The University of Akron
IdeaExchange@UAkron

Williams Honors College, Honors Research Projects

The Dr. Gary B. and Pamela S. Williams Honors College

Summer 2019

Relate Promotion Tool

David Nutt
dcn18@zips.uakron.edu

Please take a moment to share how this work helps you through this survey. Your feedback will be important as we plan further development of our repository.
Follow this and additional works at: https://ideaexchange.uakron.edu/honors_research_projects

Part of the Databases and Information Systems Commons

Recommended Citation
Nutt, David, "Relate Promotion Tool" (2019). *Williams Honors College, Honors Research Projects*. 844.
https://ideaexchange.uakron.edu/honors_research_projects/844

This Honors Research Project is brought to you for free and open access by The Dr. Gary B. and Pamela S. Williams Honors College at IdeaExchange@UAkron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Williams Honors College, Honors Research Projects by an authorized administrator of IdeaExchange@UAkron. For more information, please contact mjon@uakron.edu, uapress@uakron.edu.

David
Nutt

Relate Promotion Tool

2440:452 SENIOR CIS PROJECTS

SPRING 2019

Table of Contents

Introduction to the Problem

Goals and Purpose

Company X is a systems integrator for a point-of-service framework called Xstore offered by Oracle. Clients use a program included with Xstore called Relate to create sales, deals, and coupons. When selecting an inventory object in Relate, the user must enter the object's ID in a search field and select objects from the results list individually. If a user desires to select an object but omit a specific child object from a promotion they must instead enter all child object codes individually. This process is slow and requires the user to know the ID of whatever object(s) they wish to select. What is to be personally gained from the project is as follows: a greater familiarity with the practicalities of software development, a deeper understanding of the C# environment, and further experience with object-oriented design.

System Capabilities

The user must be able to:

- Import inventory information from a SQLServer database
- Display inventory information in a manner that is easily navigated
- View information relating to an inventory object
- Easily select and omit inventory objects
- Export selected inventory objects in comma separated value (CSV) format in such a way that minimizes values in the exported file
- Search for inventory objects based on ID and filter based on categorization (Department, Sub department, Category, Subcategory, Style, and Item)

Business Benefits

The program would:

- Allow clients to use Relate without extensive knowledge of inventory IDs
- Expediate client promotion implementation
- Provide clients with a unique benefit by choosing Company X

Significance

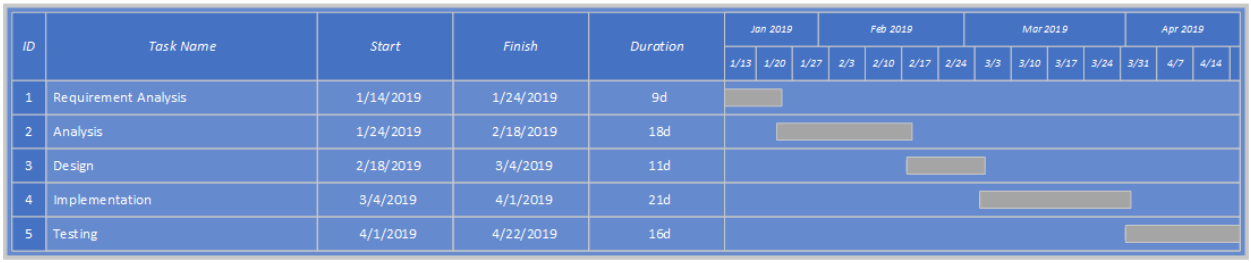
This project demonstrates a practical solution to a real-world problem, executed at a level of quality that is thorough and to be expected in a business environment.

Intended Solution

A standalone application which will allow the user to enter Relate database information and then view the inventory contents of that database in an easily navigable tree. The tree will contain all inventory objects as a child of their respective parent object. By request, the tree nodes will display the objects' inventory ID and have a hover over label that will display their description. Every tree node will also contain a checkbox, that when checked will enable all descendants and when unchecked disable all descendants and ancestors. When a tree node is highlighted (or 'clicked') by the user, the information of the object that node represents will display in a panel to the right of the tree. The user may search inventory objects by ID and filter by categorization. Once the user has selected and omitted all the desired inventory objects they may then export their selections to a comma separated value (CSV) file which will contain the highest level selected inventory object(s), as to minimize values in the CSV file, in the correct format for Relate to import.

Project Timeline

ID	Task Name	Start	Finish	Duration
1	Requirement Analysis	1/14/2019	1/24/2019	9d
2	Analysis	1/24/2019	2/18/2019	18d
3	Design	2/18/2019	3/4/2019	11d
4	Implementation	3/4/2019	4/1/2019	21d
5	Testing	4/1/2019	4/22/2019	16d



Ordered Objectives

1) A user must be able to import a Relate database.

The user must be able to begin the import process and enter the required information for a particular database. The required information is as follows: host IP address and port, database name, username, password, organization ID. The data is then converted to objects and stored in a tree.

2) A user must be able to view the objects in an imported Relate database in an easily navigable manner.

The object information should be displayed in a tree, with the hierarchies from the database preserved as nodes and items as leaves. Due to the possible length of object descriptions, the tree should display the object codes with a hover over display for the description and the tree panel should be resizable. When selected, the properties of an object should be displayed in an object properties panel. The properties are as follows: level code (one of the five user-set codes or item), ID, number of children (blank for items), and description. The level codes are customizable and must be imported from the Relate database.

3) A user must be able to select and omit objects in an imported Relate database in a quick and intuitive way.

The object information displayed in the tree detailed above should include checkboxes. When toggled an object is included in exports, otherwise it is omitted.

4) A user must be able to export the selected objects in an imported Relate database in comma separated value format and minimize entries in the export.

The user must be able to select an export option and select a name and destination for the exported CSV file. To ensure that entries are minimized, when a node is toggled in the tree all child nodes should be toggled. When a node or item is omitted all direct parent nodes should be omitted as well; however, all other child objects of said parents should remain in their previous toggle state. The CSV file will contain all toggled objects at the highest possible level of the hierarchy. This will be done recursively starting at the root node. If a node is toggled it will be added to the CVS file and the next node will be checked, if a node is not toggled all its child

objects will be checked similarly. The format for a CSV entry is as follows: "_levelCode _, hierarchy_ID, , , , ," (no quotes).

5) A user must be able to search for objects and view their parent and child objects.

The user must be able to enter an object's ID and select a filter (one of the 5 level codes or item) to view a tree containing only the object with the specified ID, that object's parents, and that object's children. This tree must function similarly to the tree containing all objects (i.e. selecting an object would display that object's properties, objects must be able to be toggled) and all objects displayed in the search tree must have consistent toggle states with their respective objects in the main tree.

Analysis

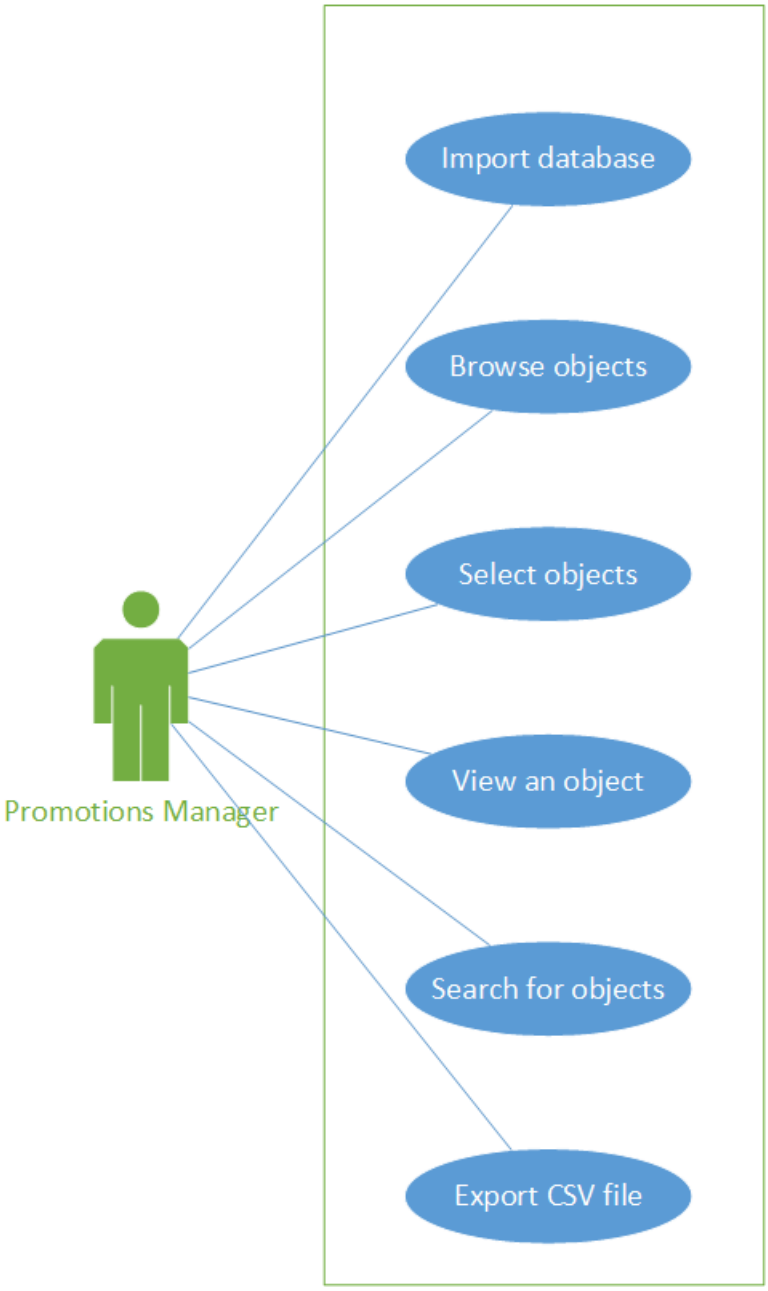
System Requirements

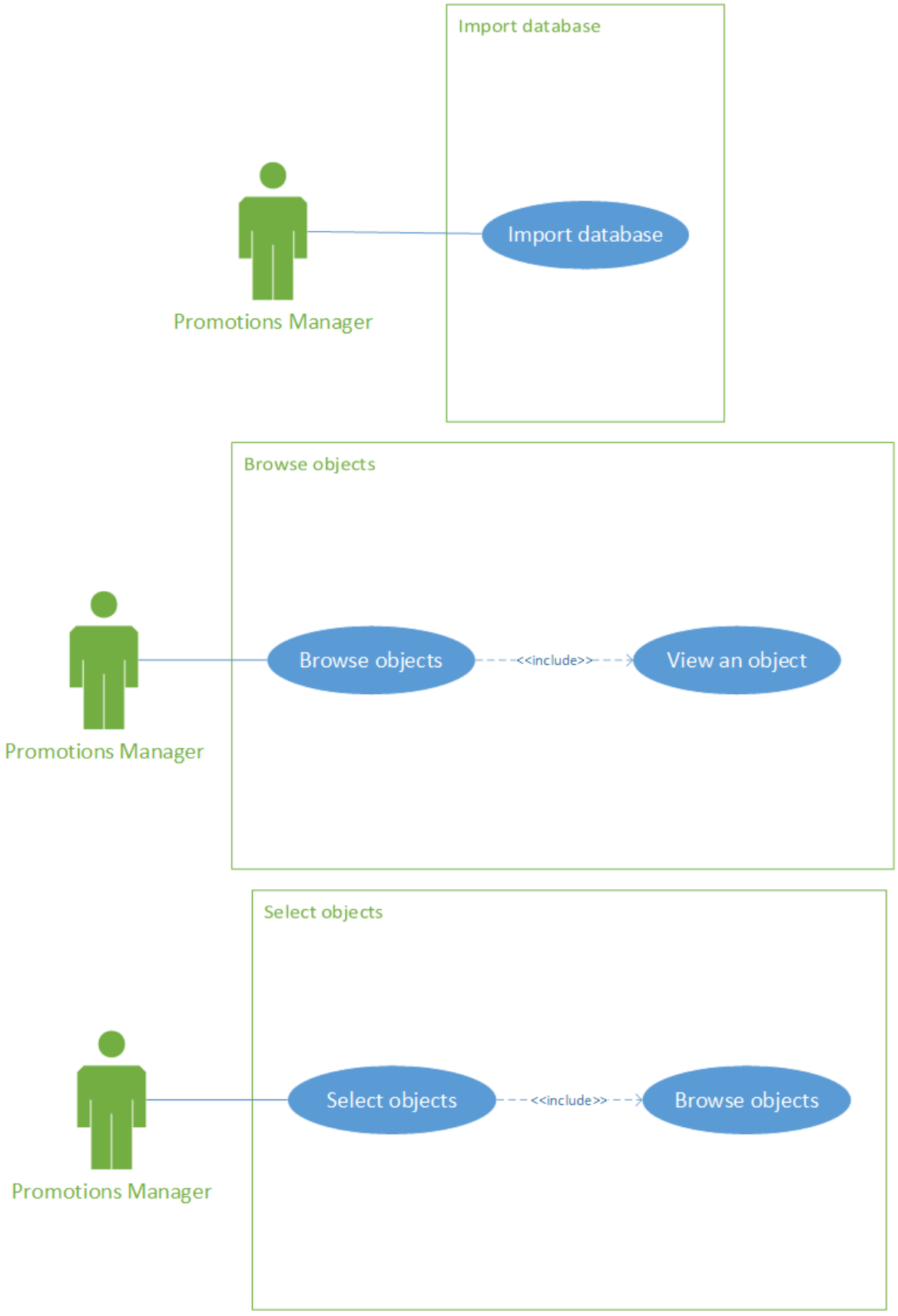
FURPS Category	Requirements
Functionality	- Import inventory objects from database - Select and omit inventory objects - View inventory object information - Export selected inventory objects to CSV file
Usability	- Display inventory objects in tree - Display inventory objects' descriptions as hover over - Checking an inventory item checks all descendants - Unchecking an inventory item unchecks all descendants and ancestors - Searching via inventory ID and filter based on category
Reliability	- Display error message on inability to locate database server - Display error message on inability to locate named database - Display error message on inability to locate organization in database - Display error message on inability to extract information from database - Display error message on failure to enter a required field
Performance	- None
Security	- None

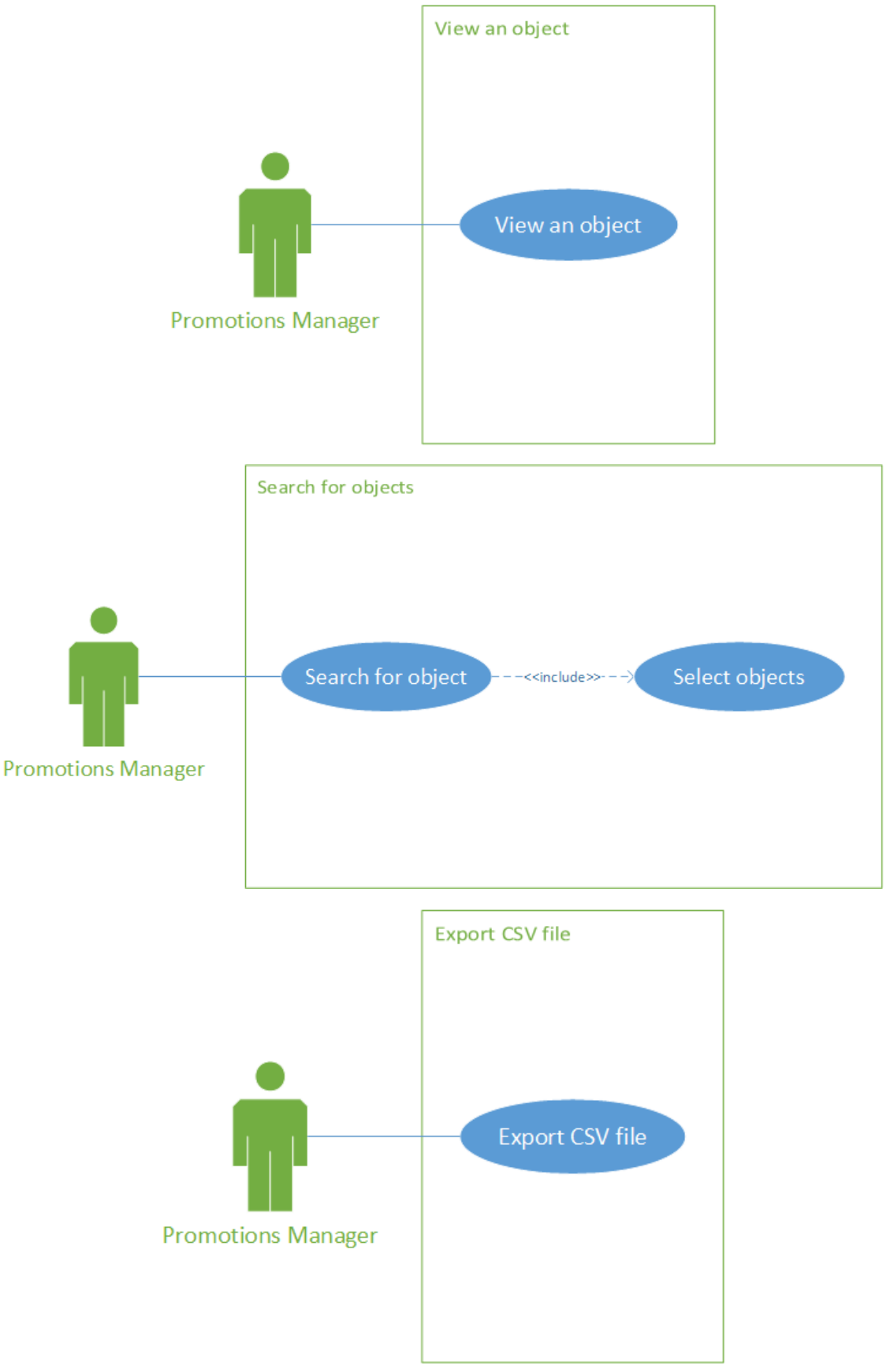
Use Case Table

Use Case	Brief Use Case Description
Import database	The user may enter database information, the system must import the relevant data from the database, the system must convert the imported data to objects.
Browse objects	The user must be able to navigate imported objects in a tree structure. The system must store imported objects in a tree according to their hierarchy IDs.
Select objects	The user may toggle an object's checkbox. If the user is enabling an object, the system must enable all child objects. If the user is disabling an object, the system must disable all child and parent objects.
View an object	The user may select an object. The system must display the objects level code, ID, number of children, and description.
Search for objects	The user may enter an object's ID and filter and the system must display a tree containing that object, that object's children, and that object's ancestors.
Export CSV file	The user may select a name and destination for the CSV file being exported. The system must create a file in specified location and write all selected objects to that file in the appropriate format.

Use Case Diagrams




Import database
Import database
Promotions Manager
Browse objects
Browse objects
<<include>>
View an object
Promotions Manager
Select objects
Select objects
<<include>>
Browse objects
Promotions Manager


View an object
View an object
Promotions Manager
Search for objects
Search for object
<<include>>
Select objects
Promotions Manager
Export CSV file
Export CSV file
Promotions Manager

Use Case Descriptions

Use case name:	Import database	
Scenario:	Import a Relate database.	
Triggering event:	Promotions manager wants to import a Relate database.	
Brief description:	The Promotions Manager may enter database information, the system must import the relevant data from the database, the system must convert the imported data to objects.	
Actors:	Promotions Manager.	
Related use cases:	None.	
Stakeholders:	Sales, Marketing.	
Preconditions:	Promotions Manager must enter database information and credentials.	
Postconditions:	Database query results must be converted to Merchandise Hierarchies, Items, and the Hierarchy Level Code List. Hierarchy Tree must be populated with Merchandise Hierarchies and Items. The main screen must be shown.	
Flow of activities:	Actor	System
	1. Promotions Manger clicks the 'Import Database' button. 2. Promotions Manger enters the specified database information and credentials.	1.1 System displays the 'Import Database' dialog. 2.1 System connects to database. 2.2 System queries the database. 2.3 System uses query results to fill the Merchandise Hierarchy Level Code List and create Merchandise Hierarchies and Items. 2.4 System populates Hierarchy Tree with Merchandise Hierarchies and Items.
Exception conditions:	2.1 Database information is incomplete. 2.1 Database information is not valid. 2.1 Database credentials are incorrect.	

Use case name:	Browse objects	
Scenario:	Browse Merchandise Hierarchies and Items in the Hierarchy Tree.	
Triggering event:	Promotions manager wants to view or select Merchandise Hierarchies and Items in the Hierarchy Tree.	
Brief description:	The Promotions Manager must be able to navigate imported objects in a tree structure. The system must store imported objects in a tree according to their hierarchy IDs.	
Actors:	Promotions Manager.	
Related use cases:	May invoke *View an object*.	
Stakeholders:	Sales, Marketing.	
Preconditions:	A Relate database must be imported. The Hierarchy Tree must be populated.	
Postconditions:	The Hierarchy Tree is visible and navigable.	
Flow of activities:	Actor	System
	1. Promotions Manager indicates desire to view the Hierarchy Tree.	1.1 System displays the Hierarchy Tree.
Exception conditions:	None.	

Use case name:	Select objects	
Scenario:	Toggle an object in the Hierarchy Tree.	
Triggering event:	Promotions manager wants to include or omit an object.	
Brief description:	The Promotions Manager may toggle an object's checkbox. If the Promotions Manager is enabling an object, the system must enable all child objects. If the Promotions Manager is disabling an object, the system must disable all child and parent objects.	
Actors:	Promotions Manager.	
Related use cases:	Invokes *Browse objects*.	
Stakeholders:	Sales, Marketing.	
Preconditions:	The Hierarchy Tree is visible and navigable.	
Postconditions:	The object's checkbox is toggled. If the object's checkbox was enabled, all the object's children are checked. If the object's checkbox was disabled, all the object's children are disabled, and all the object's ancestors are disabled.	
Flow of activities:	Actor	System
	1. Promotions Manager toggles an object's checkbox.	1.1 System toggles that object's checkbox. 1.2a If the checkbox is being enabled, all that object's children's checkboxes are enabled. 1.2b If the checkbox is being disabled, all that object's children's checkboxes are disabled. All of that object's ancestors are disabled.
Exception conditions:	None.	

