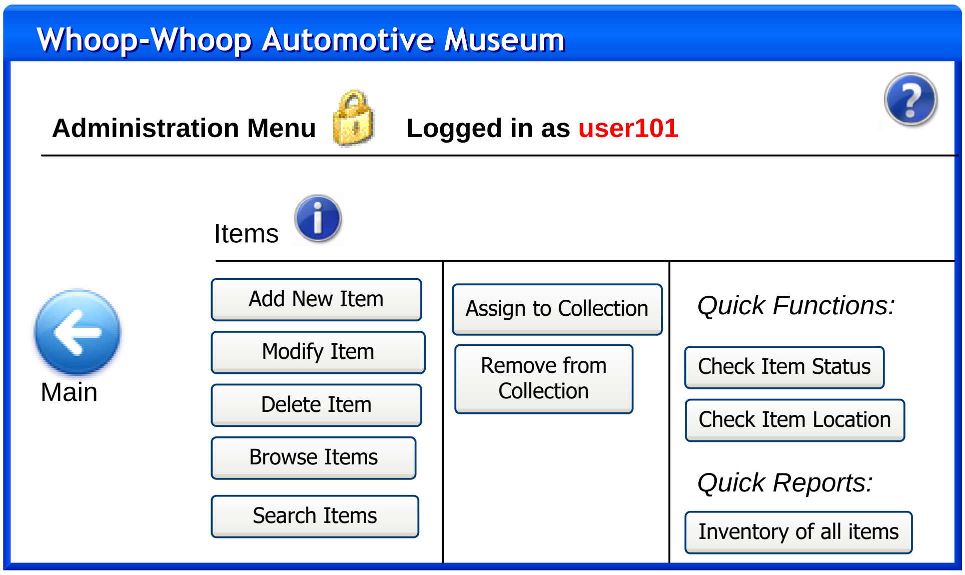
Username:

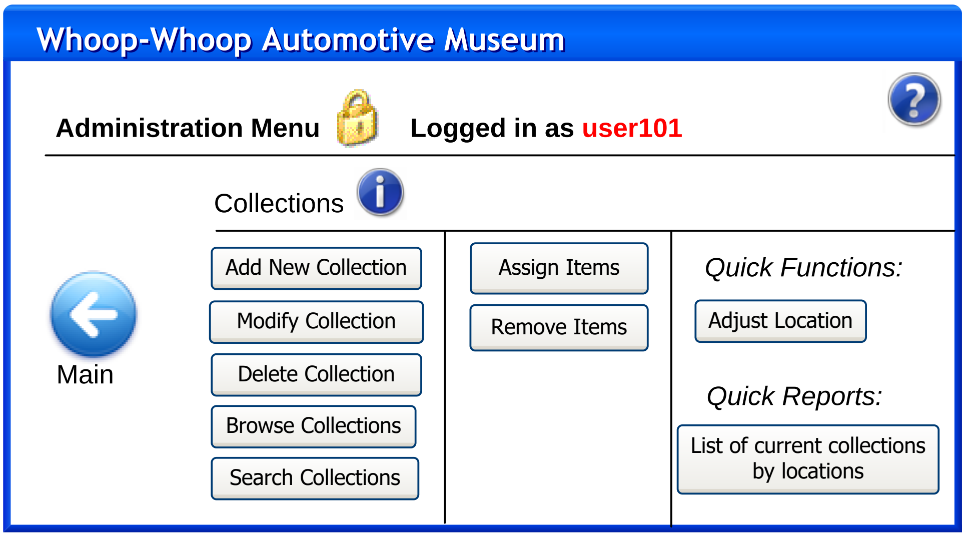
Password:

