TOME Application Model

The TOME Team

October 4, 2006

Contents

List of Figures

1 Introduction

1.1 Purpose of Document

The purpose of this document is to specify the TOME system and the application model it uses.

An application is defined as having business logic and presentation elements while an individual component is defined as providing a particular set of services or presentationbut not both. This approach allows for the structure of a modern component based distributed system.

Delivery scheduling and testing issues are also dealt with to ensure that other related applications and components can be developed in parallel.

1.2 Background

In December of 2003, several students on Dorm 41 started a system called TOME. The basic idea is that at the end of the semester, instead of everyone selling their books back to the bookstore, they all donate them to central repository. Anyone on the floor can then check out whatever books they need free of charge for a semester.

The advantage of having a computer-based system to keep track of all those books is easy to see, and one has been under development ever since the start of TOME. Since its humble beginnings as a quick solution over Christmas break, the system has grown to well over 3,000 lines of Perl code as well as HTML templates, a well-planned database schema, and significant documentation. The system not only has comprehensive facilities for tracking books and patrons, but also keeps tabs on what books are used for what classes and other alternatives to purchasing new books.

1.3 References

All project data will be stored in a combination Subversion repository and Trac environment. All of this will be made viewable at the following URL:

`http://enosh.letnet.net/trac/tome`

2 Application Overview

2.1 Scope

The TOME system is responsible for managing:

- Books
- Book information
- Class information
- Class-to-book mappings
- Book reservations
- Book checkouts

The TOME system is **not** responsible for managing:

- Book acquisition
- Book disposal
- Selling old books
- Any financial activity

2.2 Context

TOME exists as an independent system with no ties to other databases. Information from other databases, such as class listings, book information, and book-to-class associations will be used, but not in an automated fashion.

2.3 Technical Environment

TOME is a web-based Perl application. It is intended to be run under the Apache webserver, but any webserver that supports CGI should be capable of running the system. Template::Toolkit is used to process HTML templates. PostgreSQL is used as the database backend. CGI::Application is used as the framework for the system as a whole.

3 Actors

3.1 Actor Diagram

Figure 1 shows an overview of the actors involved in the TOME system and how they interact with each other. Patrons talk to TOMEkeepers who use the TOME web interface which is based on the TOME database.

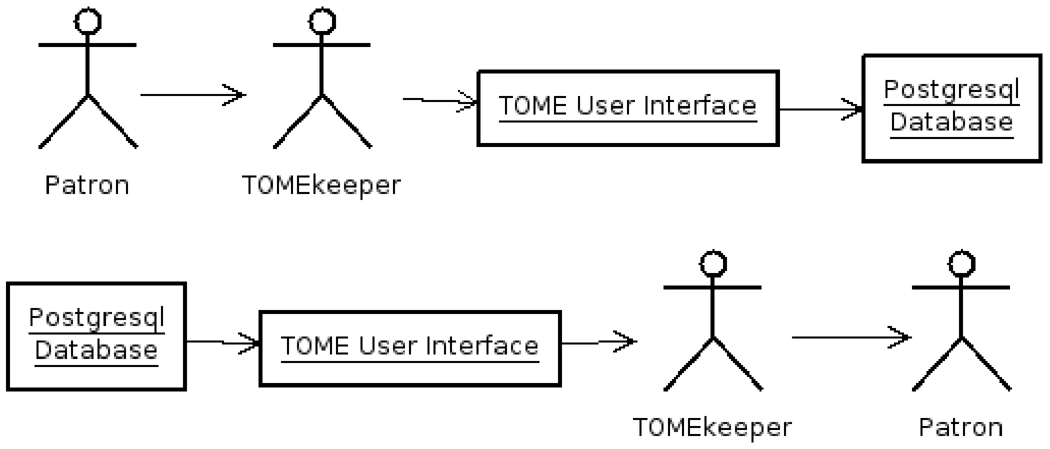

Figure 1: Actor Diagram

3.2 Actor Definitions

3.2.1 Patron

Description	The Patron is any student who wishes to use the TOME system. Only students that reside on floors with an active TOME system can become Patrons. Patrons never interact directly with TOME, only through TOMEkeepers.
Aliases	Student.
Inherits	None.
Actor Type	Passive - Person.