Use case name:	View an object	
Scenario:	Display the attributes of an inventory object.	
Triggering event:	Promotions Manager clicks an object in the Hierarchy Tree.	
Brief description:	The Promotions Manager may highlight an object. The system must display the objects level code, ID, number of children, and description.	
Actors:	Promotions Manager.	
Related use cases:	None.	
Stakeholders:	Sales, Marketing.	
Preconditions:	The Hierarchy Tree is visible and navigable.	
Postconditions:	The desired object's level code, ID, description, and if a Merchandise Hierarchy, number of children are displayed on the Information Screen.	
Flow of activities:	Actor	System
	1. Promotions Manager specifies an object in the Hierarchy Tree.	1.1 System displays the specified object's level code, ID, description, and if a Merchandise Hierarchy, number of children.
Exception conditions:	None.	

Use case name:	Search for objects	
Scenario:	Search for an object with a specific ID.	
Triggering event:	Promotions Manager clicks the Search button.	
Brief description:	The Promotions Manager may enter an object's ID and filter and the system must display a tree containing that object, that object's children, and that object's ancestors. If no ID is entered, the main Hierarchy Tree is shown.	
Actors:	Promotions Manager.	
Related use cases:	Invokes *Select objects*.	
Stakeholders:	Sales, Marketing.	
Preconditions:	A Relate database must be imported. The Hierarchy Tree must be populated.	
Postconditions:	The Hierarchy Tree is replaced with a tree containing the object of the specified Merchandise Hierarchy Level Code and Hierarchy ID/ Item ID, that object's children, and that object's ancestors.	
Flow of activities:	Actor	System
	1. Promotions Manager enters the desired object's ID, specifies a Merchandise Hierarchy Level Code, and clicks the 'Search' button.	1.1 System searches the Hierarchy Tree for an object of the specified ID and Merchandise Hierarchy Level Code. 1.2 System replaces the Hierarchy Tree with a tree containing the specified object, that object's children, and that object's ancestors.
Exception conditions:	1.1 If no ID is entered, the main Hierarchy Tree is displayed. 1.2 If no object is found with the specified ID with the specified Merchandise Hierarchy Level Code, a message is displayed saying 'No object with the specified ID and Merchandise Hierarchy Level Code found.	

Use case name:	Export CSV file	
Scenario:	Creation of CSV file.	
Triggering event:	Promotions Manager wants to export their selections.	
Brief description:	The Promotions Manager may select a name and destination for the CSV file being exported. The system must create a file in specified location and write all selected objects to that file in the appropriate format (_levelCode _, hierarchy_ID, , , , ,).	
Actors:	Promotions Manager.	
Related use cases:	None.	
Stakeholders:	Sales, Marketing.	
Preconditions:	A Relate database must be imported.	
Postconditions:	A CSV file with the desired name is located at the specified destination and contains an entry for every selected object (the format for entries being: _levelCode _, hierarchy_ID, , , , ,).	
Flow of activities:	Actor	System
	1. Promotions Manager clicks the 'Export CSV File' button. 2. Promotions Manager enters the desired name, specifies the location, and clicks the 'OK' button.	1.1 System displays the 'Save File' dialog. 2.1 System creates a file with the desired name at the specified location containing an entry of the appropriate format for every object that is selected in the Hierarchy Tree.
Exception conditions:	2.1 If there already exists a file in the specified location with the specified name, it must be specified if the export is to overwrite that file or be cancelled.	

Things Tables

Thing	Description
Item	An inventory object that is a type of unit of inventory. Can be located in any Merchandise Hierarchy Object.
Merchandise Hierarchy Object	An inventory object of the merchandise hierarchy.
Merchandise Hierarchy	The merchandise hierarchy containing all inventory objects and the ordered list of 5 level codes set by the user in Conflate.

Item

All items have these attributes:	Description
Inventory ID	A unique integer used to identify an item.
Description	A description of what a specific item is.
Level Code ID	The location of the "Item" level code in the Merchandise Hierarchy Level Code List.

Merchandise Hierarchy Object

All Merchandise Hierarchy Objects have these attributes:	Description
Inventory ID	A unique integer used to identify a Merchandise Hierarchy.
Parent ID	The hierarchy ID of a Merchandise Hierarchy's parent.
Level Code ID	The location of a Merchandise Hierarchy's level code in the Merchandise Hierarchy Level Code List.
Display Name	The name of a Merchandise Hierarchy that is displayed to users.
Description	A description of a Merchandise Hierarchy, typically longer than a display name.
Number of Children	Stores the number of children this Merchandise Hierarchy has in order to reduce computation.

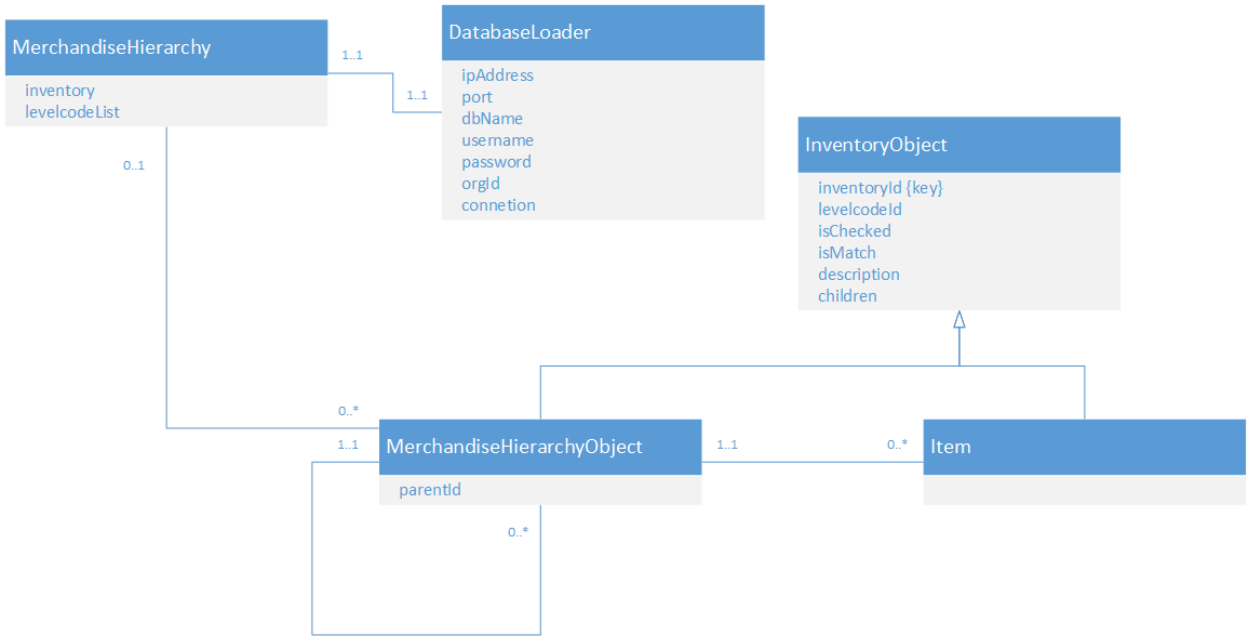
Merchandise Hierarchy

The Merchandise Hierarchy has these attributes:	Description
Inventory	The collection of all inventory objects.
Level Code List	The ordered list of level codes as defined by the user in Conflate.

Events System Responds To

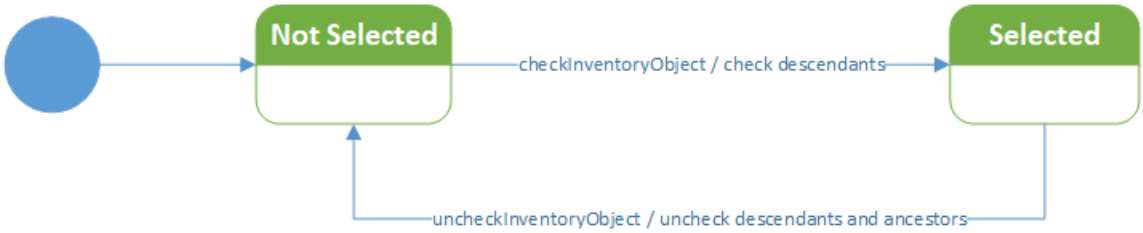
- Import database
- View inventory object
- Select/Omit inventory object
- Search for inventory object
- Export selections

