XML Graphical Creation Tool

Ismael Abdulrashid, Cameron Jewell, Aamir Rafi, Basiru Usman

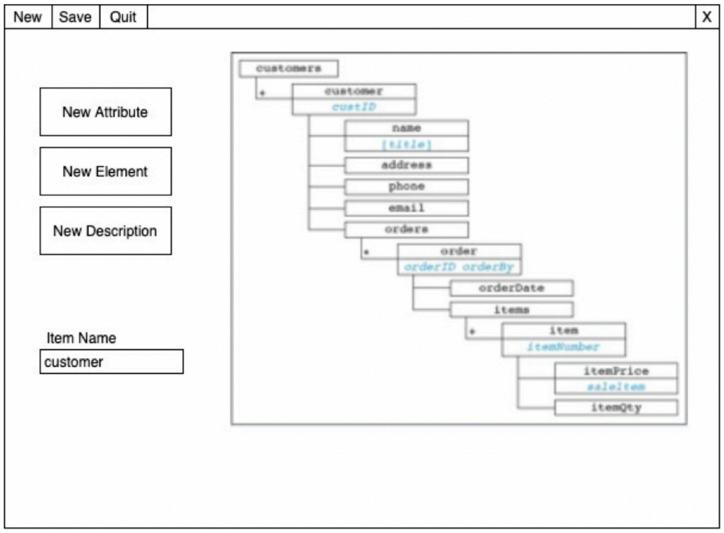
Introduction

The aim of the project will be to aid an inexperienced user in the creation of XML style schema documents with a tool simply and intuitively. The GUI will provide the user with an easy to understand UI with buttons and text boxes to essentially drag and drop portions of a valid, would be schema into place without having to write any XML code.

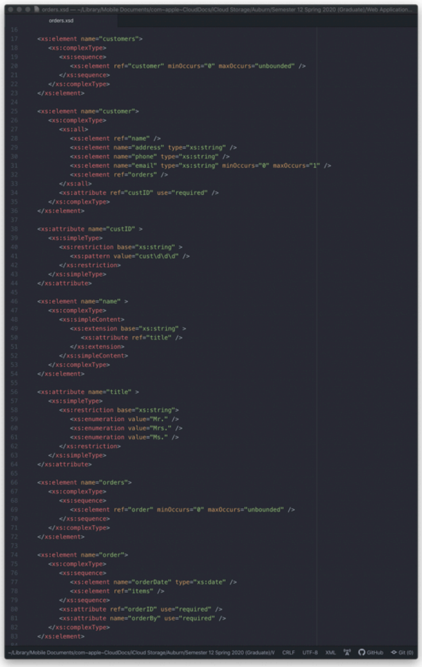
The target audience for this application would be users who are new to XML, have little to no experience using an IDE, wish to craft a more naturally complicated schema more efficiently (both in time and detail), or some combination of those aforementioned.

Design

We've taken an optimistic approach to the design and development of the GUI. Originally, we intended to use Python as a shell for the physical pieces of the GUI written in Qt, but we were discouraged away from Python and on to Java. All code to be used for translation of physical elements of the GUI to XML code snippets will be handled on the backend by Java.



Sample Wireframe


orders.xsd — ~/Library/Mobile Documents/com~apple~CloudDocs/iCloud Storage/Auburn/Semester 12 Spring 2020 (Graduate)/Web Application...
orders.xsd
~/Library/Mobile Documents/com~apple~CloudDocs/iCloud Storage/Auburn/Semester 12 Spring 2020 (Graduate)/V
CRLF
UTF-8
XML
GitHub
Git (0)