3.2.2 TOMEkeeper

Description	The TOMEkeeper is the primary user of the TOME system. They are responsible for all interactions with Patrons, all system administration, and all TOME activity.
Aliases	None.
Inherits	None.
Actor Type	Active - Person.

4 Business Use Cases

4.1 Use Case Listing

ID	**Use Case Name**	**Comments**
UC01	Book Added	When a book is donated to the TOME system
UC02	Book Reserved	When a patron requests that book be reserved for a particular semester
UC03	Book Checked Out	When a patron checks out a physical book
UC04	Class Added	When a TOMEkeeper adds a class to the system
UC05	Book Added To Class	When a TOMEkeeper adds a book to a particular class

4.2 Graphical Use Case Diagram

This section represents the business use cases of the TOME system in graphical form in Figure 2.

4.3 Business Use Cases

This section documents the complete business scenarios within the scope of this project.

4.3.1 UC01: Book Added

Description This scenario happens when a patron donates a book to the TOME on their floor. The book information must be entered in the system,

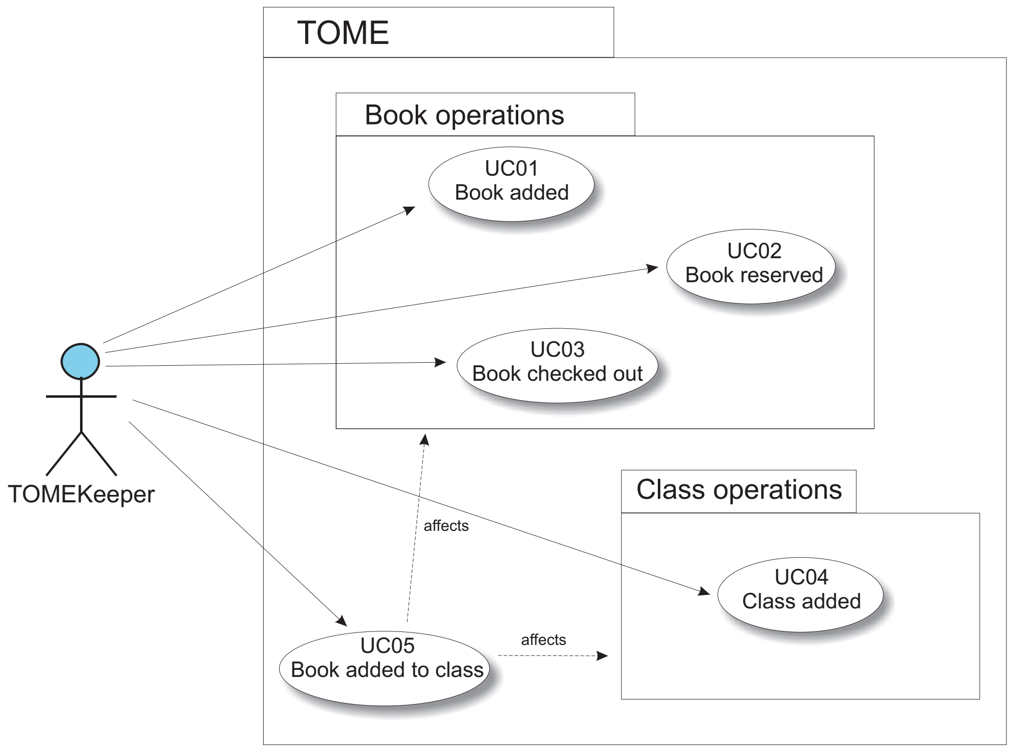

Figure 2: Use Case Diagram

and the book must be assigned a unique ID number by the system.

Actors TOMEkeeper and (indirectly) Patron

Preconditions

1. The TOMEkeeper must have an account
2. The TOMEkeeper must be logged in

Use Case Text

1. Patron requests to donate a book
2. TOMEkeeper enters the book information into the system

3. TOME assigns a book ID and presents the book information page

Alternate Courses None.

Extends None.

User Interfaces

Constraints None.

Questions None.

Notes None.

Source Documents None.

4.3.2 UC02: Book Reserved

Description This scenario happens when a patron requests that a book be reserved for them for a particular semester.