Domain Model Class Diagram

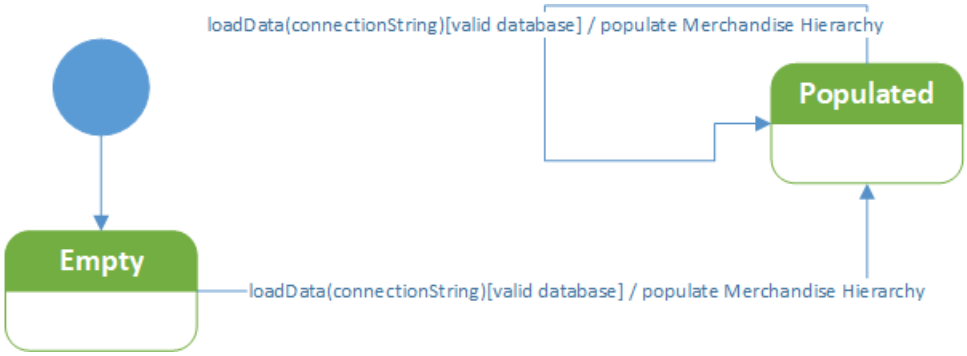


State Machine Diagrams

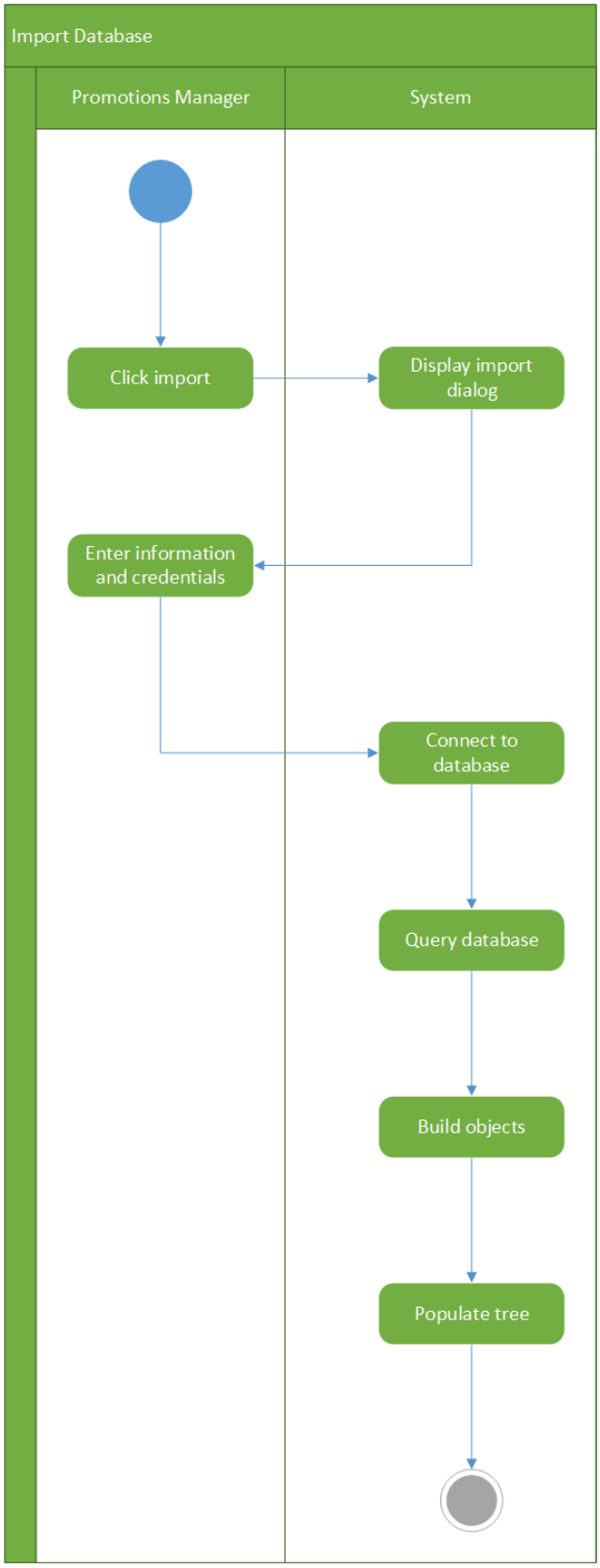
Inventory Object

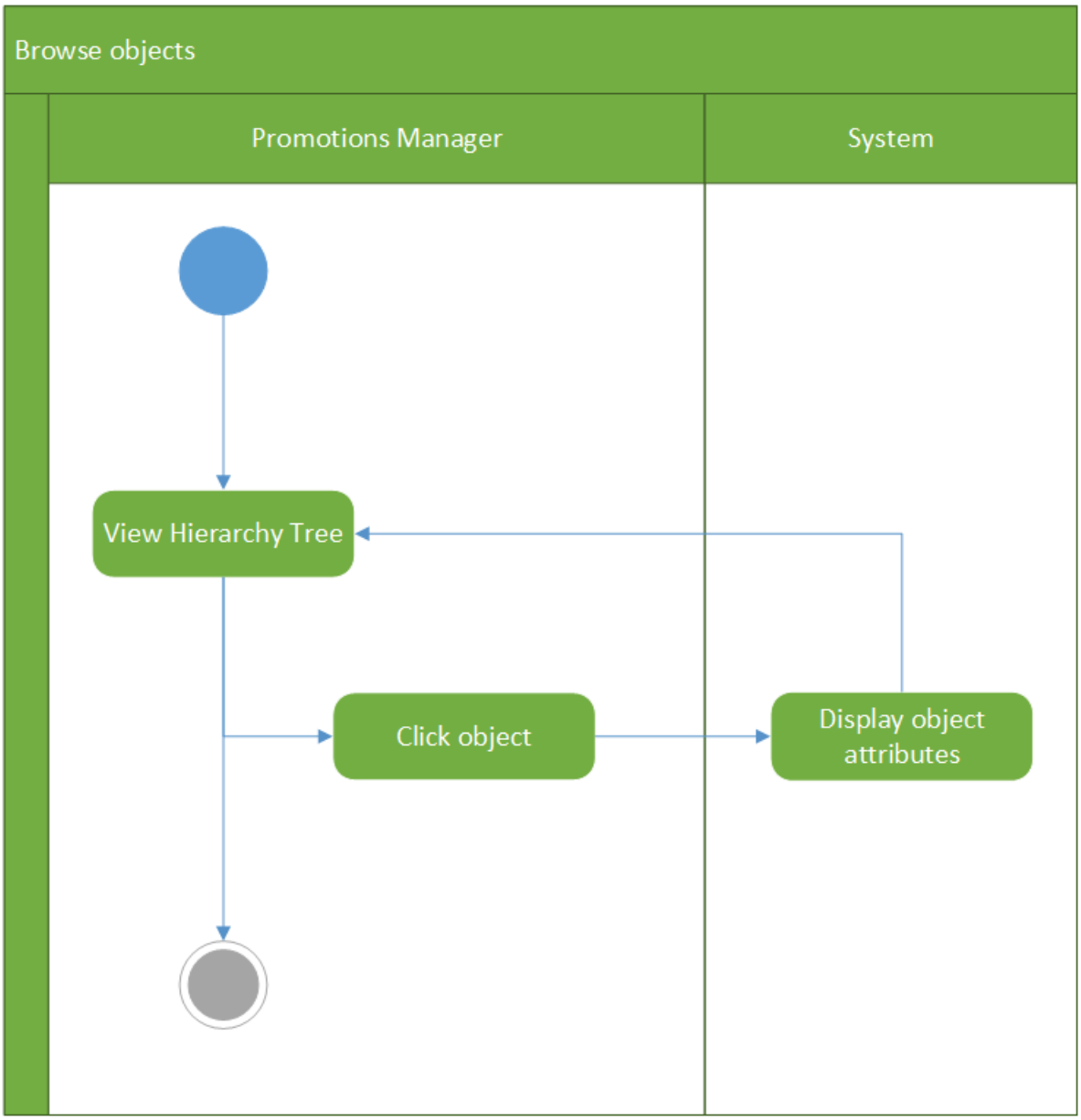


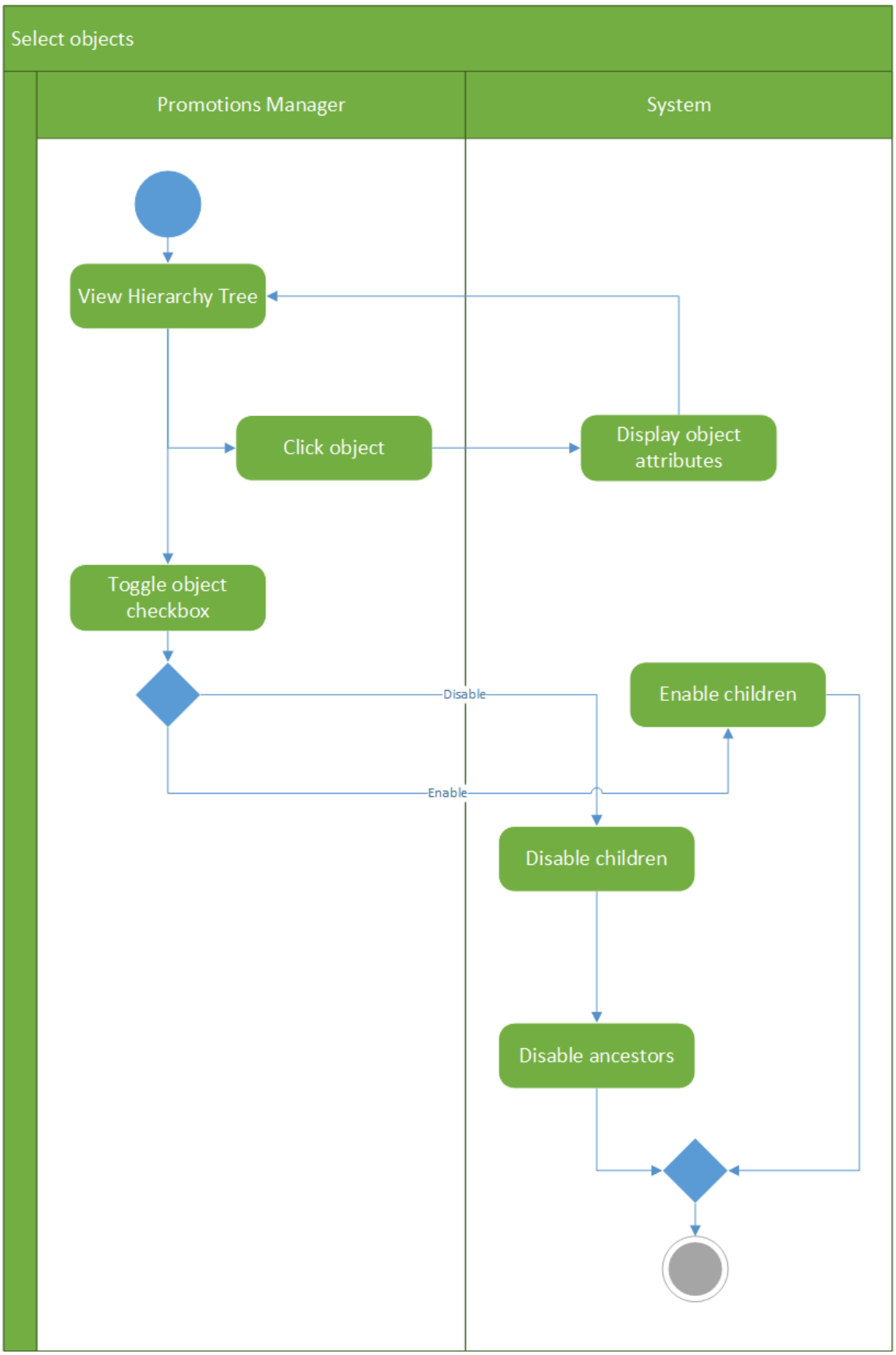
Merchandise Hierarchy

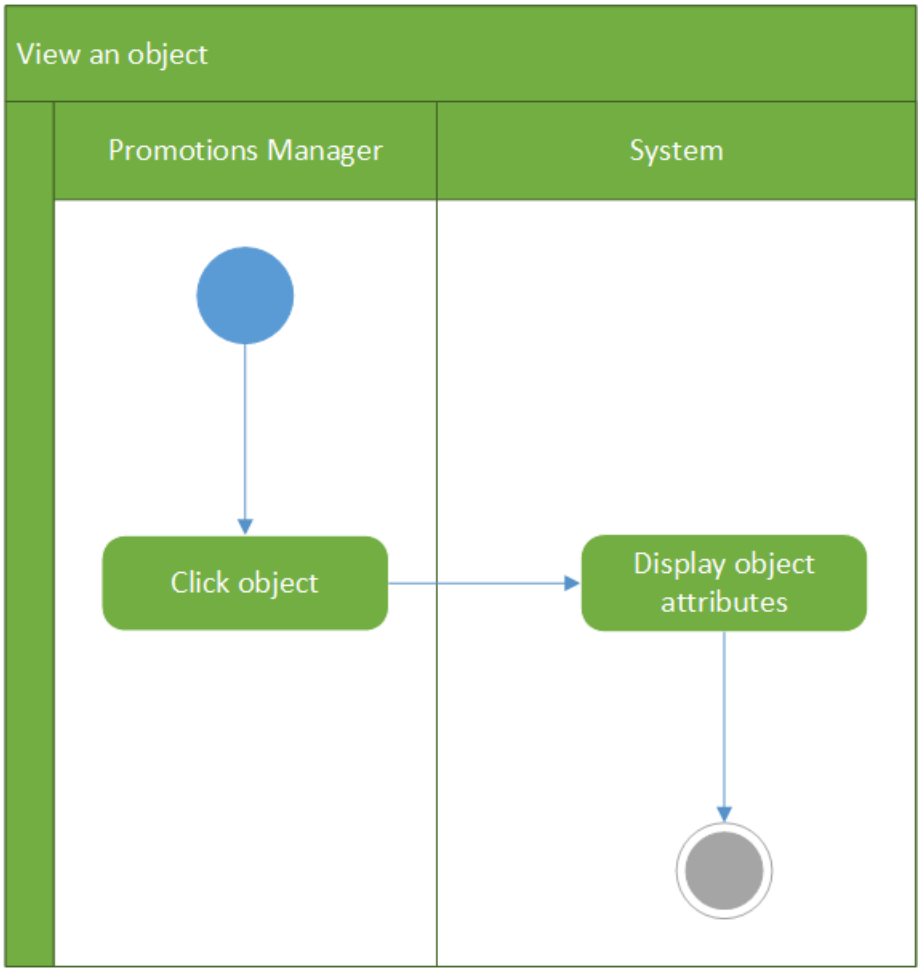


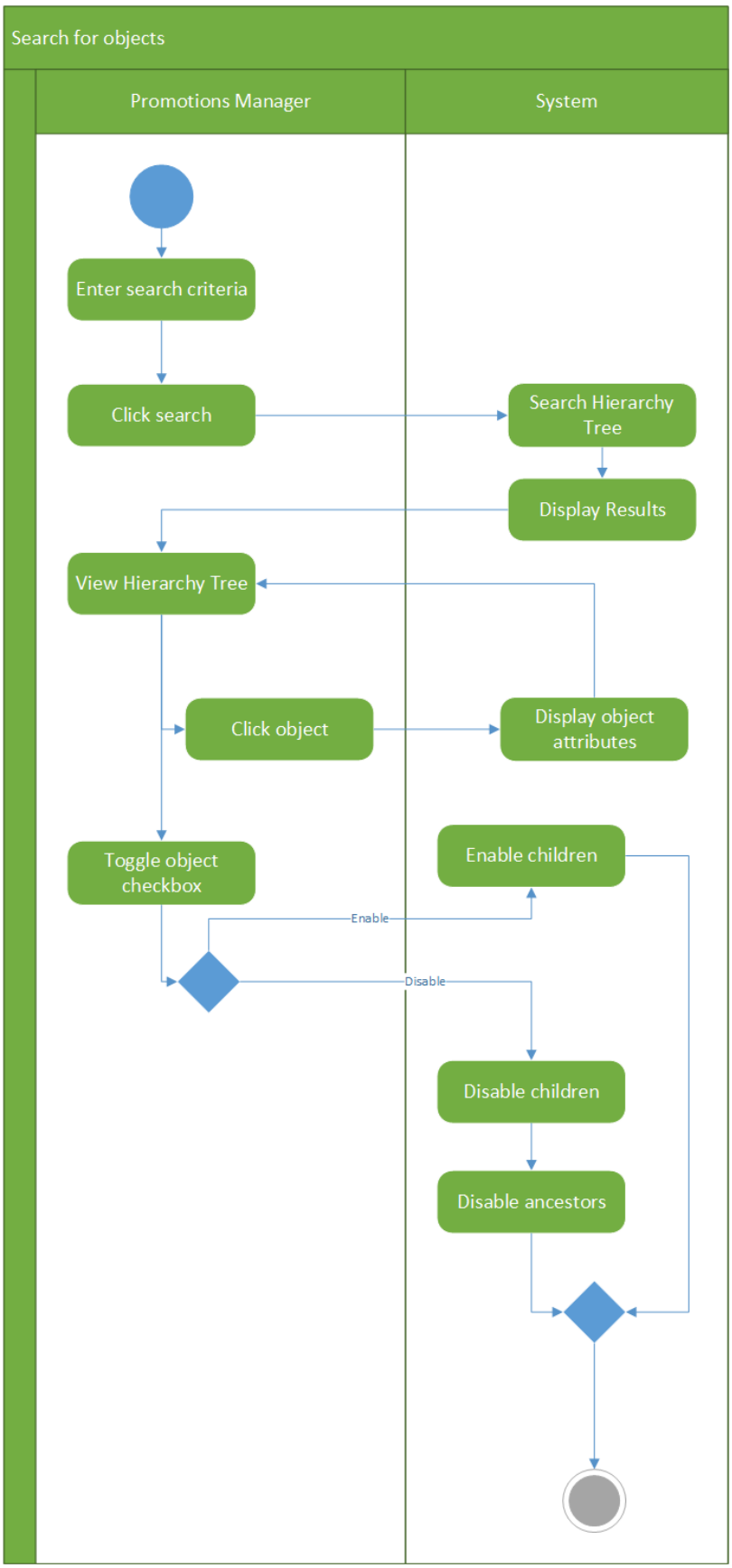
Activity Diagrams

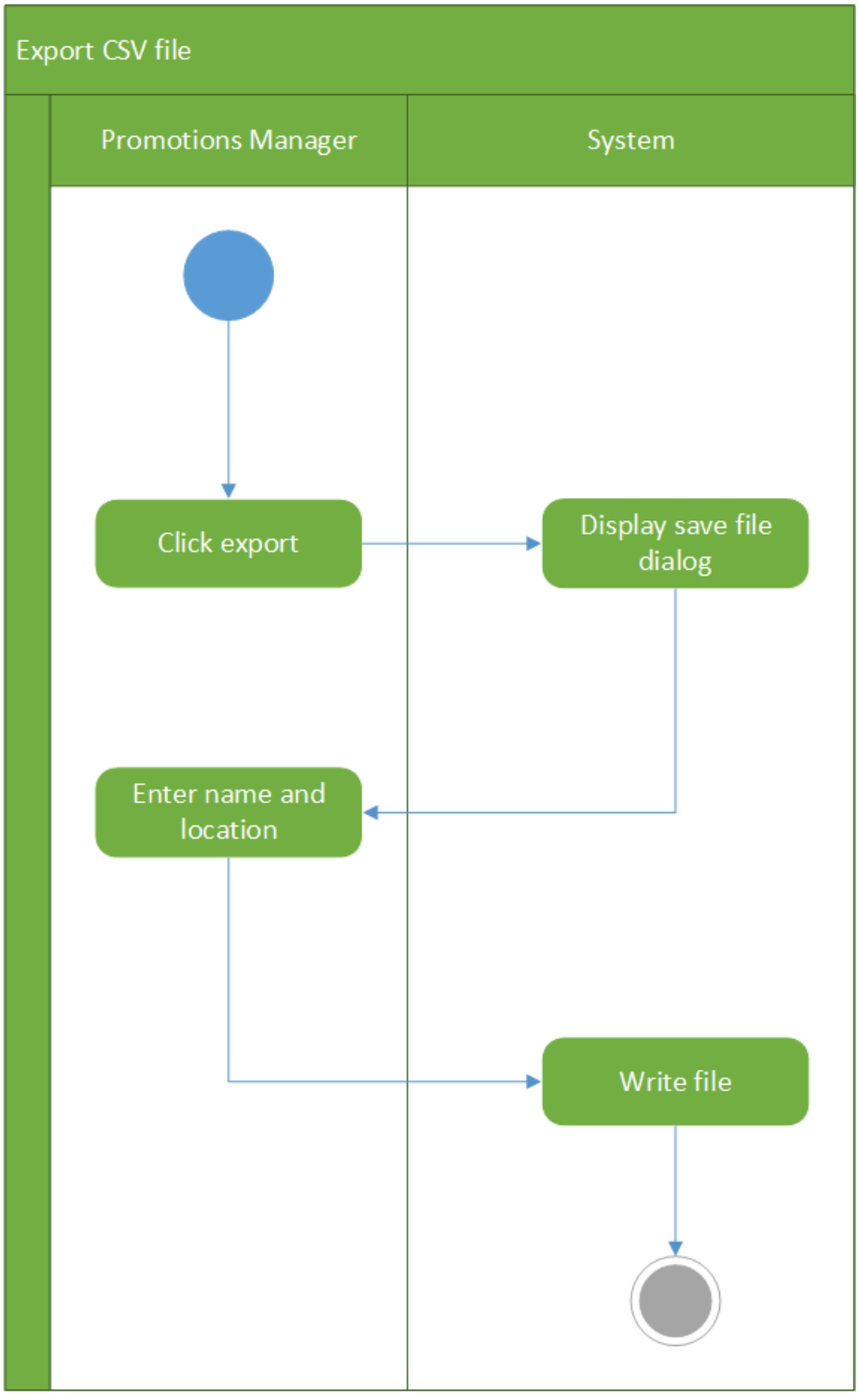



Browse objects
Promotions Manager
System
View Hierarchy Tree
Click object
Display object attributes


Select objects
Promotions Manager
System
View Hierarchy Tree
Click object
Display object attributes
Toggle object checkbox
Disable
Enable children
Enable
Disable children
Disable ancestors


View an object
Promotions Manager
System
Click object
Display object attributes

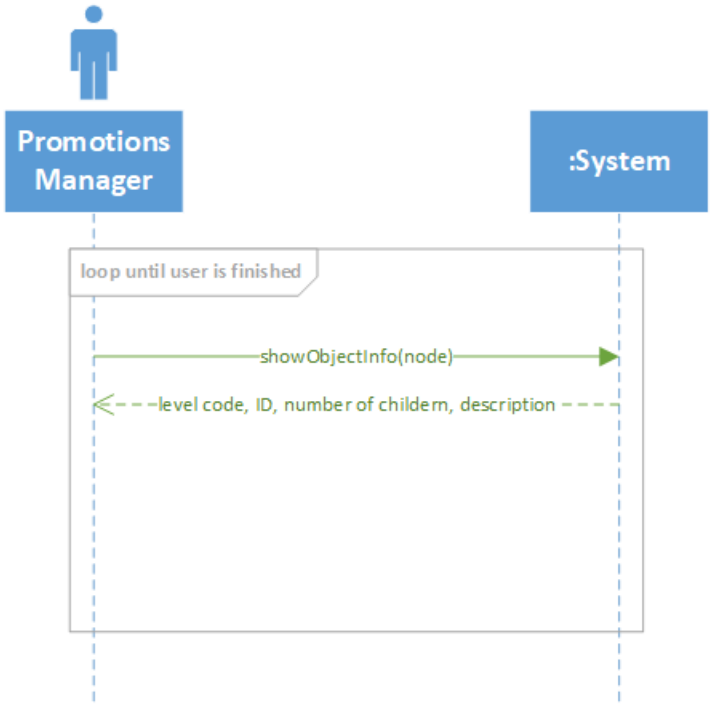

Search for objects
Promotions Manager
System
Enter search criteria
Click search
Search Hierarchy Tree
Display Results
View Hierarchy Tree
Click object
Display object attributes
Toggle object checkbox
Enable children
Enable
Disable
Disable children
Disable ancestors


Export CSV file
Promotions Manager
System
Click export
Display save file dialog
Enter name and location
Write file