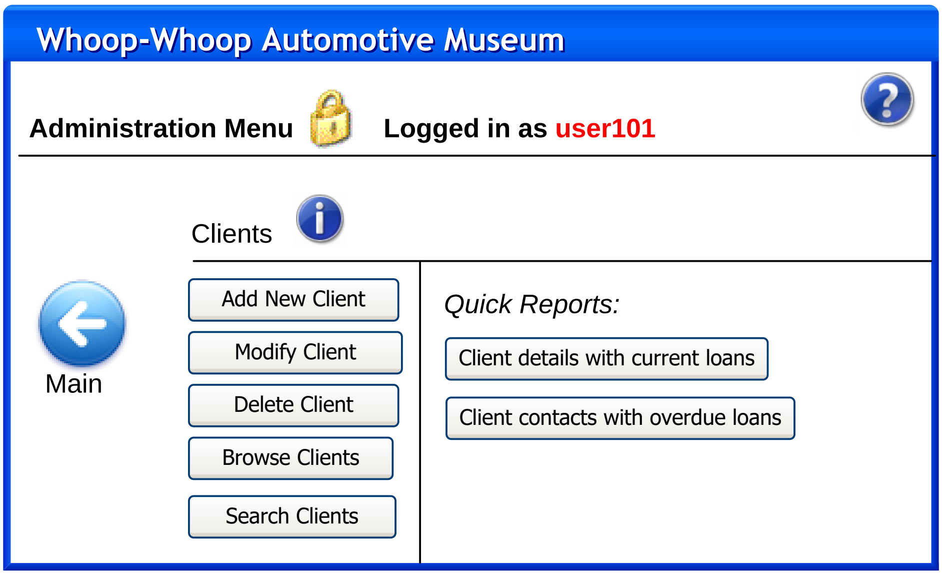
The screen the Administrator is greeted will be the Main control panel. Each sub panel must have a link back to this screen.

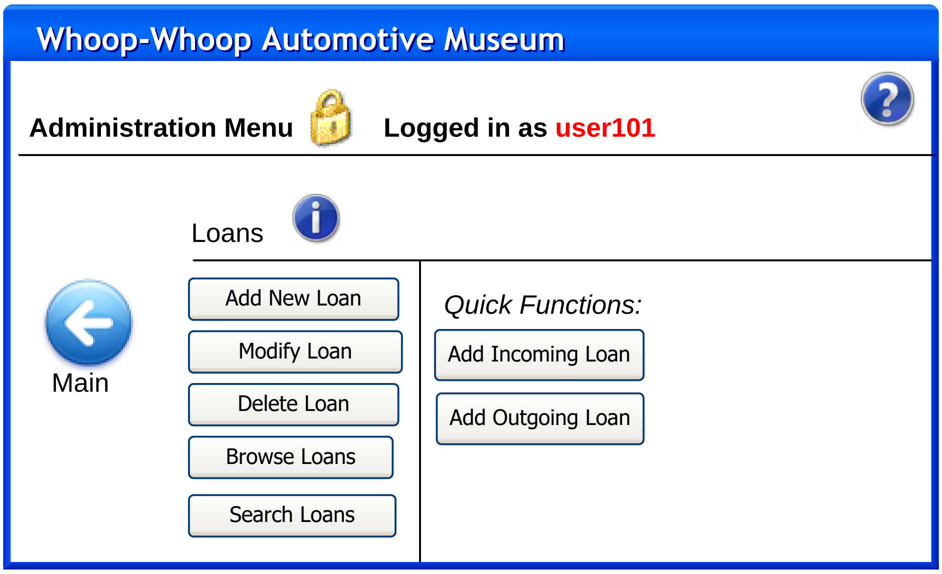


Each of the sub functions classified here will have their own sub panel, as shown below. The *quick functions* and *quick reports* facilities will be added, and based on testing, may be customized depending on **most used**. This will inherently cut down on time spent navigating panels, achieving a greater work flow:

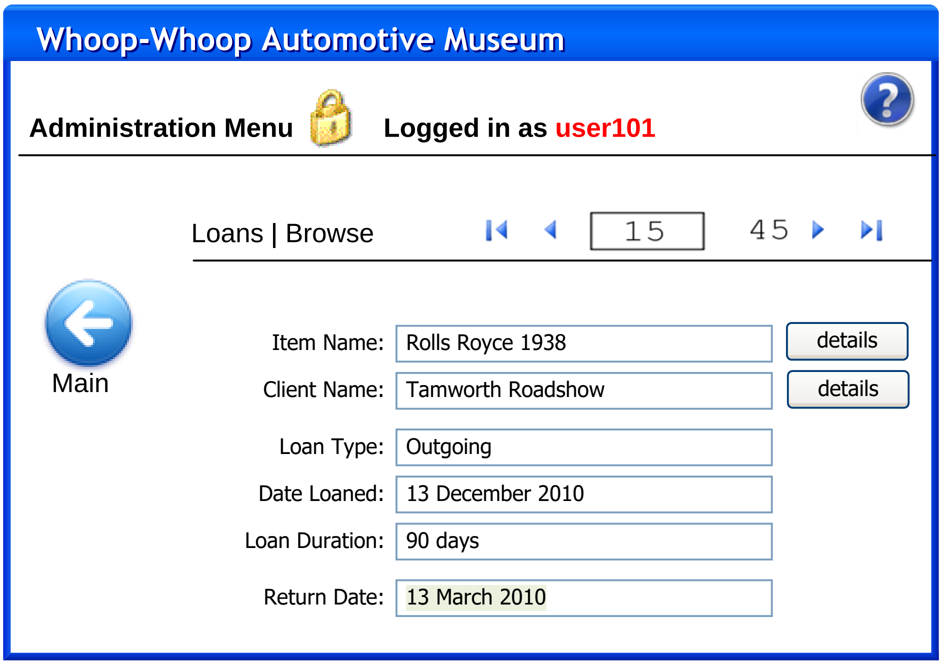



Whoop-Whoop Automotive Museum
Administration Menu
Logged in as user101
Collections
Main
Add New Collection
Modify Collection
Delete Collection
Browse Collections
Search Collections
Assign Items
Remove Items
Quick Functions:
Adjust Location
Quick Reports:
List of current collections by locations


Whoop-Whoop Automotive Museum
Administration Menu
Logged in as user101
Clients
Main
Add New Client
Modify Client
Delete Client
Browse Clients
Search Clients
Quick Reports:
Client details with current loans
Client contacts with overdue loans


Whoop-Whoop Automotive Museum
Administration Menu
Logged in as user101
Loans
Main
Add New Loan
Modify Loan
Delete Loan
Browse Loans
Search Loans
Quick Functions:
Add Incoming Loan
Add Outgoing Loan

Effectively, from these sub-panels, each will exhibit the following panels, thus only a sample has been completed for the Loans panel. The process will effectively be identical for each sub-panel function. Note that when updating / modifying a record, this will follow the process of using the browsing panel (shown below) to select the required record, which will open in the add panel (shown thereafter) with details already entered into the required fields. The user will simply make modifications and click save.