Actors TOMEkeeper and (indirectly) Patron

Preconditions

1. The TOMEkeeper must have an account
2. The TOMEkeeper must be logged in

Use Case Text

1. Patron sends their class list to the TOMEkeeper
2. TOMEkeeper enters the class information into the system
3. TOME determines what book reservations can be made, makes them, and notifies the patron via email

Alternate Courses None.

Extends None.

User Interfaces

Constraints None.

Questions None.

Notes None.

Source Documents None.

4.3.3 UC03: Book Checked Out

Description This scenario happens when a patron requests that a book be checked out for a semester

Actors TOMEkeeper and (indirectly) Patron

Preconditions

1. The TOMEkeeper must have an account
2. The TOMEkeeper must be logged in

Use Case Text

1. Patron requests that the book be checked out. This may have already been done in the form of a reservation, or the patron can make the request without a previous reservation
2. The TOMEkeeper checks a specific instance of the book out to the patron, and the system logs that checkout

Alternate Courses None.

Extends None.

User Interfaces

Constraints None.

Questions None.

Notes None.

Source Documents None.

4.3.4 UC04: Class Added

Description This scenario happens when a TOMEkeeper adds a class to the system

Actors TOMEkeeper

Preconditions

1. The TOMEkeeper must have an account
2. The TOMEkeeper must be logged in

Use Case Text

1. The TOMEkeeper finds a class that needs to be added to the system
2. The TOMEkeeper adds the class to the system

Alternate Courses None.

Extends None.

User Interfaces

Constraints None.

Questions None.

Notes None.

Source Documents None.

4.3.5 UC05: Book Added To Class

Description This scenario happens when a TOMEkeeper adds a book to a class that is already listed in the system.

Actors TOMEkeeper

Preconditions

1. The TOMEkeeper must have an account
2. The TOMEkeeper must be logged in
3. The class to be added to must already exist within the system

Use Case Text

1. The TOMEkeeper finds a book that needs to be added to a particular class
2. The TOMEkeeper navigates to the class information page
3. The TOMEkeeper adds the book to the class

Alternate Courses None.

Extends None.

User Interfaces

Constraints None.

Questions None.

Notes None.

Source Documents None.

5 Business Domain Model

The business domain model provides a listing of the modules contained in the project and how they work together. This framework gives the project team a high-level guide to use in developing the project.

5.1 Business Class Diagram

Figure 3 shows a graphical representation of the business classes involved in the TOME system.

5.2 Business Object Definitions

5.2.1 TOME

Description	TOME is the core module that processes database requests from the Web Interface.
Attributes	TOME contains all the methods used for interacting with the database as well as processing template files.
Responsibilities	TOME is responsible for all database communication, all template processing, all error handling, and any other "core" functionality
Business Rules	TOME has dependencies on many Perl modules (listed in the install document that can be located by referencing Section 1.3). These must be installed before TOME will function properly.

Figure 4 is an example sequence of events involving TOME.