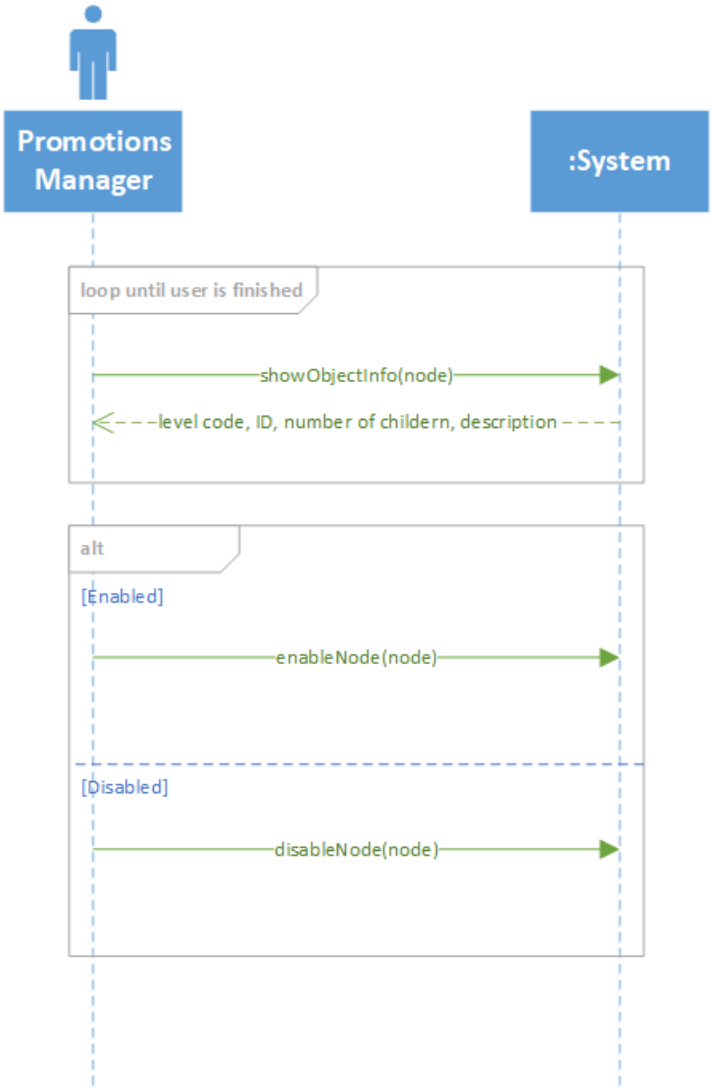
System Sequence Diagrams

Import Database

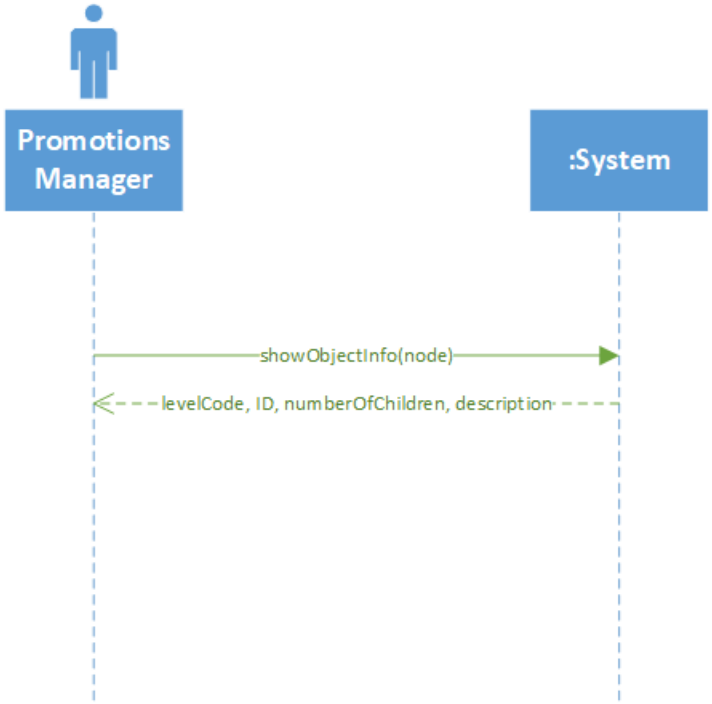
Browse Objects



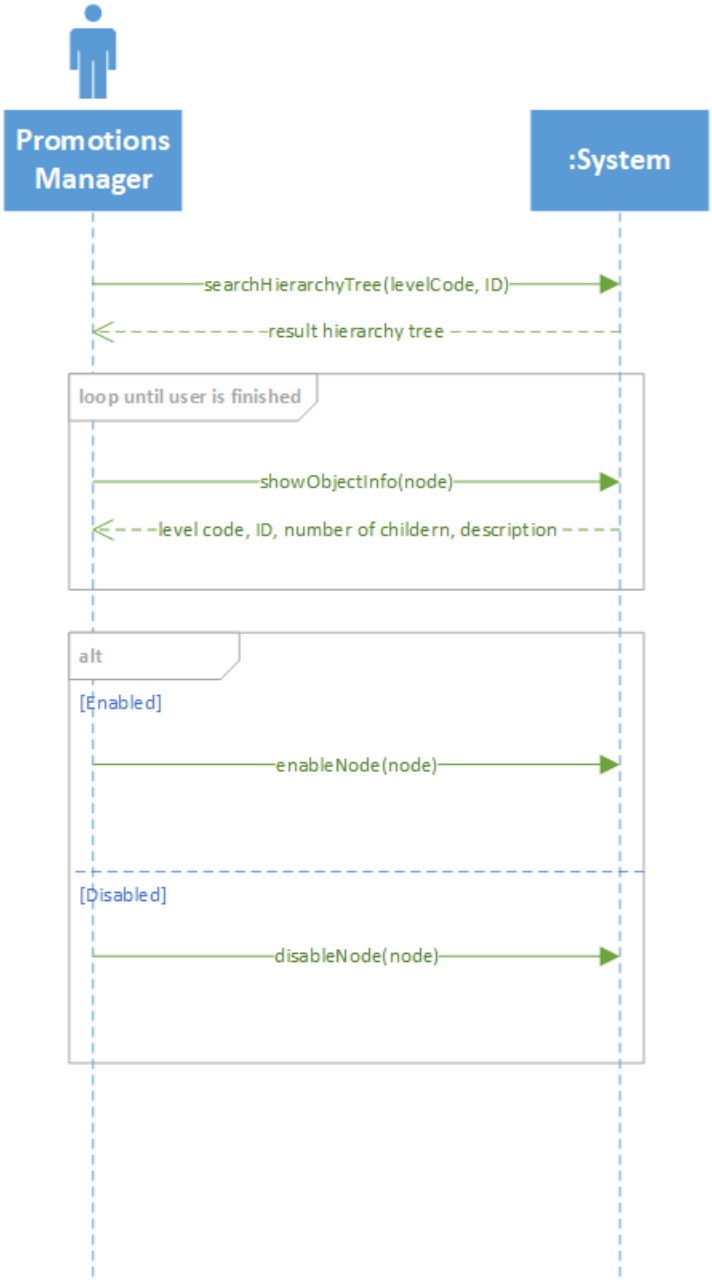
Select Objects



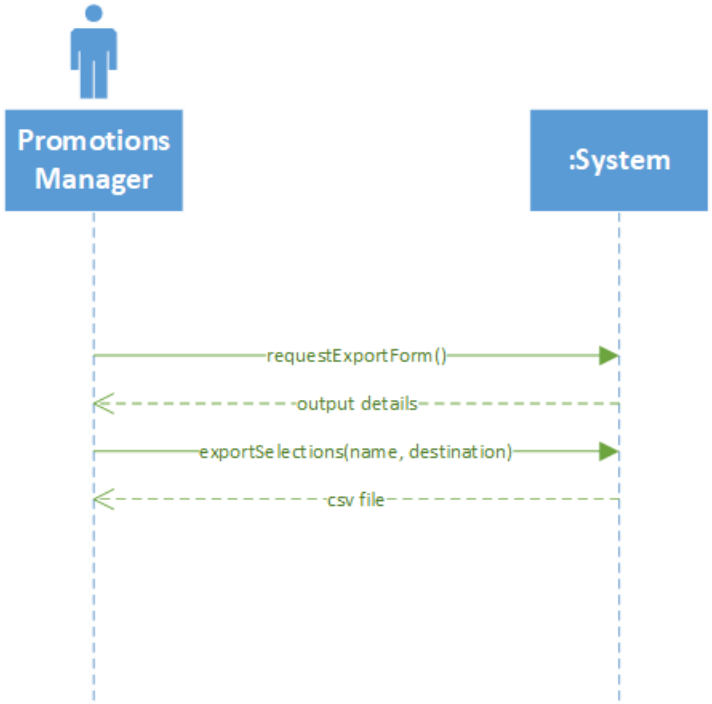
View an Object



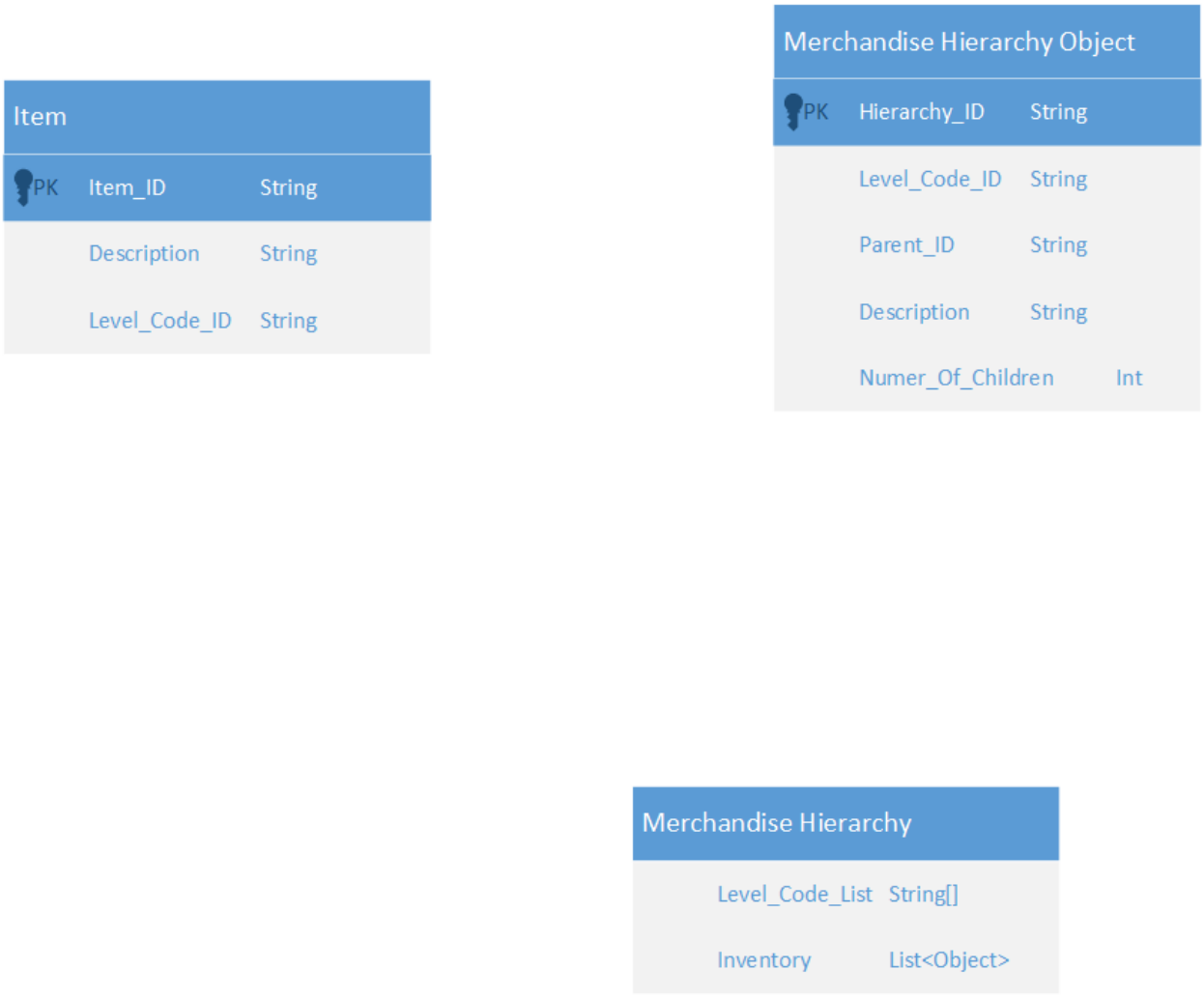
Search for Objects



Export CSV File



Logical ERD

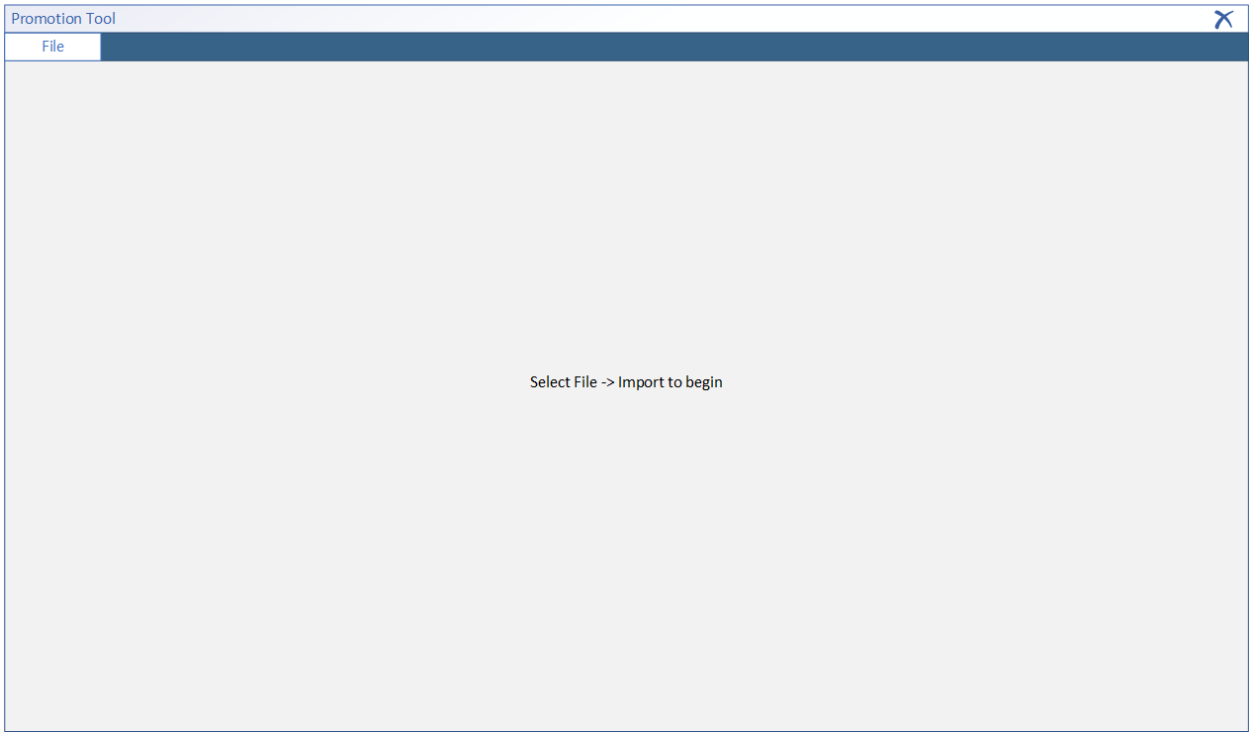

Item
PK Item_ID String
Description String
Level_Code_ID String
Merchandise Hierarchy Object
PK Hierarchy_ID String
Level_Code_ID String
Parent_ID String
Description String
Numer_Of_Children Int
Merchandise Hierarchy
Level_Code_List String[]
Inventory List<Object>

Design

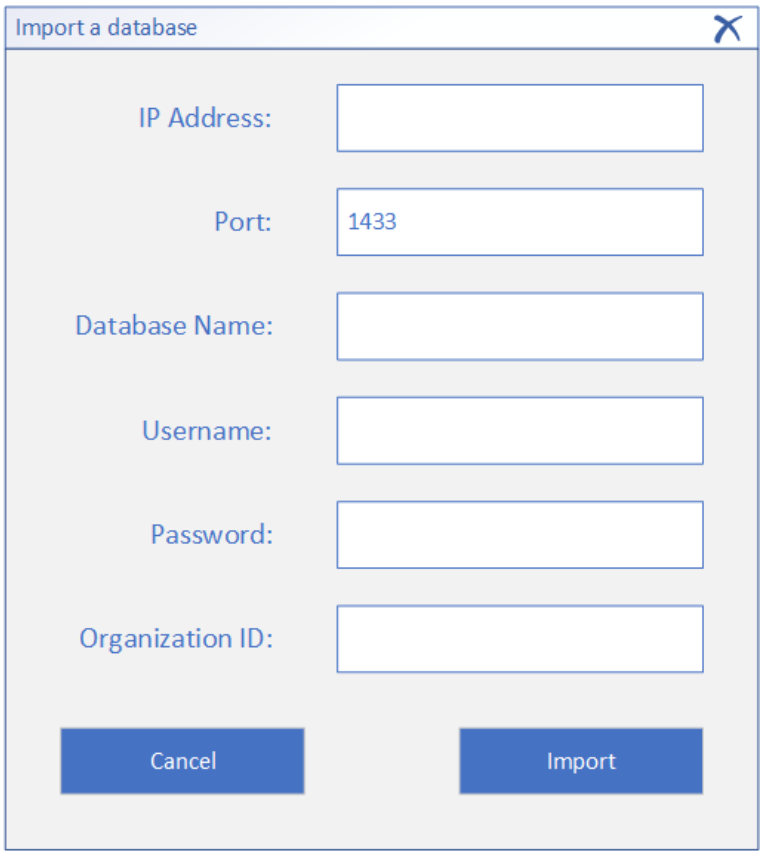
System Controls and Security Design

This project is was created to supplement an existing system. The user has no ability to add, remove, or change the data in any way. All queries performed against the database are done solely to retrieve data and are not augmented by user input. The result file, containing the user's selections, is meant to be fed directly into the Relate program via a previously implemented system that possesses its own controls and security. As a result, this project is does not contain system controls or security design as they are unnecessary.

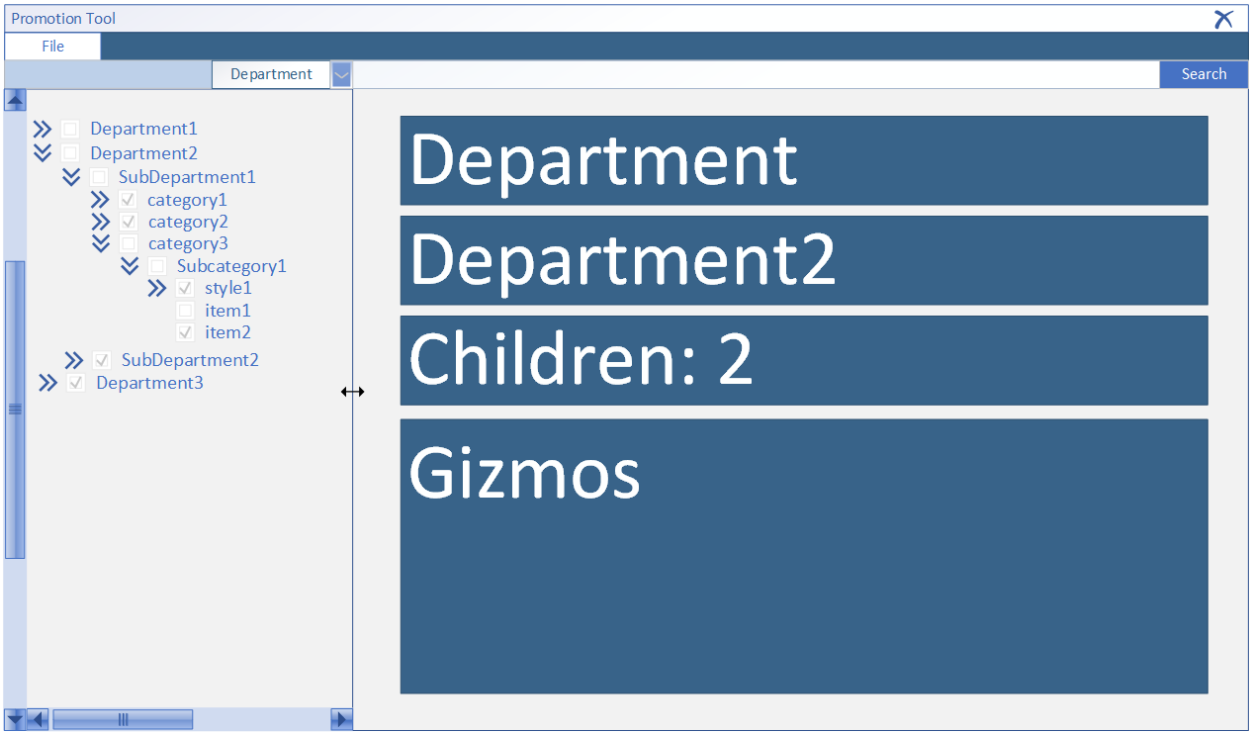
The User Interfaces Design



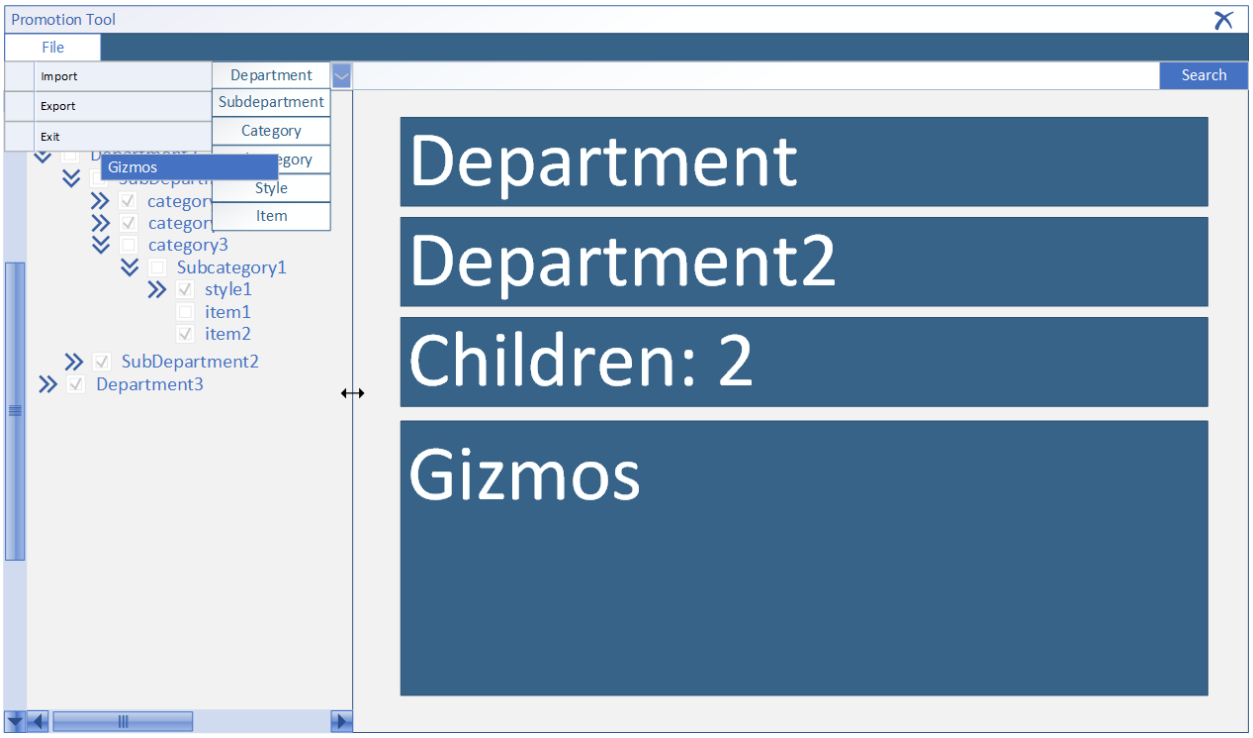
The initial screen displayed on startup.



The Import screen displayed when the user selects 'Import' under the 'File' menu.

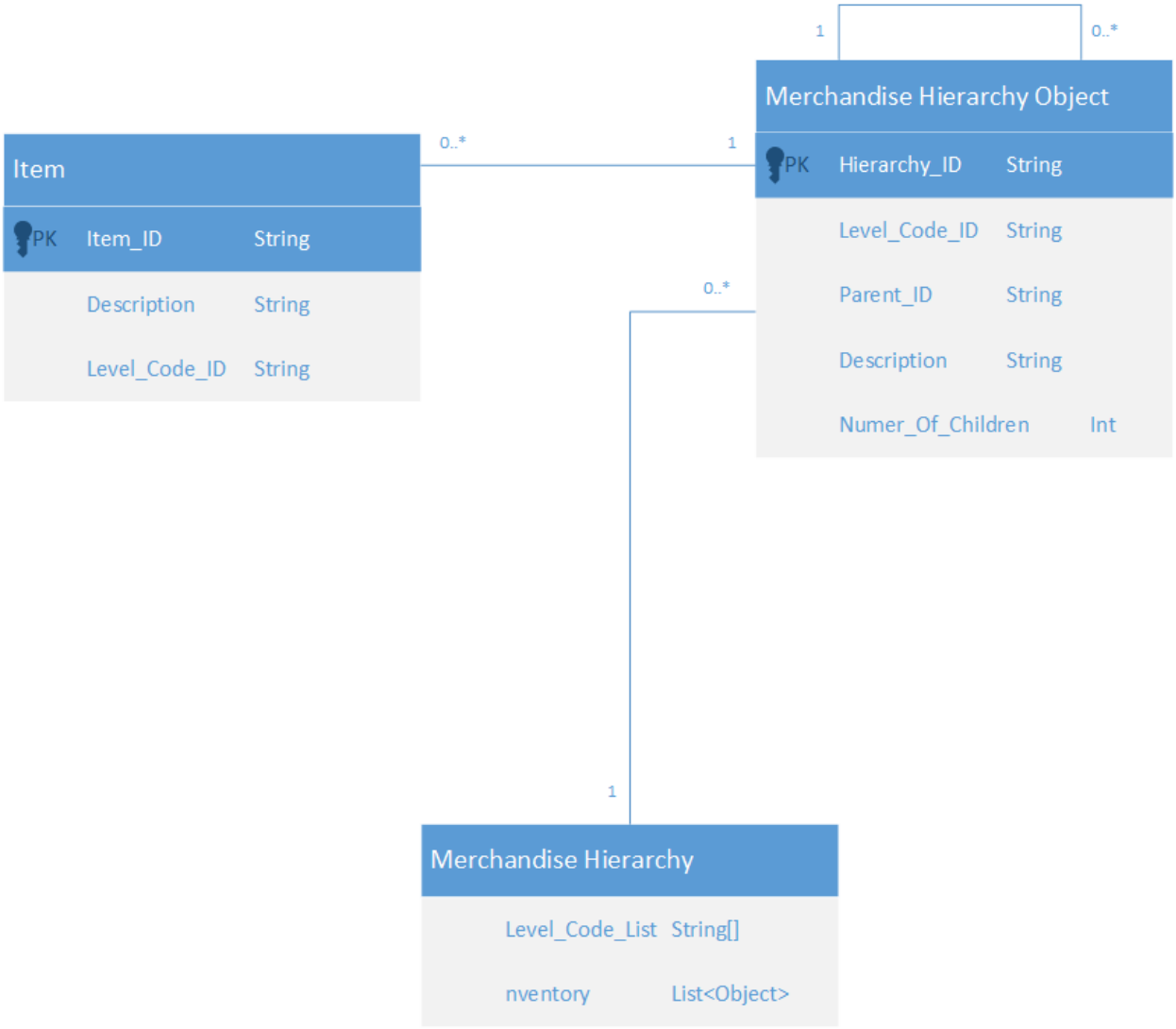


The main screen displayed after an import has been performed.

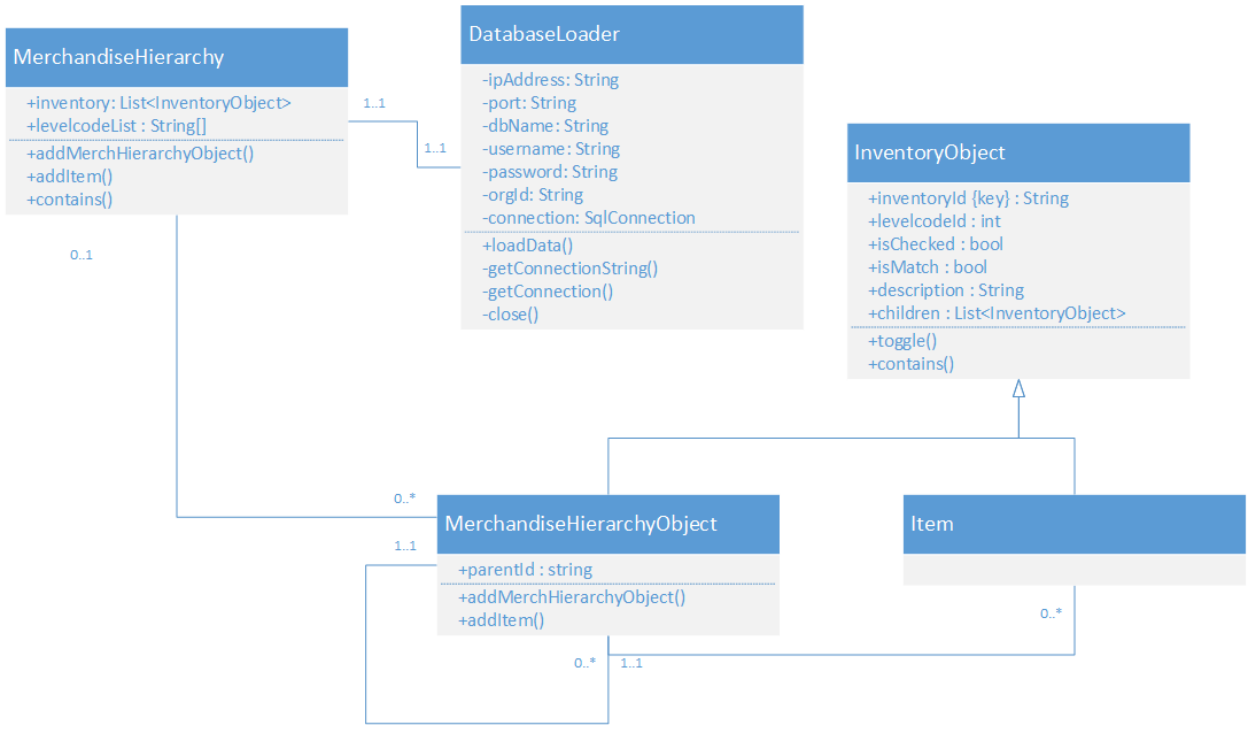


The main screen showing the contents of the 'File' menu, the contents of the search filter, and the hover over for tree objects.