Whoop-Whoop Automotive Museum

Administration Menu Logged in as user101

Loans | Browse 15 45

Item Name: Rolls Royce 1938

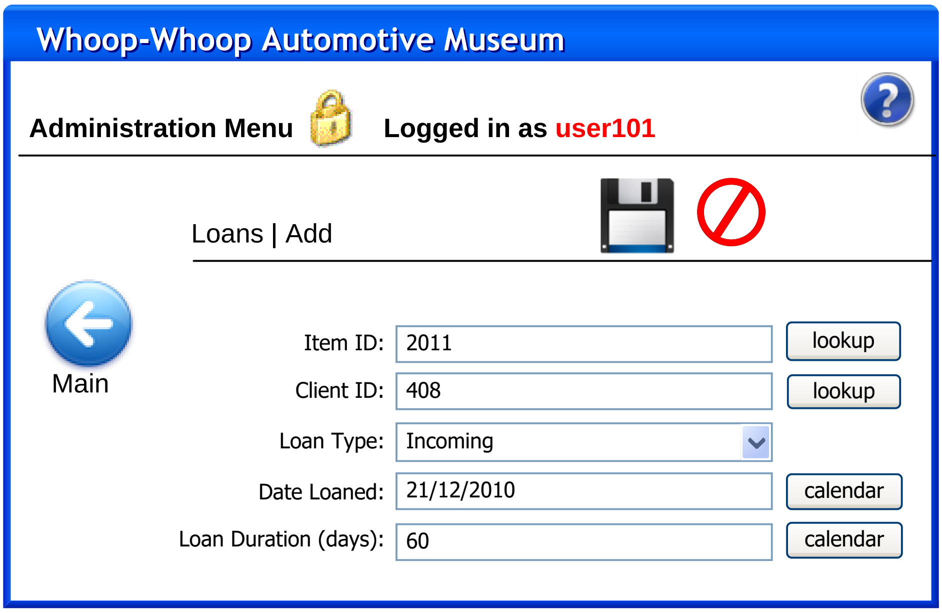
Client Name: Tamworth Roadshow

Loan Type: Outgoing

Date Loaned: 13 December 2010

Loan Duration: 90 days

Return Date: 13 March 2010



Whoop-Whoop Automotive Museum

Administration Menu Logged in as user101

Loans | Add

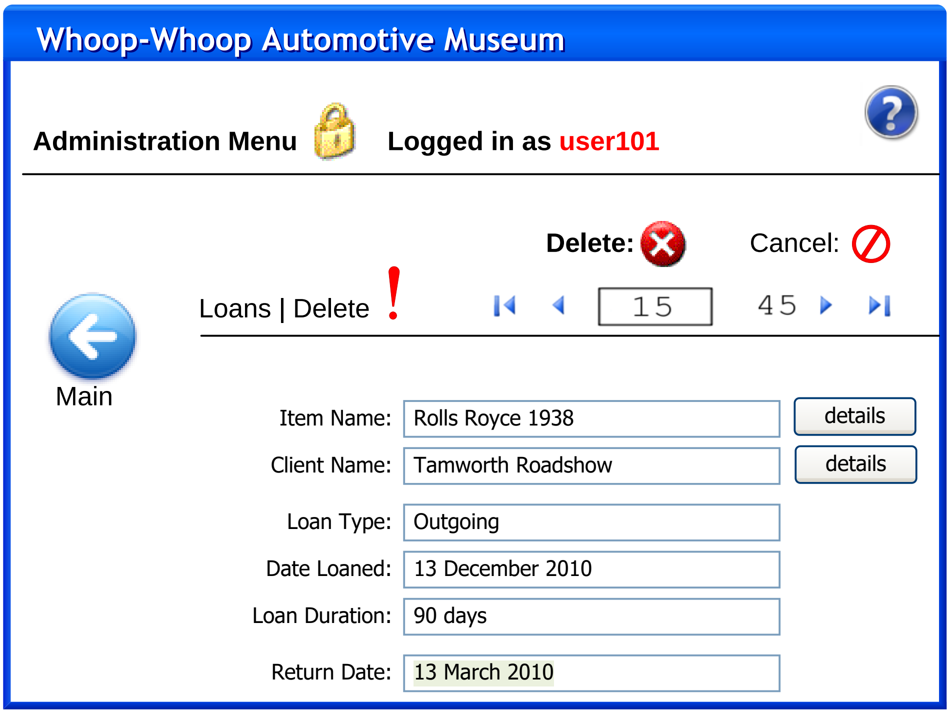
Item ID: 2011

Client ID: 408

Loan Type: Incoming

Date Loaned: 21/12/2010

Loan Duration (days): 60



Item Name: Rolls Royce 1938

Client Name: Tamworth Roadshow

Loan Type: Outgoing

Date Loaned: 13 December 2010

Loan Duration: 90 days

Return Date: 13 March 2010

Note the use of the details, lookup, calendar and help features. These will open pop-up windows which the user can either browse related records, dates and obtain other important information pertaining to the task being performed.

Future Directions

It was noted at the time of publishing this documentation that the Table Relations are in third normal form; it is expected that upon implementation, the designers may need to de-normalise these relations as to maximise search efficiency depending on alpha and beta test case data. For example, an item status may be a more commonly searched attribute than first expected, which may warrant its storage as an extra field in the Items relation. This will be decided upon in initial concept system testing.

The class structure may invariably change due to the use of:

a) multiple forms to "declutter" workspace; and
b) necessary data handling structures not identified in the specification stage.

In this case, as has been noted, the requirements and functionality of the information kiosk *upon execution* must not be compromised.

The interface design is non-exhaustive and indicative only. Further changes may be made to the form designs (above) depending on user aesthetics, accessibility needs, client needs and possible system, platform or peripheral changes encountered through the implementation. As per the class structure, it is important that the usability, integrity and efficiency of the system remain uncompromised despite any changes made to the interface.

Form resolution has altered in the interface designs; this will remain consistent based on the technology being developed for. At the time of implementation, the screens purchased will be measured and this resolution will be developed for. As such, some altering of layout will occur however functionality will not be affected.

Finally, it is **strongly recommended** that a user hierarchy be established for Administrators of the information kiosk, with associated and varying permissions based on employee position. For example, museum staff should not have access to high risk functions like deleting whole collections or loaning items to other clients; only museum curators should be allowed this privilege.

Given this, the requirements and specifications have been detailed to client satisfaction. Thus, as per the project time schedule, work will continue on the fully-functioning implementation of the concept system.

References

Unless otherwise specified, this assignment was completed using only the following references:

Sommerville, I. 2007. Software Engineering 8th Edition. Addison-Wesley, Edinburgh Gate, Harlow.