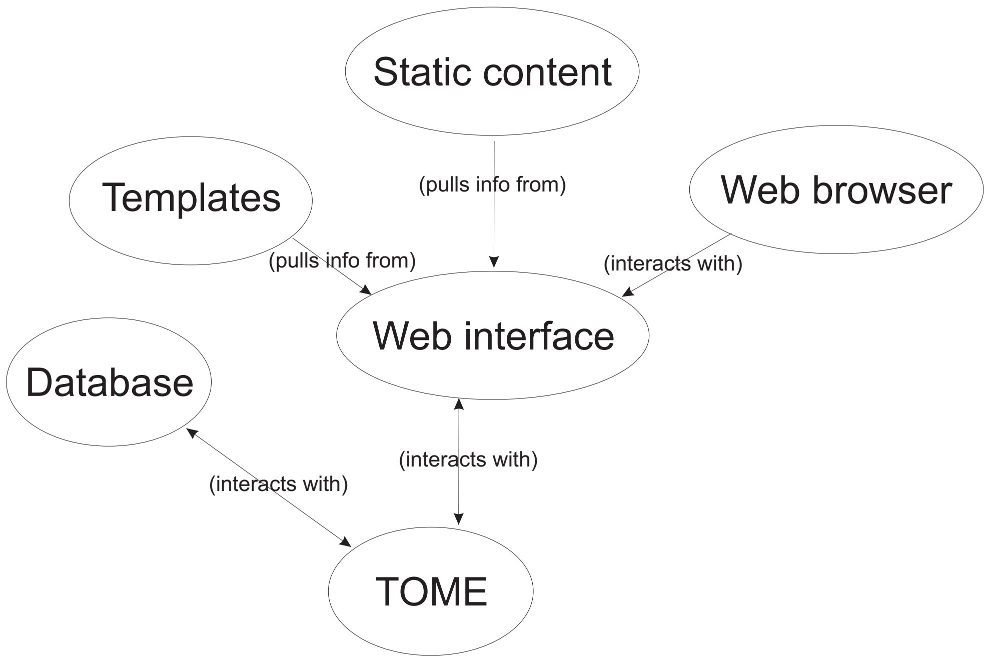

Figure 3: Business Class Diagram

5.2.2 Database

Description	The database is where all the information for TOME is stored. The current implementation uses PostgreSQL for the database and DBI for all database interactions.
Attributes	All TOME data is stored in the database. See Section 6 for more details. TOME communicates with the database to retrieve, insert, and update all data for the system.
Responsibilities	The database is responsible for maintaining a consistent state for all data at all times.
Business Rules	PostgreSQL must be installed before the system will be functional.

Figure 5 is an example sequence of events involving the database.

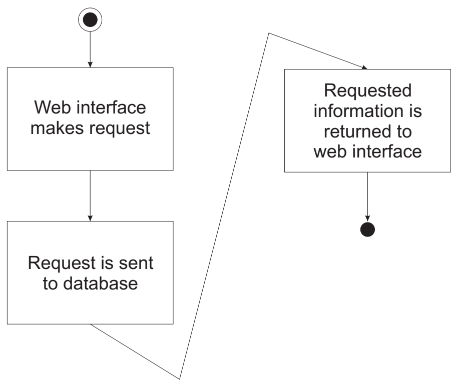
Web interface makes request
Request is sent to database
Requested information is returned to web interface

Figure 4: TOME Sequence Diagram

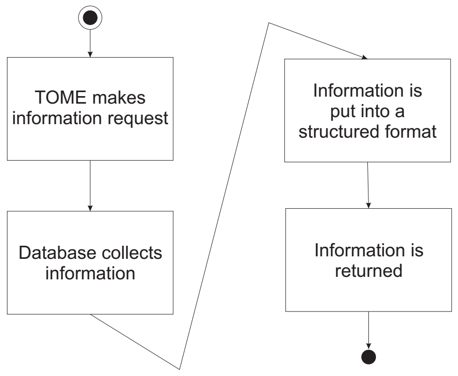
TOME makes information request
Database collects information
Information is put into a structured format
Information is returned

Figure 5: Database Sequence Diagram

5.2.3 Web Interface

Description	The Web Interface forms the connection between the Database and the Templates. It processes information that TOME retrieves from the Database and sends it to the templates. It also processes user requests and sends them to TOME.
Attributes	The Web Interface contains all methods related to interacting with the user via CGI.
Responsibilities	The Web Interface is responsible for taking any data the user submits, verifying its validity, and submitting it to TOME to be run through the Database. It is also responsible for taking information that TOME retrieved from the Database and preparing it for the templates to use.
Business Rules	The Web Interface depends on a number of Perl modules (see the Install document referenced in Section 1.3) and these must be installed before it will be functional.

Figure 6 is an example sequence of events involving the Web Interface.

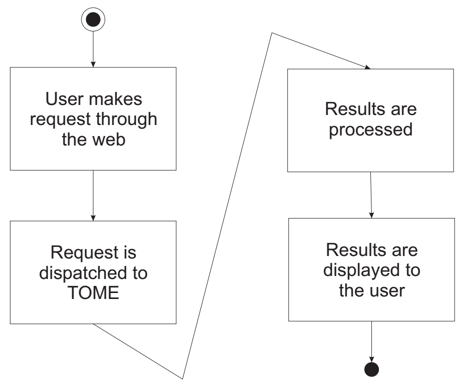
User makes request through the web
Request is dispatched to TOME
Results are processed
Results are displayed to the user

Figure 6: Web Interface Sequence Diagram

5.2.4 Templates

Description	Templates form the basis for the look and feel of the web interface. They do not give any functionality in and of themselves, but are a way to display information that is given by the Web Interface
Attributes	Templates have no methods, they only contain the data that determines how information is formatted and displayed.
Responsibilities	The templates must take information that is given to them by the Web Interface and format it. In the process of formatting that information, they may request more information through a callback mechanism.
Business Rules	The templates need to be XHTML and CSS compliant.