Physical ERD

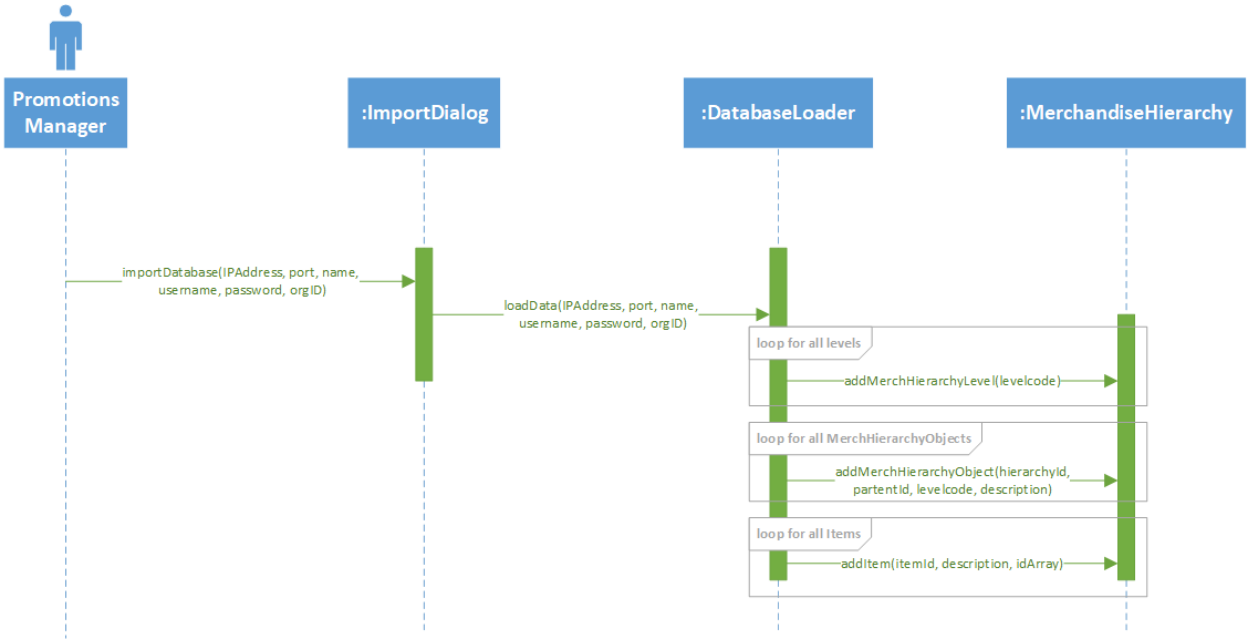


Design Class Diagram

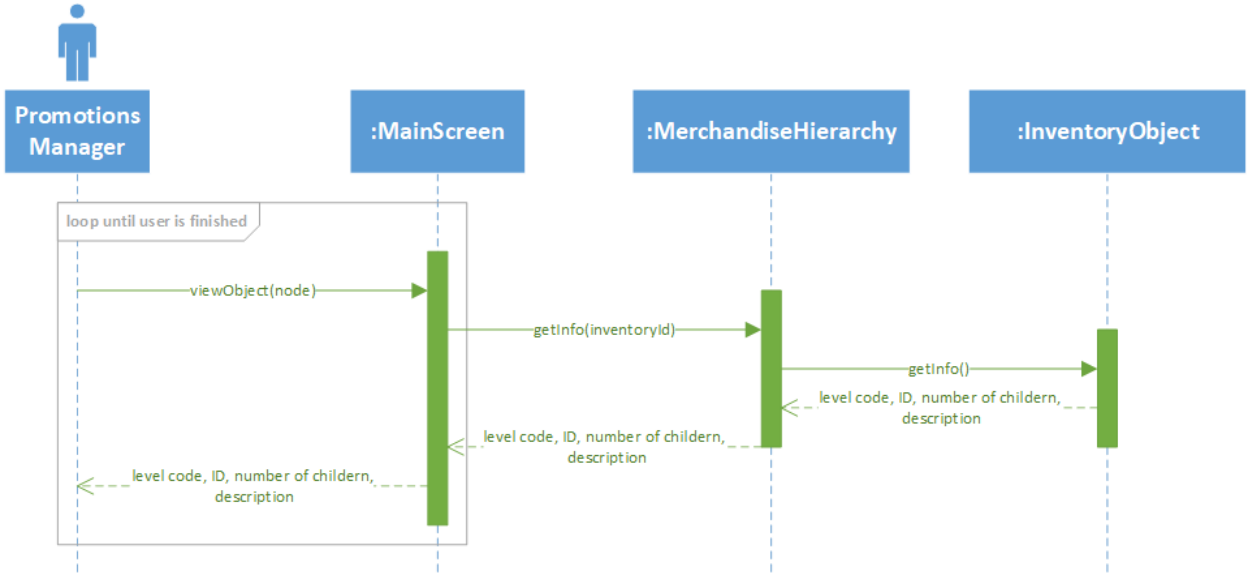


Use Case Realization

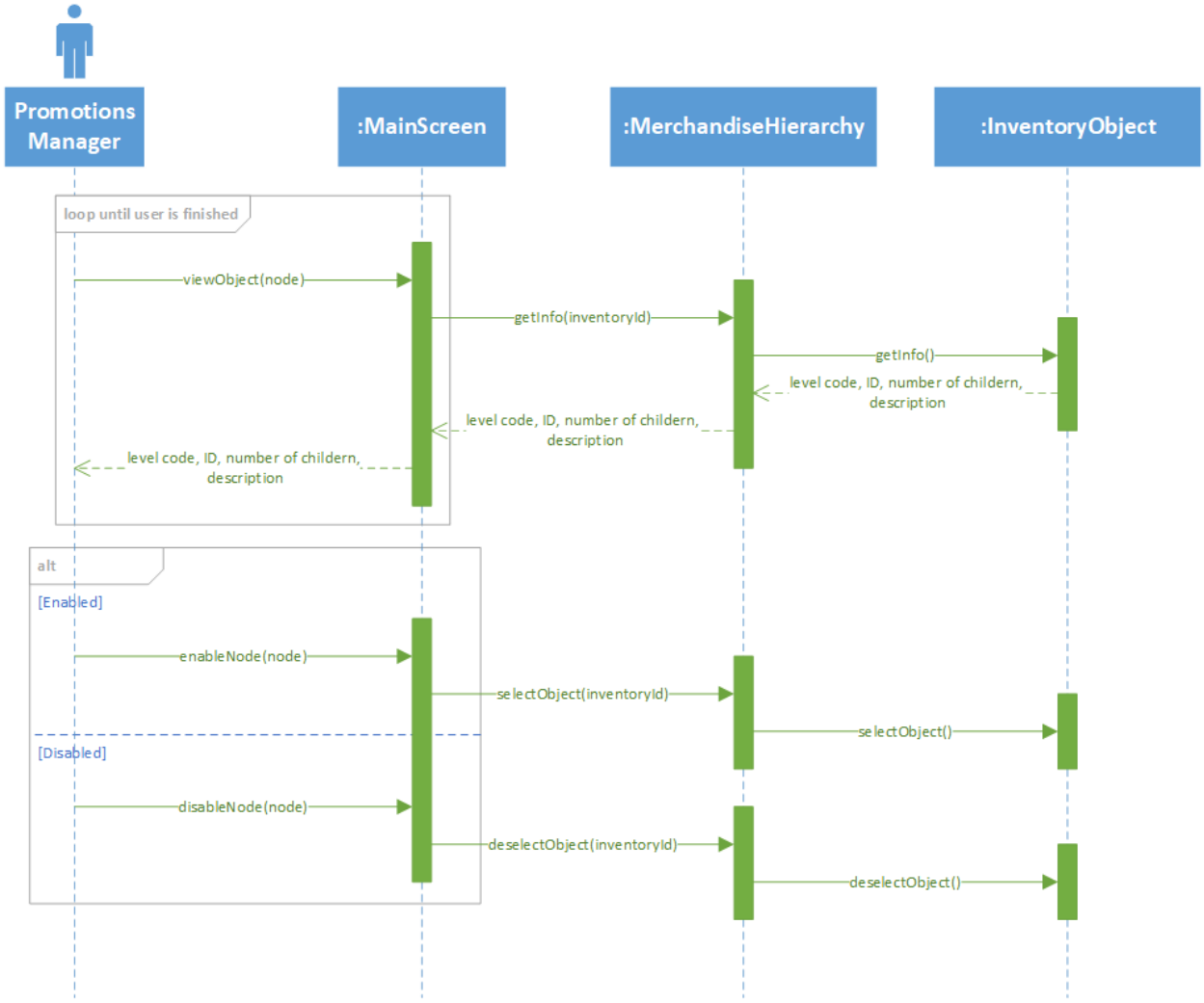
Import database



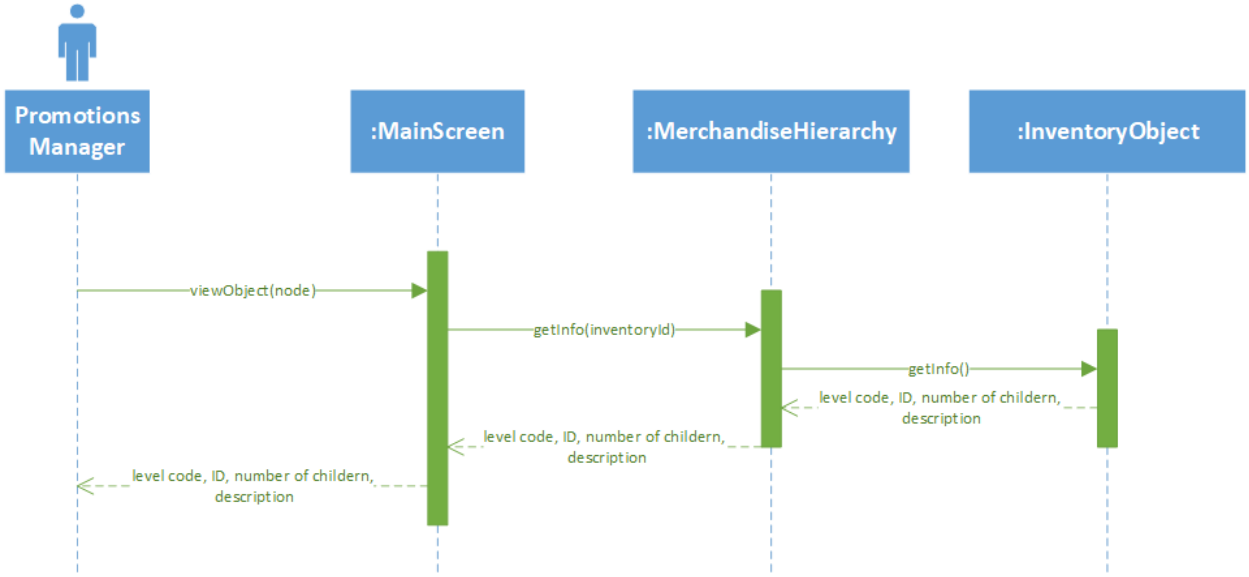
Browse objects



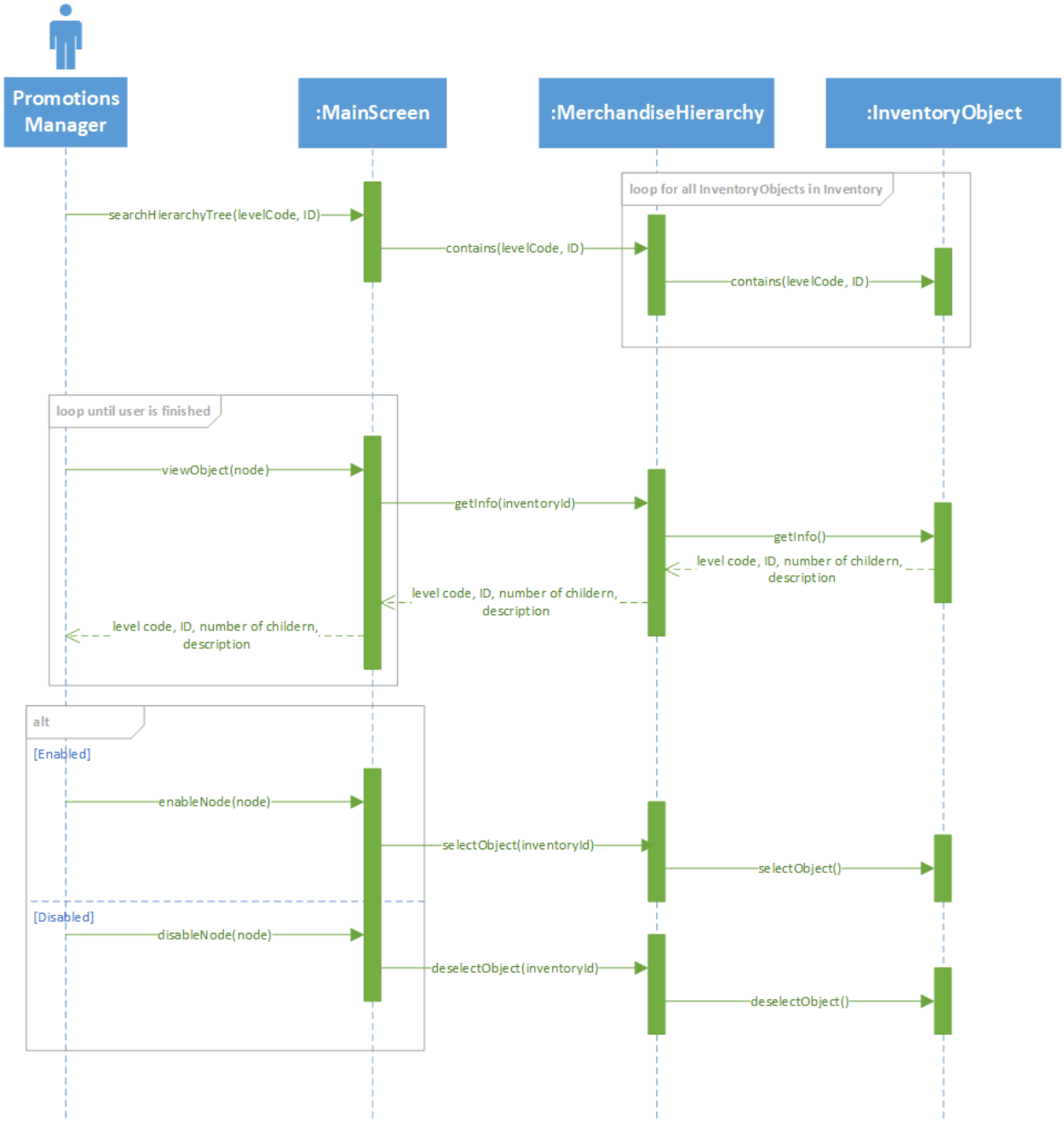
Select objects



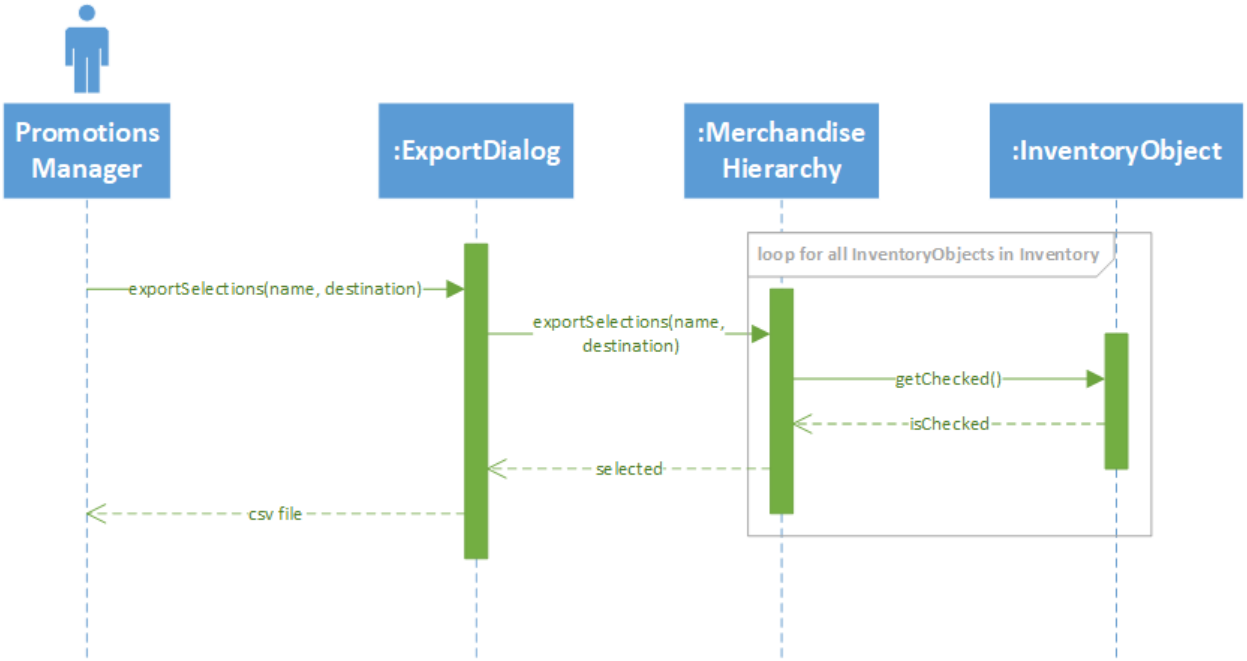
View an object



Search for objects

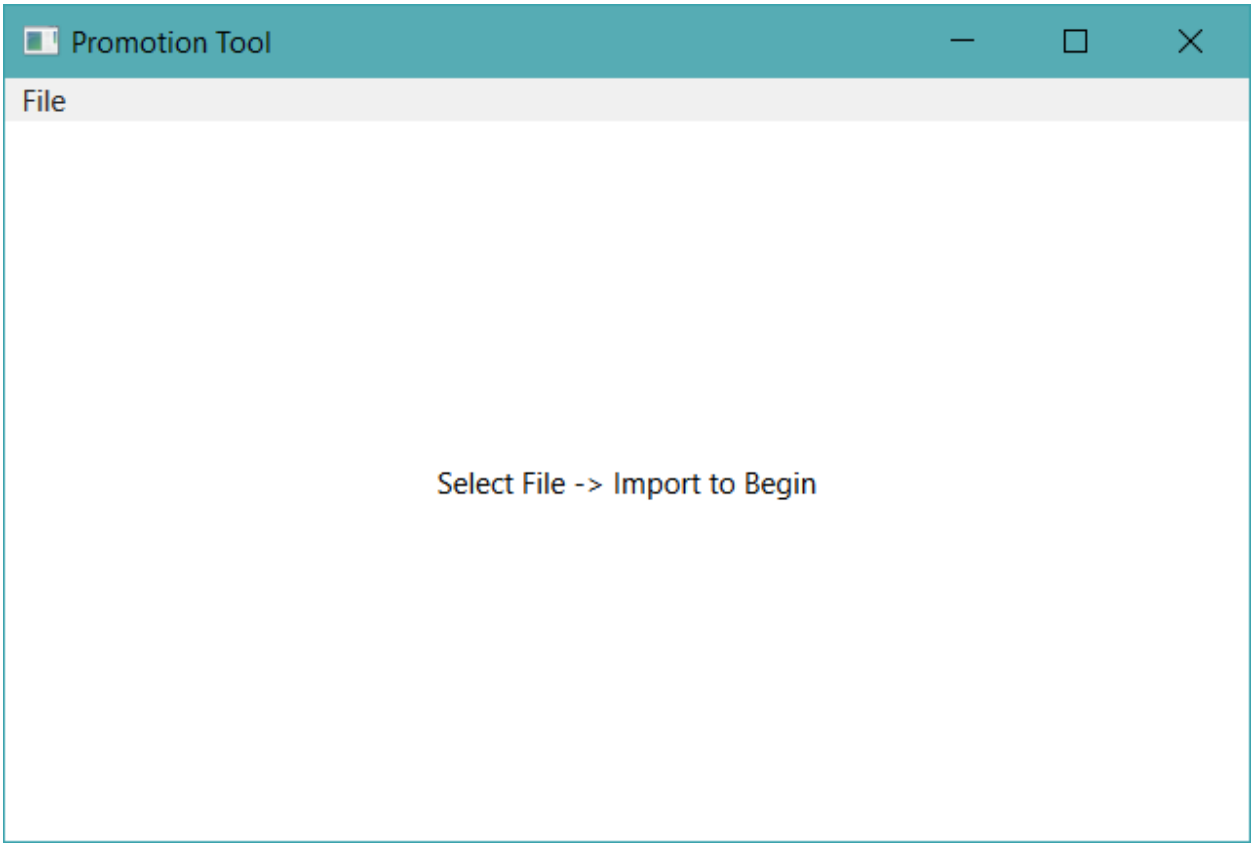


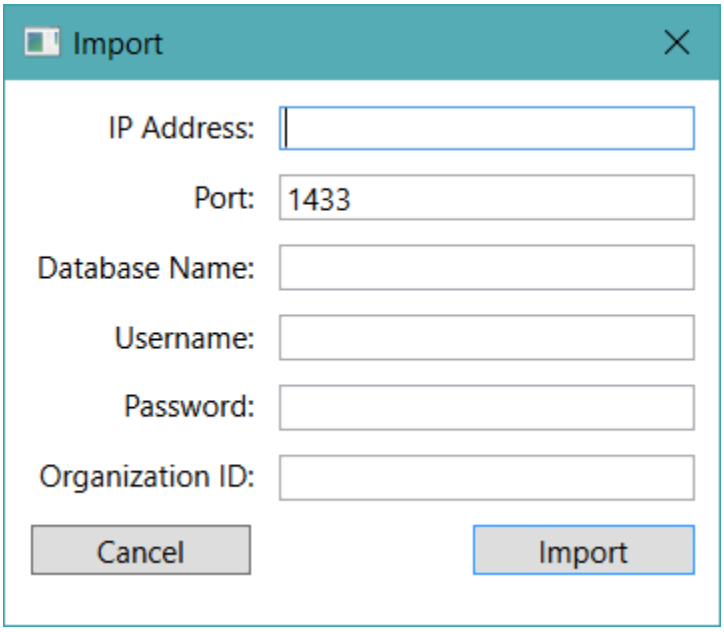
Export CSV file

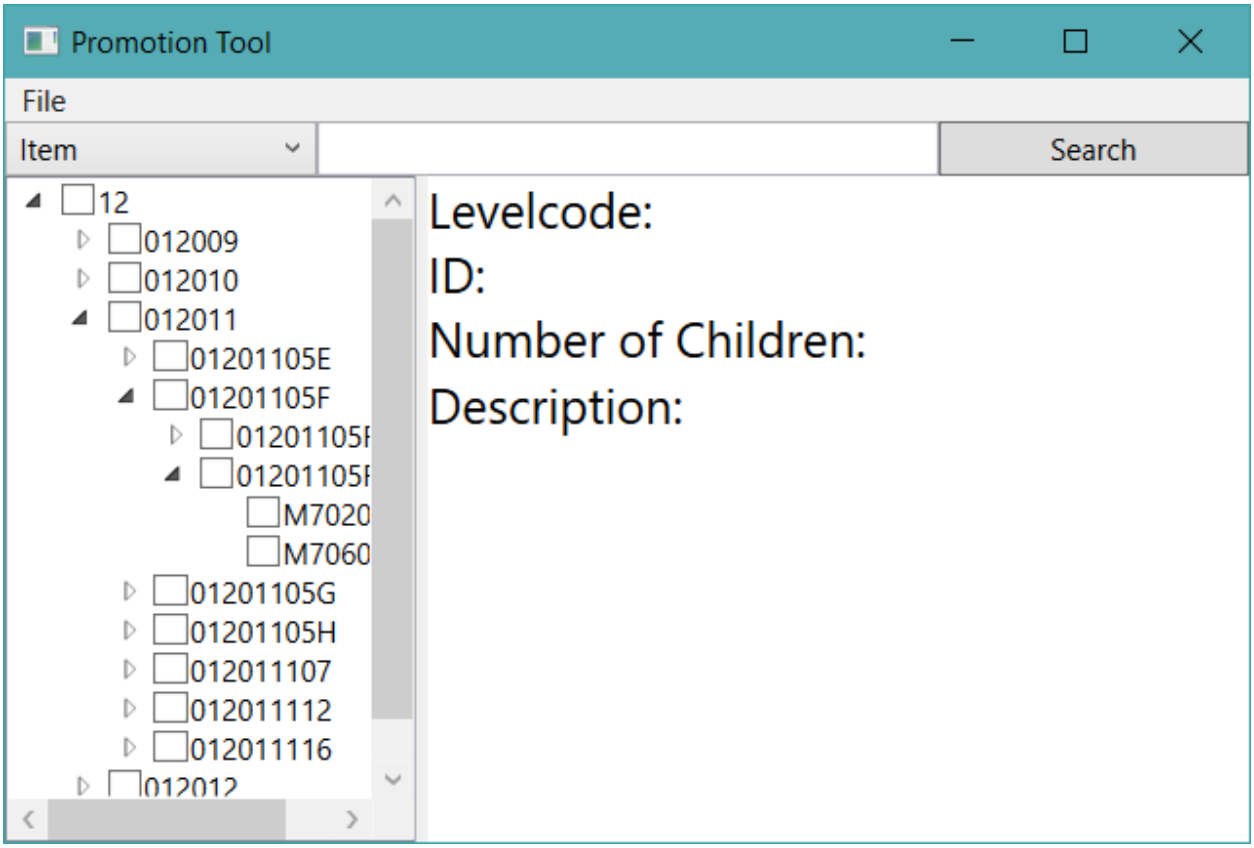


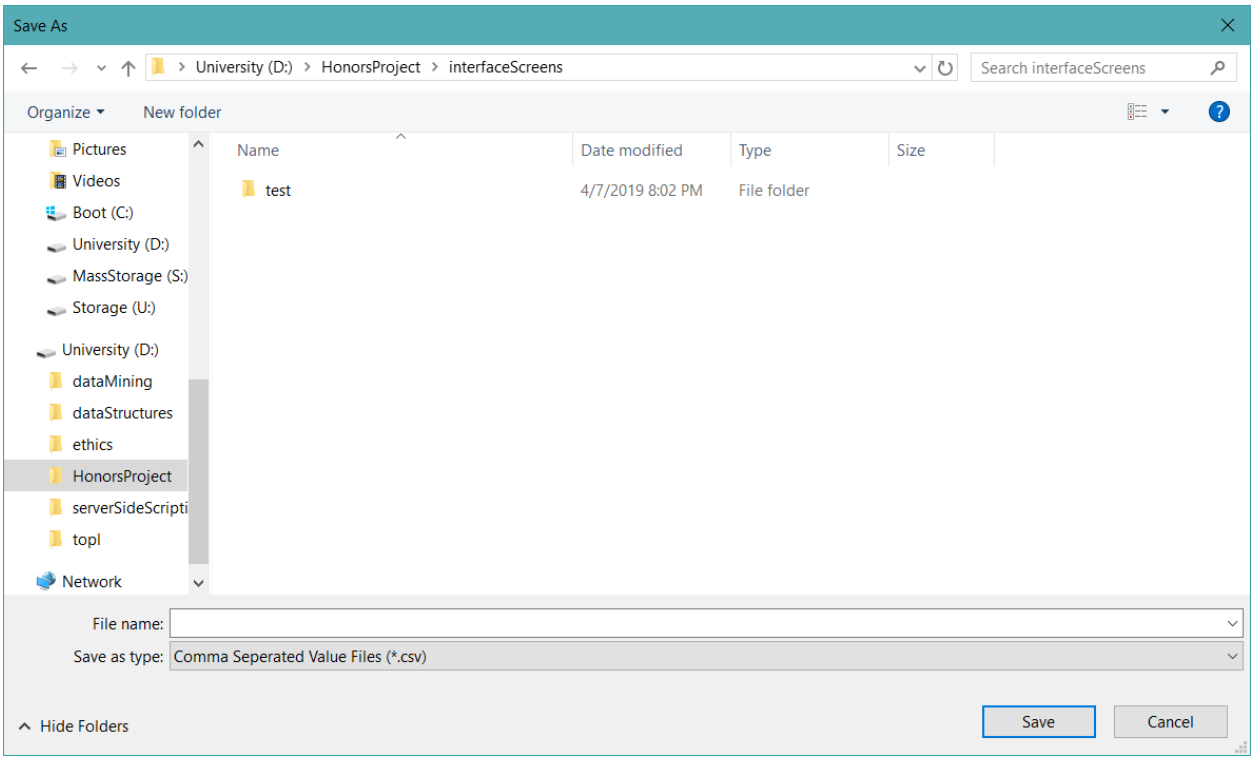
Documentation of Interface

Interface at Design Time



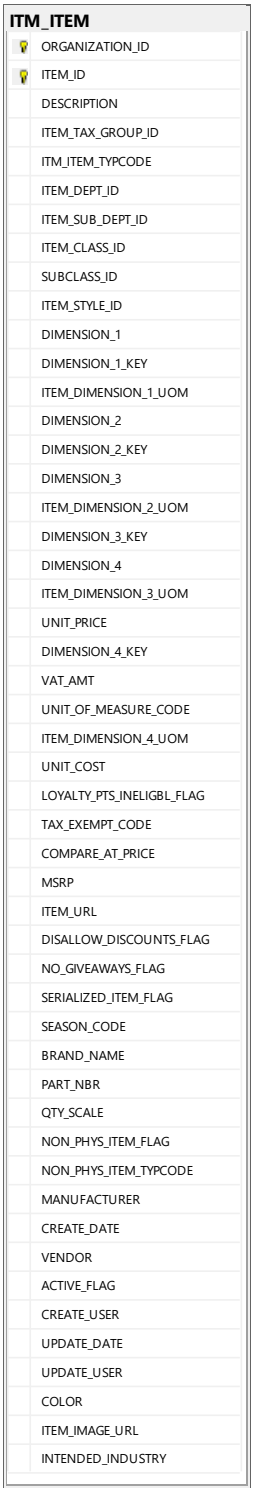


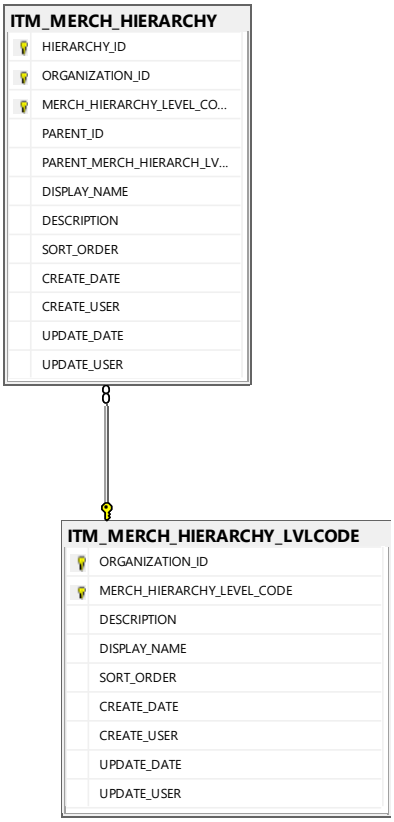

Promotion Tool
File
Item
Search
12
012009
012010
012011
01201105E
01201105F
01201105H
01201105H
M7020
M7060
01201105G
01201105H
012011107
012011112
012011116
012012
Levelcode:
ID:
Number of Children:
Description:


Save As
University (D:) › HonorsProject › interfaceScreens
Search interfaceScreens
Organize
New folder
Pictures
Videos
Boot (C:)
University (D:)
MassStorage (S:)
Storage (U:)
University (D:)
dataMining
dataStructures
ethics
HonorsProject
serverSideScripti
topl
Network
Name
Date modified
Type
Size
test
4/7/2019 8:02 PM
File folder
File name:
Save as type:
Comma Seperated Value Files (*.csv)
Hide Folders
Save
Cancel

Documentation of the Database

Schema of the Tables





Design for any Screens, Queries, Forms, Reports, and Programs

None

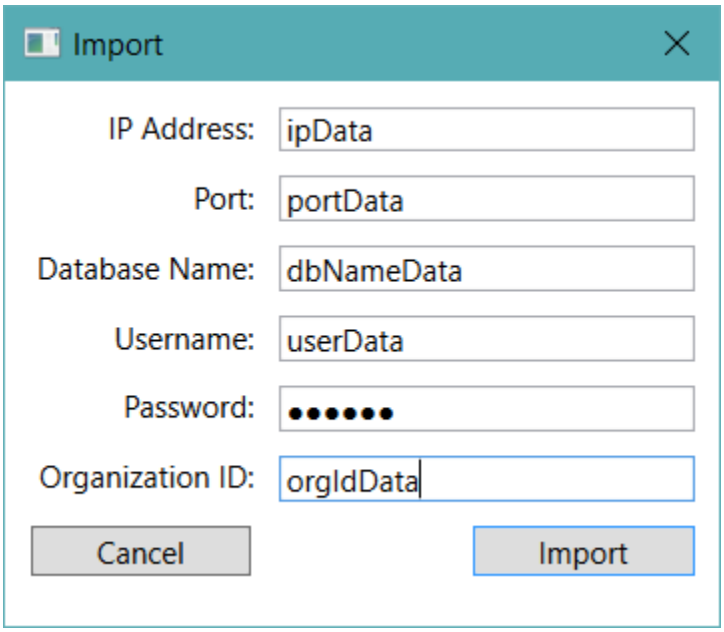
Printout of Structure and Contents of Database Files

Relate	S:\SQLServer\MSSQL14.MSSQLSERVER\MSSQL\DATA\Relate_3.ldf
Relate	S:\SQLServer\MSSQL14.MSSQLSERVER\MSSQL\DATA\Relate_1.ndf
Relate	S:\SQLServer\MSSQL14.MSSQLSERVER\MSSQL\DATA\Relate_2.ndf

Test Data

Invalid Data

The following data is input for the Import window and is used to test database connectivity and loading data from a database to convert to Inventory Objects. It is in the format (IP Address = “ipData”, Port = “portData”, Database Name = “dbNameData”, Username = “userData”, Password = “pwData”, Organization ID = “orgIdData”) which correlates to the following entry in the Import screen:



1. (IP Address = “thisiswrong”, Port = “1433”, Database Name = “relate”, Username = “”, Password = “”, Organization ID = “123”)

   Used to test the behavior of a connection string with an incorrect IP address.

2. (IP Address = “localhost”, Port = “thisiswrong”, Database Name = “relate”, Username = “”, Password = “”, Organization ID = “123”)