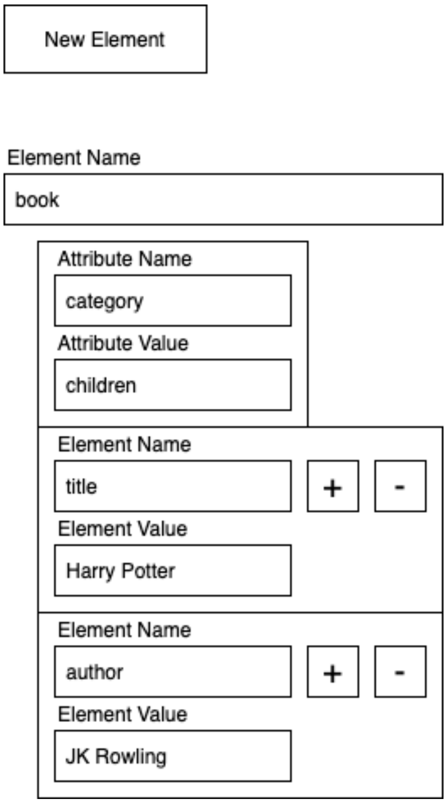
```

<xs:element name="customers">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="customer" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="customer">
   <xs:complexType>
      <xs:all>
         <xs:element ref="name" />
         <xs:element name="address" type="xs:string" />
         <xs:element name="phone" type="xs:string" />
         <xs:element name="email" type="xs:string" minOccurs="0" maxOccurs="1" />
         <xs:element ref="orders" />
      </xs:all>
      <xs:attribute ref="custID" use="required" />
   </xs:complexType>
</xs:element>

<xs:attribute name="custID" >
   <xs:simpleType>
      <xs:restriction base="xs:string" >
         <xs:pattern value="cust\d\d\d" />
      </xs:restriction>
   </xs:simpleType>
</xs:attribute>

<xs:element name="name" >
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:string" >
            <xs:attribute ref="title" />
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>

<xs:attribute name="title" >
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:enumeration value="Mr." />
         <xs:enumeration value="Mrs." />
         <xs:enumeration value="Ms." />
      </xs:restriction>
   </xs:simpleType>
</xs:attribute>

<xs:element name="orders">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="order" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="order">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="orderDate" type="xs:date" />
         <xs:element ref="items" />
      </xs:sequence>
      <xs:attribute ref="orderID" use="required" />
      <xs:attribute name="orderBy" use="required" />
   </xs:complexType>
</xs:element>
```

Given the time constraints and the unique situation this semester has come to reside in, we have settled on a rich feature set that does not totally encompass the complexities and depth XML has to offer in its entirety but does offer significant user functionality. The proposed feature set will allow the basic user to create a strong foundation for what can become a challenging and rigorous XML document/schema.

Ideally, and given enough time, the application would be able to validate any XML file you could throw into the system. One could open/import an XML file and begin editing from there rather than from scratch. With the required changes to the background processes (no python), the Qt portion of the project will be significantly more challenging. As such, and having worked with the intricacies of file revision, we have determined this is overstepping the scope of this course.

Features

Elements - everything from (including) the element's start tag to (including) the element's end tag. An element can contain text, attributes, other elements, or a mix of the three.