Figure 7 is an example sequence of events involving template generation.

6 Database Structure

The overall structure of the database can be seen in Figure 8. In addition to the structure, there are also triggers that ensure the consistency of the

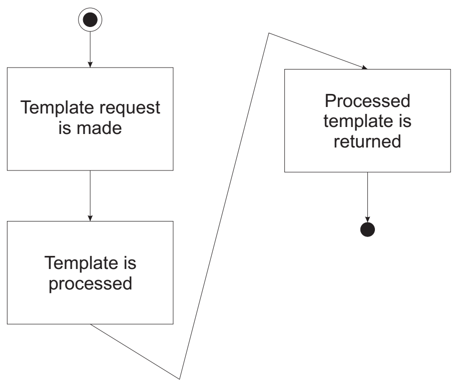

Figure 7: Template Sequence Diagram

database at all times.

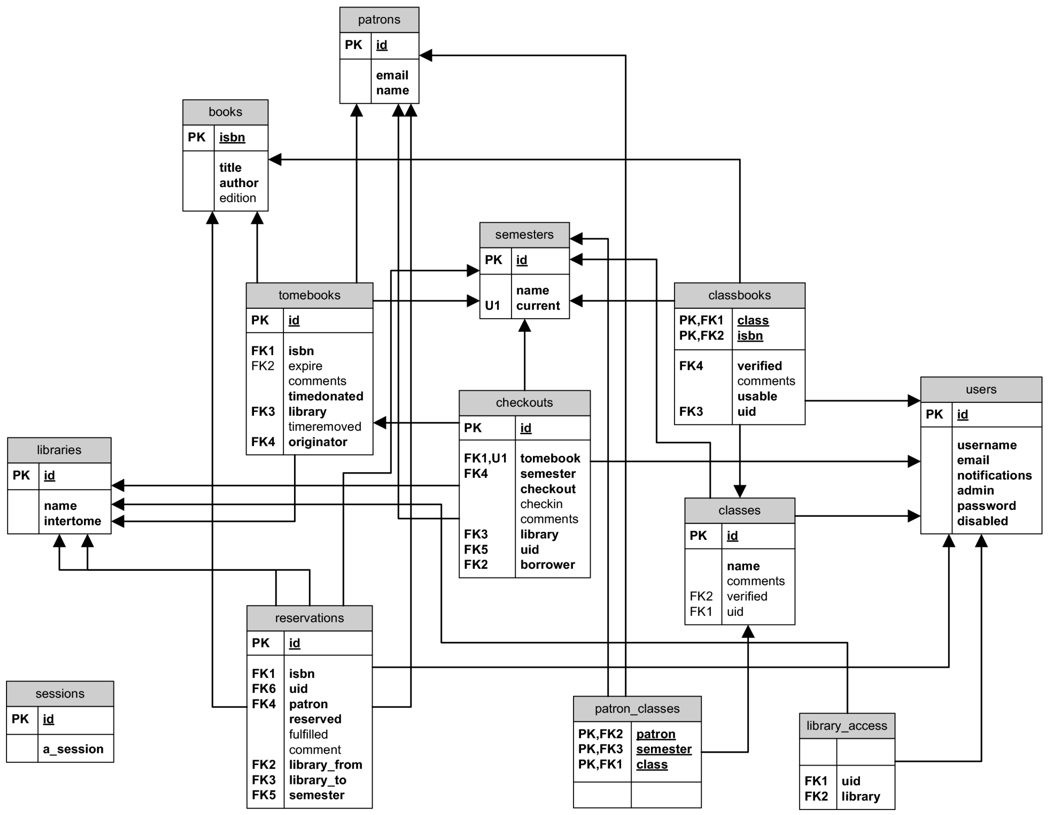

Figure 8: Database Diagram