   Used to test the behavior of a connection string with an incorrect port.

3. (IP Address = “localhost”, Port = “1433”, Database Name = “thisiswrong”, Username = “”, Password = “”, Organization ID = “123”)

   Used to test the behavior of a connection string with an incorrect database name.

4. (IP Address = “localhost”, Port = “1433”, Database Name = “relate”, Username = “thisiswrong”, Password = “thisiswrong”, Organization ID = “123”)

   Used to test the behavior of a connection string with incorrect credentials.

5. (IP Address = “localhost”, Port = “1433”, Database Name = “relate”, Username = “”, Password = “”, Organization ID = “thisiswrong”)

Used to test the behavior of a connection string with a nonexistent organization ID.

6. (IP Address = “”, Port = “1433”, Database Name = “relate”, Username = “”, Password = “”, Organization ID = “123”)

   (IP Address = “localhost”, Port = “”, Database Name = “relate”, Username = “”, Password = “”, Organization ID = “123”)

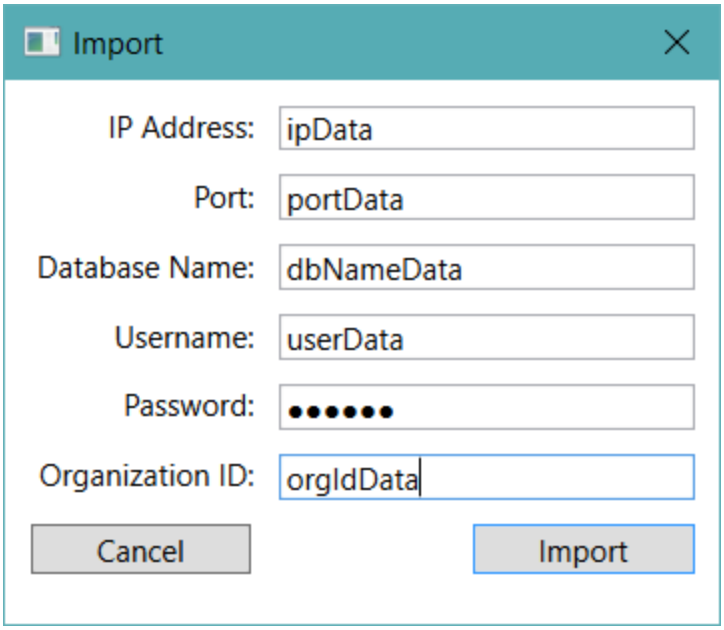
   (IP Address = “localhost”, Port = “1433”, Database Name = “”, Username = “”, Password = “”, Organization ID = “123”)

   (IP Address = “localhost”, Port = “1433”, Database Name = “relate”, Username = “”, Password = “”, Organization ID = “”)

   Used to ensure the program prompts the user to enter mandatory fields.

Valid Data

The following data is input for the Import window and is used to test database connectivity and loading data from a database to convert to Inventory Objects. It is in the format (IP Address = “ipData”, Port = “portData”, Database Name = “dbNameData”, Username = “userData”, Password = “pwData”, Organization ID = “orgIdData”) which correlates to the following entry in the Import screen:



1. (IP Address = “localhost”, Port = “1433”, Database Name = “relate”, Username = “”, Password = “”, Organization ID = “123”)

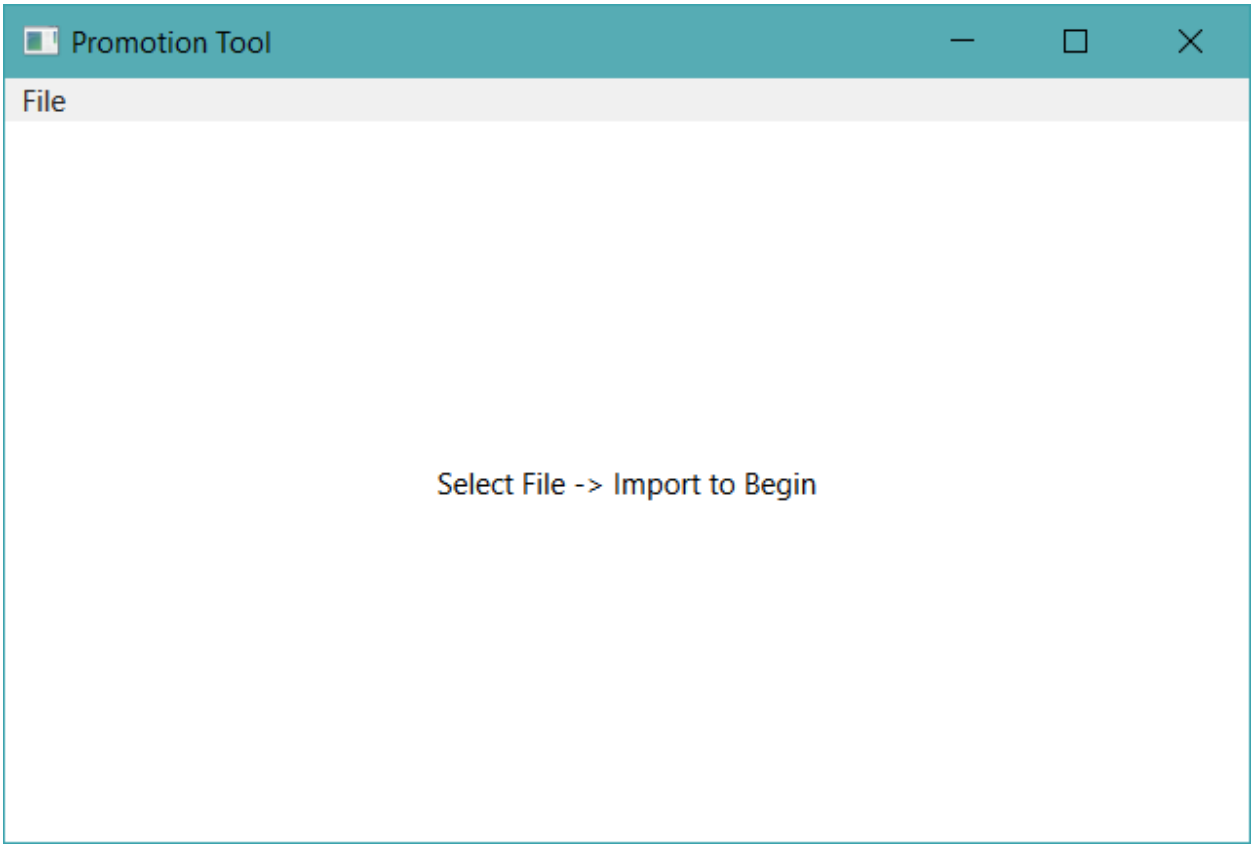
   Used to test the behavior of a correct connection string, Inventory Object creation, and tree population.

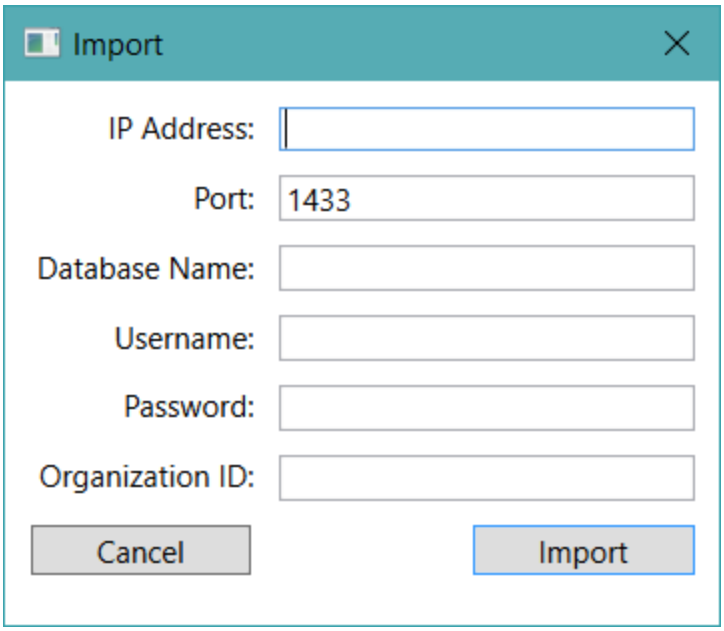
2. (IP Address = “localhost”, Port = “1433”, Database Name = “relate”, Username = “”, Password = “”, Organization ID = “402”)

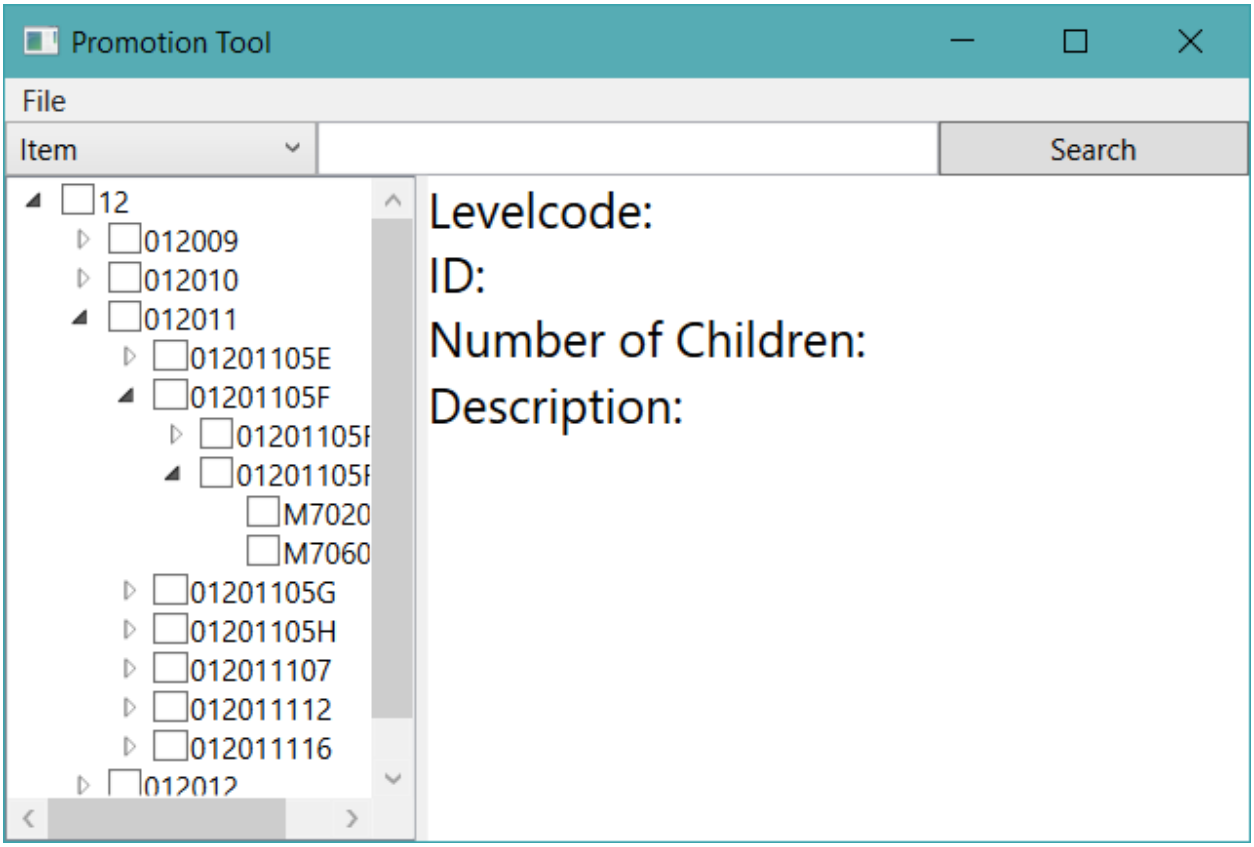
Used to test the behavior of a correct connection string, Inventory Object creation, and tree population. NOTE: the 402 organization contains ill-formatted data and data anomalies used to test the reliability of Relate implementations. The program should still perform expectedly despite said errors when organization 402 is imported.