Element Name: book

Attribute Name: category

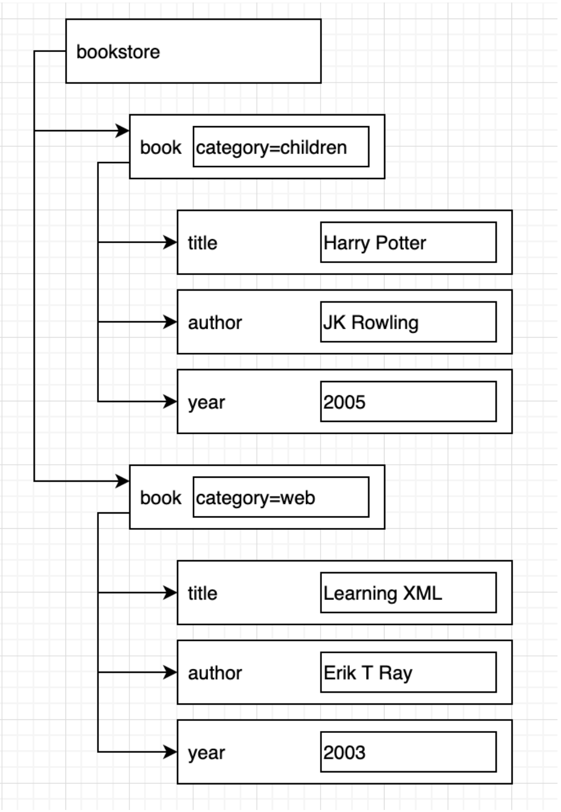
Attribute Value: children

Element Name: title

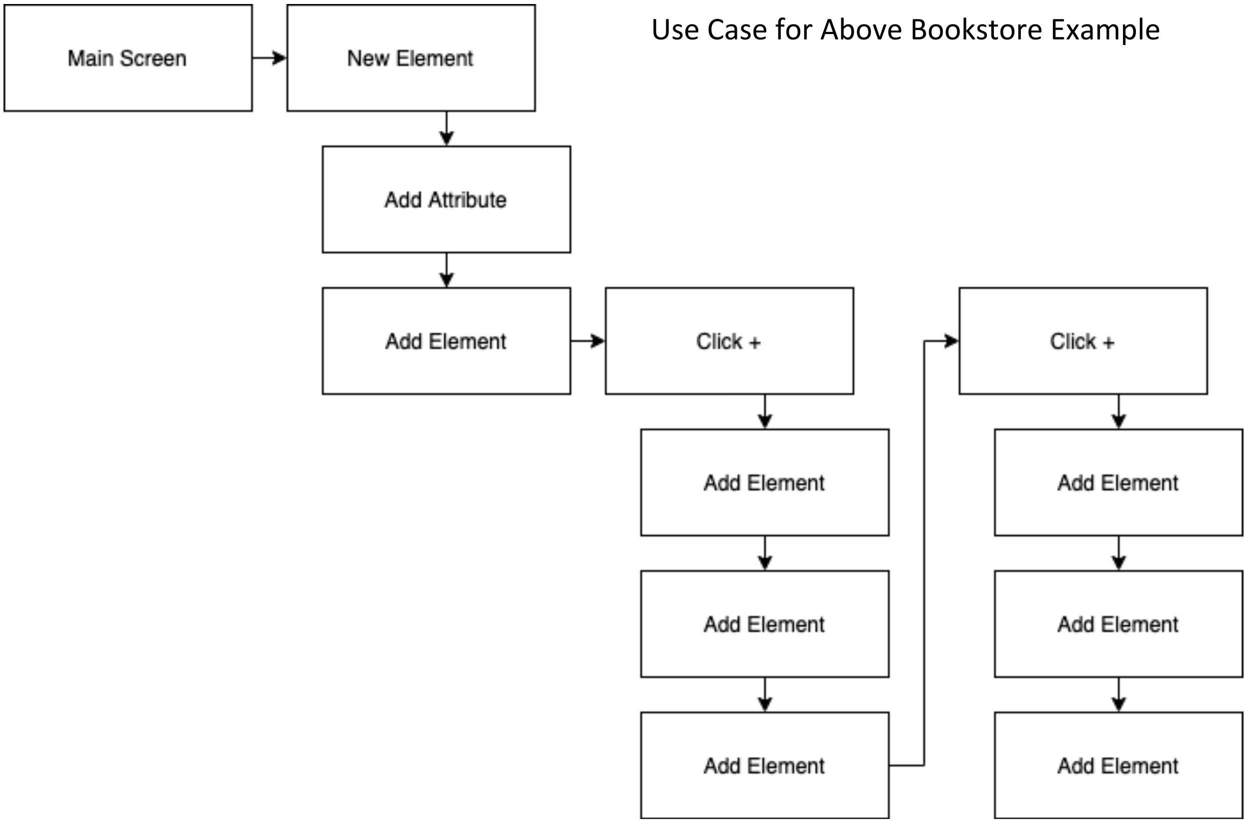
Element Value: Harry Potter

Element Name: author

Element Value: JK Rowling



Pressing the New Element button will prompt the user to give details about that element. Details include its name, attributes, child elements, etc. will be suggested below the text boxes for ease of understanding and convenience. Attributes - designed to contain data related to a specific element. The plus button located next to child attributes and elements will allow for easy addition of more elements.


Use Case for Above Bookstore Example

As new elements are created, the tree on the right-hand side of the application will be updated. To access an element and its children, click the element in the tree on the right-hand side. You will then be able to add or remove elements from the tree.

The application will have a file menu that will include options such as New, Save As, Save, and Quit. The save (as) options will generate an XML file to the specifications of the GUI. The file can be altered by the user after having been generated to obtain more specific details or supported XML options that happen to not be in the GUI itself.

Technology

There will be several technologies that going to be used to create the application. Qt will be used to create the GUI and allow the user to interact with the interface itself. XML will be used define the structure of the data outputted between user creation and file generation. For the functionality part, Java will provide the responsiveness between the application parts and be solely responsible for translation of Qt into XML.

Required languages and skills:

- Java – coding used for logic and wrapper
- QT GUI library – GUI design and development
- XML – XML output for the created schema