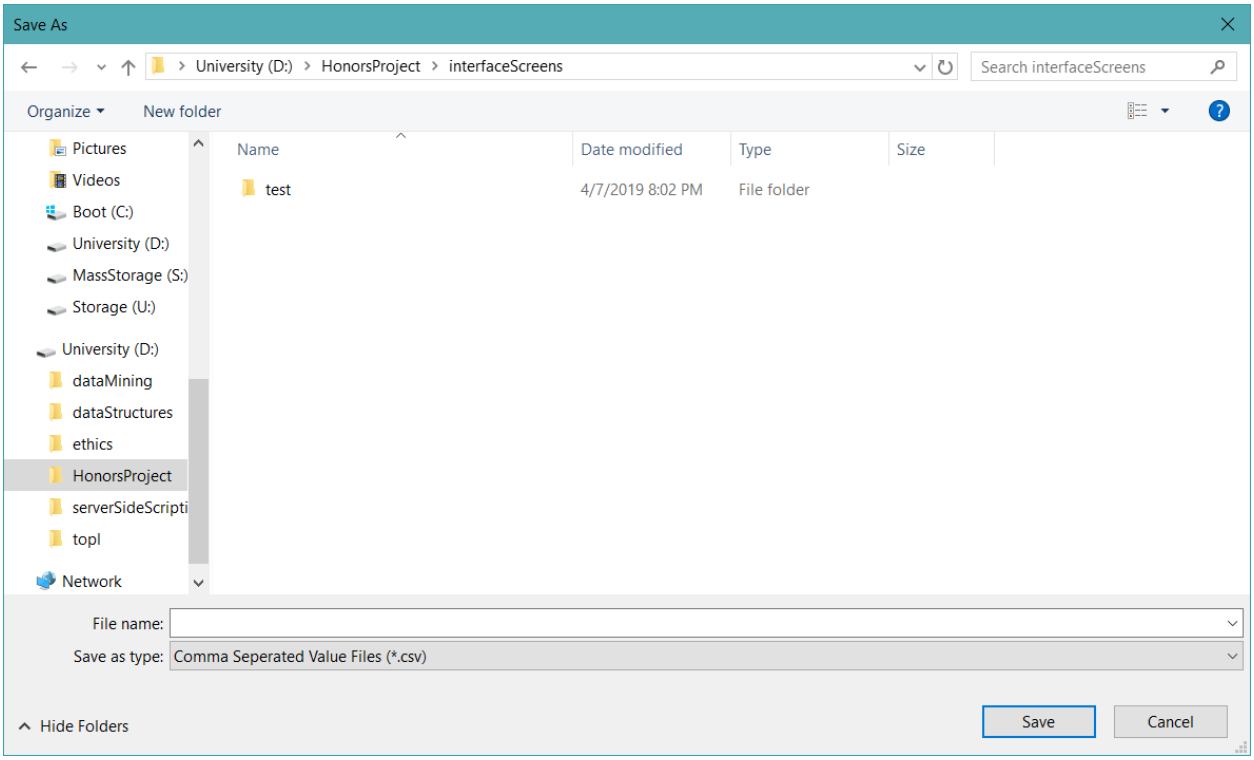
Results

Interface at Runtime




Import
IP Address:
Port:
1433
Database Name:
Username:
Password:
Organization ID:
Cancel
Import

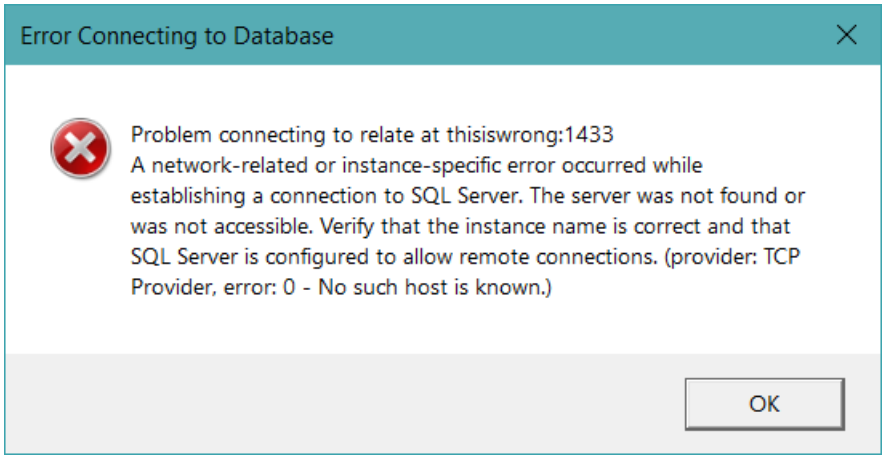

Promotion Tool
File
Item
Search
12
012009
012010
012011
01201105E
01201105F
M7020
M7060
01201105G
01201105H
012011107
012011112
012011116
012012
Levelcode:
ID:
Number of Children:
Description:



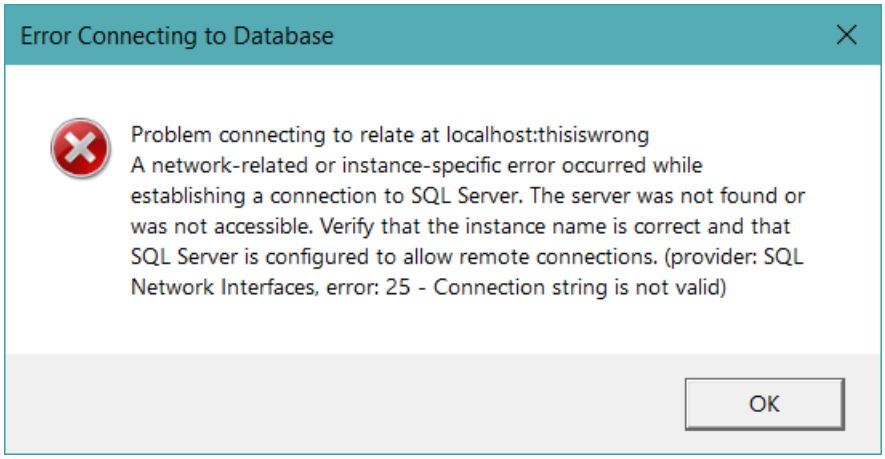
Examples of All Outputs

The following outputs correlate to the correspondingly numbered test input in the invalid data section:

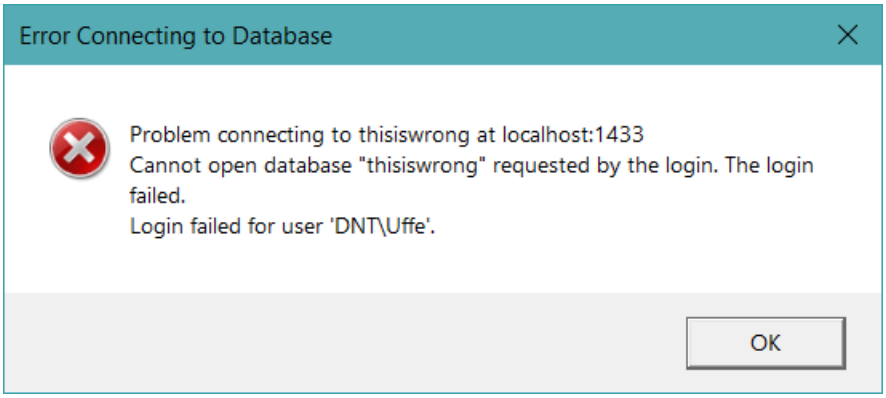
1.



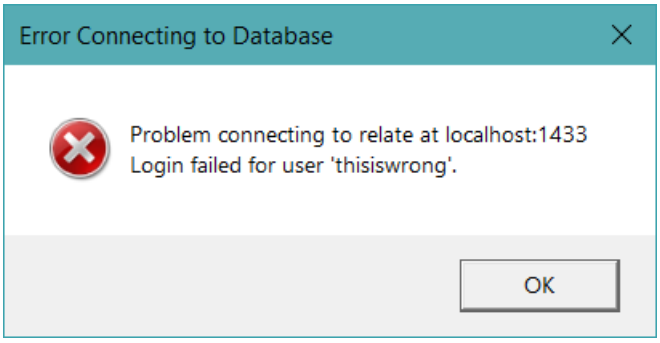
2.


Error Connecting to Database
Problem connecting to relate at localhost:thisiswrong
A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. (provider: SQL Network Interfaces, error: 25 - Connection string is not valid)
OK

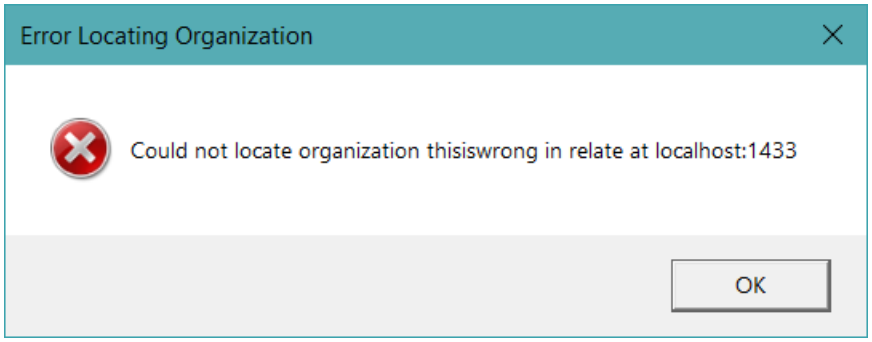
3.


Error Connecting to Database
Problem connecting to thisiswrong at localhost:1433
Cannot open database "thisiswrong" requested by the login. The login failed.
Login failed for user 'DNT\Uffe'.
OK

4.

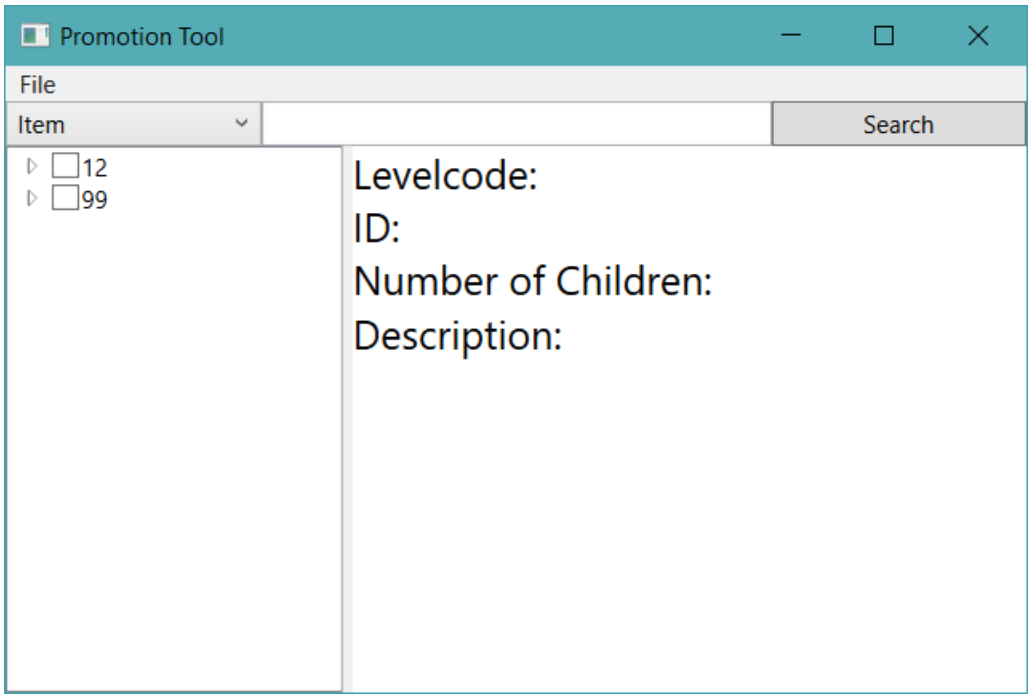

Error Connecting to Database
Problem connecting to relate at localhost:1433
Login failed for user 'thisiswrong'.
OK

5.

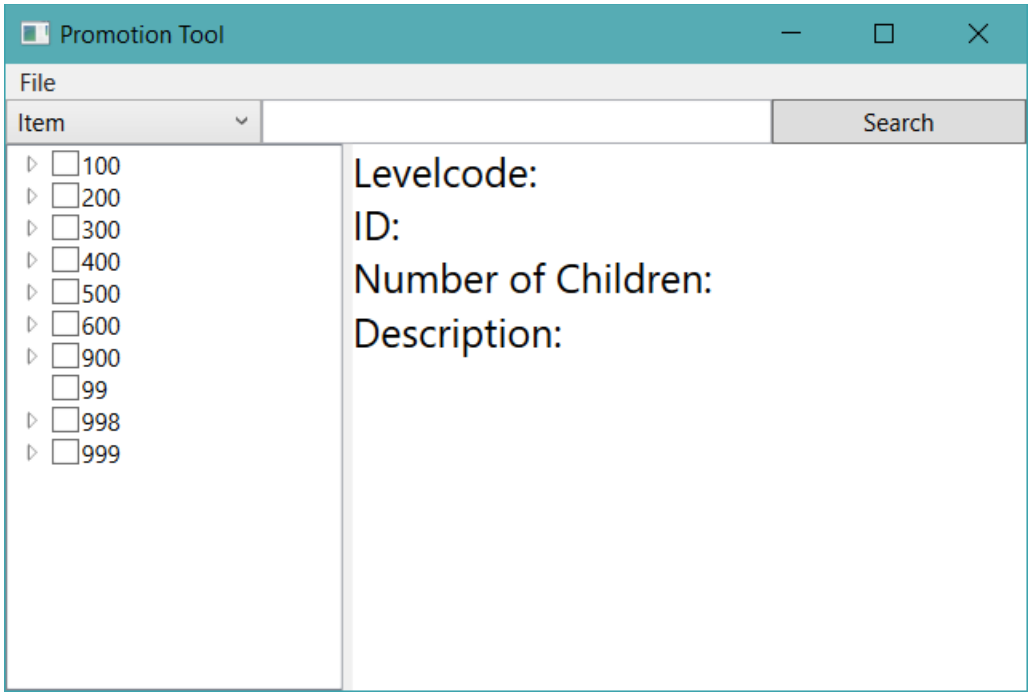


The following outputs correlate to the correspondingly numbered test input in the valid data section:

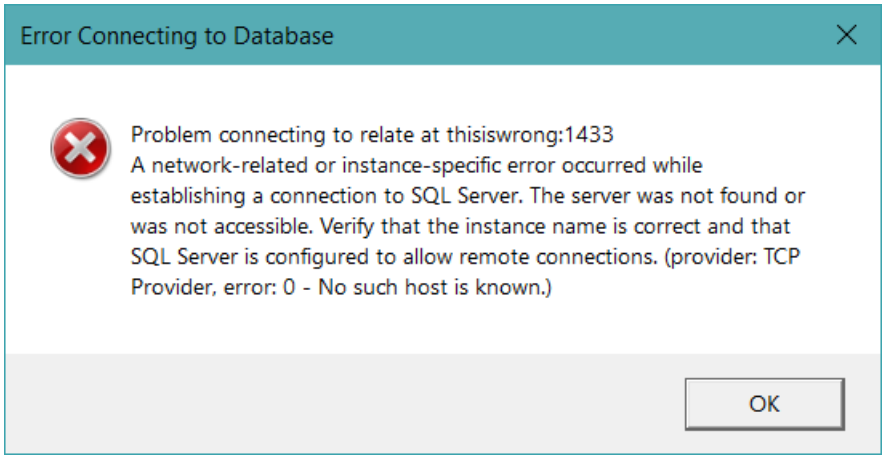
1.

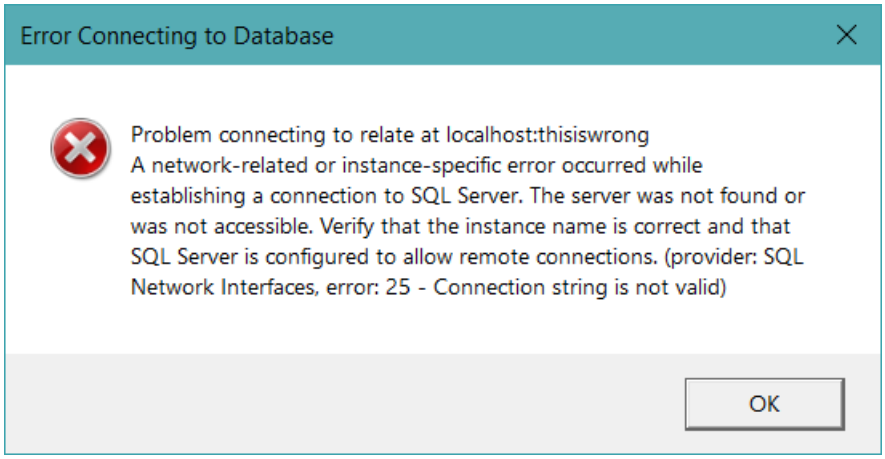


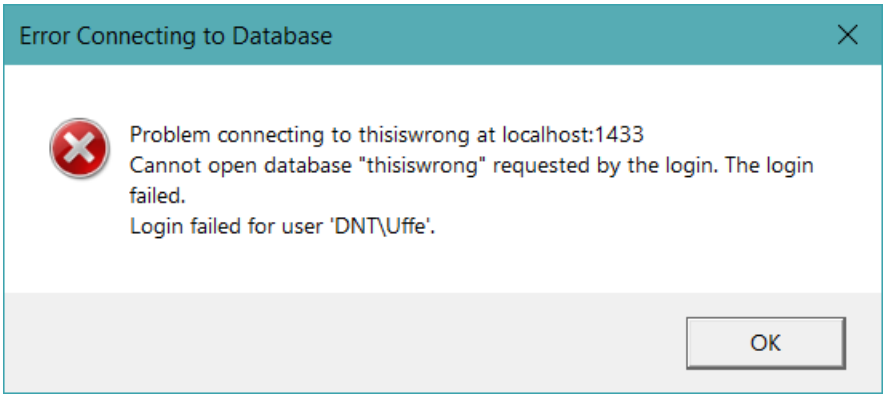
2.

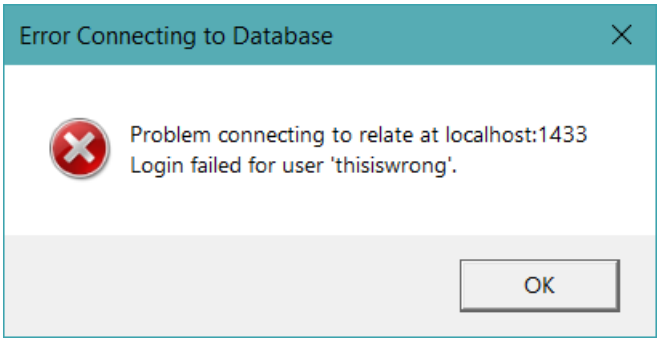


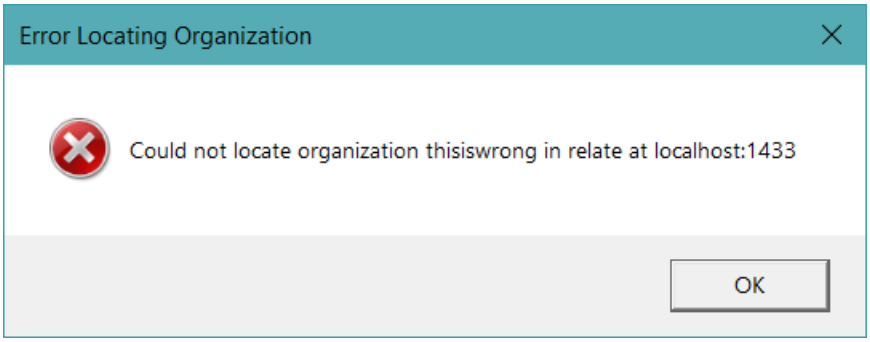
All Error Messages the Program Generates




Error Connecting to Database
Problem connecting to relate at localhost:thisiswrong
A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. (provider: SQL Network Interfaces, error: 25 - Connection string is not valid)
OK


Error Connecting to Database
Problem connecting to thisiswrong at localhost:1433
Cannot open database "thisiswrong" requested by the login. The login failed.
Login failed for user 'DNT\Uffe'.
OK


Error Connecting to Database
Problem connecting to relate at localhost:1433
Login failed for user 'thisiswrong'.
OK


Error Locating Organization
Could not locate organization thisiswrong in relate at localhost:1433
OK

Conclusion

In regard to the requirements stated previously, I have succeeded on all accounts. The program allows for much easier browsing and selection of inventory objects, as well as the ability to view a queried object's position in the merchandise hierarchy in to order to view said object's children and ancestors.

On the topic of problems and/or mistakes encountered during the development of the program, the largest and most time-consuming was the decision to use the Model View View-Model pattern. Initially I had planned on separating the content from the user interface by storing the information queried from the database in a collection and then generating view-model objects based on that collection to populate the UI tree. This was meant to serve as both a learning exercise and good practice; however, it was found to be very poor practice in implementation for reasons that I will elaborate further on. Firstly, due to the large size of the data being imported, the MVVM pattern caused prolonged load times and increased memory usage. Secondly, the MVVM pattern introduced additional complexity to a project that is otherwise fairly straightforward. Based on the subpar performance alone, the inclusion of the MVVM pattern would be an issue worthy of contention if not for the third reason, that is, this implementation of the MVVM pattern did not produce any benefits in this project. This is because it is both impossible for the user to modify any data (except for selection) in both the memory of the program and the imported database and the UI tree is populated through databinding (which implements its own approximation of MVVM).

Regarding what has been learned, it should first be noted that this project has imparted a greater understanding of the software development lifecycle as previous university projects had required limited planning due to their scope and restrictive timeline. This project has been an excellent opportunity to complete all stages of analysis and design as a continuous work on a single project, as well as to follow though the planning with implementation which allows a better understanding of what

aspects of planning are important in relation to actual development tasks. Secondly, the undertaking of this project has granted an opportunity to gather further knowledge of the C# language and the .NET environment. One such example being the use of Windows Presentation Foundation (WPF), which is a modern UI platform that allows separation of UX from code and can be used in any desktop application. Another example of a practical piece of knowledge that was acquired as part of the development of this project is how to connect and perform queries against a database in C#, which is an absolutely necessary skill in business environments.

Lastly, in reference to future work and extensions for the application, if a feature is requested by a user and is within scope, for example additional inventory object information or a different way to filter a search, I would be happy to implement it.

Bibliography

Satzinger, John W., et al. *Systems Analysis and Design in a Changing World*. 6th ed., Course Technology, Cengage Learning, 2012.

User's Manual

Importing a Database

The user may begin the import process by clicking the **Import** button under the **File** menu found in the upper right-hand corner of the application (Figure 1).

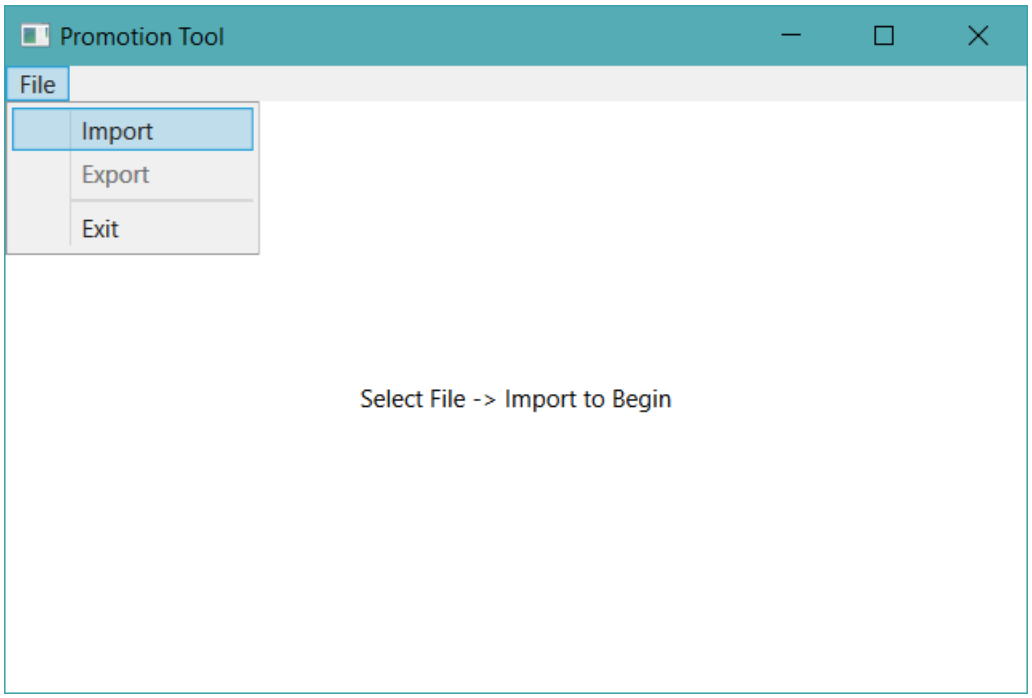


Figure 1: the import button

After clicking the **Import** button, the user will be prompted with the **Import** dialog (Figure 2).

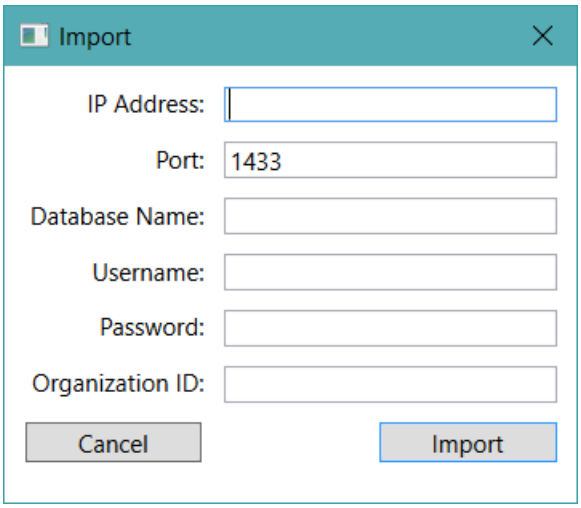


Figure 2: the Import dialog

The user must enter the following information regarding the SQLServer database they desire to connect to into the specified fields: the database's IP address in the **IP Address** field, the database's port in the **Port** field (the default port for SQLServer is 1433 is entered automatically), the database's name in the

Database Name field, and the organization ID of the desired organization in the **Organization ID** field. Additionally, the user must enter the appropriate username into the **Username** field and the respective password into the **Password** field. If the database the user desires to import uses integrated security, the user must leave the **Username** and **Password** fields blank. To begin the import, the user must then click the **Import** button on the **Import** dialog (Figure 2). To cancel the import, the user must click the **Cancel** button on the **Import** dialog.

Viewing Inventory Items

After completing the import process, the user is shown the **Main** screen (Figure 3).

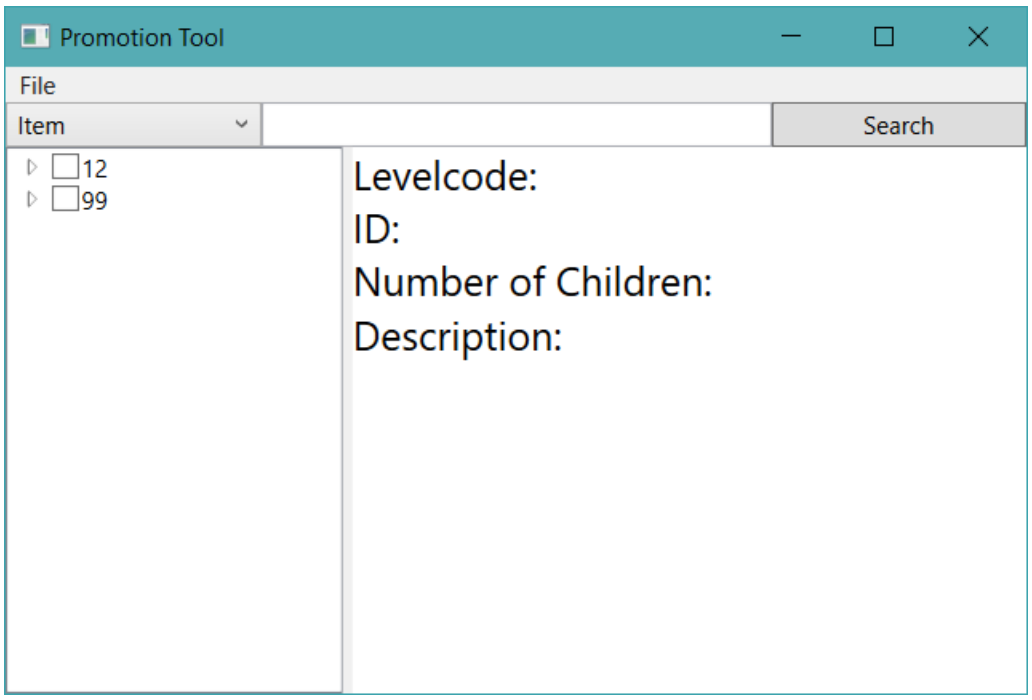


Figure 3: the Main screen

The Main screen is divided in to three sections: The **Inventory Tree**, the **Object Panel**, and the **Search Bar**. These are delineated in Figure 4.

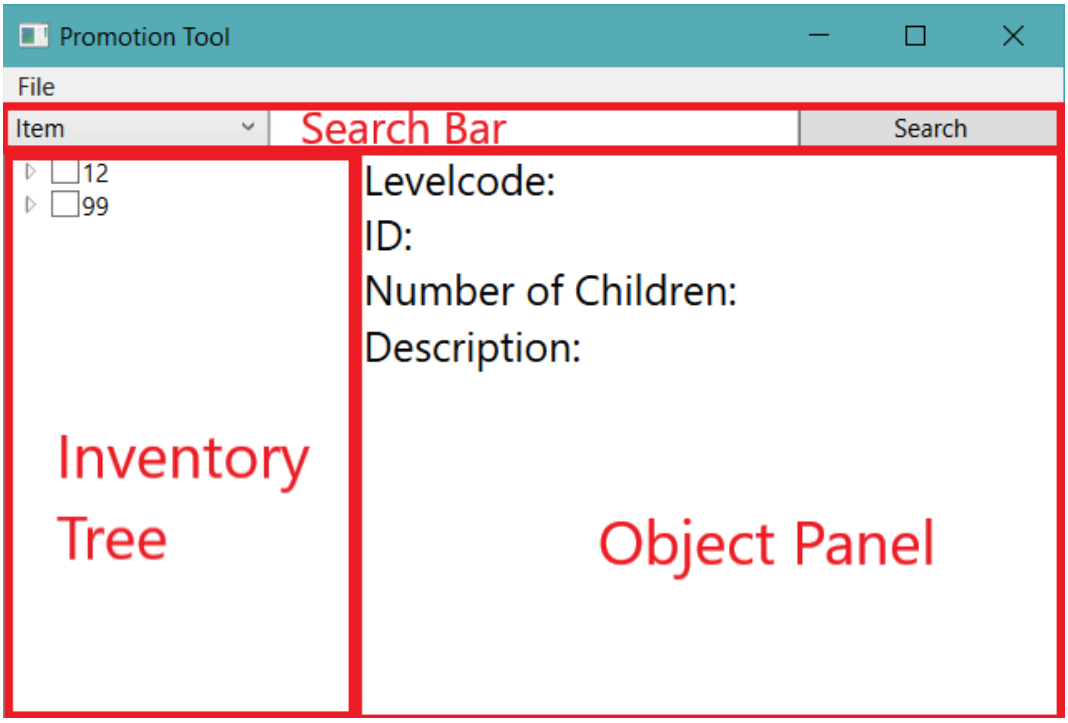


Figure 4: the sections of the Main screen

To view an inventory object's properties, the user must highlight the inventory object in the **Inventory Tree** by clicking on the inventory object's ID. The highlighted inventory object's properties will be displayed in the **Object Panel** (Figure 5).

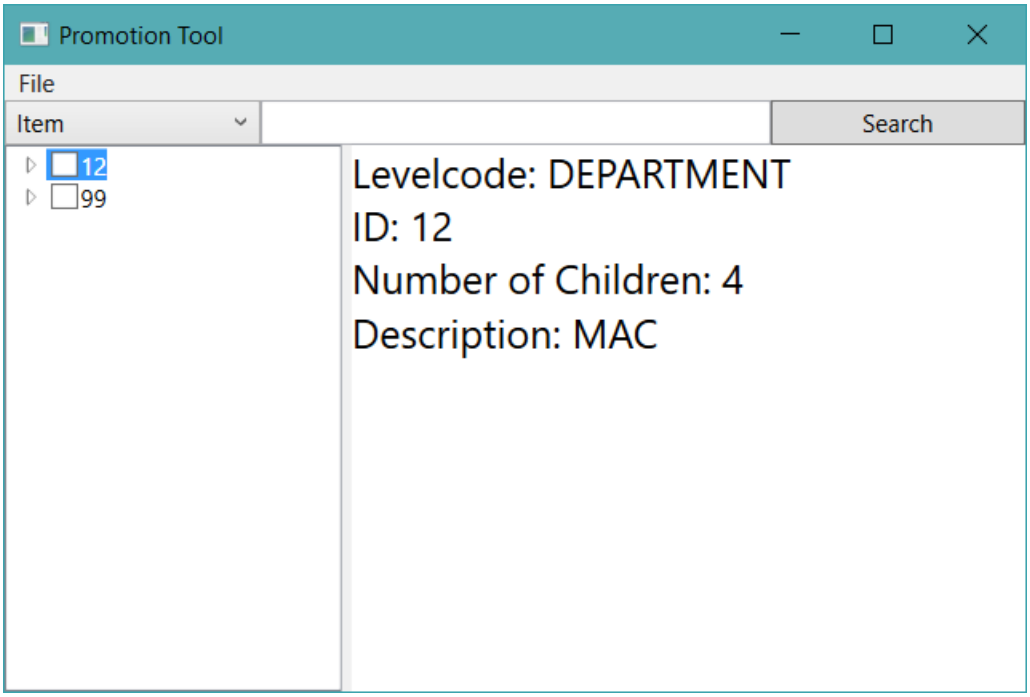


Figure 5: the Main screen with an inventory object highlighted

To view an inventory object's child objects, the user must click the caret adjacent to the inventory object's ID in the **Inventory Tree** (Figure 6).

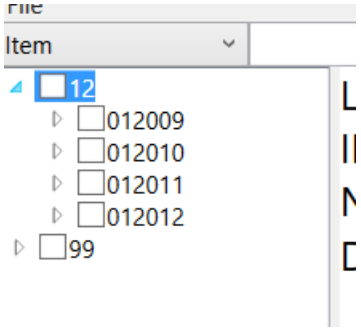


Figure 6: an inventory object's children and its caret

To view an inventory object's description without highlighting it, the user must hover the mouse cursor over the inventory object's ID in the **Inventory Tree** (Figure 7).

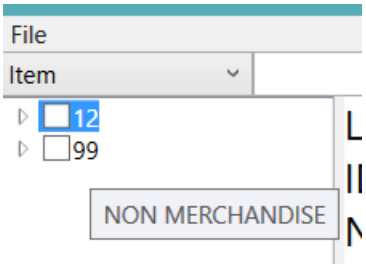


Figure 7: an inventory object's description as displayed when the cursor is hovering over it (mouse cursor not shown)

Selecting Inventory Items

To select an inventory object, the user must click the checkbox adjacent to the desired inventory object's ID (Figure 8) in the **Inventory Tree** on the Main screen (Figure 4).

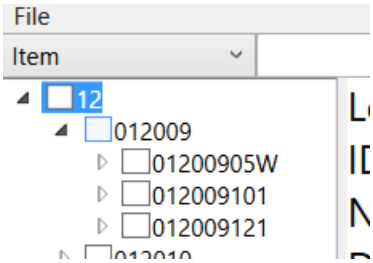


Figure 8: an inventory object's checkbox

NOTE: Selecting an inventory object will select all that object's children (Figure 9).

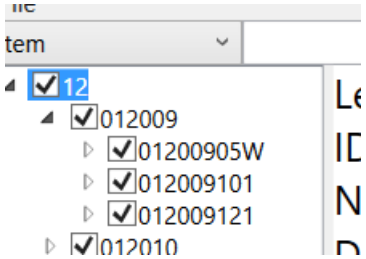


Figure 9: the result of selecting the inventory object with ID 12

Deselecting an inventory object will deselect all that object's children and that object's ancestors (Figure 10).

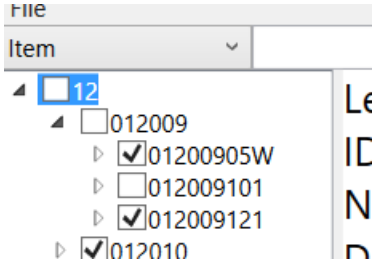


Figure 10: the result of deselecting the inventory object with ID 012009101

Searching for Inventory Objects

The user may search for an inventory object with a specific ID by using the **Search Bar** (Figure 9). To search for an inventory object, the user must enter the desired inventory object's ID into the center field and select the desired category from the dropdown located to the left of the center field. Once the user has entered an ID and selected a category, they must click the **Search** button.

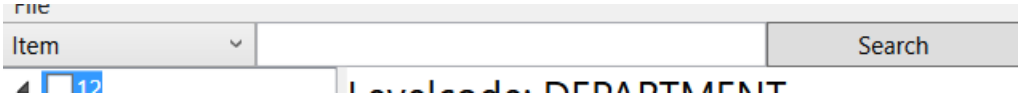


Figure 11: the Search Bar

The results of a search will be shown in the Inventory Tree and will contain the object with a matching ID and category, that object's children, and that object's ancestors (Figure 12).

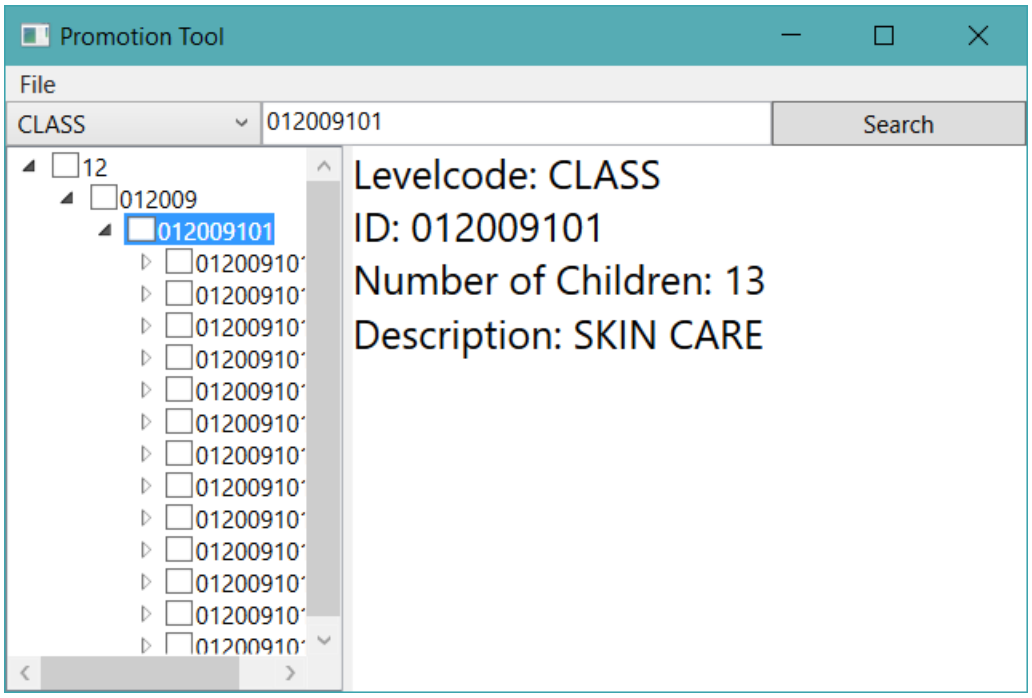


Figure 12: the results of searching for the ID 012009101 and the category CLASS

To view the Inventory Tree in its entirety after performing a search, the user must clear the center text field in the **Search Bar** and select the **Search** button.

Exporting Selections

To export the selections a user has made, the user must click the **Export** button under the **File** menu found in the upper right-hand corner of the application (Figure 13).

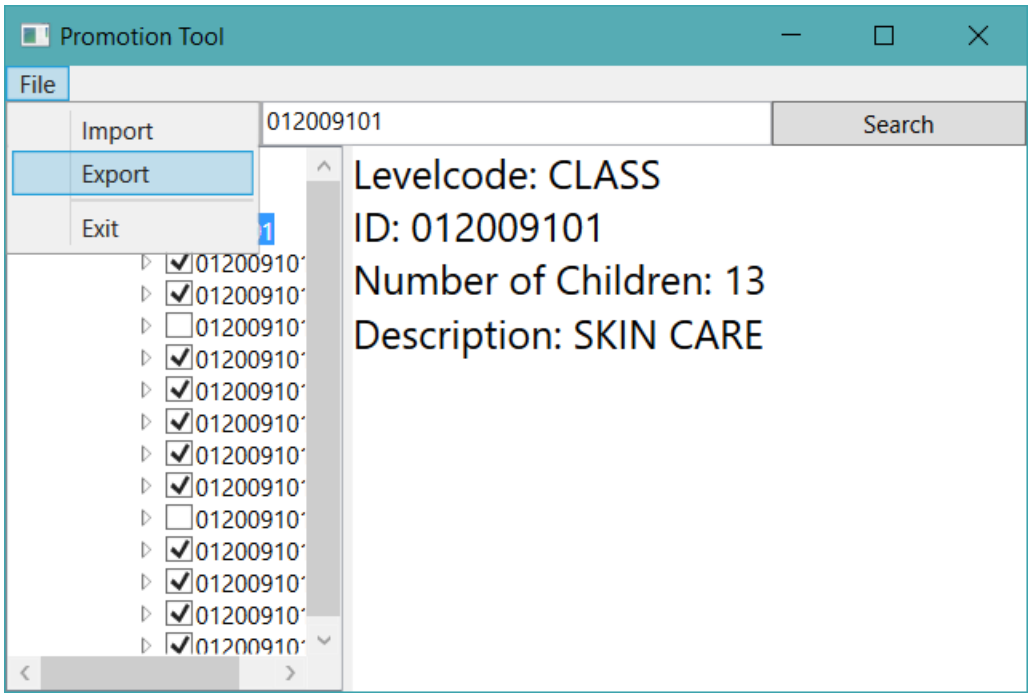


Figure 13: the Export button

After clicking the **Export** button, the user will be prompted with the **Export** dialog (Figure 14).

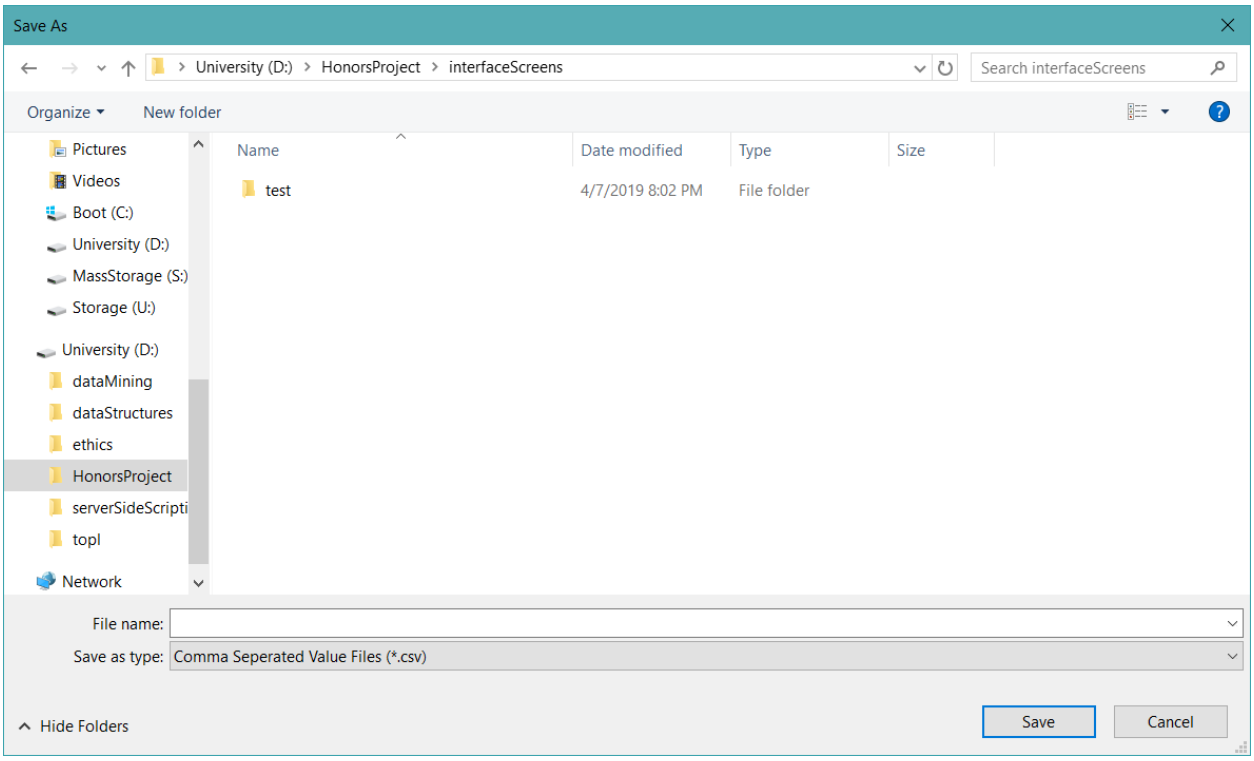


Figure 14: the Export dialog

The user must enter a file name in the **File Name** field and select a location in their file system. When the user has entered a file name and selected a location, they must click the **Save** button to finish the export process.